برمجة قواعد البيانات

برامج قواعد البيانات من أوسع برامج الحاسب انتشارا ويمكن أن تستخدم سواء في المجالات التجارية أو الصناعية ومنها علي سبيل المثال مجالات النظم المحاسبية والمالية ومجال شئون الأفراد والمشتريات، وتعتب ـر لغة الاستعلامات الهيكلية من أهم وأشهر اللغات المستخدمة في برمجة قواعد البيانات من حيث إنشاء ملفات قواعد البيانات واسترجاع البيانات منها بطرق مختلفة ومتعددة تلائم معظم احتياجات المبرمجين وتتميز بسهولة تركيبها وسهلة الفهم للجميع.

مفاهيم Concepts:

البيانات DATA:
هي الأرقام أو الحروف أو الرموز أو الكلمات القابلة للمعالجة بواسطة الحاسب مثل الرقم 9 أو كلمة بيانات.

المعلومات Information :
هي بيانات تم تنظيمها أو معالجتها لتحقيق أقصى استفادة منها.

قاعدة البيانات Database:
هي عبارة عن تجمع من البيانات ذات العلاقة ببعضها، أو هي تجميع لكمية كبيرة من المعلومات أو البيانات وعرضها بطريقة أو بأكثر من طريقة تسهل الإستفادة منها. فمثلا دليل الهاتف الذي يشتمل على أسماء وعناوين وأرقام هواتف سكان مدينة طرابلس.
ويجب ان يتوفر في قواعد البيانات الخصائص التالية:

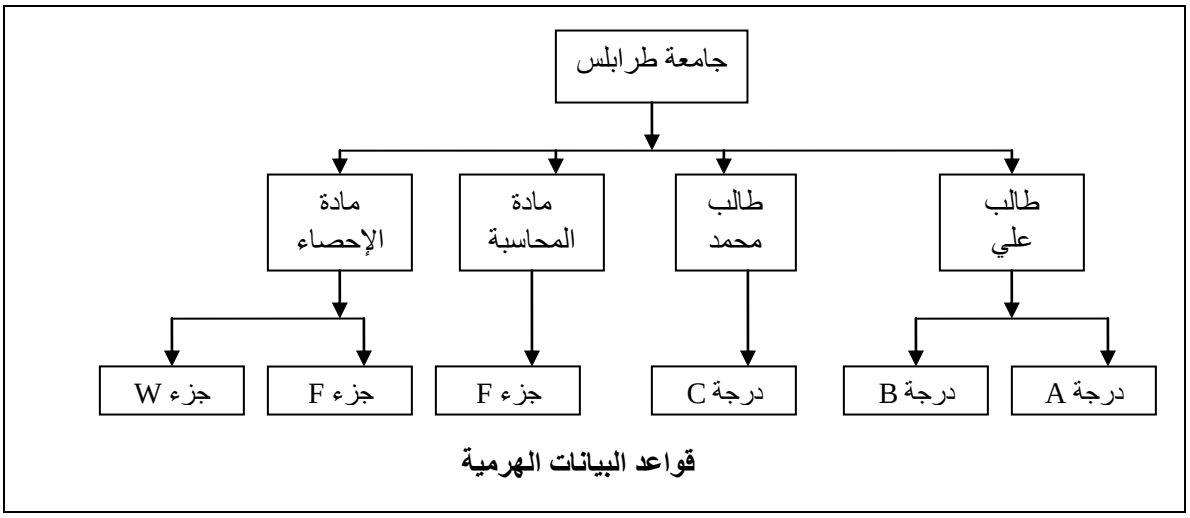
- يجب ان تعكس العالم الحقيقي الذي تمثله البيانات وأي تغيير في العالم الحقيقي ينعكس عليها.
- يجب ان تكون ذات هيكلية مترابطة بطريقة منظمة ومتعارف عليها.
- يجب ان تبنى وتصمم لغرض معين محدد وان يكون لها مستخدمين محددين.

أنواع نماذج قواعد البيانات DataBase Models:

نماذج قواعد البيانات هي تجمع من البناء المنطقي الذي يمثل هيكلية البيانات وعلاقة البيانات مع بعضها البعض داخل قواعد البيانات. تختلف نماذج قواعد البيانات حسب البناء أو التركيب على النحو التالي:

1. قواعد بيانات ذات شكل هرمي Hierarchy Databases.
2. قواعد بيانات شبكية وتسمى Network Databases.
3. قواعد بيانات الكائنات الموجهة Object Oriented Databases.
4. قواعد بيانات علائقية وتسمى Relational Databases.
5. قواعد بيانات هجينة Hybrid Databases.

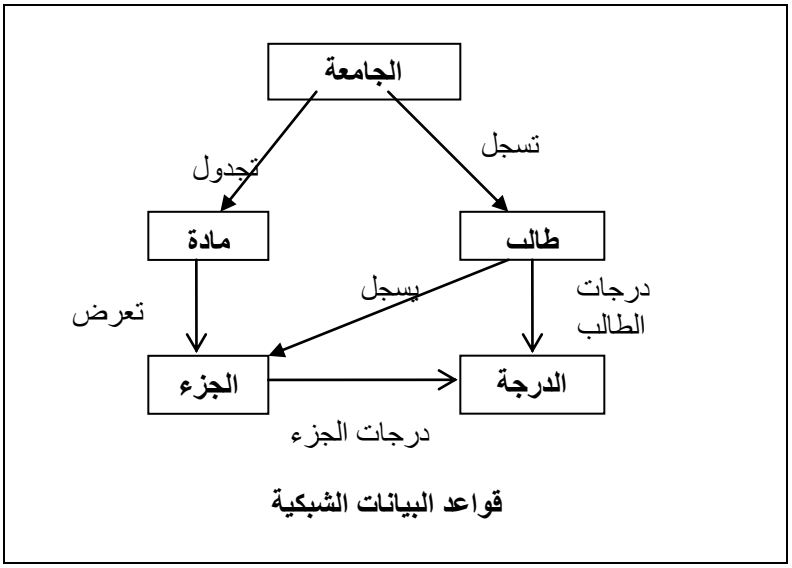
قواعد بيانات ذات شكل هرمي Hierarchy Databases:
يقوم هذا النوع بتنظيم البيانات على شكل هرمي أو على شكل شجرة مقلوبة جذورها في القمة وتخرج منها الفروع. شأن هذه التركيبة شأن شجرة الأسرة فلها جد واحد والجد له عدة أبناء والأبناء هم أباء الأحفاد. وهذا شكل توضيحى للنظم الهرمية وتفرعاتها:



قواعد البيانات الهرمية

قواعد البيانات الشبكية Network Databases:

يتغلب هيكل بيانات التركيب الشبكي علي معوقات التكوين الهرمي الذي لا يسمح للابن ان يكون له أكثر من أب واحد. وفي هذه الحالة يتم تمثيل البيانات على شكل سجلات Records ويتعامل معها على شكل مجاميع من السجلات Record Sets وهذا شكل توضيحي للتكوين الشبكي.

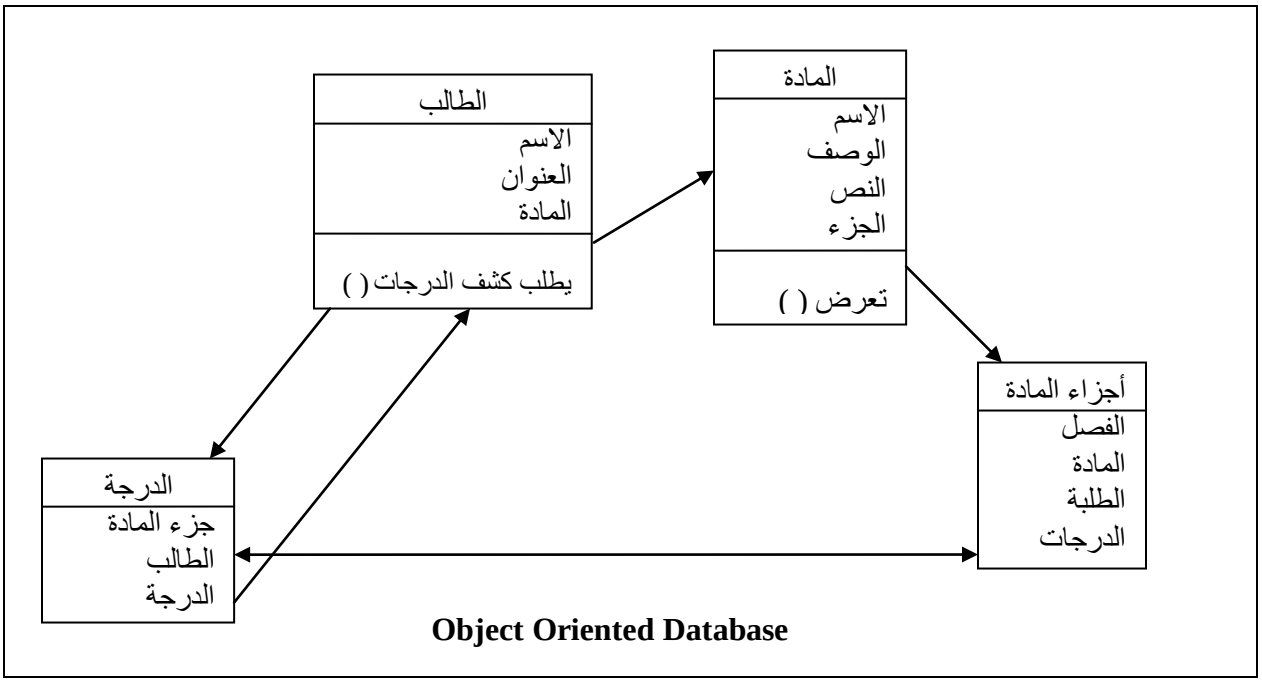


قواعد البيانات الشبكية

يقتصر تشغيل النوعين السابقين على الحاسبات الكبيرة وذلك لأنها تتطلب ذاكرة ذات أحجام كبيرة، ولها مزايا عديدة فهي أكثر كفاءة من قواعد البيانات العلائقية، وتتعامل مع كم كبير جدا من المعلومات.

قواعد بيانات الكائنات الموجهة Object Oriented Databases:

البيانات تخزن على هيئة كائنات Objects لها خصائص Attributes و عمليات Methods والعلاقة بين هذه الكائنات تخزن على هيئة كائن يشير إلى كائنات أخرى (Object references to other Objects).



Object Oriented Database

قواعد البيانات العلائقية Relational Databases:

قواعد البيانات العلائقية هي أكثر سهولة و إستخداماً مع الحاسبات الشخصية ومن مزاياها إنها لا تحتاج إلى ذاكرة أو وسائط تخزين بأحجام كبيرة مثل الأنواع الأخرى التي تعمل على الحاسبات الكبيرة كما إنها أسهل في تعلمها وبرمجتها.

هذا النوع من قواعد البيانات يستخدم طريقة الجداول في تمثيل البيانات، وكل جدول من هذه الجداول مربوط مع الأخر ضمن علاقة معروفة سابقا ضمن بيانات الجدول نفسه.

قاعدة البيانات العلائقية تنظم على هيئة مجموعة من القواعد التي تربط مجموعة من الجداول Table ويسمى علاقة Relation وكل جدول أو علاقة تتكون من مجموعة من الصفوف Row وتسمى أيضا Tuple وكل صف يتكون من مجموعة من الحقول أو الخصائص Attribute والتي يجب أن تكون في أبسط صورة لها (انظر الشكل التالي).

<u>كينونات قاعدة البيانات Database Objects:</u>

1. ***الجدول Table:*** يخزن داخله البيانات ويكون ذا بعدين وهو وحدة التخزين الأولية في أنظمة قواعد البيانات العلائقية وهو يحتوي على البيانات التي تتعلق بكائن محدد Entity (مثل: موظف، فاتورة، طالب) ويتكون من صفوف وأعمدة.

<u>الصف:</u> يحتوي على جميع البيانات التي تخص كائن واحد مثل الموظف من حيث: رقم الموظف و اسم الموظف و عنوانه.....الخ.

<u>العمود:</u> يخصص لنوع واحد من أنواع البيانات لجميع السجلات مثل الاسم أو الرقم.

وفي قواعد البيانات فإننا نسمي الصفوف بالسجلات Records ونسمي الأعمدة بالحقول Fields. لذلك يمكننا القول بأن قاعدة البيانات تتكون من الجداول والجداول تتكون من السجلات والسجلات تتكون من الحقول، وكل حقل يحتوي على معلومة واحدة، أي أن الحقل هو أصغر وحدة قاعدة البيانات.

2. ***View:*** هي الجدول المنطقي أو المتحرك أو الافتراضي الذي يحسب أو يجمع من البيانات الموجودة بالجداول الموجودة في قاعدة البيانات. التغيرات التي تحدث علي البيانات في الجداول بقاعدة البيانات تعدل أيضاً في البيانات التي تري في الـ View.

3. ***Index:*** هو Data Structure يقوم بتحسين السرعة في العمليات بالجدول في قاعدة البيانات. و يمكن بناء الـ Index باستخدام عمود أو أكثر بالجدول. ويوجد منه نوعان: Unique Index و Non-Unique Index. يتكون الفهرس من مفتاح الفهرس Key و فئة من المؤشرات Set of Pointers تشير إلى موقع البيانات التي تحمل نفس قيمة المفتاح .
4. ***الإجراءات المخزنة Stored Procedures:*** هي برامج فرعية متوفرة لأي تطبيق يرغب في التعامل مع قاعدة البيانات وعادة ما تكون مخزنة في قاعدة البيانات نفسها.
5. ***المقداح Trigger:*** هو عبارة عن برنامج أو شفرة Code يتم تنفيذه تلقائياً استجابة لحدث Event محدد في جدول محدد داخل قاعدة البيانات.

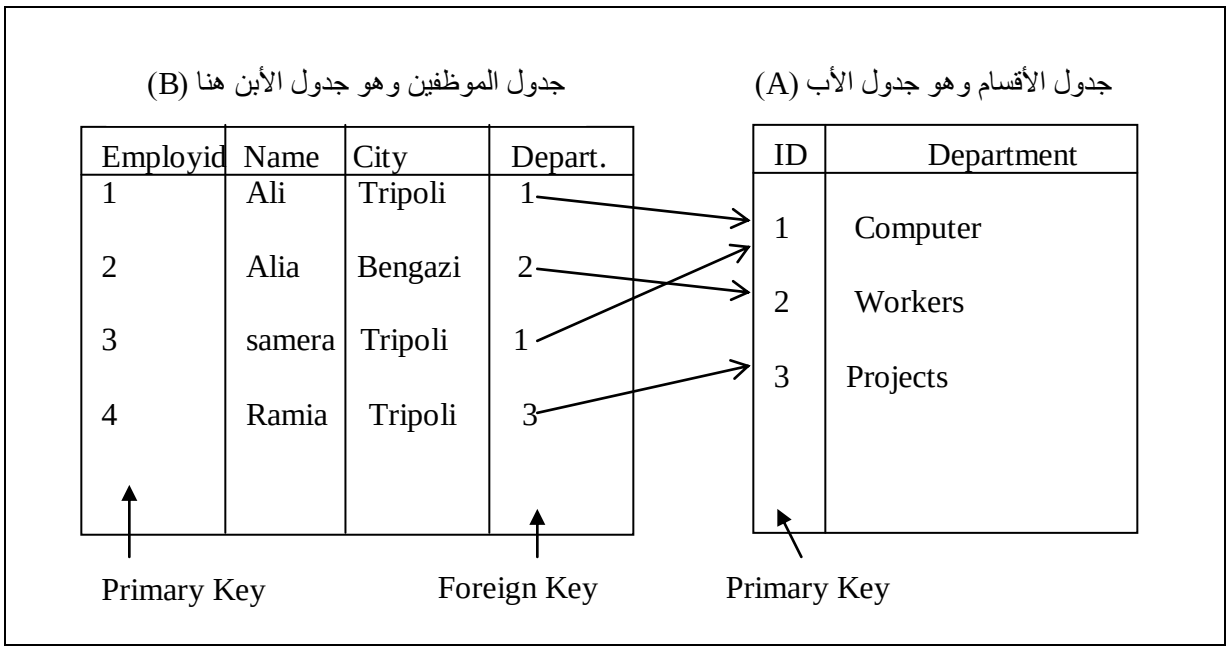
المفاتيح بالجداول Keys in a Table :

- Primary Key (المفتاح الأساسي): وتكون قيمة العمود (الذي تم اختياره ليكون Primary Key) بالجدول Unique (فريد) وكذلك به قيمة (لا تساوي Null) . لنفرض أن الجدول الذي به هذا المفتاح هو (A).
- Foreign Key (المفتاح الثانوي): وهو الحقل أو العمود بالجدول (B) الذي يناظر المفتاح الأساسي الموجود بالجدول الأخر (A). ويستعمل هذا المفتاح للربط بين البيانات بالجدولين A و B.

شروط الربط بين الجداول:

1. يجب ان يكون هناك عمود لا يتكرر في جدول الأب (Uniquely) Primary Key.
2. في أغلب الأحوال بيانات هذا العمود لا يتم تغيرها.
3. يتم إختيار عمود في جدول الإبن ليكون عمود الربط Foreign Key وتكون القيم المتواجدة بها متماثلة مع القيم في العمود المذكور فى رقم 1 أو يمكن ان تكون خالية.

مثال:



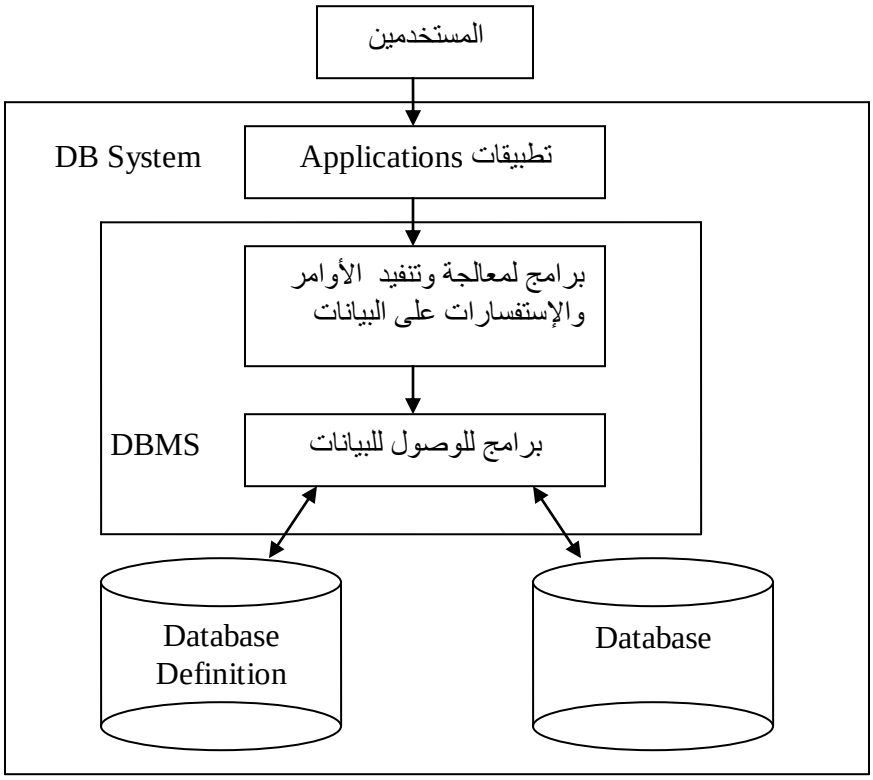
نظم إدارة قواعد البيانات DataBases Management Systems

لما كانت قاعدة البيانات تساعد في تحقيق مجموعة من الأهداف المؤثرة على الأنشطة الرئيسية في مجالات تطبيقات التجهيز الألى للبيانات، فأنه يلزم وجود نظم معينة لتنظيم وإدارة البيانات المخزنة، وهو ما يطلق عليه عبارة **نظم إدارة قواعد البيانات DataBase Management Systems (DBMS)**. وبذلك يمكن تعريف هذه النظم على أنها مجموعة من البرامج الجاهزة التي تقوم بتنفيذ جميع الوظائف المطلوبة من قاعدة البيانات من حيث:

1. إنشاء قواعد البيانات وتعريفها.
2. التعامل مع البيانات من حيث البحث و الترتيب والتعديل والإدخال والإلغاء والعرض.
3. الوصول للبيانات المخزنة على وسائط تخزين.
4. الحفاظ على سلامة البيانات Security and safety.
5. تسمح للعديد من المستخدمين للوصول للبيانات في نفس الوقت بدون التسبب في فسادها.

ومن أمثلة DBMS :
Microsoft Access - MYSQL - Oracle – MS SQL Server وغيرها.

الشكل التالي يوضح التركيب الهيكلي لبرامج نظم المعلومات.



البناء الهيكلي لنظم إدارة قواعد البيانات

أولاً: المستخدمين:
1. مديري قواعد البيانات Database Administrators: يدير استخدام DBMS والتأكد من ان قاعدة البيانات تقوم بوظائفها بشكل صحيح.
2. مصممي قواعد البيانات Database Designers: يصمم هيكلية قاعدة البيانات.
3. مستخدمي قواعد البيانات End Users: الأشخاص المستخدمين للبرامج والتطبيقات لتنفيذ العمليات اليومية للمؤسسات و الشركات.
4. محللي النظم والمبرمجين System Analysts & Programmers .

ثانياً: التطبيقات Applications:
التطبيقات والبرامج والأنظمة التي تعد بلغات مختلفة وتربط المستخدمين بقواعد البيانات المخزنة.

ثالثاً: برامج معالجة لغة الاستعلام:
هي برامج تقوم بتطبيق وترجمة الاستفسارات القادمة من التطبيقات بلغات عالية المستوى وتحسين Optimize الاستفسار Query باستخدام خطط مسبقة مثل استخدام الفهارس الموجودة أو ربما خلق فهارس جديدة.

رابعاً: برامج الوصول للبيانات:
تتعامل مع نظام التشغيل Operating System وتهتم بكيفية تخزين البيانات على القرص الصلب Hard Disk وتستخدم بعض المعلومات عن كيفية تخزينها Data Definition or Meta_Data.

المشاكل التي نتجنبها باستخدام نظم قواعد البيانات:
1. صعوبة الوصول إلى البيانات.
2. تكرار البيانات وتضاربها Redundancy and inconsistency.
3. فصل وتمييز البيانات بعضها عن بعض.
4. الوصول الغير المخول للبيانات أو التضارب.
5. مشاكل الأمان Security.
6. مشاكل التكامل بين البيانات Integration.

كثير من الشركات الكبرى قامت بتطوير برامج إدارة قواعد البيانات فظهرت لها عدة تصنيفات:
1. حسب نوع قاعدة البيانات.
2. حسب عدد المستخدمين.
3. حسب البناء الهيكلي للنظام.
4. حسب الأهداف.

حسب نوع قاعدة البيانات:
وهي برامج تختلف بإختلاف نوع قاعدة البيانات التي تعمل عليها وهي حسب الأنواع المذكورة في السابق.

حسب عدد المستخدمين:
1. نظم إدارة تدعم مستخدم واحد Single User :لا تسمح لأكثر من مستخدم واحد بإستخدام قاعدة البيانات في نفس الوقت.
2. نظم إدارة تدعم مستخدمين متعددين Multi Users: وهي تسمح لأكثر من مستخدم بالتعامل مع قاعدة البيانات في نفس الوقت.

حسب البناء الهيكلي للنظام:
1. نظم إدارة قواعد بيانات مركزية Centralized DBMS: قواعد البيانات تخزن في جهاز واحد.
2. نظم إدارة قواعد بيانات موزعة Distributed DBMS: قواعد بيانات موزعة على أكثر من موقع ومرتبطة مع بعضها البعض بواسطة شبكة حاسوب Computer Network.

حسب الأهداف:
وتنقسم إلى جزئين:
1. أغراض عامة.
2. أغراض خاصة: مثل قواعد بيانات دليل الهاتف أو حجوزات شركة الطيران.

لغة نظم إدارة قواعد البيانات DBMS Languages
اعتمدت مؤسسة ANSI لغة موحدة قياسية للتعامل مع جميع قواعد البيانات بإختلاف أنواعها و أحجامها وهي لغة الإستعلام الهيكلية Structured Query Language(SQL) و تتكون من مجموعة من الأوامر التي تستخدم في تكوين Create ومعالجة Manipulate البيانات في قاعدة البيانات العلائقية.

ما الذى توفره اللغة:

1. تمكنك من الدخول لقواعد البيانات.
2. تمكنك من إستخراج البيانات من القاعدة.
3. تمكنك من إضافة بيانات إلى قاعدة البيانات.
4. تمكنك من حذف البيانات من قاعدة البيانات.
5. تمكنك من تعديل البيانات المسجلة.
6. هي لغة سهلة التعلم والفهم.

لغة SQL يمكن تصنيفها إلى أربع أجزاء:

1. لغة تعريف البيانات SQL Data Definition Language (DDL).
2. لغة التعامل مع البيانات SQL Data Manipulation Language (DML).
3. لغة التحكم في البيانات SQL Data Control Language (DCL).
4. لغـة معالجة البيانات SQL Transaction Control Language (TCL).

لغة تعريف البيانات SQL Data Definition Language (DDL)

ه ي قسم اللغة المسئول عن تعريف البيانات وتتكون من الأوامر التالية:

1. Create Database: إنشاء قاعدة بيانات جديدة.
2. Create Table: لإنشاء جدول داخل قاعدة البيانات.
3. Create Index: لإنشاء مفتاح للبحث.
4. Drop Index: لحذف مفتاح البحث.
5. Drop Table: لحذف الجدول من قاعدة البيانات.
6. Truncate Table: لحذف البيانات.
7. Drop Database: لحذف قاعدة البيانات.
8. Alter Table: للتعديل في هيكلية الجدول.
9. Rename Table: لتغيير اسم جدول.

لغة التعامل مع البيانات SQL Data Manipulation Language (DML)

هي قسم اللغة المسئول عن معالجة البيانات وتتكون من الأوامر التالية:

1. Insert Into : إضافة بيانات جديدة.
2. Update: التعديل على البيانات المسجلة سابقا في قاعدة البيانات.
3. Delete: حذف البيانات من القاعدة.
4. Select : استخراج البيانات من قاعدة البيانات.

لغة التحكم في البيانات SQL Data Control Language (DCL)

هي قسم اللغة المسئول عن ضمان الأمان Security والتحكم في وصول المستخدمين للبيانات وتتكون من الأوامر التالية:

1. Grant: إعطاء الصلاحية للأشخاص في الوصول للبيانات.
2. Revoke: إلغاء أو نزع الصلاحيات من المستخدمين في الوصول للبيانات.

لغـة معالجة البيانات Transaction Control Language (TCL)

هي قسم اللغة المسئول عن التحكم في عمليات معالجة البيانات (التعاملات) وتتكون من الأوامر التالية:

1. Commit: يستخدم ليتم تطبيق العمليات أو التغييرات الحاصلة علي البيانات.
2. Rollback: يستخدم لإلغاء تطبيق العمليات أو التغييرات الحاصلة علي البيانات.

أولا: لغة تعريف البيانات SQL Data Definition Language (DDL)

كما عرفنا بالمحاضرة الأولي فإن DDL ه و قسم اللغة المسئول عن تعريف البيانات وتتكون من الأوامر التالية:

1. Create Database: إنشاء قاعدة بيانات جديدة.
2. Create Table: لإنشاء جدول داخل قاعدة البيانات.
3. Create Index: لإنشاء مفتاح للبحث.
4. Drop Index: لحذف مفتاح البحث.
5. Drop Table: لحذف الجدول من قاعدة البيانات.
6. Truncate Table: لحذف البيانات.
7. Drop Database: لحذف قاعدة البيانات.
8. Alter Table: للتعديل في هيكلية الجدول.
9. Rename Table: لتغيير اسم جدول.

إنشاء قاعدة بيانات CREATE DATABASE

لإنشاء قاعدة بيانات فإن تركيبة جملة SQL تكون كالتالي:

```
CREATE DATABASE database_name
```

والمقصود بـ database_name هو اسم قاعدة البيانات المطلوب إنشائها.

مثال :

لتكوين قاعدة بيانات باسم student_db فإننا نستخدم جملة SQL التالية:

```
CREATE DATABASE Student_db
```

إنشاء جدول CREATE TABLE

لإنشاء جدول بقاعدة بيانات ما فإن تركيبة جملة SQL تكون كالتالي:

```
CREATE TABLE  table_name
(
   Column_name1 data_type,
   Column_name2 data_type,
   …….
)
```

حيث أن:

table_name: هو اسم الجدول المطلوب إنشائه.
column_name: هو اسم العمود المراد إنشائه بالجدول.
data_type: هي نوع العمود الذي يتم إنشائه بالجدول.

مثال :

المثال القادم يوضح إنشاء جدول اسمه Person بأربعة أعمدة كالتالي:
First_name، Last_name، Address، Age

```
CREATE TABLE  Person
(
   First_name varchar,
   Last_name varchar,
   Address varchar,
   Age int
)
```

نلاحظ بالمثال بأننا قمنا بإعطاء النوع int لحقل العمر لأننا نريد ذكر قيمة رقمية تمثل العمر بينما استعملنا varchar وذلك لتمثيل سلاسل حرفية متمثلة في كلاً من الاسم الأول والاسم الأخير وعنوان السكن ، وهنا لم نحدد طول أي حقل من هذه الحقول، ولتحديد أكبر طول لبعض هذه الأعمدة علي سبيل المثال حقلي الاسم الأول و العمر فإننا نستعمل التالي:

```
CREATE TABLE  Person
(
   First_name varchar(30),
   Last_name varchar,
   Address varchar,
   Age int(3)
)
```

أنواع البيانات Data types:

Data types تحدد ما نوع البيانات التي يمكن أن توضع في الأعمدة داخل الجدول Table. والجدول التالي يوضح هذه الأنواع الموجودة في SQL:

النوع Data Type	الوصف
integer(size) int(size) smallint(size)	يمكن أن توضع بها القيم الصحيحة فقط وبدون كسور عشرية. size تستبدل بوضع أعلي قيمة من الأرقام التي يمكن أن توضع.
decimal(size, d) numeric(size, d)	يمكن أن توضع بها القيم الرقمية التي تحتوي علي كسور عشرية. (size) تستبدل بوضع أعلي قيمة من الأرقام التي يمكن أن توضع بالرقم بينما يوضع في (d) عدد الأرقام التي توضع علي يمين العلامة العشرية.
char(size)	يوضع بها طول محدد من السلسلة الحرفية التي تشمل علي حروف وأرقام ورموز خاصة، ويوضع الحجم الثابت مابين الأقواس.
varchar(size)	يوضع بها طول غير محدد من السلسلة الحرفية التي تشمل علي حروف وأرقام ورموز خاصة، ويوضع أعلي حجم يمكن وضعه في السلسلة مابين الأقواس.
Date(yyyymmdd)	يمكن أن يوضع بها التاريخ yyyy : للسنة mm : للشهر dd : لليوم

ولكن ما هو الفرق بين char و varchar ؟

char(10) : لو تم حجز10 حروف فإنها سوف تحجز مكان حتى ولو لم تستخدم هذه الخانة. فمثلاً :

Good------

varchar(10) : لو تم حجز10 حروف فإنها سوف تحجز مكان علي حسب القيمة المدخلة أو المستخدمة. فمثلاً :

Good

إنشاء الفهارس CREATE INDEX

الفهرس يصمم في الجداول حتي يجعل عملية الاستعلام أسرع كما يمكن إنشاء أكثر من فهرس علي عمود أو أكثر بالجدول الواحد بحيث يوجد اسم لكل فهرس ليدل عليه ويذكر عند استخدامه. المستخدم لايري هذه الفهارس ولكن يلاحظ سرعة كبيرة في الحصول علي الاستعلام المطلوب عند استعمال هذه الفهارس.

هناك نوعين من الفهارس: النوع الأول لا يمكن أن يتكرر به البيانات فعلي سبيل المثال رقم البطاقة الشخصية لا يمكن أن يتكرر، أما النوع الثاني فيمكن تكرار البيانات به، فعلي سبيل المثال اسم أو رقم المدرسة بجدول الطلاب يمكن أن يتكرر بعدد الطلاب الموجودين بالمدرسة المذكورة.

اولاً: بناء فهرس من النوع الفريد (UNIQUE INDEX)

لإنشاء فهرس من هذا النوع فإن تركيبة جملة SQL تكون كالتالي:

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

حيث أن:

index_name: هو اسم الفهرس المراد إنشائه.
table_name: هو اسم الجدول.
column_name: هو اسم العمود المراد الفهرسة علي أساسه.

ثانياً: بناء فهرس من النوع العادى (SIMPLE INDEX)

لإنشاء فهرس من هذا النوع فإن كلمة UNIQUE تهمل ولا توضع بالجملة ولهذا فإن القيم يمكن تكرارها وتكون تركيبة جملة SQL كالتالي:

```
CREATE INDEX index_name
ON table_name (column_name)
```

حيث أن:

index_name: هو اسم الفهرس المراد إنشائه.
table_name: هو اسم الجدول.
column_name: هو اسم العمود المراد الفهرسة علي أساسه.

مثال :

لتكوين SIMPLE INDEX يسمي PERSONINDEX علي الاسم الأخير LASTNAME بجدول PERSON نقوم بوضع التالي:

```
CREATE INDEX PERSONINDEX
ON PERSON (LASTNAME)
```

حيث تم فهرسة الجدول بحسب ترتيب الاسم الأخير تصاعدياً و يمكن وضع كلمة ASC أو إهمالها والتي تعني أن الترتيب تصاعدياً مثل:

```
CREATE INDEX PERSONINDEX
ON PERSON (LASTNAME ASC)
```

ولإنشاء الفهرس بترتيب عكسي {تنازلي} يمكن إضافة الكلمة المحجوزة DESC والتي هي اختصار لـ Descending order {ترتيب عكسي} و تستخدم كالتالي:

```
CREATE INDEX PERSONINDEX
ON PERSON (LASTNAME DESC)
```

مع التأكيد علي ضرورة وضع كلمة DESC عند الرغبة في الفهرسة بطريقة تنازلية.

ولإنشاء فهرس به أكثر من عمود يمكنك وضع أسماء الأعمدة داخل الأقواس ( ) مفصول كل اسم عن الأخر بفاصلة (,) والمثال التالي يوضح ذلك:

مثال:

لإنشاء فهرس بحقلين داخل الجدول وهما الاسم الأول و الاسم الأخير فإننا نقوم بعمل التالي:

```
CREATE INDEX PERSONINDEX2
ON PERSON (FIRSTNAME, LASTNAME)
```

DROP :

ترجمتها الحرفية إلقاء ، ولكن نستخدمها كأمر حذف. ويمكن حذف قاعدة بيانات أو فهرس أو جدول.

حذف فهرس (DROP INDEX)

يختلف بناء أمر حذف الفهرس بحسب نوع قاعدة البيانات كالتالي:

- نوع قاعدة البيانات MICROSOFT ACCESS

```
DROP INDEX index_name
ON table_name
```

- نوع قاعدة البيانات MS SQL SERVER

```
DROP INDEX table_name.index_name
```

- نوع قاعدة البيانات DB2 or ORACLE

```
DROP INDEX index_name
```

- نوع قاعدة البيانات MYSQL

```
ALTER TABLE table_name DROP INDEX index_name
```

حيث أن:

index_name: هو اسم الفهرس المراد إلغائه.

table_name: هو اسم الجدول.

حذف جدول (DROP TABLE)

لحذف جدول من قاعدة البيانات ليشمل الحذف جميع مكونات الجدول من حيث بناء الجدول وخواصه والفهارس التي تم إنشائها له. وتكون بناء الجملة كالتالي:

```
DROP TABLE table_name
```

حيث أن table_name هو اسم الجدول المطلوب حذفه.

حذف بيانات من داخل جدول (TRUNCATE TABLE)

لحذف البيانات من داخل جدول مع الاحتفاظ ببنية وخواص وفهارس الجدول فإننا نستخدم الأمر TRUNCATE ويكون بناء الجملة كالتالي:

```
TRUNCATE TABLE table_name
```

حيث أن table_name هو اسم الجدول المطلوب حذف البيانات داخله.

حذف قاعدة البيانات (DROP DATABASE)

لحذف قاعدة البيانات فإن بناء الجملة تكون كالتالي:

```
DROP DATABASE database_name
```

والمقصود بـ database_name هو اسم قاعدة البيانات المطلوب حذفها.

تعديل بجدول (ALTER TABLE)

يستخدم أمرALTER في أي عملية تعديل علي بنية جدول من حيث إضافة أو حذف أعمدة (حقول) بالجدول، ففي حالة إضافة حقل لجدول فإن بناء الجملة تكون كالتالي:

```
ALTER TABLE table_name
ADD column_name datatype
```

حيث أن:

table_name: هو اسم الجدول المطلوب التعديل بحقوله.
column_name: هو اسم العمود المراد إضافته بالجدول.
data_type: هي نوع العمود الذي سيتم إضافته بالجدول.

وفي حالة حذف حقل من الجدول لم نعد بحاجة إليه فإن تركيبة جملة SQL تكون كالتالي:

```
ALTER TABLE table_name
DROP COLUMN column_name
```

حيث أن:

table_name: هو اسم الجدول المطلوب حذف حقل من حقوله.
column_name: هو اسم العمود المراد حذفه من الجدول.

مثال :

يوجد لدينا جدول PERSON الذي يحتوي علي أسماء مجموعة من سكان مدينة طرابلس

FIRSTNAME	LASTNAME	ADDRESS
فتحي	ناصر	بن عاشور
علي	منصور	زاوية الدهماني

ونريد إضافة حقل للمدينة لنستطيع إدراج بيانات سكان المدن الاخري داخل الجدول بحيث يمكننا تمييز سكان كل مدينة علي حدة عند استعمال أمر SELECT، لذلك فإننا سوف نضيف الحقل CITY للجدول كالتالي:

```
ALTER TABLE PERSON
ADD CITY VARCHAR(30)
```

لذلك فإن الجدول يصبح بهذا الشكل:

FIRSTNAME	LASTNAME	ADDRESS	CITY
فتحي	ناصر	بن عاشور	
علي	منصور	زاوية الدهماني	

ملاحظة: يمكنك تعديل المدينة لكل من بيانات { فتحي و علي} باستخدام أمر UPDATE كما سنري بالمحاضرات القادمة.

أمثلة مختلفة:

1. المطلوب إلغاء العمود Age من جدول الموظفين Employee ؟

```
ALTER TABLE Employee  DROP COLUMN Age
```

2. المطلوب حذف قاعدة البيانات DAMAN ؟

```
DROP DATABASE DAMAN
```

3. المطلوب تكوين فهرس فريد باسم Num في الجدول Student وذلك اعتمادا علي الحقل Stu_no ؟

```
CREATE  UNIQUE  INDEX  Num ON Student (Stu_no)
```

4. المطلوب حذف بيانات جدول الفصل الدراسي Class_A استعدادا لتعبئته ببيانات الفصل الحالي ؟

```
TRUNCATE  TABLE  Class_A
```

ثانياً: لغة التعامل مع البيانات(SQL Data Manipulation Language (DML

كما عرفنا بالمحاضرة الأولي فإن DML هو قسم اللغة المسئول عن معالجة البيانات وتتكون من الأوامر التالية:

1. إضافة بيانات للجدول INSERT INTO
2. تعديل بيانات بالجدول UPDATE
3. حذف بيانات من الجدول DELETE
4. استخراج البيانات والمعلومات من الجداول بقاعدة البيانات SELECT

أولاً: إضافة بيانات للجدول باستخدام جملة: INSERT INTO

يمكن إضافة صف كامل لجدول باستخدام جملة INSERT INTO ويكون بناء الجملة كالتالي:

```
INSERT INTO table_name
VALUES (value1, value2, …..)
```

أو يمكن تحديد الحقول المطلوب إضافتها فقط ، وتكون الجملة كالتالي:

```
INSERT INTO table_name (column1, column2, ……)
VALUES (value1, value2, …..)
```

حيث أن:

table_name: هو اسم الجدول المطلوب إدخال البيانات به.

column: هو اسم العمود المراد إدخال القيمة (value) به، حيث أن column1 هو العمود الأول ، column2 هو العمود الثاني، value1 هي القيمة المطلوب إدراجها بالعمود الأول ، value2 هي القيمة المطلوب إدراجها بالعمود الثاني وهكذا...

مثال: لإضافة صف جديد لكل أعمدة الجدول

لنفرض أن لديك الجدول التالي Person

No	FirstName	LastName	Address	City
6	احمد	مجدي	سوق الجمعة	طرابلس

ولإضافة صف جديد لهذا الجدول بالبيانات التالية:

9 ، علي ، كمال ، بن عاشور ، طرابلس

نقوم بوضع جملة SQL :

```
INSERT INTO Person
VALUES ( 9,  'علي', 'كمال' , ' بن عاشور' , 'طرابلس')
```

وبعد تنفيذ جملة SQL السابقة فإن النتيجة سوف تكون بهذا الشكل:

No	FirstName	LastName	Address	City
6	احمد	مجدي	سوق الجمعة	طرابلس
9	علي	كمال	بن عاشور	طرابلس

مثال: لإضافة صف جديد لأعمدة (حقول) محددة بالجدول

لنفرض أن لديك الجدول التالي Person

No	FirstName	LastName	Address	City
6	احمد	مجدي	سوق الجمعة	طرابلس
9	علي	كمال	بن عاشور	طرابلس

ولإضافة بيانات (فوزي) المقيم في بنغازي والذي رقم بطاقته الشخصية 90 لهذا الجدول ، نقوم بوضع جملة SQL التالية:

```
INSERT INTO Person (No, FirstName, City)
VALUES ( 90, 'فوزي' ,' بنغازي')
```

وبعد تنفيذ جملة SQL السابقة فإن النتيجة سوف تكون بهذا الشكل:

No	FirstName	LastName	Address	City
6	احمد	مجدي	سوق الجمعة	طرابلس
9	علي	كمال	بن عاشور	طرابلس
90	فوزي			بنغازي

حيث أن القيم الافتراضية للحقلين (الاسم الأخير والعنوان) بالجدول Person هي الفراغات Spaces.

مثال:

لنفرض انك تريد إدخال البيانات التالية للجدول السابق:

الاسم : مفتاح الرقم : 70 العنوان : الحي الإسلامي

نستخدم الجملة التالية:

```
INSERT INTO Person
VALUES ( 70, ' مفتاح', null , ' الحي الإسلامي', null)
```

بالجملة السابقة إستخدمنا العبارة الصريحة null والتي تضع قيمة null إلي العمود المقابل لها.

ملاحظة:

يمكن تخصيص قيم افتراضية للحقول بالجدول، ففي حالة إدخال قيم لبعض أعمدة الجدول ولم يتم وضع قيم للأعمدة الأخرى، فإن الأعمدة المهملة تحتوي علي القيم الافتراضية default التي تم تخصيصها لأعمدة الجدول عند تصميمه، ويتم تعبئة هذه الحقول بالقيم الافتراضية ألياً. ولكن إذا لم يتم تخصيص القيم الافتراضية لهذه الحقول فإن قيمة null (والتي تعني قيمة مجهولة أو قيمة غير قابلة للتطبيق) يتم تخصيصها لهذه الحقول أو يمكن تخصيص null صراحة كما بالمثال السابق.

لاحظ أن ناتج جملة SQL السابقة مكافئ لناتج الجملة التالية:

```
INSERT INTO Person (No, FirstName, Address)
VALUES ( 70, ' مفتاح', 'الحي الإسلامي')
```

ملاحظة هامة:

ترتيب أسماء الأعمدة مع جملة INSERT يمكن أن يختلف عن الترتيب الأصلي لهذه الأعمدة الذي تم تحديده في جملة CREATE TABLE. وبهذه الحالة لابد أن تسرد هذه الأعمدة بالترتيب الجديد مثل ما هو موضح بالمثال التالي:

```
INSERT INTO Person (Address, FirstName, No)
VALUES ( 'الحي الإسلامي', ' مفتاح', 70 )
```

ملاحظة هامة:

إذا كان لديك بعض من البيانات مأخوذة من جدول أخر (أو مجموعة من الجداول) ، هذه البيانات يمكن استخدامها ليتم إدخالها بالجدول الذي لديك. لذلك نستخدم جملة الاستعلام والتي نتيجتها مجموعة من الحقول يتم إدخالها. وتكون الصيغة العامة لها:

```
INSERT INTO table_name (column1, column2, ……)
Query
```

فمثلاً قمت بتكوين جدول للموظفين القدامى بالشركة بالجملة التالية:

```
create table OLDEMP ( ENO number(4) not null,
WDATE date)
```

الآن يمكنك استخدام جدول الموظفين EMP وأخذ بيانات الموظفين الذين يكون تاريخ تعيينهم بالشركة قبل 31 ديسمبر من سنة 1960 علي سبيل المثال.علما بان حقل تاريخ التعيين بجدول الموظفين هو WorkDate. تكون الجملة كالتالي:

```
INSERT INTO  OLDEMP (ENO, WDATE)
Select   EMPNO, WorkDate from EMP
where WorkDate < '31-DEC-60'
```

ثانياً تعديل بيانات بالجدول باستخدام جملة: UPDATE

تستخدم جملة UPDATE لتعديل البيانات داخل الجدول table ويكون بناء الجملة كالتالي:

```
UPDATE table_name
SET column_name1= new_value [, {column_name2= new_value}….]
WHERE column_name= some_value
```

حيث أن:

table_name: هو اسم الجدول المطلوب تعديل البيانات به.

Set: تشير للأعمدة التي يتم التعديل بها والقيم التي توضع بها.

Where: تستخدم لتطبيق شرط معين بالجملة باستخدام معاملات المقارنة مثل >،<،=،>،=،! . يمكن أن يكون هناك مجموعة من الشروط المعقدة باستخدام and،or ،not .

مثال:

لنفرض أن لديك الجدول التالي Person

No	FirstName	LastName	Address	City
6	احمد	مجدي		
9	علي	كمال		طرابلس

*** التعديل بعمود واحد في صف:**

نقوم بوضع الجملة التالية:

```
UPDATE Person
SET Address= 'بن عاشور '
WHERE No= 9
```

والنتيجة تكون كالتالي:

No	FirstName	LastName	Address	City
6	احمد	مجدي		
9	علي	كمال	بن عاشور	طرابلس

*** التعديل بأعمدة مختلفة في صف:**

يمكنك تعديل أكثر من عمود في نفس الوقت، مثلاً تريد تعديل العنوان Address واسم المدينة City بالمثال السابق لبيانات (أحمد) ، لذلك نقوم بوضع جملة SQL التالية:

```
UPDATE Person
SET Address= سوق الجمعة ', City='طرابلس '
WHERE No= 6
```

والنتيجة تكون كالتالي:

No	FirstName	LastName	Address	City
6	احمد	مجدي	سوق الجمعة	طرابلس
9	علي	كمال	بن عاشور	طرابلس

مثال:
لنفرض أن لديك جدول العاملين Employee كالتالي:

No	FirstName	Task	Money	City
60	سفيان	Null	300	بنغازي
90	فائز	عامل	150	طرابلس

ونريد تغيير وظيفة الموظف الذي رقمه الوظيفي 60 والمقيم بمدينة بنغازي إلي وظيفة مدير للشركة حيث ان وظيفته غير معروفة وتساوي null الان. لهذا نقوم بوضع جملة SQL التالية:

```
UPDATE Employee
SET Task= 'مدير ' WHERE No= 60 AND City='بنغازي '
```

والنتيجة تكون كالتالي:

No	FirstName	Task	Money	City
60	سفيان	مدير	300	بنغازي
90	فائز	عامل	150	طرابلس

مثال:
لنفرض أن لديك جدول العاملين Employee السابق ونريد أن نضيف 50 دينار لكل موظف، فما هو الحل ؟
نستخدم جملة SQL التالية:

```
UPDATE Employee
SET Money=Money+50
```

لاحظ ان جميع الصفوف في جدول Employee سوف يجري عليها التعديل والسبب هو عدم استعمالنا لـ WHERE.

ولتتمكن من إظهار النتيجة أمامك ما عليك إلا أن تنفذ جملة SQL التالية:

```
SELECT * FROM Employee
```

والنتيجة تكون كالتالي:

No	FirstName	Task	Money	City
60	سفيان	مدير	350	بنغازي
90	فائز	عامل	200	طرابلس

ثالثاً: حذف بيانات من الجدول باستخدام جملة: DELETE

تستخدم جملة DELETE في حذف الصفوف من الجدول ، وبنائها يكون كالتالي:

```
DELETE FROM table_name
WHERE column_name= some_value
```

حيث أن:
From: تستخدم لاختيار الجدول الذي نختار منه السجلات المراد إلغائها.
table_name: هو اسم الجدول المطلوب حذف صفوف منه.
Where: تستخدم لتطبيق شرط معين بالجملة باستخدام معاملات المقارنة مثل > ، < ، = ، =! . يمكن أن يكون هناك مجموعة من الشروط المعقدة باستخدام and ، or ، not .

*** إلغاء صف:**
لنفرض أن لديك جدول العاملين Employee كالتالي:

FirstName	Task	Money	City
سفيان	null	300	بنغازي
فائز	عامل	150	طرابلس

ونريد إلغاء بيانات سفيان ، لذلك فإننا نستخدم الجملة التالية:

```
DELETE FROM Employee
WHERE FirstName= 'سفيان'
```

والنتيجة تكون كالتالي:

FirstName	Task	Money	City
فائز	عامل	150	طرابلس

*** إلغاء جميع البيانات (الصفوف) من الجدول:**

من المحتمل أن تقوم بإلغاء جميع الصفوف بجدول بدون إلغاء هذا الجدول. وهذا يعني أن بناء الجدول وخواصه وكذلك الفهارس الموجودة به تحفظ سليمة بدون أن تلغي. والصيغة العامة لجملة DELETE تكون كالتالي:

```
DELETE FROM table_name
او
DELETE * FROM table_name
```

مثال:

لنفرض أن لديك الجدول التالي Person

No	FirstName	LastName	Address	City
6	احمد	مجدي	سوق الجمعة	طرابلس
8	هالة	وائل	جنزور	الجفارة
78	ندي	علي	سرت المركز	سرت
9	علي	كمال	بن عاشور	طرابلس

والمطلوب إلغاء جميع الأشخاص المقيمين بمدينة طرابلس، لهذا ستكون جملة SQL كالتالي:

```
DELETE FROM Person
WHERE City= 'طرابلس'
```

والنتيجة تكون كالتالي:

No	FirstName	LastName	Address	City
8	هالة	وائل	جنزور	الجفارة
78	ندي	علي	سرت المركز	سرت

لو وضعت الجملة السابقة بدون WHERE كالتالي:

```
DELETE FROM Person
```

فالنتيجة تكون مغايرة كالتالي:

No	FirstName	LastName	Address	City

حيث أن البيانات كلها قد مسحت من الجدول.

ملاحظة:

يوجد فرق كبير بين DELETE و DROP TABLE :

DELETE تلغي جزئياً أو كلياً مكونات الجدول . من ناحية أخري فان DROP TABLE تلغي كل من مكونات وبنية الجدول. لذلك عند استعمال DELETE فإن الجدول يبقي موجوداً في قاعدة البيانات (احتمال أن يوجد به صفر من الصفوف) .

ولكن عند استعمال DROP TABLE فإن الجدول بأكمله يلغي من قاعدة البيانات.

توجد أنواع من SQL تدعم جملة TRUNCATE TABLE . وهذه الجملة تعطي تنفيذ أسرع من جملة DELETE بدون شرط WHERE ، لإنها تقوم بحذف مكونات الجدول صفحة بعد صفحة في حين أن DELETE تمسح مكونات الجدول سطر بعد سطر. والصيغة العامة لجملة TRUNCATE هي :

```
TRUNCATE TABLE table_name
```

رابعاً: استخراج بيانات ومعلومات باستخدام جملة:SELECT

تتيح لك جملة Select استجلاب البيانات التي ترغب في استرجاعها من قاعدة البيانات وتمكنك من تحديد كيفية ترتيبها ، وتمكنك أيضاً من إجراء عمليات حسابية علي البيانات المرجعة ، وغيرها من الأشياء. وتكتب جملة Select في أبسط أشكالها كالتالي:

```
SELECT column_name(s)  FROM table_name
```

حيث أن:

column_name(s): يوضع بعد كلمة Select اسم العمود أو الأعمدة داخل الجدول مفصول بينهم بفاصلة إذا كان عددهم أكثر من واحد ويمكن أن تكون علامة * وتعني جميع الأعمدة داخل الجدول. ومن الممكن وجود تعبيرات حسابية مع أعمدة الجدول.

From: تحدد المكان الذي نريد استعادة البيانات منه (جدول أو أكثر).

table_name: اسم الجدول المراد الاستعلام منه.

مثال:

نفرض أن جدول الأشخاص Persons كالتالي:

Firstname	Lastname	City	Address	Sal
Ali	Eletri	Tripoli	Sog eljommaa	100
siham	Eletri	Sabha	Sabha center	200
Abd elrazag	Shlebk	Bengazi	Omar el mokhtar	100
Aisha	Krekshi	Tripoli	Mezran	200

وقمت بإنشاء جملة SQL التالية:

```
Select Firstname, Lastname from persons
```

يعني أعرض جميع مكونات الاسم الأول والاسم الأخير من جدول الأشخاص Persons و الناتج هو:

Firstname	Lastname
Ali	Eletri
Siham	Eletri
Abd elrazag	Shlebk
Aisha	Krekshi

والجملة التالية:

```
Select * from Persons
```

تعني أعرض جميع مكونات جميع الأعمدة من جدول الأشخاص والناتج هو كالتالي:

Firstname	Lastname	City	Address	Sal
Ali	Eletri	Tripoli	Sog eljommaa	100
siham	Eletri	Sabha	Sabha center	200
Abd elrazag	Shlebk	Bengazi	Omar el mokhtar	100
Aisha	Krekshi	Tripoli	Mezran	200

مثال:
إذا طلب منك عرض الاسم الأول والاسم الأخير وعرض المرتب مضاف إليه 30 دينار فتكون جملة SQL كما يلي:

```
Select Firstname, Lastname, SAL +30 from Persons
```

جملة Select Distinct

تستخدم كلمة Distinct مع جملة Select لاسترجاع الكلمات المختلفة فقط من عمود محدد وتكتب بالشكل التالي:

```
Select DISTINCT column_name(s) From table_name
```

حيث أن:

column_name(s): اسم العمود أو الأعمدة داخل الجدول مفصول بينهم بفاصلة إذا كان عددهم أكثر من واحد.

table_name: اسم الجدول المراد الاستعلام منه.

فإضافة كلمة Distinct للجملة تمكنك من عرض القيم بدون تكرار أي لو كان في العمود داخل الجدول توجد بيانات متكررة كما في عمود Address في الجدول التالي ، فوضع هذه الكلمة Distinct قبل اسم العمود يعني عرض محتويات العمود بدون تكرار.

مثال:
نفرض أن جدول الأشخاص person كالتالي:

Employee_id	First_name	Last_name	Address	Department
12578	علي	فرج	بن عاشور	الحاسب
13568	محمود	الزائد	جنزور	الشئون الإدارية
15963	عائشة	المرغني	حي الأندلس	الحاسب
15991	مريم	محمودي	جنزور	المالية
16657	كريم	عبدالسلام	الدريبي	الحاسب

لو طلب منك مدير المؤسسة التي تعمل بها قائمة بعناوين الموظفين لغرض توفير وسيلة مواصلات لكل عنوان على حدي فماذا تفعل؟

الحل:
لو استخدمت جملة الاستعلام التالية:

```
Select Address from Person
```

سنعرض في هذه الحالة جميع عناوين الموظفين وتكون القائمة المعدة بالشكل التالي:

Address
بن عاشور
جنزور
حي الأندلس
جنزور
الدريبي

ماذا لو كان عدد الموظفين في الشركة 100 موظف تخيل كيف ستكون القائمة ؟ !! على كل حال أردت أخذ النتيجة إلى الشخص المسئول فاستوقفك أحد العاملين بقسم المواصلات قائلاً ما هذه القائمة الغريبة نريد العنوان بدون تكرار أي **عرض القيم المختلفة فقط** لتخصيص سيارة لكل منطقة. ماذا تفعل لعرض القائمة بالمناطق دون تكرار؟

الحل يكمن في استخدام كلمة Distinct بالشكل التالي:

```
Select Distinct Address from Person
```

فيكون الناتج كما نريد، تم عرض محتويات العمود Address بدون عرض التكرار الموجود كالتالي:

Address
بن عاشور
جنزور
حي الأندلس
الدريبي

مثال:

لعرض الأقسام **Department** فقط دون تكرار من الجدول السابق

```
Select Distinct Department from Person
```

الناتج كالتالي:

Department
الحاسب
الشئون الإدارية
المالية

خيارات Select (Select Options)

- **بند Where**
- **بند Order By**
- **بند Group By**
- **بند Having**

i. بند Where:

حتى الأن فإننا نقوم باختيار بعض الحقول للسجلات من الجدول لعرضها ، ولكن إذا طلب منك اختيار سجلات بمواصفات أو شروط معينة فإنك تضطر لاستخدام بند Where.

في بند Where توجد شروط بسيطة تعتمد علي معاملات المقارنة (> ، < ، =). فمثلاً Salary >=500 ، هو عبارة عن شرط يوضع بعد بند Where وناتج هذا الشرط صح (True) أو خطأ (False).

هذه الشروط تجمع لتكون شروط معقدة باستخدام المعاملات المنطقية (And ، Or ، Not). وتكون جملة SQL بالشكل التالي:

```
Select column_name(s)
From table_name
Where column_name operator value
```

حيث أن:

column_name(s): اسم العمود أو الأعمدة داخل الجدول مفصول بينهم بفاصلة إذا كان عددهم أكثر من واحد ويمكن أن تكون علامة * وتعني جميع الأعمدة داخل الجدول.
table_name: اسم الجدول المراد الاستعلام منه.
column_name: اسم عمود داخل الجدول.
value: القيمة المراد الاستعلام عنها داخل العمود الذي كتب اسمه في بندWhere .
Operator: المعامل ويمكن أن يكون أحد المعاملات التالية:

المعامل Operator	الوصف
=	يساوي
!=، < >	لا يساوي
>	أكبر من
<	أصغر من
> =	أكبر من أو يساوي
< =	أصغر من أو يساوي
Between	القيمة في نطاق محدد "بين"
IN	قيم محددة
Like	للبحث عن كلمة مشابهة

مثال:

نفرض انك تعمل في شركة أراد المسئول بها قائمة بجميع بيانات العاملين المتواجدين في مدينة طرابلس وكان ملف الموظفين Employee بالشكل التالي:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

فإننا نستخدم جملة Select بالشكل التالي:

```
Select * From employee
Where City='طرابلس '
```

الناتج يكون بالشكل التالي:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
15963	عائشة	المرغني	طرابلس	500

ملاحظة:

نلاحظ استخدام علامة التنصيص ' مع القيم النصية مثل كلمة طرابلس في المثال السابق. لو كانت الجملة السابقة هكذا:

```
Select * from employee where City= طرابلس
```

الجملة تعتبر خطأ لابد من كتابة كلمة طرابلس بالشكل التالي 'طرابلس ' وذلك ينطبق على أي قيمة نصية.

مثال:
لو أردنا قائمة بأسماء العاملين وقيمة المرتب للعاملين الذين يقل مرتبهم أو يساوي 300 دينار لغرض صرف إعانة بمناسبة عيد الأضحى.

الحل:

```
Select first_name, last_name, Sal
From employee
Where sale <= 300
```

الناتج:

First_name	Last_name	Sal
علي	فرج	250
مريم	محمودي	300

تم صرف الإعانة للعاملين السالف ذكرهم مما أثار حفيظة بقية العاملين وطالبوا بصرف مكافأة لهم أيضاً فطلب منك المسئول قائمة بأسماء ومرتبات باقي العاملين.

الحل:
لإيجاد القائمة الصحيحة يجب عكس الشرط السابق حتى لا يتكرر نفس الأشخاص السابقين في القائمة الجديدة وعكس أصغر من أو يساوي هو أكبر من.

```
Select first_name, last_name, Sal
From employee
Where sale > 300
```

والناتج هو:

First_name	Last_name	Sal
محمود	الزائد	450
عائشة	المرغني	500
كريم	عبدالسلام	600

ملاحظة:
القيمة 300 في جملة Select ليست قيمة نصية بل هي قيمة رقمية لذلك من الخطأ أن تكتب بالشكل التالي:

```
Select first_name, last_name, Sal
From employee                    هذه الجملة خطأ
Where sale > '300'
```

استخدام معاملات الربط And & OR ومعامل النفي NOT:
يستخدم لمعاملي الربط (And & OR) في ربط شرطين أو أكثر في الجملة Where حيث أن:
المعامل And: يربط شرطين أو أكثر ويقوم بعرض الصف فقط إذا كانت بيانات الصف تخضع **لكل** الشروط الموضوعة (أي ان الشروط كلها صحيحة True).
المعامل OR: يربط شرطين أو أكثر ولكن يقوم بعرض الصف إذا كان **أي من** الشروط الموضوعة قيمتها صحيحة True.
المعامل NOT: ويقوم معامل النفي NOT بعرض النتائج في حالة عدم تحقق الشرط أي ينفي الشرط الداخلي إذا كان خطأ FALSE يقوم بتغييره إلى صح TRUE و العكس إذا كان صحيح TRUE يغيره إلى FALSE.

أولويات التنفيذ :

عند تواجد معاملات الربط ومعامل النفي بنفس الجملة فإن العمليات التي تنفذ أولاً هي بالترتيب التالي:

1. ما بين الأقواس
2. عملية NOT
3. عملية AND
4. عملية OR

أمثلة على المعامل And & Or:

المطلوب عرض كل الأشخاص المقيمين في مدينة طرابلس و مرتباتهم أقل من 300 من جدول الموظفين Employee

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

الحل:

```
SELECT * FROM   EMPLOYEE
WHERE CITY='طرابلس'  AND SAL<300
```

الناتج يكون بالشكل التالي:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250

نلاحظ أنه تم عرض الشخص الذي ينطبق عليه الشرطين فقط.

مثال:

أعرض من ملف الموظفين EMPLOYEE الأشخاص الذين مرتباتهم أكبر من أو تساوي 200 و أصغر من أو تساوي 500

الحل :

```
SELECT * FROM   EMPLOYEE
WHERE SAL>=200 AND SAL<=500
```

والناتج يكون بالشكل التالي:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300

مثال:

أعرض جميع الأشخاص الذين مرتباتهم تساوي 300 أو يقيمون في مدينة غريان ؟

الحل :

```
SELECT * FROM    EMPLOYEE
WHERE CITY='غريان'  OR SAL=300
```

والناتج يكون بالشكل التالي:

Employee_id	First_name	Last_name	City	Sal
13568	محمود	الزائد	غريان	450
15991	مريم	محمودي	سبها	300

مثال:
من ملف الموظفين EMPLOYEE أعرض الأشخاص الذين مرتباتهم لا تساوي 500
الحل:

```
SELECT * FROM  EMPLOYEE WHERE SAL < > 500
```

والناتج يكون بالشكل التالي:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	غريان	450
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

مثال:
إذا كنت احد المشرفين على قاعدة البيانات لمؤسسة الضمان الاجتماعي فرع سبها وطلب المسئول في فرع المؤسسة الرئيسي من جميع الفروع بليبيا إحضار قوائم بأسماء الأشخاص الذين مرتباتهم اكبر من 500 أو اصغر من 400 علماً بأن ملف الموظفين بالشكل التالي:

Employee

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

الحل الخطأ:

```
SELECT  FIRST_NAME, LAST_NAME   FROM   EMPLOYEE
WHERE  CITY='سبها' AND  SAL< 400 OR SAL>500
```

الناتج هو:

First_name	Last_name
مريم	محمودي
كريم	عبدالسلام

نلاحظ بالناتج السابق أنك أحضرت العاملين في الكفرة أيضاً (الموظف كريم) ، مما يسبب في تكرار في الأسماء مع القائمة المرسلة من فرع الكفرة.

والسبب هو عدم مراعاتك للأولويات في تنفيذ العمليات، ففي الجملة السابقة سيتم تنفيذ شرط AND أولاً تم شرط OR فينفذ جميع العاملين في سبها ومرتباتهم اقل من 400 أولاً والناتج هو (الموظفة مريم) ، ثم ينفذ الشرط الأخر هو OR بمعني أو المرتب أكبر من 500 و يكون الناتج هو (الموظف كريم).

كان تسلسل التنفيذ **ثانياً** **أولاً**

(CITY='سبها' AND SAL< 400) OR SAL>500

الحل الصحيح يكون باستخدام الأقواس لتنفيذ العمليات حسب الأولوية التي نريد فالذي نريده هو:

أولاً **ثانياً**

CITY='سبها' AND (SAL< 400 OR SAL>500)

ينفذ العملية بين الأقواس أولاً تم يقارنها بالعملية خارج القوس ثانياً ففي هذه الحالة سيقارن البيانات إذا كان ينطبق عليها الشرط بين الأقواس وهو المرتب اصغر من 400 أو اكبر من 500 فيكون الناتج هو:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

ثم ينفذ الشرط الأول خارج الأقواس وهو ضرورة أن تكون المدينة سبها فيكون الحل النهائي هو:

Employee_id	First_name	Last_name	City	Sal
15991	مريم	محمودي	سبها	300

المعامل NOT

يستخدم معامل النفي NOT لنفي الشرط الذي يليه ويخضع أيضا لشروط الأولويات التي تحدثنا عنها سابقاً.

مثال : أحضر العاملين الذين لا يقيمون في مدينة غريان ؟

الحل:

```
SELECT   * FROM EMPLOYEE WHERE NOT CITY='غريان'
```

الناتج هو:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

مثال:

من جدول الموظفين أحضر الموظفين الذين مرتباتهم تساوي 600 أو أكبر من أو يساوي 300 في القائمة الأولى؟ تم أحضر باقي الموظفين في قائمة ثانية باستخدام معامل النفي NOT.

القائمة الأولى:

```
SELECT * FROM  EMPLOYEE
WHERE SAL=600 OR SAL>=300
```

الناتج هو:

Employee_id	First_name	Last_name	City	Sal
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

القائمة الثانية:

فلو أردنا عكس الشرط السابق فلابد من وضع الأقواس لتشمل عملية النفي كل الشرط ولا نكتفي بوضعها قبل الشرط.
الحل بدون أقواس وهو خطأ :

```
SELECT * FROM  EMPLOYEE
WHERE NOT SAL=600 OR SAL>=300
```

وناتج الحل الخطأ هو:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

لاحظ انه قد أحضر قائمة غريبة تتكرر محتوياتها مع القائمة الأولى وذلك لأن التنفيذ كان لا يساوي 600 لم يحضر السجل الذي يحتوي علي قيمة المرتب الذي يساوي 600 ، لكن الشرط الأخر لم يتم نفيه فأحضر الأكبر من أو تساوي 300 فشملت كل القيم الأكبر من 300 بما فيها 600 لأن الرابط هو OR (إذا انطبق أحد الشروط على بيانات يقوم بعرضها).

الحل الصحيح باستخدام الأقواس:

```
SELECT * FROM  EMPLOYEE
WHERE NOT (SAL=600 OR SAL>=300)
```

بمعنى (أي شرط يتحقق داخل الأقواس لا تحضره) ، وبهذه الحالة لا يحضر جميع الأشخاص الذين تم عرضهم في القائمة الأولى.

والقائمة الثانية الصحيحة هي:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250

العملية BETWEEN:

تستخدم لإحضار قيم ضمن نطاق محدد لتوفير كتابة الكثير من الكود وتكون بالشكل التالي:

```
SESLECT column_name (s)   FROM table_name
WHERE column_name BETWEEN small_value , big_value
```

حيث أن:

Column_name: إسم العمود داخل الجدول.
Table_name: إسم الجدول.
Small_value: أصغر قيمة في النطاق المطلوب.
big_value: أكبر قيمة في النطاق المطلوب.

مثال:
من جدول الموظفين أحضر قائمة بالموظفين الذين مرتباتهم أكبر من أو تساوي 400 و أصغر من أو تساوي 600 ؟

Employee

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

الحل:

```
SELECT * FROM EMPLOYEE
WHERE SAL BETWEEN 400, 600
```

الناتج هو:

Employee_id	First_name	Last_name	City	Sal
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500
16657	كريم	عبدالسلام	الكفرة	600

ويمكننا الحصول علي نفس النتيجة باستخدام الجملة التالية:

```
SELECT * FROM EMPLOYEE
WHERE SAL >=400 and  SAL <= 600
```

مثال:
المطلوب عرض الموظفين الذين مرتباتهم أكبر من 400 و أصغر من 600 ؟
الحل:

```
SELECT * FROM EMPLOYEE
WHERE SAL BETWEEN 401, 599
```

الناتج هو:

Employee_id	First_name	Last_name	City	Sal
13568	محمود	الزائد	غريان	450
15963	عائشة	المرغني	طرابلس	500

ولعرض باقي الموظفين الذين لاينطبق عليهم هذا الشرط تقوم بوضع:

```
SELECT * FROM EMPLOYEE
WHERE SAL Not BETWEEN 401, 599
```

العملية IN:
تستخدم لإحضار قيم محددة وتكون بالشكل التالي:

```
SESLECT column_name (s)   FROM table_name
WHERE column_name IN (value1 , value2, …)
```

حيث أن:
Column_name: اسم العمود داخل الجدول.

Table_name: اسم الجدول.
value: قيمة في الحقل.

مثال:
المطلوب عرض الموظفين الذين مرتباتهم 250 أو 500 أو600 ؟
الحل:

```
SELECT * FROM EMPLOYEE
WHERE SAL =250  OR  SAL=500  OR  SAL=600
```

أو

```
SELECT * FROM EMPLOYEE
WHERE SAL IN (250, 500, 600)
```

وناتج الجملتين السابقتين هو:

Employee_id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
15963	عائشة	المرغني	طرابلس	500
16657	كريم	عبدالسلام	الكفرة	600

ويمكن استخدام Not In لعكس النتيجة السابقة كالتالي:

```
SELECT * FROM EMPLOYEE
WHERE SAL NOT IN (250, 500, 600)
```

وناتج الجملة هو:

Employee_id	First_name	Last_name	City	Sal
13568	محمود	الزائد	غريان	450
15991	مريم	محمودي	سبها	300

معامل LIKE
يستخدم المعامل LIKE في بند الشرط WHERE للبحث عن كلمة معينة أو جزء من الكلمة فمثلاً للبحث عن جميع الأسماء التي تحتوي على حرف أو كلمة معينة.

```
SESLECT column_name (s)   FROM table_name
WHERE column_name LIKE pattern
```

حيث أن:
Column_name: اسم العمود داخل الجدول.
Table_name: اسم الجدول.
Pattern: الكلمة أو جزء من الكلمة المراد البحث عنها داخل العمود.

العلامات المستخدمة مع معامل like وبالتحديد مع pattern تستخدم في تحديد الحروف الغير مرغوب فيها قبل وبعد الكلمة التي نبحث عنها كالتالي:
% Percent: تحدد أي عدد من الحروف صفر أو أكثر.
_ Underscore: تحدد عدد حرف واحد فقط.
توجد عمليات أخرى لن نتطرق لها في هذا الفصل.

مثال:
اعرض جميع الأسماء التي تبدأ بحرف ع ؟
الحل:

```
SELECT FIRST_NAME, LAST_NAME FROM EMPLOYEE
WHERE FIRST_NAME LIKE 'ع %'
```

الناتج هو:

First_name	Last_name
علي	فرج
عائشة	المرغني

مثال:
أعرض قائمة للأشخاص التي تنتهي أسمائهم بحرف م ؟
الحل:

```
SELECT FIRST_NAME, LAST_NAME FROM EMPLOYEE
WHERE FIRST_NAME LIKE '% م'
```

مثال:
تم توظيف مدخل بيانات جديد قام بإدخال بيانات مدينة الكفرة ، وقد لوحظ عدم التزامه بتوحيد لفظ المدينة عند إدخاله إلى قاعدة البيانات ، طلب منك عرض قائمة بأرقام الموظفين و المدينة للأشخاص الذين يقيمون في مدينة الكفرة من ملف الموظفين EMPLOYEE الذي تم إدخاله مؤخراً وذلك لمراجعته لتوحيد الألفاظ في عمود المدينة .

Employee

Employee_id	First_name	Last_name	City	Sal
12578	عمار	فرج	طرابلس	250
13568	ريما	الزائد	الكفره	450
15963	عائشة	المرغني	كفرة	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

الحل:
نلاحظ أن مدينة الكفرة كتبت بأشكال عدة {كفرة ، الكفرة، الكفره} ، في هذه الحالة لا يمكن استخدام كلمة الكفرة لنحضر جميع المقيمين بها لأننا سنستثني {كفرة، الكفره} لذلك نبحث عن سلسلة أحرف أو تشكيلة حروف يشترك فيها الألفاظ الثلاث مع دلالاتها على المدينة نفسها فنبحث بالحروف المشتركة بين هذه الألفاظ وهي {كفر} وتكون الجملة كالتالي:

```
SELECT   EMPLOYEE_ID, CITY FROM EMPLOYEE
WHERE CITY LIKE '%كفر%'
```

وهذا يعنى أحضر من عمود المدينة أي كلمة تحتوي على الحروف كفر ومهما كان عدد الحروف التي قبلها و التي بعدها، والناتج يكون بهذا الشكل:

Employee_id	City
13568	الكفره
15963	كفرة
16657	الكفرة

استخدام علامة (%) مع علامة (_):

مثال:
أحضر قائمة بأسماء الموظفين من جدول الموظفين في المثال السابق الذين ثاني حرف في اسمهم الأول هو حرف الراء (ر).

الحل:

```
SELECT FIRST_NAME, LAST_NAME FROM EMPLOYEE
WHERE FIRST_NAME LIKE '_ر %'
```

والناتج يكون بهذا الشكل:

First_name	Last_name
مريم	محمودي
كريم	عبدالسلام

ii. بند Order By

تستخدم جملة Order By لترتيب النتائج حسب المطلوب وتكتب بالشكل التالي:

```
Select column(s) From table_name
Order By column1, .., column2, .. [Asc | Desc]
```

بعد كلمة Order By يتم وضع اسم العمود المراد الترتيب على أساسه وإذا كان أكثر من عمود يفصل بينهم بفاصلة.
Asc: تعني ترتيب تصاعدي.
Desc: تعني ترتيب تنازلي وإذا لم تكتب أي منهما يعتبر الترتيب تصاعدي تلقائياً. الأمثلة التالية توضح العملية أكثر.

مثال:
إذا كنت أحد العاملين على قاعدة البيانات في مؤسسة الضمان الاجتماعي وطلب منك قائمة بالمتقاعدين مرتبين حسب المرتب الأكبر مرتب والذي يليه وهكذا.

Customer

Id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	الكفرة	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

الحل:

```
Select first_name, lastname, sal From customer
Order By sal
```

وناتج جملة SQL هو:

First_name	Last_name	Sal
علي	فرج	250
مريم	محمودي	300
محمود	الزائد	450
عائشة	المرغني	500
كريم	عبدالسلام	600

إذا طلب منك ترتيب القائمة حسب المرتب تنازلياً الحل هو:

```
Select first_name, lastname, sal From customer
Order By sal Desc
```

وناتج جملة SQL هو:

First_name	Last_name	Sal
كريم	عبدالسلام	600
عائشة	المرغني	500
محمود	الزائد	450
مريم	محمودي	300
علي	فرج	250

مثال:
في نفس المثال السابق طلب منك طباعة قائمة للمتقاعدين حسب المدينة المتواجد بها وحسب المرتب تصاعدي من ملف الزبائن Customer

Customer

Id	First_name	Last_name	City	Sal
12578	علي	فرج	طرابلس	250
13568	محمود	الزائد	الكفرة	450
15963	عائشة	المرغني	طرابلس	500
15991	مريم	محمودي	سبها	300
16657	كريم	عبدالسلام	الكفرة	600

الحل:

```
Select city, first_name, lastname, sal From customer
Order By city, sal
```

وناتج جملة SQL هو:

City	First_name	Last_name	Sal
الكفرة	علي	فرج	450
الكفرة	محمود	الزائد	600
سبها	عائشة	المرغني	500
طرابلس	مريم	محمودي	250
طرابلس	كريم	عبدالسلام	500

في المثال السابق لو طلب منك إعداد القائمة السابقة مرتبة حسب الترتيب الأبجدي للمدينة تصاعدياً لكن أصحاب المرتبات المرتفعة تعرض بياناتهم قبل المرتبات المنخفضة.

الحل:

```
Select city, first_name, lastname, sal From customer
Order By city, sal Desc
```

وناتج جملة SQL هو:

City	First_name	Last_name	Sal
الكفرة	محمود	الزائد	600
الكفرة	علي	فرج	450
سبها	عائشة	المرغني	500
طرابلس	كريم	عبدالسلام	500
طرابلس	مريم	محمودي	250

ولو طلب منك ان يكون الترتيب تنازلياً حسب المدينة وتصاعدياً حسب المرتب يكون الحل:

```
Select city, first_name, lastname, sal From customer
Order By city Desc , sal Asc
```

وناتج جملة SQL هو:

City	First_name	Last_name	Sal
طرابلس	مريم	محمودي	250
طرابلس	كريم	عبدالسلام	500
سبها	عائشة	المرغني	500
الكفرة	علي	فرج	450
الكفرة	محمود	الزائد	600

الدوال SQL Functions

لغة SQL بها الكثير من الدوال العددية والحسابية ، والبناء الأساسي لأي دالة هو:

```
Select function(column_name) from table_name
```

حيث أن:
table_name : اسم الجدول.
column_name : اسم العمود.
Function : اسم الدالة.

أنواع الدوال:
توجد مجموعة مختلفة من الدوال functions في SQL يمكن تقسيمها إلي نوعين أساسيين هما:

- Aggregation Functions (دوال التجميع).
- Scalar Functions (الدوال العددية).

Aggregation Functions (دوال التجميع) :

تعمل علي مجموعة من القيم وترجع قيمة واحدة ، فعلي سبيل المثال الدوال التالية موجودة في SQL Server:

الدالة	الوصف
AVG(column)	لإيجاد الوسط الحسابي للحقل المحدد
AVG(DISTINCT column)	لمعرفة الوسط الحسابي في الحقل بدون التكرار
COUNT(*)	معرفة عدد الصفوف في الجدول (تشمل Null)
COUNT(DISTINCT column)	معرفة عدد الصفوف في الحقل بدون التكرار (لا تشمل Null)
COUNT(All column)	معرفة عدد الصفوف في الحقل مع التكرار (لا تشمل Null)
MAX(column)	معرفة أكبر قيمة سجل في الحقل
MIN(column)	معرفة أصغر قيمة سجل في الحقل
SUM(column)	معرفة إجمالي القيم في الحقل

حيث أن:
Column : اسم العمود.

Scalar Functions (الدوال العددية) :

تعمل علي قيمة منفردة وترجع قيمة واحدة بالاعتماد علي القيمة المدخلة ، فعلي سبيل المثال الدوال التالية موجودة في SQL Server:

الدالة	الوصف
Upper(c)	تحويل الحرف إلي Upper case
Lower(c)	تحويل الحرف إلي Lower case
LEN(c)	ترجع طول الحقل الحرفي
LEFT(c, number of char)	ترجع جزء اليسار المطلوب من الحقل الحرفي
RIGHT(c, number of char)	ترجع جزء اليمين المطلوب من الحقل الحرفي

حيث أن:
c : الحقل الحرفي.
number of char : عدد الحروف في الحقل الحرفي.

مجموعة من الأمثلة:
لنفرض أن لديك جدول الطلبة students وكان لديك عمود لدرجات الطلبة باسم Degree وعمود آخر لأسماء الطلبة باسم StuName وتريد استعمال مجموعة من الدوال السابقة:

StuName	Degree
احمد	10
احمد	20
علي	30
اسامة	40

الدوال المستعملة:

رقم جملة SQL	جمل SQL	المطلوب
1	Select Avg(degree) from students	المتوسط الحسابي للدرجات
2	Select Count(StuName) from students	عدد الطلبة
3	Select Max(degree) from students	اعلي درجة
4	Select Min(degree) from students	اقل درجة
5	Select Sum(degree) from students	مجموع درجات الطلبة
6	Select Count(Distinct StuName) from students	عدد الطلبة بدون تكرار الاسماء
7	Select left(StuName, 2) from students	حرفين من يسار الاسم
8	Select Right(StuName, 1) from students	حرف من اليمين
9	Select Len(StuName) from students	عدد حروف اسماء الطلبة

وناتج الجمل كالتالي:

رقم جملة SQL	ناتج جمل SQL
1	31
2	4
3	40
4	10
5	100
6	3
7	(No Column Name) اح اح عل اس

رقم جملة SQL	ناتج جمل SQL
8	(No Column Name) د د ي ة
9	(No Column Name) 4 4 3 5

ملاحظة:
يمكنك عمل مجموعة من الدوال مع بعضها البعض بنفس الجملة.

مثال:

```
select max(degree), min(degree), count(*)  from students
```

والناتج هو:

(No Column Name)	(No Column Name)	(No Column Name)
40	10	4

التجميع والفرز Having و SQL Group By

بند Group By و بند Having من ضمن خيارات جملة Select (Select Options) ، حيث درسنا بالسابق بند Where و بند Order by .

iii. بند Group By :

Group By تستخدم مع دوال التجميع حيث يتم ترجيع صف واحد لكل قيمة يتم تجميعها من الدوال فمثلاً الدالة Sum كثيراً ما تحتاج إلي التجميع Group By ، وبناء الجملة يكون كالتالي:

```
Select column_name, Sum(column_name) from table_name Group By column_name
```

حيث أن:
table_name : اسم الجدول.
Column_name : اسم العمود.

مثال:

إذا كان لدينا جدول المبيعات Sales كالتالي:

Product	Amount
طاولة	5500
كرسي	4500
طاولة	7100

ونريد أن نعرف كم لدينا من الطاولات والكراسي وغيرها من المنتجات بالمصنع ، فلو استعملنا جملة SQL التالية:

```
SELECT Product, SUM(Amount) from Sales
```

فان حل الجملة السابقة هو:

Product	(No Column Name)
طاولة	17100
كرسي	17100
طاولة	17100

نلاحظ إن جميع السجلات أخذت المجموع كله ولم نعرف مجموع كل سجل. فالحل لهذه المشكلة هو استخدام Group By بجملة الاستعلام كالتالي:

```
SELECT Product, SUM(Amount) from Sales group by Product
```

والحل هو:

Product	(No Column Name)
طاولة	12600
كرسي	4500

نلاحظ أنه تم تجميع كل سجل وحده وأصبحت النتيجة أوضح.

مثال:

لنفرض جدول الطلبة التالي Students الذي يوضح أسماء الطلبة Stu_name و رقم القيد Stu_no والفصل الدراسي لكل طالب Class والعنوان Address.

Stu_no	Stu_name	Class	Address
33	أحمد علي مسعود	الأول	بن عاشور
54	علي فريد محمد	الثالث	الدريبي
65	سلمي وائل جلال	الأول	الهاني

ونريد أن نعرف عدد الطلبة بكل فصل دراسي فإننا نقوم بالتالي:

```
Select Class, Count(*) from Students
Group by Class
```

والنتيجة تكون كالتالي:

Class	(No Column Name)
الأول	2
الثالث	1

iv. **بند Having :**

الدالة HAVING تستخدم لفرز البيانات حسب شرط معين وقد تم إضافتها إلي SQL لأن Where لا يمكن استعمالها مع Aggregation function مثل Sum. لذلك بدون استعمال HAVING لا يمكن اختبار نتائج الشروط ، وبناء الجملة يكون كالتالي:

```
SELECT column_name, Sum(column_name) from table_name
Group By column_name
Having SUM(column_name) condition value
```

حيث أن:

table_name : إسم الجدول.
Column : إسم العمود.
condition : اداة للمقارنة مثل >،< ،=،<>.
Value : القيمة التي يتم المقارنة بها.

مثال:

بجدول المبيعات Sales السابق نستخدم جملة SQL التالية:

```
SELECT Product, SUM(Amount) from Sales
group by Product
Having SUM(Amount)>10000
```

والحل هو:

Product	(No Column Name)
طاولة	12600

نلاحظ أنه تم تحقق الشرط وعرض البيانات التي أكبر من 10000.

مثال:

بجدول الطلبة السابق نريد أن نعرف كم عدد الفصول التي يزيد عدد طلابها عن 9 طلاب بافتراض أن هذا الجدول تم تعبئته ببيانات عدد كبير من الطلبة ، فجملة SQL تكون كالتالي:

```
Select Class, Count(*) from Students
Group by Class having count(*) > 9
```

أمثلة شاملة

لنفرض أن لديك الجدول Tables2

Emp	Name	Salary	Fun
1	فريد	200	المبيعات
2	مراد	300	المبيعات
3	سامي	400	المبيعات
4	مروان	350	التسويق
5	جمال	500	التسويق
6	سالم	800	المالية
7	تامر	200	الخدمات
8	سعيد	900	الخدمات

ولإيجاد متوسط مرتبات الموظفين بكل قسم نستخدم:

```
Select Fun, Avg(Salary) from Tables2
Group by Fun
```

والنتيجة تكون:

Fun	(No Column Name)
المبيعات	300
التسويق	425
المالية	800
الخدمات	550

بالمثال القادم نعرض أعلي قيمة مرتب لموظفي الأقسام ماعدا أسماء الموظفين التي تبدأ أسمائهم بحرف س.

```
Select Fun, Max(Salary) From Tables2
Where Name Not like 'س%'
Group by Fun
```

والنتيجة تكون كالتالي:

Fun	(No Column Name)
المبيعات	300
التسويق	500
الخدمات	200

ملاحظة:

العمليات الإحصائية مثل (Sum،Min ،Max، Avg، Count) توضع في Having إذا إستخدمناه كعملية شرطية غير ذلك توضع في Select.

مثال:

المثال التالي يوضح متوسط مرتبات موظفي الاقسام التي يزيد عدد موظفيها عن ثلاثة موظفين.

```
Select Fun, Avg(Salary) From Tables2
Group By Fun Having Count(*) >3
```

مثال:

```
Select Fun, Min(Salary) From Tables2
Where Emp In (1, 2, 3, 4, 5, 6, 7, 8)
Group By Fun Having Avg(Salary) >100
```

الاسم المستعار (Alias):

في SQL يستخدم Alias في تغيير اسم الجدول أو الحقل عند عرضه وذلك باستخدام كلمة As ويكون بناء الجملة كالتالي في حالة الجدول:

```
Select column(s) From table_name AS table_alias
```

ويكون بناء الجملة كالتالي في حالة الحقل:

```
Select column AS column_alias  From table_name
```

حيث أن:

table_name : اسم الجدول.
Column : اسم العمود.
table_alias : الاسم المستعار للجدول.
column_alias : الاسم المستعار للعمود.

أمثلة على استخدام الاسم المستعار للحقل (العمود):

نفرض إن جدول Persons كالتالي:

FirstName	LastName	Address	City
علي	منصور	بن عاشور	طرابلس
احمد	سعيد	الظهرة	طرابلس
عصام	فتحي	عمر المختار	طبرق

جملة SQL هي:

```
Select LastName AS Family, FirstName AS Name from Persons
```

ونتيجتها تكون:

Family	Name
منصور	علي
سعيد	احمد
فتحي	عصام

لاحظ تغيير أسماء الحقول(الأعمدة).

ملاحظة:
يمكنك وضع الاسم المستعار لناتج دوال التجميع والدوال العددية بالأعمدة.

مثال:
بجدول الطلاب السابق نريد أن نعرف عدد الطلبة بكل فصل دراسي ونريد تسمية العمود الذي يحتوي علي الأعداد باسم Count_Students فإننا نقوم بالتالي:

```
Select Class, Count(*) As Count_Students from Students
Group by Class
```

والنتيجة تكون كالتالي:

Class	Count_Students
الأول	2
الثالث	1

مثال:
بجدول الطلاب السابق نريد أن نعرف أعلي درجة وأقل درجة وعدد الطلاب فإننا نقوم بالتالي:

```
select max(degree), min(degree), count(*)  from students
```

والناتج هو:

(No Column Name)	(No Column Name)	(No Column Name)
40	10	4

نجد إن الناتج غير واضح حيث إن لا توجد تسمية واضحة للأعمدة لذلك يجب علينا أن نقوم بتسمية للأعمدة الناتجة كالتالي:

```
select max(degree) as max_degree, min(degree) as min_degree, count(*) as
count_students from students
```

والناتج هو:

max_degree	min_degree	count_students
40	10	4

نجد أن الناتج أصبح أوضح حيث انه توجد تسمية واضحة للأعمدة.

مثال على استخدام الاسم المستعار للجدول:

```
Select LastName, FirstName from Persons AS Employees
```

حيث نلاحظ إن اسم الجدول أصبح Employees وتظهر النتيجة التالية:

LastName	FirstName
منصور	علي
سعيد	احمد
فتحي	عصام

الدمج Union و Union All

1- الأمر Union :

يستخدم لاختيار معلومات لها علاقة ببعضها من جدولين مختلفين ودمجهم معاً وهي بذلك تشبه الأمر Join (الذي سيرد شرحه لاحقاً). ولدمج حقلين من جدولين مختلفين ، يجب أن يكون نوع البيانات في الحقلين واحد أي يكون نصوص أو أرقام ...إلخ.

وعند استعمال Union فإن القيم المختلفة فقط يتم اختيارها ، ويكون بناء الجملة كالتالي:

```
SQL Statement1
Union
SQL Statement2
```

حيث تكتب جملة SQL الأولي ثم كلمة Union ثم جملة SQL الثانية.

مثال:

لنفرض أن موظفي شركة طرابلس هم: Emploee_Tripoli

E_ID	E_Name
01	حسن
02	علي
03	ناصر
04	محمود

ولنفرض أن موظفي شركة Zawia هم: Emploee_ Zawia

E_ID	E_Name
01	رضوان
02	علي
03	هاني
04	مؤمن

وعندما نريد دمج الحقل E_Name في كل من الجدولين ونريد عرض أسماء الموظفين المختلفة في الشركتين فإننا نستخدم:

```
Select E_Name From Emploee_Tripoli
Union
Select E_Name From Emploee_Zawia
```

والنتيجة تكون:

E_Name
حسن
علي
ناصر
محمود
رضوان
هاني
مؤمن

نلاحظ إن النتيجة ظهرت بدون تكرار الأسماء لان الموظف "علي" ظهر مرة واحدة فقط.

2 - الأمر Union All: تستخدم مثل جملة Union ، الفرق بينهما أن هذه الجملة تقوم بعرض جميع البيانات حتى إن كان بها تكرارات ، ويكون بناء الجملة كالتالي:

```
SQL Statement1
Union All
SQL Statement2
```

مثال:

بنفس المثال السابق نريد دمج الحقل E_Name في جدولي الشركتين فإننا نستخدم:

```
Select E_Name From Emploee_Tripoli
Union All
Select E_Name From Emploee_Zawia
```

والنتيجة تكون:

E_Name
حسن
علي
ناصر
محمود
رضوان
علي
هاني
مؤمن

نلاحظ ظهور جميع البيانات مع تكرار الأسماء المشتركة بالجدولين، حيث ظهر الموظف "علي" مرتين.

مثال:
إذا كان لديك جدول الصف الأول First وجدول الفصل الثاني Second كالتالي:

First

No	Name
3	شدي
7	علي

Second

Number	Student_N
11	ندي
30	رشا

ونريد دمج بيانات الطلبة فإننا نستخدم:

```
Select No, Name From First
Union All
Select Number, Student_N From Second
```

والنتيجة تكون:

No	Name
3	شدي
7	علي
11	ندي
30	رشا

نلاحظ ظهور جميع البيانات مع الاحتفاظ بأسماء حقول الجدول الأول وإذا أردت استعمال اسم مستعار لها يمكنك استعمال الاسم المستعار كالتالي:

```
Select No as Stu_No, Name as Stu_Name From First
Union All
Select Number, Student_N From Second
```

والنتيجة تكون:

Stu_No	Stu_Name
3	شدي
7	علي
11	ندي
30	رشا

الربط Join :

من أهم مواضيع SQL هو الربط بين الجداول الذي يسهل كثيراً في عرض البيانات من الجداول التي تحتوي علي علاقة مثل الموظف والقسم، ولجعل النتيجة كاملة يجب إنشاء علاقة بين الجدولين والتي تتم عن طريق المفتاح الأساسي (Primary Key). حيث أن الحقل الذي يكون مفتاح أساسي لا يمكن تكرار البيانات بداخله. ففي جدول الموظفين Employees ، العمود Employee_ID هو مفتاح أساسي والذي يعني أنه لايوجد صفين بالجدول لديهما نفس قيمة Employee_ID. لاحظ التالي بالمثال القادم:

- العمود Employee_ID هو المفتاح الأساسي لجدول Employees.
- العمود Prod_ID هو المفتاح الأساسي بجدول Orders.
- العمود Employee_ID بجدول Orders يستخدم ليشير للأشخاص بجدول Employees وهو المفتاح الخارجي.

Employees

Employee_ID	Name
01	وفاء
02	تامر
03	سامي
04	كمال

Orders

Prod_ID	Product	Employee_ID
234	طابعة	01
657	طاولة	03
865	كرسي	03

الإشارة لجدولين:

يمكننا أن نختار بيانات من جدولين عن طريق الإشارة إلي الجدولين بالطريقة التالية:

مثال:

نريد أن نعرف من الذي طلب المنتج (الاسم) وما هو المنتج المطلوب (Product)

```
SELECT Employees.Name, Orders.Product
From Employees, Orders
Where Employees.Employee_ID=Orders.Employee_ID
```

والنتيجة تكون:

Name	Product
وفاء	طابعة
سامي	طاولة
سامي	كرسي

مثال:

نريد أن نعرف من طلب المنتج (الطابعة) ؟

```
Select Emploees.Name
From Employees, Orders
Where Employees.Employee_ID=Orders.Employee_ID
And Orders.Product='طابعة'
```

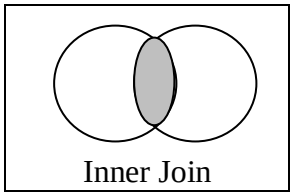
والنتيجة هي:

Name
وفاء

استخدام Joins:

الجداول المرتبطة joined tables هي مجموعة بيانات ناتجة من عمليات ربط تمت بين جدولين أو أكثر ، ويتم اختيار البيانات من الجداول عن طريق استخدام كلمة Join ، ويوجد عدة أنواع من الربط:

1 - **الربط الداخلي** inner join: وهو النوع الافتراضي ، ويحدد أنه إذا كان سجلان متوافقان من جدولين مختلفين ، ولكنهما يوافقان شرط ON الذي يربط بينهما، فعندئذ يجب تضمينهما في مجموعة البيانات ، وإلا يهملان.


Inner Join

بمعني اخر، تقوم هذه الجملة بعرض جميع البيانات المشتركة بين الجدولين، ويكون بناء الجملة كالتالي:

```
Select field1, field2, field3
From first_table
Inner Join second_table
On first_table.keyfield=second_table.foreign_keyfield
```

حيث أن:

Field : الحقل المراد جلبه في جملة الاستعلام Select ويشير field1 إلي الحقل الأول، field2 للحقل الثاني وهكذا.

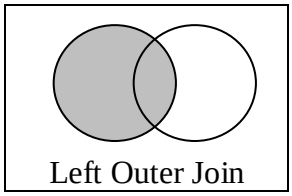
first_table: إسم الجدول الأول.

second_table: إسم الجدول الثاني.

Keyfield: حقل الربط بالجدول الأول.

foreign_keyfield: حقل الربط بالجدول الثاني.

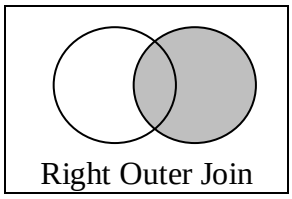
2 - **الارتباط الخارجي الأيسر** left outer join: يرجع السجلات المطابقة للشرط ، ومعها كل السجلات من الجدول المحدد يسار كلمةjoin .


Left Outer Join

ويكون بناء الجملة كالتالي:

```
Select field1, field2, field3
From first_table
Left outer Join second_table
On first_table.keyfield=second_table.foreign_keyfield
```

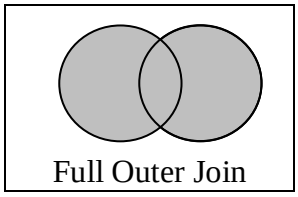
3 - **الارتباط الخارجي الأيمن** right outer join : ويرجع السجلات المطابقة للشرط ، ومعها كل السجلات من الجدول المحدد يمين كلمة join ، مثلا:



يكون بناء الجملة كالتالي:

```
Select field1, field2, field3
From first_table
Right outer Join second_table
On first_table.keyfield=second_table.foreign_keyfield
```

4 - **الربط الخارجي الكامل** full Outer join: ويحدد أن السجلات غير المتطابقة (التي لا تحقق شرط ON) والسجلات المتطابقة (التي تحقق الشرط) هي التي يتم اختيارها جميعاً. بالنسبة للسجلات غير المتطابقة ستظهر القيمة null فيها.



5 - **الارتباط المتقاطع** cross join : وهو حاصل الاختيار من كلا من الجدولين في حالة عدم تحديد فقرة where ، وفي هذه الحالة يتم ربط كل سجل من الجدول الأول مع كل سجل في الجدول الثاني ، لذا فعدد السجلات الناتجة من تطبيق مثل هذا الاستعلام = عدد سجلات الجدول الأول مضروباً في عدد سجلات الجدول الثاني . أما في حالة تحديد فقرة where فسيكون مثل الربط الداخلي inner join .

مثال شامل:

EMPNAME

ID	EMPNAME
1	دلال
2	رامي
3	بدر

EMPADDRESS

ID	ADDRESS
1	بنغازي
2	دبي
4	دبي

INNER JOIN:

```
SELECT EMPNAME.ID, EMPNAME.EMPNAME, EMPADDRESS.ADDRESS
FROM EMPNAME
INNER JOIN EMPADDRESS
ON EMPADDRESS.ID = EMPNAME.ID
```

والنتيجة تكون:

ID	EMPNAME	ADDRESS
1	دلال	بنغازي
2	رامي	دبي

:FULL OUTER JOIN

```
SELECT EMPNAME.ID, EMPNAME.EMPNAME, EMPADDRESS.ADDRESS
FROM EMPNAME
FULL OUTER JOIN EMPADDRESS
ON EMPADDRESS.ID = EMPNAME.ID
```

والنتيجة تكون:

ID	EMPNAME	ADDRESS
1	دلال	بنغازي
2	رامي	دبي
3	بدر	NULL
NULL	NULL	دبي

:LEFT OUTER JOIN

```
SELECT EMPNAME.ID, EMPNAME.EMPNAME, EMPADDRESS.ADDRESS
FROM EMPNAME
LEFT OUTER JOIN EMPADDRESS
ON EMPADDRESS.ID = EMPNAME.ID
```

والنتيجة تكون:

ID	EMPNAME	ADDRESS
1	دلال	بنغازي
2	رامي	دبي
3	بدر	NULL

:RIGHT OUTER JOIN

```
SELECT EMPNAME.ID, EMPNAME.EMPNAME, EMPADDRESS.ADDRESS
FROM EMPNAME
RIGHT OUTER JOIN EMPADDRESS
ON EMPADDRESS.ID = EMPNAME.ID
```

والنتيجة تكون:

ID	EMPNAME	ADDRESS
1	دلال	بنغازي
2	رامي	دبي
NULL	NULL	دبي

:CROSS JOIN

```
SELECT EMPNAME.ID, EMPNAME.EMPNAME, EMPADDRESS.ADDRESS
FROM EMPNAME
CROSS JOIN EMPADDRESS
```

والنتيجة تكون:

ID	EMPNAME	ADDRESS
1	دلال	بنغازي
2	رامي	بنغازي
3	بدر	بنغازي
1	دلال	دبي
2	رامي	دبي
3	بدر	دبي
1	دلال	دبي
2	رامي	دبي
3	بدر	دبي

مثال:
ولتنفيذ جملة Inner join علي مثال الجدولين Employees و Orders:

```
Select Employees.Name, Orders.Product
From Employees
Inner Join Orders
On Employees.Employee_ID= Orders.Employee_ID
```

النتيجة تكون كالتالي:

Name	Product
وفاء	طابعة
سامي	طاولة
سامي	كرسي

حيث تقوم الجملة Inner Join بعرض جميع البيانات المشتركة بين الجدولين.

مثال:
المطلوب عرض أسماء الموظفين الذين طلبوا المنتج (الطابعة) بالمثال السابق:

```
Select Employees.Name
From Employees
Inner Join Orders
On Employees.Employee_ID= Orders.Employee_ID
Where Orders.Product='طابعة'
```

والنتيجة هي:

Name
وفاء

مثال:
ولتنفيذ جملة left join علي مثال الجدولين Employees و Orders ، حيث ان المطلوب هو عرض كل الموظفين وطلباتهم في حالة وجود طلبات؟

```
Select Employees.Name, Orders.Product
From Employees
Left outer Join Orders
On Employees.Employee_ID= Orders.Employee_ID
```

تقوم الجملة بعرض البيانات من الجدول الاول Employees حتي لو لم توجد في الجدول الثاني، والنتيجة تكون كالتالي:

Name	Product
وفاء	طابعة
تامر	NULL
سامي	طاولة
سامي	كرسي
كمال	NULL

مثال:

ولتنفيذ جملة Right join علي مثال الجدولين Employees و Orders ، حيث ان المطلوب هو عرض كل الطلبات ومن طلبها في حالة وجوده ؟

```
Select Employees.Name, Orders.Product
From Employees
Right outer Join Orders
On Employees.Employee_ID= Orders.Employee_ID
```

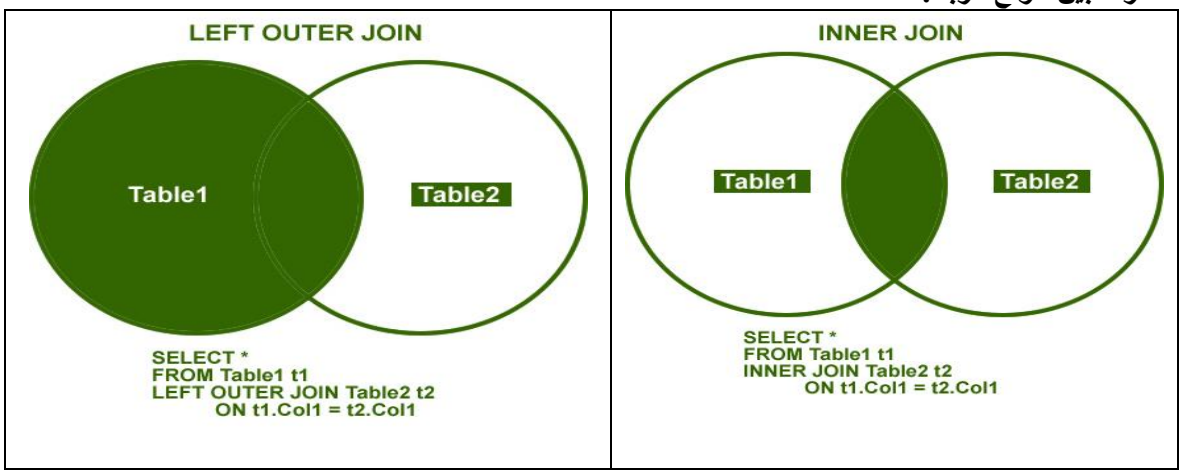
تقوم الجملة بعرض جميع البيانات من الجدول الثاني Orders حتي لو لم توجد في الجدول الاول، والنتيجة تكون كالتالي:

Name	Product
وفاء	طابعة
سامي	طاولة
سامي	كرسي

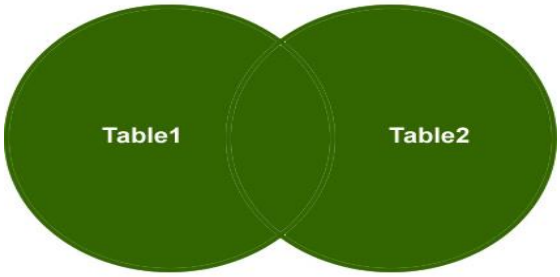
ملاحظة:

أن الربط الخارجي والربط المتقاطع يؤدي إلى إتخام نتائج بحثك بنتائج لا حاجة لها ، وأغلبها تحتوي على القيمة null إما في حقل يمثل حقل المفتاح ، أو في حقل يفيد اتخاذ قرار ما ، لذا لا داعي لاستعمالها بكثرة في حين أن الربط الداخلي ، أو الربط البسيط بين جدولين يكفي.

مقارنة بين أنواع الربط:

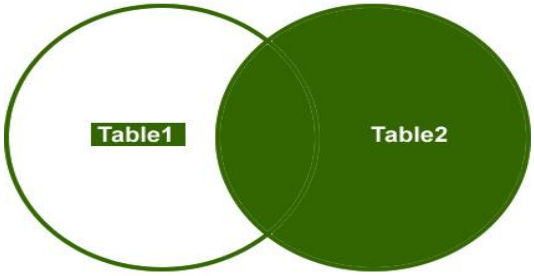


FULL OUTER JOIN




```
SELECT *
FROM Table1 t1
FULL OUTER JOIN Table2 t2
    ON t1.Col1 = t2.Col1
```

RIGHT OUTER JOIN




```
SELECT *
FROM Table1 t1
RIGHT OUTER JOIN Table2 t2
    ON t1.Col1 = t2.Col1
```

CROSS JOIN

	ID	Value
1	1	First
2	2	Second
3	3	Third
4	4	Fourth
5	5	Fifth

	ID	Value
1	1	First
2	2	Second
3	3	Third
4	6	Sixth
5	7	Seventh
6	8	Eighth

	ID	Value	ID	Value
1	1	First	1	First
2	1	First	2	Second
3	1	First	3	Third
4	1	First	6	Sixth
5	1	First	7	Seventh
6	1	First	8	Eighth
7	2	Second	1	First
8	2	Second	2	Second
9	2	Second	3	Third
10	2	Second	6	Sixth
11	2	Second	7	Seventh
12	2	Second	8	Eighth

لغة التحكم في البيانات SQL Data Control Language (DCL)

كما عرفنا بالمحاضرة الأولي فإن DCL هي قسم اللغة في الـ SQL المسئول عن ضمان الأمان Security والتحكم في وصول المستخدمين للبيانات. حيث يمكننا إعطاء المستخدمين صلاحيات معينة ، فعلي سبيل المثال يمكننا تعيين مشرف عام علي العمل ونقوم بإعطائه كافة الصلاحيات علي قاعدة بيانات معينة ،في حين لا نعطي هذه الصلاحيات لباقي الأشخاص وذلك نظراً لأسباب كثيرة منها مثلاً المحافظة علي سرية البيانات أو عدم ثقتنا الكاملة في شخص معين. وربما تكون بيانات مالية مهمة لا نريد أن يعبث بها احد كبيانات المصارف أو ربما تكون هذه البيانات أمنية وحساسة ، لذلك نقوم بتحديد صلاحيات معينة لأي شخص يتصل بقاعدة البيانات التي نقوم بتصميمها. وتتكون DCL من الأوامر Grant و Revoke.

الامر GRANT

يستخدم هذا الأمر لإعطاء صلاحيات معينة لاي مستخدم، و تركيبة جملة SQL تكون كالتالي:

```
GRANT [ALL, SELECT, INSERT, UPDATE, DELETE]
ON table_name
TO user_name
```

حيث أن:

table_name : هو إسم الجدول الذي يتم به تنفيذ المستخدم لصلاحياته

user_name : هو اسم الشخص المخول له صلاحيات معينة

ALL : إعطاء جميع الصلاحيات لهذا المستخدم من حيث الاوامر التالية:
SELECT، INSERT، UPDATE، DELETE

SELECT : إعطاء صلاحية إستخدام الأمرSELECT في الجدول
INSERT : إعطاء صلاحية إستخدام الأمرINSERT في الجدول
UPDATE : إعطاء صلاحية إستخدام الأمرUPDATE في الجدول
DELETE : إعطاء صلاحية إستخدام الأمرDELETE في الجدول

مثال :

المطلوب منك إعطاء كل الصلاحيات بالجدول EMPLOYEE إلي ALI
لذلك فإن جملة SQL الملائمة هي:

```
GRANT ALL ON EMPLOYEE TO ALI
```

طلب منك الأن إعطاء صلاحيات الإختيار والإلغاء إلي كلاً من ماجد واحمد ، يمكنك فعل ذلك عن طريق:

```
GRANT SELECT, DELETE ON EMPLOYEE
TO MAJD, AHMED
```

يمكنك إعطاء الصلاحية لعمود واحد فقط بالجدول فعلي سبيل المثال المطلوب منك إعطاء صلاحية التعديل لعمود المرتب {SALARY} بجدول MONEY إلي ALI

```
GRANT UPDATE(SALARY) ON MONEY
TO ALI
```

وقد تضطر أحياناً إلي إعطاء كافة المستخدمين لكل الصلاحيات بحيث يصبح الجدول لديك عام { PUBLIC} للجميع {ALL} لذلك تقوم بكتابة جملة SQL التالية:

```
GRANT ALL ON PERSON TO PUBLIC
```

حيث أن إسم الجدول بالمثال السابق هو PERSON

الامر REVOKE

يستخدم هذا الأمر لإلغاء صلاحيات معينة لاي مستخدم من جدول معين ، و تركيبة جملة SQL تكون كالتالي:

```
REVOKE [ALL, SELECT, INSERT, UPDATE, DELETE]
ON table_name
FROM user_name
```

حيث أن:

table_name : هو إسم الجدول الذي سيتم به إلغاء صلاحيات معينة لمستخدم

user_name : هو اسم الشخص المطلوب سحب صلاحية معينة منه

مثال :

المطلوب منك سحب صلاحية الإلغاء من MAJED بجدول EMPLOYEE لذلك فإن جملة SQL الملائمة هي:

```
REVOKE DELETE ON EMPLOYEE
FROM MAJED
```

لغة معالجة البيانات Transaction Control Language (TCL)

كما عرفنا بالمحاضرة الأولي فإن TCL هي قسم اللغة في الـ SQL المسئول عن التحكم في عمليات معالجة البيانات (التعاملات). فسلسلة التغيرات التي تحدث في قاعدة البيانات من حيث وجود سلسلة من جمل الإدخال والتعديل والإلغاء يطلق عليها حركة التعديلات أو *transaction*. هذه التغيرات تعتبر مؤقتة ولا يتم اعتمادها إلا عند وضع الجملة commit بعد سلسلة التغيرات التي نريد تطبيقها.

ويمكن للمستخدم أن يتراجع عن تطبيق مجموعة من التغيرات التي قام بوضعها بعد اخر جملة commit إذا لم يقم بوضع جملة commit بعد هذه التغيرات الجديدة، ولإلغاء تطبيق هذه التغيرات يتم وضع الجملة rollback فقط.

rollback تشبه الأمر " undo" الموجود بمعظم البرامج. مع مراعاة بأنه لايمكن للمستخدم وضع rollback ليتراجع عن تطبيق مجموعة من التغيرات إذا قام بوضع جملة commit بعدها.

ملاحظة:

أوامر data definition مثل أمر create table ينتج عنه commit داخلي، أي يتم تنفيذه ضمنياً ولاحاجة لوضعه صراحة بعد أمر create table.

لذلك يمكننا القول بأن TCL تتكون من الأوامر التالية:

3. Commit: يستخدم ليتم تطبيق العمليات أو التغييرات الحاصلة علي البيانات.
4. Rollback: يستخدم لإلغاء تطبيق العمليات أو التغييرات الحاصلة علي البيانات.