POWERING POTENTIAL

Data Aggregation

Module 6: Databases

Databases

Module 6: Contents

- Aggregate functions.
- Joins.
- Views.

Objectives

Data aggregation

Discuss how aggregate functions can be used within MySQL:

- What can we use them for?

Describe the different methods of table joins and utilise them to reference table data:

- What does the syntax look like? How can we effectively utilise it within our GAME database?

Create a view of a table in the GAME database:

- What are they primarily used for and are they useful?

Grouping and Nesting

Databases

Module 6: Data Aggregation

Aggregate functions

There are several aggregate functions which aim to get a single result from several rows:

- **COUNT:** Counts the number of fields.
- **SUM:** Gets the sum total of a field.
- **MIN/MAX/AVG:** Gets the minimum/maximum/average value from a field.

```
SELECT COUNT(name) FROM customers;

SELECT SUM(quantity) FROM games;

SELECT MIN(date_placed) FROM orders;
SELECT MAX(date_placed) FROM orders;
SELECT AVG(price) FROM games;
```

GROUP BY

GROUP BY is often used in conjunction with aggregate functions to bring together sections of data:

- The modularity of SQL allows for this sort of thing to work with no issues.
- Grouping by multiple fields is also possible.

For instance, if you wanted to find the most expensive order made by each customer, we could try something like this:

```
SELECT customer_id, MAX(total) AS max_total
FROM orders
GROUP BY customer_id;
```

Nested queries

Sometimes you may need query some data that has been returned by another query.

- We can do this using nested queries, where you simply wrap a query around another one:

```
SELECT customer_id, name, city
FROM customers
WHERE customer_id=(SELECT customer_id FROM orders WHERE order_id=1);
```

Joining tables

Databases

Module 6: Data Aggregation

Joins

Joins are used to combine different tables together, based on common data values:

- Usually, they're used in conjunction with keys, as primary and foreign keys share the same information across multiple tables.

```
SELECT * FROM table1 t1 JOIN table2 t2 ON t1.field1=t2.field2;
```

Basic select statement to list whatever fields we need to present.

Specifies the table we're selecting from and gives the table an alias to make things easier to reference later.

JOIN specifies the table to attach table1 to – we'll give this one an alias as well.

Finally, we use ON to specify which fields to join with, as they share a common value.

Joins with the GAME database

Outcome:

- Work out how joins work within the GAME database and start thinking of other ways to implement them.

Steps:

10 minutes, solo

› Write a query which utilises joins, showing all information on customers that have made orders.

› Where else might we want to use joins in the GAME database?

JOIN types

We have discussed what's called an inner join, but there are different types of joins that can be used:

- **Left outer joins:** These will produce a result that has a row for every row in the 'left' table (the one you write first in the query) regardless of whether there is a match in the 'right' table. Fill any other field with NULL values.
- **Right outer joins:** These will produce a result that has a row for every row in the 'right' table (the one you write second in the query) regardless of whether there is a match in the 'left' table. Fill any other field with NULL values.

```
SELECT c.customer_id, name, address, city,
postcode, email, placed, total
FROM customers c
JOIN orders o
ON c.customer_id=o.customer_id;
```

customer_id	name	address	city	postcode	email	placed	total
1	Simon	256 Byte Street	Nottingham	NG1 1AA	si@mail.co.uk	2019-08-06	45.99
2	Markus	47 Red Tie Road	Detroit	D37R017	markus47@post.com	2019-08-14	37.99
3	Emma	63 Number Lane	London	CM1 7CC	em@letter.box	2020-01-01	113.97
4	Jeremy	132 Islington Row	London	IS1 2BB	jez@islington.co	2018-06-09	64.99
4	Jeremy	132 Islington Row	London	IS1 2BB	jez@islington.co	2018-06-10	79.99
4	Jeremy	132 Islington Row	London	IS1 2BB	jez@islington.co	2018-06-17	45.99
4	Jeremy	132 Islington Row	London	IS1 2BB	jez@islington.co	2018-06-19	91.98

7 rows in set (0.00 sec)

What other joins could we make on these tables? Are they useful?

LEFT OUTER JOIN

cust_id	name	country
A	Aidan	US
B	Brian	CA
C	Cathy	MX
D	Daisy	DE

order_id	cust_id	total
1	A	1539
2	C	1871
3	A	6352
4	B	1456
5	Z	2137

cust_id	name	total
A	Aidan	1539
A	Aidan	6352
B	Brian	1456
C	Cathy	1871
D	Daisy	NULL

```
SELECT c.cust_id, name, total
FROM customers c
LEFT OUTER JOIN orders o
ON c.cust_id=o.cust_id;
```

RIGHT OUTER JOIN

cust_id	name	country
A	Aidan	US
B	Brian	CA
C	Cathy	MX
D	Daisy	DE

order_id	cust_id	total
1	A	1539
2	C	1871
3	A	6352
4	B	1456
5	Z	2137

cust_id	name	total
A	Aidan	1539
A	Aidan	6352
B	Brian	1456
C	Cathy	1871
NULL	NULL	2137

```
SELECT c.cust_id, name, total
FROM customers c
RIGHT OUTER JOIN orders o
ON c.cust_id=o.cust_id;
```

Database snapshots

Databases

Module 6: Data Aggregation

Views

Views are defined from existing tables/views to make data easier to manage:

- Essentially, they are reusable SELECT statements, which avoid the need to type the whole thing.
- Views will follow the most up-to-date data – they access the data at **point-of-execution**.
- Other statements will access data at **point-of-creation**.

```
CREATE VIEW viewname(col1, col2, result)
AS (SELECT col1, col2, (col3/col4*100) AS result
    FROM tablename);
```

Views with the GAME database

Outcome:

- Explore how views work with our GAME database.

Steps:

10 minutes, solo

- Create a view which shows all games that are low in stock (e.g. below 200) and get the total price of all remaining stock.
- Name this view low_stock.

Views with the GAME database

```
mysql> select * from games;
+------------+-----------------------------+----------+-------+------------+------------+
| product_id | title                       | quantity | price | age_rating | released   |
+------------+-----------------------------+----------+-------+------------+------------+
|          1 | Shoot The Cool Gun 9        |     8965 | 79.99 |         18 | 2012-10-10 |
|          2 | Gunbladers XXII             |      546 | 64.99 |         15 | 1999-09-09 |
|          3 | Paint Drying Simulator 2012 |       35 | 37.99 |          3 | 2012-10-02 |
|          4 | Sitar Hero                  |      456 | 45.99 |         12 | 2016-12-24 |
+------------+-----------------------------+----------+-------+------------+------------+
4 rows in set (0.00 sec)
```

```
CREATE VIEW low_stock (title, quantity, total_price)
AS SELECT title, quantity, quantity*price AS total_price
FROM games
WHERE quantity<200;
```

```
mysql> select * from low_stock;
+-----------------------------+----------+-------------+
| title                       | quantity | total_price |
+-----------------------------+----------+-------------+
| Paint Drying Simulator 2012 |       35 |     1329.65 |
+-----------------------------+----------+-------------+
1 row in set (0.00 sec)
```

‘Dumping’ a database

Databases can be ‘dumped’ to a .sql file and re-imported:

- This is useful for us, because we can see both the schema of our Game database and the various tables within it.
- In larger databases, this is useful for more rigorous data analysis.
- When moving to another system, this is also useful for manually moving across a database from one technology to another.

Dump with >

```
mysqldump -u root -proot gamedb > gamedb.sql
```

Restore with <

```
mysql -u root proot gamedb < gamedb.sql
```

Summary

Databases: Module 6

Discuss how aggregate functions can be used within MySQL:

- We can use aggregate functions alongside GROUP BY to find things like the minimum, maximum, and average of field values.

Describe the different methods of table joins, and utilise them to reference table data:

- Inner joins are primarily used but left and right outer joins can also be implemented.

Create a view of a table in the GAME database:

- Views are used to make data from existing tables/views easier to manage.

Thank you for listening

Any questions?