فهرس المحتويات

الفصل الأول : مقدمة حول قواعد البيانات وإستخدامتها(Databases).

الفصل الأول : مقدمة حول قواعد البيانات وإستخدامتها(Databases).

ما هى قواعد البيانات ؟

تعرف قواعد البيانات بأنها مجموعة من البيانات المهيكلة **structured** أى موضوعة وفقا لمنظومة معينة ،فالغرض الأساسى لأى قاعدة بيانات هو تنظيم معلومات كبيرة الحجم تيسيرا على المستخدم حال قيامه بعملية استعلام أو تعديل أو إضافة لهذه المعلومات.

ما هو نظام إدارة قواعد البيانات ؟ Data Base Management System (DBMS)

هو عبارة عن برنامج لإدارة قواعد البيانات وإنشائها والتعديل فيها ، أى هو أداة المستخدم فى فعل ما يشاء فى أى بيانات على حسب إمكانيات البرنامج ، فمثلا يُمكن هذا البرنامج المستخدم من إجراء الاستعلامات)ستعرفها ما معنى الاستعلام لاحقاً–)الخاصة بإرجاع البيانات وعرضها فى جداول ،أو التعديل عليها ....إلخ من العمليات.

ما هو الـ Relational DBMS؟

يستطيع الـ **DBMS**من التعامل مع البيانات فى صورة جداول)صفوف أو سجلات –أعمدة أو حقول) تشبه تلكم الجداول فى البرامج المحاسبية مثل **Excel**وهى صورة سهلة ومنطقية لتنظيم البيانات ، ومن هنا ظهرت قواعد البيانات العلائقية أو **RDBMS** التى هى مربط الفرس الآن فى التعامل مع البيانات فى معظم التطبيقات سواءاً أكانت تطبيقات سطح المكتب أو ويب أو حتى أجهزة كفية .

متى تختار بين قواعد البيانات والجداول الممتدة Spreadsheets؟

يأتى سؤال بديهى..بما أن قواعد البيانات تشبه فى عملها الجداول الممتدة **Spreadsheets**فلماذا إذا ظهرت ؟!! ....سؤال جيد ، والإجابة تكمن فى المرونة التى يوفرها لك أى نظام إدارة لقواعد البيانات من استرجاع سلس وسهل

للبيانات ، إجراء العمليات على هذه البيانات مهما كانت مفرقة فى الجداول ،إجراء عمليات على الجداول دفعة واحدة دون تجزئتها)كل هذا
سيظهر لك لاحقا لا تقلق من المصطلحات إن لم تكن تعرفها.)

لماذا نستخدم قواعد البيانات ؟

توفر لى قواعد البيانات الحلول للعديد من المشكلات التى تقابلنى فى الحياة مثال :

•**التوثيق للبيانات الكبيرة الحجم والمبعثرة** (كالإرشيف مثلا فى المؤسسات المختلفة) بدلا من الطرق التقليدية فى الحفظ كالورق والملفات .

•**السرعة** فى جلب المعلومة مهما كان حجم البيانات عندى ،أسرع بكثير حتى ولو كان النظام الورقى التقليدى عندى مرتب بأى شكل كان .

•**توفير الوقت والمجهود** المبذولان فى ترتيب وتنظيم البيانات بالطرق التقليدية والتى بدورها معرضة لأخطاء كثيرة وجسيمة .

•**الإعتمادية فى جلب المعلومة** ،فالبشر يخطئ نتيجة للضغوط اليومية فى العمل ،وبالتالى وقت الازمات تظهرلحاجة إلى معلومات وبأقصى سرعة ،بطبيعة الحال لن تجد اسرع من البيانات المميكنة فى صورة قواعد بيانات وفاءاً بطلبك فى زمن صارت فيه للثانية قيمة .

- **فوائد استخدام نظم إدارة قواعد البيانات العلائقية RDBMS؟**

تستطيع **RDBMS** Uمن تحقيق فوائد جمة عن Uطريق التحكم فى الآتى :

•**التكرارية Redundancy** :فهى تمنع أى تكرار سواءا مقصود او غير مقصود لبيانات موجودة مسبقا وبالتالى تتحكم فى توفير وعدم إهدار المساحة على القرص الصلب او أى وحدة تخزين اخرى.

• **تضارب البيانات Inconsistency** :من الفائدة السابقة نحقق ضمان عدم تضارب البيانات ودقتها فمثال ،لو أنك أدخلت اسم نفس الشخص الرباعى مرتين مثلا فهناك احتمال للخطأ فى أحدهما مع أنه نفس الشخص مما يوحى بوجود

شخصين مختلفين ،فالتحكم فى تكرار البيانات وادخالها يلغى هذه المشكلة من الأساس.

•**تكامل البيانات Data Integrity**:فالنظام يحقق نوع من الانسجام بين البيانات يمكن من خلاله استخراج معلومة صحيحة (سيتم مناقشة قضية تكامل البيانات تلك لاحقا).

•**تدارك الخطأ** :فى حالات فشل اتمام أى عملية كتحويل الأموال مثلا أو الولوج إلى أنظمة سرية مثلا أو حتى العمليات المزدوجة كإجراء تعديل من قبل شخصين على نفس البيانات فى نفس الوقت هناك خط دفاعى لتدارك هذه المشكلات عن طريقRDBMS.

• **تأمين البيانات** :ليس كل شخص يتعامل مع البيانات مخول له القيام بعمليات كاملة على البيانات ،فهناك أشخاص لعرض البيانات فقط ،وآخرون لإجراء تعديل عليها ،وآخرون لديهم كافة الصلاحيات من حذف لهذه البيانات والتعديل.....إلخ من العمليات .

•**العمليات المنتظمة واستراجع الأخطاء** :فى بعض الحالات يكون هناك تسلسل لعمليات مختلفة على قواعد البيانات يستطيع **RDBMS** من هذا ،فضلا عن استرجاعه لهذه العمليات حال حدوث خطأ مفاجئ .

•**تنظيم التخزين** :يمكنك **RDBMS** من تنظيم للبيانات المخزنة على وحدات التخزين المختلفة بميكانيكية تسهل عمليات الاسترجاع والبحث عن طريق مايسمى بـ **Internal Schema**دون تدخل منك.

- **الفرق بين RDBMS لأنظمة سطح المكتب و الخوادم Servers**

تنقسم صناعة البرمجيات اليوم المتعلقة بقواعد البيانات تقريبا إلى قسمين رئيسيين:

- قواعد بيانات أنظمة سطح المكتب اوPCs .
- قواعد بيانات أنظمة الخوادمServers.

فيما يلى الفروق الجوهرية بين النوعين:

قواعد بيانات أنظمة سطح المكتب اوPC

تتسم قواعد بيانات أنظمة سطح المكتب او PCsبقلة عدد المستخدمين او بالأحرى المنتفعين من قاعدة البيانات عن طريق أنظمة رخيصة نسبيا مثل الـ **MS)** **Access –Lotus-FoxPro-SQL Server Express**)وفى الغالب لا تتضمن هذه الانظمة العلميات المعقدة او الإمكانات الكبيرة نظراً لعدم الاحتياج إليها على أنظمة سطح المكتب فهى تلائم المؤسسات الصغيرة والمتوسطة الحجم بمنتهى الفاعلية ، لكنها تختلف عن أنظمة الخوادم فى الآتى :

•**أرخص كثيرا :** فبالقليل من المال تستطيع اقتناء رخصة لحزمة كاملة كالأوفيس متضمنة برنامج الأكسس وبالتالى أنت والمستخدم النهائى فى غنى عن المساءلة القانونية عن رخصة نظام قواعد البيانات الملحق ببرامجك وهذا الأمر هامجدا لمن عمل فى تصنيع برامج قواعد البيانات فى السوق .

•**سهولة الاستخدام :** فقط أنت تخطط لبرنامجك ثم ما عليك إلا أن تفتح برنامج ذو واجهة رسومية **GUI** سهلة كالأكسس مثلا وتبدأ فى التنفيذ ، دون الحاجة إلى إجراء استعلامات **SQL**أو أى طرق أخرى صعبة او معقدة .

- **قواعد بيانات أنظمة الخوادم Servers:**

على العكس من أنظمة قواعد البيانات الخاصة بسطح المكتب ، تتسم قواعد بيانات الـ Serversبالتعامل مع كم كبير ومعقد من البيانات دفعة واحدة ،ليس هذا فحسب بل ومن أكثر من مستخدم فى نفس الوقت دون أى خلل وهذا يرجع إلى طبيعة الخوادم وإماكانياتها الكبيرة مقارنة بالأجهزة المنزلية العادية وكمثال على قواعد بيانات الخوادم هناك العملاق **أوراكل** وكذا **MS SQL** ، وأنظمة شركة **IBM** **Sybase DB2**.

- ويكمن الإختلاف الجوهرى بين هذه الانظمة وسابقتها فى الآتى:

•**المرونة :** وحقيقة هى من أهم ميزاتها ،فقد تم تصميم هذه الأنظمة لتلائم وبمنتهى المرونة أنظمة التشغيل المختلفة **كالويندوز واللينكس واليونكس** ولتتلقى العديد من الاستعلامات فى ذات الوقت وتتعامل معها بمنتهى السهولة والسرعة أيضا.

•**الإعتمادية :** توفر قواعد بيانات أنظمة الخوادم Serversالقدرة على الاعتماد عليها بدرجة ٢٤ساعة طيلة ٧أيام متصلة طبقا لحاجة السوق إلى ذلك ..مثال هذا أنظمة البنوك والشركات العملاقة **كميكروسوفت** وكموقع كبير **كأمازون** مثلاً فهى عادة ما تلحق ببعض الميزات مثل الـ Mirroringوالـ Log Shipping .

•**سرعة الأداء :** لأنها تعمل على اجهزة الخوادم فهى تتسم بالسرعة العالية فى الاستجابة للأوامر والعمليات المختلفة ، فأجهزة الخادم دائما ما تلحق بحجم كبير من الذاكرة وسعات التخزين ، مما لاشك فيه يؤثر بالإيجاب على السرعة والدقة المطلوبة.

• **التمد د :** من الميزات المهمة جدا ميزة التمدد والاستعداد الدائم للطوارئ والزيادات فى أى وقت فى حجم البيانات وكثافتها ، نخيل مثلا لو أن بنك فى اليابان كان حجم تعاملاته اليومية ٢٠مليون عملية تم دمجه مع بنك آخر حجم تعاملاته ١٠مليون عملية ، إن لم تكن قواعد البيانات مهيأة تماما لمثل هذه الإجراءات الطارئه ...فالخسائر ستكون فادحة فى العملاء.

- دورة حياة نظم قواعد البيانات DB Life Cycle

كما فى عالم النبات والحيوان دورة حياة ، أيضا فى عالم البرمجيات دورة حياة للمشاريع ابتداءا من التصور وانتهاءا بمراحل كالتوزيع واصدار الترقيعات ....إلخ كذا فى حالة نظم قواعد البيانات يبدأ التطوير بالفكرة ثم التنفيذ ،والذى بدوره ينقسم إلى عدة مراحل ، لايتم الانتقال إلى مرحلة إلا بعد تجاوز المرحلة السابقة لها **.((Block by Block**

قبل الشروع فى تصميم اى نظام ،لابد وأنك تعمل وفقا لنموذج قياسى معين ،

Model والذى بدوره يحوى كل الخطوات اللازمة لبدأ تنفيذ فكرتك البرمجية وبالتالى فلن يواجه فريق التطوير أى مشاكل تعترضه من تداخل فى الأفكار أو العشوائية فى التنفيذ وضمان جودة برمجية عالية.

- تنقسم دورة حياة نظم قواعد البيانات إلى عدة مراحل ،بدءاً من الـ global schemaوانتهاءا بالتنفيذ والصيانة maintenance :

•**تحليل المتطلبات Requirement analysis:**قبل الشروع فى التصميم لابد وان أعى المشكلة المراد حلها بقواعد البيانات جيدا ،يتطلب هذا عدة لقاءات مع المستخدمين أو الموظفين من خلالها يُعرف كيف يدار النظام ، ومن أين وإلى أين تتدفق البيانات بهذه الطريقة تضمن توافق تنفيذك لمشروعك مع متطلبات العميل)هذه المرحلة بحق هى عصب أى مشروع برمجى).

•**التصميم المنطقى Logical design:**ياتى بعد مرحلة جمع المتطلبات ،مرحلة تصميم كروكى لما ستكون عليه البيانات ،فابستخدام العلاقات والنماذج مثل ER diagrams نستطيع توضيح هذه العلاقات والترابطات بين البيانات.

•**التصميم الحقيقى Physical design:**متى تم الانتهاء من التصميم المنطقى ،تأتى هذه المرحلة الهامة وهى وضع الجداول وإختيار المفهرسات **Indexers** لإكمال البنية الهيكلية لقاعدة البيانات.

•**مرحلة بناء قاعدة البيانات :** هنا يبدأ المجهود السابق يثمر عن قاعدة البيانات الحقيقة التى ستستخدم فى مشروعك مستخدمين نظام إدارة قواعد البيانات العلائقية **RDBMS (فى الحقيقة سنستخدم مايعرف بـ DDLوهى إختصار لـ Data Definition Language ستعرف هذا لاحقا .**

•**مرحلة التعديل على البيانات Data modification :**باستخدام لغة التعديل على البيانات **Data Modification Language**او **DML**تستطيع إجراء الإستعلامات وإنشاء المفهرسات وتحديث قاعدة البيانات ووضع القيود مثل **التكامل المرجعىReferential Integrity .**

• **مراقبة قاعدة البيانات Database Monitoring:**وتعتبر هذه المرحلة هامة جدا بعد عمليات التنفيذ السابقة ،ضماناً لتلاقى التنفيذ مع المتطلبات المنشودة ،ففى حال وجود مشاكل أو ظهور خطأ ما فى التصميم عن طريق المراقبة تستطيع تلافى هذا الخطأ بالرجوع إلى الخطوات السابقة وإجراء التعديل اللازم .وهكذا دواليك تستمر دورة حياة قاعدة البيانات بالوصول إلى هذه المرحلة ثم العودة ثانية إلى المراحل السابقة إذا لزم الأمر.

- **وصف كيفية إعداد قواعد البيانات وكذلك نظم إدارة قواعد البيانات:**

(Ddatabase Management Systems DBMS) ومن المهم خلال قراءة لهذه المحاضرة الأخذ في الاعتبار أن غالبية المحاضرة تعالج قضية قواعد البيانات الخاصة بالبيانات الوصفية attribute data وليس البيانات الهندسية geometric data.

- يمكننا تعريف قواعد البيانات كمجموعة مركبة (مهيكلة) structured collection من البيانات التي يمكن الدخول عليها (accessible) بطريقة منتظمة uniform way مما يجعلها عنصرا هاما في نظم المعلومات الجغرافية حيث يتم تنظيم البيانات باستخدام أنواع مختلفة من نظم إدارة قواعد البيانات DBMS .
- **هناك نوعين من نظم إدارة قواعد البيانات هما:**

١ -**Hybrid systems** ويتم فيها تخزين البيانات الهندسية (الإحداثيات) في قاعدة بيانات منفصلة عن قاعدة البيانات الوصفية وهي النوع الأكثر شيوعا في تطبيقات نظم المعلومات الجغرافية ويتم فيها ربط القاعدتين من خلال رقم منفرد (unique Id-number) يربط بين الأشكال الهندسية وبياناتها الوصفية وبعيدا عن هذا النوع من الربط الداخلي Internal linking فإنه من الممكن ربط هذا الرقم المنفرد مع قواعد بيانات خارجية من خلال

الشبكات حيث قد يتم الربط في شبكة داخلية صغيرة Intranet أو شبكة المعلومات الدولية Internet.

٢ -**Integrated systems** ويتم فيها تخزين البيانات الهندسية والوصفية في نفس قاعدة البيانات ويمكن الربط مع مصادر البيانات الخارجية من خلال وسيط نظم إدارة قواعد البيانات RDBMS interface.

الاحتياج لقواعد البيانات فى نظم المعلومات الجغرافية

- عادة ما تحتوي قواعد بيانات نظم المعلومات الجغرافية على كميات ضخمة من البيانات (سواء إحداثيات أو وصفية) ومن المهم جدا هيكلة هذه البيانات (والتي إذا لم يتم هيكلتها بطريقة رشيدة فإن سرعة رد النظام على العمليات المختلفة ستكون طويلة جدا) حيث يتم ذلك من خلال نظم إدارة قواعد البيانات DBMS والتي تساعد أيضا في الرد على الاستفسارات queries من خلال اختيار انواع معينة من البيانات أو أجزاء معينة من قاعدة البيانات.
- من المهم قبل إنشاء قاعدة البيانات أن يكون لدينا conceptual model لكيفية تنظيم البيانات وكيفية هيكلتها وتسمى عملية إنشاء هذا النموذج بنمذجة قاعدة البيانات database modelling وتأخذ عملية نمذجة قاعدة البيانات في اعتبارها كل المشاركين وكذا تغطي النواحي الفنية والغير فنية ومنها:

١ -التغير السريع للتكنولوجيا : حيث لابد أن تكون الطرق التقنية مستقرة (من ناحية المعدات HW والبرامج SW) مع مرور الزمن لكي لا يكون من الضروري أن يتغير هيكل قاعدة البيانات مع أي تغير سريع في الطرق التقنية الخاصة بالمعدات والأجهزة (الهيكل ثابت مع التكنولوجيا المتغيرة).

٢ -قاعدة البيانات الجغرافية غالبا ما تكون طويلة العمر وبالتالي ينبغي التخطيط لها على هذا الأساس (أن تعيش فترة طويلة من الزمن).

٣ -هيكل قاعدة البيانات database structure ينبغي أن أن يكون بسيطا قدر الإمكان ليسهل من خلاله تغييرات وإدخال واستخراج البيانات ومن هنا فليس من الضروري التفكير في حلول معقدة عند حل المشاكل البسيطة.

٤ -يجب عزل المستخدمين users عن التغييرات التي تتم في نظام قاعدة البيانات وبالتالي فإن التغيرات التي تحدث في التكنولوجيا (ظهور برامج ومعدات جديدة) لا تؤثر في عمليتي التعامل مع البيانات (تعديل / إدخال / استخراج...) أو الاتصال بين قاعدة البيانات وواجهة المستخدم.

نموذج علاقات الكيانات ونظم إدارة قواعد البيانات Entity relationship (ER) model and DBMS

- تعتمد نمذجة قواعد البيانات على عدة مفاهيم وأكثر هذه المفاهيم استخداما هو نموذج علاقات الكيانات Entity relationship (ER) model وهو يعبر عن العلاقات بين الأشياء المطلوب عمل قاعدة بيانات لها أو بمعنى أخر يعبر عن هيكل قاعدة البيانات ،وهو يتكون من ثلاثة عناصر تمثل رموز للأجزاء المختلفة التي تكون قاعدة البيانات:

١ -نوع الكيان Entity type ويتحدد من خلاله نوع الشيئ object الذي نتعامل معه.

٢ -نو البيان الوصفي Attribute type وهو يصف أنواع الكيانات

٣ -نوع العلاقة relationship type وهو يحدد العلاقة بين الكيانات وبياناتها الوصفية.

- بالرغم من أن عملية نمذجة علاقات الكيانات قد تبدو جزءا سهلا من عملية نمذجة قواعد البيانات database modelling إلا أنها مجال هام جدا في عملية إنشاء قاعدة البيانات ويوضح الشكل رقم 8 مثالا صغيرا لنموذج علاقات الكيانات في قاعدة بيانات خاصة بإدارة شبكة الطرق حيث يظهر فيه عدد قليل

من الكيانات التي يتم ربطها ببعضها من خلال بيانات وصفية معينة specific attributes ونلاحظ بالمثال أن كيان الطريق مرتبط ببيان وصفي رقم الطريق road number ومرتبط بكيان آخر هو رابط الطريق road link ويلاحظ أن درجة تعقيد النموذج تعتمد على مدى تعقيد قاعدة البيانات ، ومن خلال هذا النوع من نمذجة قواعد البيانات يمكن توضيح مدى القوة والضعف في طرق إنشاء قاعدة البيانات كما يمكن تحليلها من خلال الرسم.

- بعد الانتهاء من إعداد ال conceptual model يتم تنفيذ هيكل قاعدة البيانات داخل نظام إدارة قواعد البيانات (البرنامج نفسه SW) وأثناء عمل ذلك لابد من أخذ الآتي في الاعتبار:

١ -مرونة عملية الإنشاء construction بدرجة كافية حتى يتم أخذ العمليات المختلفة في نظام إدارة قواعد البيانات في الاعتبار.

٢ -هيكلة البيانات بما يسهل عملية استخراج (اشتقاق) البيانات.

٣ -يجب أن تراعي في عملية إنشاء قاعدة البيانات تقليل مخاطر الأخطاء داخل النظام فلابد ألا تعطى السماحية للمستخدم أن يدخل نوع خطأ من البيانات في جزء خطأ من النظام كمثلا إدخال بيان نصي في مكان يقبل فقط البيانات الرقمية.

٤ -يجب تسهيل الدخول على قاعدة البيانات والتعامل معها من خلال إمكانيات البحث الموجودة في نظام إدارة قواعد البيانات وهذا قد يشتمل على إنشاء واجهات interfaces للمستخدمين الذين ليس لديهم مهارات في إدارة قواعد البيانات مما يصعب عليهم إستخراج واشتقاق البيانات.

- من المهم جدا وجود تعريف لفظي موحد standardized semantic للتأكد من إمكانية التواصل بين المستخدمين وكذا التأكد من ان الجميع يتحدث عن نفس الشيء فمعنى الطريق لسلئق الحافلة أنه مكان يمكنه فيه قيادة الحافلة أما الطفل فقد يفكر في الطريق كمكان للعب كرة القدم بينما يفكر أشخاص آخرون في

الطريق بشكل آخر ومن المهم جدا وجود مصطلح فني موحد (بإجماع الكل على تعريف محدد) لكل كيان في قاعدة البيانات التي نعمل عليها لتوفير إمكانية التواصل بين مستخدمي هذه القاعدة.

أنواع مختلفة من النماذج

- هناك أنواع مختلفة من هياكل قواعد البيانات database structures الشائع استخدامها وهي:

١ -قواعد البيانات المتدرجة (هرميا)hierarchical databases

٢ -قواعد البيانات الشبكية network databases

٣ -قواعد البيانات المتصلة relational databases

أولا: قواعد البيانات المتدرجة hierarchical databases:

هيكل هذا النوع من قواعد البيانات يخلق شجرة بروابط بسيطة تمام تربط بين المستويات المختلفة ويسمح الهيكل بربط كيان واحد فقط من الأسفل بكيان واحد فقط من الأعلى ويوضح الشكل مثالا لهذا النوع من خلال قاعدة بيانات متدرجة لمكتبة بها مختلف الموضوعات والمؤلفين والكتب ومن الواضح أنه يمكننا فقط ربط كتاب واحد بمؤلف واحد ومؤلف واحد بموضوع واحد مما يجعل قاعدة البيانات غير كفء حيث يمكن أن يتم تأليف الكتاب من خلال أكثر من مؤلف كما يمكن أن يؤلف المؤلف عدة كتب في موضوعات مختلفة مما يجعل هذا النوع من قواعد البيانات غير شائع الاستخدام.

ثانيا: قواعد البيانات الشبكية network databases:

هيكل هذا النوع أكثر تعقيدا من سابقه حيث يمكن فيه ربط الكيانات المختلفة ببعضها بطريقة أكثر مرونة فالكتاب في قاعدة بيانات المكتبة يمكن ربطه بالعديد من المؤلفين ويمكن ربط المؤلفين بالعديد من الموضوعات وهكذا.

ثالثا: قواعد البيانات المتصلة relational databases:

هيكل هذا النوع أكثر مرونة من النوعين السابقين حيث يتم فيه تخزين الكيانات في جداول و يتم توصيف وربط (link/relate) الكيان (الجدول) بكيانات (جداول) أخرى في جداول أخرى ويوضح الشكل قاعدة بيانات المكتبة والتي يتم فيها تخزين كيان الموضوع subject كجدول به العديد من الموضوعات ثم يتم ربط هذه الموضوعات بجدول كيان المؤلف author والذي به قائمة من المؤلفين وهكذا

نموذج قاعدة البيانات المتصلة relational database model:

- هيكل قاعدة البيانات المتصلة هو أكثر الهياكل المستخدمة شيوعا في برامج نظم المعلومات الجغرافية .
- يتم تنظيم هياكل قواعد البيانات المتصلة في جداول التي يتم تنظيمها بدورها في أعمدة تحوي معلومات مختلفة وأساسيات قواعد البيانات المتصلة هي أنه إذا احتوى عمود في جدول على نفس بيانات عمود آخر في جدول آخر فإنه يمكن ربط هذين الجدولين معا والوصل بين المعلومات المخزنة بهما وهذا يسمى (حسب هياكل قواعد البيانات) العلاقة relation ويوضح الشكل ثلاثة جداول كل منها يحتوي على عمودين حيث يخزن الجدول الأول أسماء الأشخاص والمدن التي يعيشون فيها أما الثاني فيحتوي عمودا به أسماء المدن وآخر به أسماء الدول التي تقع هذه المدن داخلها وهذا التكرار للأعمدة التي تحتوي أسماء المدن يجعل من الممكن ربط الجداول المختلفة ببعضها أما الجدول الثالث فبه عمود يحتوي أسماء الدول وآخر به أسماء عواصم هذه الدول وهذا يجعل من الممكن ربط الجدول رقم 2 بالجدول رقم 3 لأن كل منهما به عمود يحتوي أسماء الدول المختلفة كما أنه يجعل ربط الجدولين 1 و 3 ممكنا لارتباط كل منهما بالجدول رقم 2 ، ويلاحظ أن استخدام المعلومات المتكررة في خلق علاقات بين الكيانات المختلفة في قواعد البيانات المتصلة هو نفس الشيء

الحادث عند استخدام عمود Id-numbers في ربط البيانات الهندسية geometric data بالبيانات الوصفية attribute data.

- يتم التعامل مع قواعد البيانات المتصلة من خلال نظم إدارة قواعد البيانات المتصلة
- Relational Database Management Systems والتي يطلق عليها اختصارا RDBMS حيث يتم تنظيم البيانات في قاعدة البيانات في أعمدة وصفوف حيث لابد من تحديد عدد الأعمدة وكذا الصفوف ويمكن تسمية الأعمدة بأسماء fields, items, or tables مما قد يحدث ارتباك في المصطلحات أحيانا أما الصفوف فيمكن تسميتها objects, post, records, or tuples.
- من المهم جدا عند خلق قاعدة بيانات جديدة تحديد هيكل بيانات الأعمدة fields المختلفة فمثلا لابد من تحديد نوع البيان المخزن وأى نوع من المعلومات ينبغي أن نسمح للمستخدم بإدخاله في العمود فمثلا إذا تم تعريف البيان كبيان نصي فإنه ينبغي أن نسمح للمستخدم بإدخال ASCII-text وتستخدم البيانات كبيانات نصية فقط بينما عند تعريفه كبيان رقمي يمكن للمستخدم تخزين أرقام وكذا استخدام العمود في الحسابات المختلفة ويلاحظ أن التعريف المحدد والحازم لكل عمود يقلل من الأخطاء الممكن حدوثها بسبب فصل أنواع البيانات data types المختلفة ومنع خلطها ببعضها (هذا فرق هام جدا بين نظم إدارة قواعد البيانات وبرامج spreadsheet مثل MS Excel التي يمكن فيها خلط أنواع البيانات المختلفة في نفس العمود) ، ومن الضروري أيضا تحديد حجم العمود بتحديد عدد الأحرف (أو الأرقام) characters المسموح بتخزينها فيه فمثلا قد يسمح بتخزين 50 حرفا في الأعمدة النصية text columns بينما في الأعمدة الرقمية فيمكن تخزين 30 وحدة رقمية digit مع 10 وحدات عشرية

decimal ، كما يلزم أيضا تحديد نوع التخزين للقيم المختلفة (في الأعمدة الرقمية) كمثلا أن يتم تخزينها على هيئة binary, integer, or real.

- يتم هيكلة البيانات في جداول يتم ربطها معا وهذا الربط يخلق علاقات بين البيانات الوصفية المختلفة وحتى إذا لم يتم ربط جدولين معا بطريقة مباشرة فمن الممكن أن يكون بينهما علاقة من خلال سلسلة من الجداول المرتبطة ببعضها مما يجعل من السهل دمج معلومات من الجدولين ولو نظرنا للارتباط بطريقة أعمق لوجدنا أن كل الأشياء وكذا الآدميين يرتبطون بطريقة أو بأخرى معا كما يوضح الشكل بأعلى.

- عند بناء الجداول في قواعد البيانات المتصلة يتم أخذ بعض المحددات في الاعتبار فمن المهم أن نخزن قيمة واحدة في كل خلية من خلايا الجدول حيث أنه ليس من الممكن مثلا تخزين عمر وعدد الناس في نفس الخلية كما لا يمكن مثلا تخزين العمر والإسم لنفس الشخص في نفس الخلية فكل خلية لابد أن تحوي قيمة منفردة unique تعبر عن صفة معينة للكيان entity المعبر عنه بالجدول.

- لابد أيضا من تلافي الاعتماد الوظيفي functional dependence ومعنى هذا أن نتلافى أن تكون قيم عمود معين بالجدول تعتمد على قيم عمود آخر سواء كان في نفس الجدول أو في جدول آخر (يتم تحويل القيم في العمود الأصلي من خلال خوارزم algorithm معين وتخزينها في عمود آخر يسمى calculated ield وهذا يرفع المساحة المستخدمة من الذاكرة وبالتالي فهو طريقة غير كفء حيث يمكن حساب هذه القيم بشكل مؤقت بدلا من تحميلها في الذاكرة وتخزينها فيها بشكل دائم).

- لابد أيضا من تلافي التكرار redundancy بقدر الإمكان عند إنشاء قاعدة البيانات وهذا التكرار يعني تخزين نفس المعلومات عدة مرات مما يجعل قاعدة البيانات أضخم وبالتالي أبطأ في استخراج البيانات منها وفي التعامل معها بشكل

عام وعادة يتم تلافي هذا التكرار بتقسيم الجداول الضخمة لعدد من الجداول الصغيرة التي يتم ربطها من خلال الأعمدة المتكررة common columns (التي تحدثنا عنها في مثال الدول والعواصم) ويتضح هذا من خلال المثال الموجود بالشكل العلوي.

- استخدام الفهارس (مثلما يحدث في دليل التليفونات) سيزيد سرعة البحث في قاعدة البيانات حيث يمكن فهرسة البيانات (على سبيل المثال) أبجديا وهذه الفهرسة ستقلل وقت استخراج واشتقاق البيانات كما يمكن استخدام الفهارس نفسها في الربط بين الجداول المختلفة ومن هنا فإن الفهارس تجعل التعامل مع قاعدة البيانات كفؤا بشكل أكبر لأنها تمنع التكرار duplication وتزيد من سرعة البحث وتحفظ سلامة المرجعية referential integrity لإمكانية استخدامها في الربط بين الجداول.
- العلاقات : Relational db

وسميت بذلك لأنها تقوم علي وجود علاقات بين الجداول وبعضها البعض ، وفيما مضي كانت db عبارة عن مجموعة جداول فقط ولهذا كانت تحتوي علي تكرارات كثيرة ولهذا فإنه كان يجب الربط بين جدولين أو أكثر للحد من التكرارات والتي تسبب العديد من المشاكل منها:

1- تكرار إدخال البيانات.

2- تأخذ مساحة كبيرة علي HD

والهدف الأساسي من Db هو تخزين البيانات والعامل الرئيسي في إي db هو الجداول" Tables" والتي تتكون من مجموعة من الحقول "Fields " ولابد من تحديد أنواع البيانات التي تدخل في الجداول , date and time " integer ,....."

- **أهمية العلاقات :**

1- عدم تكرار البيانات وبالتالي نوفر مساحة ونقلل نسبة الخطأ.

2- سرعة البحث وسهولة الوصول للبيانات.

3- حماية البيانات.

وتنقسم عمليات الربط والضم link and joins:

إيجاد العلاقات بين البيانات

تعتبر الجداول الشكل المبدئ لقواعد البيانات العلائقية ،فى الواقع يتم تخزين البيانات وعلاقاتها معا فى قاعدة البيانات فى جداول.

تتكون الجداول من صفوف وأعمدة ، كل عمود يُعبر عن معلومة جزئية .

فى قواعد البيانات لابد وأن ترتبط الحقول أو أى جزئية من البيانات مع بعضها البعض بعلاقة ما.

- تتعدد طرق ربط البيانات ومن أهما ثلاثة طرق للعلاقات:

١ -واحد لواحد one to one.

٢ -واحد لمتعدد one to many.

٣ -متعدد لمتعدد many to many.

أولا: واحد لواحد one to one:

يربط هذا النوع من العلاقات بين الجداول عن طريق قيمة منفردة تظهر مرة واحدة في كل جدول والمثال يوضح وجود عمود اسمه " order" في كل جدول ويلاحظ أن رقم الأمر order number قيمة منفردة تظهر في الجدول مرة واحدة كما يلاحظ أن العلاقة تربط صف واحد في أحد الجدولين بصف واحد في الجدول الآخر.

نموذج لعمل علاقة one to one

Code	Name	Add	Tel	Birthed
1	Ahmed			
2	Ali			
3	Hassan			
4				

Code	Brother	Sister	Child
1	1	3	2
2	3	2	-
3	2	2	1
4			

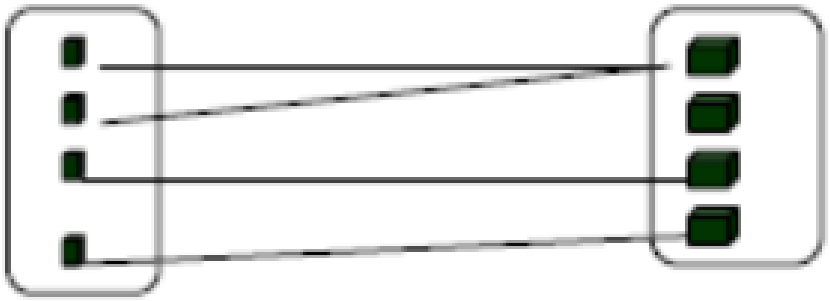
ثانيا: واحد لمتعدد one to many:

يحدث هذا النوع من العلاقات عندما يمكن ربط صف واحد في أحد الجداول بعدة صفوف في جدول آخر وهذه الطريقة هي طريقة نموذجية لتلافي التكرار redundancy في قاعدة البيانات ويوضح المثال جدولين الأول بع عمود بأسماء الدول وعمود بأرقامها أما الجدول الثاني فبه عمود بأرقام الدول وآخر بالمدن المختلفة الموجودة في هذه الدول وحيث أن الدولة بها عدة مدن فإن رقمها يمكن تطبيقه على عدة مدن داخلها (عدة صفوف من الجدول تلائم دولة واحدة) من خلال جدول أسماء المدن ، من هنا وبالنظر لقاعدة بيانات أكثر تعقيدا فإن الجدولين سيحتويا على معلومات أكثر يجب تخزينها بالطريقة الموضحة حيث سيحوي الجدول الأول معلومات (أعمدة) مثل عدد السكان والمساحة والعواصم الخاصة

بالدول الموجودة به أما الجدول الآخر فإنه معلومات عن عدد السكان والمساحة ووصلات السكك الحديد الخاصة بالمدن الموجودة به.

مما سبق فإن تقسيم الجدول باستخدام علاقة واحد لمتعدد one to many أفضل من تخزين كل شيء في نفس الجدول لأننا نكون قد تلافينا التكرار redundancy.

نموذج لعمل علاقة one to many



Code	Name	Add	Tel	Birthed
1	Ahmed			
2	Ali			
3	Hassan			
4				

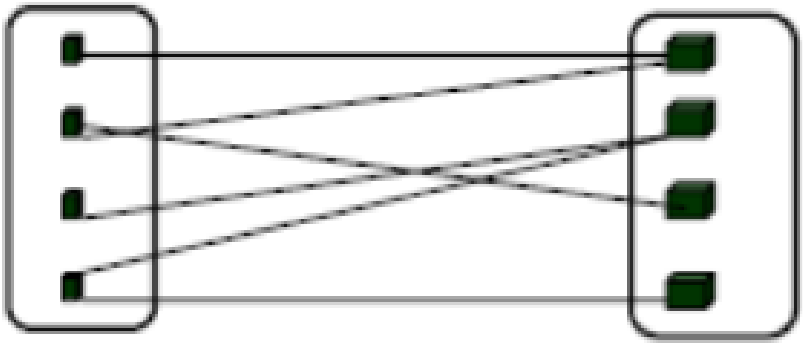
Code	Date	Gift	Price
1	1/1/2013		
2	8/4/2015		
3	5/6/2015		
4			

ملحوظة : هنا يجب أن نعطي لكل فرد رقم " التكويد PK" وذلك لأنه لا يمكن الربط بين الجداول باستخدام أي من هذه البيانات لأنها قد تتشابه والعلاقة بين هذين الجدولين " One to Many " وقد تكون معظم العلاقات في DB هي" One to Many ".

ثالثا: متعدد لمتعدد many to many:

هذا النوع من العلاقات لايوجد به أى قيم منفردة في أى من أعمدة الجداول ويوضح المثال جدولين بهما معلومات عن حالة الطقس تم اقتباسها من محطة أرصاد ويحوي كل جدول رقم المحطة وسنة الرصد وشهر الرصد ثم معدلات تساقط الأمطار في أحد الجدولين ودرجات الحرارة في الآخر وفي هين الجدولين لا يوجد مؤشرات منفردة unique identifier وللربط بين القيم المنفردة (على سبيل المثال لنفس الشهر) لابد من استخدام عدة أعمدة (رقم المحطة وسنة الرصد وشهر الرصد).

فى الغالب عند تنفيذ هذه العلاقة يكون هناك جدول ثالث للربط بين الجدولين لأن هذا النوع من العلاقات معقد نوعا ما.



خصائص العلاقة many to many

- لا تستخدم كثيراً
- وتظهر بمجرد النظر الي الجدول
- ولا يوجد لها اي متطلبات

- غالبا ما توجد لدينا مشكلة البيانات المفقودة missing data مما يسبب مشكلة عند التعامل مع البيانات الوصفية وعلاج ذلك هو ضم الجداول أو ربطها لمعرفة القيم المفقودة وعندما نريد ضم جدولين أو ربطهما فإن ذلك يمكن أن يتم من خلال إحدى طريقتين:

١ -الضم الداخلي inner join.

٢ -الضم الخارجي outer join.

أولا: الضم الداخلي inner join:

عند ضم جدولين معا باستخدام هذا الأسلوب فإنه يمكن فقط ربط link الصفوف التي يمكن ربطها من خلال علاقة واحد لواحد وبالتالي فهو يأخذ فقط الصفوف التي بها قيم في كلا الجدولين ويتم حذف كل الصفوف الباقية من قاعدة البيانات الناتجة وهذه الطريقة تستخدم لحذف البيانات التي ليس لها صلة بموضوع الدراسة عند دمج قواعد البيانات ويلاحظ في المثال أننا نفقد بيانات من الجدول الأول (1و 2) وبيانات من الجدول الثاني (6و 7) لكن تصبح بعد ذلك قاعدة البيانات ذات هيكل مدروس.

ثانيا: الضم الخارجي outer join:

عند ضم جدولين معا باستخدام هذا الأسلوب فإنه يتم الاحتفاظ بكل الصفوف من أحد الجدولين ثم إضافة الصفوف المتوافقة معها corresponding rows من الجدول الآخروفي المثال تم أخذ كل الصفوف من الجدول الأول A ثم دمجها مع الصفوف المتوافقة من الجدول الآخر B وبالتالي فإن الصفوف الموجودة في الجدول الأول وليس لها صفوف متوافقة في الجدول الثاني سيظهر فيها بيانات مفقودة في الجزء الذي مصدره الجدول الثاني B (في هذه الحالة نفقد فقط القيم 6و 7 من الجدول الثاني) ويلاحظ أنه طبقا للغرض من قاعدة البيانات يمكننا أن نحدد نوع الضم الذي نستخدمه (داخلي/ خارجي) وإذا كان للجدولين نفس الصفوف (نفس المؤشرات بما يعني أن العمود .No به نفس القيم في الجدولين) فإن مشكلة البيانات المفقودة لن تحدث.

ولإمكانية الربط لابد من توافر مفتاح Primary Key

فهم المفاتيح Keys

عرفنا أن تمثيل العلاقات يتم بين البيانات فى الجداول وفقا لطبيعة العلاقة ، ولإيجاد

هذه العلاقات لابد من وجود صفوف أو حقول مثلا للربط بين هذه الجداول ، وأمثل علاقة بين هذه الجداول هى المفاتيح ، **Keys**هذه المفاتيح ماهى إلا أعمدة متشابهة بين الجداول وذلك لتمييز الصفوف بعضها عن بعض . أى **RDBMS**يتعامل مع نوعين رئيسيين من المفاتيح

Primary Key (P.K) , Foreign Key (F. K)

١ -المفتاح الرئيسي P.K:

هو عمود او أكثر ، يتصف بأنه غير حاوى لبيانات متكررة أى بيانات فريدة **Unique**مميزا به كل صف فى الجدول .تنبه إلى الآتى عندما تختار مفتاحا رئيسيا **P.K :**

- كل سجل فى الجدول لابد Uألا يحتوى على قيمة خالية **NULL .**
- كل قيمة مدخلة إلى العمود الحاوى للمفتاح لابد أن تكون قيمة فريدة كما أسلفنا.
- ضمان الحفاظ على القيمة الموضوعة فى الجدول من عدم التغيير مستقبلا أثناء اجراء العمليات على قاعدة البيانات.
- يوجد فى الجدول مفتاح رئيسى واحد فقط.

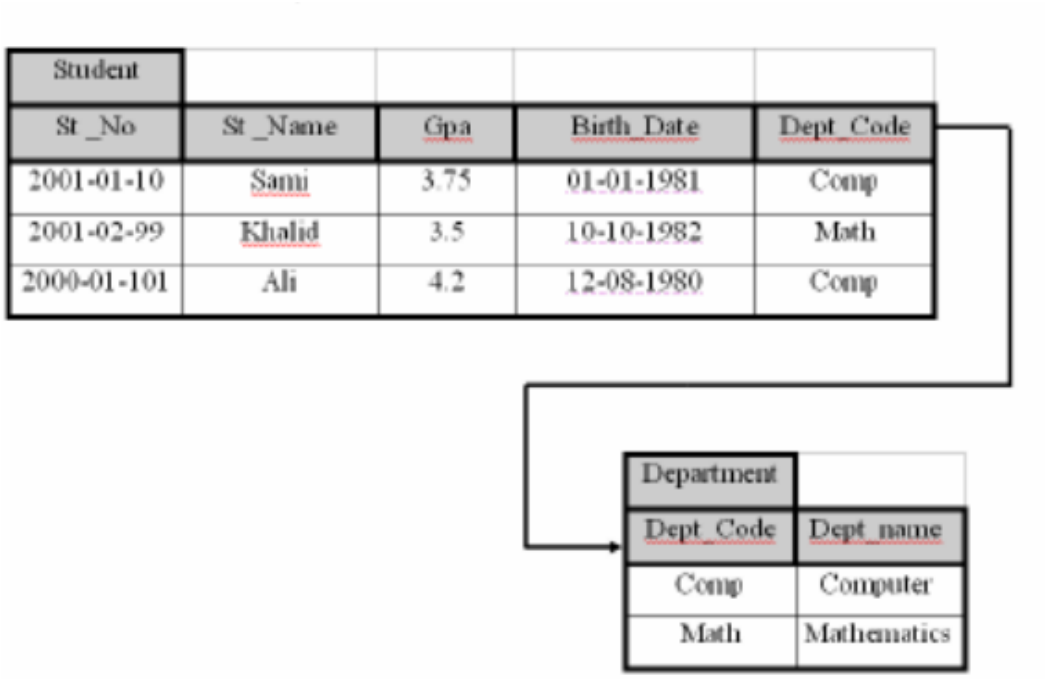
إلى جانب ضمان المفتاح الرئيسى **P.K** لفرادة البيانات الموجودة فى كل سجل ، فهو أيضا يسهل عملية البحث داخل الجدول وذلك لأنك بمجرد انشاءك للمفتاح الرئيسى يتولد اتوماتيكيا فهرس للجدول يسهل عليك عملية البحث وبالطبع هذا كله يعود إلى ال **RDBMS .**

ـ ليس هذا فحسب بل من الممكن أخذ أكثر من **Entity**وإعتبارها مفتاح رئيسى وذلك تحقيقا لمبدأ الفرادة **Uniqueness**

ويسمى هذا،كما أنه يمكن ترشيح اى مفتاح ليصبح مفتاحا رئيسيا)وفقا للصفات السابقة الذكر(ويسمى هذا **المفتاح المرشح Candidate Key** .

٢ -المفتاح الخارجىForeign Key :

المفتاج الخارجى **F.K**عبارة عن صفة مرجعية للمفاتيح الرئيسية ..أو قيمة غير مكررة **Unique**) (فى جدول آخر وهى هامة جدا فى تحقيق خاصية التكامل المرجعى والتنقل بين عناصر الجداول المترابطة ..انظر الشكل للتقريب ..ولاتنس ان الأمر مجرد تنشيط للذهن ومراجعة لما تعرفه سابقا.



Student

St _No	St _Name	Gpa	Birth_Date	Dept_Code
2001-01-10	Sami	3.75	01-01-1981	Comp
2001-02-99	Khalid	3.5	10-10-1982	Math
2000-01-101	Ali	4.2	12-08-1980	Comp

Department

Dept_Code	Dept_name
Comp	Computer
Math	Mathematics

لاحظ أن الـ **Code Dept** فى الجدول الأصغر مفتاح رئيسى ..وفى الجدول الأكبر مفتاح خارجى **F.K.**

- ومن شروط وجود P K

1- أنه لا يوجد فارغ.

2- أن لا يوجد تكرار.

ونلاحظ أنه من الناحية النظرية يجب أن يكون في كل جدول P K ، حتي يتم عمل علاقة مع جداول آخري . ولكن أنه ليس شرط أن يوجد Pk إن لم يتطلب الجدول مع عدم تطلب ربطة بجداول آخري.

-فهم تمامية البيانات Data Integrity

وتعنى ضمان صحة قيم بيانات قاعدة البيانات وكذا ارتباطها معا .وهى تنقسم إلى قسمين **Entity Integrity**و **Integrity Referentia .**

أولا: Entity Integrity

ذكرنا فى الشرط الأول من شروط اختيار المفتاح الرئيسى ألا تكون القيمة خالية **Null** وهذا لضمان وجود قيمة على طول المفتاح الرئيسى فى كافة الصفوف ، هذا الضمان هو مايعرف بـ ، **Entity Integrity** حيث يقوم الـ **DBMS** بمنع إجراء استعلامات(**Insert-Update**) إلى المفتاح الرئيسى متضمنة قيمة مكررة أو غير فريدة.

•ثانياً: Referential Integrity

عندما يتم انشاء الجداول وإيجاد العلاقات بينها فى قاعدة البيانات ،يتم اختيار المفاتيح الخارجية **F.K**والتى عن طريقها يتم إدارة العلاقات بين الجداول تحقيقا لمبدء التكامل المرجعى ، **R.I**فهو يضمن أن كل قيمة فى المفتاح الخارجى **F.K**تقابلها قيمة فى جدول آخر للمفتاح الرئيسى **P.K** .

- فهم مبادئ الـ Normalization

الـ Normalization باختصار شديد هى منع تكرار البيانات أو الزج ببيانات ليس لها داع فى أثناء قيامك بالتصميم المنطقى لقاعدة البيانات ،وبهذه الطريقة توفر على نفسك عناء التحديث المستقبلى لقاعدة البيانات ، فضلا عن سرعة الاداء بعد التنفيذ الفعلى لقاعدة البيانات .بالطبع عملية كتلك تمر بعدة مراحل إلى ان تصل الى النموذج المعيارى **3NF Third Normal form**

ويمكنك الاستزادة بقراءة .http://en.wikipedia.org/wiki/Database_normalization

- مساوئ الـ Normalization

بطبيق مبادئ الـ **Normalization**يصبح عندنا مجموعة كبيرة من الجداول وكذا العمليات الوصلية **Joins)** (لإسترجاع البيانات، وبما أن البيانات موزعة على الجداول فيما بينها ،فإن مثل هذه العلميات تكون معقدة وبالتالى ترهق عملية المعالجة وتستنزف وقتها ،فمن الممكن وقتها نتخلى عن بعض مبادئ **الـ Normalization** فى سبيل التخفيف عن المعالجة بعض الشئ ،تذكر أن الغرض من هذه العلميات ماهو إلا الوصول إلى التصميم الأمثل لقاعدة البيانات تجنبا لمشاكل الأداء والتحديث.

الفصل الثانى لغة الاستفسار المهيكل structured query language(SQL):

الفصل الثاني لغة الاستفسار المهيكل structured query language(SQL):

١. لغة الاستعلامsql:

هى عبارة عن مجموعة من الاوامر التى يحتاجها المبرمجين وكذلك المستخدمين للوصول للبيانات الموجودة ضمن قاعدة اوركل.

وتعريف :كلمة SQL هي اختصار ل ، Structured Query Languageوتعني لغة الاستعلامات المرتبة، وتستعمل من أجل إجراء عمليات على قواعد البيانات.

حتى نستوعب هذا المعنى بصفة دقيقة، فلغة SQLهي التعبير البرمجي للجبر العلائقي الذي رأيناه في مستهل الكتاب، ظهرت هذه اللغة سنة ، 1974ثم بعد ذلك في سنة 1986تم اعتمادها من طرف (،ANSI) ،وفي سنة 1987تم اعتمادها من قبل ISO لتصبح بذلك اللغة الأكثر شيوعا في أنظمة إدارة قواعد البيانات العلائقية RDBMS.

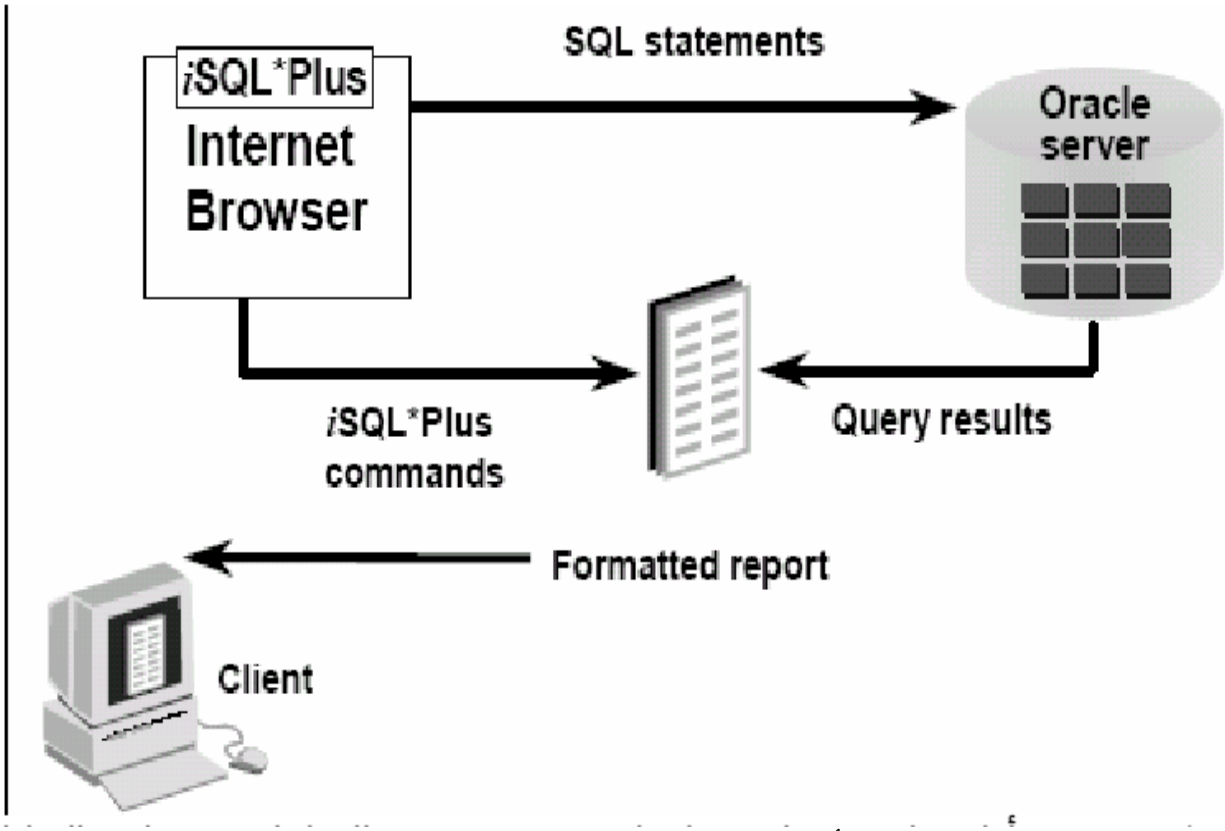
بالنسبة لبرنامج Microsoft SQL Server فهو يستعمل نسخة متطورة من SQL، تسمىTransact SQL وتكتب اختصارا T-SQLوهي تضم المهام التالية:

أولاً: مهام لغةT-SQL:

لغه SQLهى لغه قياسيه : هى لغه من اللغات القياسيه الخاصه بمعهد ANSI (American National Standards Institute) تمكنك من دخول ومعالجه نظم قواعد البيانات Database System جمل SQLتعمل مع برامج قواعد البيانات مثل :

Ms-Access, Ms-SQL Server, DB2, Oracle, etc.

الفرق بين ISQL*PLUS و SQL



SQL*PLUS هي أوامر ثابتة متعارف عليها و تستخدم في التعامل مع قواعد البياناتISQL*PlUS .هي أوامر خاصة للعمل في مكانisql*plus

الفرق بين SQL*PLUS و iSQL* PLUS

iSQL* PLUS	SQL
هي أوامر خاصة بالعمل فيisql*plus	أوامر sqlهي أوامر ثابتة متعارف عليها وتستخدم في التعامل مع قواعد البيانات بكافة اشكالها.
مكن اختصار أوامر isql*plus	لا يمكن اختصار أوامرها.
لا يذهب آخر أمر في isql*plus إلى buffer	تذهب أوامر sqlإلى جزء في الذاآرة يسمي bufferوذلك لحفظ آخر أمر لسهولة تعديله وهو جزء في الذاآرة يحفظ آخر أمر sql لسهولة الاسترجاع والتعديل
تاتى بالبيانات من على SERVERبطريقة غير مباشرة .بمعنى عدم وجود قاعد بيانات على نفس الجهاز.	تاتى بالبيانات من على SERVERمباشرة بمعنى انة لابد من وجود قواعد بيانات على نفس الجهاز.
يمكنك استخدام علامة)--(لكى يمكنك من ايقاف الكود الذى على نفس السطر	لا يمكنك من من استخدام تلك العلامة
يمكنك من تعديل الكود اذا حدث بة خطأ بسهولة	يمكنك من تعديل الكود اذا حدث بة خطأ عن طريق آتابة ED ثم اضغط ENTERفتظهر لة NOT PADيتم عمل تعديل الكود بداخلها.

ثانياً: نبذة تاريخية:

لغة الاستفسار البنيوية تمكن المستخدم من استرجاع البيانات المخزنة في قواعد البيانات العلائقية. كباقي اللغات المصممة للتعامل مع الحاسوب لغة SQL لها قواعد صارمة. صممت لغة SQL لتمكن المستخدم من التفاعل المباشر مع الحاسوب. لذا فان بنية الأوامر في اللغة تمكن من استخدامها بشكل تفاعلي من اجل ايجاد إجابات سريعة عن الاستفسارات التي يحتاجها المستخدم.

اول لغة برمجة حققت الاهداف المذكورة كانت نتاج بحث علمي في مختبرات شركة IBM وأطلقوا عليها الاسم SEQUEL وهذا اللفظ اختصار لعبارة "اللغة الإنجليزية البنيوية للاستفسارات " Structured English Query Language لكنهم تخلوا عن هذا الاسم سريعاً عندما اكتشفوا أنه علامة تجارية لشركة بريطانية تعمل في حقل الطيران واستبدلوه بالاسم الذي أصبح مستخدم حتى اليوم وهو لغة الاستفسار البنيوية SQL . قامت شركة IBM باستخدام SQL في إنتاج عدد من نظم إدارة قواعد البيانات العلائقية مثل النظام System R والنظام System/38 والنظام SQL/DS وأخيراً النظامDB2 . لكن النظام الذي حقق النجاح الأكبر هو النظام الذي اعتمدته شركة Oracle وحمل أسمها والذي طرحته لأول مرة عام 1979.

منذ ابتكار لغة الاستفسارات البنيوية في مطلع السبعينات، خضعت للعديد من التعديلات والتطورات، كما قامت العديد من الشركات والمؤسسات البحثية بتصميم نسخها الخاصة من لغة الاستفسارات البنيوية، للتعامل مع هذا الوضع قام المعهد الوطني الامريكي American National Standards Institute ANSI بإصدار أول نسخة معيارية من لغة الاستفسارات البنيوية عام 1987 وهي التي عرفت باسمSQL1987 ، ثم توالت التعديلات والطبعات المعيارية، وحالياً فإن اللغة المعيارية المعتمدة هيSQL2003 .

ويمكن فهم لغة SQL بسهولة دون ان نتعلم الجبر العلائقي الا ان الجبر العلائقي باعتباره الاساس النظري للغة SQL يسهل عملية فهم و بناء الاستفسارات المعقدة بالاضافة الى انه يسهل فهم كيف يقوم نظام قواعد البيانات بتنفيذ الاستفسارات داخلياً الذي من شانه ان يعمق فهم اللغة و يعزز من مهارات و قدرات مستخدمها.

ثالثاً: مميزاتها:

•انها قاعدة بيانات قوية مقارنة بمثيلتها مثلAccess) (و Microsoft SQL) (Server

•انها تتمتع بقدر كبير من الامان وهو السبب وراء انتشارها

• سريعة جدا فى عملية البحث من خلالها.

• و يمكن التعامل مع قاعدة البيانات من خلالها ومن خلال هذه اللغة يمكن اعطاء صلاحيات وامتيازات ممارسة عمليات معينة مثل:

1 - انشاء جداول (create table)

2 -التعديل فيها (alter)

3 -حذف جداول (drop)

4 -ملء جداول البيانات (insert)

5 -حذف البيانات المدخلة (delete)

6 -التعديل على البيانات المدخلة(update)

7 –البحث عن البيانات (query)

رابعاً: مكوناتها:

تنقسم لغة SQLالى ثلاث اقسام حيث تشكل آل مجموعة اوامر لغة فرعية من هذه اللغة وهى آالاتى:

1- اوامر تعريف البيانات data definition language (DDL):

وتحتوى على ثلاث اوامر وهى:

•امر(Create table) يستخدم لانشاء الجداول

•امر(Alter table) يستحدم للتعديل فى جدول منشا سابقا

•امر(Drop table) يستخدم فى حذف جدول غير مرغوب فيه

حيث يقتصر عمل هذه الاوامر على الجداول وحقولها فقط دون التعرض للبيانات التى بداخل الجداول

2- اوامر لغة معالجة البياناتdata manipulation language (DML)

وتحتوى على اربعة اوامر وهى:

•امر (Insert into) يستخدم فى ادخال البيانات الى الجداول

•امر (Update) يستخدم فى تعديل البيانات فى الجداول

•امر (Delete) يستخدم فى حذف البيانات من الجدول

•امر (Select) يستحدم فى الاستعلام عن شيء معين بيانات الجدول

3- اوامر التحكم بالبيانات data control language (DCL) :

عمل مقارنة بين QBE and SQL

لكى نتعامل مع قواعد البيانات العلائقية هناك نوعان من اللغات للتواصل بين المستخدم وبين قاعدة البيانات وهما .QBE- SQL)

QBE- :هى اختصار للجملة Query By Example وهى طريقة لاجراء الاستعلامات على قواعد البيانات معتمدة على الواجهه الرسومية معتمدة على مبدأ Point and Click).(لقد تم تطوير QBE على يد موشى زلووف فى معامل IBMبالتزامن مع تطوير SQLفى منتصف السبعينات ، وهى تختلف عن SQLفى سهولة اجراء الاستعلامات معتمدين كما قلت على الواجهة الرسومية فقط ارسم الجدول وقم بالاشارة على ماتريد استخلاصة من بيانات او حذف او تعديل بيانات وسيستجيب لك نظام ادارة قواعد البيانات ،وهى طريقة تعتبر مثالية فى حالة قواعد البيانات الغير معقدة والتى يتم تمثيلها بالقليل من الجداول .

وعلى الرغم من انها تطوير شركة IBM إلا أن شركات كميكروسوفت قامت باجراء بعض التعديلات عليها واستخدامها فى قواعد بيانات اكسس وتظهر جليا عند اجراء استعلامات على الـ Forms

-SQL:هى اختصار للجملة ، Structured Query Languageوهى لغة تم تطويرها فى معامل IBM ايضا من قبل مجموعة تطوير فى مركز ابحاث سان جوز -شامبرالن وريموند -وهى أساسا قد تم تطويرها للتعامل مع نظام قواعد بيانات يسمى System R وهو نظام مبنى على فرضيات ، Codd ولمن لايعرف فان Coddهذا يعتبر الأب الشرعى لقواعد البيانات العلائقية بعد نشره ورقة بحث يوضح فيها الأسس والأطر لهذا النظام الجديد .فى عام ١٩٨٦تم اعتماد لغة SQLمن قبل ANSIوأيضا من قبل ISOعام ١٩٨٧وتم نشر اللغة على وضعها القياسى هذا تحت اسم SQL1 . ومن هذه الانطلاقة و سكول آخذة فى التطوير ففى عام ١٩٨٩وعام ١٩٩٢تم عمل نقلة نوعية فى اللغة ثم مع العام ١٩٩٩واصدار SQL3التى دعمت مميزات كالأهداف الموجهه والتى كانت نواه لقواعد البيانات الموجهه بالكائناتObject Relational DB.

وعلى الرغم من وضعها SQL كمعيار لكن هناك شركات مثل اوراكل وميكروسوفت اصدرت انتاجها الخاص من SQL تيسيرا لبعض المهام على قواعد البيانات الخاصة بها ، وفى كافة الاحوال فهذا لايختلف كثيرا عن الاصدار المعيارى من SQL ومثال على ذلك T-SQLالمستخدمة فى هذا الكتاب ، ربما لن تعمل اذا حاولت اجراءها على قواعد بيانات اخرى بخلاف SQL –Serve.

- البدء مع كتابة الاستعلامات Queries

كتعريف مبسط للإستعلام: هو عملية استخلاص للمعلومات من قاعدة البيانات.يستلزم ذلك وجود منصة أو نافذة لكتابة هذه الاوامر او الاستعلامات لكى نستخلص البيانات من قاعدة البيانات .

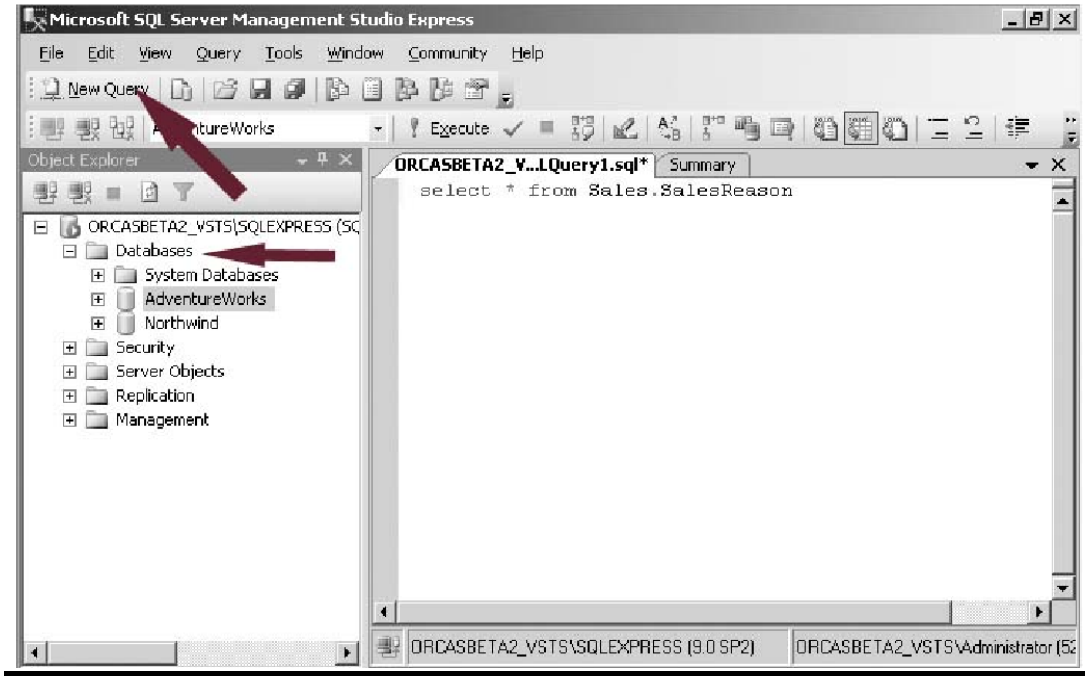
نبدأ بكتابة أول استعلام لنا ، لكن قبل ذلك تأكد من وجود SQL Server Management Studio Expressوالذى سنشير له دوما بـ SSMSE (راجع الملحق لكى تتعلم كيفية تثبيته والتعامل مع نوافذه.)

لتنفيذ استعلام بسيط جرب هذه:

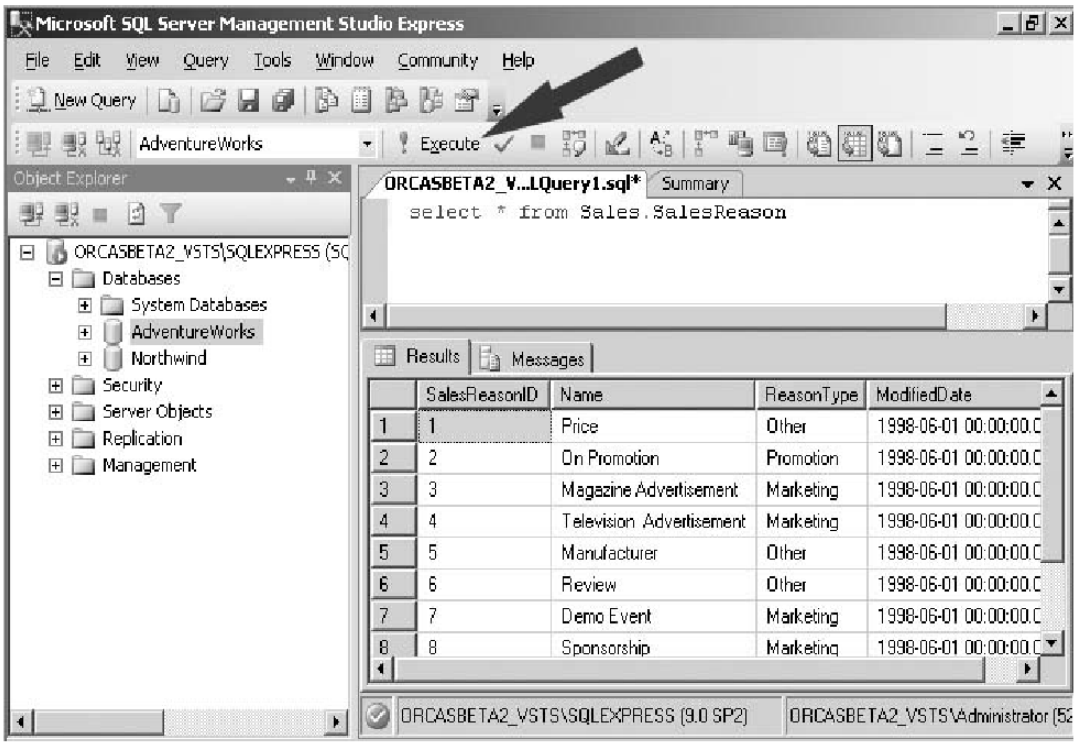
١ -افتح الـ SSMSE ثم قم بتمديد شجرة Databases واختر Adventure Works database.

٢ -اضغط على الزر New Query كما بالشكل واكتب هذا الاستعلام بداخل النافذة التى ستظهر لك

Select * from Sales .Sales Reason



٣ -اضغط على زر Execute او اضغط F5 او اختر قائمة Query -> Execute ليظهر ناتج الاستعلام كما بالشكل.



لشرح ما حدث ببساطة فإن النجمة * مع كلمة SELECT تشير الى ارجاع كافة الأعمدة من الجدول المراد عمل استرجاع لبياناته.

- اجراءات وأوامر التعامل مع الجداول

والتى سنشير اليها بـ CTEاو Common Table Expressionsوهى خاصية جديدة تم اضافتها الى SQL Server 2008 وهى ببساطة حالة مؤقتة من اجراءك لاستعلام معين مثل SELECT , INSERT,DELET UPDATEوتظهر ميزتها جليا حين تقوم بعمل استعلام ما على جدول تم اشتقاقه بدلا من استخلاصها فى جدول مؤقت ثم اجراء الاستعلامات عليه ثم حذفه ، كان هذا فى السابق مضيعة للموارد والوقت.

تتكون الـ CTE من ثلاثة أجزاء رئيسية:

1- اسم الـ CTEملحقا بكلمة WITH.

2- اسم العمود (اختيارى)

3- الاستعلام الذى يظهر بين القوسين (بعد كلمة A).

- لاشتقاق البيانات أو لأداء عدة أنواع من العمليات على قاعدة البيانات نستخدم لغة الاستفسار المهيكل (SQL) والتي تستخدم لعمل بحث search أو إحلال replace أوتعديل edit أوتحديث update أو مسح بيانات وصفية وكذا تستخدم لتخليق مخرجات جديدة من قاعدة البيانات ، وغالبا ما تستخدم هذه اللغة لاختيار أجزاء من قاعدة البيانات طبقا لمعايير criteria المستخدمين الخاصة والتي تستخدم في عمليات التحليل أو العمليات الأخرى ويلاحظ أن لهذه اللغة قواعد صياغة محددة وحازمة ومن المهم فهم معاني مفردات هذه اللغة وخاصة العوامل الشرطية conditional operators فهما جيدا.
- **تتكون جملة لغة الاستفسار المهيكل SQL من ثلاثة أجزاء مختلفة وهي:**

١ -جزء **"إختر"** select وهو يحدد البيانات الوصفية التي ينبغي تواجدها في النتيجة وهذا يحدث على سبيل المثال عندما نهتم بمجموعة محددة من الأعمدة الموجودة في قاعدة بيانات ولا نهتم بباقي القاعدة.

٢ -جزء **"من"** from ويتحدد من خلاله أسماء جداول المصدر أو قاعدة بيانات المصدر التي نستخرج منها البيانات.

٣ -جزء **"حيث"** where وهو يحدد الشروط التي ينبغي أن تحققها الصفوف الموجودة في قاعدة البيانات حتى يتم اختيارها في النتيجة ويلاحظ أن الشروط قد تكون شروطا منطقية logical expressions أو شروطا حسابية mathematical expressions أو أخرى كما يمكن وضع العديد من الشروط للاختيار مستخدمين العوامل المنطقية Boolean operators لضم النتائج لبعضها.

- يحدد العامل المنطقي Boolean operator الشرط الذي يتم تقييمه وبالتالي ينتج عن ذلك إما أن يتحقق الشرط وتكون النتيجة صحيحة true أو لا يتحقق وتكون النتيجة خطأ false وتستخدم العوامل المنطقية لضم الشروط المتعددة معا وذلك في جزء "حيث" where الموجود في جملة لغة الاستفسار المهيكل

SQL-sentence وهذه العوامل المنطقية (والتي يمكن شرحها على المثال الموضح معتبرين أن A بها بيانات مجموعة من الأولاد أما B فبها بيانات مجموعة من البنات) هي:

١ -"و" and وهو يبحث عن الشروط التي تتحقق في الجدولين أو قاعدتي البيانات فعلى سبيل المثال ومن هنا وكما هو واضح بالمثال فإنه لابد أن يتحقق الشرطان A, B وقد يكون الشرط مثلا هو أن نختار الآباء الذين لديهم ولد (جدول A) و AND لديهم في نفس الوقت بنت (جدول B).

٢ -"أو" or وهي تبحث عن الشروط التي تتحقق (true) في أحد الجدولين أو في كليهما وقد يكون الشرط مثلا أن نختار الآباء الذين لديهم إما ولد أو OR بنت (أو ولد وبنت).

٣ -**Xor** ويبحث عن الشروط التي تتحقق (true) في أحد الجدولين وليس في كليهما وقد يكون الشرط هو اختيار الآباء الذين لديهم إما ولد أو بنت (مع استبعاد من كان لديه ولد وبنت في نفس الوقت).

٤ -**Not** ويبحث عن الشرط الذي لا يتحقق (false) وقد يكون الشرط أن نختار الآباء الذين لديهم فقط أولاد (وليس لديهم بنات) أو الآباء الذين لديهم فقط بنات (وليس لديهم أولاد).

- يوضح المثال كيفية البحث واشتقاق البيانات باستخدام SQL وفيه نبحث عن الطلبة الذين يدرسون مادة "history A" وبالتالي "نختار" select الأعمدة التي ينبغي تواجدها في النتيجة وهي (على سبيل المثال) الإسم name والمادة subject والتقدير mark ونختارها من from جدول الطلبة students حيث where الشرط هو أن المادة المدروسة هي history A (=subject" history A") ومن هنا يبحث الكمبيوتر العمود " subject" في جدول "students" ليصل إلى الصفوف التي تحقق الشرط الذي تم تحديده من خلال

"حيث"where ثم يتم اختيار هذه الصفوف بالبيانات الموجودة في العمودين Name, Mark واستخراجها معا .

- نرى في هذا المثال عبارات متعددة للغة الاستفسار المهيكل SQL حيث نختار فيها كل الأعمدة باستخدام علامة wildcard "*" والتي توضع في جزء "إختر" select حيث نختار البيانات من from جدول "students" حيث where الشرط هو ("name= Robert Strand" AND "Mark>4") وهذا الشرط يتحقق فقط في صف واحد هو الذي يظهر في النتيجة لأن هذا الشخص (Robert Strand) هو الوحيد الذي حصل على تقدير أعلى من 4.

التنقيب عن البيانات أو كتالوج البيانات (البيانات التوثيقية) metadata:

- يمكن تعريف ال metadata بأنها المعلومات المخزنة عن البيانات (الموجودة في قاعدة البيانات) سواء كانت بيانات هندسية أو بيانات وصفية كما يمكن تعريفها بأنها معلومات توثيقية عن البيانات أو وصف أكثر تفصيلا للبيانات وتعد ال metadata موضوعا هاما جدا لأن توثيق قاعدة البيانات هام جدا لتعزيز المعايير المحددة الموحدة sustain certain standards لقاعدة البيانات وبالتالي فهي تضمن مستوى معين من الجودة والمرونة وكذا مستوى معين من إمكانية الاعتماد على قاعدة البيانات من خلال مستخدميها.
- أغلب الدول إما أن يكون لديها معايير محددة للبيانات التوثيقية metadata أو في طريقها لوضع هذه المعايير والأشكال التالية توضح أمثلة لبعض المعايير المقتبسة من المعيار الأوروبي لل metadata:

١ -إسم قاعدة البيانات واسم مالكها.

٢ -نظرة ووصف عام للبيانات الهندسية والبيانات الوصفية وكذا الاستخدام المقترح (المناسب) لقاعدة البيانات.

- ينبغي أن تحتوى البيانات التوثيقية metadata على المعلومات الخاصة بعلم الخرائط كالإسقاط projection ونظام الإحداثيات المستخدم coordinate

system وكذا استدلال الخريطة datum ....إلخ وإذا لم تتوافر هذه المعلومات فمن الصعب معرفة نوع الإسقاط الذي نستخدمه للتعامل مع الخريطة (إذا تم اختيار إسقاط خاطء فإن ذلك يحدث تشويها للخريطة) كما يصعب تحديد كيفية تحويل قواعد البيانات بين نظم الإحداثيات المختلفة.

- من المهم أيضا توثيق البيانات الخاصة بهياكل قواعد البيانات database structures المخزنة (سواء بيانات هندسية أو وصفية) وكذا المصطلحات الفنية وهيكل الملف file structure وكيفية تنسيق البيانات data format وذلك لتسهيل تبادل البيانات بين النظم المختلفة (سواء كانت برامج أو معايير standards) .
- يجب أيضا توثيق البيانات بالامتداد المستقبلي المخطط له (بالنسبة للبيانات) بالإضافة إلى الامتداد الواقعي الذي يمثل كل من الامتداد المكاني (الإحداثيات والوحدات الإدارية...إلخ) والامتداد الزمني وبالتالي لابد أن نفكر هل البيانات ملائمة للاستخدام (حاليا/ بعد شهرين/ بعد 6 أشهر فمثلا بعض قواعد البيانات مثل بيانات التعدلد السكاني قد تكون صالحة لفترة محدودة مما يوجب علينا توضيح ذلك في البيانات التوثيقية metadata) وهل سيتم جمع بيانات أخرى مستقبلا وكذا ما هي تواريخ جمع البيانات وما هي فترة التحديث الدوري لها وهذه المعلومات ستساعد المستخدم لتحديد مدى ملائمة البيانات للاستخدام كما أنها فرصة ليعلم المستخدم عن الامتدادات المستقبلية لقاعدة البيانات.
- يمكننا اعتبار البيانات التوثيقية metadata كإفادات للجودة والتي تعد مكونا هاما من مكونات البيانات التوثيقة التي تحتوي على:

١ -أصل قاعدة البيانات مع وصف لمصدر البيانات وعملية تطور القاعدة (معلومات عن المكان الذي استخرجت منه البيانات وكيف تم معالجتها processed قبل إدخالها لقاعدة البيانات فمثلا هل تم فحصها أو إجراء تحليلات إحصائية عليها).

٢ -الدقة الهندسية (منهجية تجميع البيانات الهندسية وما هي الدقة المستهدفة إذا تم جمع البيانات من خلال أجهزة مساحية متقدمة فإنها تكون أكثر دقة من البيانات الهندسية المستخرجة من الصور الجوية) والتي تؤثر على دقة الإحداثيات (إحداثي النقطة على الخريطة يختلف عن إحداثيها في الطبيعة بمسافة 1 متر أم 10 متر) ويستخدم لقياس هذه الدقة في بعض التطبيقات ما يطلق عليه Root Mean Square (RMS).

٣ -دقة البيانات الوصفية (كما بالبيانات الهندسية فإن منهج جمع البيانات الوصفية سيؤثر على الدقة) وهل تم جمع كل البيانات من الطبيعة بقياسات حقيقية أم تم تقديرها أو عمل interpolation لها.

٤ -الدقة الزمنية لقاعدة البيانات (هل هي صالحة للاستخدام خلال فترة معينة وما هي هذه الفترة هل هي شهر أم سنة ...إلخ).

٥ -Logical consistency (كيف تم وضع البيانات معا وهل كل كيان متصل بالآخر وهل كل كيان له علاقة بالآخرين) وهو ما يفصل طريقة إنتاج قاعدة البيانات والعمليات الفرعية المستخدمة في ذلك ومن المهم أيضا في هذا العنصر تحديد نوع ال topology (spaghetti topology of full polygon topology).

٦ -الكمال (التمام) completeness وتحدد ما إذا كانت كل المواصفات السابقة صحيحة للمساحة الجغرافية (المرصودة في قاعدة البيانات) كلها أم هناك اختلافات في الجودة أو الدقة الزمنية بين المناطق الجزئية وبعضها فقد تكون دقة الرسم في إحدى المناطق الجزئية ممتازة ولكنها أقرب للكروكي في منطقة جزئية أخرى .

- ينبغي أن تحتوي البيانات التوثيقية على البيانات الإدارية الخاصة بقاعدة البيانات وذلك كخدمة لمستخدمي قاعدة البيانات المحتملين في المستقبل لأن قاعدة البيانات يتم بيعها في أغلب الحالات ولا يتم إعطاءها مجانا وبالتالي فإن هذه البيانات الإدارية تكون بمثابة الدعاية و الإعلان للعملاء المحتملين في المستقبل

وتساعدهم في الاستفسار عن البيانات وكذا في كيفية طلب شراء هذه البيانات كما يتحدد من خلالها التنسيق format الذي يتم تسليم البيانات على صورته وكذا الوسيلة media (عن طريق البريد الإلكتروني e-mail أو CD-ROM or diskette) وكذا الخدمات الإضافية التي يمكن لمنتج البيانات القيام بها مثل إمكانية عمل التحليلات المختلفة على هذه البيانات.

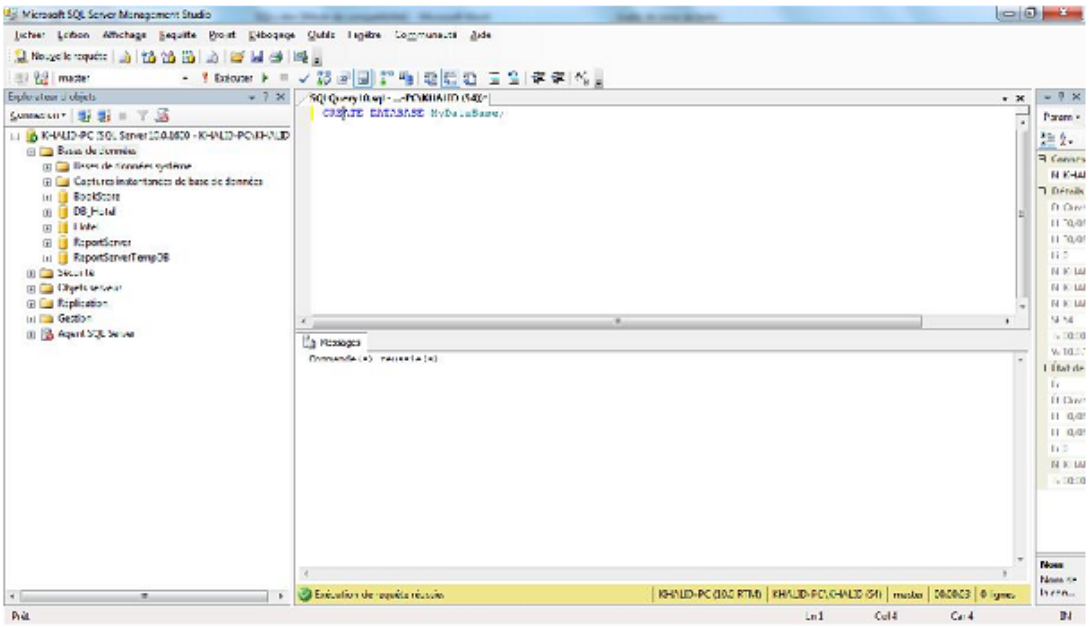
- هناك أيضا ما يسمى البيانات التوثيقية الخاصة بالبيانات التوثيقية metadata about metadata وهي هامة نظرا لأنه من المحتمل أن يكون التحديث مستمرا لقاعدة البيانات مما يظهر الحاجة لتحديث البيانات التوثيقية الخاصة بقاعدة البيانات لكننا لابد أن نأخذ في اعتبارنا أن المهمة الرئيسية ليست هي عمل البيانات التوثيقية وأن مدى شمولية البيانات التوثيقية يجب أن تكون منطقية ليس مبالغا فيها حتى نمنع العاملين من الوصول لمرحلة إنتاج البيانات التوثيقية بدلا من قواعد البيانات والتي هي أساس العمل ....التوازن مطلوب بين كل منهما.

إنشاء قواعد البيانات:

لإنشاء قاعدة بيانات بلغة SQL، فالصيغة كما يلي:

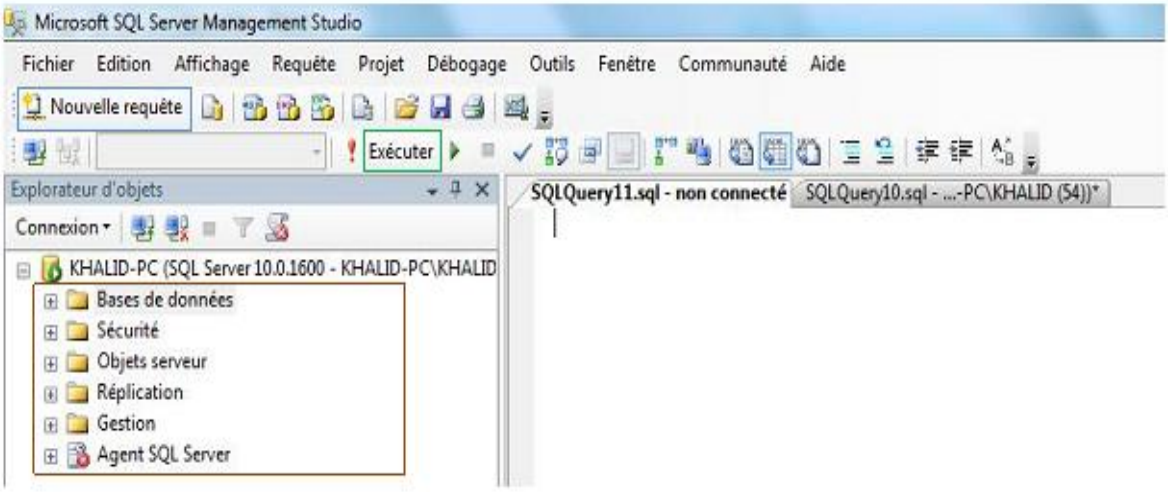
```
CREATE DATABASE MyDatabase ;
```

بحيث My Database هو اسم قاعدة البيانات التي نريد إنشاءها .



صورة لواجهة برنامج Microsoft SQL Server

والآن سنعرض أهم القوائم في هذه الواجهة، وسأترجم كل كلمة في البرنامج إلى الانجليزي:



لمنطقة المحاطة بالأزرق والتي في النسخة الانجليزية اسمها ،New Queryتمكننا من فتح صفحة جديدة لكتابة أوامر T-SQL.

المنطقة المحاطة بالأخضر اسمها ، Executeوهي خاصة بتنفيذ أوامر SQLالمكتوبة في المحرر، بإمكانك الضغط عليها أو على الزر F5.

المنطقة المحاطة باللون البني تضم مستعرض الكائنات ، Object Explorerالذي يمكننا من إنشاء الكائنات على مستوى الخادم ، Serverقواعد بيانات، جداول، ... بالإضافة إلى عمليات الحماية Securityوإدارة المستخدمين Users والإتصالات Connctions.

بإمكاننا أيضا إنشاء قاعدة البيانات يدويا عبر المعالج،من غير حاجة إلى كتابة الأمر.

- **حذف قاعدة البيانات**

نكتب:SQL لحذف قاعدة بيانات بواسطة أوامر :

```
DROP DATABASE MyDatabase;
```

- **الجداول Tables:**

الجداول عبارة عن وحدات لتخزين البيانات على شكل مصفوفة ثنائية الأبعاد تتكون من (أسطر Rows وأعمدة Columns).

- **قواعد حول التسمية :**

عند إنشاء جدول، يلزم عموما ما يلي:

- أن لا يكون اسم الجدول كلمة محجوزة في لغة SQL

- أن لا يبتدئ برقم أو برمز ماعدا بعض الرموز لن نوردها لأنها ليست موحدة بين البرامج ، فمثلا ORACLE يسمح بأن يبدأ اسم الجدول في حين بالرمز $ برنامج في حين لا يسمح برنامج SQL Server بذلك.

- إنشاء الجداول

لإنشاء جدول بواسطة أوامر SQL فالصيغة كما يلي

```
CREATE TABLE MyTable (
                        ID INT,
                        FullName VARCHAR(50),
                        BirthDate DATETIME)
```

الأمر أعلاه يقوم بإنشاء جدول MyTable ويتكون من حقول ثلاثة، الأول نوعه رقمي ، والثاني نوعه نصي يتسع لـ50 حرف، والأخير من DataTime اسمه نوع التاريخ.

-حذف الجداول:

لحذف جدول نقوم بكتابة الأمر التالي:

```
DROP TABLE MyTable;
```

بحيث MyTable هو اسم الجدول المراد حذفه.

- تعديل الجداول:

لإضافة بعض الحقول إلى جدول ما فالصيغة دائما هكذا:

```
ALTER TABLE MyTable ADD Age int;
```

هذا إذا أردنا إضافة حقل واحد للجدول عن طريق أوامر SQL ، فقط نكتب بعد الكلمة ADD اسم الحقل ونوعه لتتم إضافته إلى الجدول بعد تنفيذ الأمر.

أما إذا أردنا إضافة مجموعة من الحقول دفعة واحدة، نفصل بينها بفاصلة هكذا:

```
ALTER TABLE MyTable ADD Age int, Address VARCHAR(250);
```

-أنواع البيانات:

كل حقل من حقول أي جدول له بالضرورة نوع معين من البيانات، حسب القيمة المراد تخزينها فيه، وتنقسم أنواع البيانات إجمالا إلى:

1- الأنواع الرقمية:

وتستعمل لتخزين القيم الرقمية، مثلا لو عندنا حقل Ageفي إحدى الجداول العمر فحتما علينا اختيار نوع رقمي لتخزين قيم الأعمار، والأنواع الرقمية بدورها تنقسم إلى:

- TEXT: وهي قديمة الاستخدام وغير محددة المساحة ومثلها أيضاً NEXT .
- BIT: تستخدم لتخزين الأرقام وليس الحروف فهي تستطيع تخزين أرقام (0 - 255) وهي قيمة وليس عدد، بمعني أنة لو قمنا بإختيار نوع البيانات bit وفي العمود المحصص أدخلنا رقم 256 لا يقبله البرنامج.
- SMALL INTEGER: تخزن أرقام صحيحة وموجبة ولكن أقل من integer فهي تخزن حتي 32.767 وهو أيضاً قيمة وليس عدد.
- INTEGER: وتستخدم في تخزين أرقام موجبة وسالبة صحيحة بدون كسور وتستطيع تخزين كمية كبيرة نسبياً من الأرقام.
- BIG INTEGER: يخزن الأرقام السالبة والموجبة ، صحيحة بدون كسور ولكن بكميات كبيرة جداً.
- TINY INTEGER : وهي تساوي bit وتخزن من 5- 255 حرف فقط .
- REAL FLOAT : وتستخدم في تخزين الأرقام الكسرية ولكن real تخزن كمية أكبر مما تخزنه float وتخزن أرقا موجبة وسالبة .
- NUMERIC, DECIMAL: وهو تعني أن هذا العمود يوضع به أرقام وهي أعداد صحيحة أو كسرية.

2- الأنواع النصية :

تستخدم لحفظ البيانات من نوع نصي، على سبيل المثال لو عندنا حقل لحفظ اسم أو عنوان أو أي قيمة نصية، فيلزم أن نختار نوع بيانات نصي، وتنقسم الأنواع النصية إلى:

- VAR CHAR وكلمة var إختصار لكلمة variable والتي تعني متغير وهذا النوع سوف يخصص مساحة لل DB نأخذ منه ما نريد وتأخذ هي الباقي من المساحة مرة أخري فهي تمكننا من التحكم في المساحة.
- Char : وهي تستخدم لكتابة البيانات أو الأرقام ، ولكن الأرقام هنا تتعامل معاملة الحروف، ونلاحظ أن الحروف لا تشغل مساحة علي HD علي عكس الأرقام التي تشغل مساحة كبيرة.
- TEXT: وهي قديمة الاستخدام وغير محددة المساحة ومثلها أيضاً NEXT .
- N CHAR: وهي مثل charتستخدم مع الأرقام والحروف ولا تسترجع إي شيء من المساحة.
- NVAR CHAR: وهي التي تجمع بين مميزات n و var فهي تستخدم في كتابة الأرقام والحروف، كما أنها تجعنا نتحكم في المساحة .

- VAR CHAR MAX: وهي نفسها var char مع إختلاف أنها هنا لا تحدد إي حد أقصي للساحة، ولكن تتركها مفتوحة.
- N VAR CHAR MAX: وهي تجمع بين كل المميزات السابقة لل N و var و MAX.

3- التاريخ والوقت

نحتاج هذا النوع من البيانات لحفظ بعض القيم التي تكون على شكل تاريخ ووقت مثل: تاريخ البيع أو الشراء...، ومن أهم أقسام

- VAR VAR CHAR DATE, TIME: وهي تخزن تاريخ ووقت، أو التاريخ فقط.
- TIME STAMP: وهي بعد إدخال البيانات من قبل المستخدم يقوم البرنامج بتسجيل الوقت والتاريخ الذي تم فيه إدخال البيانات ولا يعرفه المستخدم.
- SMALL DATE TIME: يسجل التاريخ والوقت ولكن بشكل مختصر.
- DATE, TIME: وهي تخزن تاريخ ووقت، أو التاريخ فقط.
- SQL VARIANT: وهي إدخال أي نوع من البيانات سواء كانت أرقام، وحروف، وصور، أو تاريخ ولكنها متصلة.
- BINARY: وتستخدم لتخزين الأعداد ذات النظام الثنائي .
- MANY: وهو يخزن الأموال والفرق بينه وبين الأنواع التي تخزن الأرقام هو أنه يميزها بالعملة التي تستخدمها في كل بلد.
- IMAGE: وهي التي تسمح بتخزين الصور.

<u>بعد إنشاء الجدول يلاحظ ما يلي:</u>

1- ظهور قوسين [ ] علي أسم Field عند إدخال البيانات وهو بسبب.

- وضع مسافة أثناء كتابة أسم Field .
- وضع Dote.
- كتابة كلمة Add وهي معروفة في البرمجة.

2- ظهور رسالة خطأ أثناء إدخال البيانات وذلك عندما تم عمل تعديل في الجدول أثناء عملية الإدخال . ولهذا علينا عد عمل أي تعديل في بيانات الجدول إلا والجدول مغلق.

الفصل الثالث: خاصيات الإدخال Constraints :

الفصل الثالث: خاصيات الإدخال Constraints :

وهي مجموعة من الأوامر التي نطبقها على الحقول، من أجل التحقق من القيمة المراد حفظها، وأول هذه الكلمات هي:

- **NOT NULL:**

وتستعمل هذه الخاصية لمنع ترك قيمة حقل معين فارغة، وصيغتها هكذا

```
CREATE TABLE MyTable (
                        ID INT NOT NULL,
                        FullName VARCHAR(60))
```

ويمكننا تطبيق الخاصية NOT NULLحتى بعد إنشاء الجدول كما يلي:

```
--Duting creation of table
CREATE TABLE MyTanble (
                      ID INT NOT NULL,
                      FullName VARCHAR(60))

--After creation of table :

ALTER TABLE MyTable
ALTER COLUMN ID INT NOT NULL
```

كما تلاحظ قمنا بتحديد اسم ونوع الحقل المراد تطبيق الخاصية عليه

- **IDENTITY:**

هذا النوع من الكلمات يطبق فقط على الحقول التي يكون نوعها رقميا من قيمها تزداد تلقائيا عند إضافة أي سطر جديد، مثلا لو أنشأت أجل جعل هذه الكلمة على الحقل ID، الجدول التالي وطبقت فسوف تزداد قيمته تصاعديا :

```
CREATE TABLE MyTable (
                        ID INT IDENTITY(1,1),
                        FullName VARCHAR(60))
```

عند إضافة أي سطر جديد، ستلاحظ بأن خانة الحقل ID تمنعك من الكتابة فيها، ولكن بمجرد ما تتجاوزها تقوم برفع قيمة الحقل بواحد تلقائيا، كما تبين الصورة التالية:

	ID	FullName
	1	Khalid
	2	Ahmed
	3	Youssef
▶*	NULL	NULL

تستطيع تغيير القيمة التي يبدأ منها الحقل، وأيضا معامل زيادة القيمة هكذا:

```
CREATE TABLE MyTable (
                        ID INT IDENTITY(2,3),
                        FullName VARCHAR(60))
```

في هذا المثال، ستبدأ قيمة الحقل من الرقم 2، ويكون معامل الزيادة هو 3، أي أن القيمة الأولى ستكون 2، والقيمة الثانية ستكون 5، والثالثة ستكون 8 ... إلخ.

- **PRIMARY KEY:**

تطبق هذه الخاصية على الحقل لجعل قيمته متفردة، غير قابلة للتكرار ولا ينبغي أن تترك قيمة الحقل الذي تطبق عليه هذه الخاصية فارغة أي ينبغي أن تكون NOT NULL، فمثلا لو عندنا جدول يضم بيانات المواطنين، فمن غير شك هنالك حقل يميز كل مواطن عن غيره، في هذه الحالة الحقل الذي سيكون مفتاحا أساسيا PRIMARY KEY هو الحقل الذي يضم أرقام بطاقات التعريف الوطنية، لأنه من الممكن أن تجد أكثر من مواطن يحملون نفس الاسم، ولكن يستحيل أن يحملوا نفس رقم بطاقة التعريف، عند التعامل مع جداول متصلة فيما بينها عن طريق العلاقات يصبح إلزاميا استخدام المفاتيح الأساسية.

لجعل أحد الحقول مفتاحا أساسيا، نكتب الصيغة التالية

```
CREATE TABLE MyTable (
                        ID INT PRIMARY KEY,
                        FullName VARCHAR(60))
```

نستطيع تسمية خاصية التحقق PRIMARY KEY أثناء تطبيقها على إحدى الحقول بالصيغة التالية:

```
CREATE TABLE MyTable (
                       ID INT CONSTRAINT PK_MyConstraint PRIMARY KEY,
                       FullName VARCHAR(60))
```

في هذه الحالة أسمينا الخاصية PRIMARY KEY المطبقة على الحقل ID ب PK_MyConstraint.نستطيع أيضا تطبيق هذه الخاصية حتى بعد إنشاء الجدول، عن طريق الخاصية TABLE،ALTER أي هكذا:

```
--creation of table
CREATE TABLE MyTable (
                       ID INT NOT NULL,
                       FullName VARCHAR(60))

--Add primary key constraint :

ALTER TABLE MyTable ADD CONSTRAINT PK_MyConstraint PRIMARY KEY(ID)
```

وذهبت إلى الجدول في القائمة، ودخلت إلى الأعمدة Columns ستجده بالشكل التالي:

رمز المفتاح للدلالة على أن الحقل IDحقل أساسيPRIMARY KE.

- **UNIQUE**:

تستعمل هذه الخاصية لمنع قيمة حقل معين من التكرار، وبالتالي لا يمكن أن يضم في الحقل قيمة أكثر من مرة، الفرق بين الخاصية UNIQUEوبين الخاصية PRIMARY KEY، أن هذه الأخيرة لا يمكن تطبيقها على الحقول التي تسمح بالقيم الفارغة NULL، بينما تسمح الخاصية UNIQUEبالقيمة NULL، ويمكننا تطبيق هذه لخاصية على الحقل هكذا:

```
CREATE TABLE MyTable (
                        ID INT UNIQUE,
                        FullName VARCHAR(60))
```

ويمكننا تطبيقها على جدول منشأ مسبقا كما يلي:

```
--creation of table
CREATE TABLE MyTable (
                      ID INT UNIQUE,
                      FullName VARCHAR(60))

--Add unique constraint :

ALTER TABLE MyTable ADD CONSTRAINT U_MyConstraint UNIQUE (ID)
```

- **REFERENCE**:

هذه الخاصية تطبق على الحقول الأجنبية القادمة من جدول آخر، وتستعمل لتعريف الحقل الأجنبي وتحديد الجدول القادم منه.

في هذا المثال سننشئ جدولين، ونحاول تطبيق الخاصية REFERENCEعلى الجدول الثاني :

```
--creation of the first table
CREATE TABLE MyTable1 (
                        ID INT NOT NULL PRIMARY KEY ,
                        FullName VARCHAR(60))
```

```
--creation of the second table
CREATE TABLE MyTable2 (
                    Code INT NOT NULL PRIMARY KEY ,
                    ID INT CONSTRAINT PK_MyTable2  REFERENCES MyTable1(ID)
                          )
```

لجدول الثاني يتكون من حقلين، الأول اسمه Codeوهو الحقل الأساسي، والثاني اسمه IDوهو أجنبي قادم من الجدول الأول .

لو ذهبت إلى الجدول في القائمة، ودخلت إلى الأعمدة Columnsستجده بالشكل التالي:

المفتاح الذهبي للدلالة على أن الحقل أساسي ، PRIMARY KEY والمفتاح الرمادي للدلالة على أن هذا الحقل أجنبي FOREIGN KEY.

ونستطيع أيضا تطبيق الخاصية REFERENCESعلى جدول تم إنشاؤه مقدما، وتكون الصيغة هكذا:

```
ALTER TABLE MyTable2
ADD CONSTRAINT FK_Constraint
FOREIGN KEY (ID)
REFERENCES MyTable1(ID)
```

بحيث FK_Constraint هو اسم الخاصية REFERENCES التي نحن بصدد إنشائها، أما IDالأولى المقرونة ب FOREIGN KEY فهو اسم الحقل الأجنبي في الجدول الثاني، و ID الثانية المقرونة ب REFERENCES هو اسم الحقل الأجنبي في الجدول المصدر)في حالتنا هذه هو الجدول الأول .)

لتفادي وقوع مشاكل، يستحسن إضافة الخاصية REFERENCES بعد إنشاء الجداول، لأن الجداول التي تضم الحقول الأساسية يجب أن تنشأ أولا قبل الجداول التي تضم الحقول الأجنبية.

- **DEFAULT:**

الغاية من هذه الخاصية هي تحديد قيمة افتراضية للحقل المطبقة عليه تفاديا للفراغ ،NULL وطبعا هذه القيمة الافتراضية لا تحفظ إلا إذا ترك المستخدم قيمة الحقل فارغة، ولتطبيق هذه الخاصية على حقل معين فالصيغة كما يلي:

```
CREATE TABLE MyTable(MyColumn nvarchar(25) DEFAULT 'UnKnown')
```

إذا كان نوع الحقل رقميا، نكتب القيمة من غير علامات التنصيص، إذا أردنا تطبيق هذه الخاصية على حقل موجود مسبقا فالصيغة كما يلي:

```
ALTER TABLE MyTable ADD CONSTRAINT D_Constraint DEFAULT 'UnKnown' For
MyColumn
```

- **CHECK:**

تستعمل هذه الخاصية للتحقق من قيمة الحقل قبل إدخاله، فإن كانت القيمة تتوافق مع الشرط المصحوب بالخاصية تمت عملية الإدخال، وإن كانت القيمة تخالف الشرط يتم منع عملية الإدخال، وصيغتها هكذا:

```
CREATE TABLE Person(
      CIN CHAR(9),
      FullName NVARCHAR(75),
      AGE INT CONSTRAINT C_Constraint CHECK (AGE BETWEEN 5 AND 160))
```

في هذا المثال سيسمح فقط بإدخال قيمة العمر المحصورة بين 5سنوات و

160سنة، ويمكنك تعويض الشرط الموجود بين القوسين بأي شرط تريد .

يمكننا إضافة الخاصية CHECKحتى بعد إنشاء الجدول، بالصيغة التالية:

```
ALTER TABLE Person
ADD CONSTRAINT C_Constraint
CHECK (Age BETWEEN 5 AND 160)
```

- حذف خاصيات الإدخال:

لحذف خاصيات الإدخال، نكتب الصيغة التالية:

```
ALTER TABLE Person
DROP CONSTRAINT C_Constraint
```

بحيث Personهو اسم الجدول، و C_Constraintاسم خاصية الإدخال المراد حذفها.

- المشاهدViews:

المشاهد Viewsعبارة عن جداول وهمية، والتي لها نفس دور الجداول الحقيقية إلا أنها أخف وأسرع لأ نها لا تحتوي على البيانات وإنما تحتوي فقط على الاستعلام الذي يجلب البيانات من الجداول ، بالإضافة إلى هذه المزية تمكننا المشاهد من تحديد عدد الحقول المراد تضمينها في الاستعلام، ناهيك عن كو苦ا تمكننا من جلب البيانات من مجموعة من الجداول المترابطة وتجميعها على شكل جدول واحد.

حيث أن الجداول المكونة لقاعدة البيانات لها وجود فعلي مادي، هنالك مساحة على القرص الصلب تشغلها البيانات الموجودة في الجدول. يمكن ان نعرف جداول اخرى بناء على تلك الجداول غير موجودة ماديا ولا تشغل حيز لكن يمكننا ان نتعامل معها وكانها موجودة حقيقةً هذه الجداول تسمى مناظر views.

المنظر هو عبارة عن جدول مبني من كل او جزء من جدول او اكثر. لان المناظر غير موجودة بشكل مادي تسمى في بعض الاحيان جداول افتراضية virtual

tables. و الجداول الاخرى المادية تسمى الجداول الحقيقية او الاساسية. المناظر مثل الجداول يتم انشائها باستخدام جملة CREATE و خلافا للجداول تكون السجلات المكونة لها موجودة مسبقا في قاعدة البيانات.

الصيغة العامة لجملة انشاء المنظر هي:

```
CREATE VIEW <view> ( <column names>) AS
SELECT <column names>
FROM <table>
WHERE <predicate>
```

اسم المنظر يتبع كلمة VIEW ثم يليه اسماء الاعمدة داخل اقواس تفصلها فواصل دون تحديد انواعها، يمكن عدم تحديد اسماء الحقول حيث يتم تسميتها بناء على اسماء الحقول في نتيجة الاستفسار، و بعد كلمة AS تأتي جملة الاستفسار التي يمكن ان تكون اي جملة استفسار بدون أي تقييد لمدى تعقيدها سواء كانت من جدول او اكثر او حتى باستخدام جملة GROUP BY، او باستخدام عمليات حسابية و تعابير رياضية.

من استخدامات المناظر الشائعة بعض المعلومات عن بعض المستخدمين فمثلا اذا اردنا ان نعطي للطالب امكانية ان يرى اسماء المدرسين و بعض المعلومات عنهم و نريد ان نخفي عنه رواتبهم مثلا، يمكن ان نعطي الطالب امكانية الاستفسار من المنظر كما نرى في المثال التالي.

مثال: أنشيء منظر يحتوي على جميع حقول teachers عدا حقل الراتب.

```
CREATE VIEW faculty AS

SELECT teacher_num, teacher_name, phone

FROM teachers
```

في هذا المثال لم نعطي اسماء لحقول المنظر وبالتالي تكون أسماء الحقول في المنظر مطابقة لأسماء الحقول في الجدول الاصلي. يستخدم المنظر كما يستخدم الجدول العادي حيث يمكن استرجاع المعلومات منه بالطريقة التي نريد لكن هنا نضمن ان الراتب لن يظهر في اي استفسار على المنظر.

مثال: أنشيء منظر يحتوي على اسماء و ارقام الطلبة من المستوى الرابع

```
CREATE VIEW grad_students AS

SELECT sname, sid, level

FROM student

WHERE level = 4
```

يمكن انشاء مناظر تحتوي على حقول غير موجودة في اي من الجداول الاساسية لكنها مشتقة من الجداول الاساسية. تسمى الاعمدة المشتقة(derived columns).

مثال: انشيء منظر يحتوي على اسماء المدرسين و ارقامهم وراتبهم السنوي

```
CREATE VIEW year_sal (teacher_name, teacher_num, y_salary) AS

SELECT teacher_name, teacher_num, salary*12

FROM teachers
```

اضافة السجلات الى المنظر ليست بالبساطة التي يمكن ان تنفذ على الجداول الاساسية لان المنظر ليس له وجود مادي اي ان اضافة سطر على المنظر تعني اضافة على الجدول الاساسي. و يمكن ات تكون اضافة الى اكثر من جدول و في احيان اخرى قد يحتوي المنظر على حسابات مما يعقد الامور او قد يجعلها مستحيلة. كقاعدة عامة اذا كان المنظر يستخدم جملة استفسار تحتوي على GROUP BY او دالة تجميع او عمود مشتق يكون للاستفسار فقط.

ما ذا يحدث اذا اردنا ان نضيف سجل لا يحقق شرط المنظر كما في المثال التالي:

```
CREATE VIEW grad_students AS

SELECT sname, sid, level

FROM student

WHERE level = 4
```

اذا اردنا ان نضيف السجل

```
INSERT INTO grad_student VALUES (‘Jamal’, 100, 3)
```

هذا السطر لا يسبب مشكلة اذا اضيف الى الجدول الاساسي لكن لا يمكن ان يظهر في المنظر لانه لا يحقق الشرط. هنا يمكن اضافة السطر. و يمكننا ان نمنع اضافة السطر اذا اضفنا عبارة WITH CHECK OPTION كما يلي:

```
CREATE VIEW grad_students AS

SELECT sname, sid, level

FROM student

WHERE level = 4

WITH CHECK OPTION CONSTRAINT grad_constraint
```

وجود هذه العبارة يمكنع اضافة اي سجل لا يحقق الشرط المحدد في جملة الاستفسار المستخدمة لإنشاء المنظر.

اذا لم نعد بحاجة الى المنظر يمكننا ان نحذفه دون التاثير على وجود الجداول الاساسية كما يلي:

```
DROP VIEW grad_student
```

إنشاء المشاهد:

لإنشاء مشهد Viewمعين، فالصيغة العامة كما يلي:

```
CREATE VIEW View_Name
AS Query
```

بحيث View_Nameهو اسم المشهد المراد إنشاؤه، وهذا مثال لكيفية إنشاء مشهد يجلب بيانات الموظفين الذين يسكنون مدينة الرياض:

```
CREATE VIEW Emp_Riad
AS
SELECT ID, FullName, Age FROM Employee WHERE [City]='Riad'
```

لو أردت رؤية المشهد الذي قمت بإنشائه اذهب إلى قاعدة البيانات، ستجد في التبويب الذي يوجد فيه الجداول، تبويبا اسمه ، Viewsادخل إليه وستجد المشهد بالإسم الذي أعطيته إياه.

ادخل إليها ستجدها شبيهة جدا بالجداولTables .

بإمكانك إنشاء المشاهد أيضا من هناك، من دون الحاجة إلى كتابة أوامر SQL، فقط اتبع مراحل الإنشاء، بعد الضغط بيمين الفأرة على Viewsواختيار New View.

- حذف المشاهدViews:

لحذف مشهد معين، قم بالضغط عليه بيمين الفأرة واختر الأمر Delete، أو اكتب الأمر التالي:

```
DROP VIEW View_Name
```

- الفهارسIndexes:

هي شبيهة بالفهارس التي تعرفها، والتي تكون موجودة غالبا في أواخر صفحات الكتاب، كما تعلم فهذه الفهارس توضع من أجل تمكين القارئ من الوصول إلى الموضوع الذي يبحث عنه بكل سرعة. نفس الشيء ينطبق على الفهارس في لغة SQL، إذ أن الغاية من إنشاء الفهارس هو تسريع عمليات جرد البيانات (Select،) لكن عيبها هو أنها تبطئ عمليات تحديث البيانات من إضافة وحذف وتعديل، لأن كل عملية تحديث تتسبب في إعادة إنشاء الفهرس، لهذا وحسن التقليل منها .

إنشاء الفهارس:

لإنشاء فهرس جديد على إحدى الحقول في جدول ما، فإن الصيغة كما يلي:

```
CREATE INDEX Index_Name
ON Table_Name (Column_Name Desc | Asc)
```

بحيث Index_Name هو اسم الفهرس، و Table_Name اسم الجدول المعني بالأمر، و Column_Name اسم العمود /الحقل المراد عمل فهرسة له . أما Descو Asc فهي تحدد طريقة جرد البيانات وهي إما تصاعدية Ascendingأو تنازلية Descending.وهذا مثال لإنشاء فهرس:

```
CREATE INDEX MyIndex
ON MyTable (ID Desc)
```

حذف الفهارس:

لحذف فهرس بواسطة أوامر SQL، فالصيغة كما يلي:

```
DROP INDEX MyIndex
ON MyTable
```

الفصل الرابع: معالجة البيانات بإستخدام لغة الإستعلام SQL

الفصل الرابع: معالجة البيانات بإستخدام لغة الإستعلام SQL

تحدثنا عن مفهوم معالجة البيانات في بداية الفصل الثاني، وقلنا أنها تشمل كل عمليات الإضافة والتعديل والحذف التي تطال الجداول.

Insert : **اضافة سجلات للجداول:**

بعد انشاء الجدول يمكننا ان نملأه قيم. جميع القيم المخزنة في الجداول تخزن على شكل سجلات لذا لاضافة قيمة على الجدول يتعين علينا ان نضيف سجل كامل.
لاضافة سجل نستخدم جملة INSERT.

الشكل العام لجملة INSERT هو:

INSERT INTO <table> (<column names>) VALUSE (<values>)

تضيف هذه الجملة سطر على الجدول المسمى <table>. يمكن ان يحتوي السجل الجديد على قيم لكل او بعض الحقول في السجل. يمكن تحديد اسماء الحقول المراد اضافة قيم عليها بعد اسم الجدول و يمكن الاستغناء عن تحديد اسماء الحقول اذا اردنا اضافة قيم على جميع الحقول. ثم تاتي كلمة VALUES متبوعة بالقيم التي نريد ان نضيفها الى السجل.

مثال: لاضافة سجل طالب جديد على جدول students المعرف في اعلاه.

```
INSERT INTO students
(sid,    name,    login,age,  avg)
VALUES ( 303,‘Sulaiman’,‘sul303’,19,70.1 )
```

او

```
INSERT INTO students  VALUES (
303,‘Sulaiman’,‘sul303’,19,70.1 )
```

كلا الجملتين تضيف سجل الى الجدول.

يجب ان تتلائم انواع القيم المدخله مع انواع الحقول. على سبيل المثال لا يمكن ان نضع القيمة "غير معرف" الى حقل المعدل avg لان القيمة هي CHAR و الحقل معرف على ان يكون من نوع FLOAT.

السجلات لا تكون مرتبة داخل الجدول اي ان السجل اما ان يكون في الجدول او لا يكون ولا معنى للسؤال اين يوجد السجل داخل الجدول او ما هو ترتيبه. اضافة سجل الى الجدول لا تعني انه يضاف الى اخر الجدول او الى اول الجدول او بعد سجل معين، فقط يصبح السجل عنصر جديد في الجدول.

رأينا في المثال انه يمكن تحديد اسماء الحقول او عدم تحديدها. لكن يجب ان تكون القيم مرتبة حسب الترتيب الذي رتبت به الحقول وقت انشاء الجدول. اذا اردنا ادخال القيم بترتيب غير ترتيب الحقول يترتب علينا ان نعطي قائمة بالحقول مرتبة حسب الترتيب المراد اعطاء القيم به.

مثال:

```
INSERT INTO students
    (name,      login,   sid,  age,  avg)
VALUES ( ‘Sulaiman’,‘sul303’, 303, 19,  70.1 )
```

هنا قمنا بادخال القيم بترتيب غير الترتيب الذي انشأت به الحقول. لاحظ ان الحقول مرتبة و السجلات غير مرتبة.

ماذا يحدث اذا اضفنا قيم في بعض الحقول لسجل جديد؟ مثال:

```
INSERT INTO students
(sid, name, login )
VALUES ( 303,‘Sulaiman’,‘sul303’ )
```

ما هي القيمة التي سوف ياخذها الحقول age, avg ؟ كل حقل يجب ان ياخذ قيمة لذا فان القيمة التي ستخزن في الحقول التي لم يتم تحديد قيم لها هي NULL. طبعا اذا اردنا ان نضيف سجل يجب ان نعطي قيمة لجميع الحقول التي وضعنا عليها المحدد NOT NULL.

لجملة INSERTشكل اخر حيث يمكن اضافة حقول باستخدام جملة استفسار SELECT. الصيغة العامة للجملة هي:

```
INSERT INTO <table>(<column names>)
SELECT <column names>
FROM <table>
WHERE < PREDICATE>
```

مكان كلمة VALUES و قائمة القيم نضع جملة استفسار، يمكن أن تكون جملة الاستفسار بسيطة او معقدة بالشكل اللذي نريد لافرق. لا يقوم المستخدم بتحديد القيم المراد ادخالها. يتم اضافة نتيجة الاستفسار من جدول او اكثر الى الجدول المحدد في جملة INSERT. مثلا نفرض اننا نريد اضافة الطلاب الحاصلين على معدل 90 فما فوق الى جدول المدرسين كمساعدي تدريس. طبعا الحقول في جدول الطلاب يختلف عن الحقول في جدول المدرسين. يمكن ادخال اسم الطالب في حقل اسم المدرس و رقم الطالب في حقل رقم المدرس. اما باقي الحقول فتبقى فارغة.

```
INSERT INTO teachers (teacher_num, teacher_name)
```

```
SELECT sid, name

FROM students

WHERE avg >=90
```

عند تنفيذ هذه الجملة يتم اضافة عدة حقول الى جدل المدرسين في كل حقل جديد رقم الهاتف والراتب يكون بهم القيمة الخالية NULL.

مثلا عندي جدول يضم قائمة الموظفين، وأريد أن أضيف موظفا جديدا إلى هذا الجدول، سأفترض أن الجدول يضم حقلين فقط وهما: رقم الموظف واسمه الكامل، لفعل ذلك فالصيغة كما يلي:

```
INSERT INTO MyTable
(ID, FullName)
VALUES
(1, 'Khalid')
```

بحيث My Table هو اسم الجدول الذي نريد إضافة البيانات إليه، و IDو Full Name هما الحقلان المشكلان لهذا الجدول، أي أنها حقل من حقول الجدول لا يكتب بين الأقواس توضع فيه القيمة NULL افتراضيا .

إذا كان أحد الحقول يطبق الخاصية IDENTITY التي تجعل الرقم يزداد تلقائيا، فلا ينبغي أن نضعه مع الحقول.

إذا كنا متأكدين من ترتيب الحقول، فبإمكاننا إلغاء ذكر الحقول في الصيغة والاكتفاء فقط بإدخال القيمة، كما نعرض هنا.

```
INSERT INTO MyTable
VALUES
(1, 'Khalid')
```

نسخ البيانات من جدول إلى آخر:

بإمكاننا نسخ بيانات جدول معين، ونقلها إلى جدول ثان والصيغة كما يلي:

```
INSERT INTO MyTable2 (ID,FullName)
SELECT ID,Fullname
FROM MyTable1
```

في السطر الأول قمنا بتحديد اسم الجدول المراد نسخ البيانات إليه مع تحديد أسماء الحقول، ثم بعد ذلك نقوم بجلب قيم الحقول من الجدول المصدر عن طريق الكلمة Selectالتي سنراها بالتفصيل فيما يلي:

-إضافة البيانات إلى جدول في نفس لحظة إنشائه:

بإمكاننا تعبئة جدول في لحظة إنشائه بعناصر جدول آخر موجود مسبقا، للقيام بذلك فالصيغة كما يلي:

```
SELECT *
INTO NewTable
FROM OldTable
```

لنجمة *تعني جلب جميع الحقول، إذا قمت بتنفيذ هذا الكود سوف تكون النتيجة عبارة عن جدول جديد، اسمه New Table معبأ بنفس بيانات الجدول Old. Table

-حذف البيانات Delete:

الصيغة العامة لالغاء السجلات من الجداول هي:

```
DELETE FROM <table>

WHERE <predicate>
```

يتم حذف جميع السجلات التي ينطبق عليها شرط عبارة WHERE من الجدول المحدد بعد كلمة FROM. تشبه الى حد ما جملة الاستفسار لكن بدلا من ان نقوم باسترجاع السجلات يتم حذفها. (يمكنك تنفيذ جملة استفسار قبل الحذف للتأكد من السجلات المراد حذفها ثم استبدال عبارة SELECT بكلمة DELETE مع الابقاء على عبارة WHERE.

تقوم جملة DELETE بحذف سجل او اكثرا اعتمادا على الشرط المحدد في جملة WHERE مثال اذا اردنا حذف سجل الطالب رقم 234 ننفذ الجملة التالية:

```
DELETE FROM student
WHERE sid =234
```

يمكن ان نزيد من تعقيد جملة WHERE كما نشاء بنفس الطرق التي طبقناها في جملة الاستفسار.

اذا حذفنا جملة WHERE يتم حذف جميع السجلات الموجوة في الجدول مع الابقاء على تعريف الجدول اي يتم حذف السجلات دون الغاء وجود الجدول , و يمكننا بعد ذلك اضافة سجلات على الجدول. مثال:

```
DELETE  FROM student
```

يتم حذف جميع السجلات في الجدول student

مثال: احذف جميع السجلات في جدول enroll بحيث يكون حقل العلامة فارغ

```
DELETE FROM enroll
WHERE mark IS NULL
```

مثال: احذف جميع السجلات الموجودة في جدول enroll بحيث لا يكون له سجل مقابل في جدول student

```
DELETE FROM enroll E

WHERE NOT EXISTS

      ( SELECT sid from student S

            WHERE S.sid= E.sid)
```

مثال: احذف جميع السجلات التي تحتوي على علامات خاطئة في جدول الطلاب student

```
DELETE FROM student

WHERE mark >100 OR mark <35
```

ومثال لذلك نفترض أن لدينا جدولا يضم مجموعة من الموظفين، ونريد حذف الموظفين الذين يقطنون بمدينة الرباط.

```
DELETE FROM Employee
WHERE Adress= 'Rabat';
```

مثال آخر يقوم بحذف الموظفين الذين عمرهم أكبر من 60سنة:

```
DELETE FROM Employee
WHERE Age > 60;
```

-تعديل البيانات Update Data:

بعد اضافة السجلات الى الجداول نكون في بعض الاحيان بحاجة الى تعديد القيم الموجودة في السجلات. جملة التعديل في لغة SQL لها الصيغة العامة التالية:

```
UPDATE <table>
SET <column name> = <value>
WHERE <predicate>
```

اسم الجدول المراد تعديل سجلاته يظهر بعد كلمة UPDATE مباشرة. و بعد كلة SET نضع مجموعة من جمل التخصيص لتعديل القيم الموجودة في الحقول على صيغة (اسم الحقل = القيمة). يجب ان نضع اسم الحقل و بعد اشارة = نضع قيمة ثابتة او تعبير من اي نوع. يمكن في النهاية اضافة عبارة WHERE لتحديد السجلات المراد تعديل حقولها. اذا لم تظهر جملة WHERE يسري التعديل على جميع سجلات الجدول المحدد.

مثال : نريد تعديل معاش (راتب) المدرس رقم 121 الى 800.

```
UPDATE teachers
SET salary = 800
WHERE teacher_num = 121
```

يمكن استخدام التعابير الحسابية لتعديل القيم في الحقول

مثال: نريد اضافة غلاء معيشة بنسبة 5% الى راتب المدرس رقم 121

```
UPDATE teachers
SET salary = salary*1.05
WHERE teacher_num = 121
```

لاحظ استخدام اسم الحقل بعد اشارة = حيث تعنيي القيمة الموجودة في الحقل قبل التعديل. يمكن تعديل حقلين في الان ذاته

مثال:

```
UPDATE teachers
SET teacher_name = 'Khalil' ,
phone = '12348743'
WHERE teacher_num = '124'
```

لاحظ الفاصلة التي تفصل بين حقل واخر.

يمكن حذف عبارة WHERE لتعديل جميع السجلات.

مثال: لاعطاء علاوة غلاء المعيشة لجميع المدرسين بنسبة 5%

```
UPDATE teachers
SET salary = salary*1.05
```

يمكن ان تكون عبارة WHERE معقدة، فمثلا اذا اردنا ان نرفع معاشات المدرسين اللذين تقل معاشاتهم عن المعدل بنسبة 7.5% ننفذ الجملة التالية:

```
UPDATE teachers
SET salary = salary*1.075
WHERE salary < (SELECT AVG(salary)
                FROM teachers)
```

لاحظ انه يتم تنفيذ الجملة الفرعية اولا لتحديد المعدل ثم تنفيذ الجملة الاساسية.
عكس الترتيب يغير من النتيجة لان الجملة الخارجية تغير من قيمة المعدل.

في معظم الأمثلة القادمة سنشتغل على جدول الموظفين، لهذا يستحسن أن تنشئه الآن لتجريب الأوامر، بإمكانك إنشاؤه يدويا عن طريق نافذة التصميم، أو عبر أوامر SQL التي رأيناها آنفا.
بالنسبة لحقول الجدول فلست ملزما باتباع نفس الحقول التي أستعملها، لأنني أغير الحقول وفق المفهوم الذي أعرضه، على العموم تستطيع ابتداءً إنشاء جدول الموظفين بالحقول التي أوردها في استعلام الإنشاء التالي:

```
CREATE TABLE Employee
(ID INT NOT NULL PRIMARY KEY,
FullName NVARCHAR(40),
Adress NVARCHAR(255),
Age TINYINT)
```

للقيام بعملية تعديل البيانات في جدول ما، فإن الصيغة تكون كما يلي:

```
UPDATE Employee
SET FullName='UnKnown', Adress='Jeddah'
```

لمثال أعلاه يقوم بتعديل بيانات كل العناصر الموجودة داخل جدول ، Employees وبالتالي لجعل التعديل يشمل بعض العناصر دون غيرها يلزمنا تحديد شرط التعديل والذي يأتي بعد الكلمة ، WHERE في المثال الآتي سنغير معلومات الموظف الذي يحمل الرقم1 :

```
UPDATE Employee
SET FullName='UnKnown', Adress='Jeddah'
WHERE ID=1
```

وهذا مثال آخر يقوم بتغيير أسماء الموظفين القاطنين بمدينة الرياض، وتعويض أسمائهم ب "حمزة:"

```
UPDATE Employee
SET FullName='Hamza'
WHERE Adress='Riad'
```

- جرد البيانات Select:

-استخدامات جملةSELECT :

***الإسقاط (PROJECTION):**و بهذا يمكن استخلاص عمود أو أآثر من عمود من الجدول ولكن بدون امكانية تحديد صف معين.

***الاختيار(SELECTION):**وهذا يعنى انة يمكن أَن تختار مجموعة من الصفوف ذلك عن طريق اَستعمال مجموعة معايير يمكن بها تحديد الصفوف التي تراها) اختيار افقى.)

***الربط (JOIN):**أنت يمك أَن ن تستعمل الربط في SQLوهذا يعنى امكانية استخلاص بيانات من جدولين أو أآثر من جدول عن طريق إنشاء ربط بينهم باحدى الطرق المختلفة.

SELECT * | { [DISTINCT] column | expression [alias] ,...}
FROM table ;

من الشكل السابق ، جملة SELECTيمكن أَن تتضمن التالي:

- فقرة ،SELECT تحدد الأعمدة التي سوف تعرض سواءكان عمود أو أكثر.
- FROM تحدد الجداول التي تحتوي على الأعمدة التي وجدت في فقرة SELECT السابقة على النحو التاليِ:

- SELECT تعنى اختيار عمود أو أكثر من عمود.
- ( ) تعنى اختيار آل الأعمدة بدون تحديد أسمائهم.
- DISTINCTاخفاء التكرار فى بيانات العمود
- Column /expressionالعمود الذى تم العمل علية عملية حسابية بداخل جملةSELECT

- Alias الاسم المستعار يعطي الأعمدة المختارة عناوين غير أسمائها الحقيقية.

- **كيفية عمل ALIAS :**

عندما تريد تغير اسم العمود من last_name الى name كما فى المثال التالى فهذا هو ما يسمى alias وهذا يكون فى الجدول المعروض فقط و يبقى اسم العمود بالجدول داخل قاعدة البيانات كما هو . دون تغير FROM table تحدد الجداول التي تحتوي على الأعمدة

- طرق استخدام Alias :

- يمكن كتابة الاسم المستعار(ALIAS) (بعد آتابة اسم العمود الحقيقى بشرط وجود مسافة بين الاسمين وان يكون الاسم المستعار لا يحتوى على مسافة.
- يمكن كتابة الاسم المستعار(ALIAS) (بة مسافة ولكن لابد من آتابة الاسم المستعار بين double quotation

Example 1

```
Select ename NAME from "T emp" ;
```

تم وضع double quotationلان الاسم المستعار يحتوى على مسافة.

Example 2

```
Select FIRST_NAME NAMESSS ,JOB_ID
from employees
```

نلاحظ ان هنا FIRST_NAME اسمه تحول الىNAMESSS

مثال لاختيار آل الأعمدة بدون ذلكو اسمائها:

في **المثال** تم عرض آل الصفوف بالجدول وذلك لأن أمر SELECTلم يتبعه شرط تحديد عدد الصفوف.

- استخدام أوامرSELECT SQL

وهو يستعرض بيانات الجدول وتشير النجمة المستخدمة * الى إظهار جميع أعمدة الجدول ثم نكتب أمر FROMيليه أسم الجدول الذي يراد الاستعلام عن بياناته وبعد انتهاء الأمر يتم وضع;

لاحظ أن أوامر SQL تطلب لكتابة فصلة منقوطة(؛) في آخر الأمر وهنا سيتم استعراض جميع بيانات أعمدة الجدول ملحوظة أوامر SQLلا تختصر مثل لكتابة . SEL أنت يمكن أَن تعرض آل أعمدة الجدول بكتابة الكلمة الرئيسيةSELECT ويليها بنجمة(*).

من المثال السابق نجد أن جدول الاقسام يحتوي علي أربعة أعمدة وهم:

DEPARTMENT ID , DEPARTMENT NAME, MANAGER ID , LOCATION ID .

أنت يمكن أَن تعرض أيضا آل الأعمدة في الجداول بتسجيل آل الأعمدة بعد SELECT لكلمة بدلا من استخدام النجمة(*).

Example:

```
SELECT department id, department name, manager
id,location id
FROM departments ;
```

ويمكن ان نختار عدد محدد من الاعمدة فقط ذلك عن طريق لكتابة اسماء الاعمدة المراد عرضها بعد كلمة SELECTووضع coma (,) بين آل عمود كما فى المثال التالى:

- اختيار الأعمدة المحددة

وهذا يعنى اننا من الممكن عرض أعمدة محددة وليست آل الأعمدة.

ولاحظ ان ترتيب ظهور الاعمدة يكون على اساس ترتيب تلك الاعمدة فى جملة SELECT.

فالفرق بين المثالين حيث الأول كان SELECT بعدها department id وعرضت أولا يسارا في الجدول عكس المثال التالى حيث جاءت في الترتيب الثاني

SELECT location id, department id
FROM departments ;
LOCATION ID DEPARTMENT ID

- **كتابة جملSQL**

1 .جمل SQLيمكن أ َن تكون فى سطر أو أآثر من سطر أي يمكن آتابتها فى عدة سطور.

2 .هاآلمات الرئيسية لا يمكن أ َن تختصر أو تنفصل عبر السطور.

3 .فقراتها توضع عادة على سطور منفصلة لتسهيل مراجعتها.

4 .ترك مسافات بين مكونات الجملة كي تحسن القراءة.

- تنفيذ جملSQL

في ،ISQL*Plus انقر على زر التنفيذ (Execute) لكي يتم التنفيذ.

- ملاحظة

يتطلب في ISQL*PLUSاو SQL*PLUSأن تضع ;) semicolon في نهاية الفقرة الأخيرة.

•أختلاف عنوان العمود بينSQL*PLUS & ISQL*Plus:

:Sql*plus

1 .عناوين الحروف و التواريخ لرأس العمود تظهر يسارا.

2 .عناوين رأس العمود للأرقام تظهر يمينا.

3 .تعرض العناوين بحروف كبيرة.

```
SELECT ename, date, sal
FROM emp ;
ENAM DATE SAL
KING 17-JUN-87 1500
```

:ISQL*Plus

-عنوان العمود يوضع في المنتصف.

-عرض العنوان في أعلى الصفحة بحروف كبيرة وفى الوسط.

```
SELECT last name, hire date, salary
FROM employees;
```

العمليات الحسابية

العملية	الوصف
+	الإضافة
-	الطرح
*	الضرب
/	القسمة

وربما تحتاج أَن تعدل طريقة عرض البيانـات لتـؤدي حـسابات معينـة ،دون أن تـؤثر علـى البيانات المخزنة في الجدول. هذا يتم بأَستعمال العمليات الحـسابية. يمكـن أَن يحتـوي التعبير الحسابي علي أسم العمود،او قيمة ثابتة، والعوامل الحسابية مثل.(-,*,+,/).

العوامل الحسابية

يشرحُ الجدول السابق العوامل الحسابية المتوفرة في SQL.أنت يمكن أَن تستعمل العوامل الحسابية في أي فقرة SQLماعدا فى الفقرةFROM

ملاحظة:

يمكن أَن تستعمل مع التاريخ العوامل الجمع والطرحِ فقط.

استخدام العماليات الحسابية

المثال :يوضح استعمال عملية الجمع حيث يحسب زيادة للراتب 300$مع لكل الموظفون و يعرضون المرتب الجديد + 300في العمود.

ملاحظة الناتج من حساب)راتب العمود + (300ليست عمود جديد في جدول الموظفين إنما هو عمود للعرض . فقط أسم العمود الجديد يجيء من ناتج حساب + المرتب) راتب300) + وهنا يمكن استعمال(ALIAS) حتى نتمكن من تسمية تلك العمود.

الملاحظة ORACLE 10G : يهمل المسافات الفارغة قبل وبعد العوامل الحسابية اسبقية عمل العوامل الحسابية- + / * :

1 .الأولوية تكون للضرب الأول ثم القسمة ثم الجمع ثم الطرح.

2 .الأولوية من اليسار آلي اليمين.

3 .تستخدم الأقواس حتى تتمكن من تنفيذ الجمع او الطرح اولا قبل الضرب او القسمة.

أسبقية العوامل:

- إذا كان هناك اكثر من عملية مختلفة ، ضرب وقسمة وجمع و طرح. أنت يمكن أَن تستعمل الأقواس لضمان تنفيذ العمليات التى بداخل الأقواس أولاً. ثم الباقي بالترتيب من اليسار إلى اليمين.

- تستطيع أن تستخدم الأقواس آي تنفذ الذى بداخلة اولا. أي انك عندما تضعهم فى الأقواس ينفذ الاقواس اولا سواء آان جمع أو طرح حيث ان بين الأقواس ينفذ اولا.

مثال على أسبقية العوامل

أسبقية العامل :المثال السابق يعرض الاسم الأخيرء المرتب،

المرتب*.(100+12). عملية الضرب يتم تنفيذها قبل إضافة رقم 100.

ملاحظة :تستخدام الأقواس كي توضح اولوية العملية الحسابية وتسهل

تـصحيح الاخطاء.

لاحظ يمكن أَن يكتب (12* salary) +100 لا يحدث تغيير في النتيجة السابقة لكن التعبير اصبح اكثر وضوحاً.

```
SELECT EMPLOYEE ID , SALARY *15/100
FROM employees;
```

نلاحظ فى هذا المثال ان تم ضرب المرتب salary فى 15ثم القسمة /100

- استخدام الأقواس:

يمكن أَن تتجاوز قواعد الأسبقية فى تنفيذ العمليات الحسابية باستخدام الأقواس لتحدد ترتيب حساب العمليات الحسابية التي تنفذ اولا حسب احتياجاتك.

المثال السابق يقوم بعرض الاسم الأخيرِ المرتب، (المرتب+ 100)*12 هو بذلك يقوم بجمع 100على المرتب اولا كل موظف ثم بعد ذلك يقوم بضرب الناتج فى12بسبب الأقواس،يمكن لعملية الجمع ان تنفذ قبل الضرب . فبسبب الأقواس يتم حساب(salary+100) أولا ثم يضرب الناتج × .12

مثال اخر:

```
SELECT EMPLOYEE ID, 15*(SALARY +100 )
from employees;
```

نلاحظ ان عملية الجمع تمت اولا ثم عملية الضرب

-التعامل مع القيمةNull

- هي قيمة غير متاحة غير محددة غير معروفة وغير قابلة لاجراء اى عملية حسابية عليها.
- قيم اى خلية فى الجدول فارغة تكونNULL.
- لا تساوى صفر أو مسافة فارغة .حيث ان الصفر عدد، والفراغ يعتبر حرف.
- أي نوع من البيانات يمكن أَن تحتوي علىnull.

- على أية حال، يمكن وضع بعض القيود على الجدول مثل NOT NULLو PRIMARY KEY

- وهذا يعنى عدم امكانية أحتواء العمو NULL . ذلك كما سوف نرى فيما بعد.

لاحظ فى المثال السابق ان الذي لديه عموله هو مدير مبيعات أو مندوب المبيعات وباقى الموظفون ليس لديهم نسبة عمولات لذلك قيم العمولة عندهم null .

ملحوظة:

عندما تحاول أ َن تقسم أي رقم على صفر، تصبح النتيجة خطأ. وايضا عندما تحاول اجراء اى عملية حسابية على،Nullالتي تكون نتيجةNULL.

نم المثال السابق لاحظ ان (KING) لا يحصل على أي عمولة ولكنة يحصل على مرتب.ومع ذلك النتيجة اصبحت Nullفى الخلية الخاصة بة و هذا يؤكد ان اجراء اى عملية حسابية على NULLتؤدى الى NULL.

علامة اللصق (الضمConcatenation) || ( )

وهي تستخدم لكي نضم عمودين فى عمود واحد فيتم استخدام علامة || آي تلصق الأعمدة.

Example تم دمج عمود اسم الموظف مع وظيفتة فى عمود واحد وسمى Employees.

نلاحظ من المثال السابق ان الاسم الاول والثانى اصبحا معا تحت اسم name of employeeسلسلة الحروف الحرفية

1- LITERAL .تعبر عن حرف او عدد أو تاريخ و تتضمن في جملة SELECT.

2 -اى بقيمِ حرفية يجب أ َن تكون مرفقة ضمن العلامة (‘ ‘).

3 -LITERAL تكون حرف او عدد أو تاريخ.

4 -الحروف و التواريخ يوضعان بين العلامة. (‘ ‘) .

5- استخدام مجموعة الحروف الحرفية.

مثال اخر:

المثال يعرض الأسماء و الوظيفة لكل الموظفون. وعنوان العمود . Employee Details الفراغ يحسن قراءة الناتج .

عندما نريد ان نضيف كلمة او اكثر من كلمة قبل او بعد ظهور العمود لابد من:

اولا :يجب وضع الكلمات التى تريد ادراجها بين single quotation.

ثانيا :عمل concatenation مع هذة الكلمات .

ففى **المثال** التالي، تم ضم الاسم الأخير مع جملة= (: month salary1 مع الراتب لكل موظف و**لاحظ** انك لابد من وضع تلك الجملة بين single. quotation

*اذا كنت تريد وضع علامة Single Quotation داخل الجملة التى تريد ظهورها قبل او بعد او بين العمود يجب عليك وضع علامة q ووضع الجملة المراد أضافتها بين] [

```
SELECT last name ||': 1 Month salary = '||salary
Monthly
FROM employees;
```

(أي أنك عند الرغبة في وضع حروف بين عمودين يراد لصقهما فأننا نكتب هذه الحروف بين علامتين تنصيص مفردتين (|| ' .. ' ||) ولو أرقام فلا توضع بين شئ.

- كيفية ازالة الصفوف المتكررة

إنّ الاستعلام يكون لكل الصفوف بالجدول و يشمل الصفوف المتكررة.

ملحوظة

-أعداد القسم تتكرر.

- إزالة الصفوف المتكررة

- إزالة الصفوف المتكررة باستخدامDISTINCT

لكي تزيل الصفوف المتكررة يجب وضع كلمة DISTINCT قبل اسم العمود المراد ازالة التكرار منة ففى المثال السابق،جدول الموظفون الفعلي يتكون من 20صف لكنة ظهر بدون تكرارات بعد استخدام آلمةDISTINCT .

Example:

SELECT DISTINCT department id

FROM employees;

رأينا فيما سلف كيف يقوم الأمر Selectبجلب البيانات من الجداول، الآن سنتعرف عليه أكثر، وكنقطة للبداية سنورد صيغة هذا الأمر:

```
SELECT Field1, Field2, ...
FROM TableName
WHERE Condition
```

في الجزء الأول من Select نقوم بتحديد الحقول التي نريد استعراض قيمها، وفي الجزء الثاني نحدد الجدول المراد جلب البيانات منه، وفي الجزء الثالث نستطيع أن نضع شرطا يضبط جلب البيانات، كما يظهر المثال التالي:

```
SELECT ID, FullName, Adress
FROM Employee
WHERE Adress='Mekkah'
```

ستكون النتيجة كما يلي:

	ID	FullName	Adress
1	2	Hamid MAKBOUL	Mekkah
2	3	Mohamed ALQAHTANI	Mekkah
3	4	Youssef Ahmed	Mekkah

لجلب جميع الحقول، نستغني عن كتابة الأسماء ونعوضها بالرمز نجمة)*(الذي يعني كل الحقول:

```
SELECT *
FROM Employee
WHERE Adress='Mekkah'
```

الاستعلام السابق سيعطي نفس النتيجة.

بمقدورنا وضع أكثر من شرط بعد الكلمة WHERE، كما يوضح المثال التالي:

```
SELECT *
FROM Employee
WHERE Adress='Mekkah'
And Age < 30
```

هذه المرة سيجلب الموظفين القاطنين في مكة المكرمة والدين عمرهم أصغر من 30سنة.

وهذا مثال آخر يقوم بجلب الموظفين القاطنين في مكة المكرمة أو المدينة المنورة:

```
SELECT *
FROM Employee
WHERE Adress='Mekkah'
OR Adress='Madina'
```

سيقوم الاستعلام أعلاه بجلب كل الموظفين القاطنين بمكة والمدينة، والنتيجة كما يلي:

	ID	FullName	Adress	Age
1	1	Karim Hamdi	Madina	24
2	2	Hamid MAKBOUL	Mekkah	23
3	3	Mohamed ALQAHTANI	Mekkah	35
4	4	Youssef Ahmed	Mekkah	42
5	5	Nihad Chawqi	Madina	21

بإمكاننا تغيير أسماء الحقول عند العرض، ولفعل ذلك نحدد اسم الحقل المراد عرضه بعد الكلمةA:

```
SELECT ID AS الرقم,
FullName AS  [الاسم الكامل],
Adress AS العنوان,
Age AS  العمر
FROM Employee
```

والنتيجة كما يلي:

	الرقم	الاسم الكامل	العنوان	العمر
1	1	Karim Hamdi	Madina	24
2	2	Hamid MAKBOUL	Mekkah	23
3	3	Mohamed ALQAHTANI	Mekkah	35
4	4	Youssef Ahmed	Mekkah	42
5	5	Nihad Chawqi	Madina	21

- دمج الحقولConcatenation:

أحيانا نريد إظهار قيم بعض الحقول مرتبطة فيما بينها كما لو أ نها قيمة واحدة، تسمى هذه العميلة بالدمج ، Concatenation للقيام بذلك نفرق بين الحقول بعلامة +، كما يعرض المثال التالي:

```
SELECT  FullName + '   Is From:  ' + Adress as 'About this emplyee'
From Employee
```

ستكون النتيجة هكذا:

	About this emplyee
1	Karim Hamdi Is From: Madina
2	Hamid MAKBOUL Is From: Mekkah
3	Mohamed ALQAHTANI Is From: Mekkah
4	Youssef Ahmed Is From: Mekkah
5	Nihad Chawqi Is From: Madina

- **جرد الأسطر الأولىSELECT TOP:**

أحيانا نريد إظهار الأسطر الأولى فقط من جدول ما، للقيام بذلك نستخدم الكلمة TOPبعد الكلمة SELECT ثم نحدد عدد الأسطر الأولى المراد استعراضها، كما يوضح المثال التالي:

```
SELECT TOP 3 *
FROM Employee
```

نتيجة المثال أعلاه كما يلي:

	ID	FullName	Adress	Age
1	1	Karim Hamdi	Madina	24
2	2	Hamid MAKBOUL	Mekkah	23
3	3	Mohamed ALQAHTANI	Mekkah	35

كما تلاحظ قام بجلب الثلاثة الأوائل فقط.

- جلب البيانات عشوائيا RANDOM SELECT:

إذا أردنا جلب عناصر عشوائية فعلينا النداء على الدالة (NewId مصحوبة بأمرالترتيب Order By الذي سنراه لاحقا، وكمثال لذلك نورد:

```
SELECT  *
FROM Employee
ORDER BY newid()
```

في كل مرة تقوم بتنفيذ هذا الاستعلام، ستلاحظ بأن البيانات تأتي عشوائيا، قد تحتاج هذا الأمر إذا كنت بصدد إنجاز برنامج يحتاج إلى تولد اختيارات عشوائية مثل مسابقة "من سيربح المليون؟."

- جلب البيانات غير مكررةSELECT DISTINCT:

تمكننا الدالة DISTINCT من جلب البيانات من حقل معين مع تفادي التكرار، بحيث لو كان يضم الحقل نفس القيمة في أكثر من سطر، تجلب هذه القيمة مرة واحدة، وصيغة هذه الدالة كما يلي:

```
SELECT DISTINCT FullName
FROM Employee
```

لنفترض أن جدول الموظفين يضم البيانات التالية:

ID	FullName
1	Khalid
2	Ahmed
3	Khalid
4	Karim

بعد تنفيذ الاستعلام أعلاه، ستكون البيانات المطلوبة كما يلي:

FullName
Khalid
Ahmed
Karim

- **جلب البيانات المشابهة LIKE:**

لعلك تتساءل كيف تقوم محركات البحث)غوغل مثلا(بجلب نتائج مشابهة للكلمة التي تبحث عنها، فيما يلي سنتعرف على كلمة تقوم بنفس العمل، إنها الكلمة Like. هذا المثال يقوم بجلب الموظفين الذين يبدأ اسمهم بحرف "M":

```
SELECT  *
FROM Employee
WHERE FullName like 'M%'
```

لاستعلام أعلاه يعني جرد الموظفين الذين يبدأ اسمهم بحرف M ، ورمز النسبة المئوية % يعني لا يهمنا ما يأتي بعد الحرف M، وعليه فالنتيجة ستكون كما يلي:

	ID	FullName	Adress	Age
1	3	Mohamed ALQAHTANI	Mekkah	35

وهذا مثال آخر يقوم بجلب الموظفين الذين تنتهي أسماؤهم بحرف بالحرف"I" :

```
SELECT  *
FROM Employee
WHERE FullName Like '%I'
```

النتيجة كما يلي:

	ID	FullName	Adress	Age
1	1	Karim Hamdi	Madina	24
2	3	Mohamed ALQAHTANI	Mekkah	35
3	5	Nihad Chawqi	Madina	21

وهنا مثال لجلب الموظفين الذين تضم أسماؤهم حرف"N" :

```
SELECT  *
FROM Employee
WHERE FullName Like '%N%'
```

- **ترتيب البياناتORDER BY:**

نريد جرد الموظفين مرتبين حسب أعمارهم، من الأكبر إلى الأصغر، للقيام بذلك نستخدم الكلمة ORDER BYالتي تمكننا من ترتيب البيانات إما تصاعديا أو تنازليا:

```
SELECT  *
FROM Employee
ORDER BY Age DESC
```

الاستعلام أعلاه يقوم بجرد الموظفين مرتبين وفق أعمارهم تنازليا من الأكبر إلى الأصغر، والنتيجة كما يلي:

	ID	FullName	Adress	Age
1	4	Youssef Ahmed	Mekkah	42
2	3	Mohamed ALQAHTANI	Mekkah	35
3	1	Karim Hamdi	Madina	24
4	2	Hamid MAKBOUL	Mekkah	23
5	5	Nihad Chawqi	Madina	21

ويمكننا القيام نفس العملية تصاعديا، أي من الأصغر إلى الأكبر فقط بتبديل الكلمة DESC (وهي اختصار للكلمة descendant)بالكلمة ASC (وهي اختصار للكلمة ascendant).وهذا مثال للتوضيح:

```
SELECT  *
FROM Employee
ORDER BY Age ASC
```

ويمكننا القيام بالترتيب المتعدد عن طريق تحديد العديد من الحقول بعد الكلمة BY،ORDER كما يعرض هذا المثال:

```
SELECT  *
FROM Employee
ORDER BY Age, Adress
```

الفصل الخامس : الدوال Functions

الفصل الخامس : الدوال Functions:

الدوال عبارة عن مجموعة من البرامج المنجزة مسبقا، والتي تتيح للمستخدم بعض الخدمات التي تغنيه عن كتابتها بواسطة الأكواد، من قبيل الدوال الرياضية (المجموع Sum، المتوسط Average،..) التي قد يحتاجها المستخدم . توفر لغة SQL مجموعة كبيرة من الدوال لإنجاز بعض المهام التي يحتاجها المستخدم، في هذا الجزء سنورد بعض الدوال الأكثر شيوعا والتي قد تحتاجها مستقبلا.

- **الدوال التجميعية :Aggregate Functions**

- الدالة COUNT:

وتعيد لنا هذه الدالة عدد عناصر جدول معين، وصيغتها هكذا:

```
SELECT COUNT(*)
FROM Employee
```

لنتيجة عبارة عن قيمة رقمية تمثل عدد العناصر التي يجلبها الاستعلام لتعطينا عدد القيم في العمود او الاعمدة و في حال استخدام DISTINCT تصبح عدد القيم المختلفة.

الدالة SUM:

١ -تعيد لنا الدالة،SUM قيمة تمثل مجموع قيم الحقل الرقمي المحدد، وفي حالة استخدام DISTINCT تصبح مجموع القيم المختلفة، فمثلا لو افترضنا أنه لدينا جدول الموظفين التالي:

ID	Name	Job	Hours
1	Mohamed	Developper	56
2	Hamid	Web Master	45
3	Younes	Conceptor	78
4	Khalid	Designer	84

لو أردنا معرفة مجموع الساعات التي اشتغلها هؤلاء الموظفون، فعلينا استعمال الدالة SUM كما يلي:

```
SELECT SUM(Hours)
FROM Employee
```

لابد أن يكون نوع الحقل الذي نطبق عليه الدالة SUMمن نوع رقمي .

- الدالة AVG :

وتقوم هذه الدالة بحساب متوسط الحقل المحدد، وكما هو معلوم في الرياضيات فالمتوسط يساوي مجموع قيم العناصر مقسوما على عدد العناصر .

صيغة الدالة AVG كما يلي:

```
SELECT AVG(Hours)
FROM Employee
```

النتيجة ستكون عن عبارة قيمة رقمية تمثل مجموع الساعات مقسوم على عدد الموظفين، تطبق هذه الدالة فقط على الحقول الرقمية .

بإمكاننا الحصول على نفس النتيجة من غير استخدام الدالة AVG، وذلك كما يلي:

```
SELECT SUM(Hours)/ COUNT(Hours)
FROM Employee
```

لأن المتوسط يساوي المجموع مقسوم على عدد العناصر.

- الدالة MIN :

وتعيد لنا هذه الدالة أصغر قيمة في الحقل المحدد، وصيغتها كما يلي:

```
SELECT MIN(Hours)
FROM Employee
```

- الدالة MAX :

وتعيد لنا هذه الدالة أكبر قيمة في الحقل المحدد، وصيغتها كما يلي:

```
SELECT MAX(Hours)
FROM Employee
```

- **تجميع البيانات GROUP BY:**

تستعمل الكلمة GROUP BY مع إحدى الدوال التجميعية ، Aggregate functions وتمكننا من تجميع البيانات وفق حقل معين، مثلا تحديد عدد الموظفين القاطنين في كل مدينة، أو عدد التلاميذ الذين يدرسون في كل فصل...إلخ.

وصيغتها كما يلي:

```
SELECT
COUNT(Adress) as 'عدد سكان كل مدينة'
FROM Employee
GROUP BY Adress
```

وهذا مثال توضيحي لاستعمال الكلمة GROUP BY:

```
SELECT Age as 'السن',
COUNT(Age) as 'عدد الموظفين الذين لهم هذا السن'
FROM Employee
GROUP BY Age
```

بعد تنفيذ هذا الاستعلام، سنحصل على حقلين، الأول يعرض كل الأعمار الموجودة في جدول الموظفين، والحقل الثاني يعرض عدد الموظفين الذين يبلغون كل عمر، وعليه فالنتيجة ستكون هكذا:

	السن	عدد الموظفين الذين لهم هذا السن
1	21	2
2	23	1
3	24	3

- **شرط التجميع HAVING**:

تستخدم الكلمة HAVINGبعد الكلمة، GROUP BY لتحديد شرط جلب البيانات، مثلا بعد أن تقوم GROUP BYبتجميع الموظفين حسب المدن، تقوم HAVING بجلب المجموعات القاطنة في مدينة الرياض مثلا.

صيغتها كما يلي:

```
SELECT Adress,
COUNT(Adress)
FROM Employee
GROUP BY Adress
HAVING adress='Riad'
```

الاستعلام أعلاه يقوم بجلب المدن، وأمامها عدد الموظفين القاطنين بها، لكن بعد أن كتبنا HAVING سيقوم بجلب سكان مدينة الرياض فقط .

في المثال التالي، سنجلب كل الأعمار، وأمامها عدد الموظفين البالغين لها، ثم نقوم بالفرز بواسطة، HAVING لنجلب فقط الأعمار التي يفوق عدد بالغيها أربعة أشخاص:

```
SELECT age,
COUNT(Age )
FROM Employee
GROUP BY age
HAVING COUNT(age)>4
```

قم بتجريب المثال وتتعرف على الكلمة HAVING أكثر.

- **الدوال الحسابية :Arithmetic Functions**

- الدالة ABS :

الدالة ABS تعيد لنا القيمة المطلقة Absolute Value للحقل المحدد، وصيغتها كما يلي:

```
SELECT ABS(-67)
```

هذا المثال سنحصل على القيمة المطلقة للرقم ،67-وهي67.

- الدالة SQRT

تعيد لنا هذه الدالة قيمة الجذر المربع للرقم المحدد، وصيغتها مثل الدوال السالفة.

يوجد المزيد من الدوال الرياضية، ما أوردناه ليس كل شيء.

- **الدوال النصيةString Functions:**

- الدالة: SUBSTRING

تعيد لنا هذه الدالة جزء من النص أو الحقل المحدد، وصيغتها العامة كما يلي:

```
SUBSTRING(طول الاجتزاء,بداية الاجتزاء,النص أو الحقل المراد اجتزاؤه)
```

المعامل الأول" النص أو الحقل المراد اجتزاؤه "نعوضه باسم الحقل الذي نريد اقتطاع النص منه.

المعامل الثاني" بداية الاجتزاء " نعوضه برقم الحرف الذي نريد ابتداء التقطيع انطلاقا منه.

المعامل الثالث" طول الاجتزاء " نعوضه بعدد الأحرف المراد تقطيعها انطلاقا من بداية الاجتزاء.

وهذا المثال يعرض كيف نستخدم الدالة Substring مع حقل نصي:

```
SELECT FullName AS 'الاسم الكامل',
SUBSTRING(FullName, 3,5) AS 'جزء من الاسم'
FROM Employee
```

في هذا المثال سنحصل على نتيجة متكونة من حقلين، أحدهما يعرض الاسم كاملا، والآخر يعرض خمسة أحرف من الاسم انطلاقا من الحرف الثاني، وعليه فإن النتيجة كما يلي :

	الاسم الكامل	جزء من الاسم
1	Karim Hamdi	rim H
2	Hamid MAKBOUL	mid M
3	Mohamed ALQAHTANI	hamed
4	Youssef Ahmed	ussef
5	Nihad Chawqi	had C

- الدالة: LEFT:

تقوم هذه الدالة باقتطاع النص انطلاقا من اليسار وانتهاء بقيمة طول الاجتزاء، وصيغتها كما يلي:

```
LEFT(الحقل أو النص المراد اجتزاؤه, طول الاجتزاء)
```

سنطبقها على نفس المثال السابق:

```
SELECT FullName AS 'الاسم الكامل',
RIGHT(FullName, 5) AS 'جزء من الاسم'
FROM Employee
```

هذه المرة ستكون النتيجة عبارة عن حقلين، أولهما يعرض الاسم كاملا، والثاني يعرض جزء من الاسم طوله خمسة أحرف ابتداء من اليسار:

	الاسم الكامل	جزء من الاسم
1	Karim Hamdi	mdi
2	Hamid MAKBOUL	OUL
3	Mohamed ALQAHTANI	ANI
4	Youssef Ahmed	med
5	Nihad Chawqi	wqi

- الدالة RIGHT:

تقوم بنفس دور الدالة السابقة، ولكن انطلاقا من اليمين، بالنسبة لصيغتها فهي أيضا هكذا :

```
RIGHT(النص المراد اجتزاؤه,الحقل أو طول الاجتزاء)
```

- الدالتان : LTRIM و RTRIM:

تقوم الدالة RTRIM بمسح الفراغات عن يمين النص أو الحقل

وبالمقابل تقوم الدالة LTRIM بمسح الفراغات عن يسار النص أو الحقل وصيغتهما كما يلي

```
-- Remove   Blanks From The right
SELECT RTRIM('Khalid      ');

-- Remove Blanks From the left
SELECT LTRIM('      Khalid');
```

في الحالتين معا ستكون النتيجة عبارة عن الكلمة"Khalid" من دون فراغات.

-الدالتان UPPER و LOWER

تقوم هاتان الدالتان بتغيير حالة النصوص، بحيث تمكننا UPPER من تحويل أحرف النص إلى أحرف Upper case ، وبالمقابل تمكننا الدالة LOWER، من تحويل حالة الأحرف إلى حالتها Lower caseكبيرة ، وصيغة الدالتين الصغيرة كما يلي:

```
-- Change Case To Upper case
SELECT UPPER('Khalid') AS 'UPPER Case';

-- Change Case To Lower case
SELECT LOWER('Khalid') AS 'LOWER Case';
```

نتيجة الاستعلام الأول ستكون هكذا:

	UPPER Case
1	KHALID

بينما ستكون نتيجة الاستعلام الثاني هكذا:

	LOWER Case
1	khalid

-الدالة CHARINDEX:

تعيد لنا هذه الدالة رتبة الحرف أو النص المحدد، وصيغتها كما يلي:

```
SELECT CHARINDEX('النص الكامل','الحرف أو النص المراد البحث عن رتبته');
```

وهذا مثال لاستعمال الدالة CHARINDEX

```
SELECT FullName,
CHARINDEX('d',FullName) AS 'Position Of "d"'
FROM Employee
```

في كل الأسماء، ستكون نتيجة الاستعلام dيقوم هذا المثال بالبحث عن رتبة الحرف على هذا الشكل:

	FullName	Position Of "d"
1	Karim Hamdi	10
2	Hamid MAKBOUL	5
3	Mohamed ALQAHTANI	7
4	Youssef Ahmed	13
5	Nihad Chawqi	5

- الدالة LEN:

تعيد لنا هذه الدالة طول النص أو الحقل المحدد، وصيغتها كما يلي:

```
SELECT LEN('Khalid');
```

نتيجة هي عدد الأحرف المكونة للكلمة Khalid ، وهذا مثال للتوضيح:

```
SELECT FullName,
LEN(FullName) AS 'Length Of FullName'
FROM Employee
```

في هذا المثال ستكون النتيجة كما يلي:

	FullName	Length Of FullName
1	Karim Hamdi	11
2	Hamid MAKBOUL	13
3	Mohamed ALQAHTANI	17
4	Youssef Ahmed	13
5	Nihad Chawqi	12

- **دوال التاريخ :Date Functions**

قبل أن نبدأ استعراض دوال التاريخ، سنقوم بإضافة Date Of Working الي Employee، نوع هذا الحقل هو Date Time، يمكنك إضافته يدويا من الجدول خلال نافذة إضافة الحقول، أو عبر تنفيذ الأمر التالي:

```
ALTER TABLE Employee
Add Date_Of_Working DateTime
```

-الدالة DATEADD:

الدالة DATEADD تقوم بإضافة قيمة رقمية إلى جزء من حقل من نوع تاريخ، كأضافة بعض الأيام أو الشهور أو السنوات إلى تاريخ معين، وصيغة الدالة كما يلي:

- كما يوجد مجموعة من المعاملات المستخدمة في دوال التاريخ أهما :

المعامل الأول: Part of date يأخذ إحدى القيم الثلاثة:

- Year: لإجراء الإضافة على الجزء الخاص بالسنوات
- Month: لإجراء الإضافة على الجزء الخاص بالشهور
- Day: لإجراء الإضافة على الجزء الخاص بالأيام

المعامل الثاني numbe يأخذ القيمة الرقمية المراد إضافتها

المعامل الثالث date هو الحقل أو القيمة التاريخية المراد إجراء الإضافة إليها

وهذا مثال لاستعمال الدالة DATEADD على حقل من نوع تاريخي:

```
SELECT FullName,
Date_of_working as 'Date of working',
DATEADD(year,3,Date_of_working) as 'Using DATADD'
FROM Employee
```

الاستعلام أعلاه، يقوم بجلب أسماء الموظفين وتواريخ توظيفهم، بالإضافة إلى حقل ثالث يحتوي على تاريخ اشتغالهم زائد ثلاث سنوات، وعليه فإن النتيجة ستكون كما يلي:

	FullName	Date of working	Using DATADD
1	Karim Hamdi	2001-01-01 00:00:00.000	2004-01-01 00:00:00.000
2	Hamid MAKBOUL	2003-12-13 00:00:00.000	2006-12-13 00:00:00.000
3	Mohamed ALQAHTANI	1998-08-23 00:00:00.000	2001-08-23 00:00:00.000
4	Youssef Ahmed	2006-09-06 00:00:00.000	2009-09-06 00:00:00.000
5	Nihad Chawqi	2000-07-14 00:00:00.000	2003-07-14 00:00:00.000

بنفس الطريقة تستطيع إضافة الأيام والأشهر.

-الدالة DATEDIFF

هذه الدالة تقوم بطرح تاريخ من تاريخ آخر، بحيث تكون القيمة الناتجة هي فارق التاريخين سواء بالأيام أو الأشهر أو السنوات، وصيغتها كما يلي:

```
DATEDIFF(Part_of_date, date1, date2)
```

المعامل الأول خاص بتحديد جزء التاريخ المراد إنجاز عملية الطرح عليه، والمعاملان الثاني والثالث هما الحقلان أو القيمتان التاريخيتان اللتان ستكونان طرفي عملية الطرح .

وهذا المثال يبين كيفية استخدام الدالة DATEDIF:

```
SELECT DATEDIFF(Month, '12/01/2012', '12/08/2012')
```

النتيجة ستكون هي خارج عملية طرح التاريخ الأول من التاريخ الثاني بالأشهر أي أن لنتيجة ستساوي 7.

-الدالة DATEPART:

هذه الدالة تعيد لنا جزء من قيمة تاريخية أو حقل من نوع التاريخ، وصيغتها كما يلي:

```
DATEPART(part_of_date, date)
```

المعامل الأول هو جزء التاريخ المراد استخراجه، والمعامل الثاني هو التاريخ المراد استخراج الجزء منه.

وهذا مثال لاستعمال الدالة:

```
SELECT date_of_working AS 'Date of working',
DATEPART(YEAR, date_of_working) AS 'year',
DATEPART(MONTH, date_of_working) AS 'month',
DATEPART(DAY, date_of_working) AS 'day'
FROM Employee
```

هذا المثال يقوم بعرض التاريخ كاملا، ثم يظهر أمامه تقطيع التاريخ على شكل أيام وشهور وسنوات، وهذه صورة للنتيجة المحصل عليها:

	Date of working	year	month	day
1	2001-01-01 00:00:00.000	2001	1	1
2	2003-12-13 00:00:00.000	2003	12	13
3	1998-08-23 00:00:00.000	1998	8	23
4	2006-09-06 00:00:00.000	2006	9	6
5	2000-07-14 00:00:00.000	2000	7	14

لكن على وجه DATEPART ويوجد دوال أخرى تقوم بنفس دور الدالة التخصيص، بحيث نجد:

-الدالة MONTH : تقوم باجتزاء الشهر من حقل أو قيمة تاريخية

-الدالة DAY : تقوم باجتزاء اليوم من حقل أو قيمة تاريخية

-**الدالة YEAR** : تقوم باجتزاء السنة من حقل أو قيمة تاريخية

وهذا مثال جامع للدوال الثلاثة الأخيرة:

```
SELECT      MONTH(date_of_working),
            DAY(date_of_working),
            YEAR(date_of_working)
FROM Employee
```

- **الدالة : GETDATE:**

تعيد لنا هذه الدالة التاريخ الحالي من نظام التشغيل، وصيغتها كما يلي:

```
SELECT  GETDATE() as 'Current Date & time'
```

النتيجة المطبوعة ستكون كما يلي:

	Current Date & time
1	2013-01-25 23:09:38.270

ولنختم مع الدوال التاريخية، سنورد مثالا أخيرا نستعمل فيه الدالة GETDATE والدالة DATEDIFF:

```
SELECT date_of_working as 'Employment date',
      getdate() as 'current date',
      datediff(year, date_of_working, getdate()) as 'Employment duration'
FROM Employee
```

في هذا المثال ستضم النتيجة ثلاثة حقول، الحقل الأول يعرض تاريخ بداية الاشتغال Employment date، والحقل الثاني سيعرض التاريخ الحالي ، Current date والحقل الثالث يعرض عدد السنوات التي قضاها الموظفون في الشغل انطلاقا من تاريخ بداية الاشتغال وانتهاءً بالتاريخ الحالي، ستكون النتيجة كما يلي:

	Employment date	current date	Employment duration
1	2001-01-01 00:00:00.000	2013-01-25 23:17:40.910	12
2	2003-12-13 00:00:00.000	2013-01-25 23:17:40.910	10
3	1998-08-23 00:00:00.000	2013-01-25 23:17:40.910	15
4	2006-09-06 00:00:00.000	2013-01-25 23:17:40.910	7
5	2000-07-14 00:00:00.000	2013-01-25 23:17:40.910	13

- **دوال التحويل Conversion Functions:**

التحويل بين أنواع البيانات من أهم العمليات التي قد تحتاجها مستقبلا، ولهذا سنورد فيما يلي بعض الدوال لتحويل البيانات من نوع إلى نوع آخر.

- الدالة STR:

تقوم هذه الدالة بتحويل القيم الرقمية إلى قيم نصية، وهذا مثال لاستعمالها:

```
SELECT 'The age of employee '+Fullname +' is: '+ STR(age) as 'Info'
FROM Employee
```

ستكون النتيجة كما يلي:

	Info
1	The age of employee Karim Hamdi is: 24
2	The age of employee Hamid MAKBOUL is: 23
3	The age of employee Mohamed ALQAHTANI is: 35
4	The age of employee Youssef Ahmed is: 42
5	The age of employee Nihad Chawqi is: 21

لو ألغينا دالة التحويل STR التي تقوم بتحويل قيمة العمر Ageالرقمية إلى قيمة نصية، سوف نحصل على خطأ لأنه غير مسموح بدمج النصوص مع الأرقام إلا بعد القيام بعملية التحويل.

- **الدالة CONVERT** :

من مزايا هذه الدالة أنها ليست حكرا على نوع معين، بل من خلالها تستطيع تحويل أي نوع إلى نوع آخر، وهذه صيغتها:

```
CONVERT(Data_Type, value_or_column_to_convert)
```

في المعامل الأول نضع نوع البيانات المراد التحويل إليه، وفي المعامل الثاني نضع الحقل أو القيمة المراد تحويل نوعها، وهذا المثال يعرض كيفية استخدام دالة التحويل CONVER:

```
SELECT CONVERT(int, '12') AS 'String To Int',
CONVERT(varchar, 56) AS 'Int To String'
```

لسطر الأول يقوم بالتحويل من القيمة الرقمية إلى قيمة نصية، والسطر الثاني يقوم بالعكس اعتمادا على الدالة CONVERT.

-الدالة CAST:

دورها شبيه تماما بالدالة CONVERT وصيغتها كما يلي:

```
CAST(value_or_column_to_convert as Data_Type)
```

بين القوسين نضع القيمة أو الحقل المراد تحويل نوعه، متبوعا بالكلمة AS ثم بنوع البيانات المراد التحويل إليه، وهذا مثال يوضح كيفية استخدام دالة التحويلCAST:

```
SELECT CAST(age AS CHAR(2)) FROM Employee
```

الفصل السادس: كيفية تصميم قاعدة بيانات بلغة الإستعلامSQL

الفصل السادس: كيفية تصميم قاعدة بيانات بلغة الإستعلامSQL

لعمل إي برنامج يجب توافر ما يلي:

System analysis $ design

تحليل للنظام	1- System analysis
تصميم للنظام	2- System design
برمجة	3- coding (programming)
إختبار	4- tesing

1- تحليل النظام system analysis

وهنا يقوم المحلل(analysis) بسؤال العميل عدة أسئلة يستطيع منها الوصول إلي طلبه أو تحديد مايريد، وبعد ذلك يسلم المحلل (analysis) متطلبات العميل إلي المصمم لعمل تصميم designالبرنامج.

2- تصميم النظام system design

ويقوم بهذه الوظيفة المصمم designer، ويقوم هنا بتصميم البرنامج علي الورق فقط، قاعدة البيانات ، تقارير ، الشاشات ،.... ، وعندما ينتهي من ذلك يسلم العمل إلي المبرمج حتي ينفذ علي الجهاز.

3- البرمجة"coding "programming

ويقوم بها المبرمج "programmer" حيث أنه ينفذ كل ما قام به المصمم ولكن علي الكمبيوتر بإستخدام إحدي لغات البرمجة، وعندما ينتهي

المبرمج لابد من تجريبة ما قام به حتي يتأكد من مدي نجاح البرنامج وأنه كما يريد العميل ويقوم بذلك "tester"

ملحوظة كل العمليات السابقة يقوم بها عدة أشخاص مرتبون حسب الخبرة وهم :

الأقل خبرة Tester

الأعلي خبرة Analyst

- أنواع البرامج:

1- برامج تحتاج إلي قواعد بيانات DB مثل

"Arc gis , Auto cad"

2- برامج لا تحتاج إلي قواعد بيانات DB مثل :

"paint, word , calculater" فهي مجرد application

- برامج تحتاج إلي قواعد بيانات تتكون من:

1- قاعدة بيانات data base تحتوي علي جمع البيانات

2-البرامج applicationوهي الشاشات التي تستخدم للإدخال البيانات

3-تقارير Reports تحتوي علي البيانات التي تصل الي المستخدم

- عندما نريد عمل db هناك عدة برامج هي:

1- Access

2- Sql server

3-Oracle

- عندما نريد عملapplication

1- vaguely . Basic

2-Java

- تقارير Reports

خطوات تصميم برنامج مكتبة:

١)تحديد النقاط الأساسية.

٢)تحديد النقاط الفرعية.

٣)تحديد نوع وحجم كل نقطة فرعية.

٤)تحديد الفتاح الأساسي لكل نقطة رئيسية إن وجد.

٥)تحديد أنواع العلاقات.

٦) تحديد النقاط الفرعية التي سيتم إختيارها وهل ثابتة أم تحتاج لجدول إدخال.

٧)إنشاء جداول للنقاط التي سيتم إختيارها.

٨)تصميم شاشات إدخال البيانات.

٩)تصميم شاشات البحث أو الإستعلام.

١٠) تصميم التقارير.

١١) تحديد متطلبات البرنامج.

وفيما يلي شرح مفصل لأهم عناصر كل محور من المحاور الحادية عشر السابقة :

1- تحديد النقاط الأساسية: وهي الجداول نفسها.

الكتب 1- BOOKS

الأعضاء 2-MEMBERS

الاستعارة 3-BORROW

2- تحديد النقاط الفرعية: وهي الحقول FIELDS التي توجد داخل الجداول.

الكتب - BOOKS

١) BCODE كود الكتاب.

٢) BTITLE عنوان الكتاب.

٣) BAUTHOR الناشر .

٤) BDATE تاريخ النشر .

٥) BSUBJECT نوع الكتاب .

٦) BVOLUME الاصدار .

٧) BENTRY DATE تاريخ الإدخال.

المشتركين - MEMBERS

١) MCODE كود العضو.

٢) MNAME اسم العضو.

٣) MBRITHDAY تاريخ الميلاد.

٤) MTEL تليفون العضو.

٥) MID رقم بطاقة العضو.

٦) MADDRESS عنوان العضو.

٧) MEMBER DATE تاريخ العضوية.

٨) MMOBILE موبايل العضو.

٩) MNATIONALITY جنسية العضو إذا كان هناك اشتراك للأجانب.

- BORROW الإستعارة

١) MCODE كود العضو.

٢) BCODE كود الكتاب.

٣) BDATE تاريخ الاستعارة.

٤) RDATE تاريخ الاستلام.

3- تحديد نوع وحجم كل نقطة فرعية:

ونلاحظ قبل تحديد نوع البيانات لابد وأن نعرف أننا لا يجب أن نختار نوع " INTEGR" إلا في الضرورة القصوي عندما يتم علي هذا الحقل عليات حسابية أو ترتيب الأرقام ، وفي هذه الحالة فقط يمكن إستخدام "INTEGR " ، وذلك لأنها تسحب مساحة كبيرة من ال "RAM" ولهذا علينا أن نستخدم " CHAR"لأنه لا يعني فقط حروف وإنما هي كل ما يوجد علي KEY BOARDولكن سوف تعامل عاملة الحروف ولا تقبل القيام بالترتيب أو أي عمليات حسابية، وهنا تكون الأنواع كالتالي:

NAME	TYPE	SIZE
B CODE	Var char	20
B TITLE	N var char	100
B AUTHOR	N var char	100
B DATE	Date& time	-
B SUBJECT	N var char	50
B ENTRY DATE	Date and time	-
B VOLUME	N var char	50

NAME	TYPE	SIZE
M CODE	var char	10
M NAME	N var char	MAX
M BRITHDAY	Date and time	-
MADDRESS	N var char	MAX
M TEL	var char	10
M MOBILE	var char	10
M ID	var char	14
M EMBER DATE	Date and time	-
M NATIONALITY	var char	15

NAME	TYPE	SIZE
M CODE	var char	10
B CODE	var char	10
B DATE	Date and time	-
R DATE	Date and time	-

4- تحديد المفتاح الأساسي لكل نقطة رئيسية إن وجد:

1- في جدول BOOK B CODE PK

2- في جدول MEMBERS M CODE PK

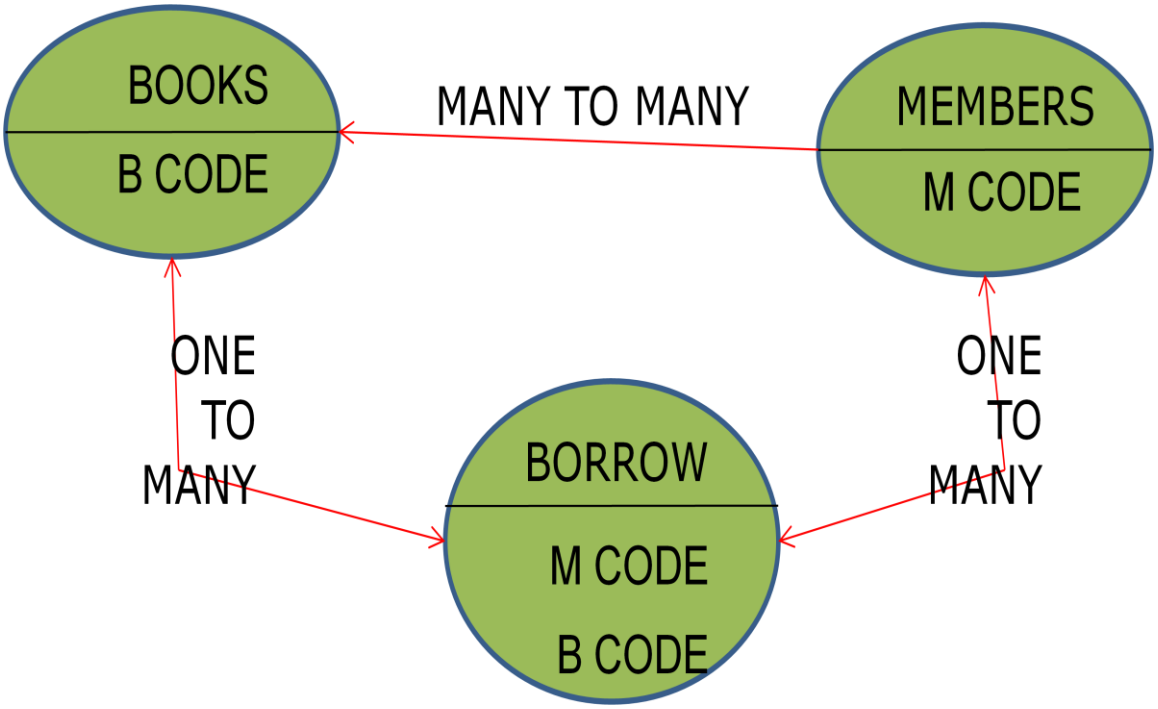
5- تحديد أنواع العلاقات :

1- بين BOOK و BORROW : العلاقة ONE TO MANY

2- بين MEMBERS و BORROW: ONE TO MANY

3- بين BOOKSو MEMBERS : MANY TO MANY

وهذه العلاقة تظهر في حالة ظهور جدول ثالث مثل BORROW



6- تحديد النقاط الفرعية التي سيتم اختيارها وهل ثابتة أم تحتاج لجدول إدخال.

يلاحظ أنه كلما قام المستخدم بالإختيار وليس بالكتابة كان ذلك أفضل وذلك لتقليل الأخطاء ، في حالة البيانات التي يمكن إختيارها مثل (المدن ، تاريخ الميلاد، البلاد، النوع، الديانة، فصيلة الدم) أما البيانات التي يصعب حصرها مثل الأسماء تترك للمستخدم يقوم بعملية الإدخال " الكتابة" .

وفي الغالب ستكون الأشياء أو البيانات التي يمكن إختيارها هي البيانات التي يستعلم عنها ولهذا لا بد أن يوجد الأسلوب الذي ندخل به هذه البيانات ونبحث عنها.

- **تابع مثال المكتبة :**

ونلاحظ في هذا المثال أن اسم الكتاب سوف يتم إدخاله من قبل المستخدم عكس أسم المؤلف الذي يتم إختياره.

١) ENTER B CODE

٢) ENTER B TITLE

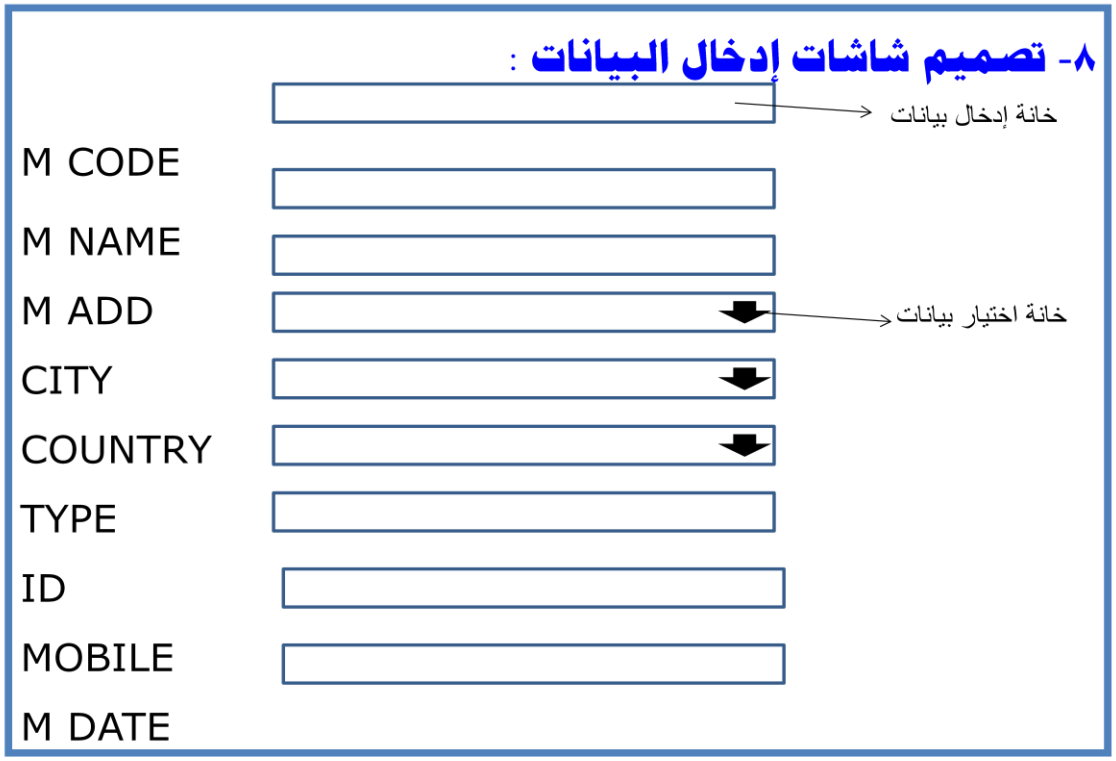
٣) CHOICE	B AUTHOR
٤) ENTER	B DATE
٥) CHOICE	B SUBJECT
٦) ENTER	B VOLUME
٧) ENTER	MEMBERS CODE
٨) ENTER	M NAME
٩) CHOICE	M ADD
١٠) CHOICE	CITY
١١) CHOICE	COUNTRY
١٢) ENTER	ID
١٣) ENTER	MOBILE
١٤) ENTER	MEMBER DATE
١٥) CHOICE	TYPE

تم تحديد 6 نقاط سوف تكون ثابتة يتم اختيارها،ولكن نلاحظ أن هذه النقاط لن تكون كلها ثابتة فثلا في المدينة سوف تكون احد البيانات التي يتم إختيارها ولكن السؤال ماذا نفعل في حالة ظهور مدينة جديدة.

7- إنشاء جداول للنقاط التي سيتم اختيارها:

وهنا يتم إنشاء جداول جديدة للنقاط التي سوف يتم اختيارها فمثلاً المدينة ليست ثابتة وهنا ينشأ جدول جديد في حالة ظهور مدينة جديدة يقوم المستخدم بإدخال أسم هذه المدينة مرة واحدة فقط وسوف تضاف مباشرة في الجدول

وعليها سوف يتم إنشاء جداول أخري لكل من المؤلف والدولة و subject وحتي هنا انتهت مرحلة db وكل ما يخصها وبدأت مرحلة تصميم الشاشات.



وعلينا أن نلاحظ أنه لابد من إنشاء شاشات إدخال لكل جدول حتي الجداول التي تم إنشاءها لانها بيانات سيتم إختيارها وغير ثابتة:

CITY

نحتاج في شاشة إدخال البيانات الي مجموعة من الإيقونات تساعد في عملية الإدخال مثل :

1- SAVE
2- CANCEL
3- EDIT
4- DELET

5- EXIT

6- PRINT

7- SEARCH

وفيما مضي كانت تستخدم بعض الإيقونات مثل :

1- FIRST

2- PREVIOUS

3- NEXT

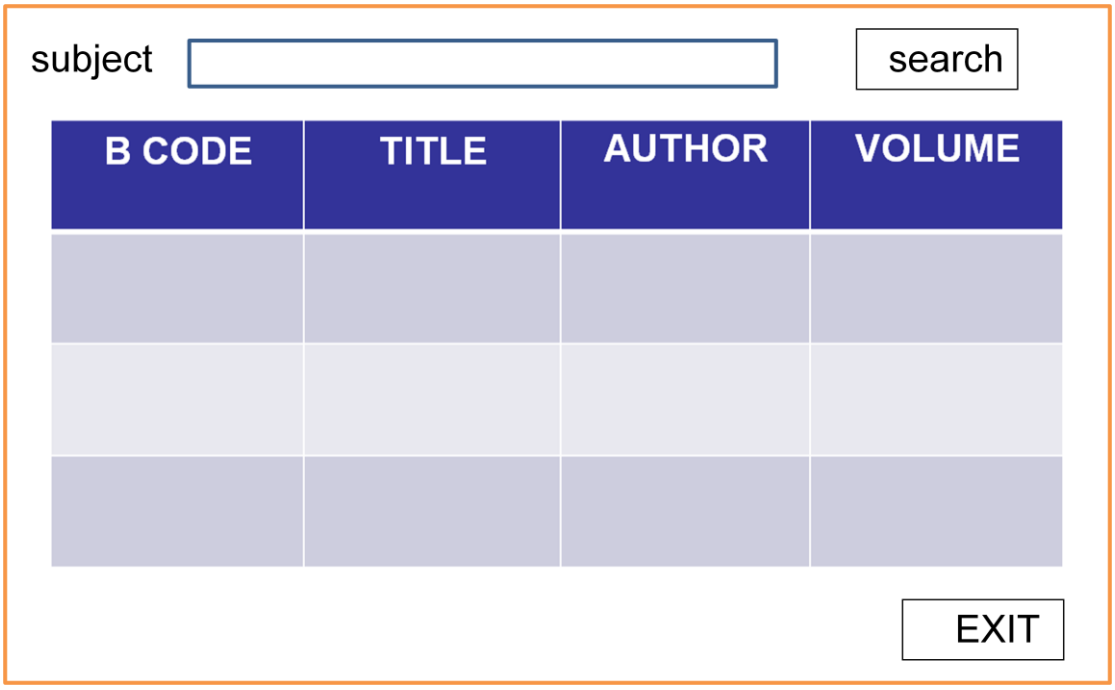
4- LAST

ولكنها أصبحت قليلة الاستخدام حاليا .

ونلاحظ أن هذه الأيقونات هي التي تستخدم في شاشات DB عند عملية الإدخال ولكن ليس شرط أن نستخدم كل هذه الإيقونات ولكن حسب احتياج البرنامج والبيانات.

9- تصميم شاشات البحث أو الإستعلام:

فمثلا إذا اردنا البحث عن كتاب معين:



10- تصميم التقارير:

ما الفرق بين التقرير والبحث

التقرير	البحث
يمكن طباعته لانه يصمم علي شكل ورقة PRINTER	لا يمكن طباعته
يحتوي علي PROCESSING أو معالجة علي البيانات	لا يحتوي علي تصنيف ولكنه يبحث عن البيانات كما هي

ونلاحظ أن الشاشة الخاصة بالتقرير هي شاشة وتقرير معاً: ودائماً يصمم التقارير علي صورة جدول

فمثلاً: نريد تقارير عن الكتب التي تتم استعارتها ولم تعود حتى الآن:

B CODE	TITLE	M CODE	M NAME	BORROWDAT	TEL	ADD

11- تحديد متطلبات البرنامج:

1- نحتاج PRINTER أم لا، أبيض وأسود أم آلوان.

2- نحتاج شبكة أم لا.

3- نحتاج SERVER أم لا.

4 - التعامل بنظام BAR COD.

5- نحتاج CAMERA للتصوير أم لا.

خطوات تصميم برنامج سوبر ماركت:

1- تحديد النقاط الأساسية: وهي الجداول نفسها.

البضائع	1- Items
الموردين	2-Suppliers
المشتريات	3-Buy
المبيعات	4-Sales
المدفوعات للتصليح	5-Pales
مدفوعات الموردين للبضائع	6-Pales-Supp
المشتريات	7-Buy

2- تحديد النقاط الفرعية: وهي الحقول FIELDSالتي توجد داخل الجداول.

البضائع - Items

١) I CODE كود الصنف.

٢) I NAME أسم الصنف.

٣) I PRICE السعر .

٤) QTY الكمية.

٥) TYPE النوع.

٦) UNIT الوحدة.

٧) LIMIT الحد الذي عند الوصول الصنف اليه يعطي البرنامج تنبيه للمستخدم.

الموردين - SUPPLIERS

١) S CODE كود المورد.

٢) S NAME اسم المورد.

٣) S ADD إضافة.

٤) S TEL تليفون المورد.

٥) S CITY مدينة المورد.

٦) S MOBILE موبايل المورد.

٧) ACCOUNT حساب المورد.

المشتريات - BUY

١) B DATE تاريخ الشراء.

٢) I CODE كود السلع.

٣) QTY الكمية.

٤) R PRICE سعر الشراء.

٥) S PRICE سعر البيع.

٦) S CODE كود المورد.

٧) INV- NO رقم الفاتورة.

- SALES المبيعات

١) S DATE تاريخ البيع.

٢) I CODE كود السلع.

٣) QTY الكمية.

٤) PRICE السعر.

٥) REST- NO الفاتورة.

3- تحديد نوع وحجم كل نقطة فرعية:

- ITEMS البضائع

١) VAR CHAR(20) B CODE

٢) VAR CHAR(100) B NAME

٣) CHOICE QTY

٤) NUMERIC (9,3) B DATE

٥) NUMERIC (6,2) PRICE

٦) CHOICE VAR CHAR(20) TYPE

٧) CHOICE VAR CHAR(20) UNIT

٨) NUMERIC LIMIT

الحد الأقصي وهو عند الوصول الصنف اليه يعطي إنذار.

- نوع البيانات MONEY لأن البرنامج لا يفهم التنوع في هذه العملات.

- ملحوظة:

يحتوي نوع البيانات NUMERIC علي رقمين رقم يدل علي العدد الصحيح ، والآخر يدل علي العدد العشري.

- SUPPLIERS الموردين

١) VAR CHAR(20)	S CODE
٢) VAR CHAR(100)	S NAME
٣) VAR CHAR(100)	ADDRESS
٤) VAR CHAR(30) CHOICE	CITY
٥) VAR CHAR(11)	TEL
٦) VAR CHAR(11)	MOBILE
٧) NUMERIC (9,2)	ACCOUNT

- BUY المشتريات

١) DATE -TIME	DATE
٢) CHOICE VAR CHAR(20)	I CODE
٣) NUMERIC (9,3)	QTY
٤) NUMERIC (6,2)	B PRICE
٥) NUMERIC (6,2)	S PRICE
٦) VAR CHAR (20) CHOICE	S CODE
٧) VAR CHAR (10)	INV- NO

المبيعات **-SALES**

١) DATE -TIME	DATE
٢) CHOICE VAR CHAR(20)	I CODE
٣) NUMERIC (9,3)	QTY
٤) NUMERIC (6,2)	PRICE
٥) VAR CHAR (10)	REST- NO

المدفوعات **- PAIES**

١) DATE -TIME	DATE
٢) CHOICE VAR CHAR(50)	PAYITEM
٣) NUMERIC (6,2)	AMOUNT
٤) VAR CHAR (20)	REMARKS
٥) VAR CHAR (10)	P CODE
٦) VAR CHAR (10)	INV- NO

4- تحديد المفتاح الرئيسي لكل نقطة رئيسية:

١ -المفتاح الرئيسي لجدول البضائع . It code

٢ -المفتاح الرئيسي لجدول الموردين . Sup code

٣ -المفتاح الرئيسي لجدول المصروفات .Reset code

٤ -العلاقات بين الجداول:

١ -العلاقة بين جدول البضائع والشراء.

- N ITEM BUY ONE TO MANY

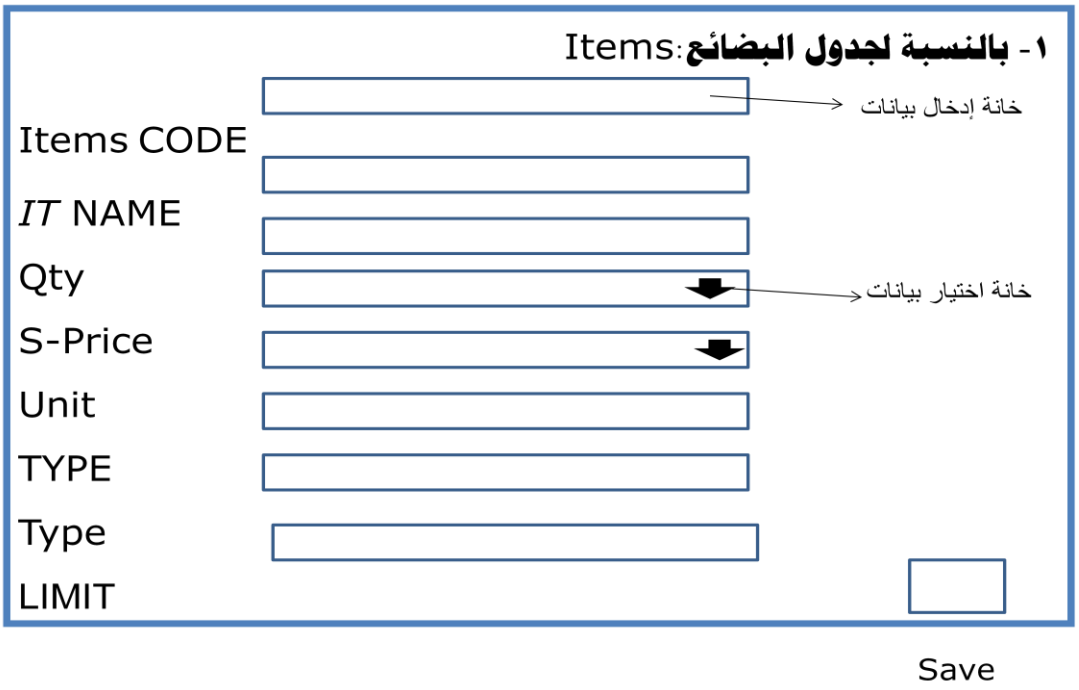
٢ -العلاقة بين جدول الموردين والمدفوعات.

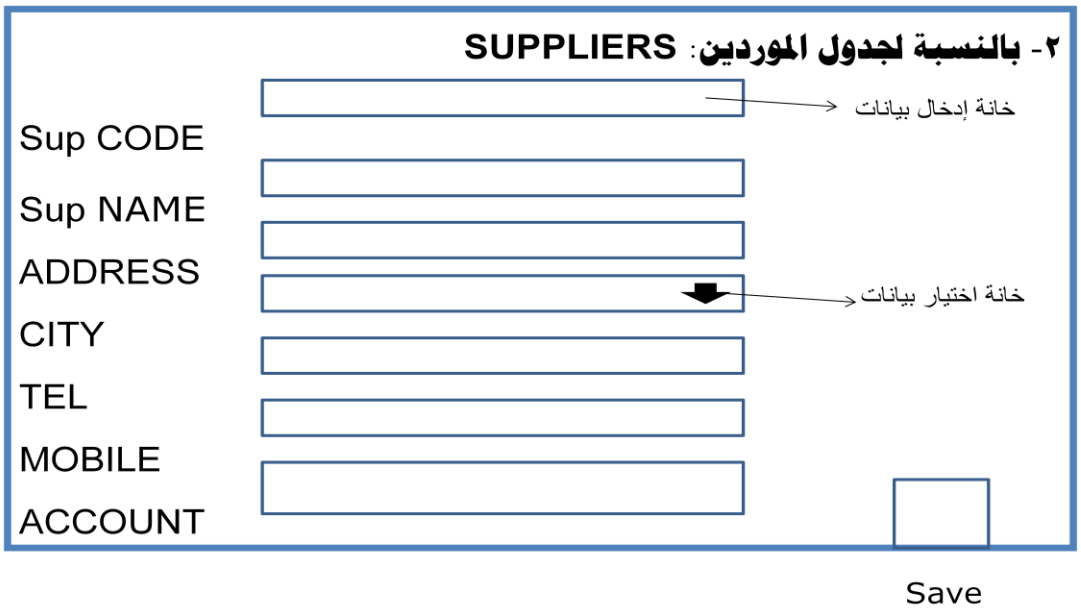
- ITEM SALE ONE TO MANY

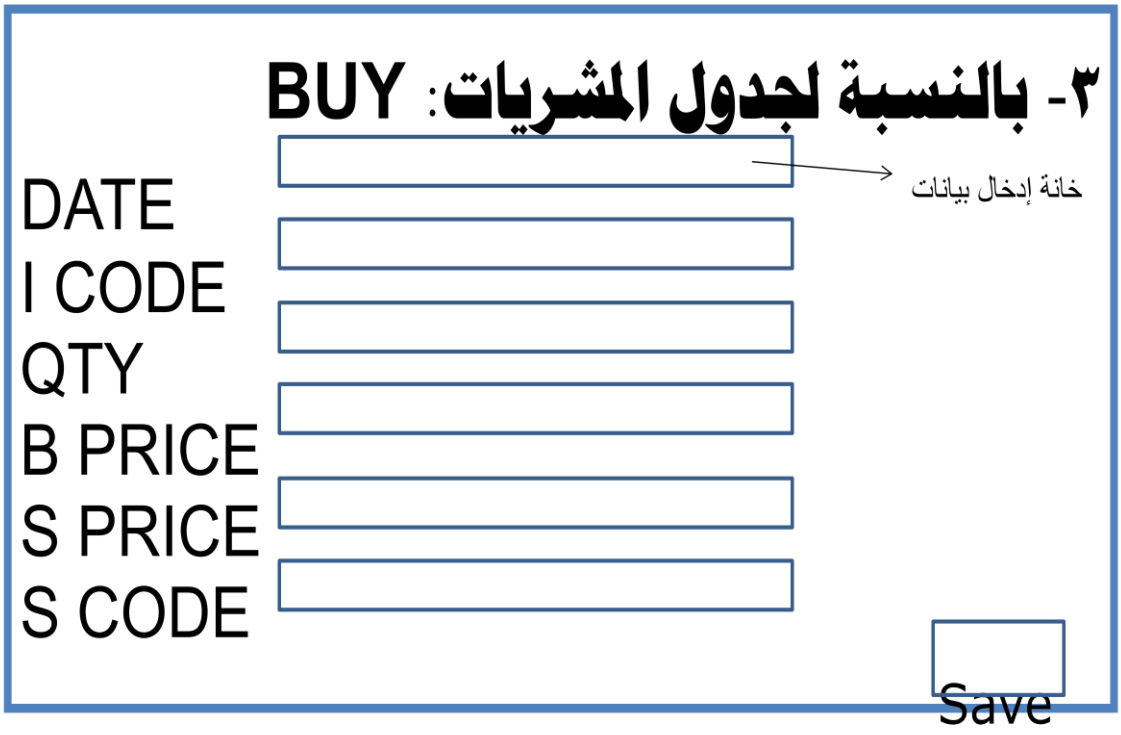
٣ -العلاقة بين جدول البضائع والمبيعات.

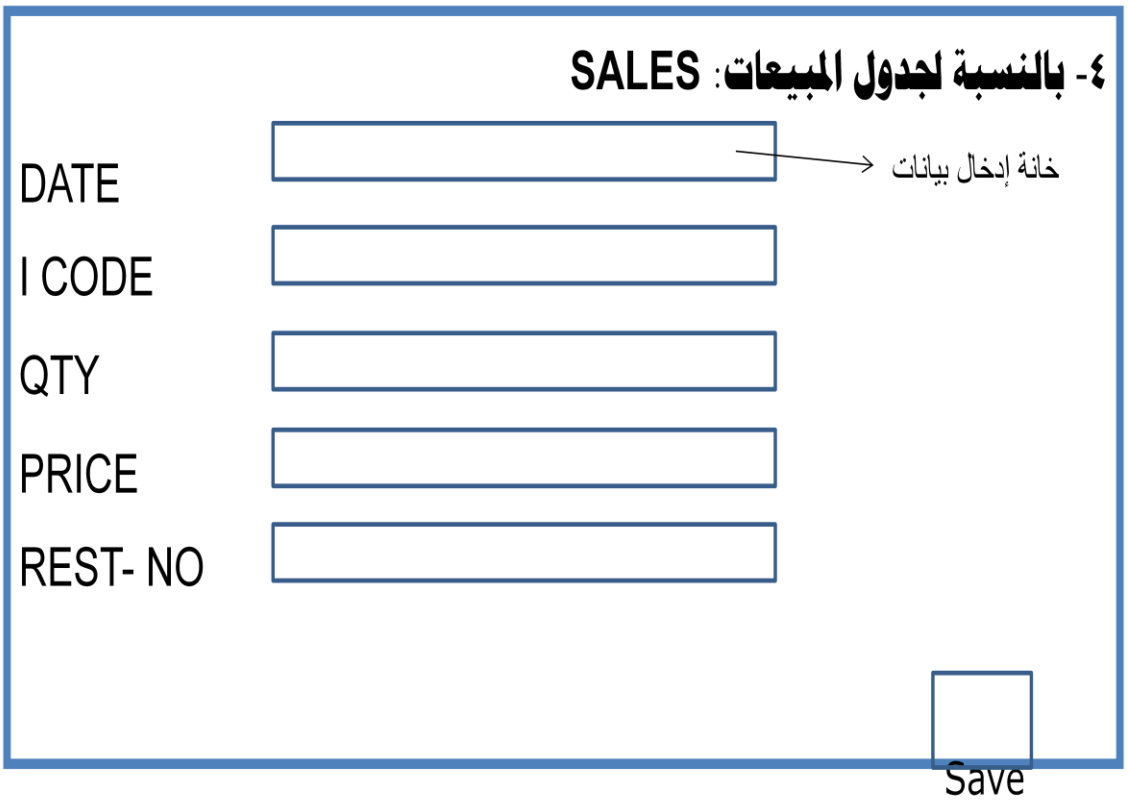
- SUPPLAIR ITEM MANY TO MANY

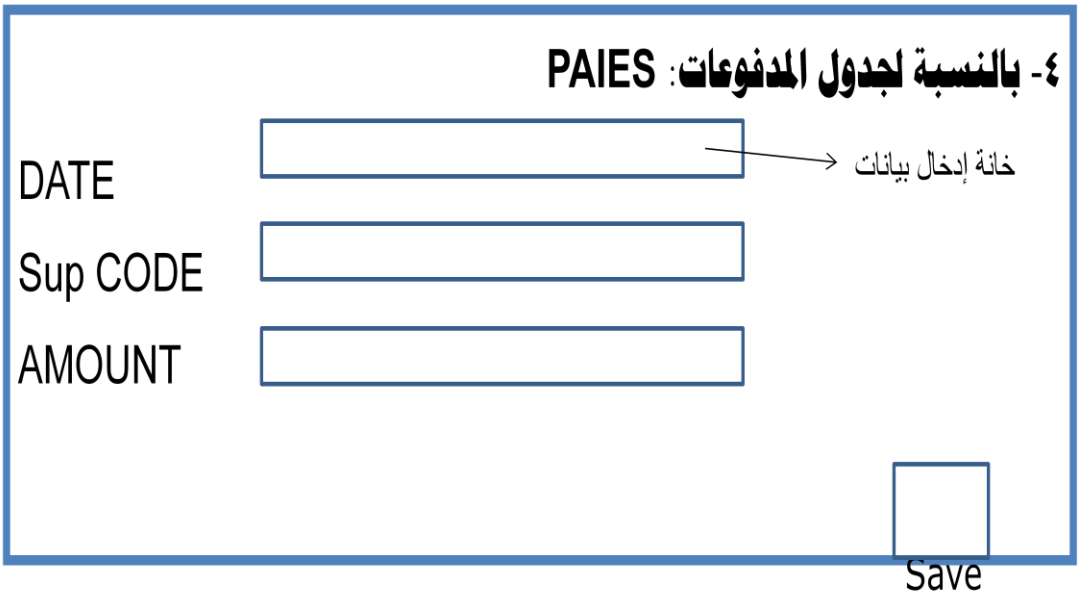
٥ -تصميم شاشات إدخال البيانات.




٢- بالنسبة لجدول الموردين: SUPPLIERS
خانة إدخال بيانات
Sup CODE
Sup NAME
ADDRESS
خانة اختيار بيانات
CITY
TEL
MOBILE
ACCOUNT
Save


٣- بالنسبة لجدول المشريات: BUY
خانة إدخال بيانات
DATE
I CODE
QTY
B PRICE
S PRICE
S CODE
Save


٤- بالنسبة لجدول المبيعات: SALES
DATE
I CODE
QTY
PRICE
REST- NO
خانة إدخال بيانات
Save

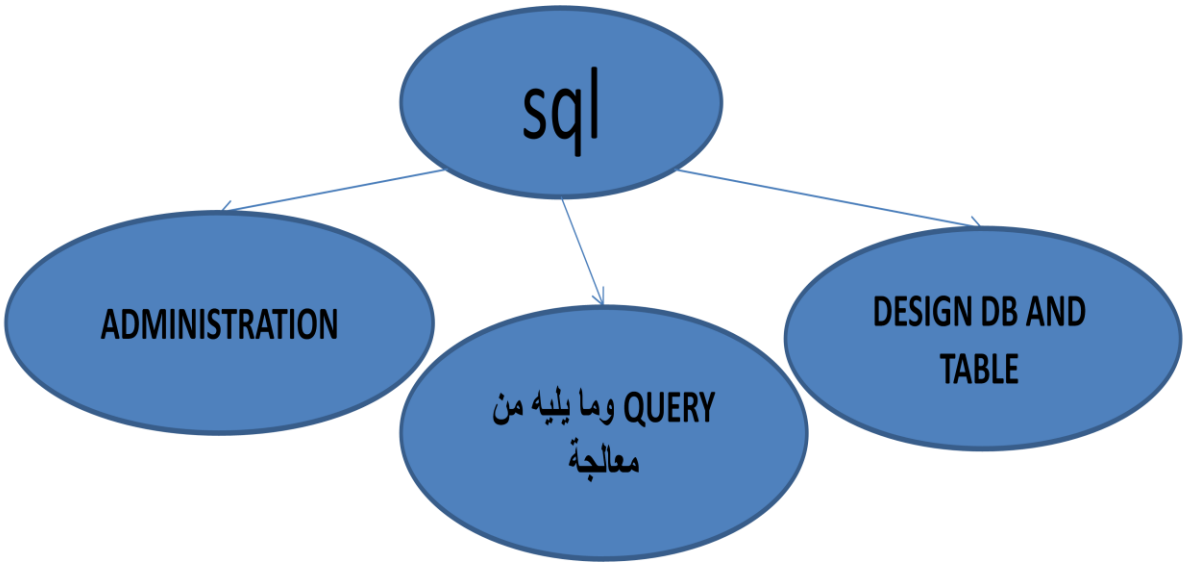

٤- بالنسبة لجدول المدفوعات: PAIES
DATE
Sup CODE
AMOUNT
خانة إدخال بيانات
Save

٦ -تحديد متطلبات البرنامج.

- **Bar Code Printer** **مطبع الكود علي المنتج**
- **Bar Code Reader** **لقراءة كود البضائع**
- **Reset Drinter** **مطبع الفاتورة**
- **Computer** **كمبيوتر**
- **Serve** **شبكات**
- **Network** **لربط الأجهزة ببعضها**
- **Laser Printer** **لطبع التقارير**

الفصل السابع : كيفية التحكم في قواعد البيانات :

الفصل السابع: التحكم في قواعد البيانات



- **وظيفة: ADMINISTRATION**

1- تسجيل الأشخاص وبيانتهم. "PASSWORD, NAME" .

2- توزيع الصلاحيات علي المستخدمين وماذا يفعل كل نهم مثل " الدخول علي SERVER و DB والتي يمكن أن يتعال معها ويراها، وماذا يمكن أن يفعل معها من حيث القراءه فقط، والتعديل و الكتابه، والإدارة ....، والجداول التي يمكن أن يتعامل عها ويراها، والحقول أيضاً" .

(SERVER← DB← USE DB← TABLES ← FIELDS).

٣ -الحفاظ علي البيانات وعمل BACK UP للبيانات وتحديد النوع المناسب منها.

٤ -كيفية التعامل مع البيانات.

- **كيفيه إضافة NEW USER وإعطاء صلاحيات.**

1- من SECURITY التي توجد اسفل كلمة DB

SECURITY ⟶ LOGIN

ويحتوي LOGIN علي قائمة المستخدمين الذي يسمح لهم برؤية DB والتعامل معها ومن بينهم

. SA ⟵ DEFAULT

وهو اعلي USER في ADMIN DB .

R .C LOGIN ⟵ NEW LOGIN حتي نضيف المستخدم

2- تظهر نافذة تحتوي علي الأتي :

- LOGIN NAME اسم المستخدم الجديد

ويوجد في النافذة اختيار من بين

- WINDOWS AUTHENTICATION.
- SQL SERVER AUTHENTICATION .

ويظهر الفرق بينهم في المثال التالي DB ADMIN وSQL SERVER

3- يمكن لأي مستخدم معه USER NAME و PASSWORD وخاصية SQL S AUTHENTIC أن يدخل علي SERVER من أي برناج خارج الشركة والوصول الي DB .

- **<u>نافذة PASSWORD :</u>**

CONFIRM PASSWORD

ونلاحظ أن عند إدخال مستخدم جديد لا يقبل البرنامج أي PASSWORD ضعيفة ولكنه يعطي عدة اختيارات هي :

1- ENFORCE PASSWORD POLICY .

وهي تطبق قواعد ال PASSWORD العالمية حيث :

١ -لابد من احتوائه حروف كبيرة .

٢ -حروف صغيرة .

٣ -كذلك لابد من وجود أرقام.

٤ -ووجود علامات.

٥ -لا تقل عن 8 حروف .

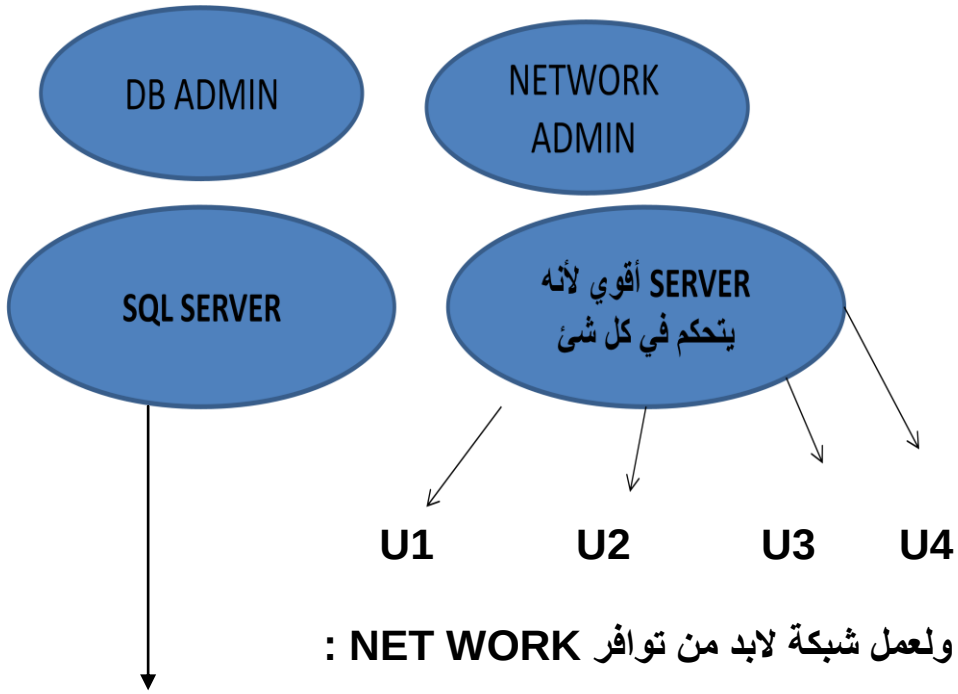
2- ENFORCE PASSWORD EXPIRATION

وهو يضع فترة معينة نستخدم فيها ال PASSWORD وبعدها يتوقف عملها. ويوجد في النافذة اختيار من بين :

- WINDOWS AUTHENTICATION.
- SQL SERVER AUTHENTICATION .

ويظهر الفرق بينهم في المثال التالي DB ADMIN و SQL SERVER .

3- يمكن لأي مستخدم معه USER NAME و PASSWORD وخاصية SQL S AUTHENTIC أن يدخل علي SERVER من أي برناج خارج الشركة والوصول الي DB .



ولعمل شبكة لابد من توافر NET WORK :

عبارة عن

1- COMPUTER

2- SQL SERVER

3 -SERVER

1- من هذا الشكل نلاحظ أن DB ADMiN يمكن نع NET WORK ADMIN من دخول SQL SERVER.

2- عند إضافة مستخدم جديد يستقبله NET WORK ADMIN ويعطي ل USER NAME و PASSWORD ويوصفه علي صلاحياته، وإذا اراد المستخدم الدخول علي db يذهب أيضاً إلي db admin حتي يضيفه علي SERVER وفي عملية الإضافة يختار ما بين والذي يعطي صلاحيات أكبر ، ويعطي security أعلي

- windows authentication

- Sql server authentication

3- صلاحيات NET WORK ADMIN تجعله يتحكم في server

ملحوظة :

هناك شخصان يمكن أن يتعاملوا مع sql server ويمكن إضافتهم علي db هما :

1- مستخدم عادي يتعامل مع db فقط من خلال عمل تقرير ، query ، يدخل بيانات . وهنا يمكن أن يري ال db كلها أو جزء منها .

2- db – admin وهو يدير db بالكامل.

- **Server roles**

وتحتوي علي الصلاحيات الآتية:

1-bulk admin مسئول عن كل العمليات التي تتم علي db

2-db creator يسمح له بتخليق db جديدة لا غير

3-disk admin مسئول عن hd وتقسيمه وصيانته ، وتوفير مساحه عليه

4-process admin إجراء العمليات علىdb

5-security admin مسئول عن تأمين dbوحماية جميع البيانات والجداول التي توجد بها

6-server admin

7-setup admin

مراحل تحديد صلاحيات كما يلي : new user -

1-General وهي لإعطائه password , username

2-server roles server تحديد صلاحيات المستخدم علي ال

3-user mapping تحديد صلاحيات كل مستخدم علي كل dbالموجودة

4-securables تحديد صلاحيات كل جزء من db

5-status تحديد حالة المستخدم علي server يعمل أم لا يعمل

من أهم صلاحيات new user : -

1-db – data reader statement

يستطيع قراءة db فقط وعملquery .

يستطيع الكتابة في db فقط 2-db – data writer

3-db–security admin

مسئول عن تأين db كما يستطيع حذف أو إضافة إي مستخدم.

4-db–owner

هو صاحب db يستطيع تنفيذ كل الأوامر منها تعديل، قرأة ، كتابة، تغيير، حذف ، إضافة.

5-deny data reader / writer

منع المستخدم من القراءة أو الكتابة وليس أي مستخدم ، أنما المستخدم الذي معه علي مستوي عالي من security وهو الذي يدخل علي server من خلال windows authentication لأنه يمكنه الدخول علي sql server وعلي db من خلال deny .

أما المستخدم العادي لا نعطي له السماح بالقراءة أو الكتابة من خلال الصلاحيات المتاحة.

- **user mapping**

وهي التي تحدد صلاحيات كل مستخدم حيث تكون الصفحة مقسمة لقسمين هما :

1- نحدد ال db التي يمكن أن يراها المستخدم.

2- نحدد الصلاحيات علي هذه db كلها.

وهذه الصلاحيات هي:

1-db–access admin

مسئول عن إضافة tables أو حذف جدول أو تعديل في db ولكن لا يستطيع أن يحذف مستخدم أو يعدل في بياناته أو يغير من صلاحياته أو يضيف مستخدم جديد.

2-db–backup operator

مسئول عن حفظ نسخ إحتياطية من db حتي إن لم يكن يري هذه db أو يكتب بها.

إذا كان لدينا DB كبيرة وأراد شخص ما أن يتعامل مع هذه DB وأردنا أن نحدد له جدول واحد فقط نستخدم Securable:

Securable

١ -نقوم بفتحها من خلال

Data Base ⟵ اسم DB التي تريدها ⟵ Security ⟵ Users ⟵

علي اسم Data Base المستخدم الذي نريده.

٢ -تظهر نافذة تحتوي علي كلمة Add نضغط عليها تظهر نافذة آخري تحتوي علي What objects do you wish to Add?

وتخيرنا ما بين

-Specific objects ومنها نحدد جدول معين أوView معين بالاسم .

- All objects of the types.

ونحدد كل الاشياء من نوع معينView مثلا أو كل Data Base .

-The server : admin وهنا تكون علي serve نفسه ويكون من خلال

٣ -نختار الإختيار الاول وبعدها OK.

٤ -تظهر نافذة تضغط علي Object types.

STATUS:

وهي الحالة العامة لل user، وتحتوي هذه النافذة علي :

-Permission to connect to database engine:

Grant يسمح له Deny لا يسمح له

-Login

Enabled من حقه الدخول Disabled ليس من حقه الدخول

وتستخدم هذه الخاصية في الأتي :

١ -موظف كان يعمل علي Server وترك العمل، نلغي عمل (passwordو User name) الخاصة به حتي لا يستطيع الدخول علىServer من برنامج خارج الشركة.

٢ -شخصية علي المستوي Windows authentication ولا نريد الدخول علي Data Base أو علي Sql Server لهذا ندخل اسمه علي Server ونعمله Deny و Disable.

٣ -في حالة وجود اثنان من المستخدمين يقوموا بعمليتين مختلفتين علي نفس الجدول يحدث ما يسمي LOCK وتعرف هذا من خلال

Activity monitor ←Locks by objects

٤ -هناك جملتين : Rolled back أو Commit : ويستخدما في حالة وجود مستخدم يقوم بعمل Insert لبعض البيانات وهي معتمدة علي بعضها وقام ADMINISTRATOR وقم بعمل Kill process يكون الوضع كما يلي:

وفي الحالة السابقة لابد أن نعمل Commit حتي يحفظ البرنامج البيانات وما تم عليها من عمليات وفي حالة حدوث أي مشكلة بدون عمل Commit يقوم البرنامج بعمل Rolled back ويصل للبداية مرة آخري.

٥ -بعد ذلك نضغط علي Browse ونحدد أسماء Database التي نريدها وبعدها OK .

٦ -تظهر أسماء الجداول Database وتحدد الشروط الذي نريدها وهي :

- Alter : يعدل في نوع البيانات أو في بناء الجداول أو في حجم البيانات

-Control : التحكم في الجدول نفسه من حيث التعديل أو إضافة User

-Delete : التحكم في مسح الجدول

-Insert : التحكم في الإضافة علي الجدول

-Select : يمكنه عمل Query علي الجدول

-Update : يمكنه من تحديث بيانات الجدول

-View definition : يقوم بعرض كل ما يخص data base

التمرين الاول:

على اعتبار ان لديك الجدولين التاليين واللذين يمثلان قاعدة بيانات لمتجر حواسيب.
اجب عن الاسئلة التالية:

```
CREATE TABLE Manufacturers (
    Code INTEGER PRIMARY KEY NOT NULL,
    Name TEXT NOT NULL );

CREATE TABLE Products (
    Code INTEGER PRIMARY KEY NOT NULL,
    Name TEXT NOT NULL ,
    Price REAL NOT NULL ,
    Manufacturer INTEGER NOT NULL
        CONSTRAINT fk_Manufacturers_Code
REFERENCES MANUFACTURERS(Code));
```

١ -جد اسماء جميع المنتجات في المتجر
٢ -جد اسماء و اسعار جميع المنتجات في المتجر
٣ -جد اسماء و اسعار المنتجات التي يقل سعرها عن 200$
٤ -جد اسماء و اسعار المنتجات التي تنحصر اسعارها بين 60$ و 120$
٥ -جد اسماء و اسعار المنتجات بالدينار الاردني (سعر الصرف 0.71)
٦ -جد معدل اسعار المنجات جميعها
٧ -جد معدل اسعار المنتجات التي ينتجها المنتج صاحب الرمز 2
٨ -جد عدد المنتجات التي يساو سعرها او يزيد عن 180$
٩ -جد اسماء و اسعار المنتجات التي يزيد سعرها عن 180$ مرتبة اولا تنازليا حسب السعر ثم تصاعديا حسب الاسم.

١٠ - جد اسماء المنتجات مع اظهار اسم المصنع لكل منتج.

١١ - جد معدل اسعار المنتجاتن لكل مصنع.

١٢ - جد اسماء المصنعين اللذين يصنعون منتجات معدل اسعارها اكبر من $180

١٣ - جد اسم وسعر ارخص منتج في قاعدة البيانات

١٤ - جد اسم كل مصنع و اسم اغلى سلعة يصنعها.

١٥ - اضف الى قاعدة البيانات المنتج سماعات بسعر $70 للمصنع رقم 2.

١٦ - عدل ايم المنتج رقم 8 الى طابعة ليزر

١٧ - عدل الاسعار لكل المنتجات بحيث تطبق خصم 10%

١٨ - عدل اسعار المنتجات التي يزيد سعرها عن $180 بحيث تطبق خصم مقداره 5% عليها.

التمرين الثاني:

على اعتبار ان لديك الجدولين التاليين و اللذين يمثلان قاعدة بيانات لمتجر حواسيب. اجب عن الاسئلة التالية

```
CREATE TABLE Movies (
  Code INTEGER PRIMARY KEY NOT NULL,
  Title TEXT NOT NULL,
  Rating TEXT );

CREATE TABLE MovieTheaters (
  Code INTEGER PRIMARY KEY NOT NULL,
  Name TEXT NOT NULL,
  Movie INTEGER
```

```
CONSTRAINT fk_Movies_Code REFERENCES
Movies(Code));
```

١ -جد اسماء جميع الافلام

٢ -جد كل التصنيفات Rating في الجدول.

٣ -جد اسماء الافلام غير المصنفة.

٤ -جد اسماء المسارح التي لا تقوم بعرض افلام حاليا.

٥ -جد معلومات جميع المسارح و معلومات الافلام التي تعرض في المسرح ان كان هناك فلم يعرض في المسرح.

٦ -جد اسماء الافلام التي لا تعرض في اي من المسارح.

٧ -ضع التصنيف "عام" لجميع الافلام غير المصنفة.

قائمة المصطلحات

انجليزي	عربي
Union	اتحاد (مجموعات)
Query	استفسار
Alias	اسم المستعار
Ambiguity	التباس
Order	ترتيب
Alphabetical order	ترتيب الابتثي
Ascending order	ترتيب تصاعدي
Descending	ترتيب تنازلي
Concatenation	تشبيك (سلاسل الاحرف، النصوص)
Expression	تعبير حسابي
Intersection	تقاطع (مجموعات)
Table	جدول
Query statement	جملة استفسار
Field	حقل
Row	سطر (السجل)
String	سلسلة احرف
Selection condition	شرط اختيار
Join condition	شرط ربط

Phrase	عبارة
Column title	عنوان /ترويسة العمود
Difference	فرق(مجموعات)
Index	فهرس
Null	قيمة الخالية
Efficiency	كفاءة التنفيذ
Structured Query Language	لغة الاستفسار البنيوية
Data Definition Language	لغة تعريف البيانات
Data Manipulation Language	لغة تغيير البيانات
Set	مجموعة
Range	مدى
Sort key	مفتاح الترتيب
Query results	نتائج الاستفسار
Text	نص
Pattern	نمط