الفصل الحادى عشر
استخدام SQL

تعتبر لغة SQL من أساسيات قواعد البيانات حيث تحتـــوى على العديد من الأوامر اللازمة لأداء العمليات المختلفة على قواعد البيانات.

بانتهاء هذا الفصل، ستتعرف على:

- ***استرجاع المعلومات من قواعد البيانات باســـتخدام أوامـــر SQL.***
- ***التعديل فى تصميم قواعد البيانات باستخدام أوامر SQL.***
- ***التعامل مع بيانات الجداول بالإضافة أو الحذف أو التعديل***
- ***استخراج إحصائيات عن بيانـــات الجـــداول (مجمـــوع – متوسط– أصغر/أكبر قيمة).***

مفهوم لغة SQL

كلمة **SQL** اختصار للعبارة **Structured Query Language**. وهى كما يبدو من اسمها عبارة عن مجموعة من التعليمات تستخدم للحصول على **Quires** أو استعلامات عن البيانات فى قواعد البيانات.

ويمكن اعتبار **Structured Query Language** (**SQL**) نوع من لغات البرمجة التى تستخدم بصفة أساسية في الأغراض الآتية:

- استخراج بيانات من جدول أو أكثر في واحدة أو أكثر من قواعد البيانات
- معالجة بيانات الجداول بالإضافة أو التعديل أو الحذف
- استخراج إحصائيات عن بيانات الجداول مثل المجموع والمتوسط الحسابي وأصغر/أكبر قيمة.
- إنشاء جداول البيانات وتعديلها وحذفها
- إنشاء أو تعديل الفهارس في قاعدة البيانات

الشكل العام لعبارة SQL

تتيح العبارة الواحدة أو الجمل القليلة من لغة **SQL** للمصمم أن يحصل على وظائف عن قواعد البيانات، لا يمكن الحصول عليها إلا بمئات التعليمات باستخدام بعض لغات البرمجة الأخرى. فيمكن أن تستخدم العبارة الواحدة لتحديد الجدول أو الجداول والحقول المطلوبة من كل منها للاستعلام والسجلات التى سيشملها الاستعلام والترتيب الذي تظهر به في الاستعلام.

والشكل العام لعبارة **SQL** هو:

[parameters declarations] Manipulative statement [options]

ومن هذا الشكل العام تلاحظ أن عبارة **SQL** تحتوى على ثلاثة أجزاء نوضحها فيما يلي:

- **Parameter Declarations**: وهي المتغيرات التى يتم إدخالها من البرنامج إلى العبارة. وهذه المتغيرات اختيارية.

- **Manipulative Statement**: الأمر المنفذ داخل العبارة مثل أمـــر **SELECT** أو أمر **DELETE**.
- **Options**: اختيارات إضافية يمكن إضافتها للأمر مثلاً لتحديد طريقـــة ترتيـــب البيانات أو لتحديد شرط أو أكثر للحصول على الاستعلامات.

أوامر SQL

سنركز في هذا الفصل علـــى الجـــزء المنفـــذ مـــن عبـــارة **SQL (Manipulative Statement)** والاختيارات (**Options**) التى يمكن إضافتها للأمر لتوجيه ناتج الأمـــر مثلا لتحديد شروط الاستعلام أو نطاق السجلات أو ترتيبها. واستخدام هذين الجـــزأين كافٍ للحصول على استعلامات تؤدي معظم وظائف قواعد البيانات.

يحتوي الجدول ١١-١ التالى على الأوامر الرئيسية ووظيفة كلٍ منها.

جدول ١١-١ الأوامر الرئيسية المستخدمة داخل لغة **SQL**

الأمر	وظيفته
SELECT	اختيار مجموعة محددة من السجلات ووضع النتيجة فى جدول أو **Dynaset**
INSERT INTO	إضافة مجموعة سجلات إلى جدول
UPDATE	التعديل فى الحقول داخل جدول
DELETE	حذف مجموعة سجلات من جدول

الأمر Select

يستخدم هذا الأمر لاختيار مجموعة من السجلات أو مجموعة من الحقول داخل ســـجلات ثم يقوم بوضع هذه المعلومات فى جدول أو **Dynaset**. يأخذ الأمر **Select** في أبســـط صورة له الشكل التالى:

Select {FieldList} FROM {TableList}

وفي هذا الشكل المبسط يقوم الأمر **Select** باختيار مجموعة الحقول المحددة فى **FieldList** من مجموعة الجداول **TableList**. يمكن استخدام الرمز (*) بدل من الكلمة **FieldList** بمعنى جميع الحقول. فمثلاً نستخدم الأمر التالى للحصول على جميع الحقول الموجودة بجدول **Customers**.

SELECT * FROM Customers

لاختيار مجموعة من الحقول الموجودة في الجدول، اكتب أسماء هذه الحقول وبينها العلامة (،) كما فى الأمر التالى الذى يستخدم للاستعلام عن اسم العميل ورقم تليفونه من جدول **Customers**:

SELECT Cust_name, Phone FROM Customers

إذا كان اسم الحقل يحتوى على فراغ مثل **Cust Name** بدلاً من **Cust_name** فيتم وضع هذا الحقل داخل أقواس مربعة هكذا **[Cust Name]**.

اختيار مجموعة من الحقول من أكثر من جدول

للحصول على مجموعة من الحقول من أكثر من جدول يتم كتابة اسم الجدول متبوعاً بنقطة (.) ثم اسم الحقل المطلوب استخراجه من هذا الجدول. كما يتم كتابة أسماء جميع الجداول ويفصل بينهم بالحرف (،) بنفس الطريقة التى نتعامل بها مع الحقول.

يستخدم الأمر التالى للحصول على رقم الصنف واسمه من جدول **Items** واسم المورد ورقم تليفونه من جدول **Suppliers**

```
SELECT Items.Item_code, Items.Item_Name,
Suppliers.Supp_Name, Suppliers.Phone FROM Items, Suppliers
```

الاستعلام عن حقل باستخدام الأمر Select

يقوم الأمر التالى بالاستعلام عن سجل يحتوى على أسماء العملاء **Cust_name** وأرقام تليفوناتهم **Phone** من جدول **Customers**، والأصناف المطلوبة **Unit1** وإجمالى قيمتها

(وهو حقل مستنتج من حاصل ضرب سعر الصنف Unit1_Price فى كمية الصنف Unit1_Count) من جدول Orders (لاحظ أن هذا الحقل لا يمكن التعديل فيه).

```
SELECT Customers.Cust_name, ustomers.Phone, Orders.Unit1,
Order.Unit1_Price *  Orders.Unit1_Count   FROM Customers,
Orders
```

استخدام الشرط Where مع الأمر Select

للاستعلام عن السجلات التى تبلغ قيمة الطلب الواحد منها ١٠٠ جنية أو أكثر فإننا سنكتب الأمر السابق وسنضيف الشرط Where كما يلي:

```
Where Orders.Unit1_Price * Orders.Unit1_Count  > = 100
```

ومعناه بشرط أن تكون قيمة الطلب (سعر الصنف مضروبا في الكمية المطلوبة) تساوي أو أكبر من ١٠٠. في هذه الحالة يتم استبعاد السجلات التى لا ينطبق عليها هذا الشرط.

وسيكون الأمر بعد إضافة الشرط هكذا:

```
SELECT Customers.Cust_name, Customers.Phone,
Orders.Unit1,Order.Unit1_Price *  Orders.Unit1_Count   FROM
Customers , Orders Where Orders.Unit1_Price *
Orders.Unit1_Count  > = 100
```

تسمية الحقل المستنتج داخل الأمر SELECT

يتم تسمية الحقل بإضافة الاختيار AS لأمرSELECT ثم كتابة اسم الحقل وذلك بعد كتابة الأمر اللازم لاستنتاجه. في المثال السابق إذا أردنا تسمية حقل إجمالى السعر (سعر الصنف مضروبا في الكمية المطلوبة) بالاسم Total، يجب إعادة كتابة الأمر كما يلي:

```
SELECT Customers.Cust_name,Customers.Phone,Orders.Unit1,
Order.Unit1_Price * Orders.Unit1_Count  As Total
FROM Customers,Orders
```

استخدام الأسماء المختصرة مع العبارة Select

راجع الأمر السابق تجد أنه طويل نسبياً ويمكن اختصاره بالتعويض عن أسماء الجداول المستخدمة (Customers, Orders) بالأحرف (CS, OS) على الترتيب ليصبح الأمر

كالآتى:

```
SELECT CS.Cust_name, CS.Phone, OS.Unit1,OS.Unit1_Price *
OS.Unit1_Count  As Total From CS AS Customers, OS AS
Orders
```

لاحظ فى هذا المثال أنه بعد كتابة الأسماء المختصرة واستخدامها فى جميع المواضع يتم التنويه عنها باستخدام الاختيار **FROM** بأن **(CS)** اختصار لاسم الجدول **Customers** و **(OS)** اختصار لاسم الجدول **Orders**.

استبعاد التكرارات فى بيانات حقل معين

يستخدم الاختيار **Distinct** لاستبعاد السجلات المكررة. فمثلاً في المثال السابق إذا أردنا إظهار أسماء العملاء الذين لهم طلبات عندنا، بحيث يظهر اسم العميل مرة واحدة فقط وتستبعد السجلات المكررة، يجب إضافة الاختيار **Distinct** لأمر **SELECT** هكذا:

```
SELECT  Distinct CS.Cust_name, CS.Phone, OS.Unit1,
OS.Unit1_Price * OS.Unit1_Count  As Total From CS AS
Customers, OS AS Orders
```

يستخدم الاختيار **Distinct Row** لاستبعاد أى تكرار فى السجل كله (أى إذا تكررت جميع قيم الحقول بدون استثناء).

ربط الجداول

عرفت من قبل أننا نقوم عند تصميم جداول قاعدة البيانات بتعيين مفتاح أساسي (**Primary Key**) في كل جدول. يستخدم هذا المفتاح الأساسي لربط الجداول في قاعدة البيانات. فمثلاً يمكن استخدام حقل "رقم الموظف" في جدول "الموظفين" لربط بيانات الموظف مع جدول "بيانات الموظفين". بفرض أن جدول "الموظفين" يحتوى على بيانات الموظف الوظيفية مثل المرتب والدرجة المالية والإدارة التى يعمل بها، بينما يحتوى جدول "بيانات الموظفين" على البيانات الشخصية للموظف مثل الاسم والعنوان ...الخ. ونلجأ لاستخدام جدولين في مثل هذه الحالات بدلا من كتابة كل بيانات الموظف الشخصية

والوظيفية في جدول واحد لأننا لا نحتاج غالباً لبيانات الموظف الشخصية في كل الحالات. يستخدم المفتاح الأساسي (رقم الموظف في هذه الحالة) داخل عبارة **SQL** لربط الجدولين. وعامةً تستخدم لغة **SQL** طريقتين لربط الجداول:

- استخدام الاختيار **JOIN**: ويقوم بربط الجدولين بناءً على الحقول المختارة من كل جدول ونوع الربط
- استخدام الاختيار **WHERE**: ويستخدم عادةً لتصفية السجلات المستخرجة من كلا الجدولين. وفيما يلي نشرح كلاً من الطريقتين.

استخدام JOIN لربط الجداول

يستخدم الاختيار **Join** لربط جدولين وذلك طبقاً لمحتويات الحقول المحددة لإظهارها ونوع الأداة **Join** المحددة.

والشكل العام للاختيار **JOIN** داخل عبارة **SQL** كما يلي:

```
table1 {INNER/LEFT/RIGHT} JOIN table2 ON table1.key1 =
table2.key2
```

ومن هذا الشكل العام يتضح أن هناك ثلاث حالات لاستخدام **JOIN** لربط الجداول هي **INNER** و **LEFT** و **RIGHT**. ويوضح الجدول ١١-٢ التالي السجلات التي نحصل عليها في كل حالة من هذه الحالات.

جدول ١١-٢ أنواع ربط الجداول

نوع الربط	السجلات الناتجة من الجدول الأيسر	السجلات الناتجة من الجدول الأيمن
INNER	السجلات المتعلقة بالجدول الأيمن فقط	السجلات المتعلقة بالجدول الأيسر فقط
LEFT OUTER	جميع السجلات	السجلات المتعلقة بالجدول الأيسر فقط

نوع الربط	السجلات الناتجة من الجدول الأيسر	السجلات الناتجة من الجدول الأيمن
RIGHT OUTER	السجلات المتعلقة بالجدول الأيمن فقط	جميع السجلات

ويجب الانتباه إلى أن الجدول الأيسر هو الجدول الموجود يسار كلمة **JOIN** والجدول الأيمن هو الجدول الموجود يمينها. فمثلاً في الشكل العام للاختيار الذي عرضناه قبل قليل، يكون **table1** هو الجدول الأيسر، و **table2** هو الجدول الأيمن.

الجدول السابق يعنى أنك إذا استخدمت الأمر **Join** مع النوع **Inner** وتُكتب داخل الأمر هكذا:

INNER JOIN

للربط بين جدولين بينهما علاقة، فإن الناتج سيكون فى هذه حاله عبارة عن سجلات مستخلصة من الجدول الأول ولها علاقة بالجدول الثانى بالإضافة إلى سجلات مستخلصة من الجدول الثانى ولها علاقة بالجدول الأول. أما إذا استخدمنا النوع الثانى **LEFT JOIN** فإن الناتج سيكون جميع السجلات الموجودة داخل الجدول الأول بالإضافة إلى السجلات الموجودة داخل الجدول الثانى والتى لها علاقة بالجدول الأول. وسيحدث العكس فى حالة استخدام النوع الثالث **RIGHT JOIN** وذلك طبقاً للجدول الموضح سابقاً.

يمكن استخدام أى عوامل مقارنه داخل الاختيار **JOIN** لربط جدولين، مثل: **<>, >=, <= , >, <.**

والآن لنرى مثالاً على كل حالة من هذه الحالات.

مثال ١ :

```
SELECT CS.Cust_name, CS.Phone, OS.Unit1, OS.Unit1_Price *
OS.Unit1_Count  As Total  From CS AS Customers ,OS AS
Orders, CS INNER JOIN OS ON CS.Cust_id =OS.Cust_id
```

في هذا المثال سنحصل على استعلام عبارة عن السجلات التى يتساوى فيها رقم العميل Cust_id في الجدول الأول CS مع رقم العميل في الجدول الثاني OS وذلك باعتبار أن حقل رقم العميل Cust_id هو حقل المفتاح الأساسي، وسبق أن شرحنا أن السجلات ستحتوى على حقول مختارة بالإضافة إلى حقل محسوب.

مثال ٢ :

```
SELECT CS.Cust_name, CS.Phone, OS.Unit1,OS.Unit1_Price *
OS.Unit1_Count  As Total From CS AS Customers ,OS AS
Orders, CS LEFT OUTER JOIN OS ON CS.Cust_id =OS.Cust_id
```

في هذا المثال سنحصل على استعلام عبارة عن كل سجلات الجدول CS بالإضافة إلى السجلات التى يتساوى فيها رقم العميل Cust_id في الجدول الأول CS مع رقم العميل في الجدول الثاني OS فقط، وذلك باعتبار أن حقل رقم العميل Cust_id هو حقل المفتاح الأساسي.

مثال ٣ :

```
SELECT CS.Cust_name, CS.Phone, OS.Unit1, OS.Unit1_Price *
OS.Unit1_Count  As Total  From CS AS Customers , OS AS
Orders, CS RIGHT OUTER JOIN OS ON CS.Cust_id =OS.Cust_id
```

في هذا المثال سنحصل على استعلام عبارة عن كل سجلات الجدول OS بالإضافة إلى السجلات التى يتساوى فيها رقم العميل Cust_id في الجدول CS مع رقم العميل في الجدول OS فقط، وذلك باعتبار أن حقل رقم العميل Cust_id هو حقل المفتاح الأساسي.

استخدام WHERE *لربط الجداول*

يعطى هذا الاختيار نفس النتيجة التى يعطيها الاختيار JOIN مع النوع INNER. المثال التالي يستخدم الاختيار WHERE لربط جدولين بنفس طريقة استخدام INNER JOIN التى شرحناها قبل قليل ويعطي نفس النتيجة:

```
SELECT CS.Cust_name, CS.Phone, OS.Unit1,OS.Unit1_Price *
OS.Unit1_Count  As Total From CS AS Customers, OS AS
Orders, WHERE CS.Cust_id = OS.Cust_id
```

ينتج عن استخدام الاختيار **WHERE** لربط الجداول سجلات يمكن قراءتها فقط. فإذا أردت الحصول على سجلات قابلة للتعديل، يمكنك استخدم الأمر **JOIN**.

استخدام معايير التصفية Filter Criteria

تحتاج فى أحوال كثيرة إلى مجموعة محددة من السجلات تحتوى على بيانات معينة. فمثلاً عندما تريد الحصول على جميع العملاء الذين تاريخ ميلادهم قبل ١٩٦٠/١/١. لكى تحصل علي السجلات ذات تاريخ ميلاد قبل ١٩٦٠/١/١يجب أن تستخدم معيار تصفية أو **Filter Criteria**. ويقال عنه معيار تصفية لأننا تقريباً نصفي فقط السجلات التى تقابل الشرط أو معيار التصفية المحدد في عبارة **SELECT**.

ويتم تحديد معيار التصفية داخل عبارة **SQL** باستخدام الاختيار **WHERE**. يستخدم **SQL** عدد من معاملات المقارنة ومعاملات التشغيل التي يمكن استخدامها مع الاختيار **WHERE** لتحديد معيار التصفية والموضحة فى جدول ١١-٣ و١١-٤ التاليين:

جدول ١١-٣ معاملات المقارنة

المعامل	معناه
<	أصغر من.
= <	أصغر من أو تساوى.
=	تساوى.
>	أكبر من.
> =	أكبر من أو تساوى.
<>	لا تساوى.

وهذه المعاملات معروفةً لك من دراستك للجزء الأول من هذه السلسلة.

أمثلة:

```
SELECT  * FROM Customers Where Last_name ='ناصر'
SELECT  * FROM Customers Where Date < # 1/1/1960#
SELECT * FROM Orders Where Unit1_Price > = 50
```

- في المثال الأول نستعلم عن العملاء الموجودين في جدول **Customers** واسم عائلاتهم **Last_name** هو "ناصر". وقد استخدمنا المعامل = لمقارنة محتويات الحقل مع الاسم الموجود في الأمر. وقد وضعنا النص الذي نود مقارنته مع محتويات الحقل بين علامتى " " حيث يجب أن توضع البيانات النصية بين هاتين العلامتين.
- وفي المثال الثاني نستعلم عن جميع العملاء الذين تاريخ ميلادهم قبل ١٩٩٦٠/١/١وقد وضعنا التاريخ الذي نود مقارنته مع التاريخ الموجود بالحقل بين علامتى # # حيث يجب أن توضع البيانات التاريخية بين هاتين العلامتين.
- وفي المثال الثالث نستعلم عن الأسعار التى تساوي أو تزيد عن ٥٠. وكما تلاحظ فإننا لا نحتاج لوضع الأرقام بين علامات مميزة.

يتم ضبط التاريخ بالشكل "اليوم / الشهر / السنة" فى بيئة **Windows** من داخل الإعدادات الإقليمية بلوحة التحكم **Control Panel**.

جدول ١١-٤ معاملات التشغيل

المعامل	الاستخدام
Like	يستخدم عادة لمقارنة محتويات حقل مع بيانات متشابهة مثل *CS
In	يستخدم عادة لمقارنة محتويات حقل بمجموعة قيم
Between	يستخدم عادةً لمقارنة محتويات حقل بمجموعة من القيم تقع في نطاق أو مدي معين

وتستخدم هذه المعاملات لمقارنة محتويات حقل بقيمة معينة وذلك كما يلى:

المعامل Like

يستخدم المعامل **Like** للمقارنة بين محتويات الحقل وشكل **(Pattern)** يحتوى على مجموعة من الرموز موضحة بالجدول ١١-٥ التالى.

جدول ١١-٥ الرموز المستخدمة مع المعامل **Like**

الرمز	يستخدم لمقارنة	مثال	مثال للنتيجة
*	مجموعة من الحروف	A*	Ahmed و Ali و Amir
?	حرف واحد	A?A	ABA و ACE و AZE
#	رقم واحد	#٤١١	34, 1154,....١١
[List]	حرف يقع في مجموعة الحروف المحددة	[X- Z]	X, Y, Z
[! List]	حرف لا يقع ضمن مجموعة الحروف	[!X-Z]	A, M, h

مثال

```
SELECT * FROM Customers Where Name Like "A*"
```

عند تنفيذ هذا الأمر يتم البحث فى الجدول **Customers** داخل الحقل **Name** عن جميع الأسماء التى تبدأ بالحرف **A** ولا يهم عدد أو ماهية بقية حروف الاسم.

المعامل In

يستخدم هذا المعامل للمقارنة بين مجموعة القيم التى يتم تحديدها في الأمر وبين قيم الحقل أو مجموعة الحقول المطلوبة من الجدول. يمكنك مثلاً استخدام العبارة التالية للحصول على جميع العملاء الذين يسكنون في: الإسكندرية والقاهرة والأقصر.

```
SELECT * FROM Customers Where City IN ( "الإسكندرية", "القاهرة", "الأقصر" )
```

المعامل ***Between***

يستخدم هذا المعامل لمقارنة محتويات حقل بمجموعة من القيم تقع في نطاق أو مدي معين. المثال التالي يستعلم عن جميع الطلبات الموجودة فى جدول **Orders** والتى تقع قيمتها بين ٢٠ جنية و١١ جنية:

```
SELECT * FROM Orders WHERE TotalCost Between 20 and 50
```

استخدام أكثر من معيار

إذا أردنا الحصول على أسماء جميع البائعين التى تبدأ بالاسم "عمر" ويقع تاريخ ميلادهم بين أول يناير ١٩٦٠ إلى آخر ديسمبر ١٩٧٠، فإننا نستخدم أكثر من معيار داخل الاختيار **WHERE**. والأمر المناسب لهذه الحالة هو:

```
SELECT * FROM Sales Where Sale_Name Like "عمـــر" * AND Date
Between  # 01/01/60# and #31/11/70#
```

حيث استخدمنا **AND** داخل **SELECT** لتحديد أكثر من معيار للبحث، وفي هذه الحالة يجب أن ينطبق كلا الشرطين على السجلات التى سيتم الحصول عليها من الجدول. في هذا المثال الشرط الأول أن يكون اسم البائع هو "عمر" والشرط الثاني أن يقع تاريخ المـــيلاد بين أول يناير ١٩٦٠ وآخر ديسمبر ١٩٧٠.

يمكنك استخدام **OR** للبحث باستخدام أحد المعيارين. فإذا اســـتبدلنا المعامـــل **AND** في المثال السابق بالمعامل **OR** فإننا نحصل على السجلات التى ينطبق عليها أحـــد الشـــرطين المحددين في الأمر.

ترتيب السجلات

إذا أردت ترتيب السجلات التي تحصل عليها من الاستعلام، استخدم الاختيـــار **Order By** ومعناه رتب السجلات الناتجة. والترتيب إما أن يكون تصاعديا (من الألف إلى اليـــاء ومن صفر إلى ٩) وإما أن يكون تنازلياً (من الياء إلى الألف ومن ٩ إلى صفر). والترتيـــب التلقائي الذي تختاره **SQL** هو الترتيب التصاعدى.

إذا كان الحقل المتخذ كأساس للترتيب مفهرساً، فإن ذلك يساعد على سرعة تنفيذ الأمر وبالتالي الحصول على السجلات المطلوبة.

للحصول على سجلات العملاء الموجودين بالجدول **Customers** حسب الترتيب الأبجدى لاسم العميل (ترتيباً تصاعدياً)، استخدم عبارة **SELECT** كما يلي:

```
SELECT * FROM Customers Order By Cust_Name
```

أما إذا أردنا ترتيب نفس العملاء الموجودين بالجدول ترتيباً تنازلياً، فيجب إضافة كلمة **Desc** بعد اسم الحقل المتخذ أساساً لترتيب السجلات كما يلي:

```
SELECT * FROM Customers Order By Cust_Name Desc
```

في حالة الترتيب التصاعدى لا يلزم ذكر كلمة **Asc** كما فى المثال الأول أما إذا كان المطلوب ترتيب السجلات تنازلياً، فيجب ذكر كلمة **Desc** بعد اسم الحقل المتخذ أساساً لترتيب السجلات كما في المثال الثاني.

استخدام الدوال

يمكنك استخدام عبارة **SELECT** لتنفيذ بعض العمليات الحسابية على بيانات الجدول باستخدام بعض الدوال **Functions**. يوضح الجدول ١١-٦ التالي الدوال التي يمكن استخدامها مع عبارة **SELECT** ووظيفة كل منها.

جدول ١١-٦ الدوال المستخدمة مع عبارة **Select**

الدالة	وظيفتها
AVG	تحسب المتوسط الحسابي لمجموع القيم الموجودة في حقل معين للسجلات التى ينطبق عليها الشرط الموجود في الاختيار **Where**
Count	تحسب عدد السجلات التى ينطبق عليها الشرط الموجود في الاختيار **Where**
Min	تحسب القيمة الصغرى لمجموعة قيم موجودة في حقل معين للسجلات التى ينطبق عليها الشرط الموجود في الاختيار **Where**

الدالة	وظيفتها
Max	تحسب القيمة العظمى لمجموعة قيم موجودة في حقل معين للسجلات التى ينطبق عليها الشرط الموجود في الاختيار Where
Sum	تحسب مجموع القيم الموجودة في حقل معين للسجلات التى ينطبق عليها الشرط الموجود في الاختيار Where

مثال: المثال التالي يستخدم عبارة SELECT لحساب المجموع والمتوسط الحسابي والقيمة العظمي للأسعار الموجودة في جدول Orders:

```
SELECT SUM  (Orders.Price) AS Total, AVG  (Orders.Price) AS
Average, MAX (Orders.Price) AS Minimum FROM Orders WHERE
Orders.Price > 0
```

تجميع السجلات

تجميع السجلات معناه استخراج سجل واحد لكل مجموعة سجلات متشابهة ممثلاً سجل واحد لجميع عملاء مدينة القاهرة وسجل واحد من جميع عملاء مدينة الإسكندرية ... وهكذا. يستخدم الاختيار Group By داخل عبارة SELECT لهذا الغرض. وتصبح هذه الميزة مفيدة عند الرغبة في الحصول على ملخصات عن بيانات الجدول كما في المثال السابق.

في المثال السابق إذا أردنا تجميع بيانات الطلبات طبقاً لرقم الطلب بحيث يظهر سجل واحد لكل طلب (رغم أن الطلب الواحد يشتمل على أكثر من سجل) مشتملاً على المجموع الكلي أو المتوسط الحسابي أو أعلي قيمة ... الخ. يجب إضافة الاختيار Group By إلى عبارة SELECT كما يلي:

```
SELECT SUM  (Orders.Price) AS Total, AVG  (Orders.Price) AS
Average, MAX (Orders.Price) AS Minimum FROM Orders WHERE
Orders.Price > 0 GROUP BY Order_id
```

طبقاً للعبارة السابقة سيظهر سجل واحد لكل طلب، وتظهر به البيانات الإحصائية (المجموع ، المتوسط الحسابي ، القيمة العظمي).

إضافة الاختيار Having

بإضافة الاختيار **Having** إلى الاختيار **Group By** الذي شرحناه في البند السابق يمكن إظهار السجلات التى ينطبق عليها شرط معين من مجموعة السجلات الناتجة من الأمر **SELECT**. وهى وظيفة مشابهة جداً لوظيفة **Where** التى تقوم بتحديد مجموعة من السجلات طبقاً لشرط معين من الجدول كله أو مجموعة الجداول التى تتعامل معها. إذاً الفرق الوحيد أن **Having** تبحث عن السجلات التى توافق الشرط داخل السجلات الناتجة من عبارة **Select** ، أما **Where** فتبحث عن السجلات التى توافق الشرط داخل الجدول كله.

بالرجوع إلى المثال السابق إذا أردنا تجميع الطلبات (والحصول على الإحصائيات المذكورة) بشرط أن تزيد قيمة الطلب عن ٢٠٠ جنية. سنقوم الآن بكتابة عبارة **SELECT** ونضيف إليها الشرط الأخير مستخدمين الاختيار **Having** كما يلى:

```
SELECT SUM (Orders.Price) AS Total, AVG (Orders.Price) AS
Average, MAX (Orders.Price) AS Minimum FROM Orders WHERE
Orders.Price > 0 GROUP BY Order_id HAVING SUM
(Orders.Price) > 200
```

حذف السجلات

يستخدم الأمر **DELETE** لحذف مجموعة سجلات من جدول طبقاً لشرط معين. فمثلاً إذا أردنا أن نحذف سجلات لعملاء الذين يقطنون مدينة الإسكندرية، فإن العبارة المطلوبة لذلك هى:

```
Delete FROM Customers Where City = "الإسكندرية"
```

إضافة السجلات

يستخدم الأمر **INSERT** لإضافة مجموعة من السجلات إلى جدول معين. إذا أردنا أن استخراج سجلات العملاء الذين يقطنون مدينة القاهرة فقط من جدول **Customers** ثم نضيفهم إلى جدول جديد اسمه **Custom** فإن العبارة المطلوبة لذلك هى:

```
INSERT INTO Custom (SELECT * FROM Customers WHERE City
( "القاهرة"=
```

تحديث السجلات

يستخدم الأمر **Update** فى تعديل بيانات حقل أو أكثر فى مجموعة من السجلات داخل جدول. فمثلاً إذا أردنا أن نعدل فى القيمة الموجودة بالحقل **City** للعملاء الذين يقطنون الإسكندرية وهى القيمة "الإسكندرية" لتكون القيمة **'Alex'** فإن العبارة اللازمة لذلك هى:

```
UPDATE Customers SET City = "Alex" Where City = "الإسكندرية"
```

الإجراءات وتكامل البيانات

الإجراء **Transaction** عبارة عن جزء من العمل مع قاعدة البيانات يتضمن العديد من الأجزاء الفرعية البسيطة وهذا الجزء قابل للنجاح أو الفشل بمجمله. ولعل أقرب الأمثلة العملية على ذلك تتمثل فى نقل النقود من حساب فى البنك إلى حساب آخر، حيث تتكون هذه العملية من جزأين، أحدهما هو طرح النقود من الحساب الأول والآخر هو إضافة هذه النقود إلى رقم الحساب الثانى، حيث يجب تنفيذ الجزأين معاً أو فشلهما معاً. فتنفيذ أحدهما فقط يعنى عدم وجود اتزان فى أحد الحسابين.

وحقيقةً تعمل الإجراءات داخل قواعد البيانات على الوصول إلى تكامل البيانات **Data Integrity** حينما تفشل عبارة **SQL** فى التنفيذ. لتوضيح ذلك، نفترض أنك قمت بإضافة ٢٠ عنصراً لأحد طلبات الأمر من خلال ٢٠ عبارة إضافة باستخدام الأمر **INSERT** ولكن لسوء الحظ ولسببٍ من الأسباب لم يتم تنفيذ العبارة رقم ٢٠ فماذا تفعل ؟ الحل الأمثل فى هذه الحالة هو وضع هذه العبارات برمتها داخل إجراء، فإذا تم تنفيذ محتويات الإجراء كاملةً كان به، وإلا يتم التراجع عن جميع العمليات التى تمت داخل الإجراء.

وعلى الرغم من الأهمية الكبيرة للإجراءات **Transactions** إلا أننا لا نحبذ استخدام عبارات **SQL** معها وإنما يكون من الأفضل استخدام كائنات **Visual Basic** كما سنرى فيما بعد. ولأن هذا الفصل يختص أساساً بالحديث عن لغة **SQL**، فسنقوم بإلقاء نظرة خاطفة على العبارات المستخدمة لتعريف الإجراءات باستخدام **SQL** وهى:

- الأمر **BEGIN TRANS** ويتسبب فى بدء تنفيذ إجراء جديد
- الأمر **COMMIT** ويستخدم لإخبار قاعدة البيانات باكتمال الإجراء
- الأمر **ROLLBACK** ويستخدم للتراجع عن الإجراء

فمثلاً للتراجع عن عمليات الحذف الغير مقصودة من خلال إجراء **Transaction**، تابع معنا الخطوات الآتية:

١. قم أولاً ببدء الإجراء وتنفيذ عملية الحذف كما يلى:

```
BEGIN TRANS
DELETE FROM Customers Where FirstName='وليد'
```

٢. بعد ذلك يمكنك التأكد من حذف العميل من خلال الأمر **SELECT** كما يلى:

```
SELECT * FROM Customers Where FirstName='وليد'
```

تجد أن عبارة الاستعلام لا تقوم بإرجاع أى بيانات لحذف جميع العملاء ذوى الاسم "وليد".

٣. قم بتنفيذ الأمر **ROLLBACK** للتراجع عن الإجراء ثم قم بتنفيذ عبارة الاستعلام السابقة مرةً أخرى، تحصل على جميع العملاء ذوى الاسم "وليد" مرةً أخرى، أى تم التراجع عن عملية الحذف. أما إذا أردت حذف هذه السجلات نهائياً، فيمكنك استخدام الأمر **COMMIT** بدلاً من الأمر **ROLLBACK**.