

# the Master Course

{C0DENATION}

# SQL CRUD operations and Aggregate functions

# Learning Objectives

**Understand and use the CRUD commands when working with SQL.**  
**To be able to use aggregate functions in SQL queries.**

# CRUD

**What does CRUD stand for?**

# CRUD

**Create**

**Read**

**Update**

**Delete**

# Databases

We saw how to **create** in the previous session.

Let's look at **reading** from our database.



# SELECT

The **SELECT** command is what we use to read from our database.

**SELECT \* FROM table;**

This command selects **all columns** from a table.

**A more general SELECT query would look like this:**

**SELECT** column\_name **FROM** table;

**SELECT**  
column\_name1,  
column\_name2  
**FROM**  
table;



**We can list the columns we want here separated by commas**



**The WHERE clause gives us more specificity.**

```
SELECT column_name  
FROM table  
WHERE column_name = 'something';
```

```
[mysql> SELECT * FROM staff;
```

first_name	age	gender
dan	33	M
emily	28	F
ben	21	M
charlie	21	F

```
4 rows in set (0.00 sec)
```

```
[mysql> SELECT first_name FROM staff;
```

first_name
dan
emily
ben
charlie

```
4 rows in set (0.01 sec)
```

```
[mysql> SELECT first_name FROM staff WHERE age = 21;
```

```
+-----+
```

```
| first_name |
```

```
+-----+
```

```
| ben        |
```

```
| charlie    |
```

```
+-----+
```

```
2 rows in set (0.00 sec)
```

```
[mysql> SELECT first_name, gender FROM staff WHERE age = 21;
```

```
+-----+
```

```
| first_name | gender |
```

```
+-----+
```

```
| ben        | M      |
```

```
| charlie    | F      |
```

```
+-----+
```

```
2 rows in set (0.01 sec)
```

## Task:

I will give you an SQL script which will create a new database, a new table and insert some data. Double click the file to open it (it will open in MySQL workbench) then run the file.

1. **SELECT everything.**
2. **SELECT only the emails column.**
3. **SELECT only the car column.**
4. **SELECT only the cars which are corvettes.**
5. **SELECT all the columns for Lucian Larrie.**
6. **What car does Loise Wheatcroft Drive?**  
(Write a query that returns **ONLY** the car)
7. **What is the email address for the car owner whose id is 218?**  
(write a query that returns **ONLY** the email address)
8. **Who owns the oldest Golf?**



# UPDATE

The **UPDATE** command is what we use to **update** data in our database.

**UPDATE** table **SET** column = newdata  
**WHERE** column = olddata

**Example using car\_owners table.**

```
UPDATE car_owners SET first_name = 'TommyK'  
WHERE first_name = 'Ashlin';
```

## **TASK:**

**In the car\_owners table, select the person with the id of 6. Update his first name, changing it to 'Ben'.**



**The columns don't have to be the same.  
WHERE targets the correct rows, and SET  
updates the data.**

**Example using car\_owners table.**

```
UPDATE car_owners SET first_name = 'TommyK'  
WHERE car = 'challenger';
```



# DELETE

The **DELETE** command is what we use to **delete** data in our database.

**DELETE FROM** table  
**WHERE** column = data;

**DELETE FROM** table; ←

This on its own will delete all the data from the table.

**A general rule of thumb is to `SELECT` the data first to make sure you are targeting the correct row before deleting anything.**

## **TASK:**

**There should be 512 users in the `car_owners` table. Delete the last one.**

**Remember that deleting data is NOT the same as drop. Deleting all the data in the table will leave an empty table. Dropping a table will completely remove it.**

## String functions.

**We can use functions in SQL syntax to achieve certain results.**

**For example `CONCAT()`**



**Run the command:**

```
SELECT CONCAT(first_name, last_name)  
FROM car_owners;
```



**Run the command:**

```
SELECT CONCAT(first_name, ' ', last_name)  
FROM car_owners;
```



**There is a space here.**

# ALIASES

From the previous command, look at the column name. We can actually call this whatever we want by using an alias, using the **AS** keyword.

**Run the command:**

```
SELECT CONCAT(first_name, ' ', last_name)  
AS full_name FROM car_owners;
```





# DISTINCT

This command allows us to remove duplicates from the returned data.

**SELECT DISTINCT** column **FROM** table;

## Task:

How many unique types of cars are there in the car\_owners table?

```
SELECT DISTINCT car FROM car_owners;
```

## ORDER BY

We can use the **ORDER BY** command to list our returned data in a particular order. By default, numbers are ascending and strings are ascending alphabetically.

**Run the command:**

```
SELECT * FROM car_owners  
ORDER BY car_year;
```

```
SELECT * FROM car_owners  
ORDER BY first_name;
```

**For descending:**

```
SELECT * FROM car_owners  
ORDER BY car_year DESC;
```

```
SELECT * FROM car_owners  
ORDER BY first_name DESC;
```

# LIMIT

We can use the **LIMIT** command to limit the number of rows returns from our queries.

**Run the command:**

```
SELECT * FROM car_owners  
LIMIT 3;
```

```
SELECT * FROM car_owners  
ORDER BY first_name DESC  
LIMIT 5;
```

## **Task:**

**Using the car\_owners data again -**

- 1. SELECT everything.**
- 2. SELECT the first 10 car owners in our table and order them by last name in ascending alphabetical order.**
- 3. SELECT only the car column.**
- 4. Who owns the newest car?**  
(Write a query that returns ONLY that row)
- 5. Who owns the oldest Golf?**  
(write a query that returns only that row)



# LIKE

LIKE allows for better searching...

'%string%'



These are called wildcards

## EXAMPLE

**LIKE** '%da%'

Searches for strings which contain 'da'

**LIKE** 'da%'

Searches for strings which start with 'da'

**LIKE** '%da'

Searches for strings which end with 'da'

Run this command:

```
SELECT email  
FROM car_owners  
WHERE email LIKE '%github%';
```



## Task:

Using the car\_owners data again –

1. How many users have an email ending with **.com**
2. How many users contain the word 'dan' in their first name?

# COUNT()

**COUNT()** counts the number of rows in a table that fit a specific criteria.

```
SELECT COUNT(*) FROM car_owners;
```

COUNT(\*) returns a count of all the rows in this table.

```
SELECT COUNT(*) FROM car_owners  
WHERE car = 'golf';
```

**SELECT** COUNT(column) **FROM** table;

We can use COUNT( ) to count the number of rows in a column.

## GROUP BY

**This command groups rows together when they share identical data.**





```
[mysql> select * from staff;
```

id	employer	name
1	codenation	dan
2	codenation	ben
3	Prodo	Gary
4	ICE	Mark
5	codenation	Leon
6	Prodo	Briony
7	ICE	Duncan
8	codenation	Charlie

```
8 rows in set (0.00 sec)
```



```
[mysql> SELECT employer, count(*) AS employees FROM staff GROUP BY employer;
+-----+-----+
| employer | employees |
+-----+-----+
| codenation |         4 |
| Prodo     |         2 |
| ICE       |         2 |
+-----+-----+
3 rows in set (0.01 sec)
```

You can think of **GROUP BY** like it is creating super rows, **COUNT(\*)** will count the number of rows in each super row. { CN }<sup>®</sup>

## **Task:**

**I will send you a new sql file which will create a new database, a books table and insert some data into it. Run it in MySQL workbench.**

- 1. Use count( ) to return the number of books in the table.**
- 2. How many DISTINCT first names are there in table?**
- 3. How many titles contain 'the'?**

## GROUP BY

**When we use group by, we are essentially grouping our data into rows with unique values.**

**So if I group the authors by their last name...**



```
[mysql> select author_lname from books group by author_lname;
```

```
+-----+
```

```
| author_lname |
```

```
+-----+
```

```
| Lahiri       |
```

```
| Gaiman       |
```

```
| Eggers       |
```

```
| Chabon       |
```

```
| Smith        |
```

```
| Carver       |
```

```
| DeLillo      |
```

```
| Steinbeck    |
```

```
| Foster Wallace |
```

```
| Harris       |
```

```
| Saunders     |
```

```
+-----+
```

```
11 rows in set (0.00 sec)
```



**However...**

**If we look again at the full list of authors,  
we will see a problem here.**





```
[mysql> select author_fname, author_lname from books;
```

```
+-----+-----+
| author_fname | author_lname |
+-----+-----+
| Jhumpa      | Lahiri       |
| Neil       | Gaiman       |
| Neil       | Gaiman       |
| Jhumpa      | Lahiri       |
| Dave       | Eggers       |
| Dave       | Eggers       |
| Michael    | Chabon       |
| Patti      | Smith        |
| Dave       | Eggers       |
| Neil       | Gaiman       |
| Raymond    | Carver       |
| Raymond    | Carver       |
| Don        | DeLillo      |
| John       | Steinbeck    |
| David      | Foster Wallace |
| David      | Foster Wallace |
| Dan        | Harris       |
| Freida     | Harris       |
| George     | Saunders     |
+-----+-----+
19 rows in set (0.00 sec)
```

**There are two different authors with the last name Harris. If we group authors by last name, these two will be grouped together.**



**We can actually group data by more than one value.**

**If we group by both first name AND last name, we will be creating grouped rows where both the first and last names are unique.**





```
[mysql> SELECT author_fname, author_lname FROM books GROUP BY author_fname, author_lname;
```

author_fname	author_lname
Jhumpa	Lahiri
Neil	Gaiman
Dave	Eggers
Michael	Chabon
Patti	Smith
Raymond	Carver
Don	DeLillo
John	Steinbeck
David	Foster Wallace
Dan	Harris
Freida	Harris
George	Saunders

```
12 rows in set (0.01 sec)
```

## Challenge:

- 1. Write a query that returns the number of books each author has written.**
- 2. Write a query that returns the number of books from each release year.**

# MIN()

# MAX()

**These functions will search a column for the minimum or maximum value.**

# Example

```
SELECT MIN(column) FROM table;
```

```
SELECT MAX(column) FROM table;
```

## **Task:**

**Still using the books data.**

- 1. Write a query that returns the lowest number of pages of any book in our book table.**
- 2. Write a query that returns the latest release year of any books in our book table.**

## **Extra:**

**Write a query that returns title and release year of the 3 oldest books. (Hint: you don't need min or max to do this one).**



# Subqueries.

**Sometimes we will need to use the return value of a query inside another query. This happens a lot when we want to link data together. Let's have a look at an example**

```
[mysql> select title, min(pages) from books;
+-----+-----+
| title          | min(pages) |
+-----+-----+
| The Namesake   | 176        |
+-----+-----+
1 row in set (0.01 sec)
```

**If I run this query, it will just return the first title, and the book with the minimum amount of pages – even though the two are not related.**

```
[mysql> SELECT title, pages FROM books WHERE pages = (SELECT MIN(pages) FROM books);  
+-----+-----+  
| title                                     | pages |  
+-----+-----+  
| What We Talk About When We Talk About Love: Stories |    176 |  
+-----+-----+  
1 row in set (0.01 sec)
```

**MySQL will first run the **subquery**, then use that result in the main query. So we get back the actual result we want.**



## **Task:**

**Still using the books data.**

- 1. Write a query that returns the title, and amount of pages for the longest book.**

# SUM(column)

# AVG(column)

**Sum** will add up all the values in a column.

**Avg** will find the average value in a column (so add them all up and divide by how many there are).

**SELECT SUM(column) FROM table;**

**SELECT AVG(column) FROM table;**

## Task:

Still using the books data.

1. Find the combined total of all the pages in our books table.
2. Find the average number of pages from all the books in our books table.
3. What is the **average release year** for all the books in the books table?

# The aggregate functions challenge:

1. Write a query that returns the number of books in the table.
2. Write a query that returns how many books were released in each year.
3. Print out the total number of books in stock.
4. Find the average release year for each author.
5. Write a query that returns the full name of the author who wrote the longest book.

# Revisiting Learning Objectives

**Understand and use the CRUD commands when working with SQL.  
To be able to use aggregate functions in SQL queries.**