MySQL,
The Comprehensive Course
الدورة الشاملة

إعداد وتقديم: د. عبدالناصر ضياف

05

الاستعلام عن البيانات واسترجاعها
Data Query & Retrieval

- يوفر MySQL إمكانية البحث في البيانات بضوابط متعددة ومتنوعة للحصول على المعلومات المستهدفة وبالشكل والنمط المطلوبين،
  - كل ذلك من خلال تعليمة `SELECT` وملحقاتها

تعليمة استجلاب البيانات SELECT

```
SELECT <expression_1>, <expression_2>, …
 [FROM <table_name>
  [WHERE <condition/criteria>]];
```

- هذه التعليمة تقوم باسترجاع قيم التعابير المحددة لكل الصفوف المُحققة للشروط بالجدول المحدد
- التعبير expression قد يمثل عمود في الجدول أو ثابت أو عملية هجينة تُنتج قيمة
- تعليمة `SELECT` تقوم بإرجاع فئة من السجلات تُدعى بـ recordset أو resultset
- تُحدد عبارة select ما سيُعرض من أعمدة بينما تُحدد عبارة where الصفوف
- بالرغم من أن لغة SQL غير حساسة لحالة الأحرف case-insensitive إلا أنه يُنصح باستخدام الحروف الكبيرة كعُرف في كتابة الكلمات المحجوزة باللغة

تعليمة استجلاب البيانات SELECT

- لاستجلاب عمود اللقب lastName فقط من جدول الموظفين employees نعطي التعليمة التالي:

```
SELECT lastName FROM employees;
```

حيث تكون النتيجة مشابهة للتالي:

```
+-----------+
| lastName  |
+-----------+
| Murphy    |
| Patterson |
| Firrelli  |
...
```

- ولاستجلاب كل من اللقب والاسم الأول والوظيفة نعطي التعليمة التالية:

```
SELECT lastName, firstName, jobTitle FROM employees;
```

حيث تكون النتيجة مشابهة للتالي:

```
+-----------+-----------+----------------------+
| lastName  | firstName | jobTitle             |
+-----------+-----------+----------------------+
| Murphy    | Diane     | President            |
| Patterson | Mary      | VP Sales             |
| Firrelli  | Jeff      | VP Marketing         |
...
```

تعليمة استجلاب البيانات SELECT

- يمكن استخدام رمز * كما في `SELECT *` للتعبير على اختيار جميع الأعمدة بالجدول وتسمى عادة بـ select star أو select all، فالتعليمة التالية:

```
SELECT * FROM employees;
```

مشابهة تمامًا لهذه التعليمة:

```
SELECT employeeNumber, lastName, firstName, extension,
   email, officeCode, reportsTo, jobTitle FROM employees;
```

- ويمكن لتعليمة SELECT ألا تحتوي على فقرة FROM عندما لا تمثل الجداول مصادر بيانات كاستدعاء دالة مستقلة أو عملية حسابية أو متغير نظام مثل:

```
SELECT version(), 2*PI(), current_date;
```

version()	2*PI()	current_date
8.0.31	6.283185	2023-10-31

- ترتيب النتائج Sorting Resulsets
- سرد القيم الفريدة Listing Unique Values
- دوال المجاميع Aggregate Functions
- تجميع البيانات Grouping Data

ترتيب النتائج Sorting Resultsets

- العبارة ORDER BY تستخدم بشكل خاص عندما يكون ترتيب العرض مهمًا
- تُمكنك هذه العبارة من اختيار نوع الترتيب
  - تصاعديًا من خلال استخدام ASC
  - تنازليًا أو عكسيًا من خلال استخدام DESC

  مع العلم بأن الترتيب التلقائي هو التصاعدي
- النمط العام:

```
SELECT column_list
FROM table_list
[WHERE condition]
[ORDER BY exp_1 [ASC | DESC], …, exp_n [ASC | DESC]];
```

ترتيب النتائج Sorting Resultsets

- تنبيه: عبارة ORDER BY يجب دائمًا أن تكون في آخر تعليمة SELECT
- المطلوب عرض قائمة بألقاب وأسماء كل مندوبي الزبائن من جدول الزبائن مرتبة تصاعديًا حسب اللقب

```
SELECT contactLastname, contactFirstname
FROM customers
ORDER BY contactLastname ASC;
```

- المطلوب عرض قائمة بألقاب وأسماء كل مندوبي الزبائن من جدول الزبائن مرتبة تنازليًا حسب اللقب وتصاعديًا حسب الاسم

```
SELECT contactLastname, contactFirstname
FROM customers
ORDER BY contactLastname DESC, contactFirstname;
```

- الترتيب التنازلي أو العكسي مهم جدا في الحياة العملية خصوصًا في العمليات التجارية والمالية والمصرفية وغيرها عندما يتم عرض العمليات الحديثة أولاً أو ذات القيم الكبيرة أولاً.. إلخ

ترتيب النتائج Sorting Resultsets

- المطلوب عرض قائمة برقم الطلبية وخط الإنتاج وإجمالي القيمة من جدول تفاصيل الطلبيات مرتبة تنازليًا حسب إجمالي القيمة

```
SELECT orderNumber, orderlinenuber, quantityOrdered * priceEach
FROM orderdetails
ORDER BY quantityOrdered * priceEach DESC;
```

- المطلوب عرض القيم المالية المدفوعة وتاريخ سدادها بحيث تظهر العمليات الحديثة أولاً.

```
SELECT amount, paymentDate
FROM payments
ORDER BY paymentDate DESC;
```

- تنبيه: القيم المنعدمة *null* في عمود أو أعمدة الترتيب يتم عرضها أولاً أو آخرًا اعتمادًا على نظام إدارة قواعد البيانات المستخدم
  - يعتبر *null* القيمة الأصغر في MySQL من حيث الترتيب فقط.

ترتيب النتائج Sorting Resultsets

- في حال كان لدينا ترتيب نصي خاص بنا ولا يمكن تطبيق التسلسل الهجائي عليه، يمكننا هنا الاستعانة بدالة `FIELD()` لمساعدتنا في ذلك،
  - تقوم هذه الدالة بإرجاع موضع النص المعطى في قائمة النصوص الممثلة للترتيب ابتداءًا من 1، فإذا لم يكن النص موجودًا في القائمة فإنها تُرجع 0.
- المطلوب عرض أرقام وتواريخ وحالات طلبيات الزبائن مرتبة حسب حالاتها والتي هي على التوالي: (**تحت الإجراء، انتظار، ملغية، جاهزة، محل نزاع، تم شحنها**).

```
SELECT orderNumber, orderDate, status FROM orders
ORDER BY FIELD(status,
'In Process', 'On Hold', 'Cancelled', 'Resolved', 'Disputed',  'Shipped');
```

- تذكر: لديك نوع بيانات يغنيك عن الحاجة لاستخدام هذه الدالة وهو ENUM.

سرد القيم الفريدة Listing Unique Values

- قد نحتاج لمعرفة الزبائن الذين تعاملوا معنا خلال فترة ما بغض النظر عن حجم التعامل
- تعليمة SELECT البسيطة لن تكون مجدية في حال احتوى الجدول مئات أو آلاف الحركات في هذه الفترة والتي تحوي تعدديةً حتى على مستوى الزبون
- لذا سيكون الحل الوحيد هو المرور على المخرجات يدويا وصفًا صفًا لاستخلاص أسماء الزبائن وتجنب التكرار
- لحسن الحظ فإن SQL قدمت لنا هذه الأداة من خلال استخدام عبارة **DISTINCT**
- تقوم **DISTINCT** بإنتاج قائمة بتلك القيم التي تختلف عن بعضها البعض

سرد القيم الفريدة Listing Unique Values

- المطلوب الحصول على قائمة بأرقام الزبائن الذين تعاملوا معنا ماليًا اعتبارًا من 1/1/2005 بغض النظر عن حجم التعامل

```
SELECT DISTINCT customerNumber
FROM payments
WHERE paymentDate >= '2005-01-01';
```

- **تذكّر:** يتم التعامل مع *null* كما لو كانت قيمة من القيم (فيما يخص DISTINCT)

مجاميع بسيطة Simple aggregates

- تقدم SQL عدد من الدوال المفيدة والتي تقوم بـ
  - التعداد Counting
  - البحث عن القيم الكبرى والصغرى Finding maximum and minimum
  - حساب المجموع والمتوسط Computing summation and average

- تسمح SQL باقتصار تنفيذ الاستفسار query على المدخلات الغير متكررة أو المدخلات التي يمكن تجميع أو تصنيف تكراراتها على هيئة مجموعات.

دوال المجاميع Aggregate Functions

- بإمكان SQL إنجاز خُلاصات متنوعة كتعداد الصفوف التي تحقق شرط أو معايير معينة، أو إيجاد القيم الكبرى أو الصغرى لحقلٍ أو عمودٍ ما، أو جمع قيم الحقل أو العمود، أو حساب المتوسط له.

الدالة	الخرج
COUNT	عدد الصفوف التي تحتوي قيمًا غير منعدمة non-null
MIN	القيمة الصغرى في العمود المُعطى
MAX	القيمة العظمى في العمود المُعطى
SUM	مجموع كل القيم في العمود المُعطى
AVG	المتوسط الحسابي لجميع القيم في العمود المُعطى

- **تذكر:** هذه الدوال تُنتج قيمة واحدة اعتمادًا على القيم الموجودة بالجدول

دوال المجاميع Aggregate Functions

- COUNT: تستخدم للحصول على العدد الكلي للقيم الغير منعدمة non-null للعمود المعني
  - يمكن استخدام COUNT مقترنًا مع DISTINCT
- المطلوب معرفة عدد الزبائن الذين تعاملوا معنا ماليًا اعتبارًا من 1/1/2005 بغض النظر عن حجم التعامل

```
SELECT COUNT(DISTINCT customerNumber)
FROM payments
WHERE paymentDate >= '2005-01-01';
```

  - قارن مخرجات التعليمة السابقة مع مخرجات التعليمة التالية:

```
SELECT COUNT(*)
FROM payments
WHERE paymentDate >= '2005-01-01';
```

دوال المجاميع Aggregate Functions

- MAX & MIN:
  - التعليمة التالية تبدو صحيحة لسرد العمليات المالية للأعلى قيمةً! فهل هي فعلاً كذلك؟

```
SELECT * FROM payments WHERE amount=MAX(amount);
```

  - هذا غير ممكن إذ يجب أولاً استخلاص القيمة العظمى من الجدول قبل الاعتماد عليها
  - لا يتم ذلك إلا من خلال تعليمة SELECT
  - لإنجاز ذلك نحتاج لاستخدام الاستفسار المتداخل Nested Query
  - ستتكون التعليمة من استفسارين:
    - داخلي Inner Query
    - خارجي Outer Query

دوال المجاميع Aggregate Functions

- يمثل دائمًا الـ outer query الاستفسار العام المطلوب وتمثل الـ inner queries الاستفسارات المطلوبة لتنفيذ الاستفسار الخارجي
- يتم تنفيذ الاستفسارات الداخلية أولا ثم الاستفسار الخارجي
- **تذكر:** أول SELECT تعترضك تعتبر هي الاستفسار الخارجي
- إذًا تكون الصورة الصحيحة للتعليمة السابقة هي:

```
SELECT * FROM payments
WHERE amount = (SELECT MAX(amount) FROM payments);
```

- يمكن استخدام MAX, MIN مع التواريخ لإيجاد أحدث تاريخ أو أقدم تاريخ
- **تمرين:** إيجاد تاريخ آخر طلبية أو أول طلبية افتُتح بها العمل.

دوال المجاميع Aggregate Functions

- SUM & AVG:
- **تذكر:** يمكن للدوال MIN, MAX, AVG, SUM أن تتعامل مع تعبيرات رياضية
- ألمطلوب معرفة حجم المعاملات المالية اعتبارًا من 1/1/2005.

```
SELECT SUM(amount)
FROM payments
WHERE paymentDate >= '2005-01-01';
```

- المطلوب معرفة متوسط قيم المعاملات المالية اعتبارًا من 1/1/2005.

```
SELECT AVG(amount)
FROM payments
WHERE paymentDate >= '2005-01-01';
```

تجميع البيانات Grouping Data

- تستخدم عبارة GROUP BY بشكل عام عند وجود أعمدة مرتبطة مع دوال مجاميع Aggregate Functions في تعليمة SELECT

```
SELECT column_list
FROM table_list
[WHERE condition]
[GROUP BY column_list
[HAVING condition]]
[ORDER BY column_list [ASC | DESC]];
```

- المطلوب حساب حجم المعاملات المالية لكل زبون على حدة

```
SELECT customerNumber, SUM(amount)
FROM payments
GROUP BY customerNumber;
```

تجميع البيانات Grouping Data

- **تنبيهات:** عند استخدام GROUP BY:

1. يجب أن تتضمن قائمة الأعمدة في تعليمة SELECT توليفة من أسماء الأعمدة ودوال المجاميع
2. عبارة GROUP BY يجب أن تضم جميع الأعمدة التي تتضمنها تعليمة SELECT
3. تستطيع GROUP BY تضمين أعمدة من الجدول أو الجداول في عبارة FROM حتى ولو لم تظهر في قائمة أعمدة SELECT

- ميزة HAVING الخاصة بعبارة GROUP BY
  - تعمل HAVING عمل WHERE، فبينما تكون فاعلية WHERE على حقول وتعبيرات كل صف مستقل بالجدول أو العلاقة، فإن فاعلية عبارة HAVING تكون على مخرجات عملية GROUP BY

تجميع البيانات Grouping Data

- المطلوب حساب حجم المعاملات المالية لكل زبون مع استثناء من لم تتعدى حجم معاملاتهم العشرة آلاف دولار.

```
SELECT customerNumber, SUM(amount)
FROM payments
GROUP BY customerNumber
HAVING SUM(amount)>=10000;
```

- المطلوب في المثال السابق مرتبًا ابتداءً من الأكبر قيمةً.

```
SELECT customerNumber, SUM(amount) AS s
FROM payments
GROUP BY customerNumber
HAVING s>=10000;
```

تجميع البيانات Grouping Data

- المطلوب حساب حجم المعاملات المالية لكل زبون للفترة من 1/1/2005 مع استثناء من لم تتعدى حجم معاملاتهم العشرة آلاف دولار ومرتبةً ابتداءً من القيمة الأكبر

```
SELECT customerNumber, SUM(amount) AS s FROM payments
 WHERE paymentDate >= '2005-01-01'
GROUP BY customerNumber
HAVING s>=10000
 ORDER BY s DESC;
```

- المطلوب حساب حجم المعاملات المالية لكل زبون لفترة آخر ثلاثة أشهر اعتبارًا من تاريخ آخر عملية مع استثناء من لم تتعدى حجم معاملاتهم العشرة آلاف دولار ومرتبةً ابتداءً من القيمة الأكبر

```
SELECT customerNumber, SUM(amount) AS s FROM payments
 WHERE paymentDate >=
 DATE_SUB((SELECT MAX(paymentDate) FROM payments), INTERVAL 3 MONTH)
GROUP BY customerNumber
 HAVING s>=10000
 ORDER BY s DESC;
```