

Week 3

Accessing Data

Week 3 Agenda

- Lecture
 - SQL
 - Filtering Data Queries with Logical Operators
- Lab 3 (5%)

SQL Keywords

LIMIT / OFFSET : `SELECT * FROM movie LIMIT 5 OFFSET 7;`

AND : `SELECT * FROM movie WHERE genre='Drama' AND release_year>2020;`

OR : `SELECT * FROM movie WHERE genre='Drama' OR genre='Comedy';`

NOT : `SELECT * FROM movie WHERE NOT genre='Animation';`

LIKE : `SELECT * FROM movie WHERE title LIKE 'The%';`

BETWEEN : `SELECT * FROM movie WHERE title BETWEEN 'A' AND 'J';`

IN : `SELECT * FROM movie WHERE genre IN ('Comedy', 'Drama', 'Romance');`

ORDER BY : `SELECT * FROM movie ORDER BY title;`

IS NULL : `SELECT * FROM movie WHERE release_year IS NULL;`

LIMIT¹

- **LIMIT** is used to specify the number of records to return
 - Example: Return the first 5 movies in the table

```
SELECT * FROM movie LIMIT 5;
```

movie_id	title
1	The Banshees of Inisherin
2	The Truman Show
3	Eternal Sunshine of the Spotless Mind
4	The Dark Knight
5	The Grand Budapest Hotel

Syntax

```
SELECT column_1, column_2, ...  
FROM table_name  
LIMIT <limitNum>;
```

¹ https://www.w3schools.com/sql/sql_ref_limit.asp

OFFSET¹

- **OFFSET** is used to skip the number of rows specified
 - Example: Return the 6th to 10th movies in the table

```
SELECT * FROM movie
      LIMIT 5 OFFSET 5;
```

movie_id	title
6	Spider-Man: Into the Spider-Verse
7	Shrek
8	Spirited Away
9	Incredibles 3
10	Airplane!

- **OFFSET** must be used with **LIMIT**

Syntax

```
SELECT column_1, ...
      FROM table_name
      LIMIT <limitNum>
      OFFSET <offsetNum>;
```

```
SELECT column_1, ...
      FROM table_name
      LIMIT <offsetNum>, <limitNum>;
```

¹ <https://www.guru99.com/limit.html>

Logical Operators¹

- Logical operators are used to refine SQL queries even more
- These operators allow multiple comparisons to be made in 1 query
- Example: Select all Drama or Comedy films

Without Logical Operator

```
SELECT * FROM movie WHERE genre='Drama' ;
```

```
SELECT * FROM movie WHERE genre='Comedy' ;
```

With Logical Operator

```
SELECT * FROM movie WHERE genre='Drama' OR genre='Comedy' ;
```

¹ https://www.w3schools.com/sql/sql_operators.asp#:~:text=SQL%20Logical%20Operators

Logical Operators¹

- Logical operators are used to refine SQL queries even more
- These operators allow multiple comparisons to be made in 1 query
- Example: Select all Animation films under 100 minutes

Without Logical Operator

```
SELECT * FROM movie WHERE genre='Animation';
```

```
SELECT * FROM movie WHERE runtime<100;
```

With Logical Operator

```
SELECT * FROM movie WHERE genre='Animation' AND runtime<100;
```

¹ https://www.w3schools.com/sql/sql_operators.asp#:~:text=SQL%20Logical%20Operators

Logical Operators¹

AND → **TRUE** if all the conditions separated by **AND** is **TRUE**
eg. **WHERE** genre='Animation' **AND** runtime<100;

OR → **TRUE** if any of the conditions separated by **OR** is **TRUE**
eg. **WHERE** genre='Animation' **OR** runtime<100;

IN → **TRUE** if the operand is equal to **one** of a list of expressions
eg. **WHERE** genre **IN** ('Drama', 'Romance', 'Comedy');

LIKE → **TRUE** if the operand **matches a pattern**
eg. **WHERE** title **LIKE** 'The%';

BETWEEN → **TRUE** if the operand is **within the range** of comparisons
eg. **WHERE** runtime **BETWEEN** 60 **AND** 110;

NOT → Displays a record if the **condition(s) is NOT TRUE**
eg. **WHERE** column_name **NOT** <SomeComparison(s)>;

¹ https://www.w3schools.com/sql/sql_operators.asp#:~:text=SQL%20Logical%20Operators

AND¹

- **AND** operator is used to filter records based on more than 1 condition
 - Example: Return all Drama movies released after 2020

```
SELECT * FROM movie
WHERE genre='Drama' AND release_year>2020;
```

title	runtime	genre	release_year
The Banshees of Inisherin	109	Drama	2022

Syntax

```
SELECT column_1, column_2, ...
FROM table_name
WHERE <condition_1> AND <condition_2> AND ...;
```

¹ https://www.w3schools.com/sql/sql_and.asp

OR¹

- **OR** operator is used to filter records based on more than 1 condition
 - Example: Return all Animation movies, but also comedy movies

```
SELECT * FROM movie
WHERE genre='Animation'
OR genre='Comedy';
```

title	runtime	genre
The Grand Budapest Hotel	99	Comedy
Spider-Man: Into the Spider-Verse	116	Animation
Shrek	89	Animation
Spirited Away	125	Animation

Syntax

```
SELECT column_1, column_2, ...
FROM table_name
WHERE <condition_1> OR <condition_2> OR ...;
```

¹https://www.w3schools.com/sql/sql_or.asp

NOT¹

- **NOT** will give the opposite(or negative) result of the comparison it acts on

```
SELECT * FROM movie WHERE genre<>'Animation';
```

Is the same as

```
SELECT * FROM movie WHERE NOT genre='Animation';
```

- **NOT** can be used to exclude rows
- Example:

```
SELECT * FROM movie  
WHERE NOT (genre='Animation' OR genre='Comedy');
```

¹ https://www.w3schools.com/sql/sql_not.asp

OR NOT

- **NOT** with **OR** used to with exclude rows based on more than 1 condition
 - Using **NOT** to get the opposite: Return all movies that are **NOT** Animation or Comedy

```
SELECT * FROM movie
WHERE NOT (genre='Animation'
OR genre='Comedy');
```

title	runtime	genre
The Banshees of Inisherin	109	Drama
The Truman Show	107	Drama
Eternal Sunshine of the Spotless Mind	108	Romance
The Dark Knight	152	Action

- **NOT** will be applied to all statements inside the brackets
- Without brackets the **NOT** is only applied to the first comparison

```
SELECT * FROM movie
WHERE NOT genre='Animation'
OR genre='Comedy';
```

title	runtime	genre
The Banshees of Inisherin	109	Drama
The Truman Show	107	Drama
Eternal Sunshine of the Spotless Mind	108	Romance
The Dark Knight	152	Action
The Grand Budapest Hotel	99	Comedy

AND NOT

- **NOT** can be used without brackets to apply to only the first condition
 - Return all movies that are were released before 2005, but not animation films

```
SELECT * FROM movie
```

```
WHERE release_year<2005 AND NOT genre='Animation' ;
```

movie_id	title	runtime	genre	release_year
2	The Truman Show	107	Drama	1998
3	Eternal Sunshine of the Spotless Mind	108	Romance	2004

Multiple AND/OR

- Queries can contain one or many **AND/OR** operators
- Queries can chain together to check multiple conditions

Example: Return all Animation movies released in 2001 and a runtime over 100 minutes

```
SELECT * FROM movie  
WHERE genre='Animation' AND release_year=2001 AND runtime>100;
```

Example: Return all Action movies, movies released before 2000, also movies with runtime under 100

```
SELECT * FROM movie  
WHERE genre='Action' OR release_year<2000 OR runtime<100;
```

Example: Return all Drama movies, Action movies, released before 2010

```
SELECT * FROM movie  
WHERE (genre='Drama' OR genre='Action') AND release_year<2010;
```

Logical Operators / Truth Tables

NOT

P	!P
true	
false	

AND

P	Q	P AND Q
true	true	
true	false	
false	true	
false	false	

OR

P	Q	P OR Q
true	true	
true	false	
false	true	
false	false	

LIKE¹

- **LIKE** operator is used to search for a specified pattern in a column
 - Example: Return all movies that begin with the word 'The'

```
SELECT * FROM movie
WHERE title LIKE 'The%';
```

title
The Banshees of Inisherin
The Truman Show
The Dark Knight
The Grand Budapest Hotel

- There are two wildcards used in conjunction with the **LIKE** operator
 - The percent sign **%** represents zero, one, or multiple of any characters
 - The underscore sign **_** represents any one, single character

Syntax

```
SELECT column_1, column_2, ... FROM table_name
WHERE column_1 LIKE 'Som_ Text%';
```

¹ https://www.w3schools.com/sql/sql_like.asp

BETWEEN¹

- **BETWEEN** selects values (numbers, text, or dates) within a given range
 - Example: Return all movies from A to J

```
SELECT * FROM movie
WHERE title BETWEEN 'A' AND 'J';
```

title
Eternal Sunshine of the Spotless Mind

- Inclusive: begin and end values are included

Syntax

```
SELECT column_1, column_2, ... FROM table_name
WHERE column_1 BETWEEN <start_value> AND <end_value>;
```

¹ https://www.w3schools.com/sql/sql_between.asp

IN¹

- **IN** operator allows you to specify multiple values in a **WHERE** clause
 - Example: Return all Comedy, Drama, Romance movies

```
SELECT * FROM movie
    WHERE genre IN ( 'Comedy' , 'Drama' , 'Romance' );
```

- **IN** operator is a shorthand for multiple **OR** conditions
- Values may be numbers, text, or dates

Syntax

```
SELECT column_1, column_2, ... FROM table_name
    WHERE column_1 IN (<value_1>, <value_2>, ...);
```

¹ https://www.w3schools.com/sql/sql_in.asp

ORDER BY¹

- **ORDER BY** is used to sort the output in ascending or descending order
 - Example sort all the movies in the table alphabetically

```
SELECT * FROM movie
ORDER BY title;
```

title
Airplane!
Frances Ha
Roller Town

- To reverse the order we add the keyword **DESC**

```
SELECT * FROM movie
ORDER BY title DESC;
```

title
The Benchwarmers
The Babysitter
Shiva Baby

¹ https://www.w3schools.com/sql/sql_orderby.asp

ORDER BY¹

- **ORDER BY** can also sort by multiple columns
 - Example sort all the movie by runtime and then alphabetically

```
SELECT * FROM movie ORDER BY runtime, title;
```

- By using ASC and DESC the order for each column can be set

```
SELECT * FROM movie ORDER BY runtime ASC, title DESC;
```

Syntax

```
SELECT column_1, column_2, ... FROM table_name  
      ORDER BY column_1, column_2, ... ASC|DESC;
```

¹ https://www.w3schools.com/sql/sql_orderby.asp

NULL¹

- An empty field in a table row will be filled with the **NULL** value
- **NULL** tells us there is an absence of data, the data **does not exist**
- It is not possible to test for **NULL** values with comparison operators (=, <, <>)
- We will have to use the **IS NULL** and **IS NOT NULL** operators instead
 - Example: Find all movies without a release year set
 - Example: Find all movies with a valid runtime

```
SELECT * FROM movie
WHERE ??? ;
```

¹ https://www.w3schools.com/sql/sql_null_values.asp

IS NULL¹

- **IS NULL** operator is used to test for empty values (NULL values)
 - Example: Find all movies without a release year set

```
SELECT * FROM movie
      WHERE release_year IS NULL;
```

Syntax

```
SELECT column_1, column_2, ... FROM table_name
      WHERE column_1 IS NULL;
```

- *Always use **IS NULL** to look for **NULL** values since NULL is not compatible with normal comparison operators

¹ https://www.w3schools.com/sql/sql_null_values.asp

IS NOT NULL¹

- **IS NOT NULL** is used to test for non-empty values (NOT NULL values)
 - Example: Find all movies with a valid runtime

```
SELECT * FROM movie  
      WHERE runtime IS NOT NULL;
```

Syntax

```
SELECT column_1, column_2, ... FROM table_name  
      WHERE column_1 IS NOT NULL;
```

¹ https://www.w3schools.com/sql/sql_null_values.asp

SELECT Syntax

```
SELECT column_1, column_2, ...  
    FROM table_name  
    WHERE condition(s)  
    ORDER BY column ASC|DESC  
    LIMIT num_limit  
    OFFSET num_offset;
```


Next (Week 4)

- **Quiz (3%)**
- **Lesson - Aggregate Functions**
- **Lab 4 (6%)**

Terminology

- Logical Operators: Programming-language symbols that denote logical operations
 - In SQL - **AND**, **OR**, **NOT**, **LIKE**, **BETWEEN**, **IN**
- **NULL** - The value to represent the absence of a value

SQL Keywords

LIMIT / OFFSET : `SELECT * FROM movie LIMIT 5 OFFSET 7;`

AND : `SELECT * FROM movie WHERE genre='Drama' AND release_year>2020;`

OR : `SELECT * FROM movie WHERE genre='Drama' OR genre='Comedy';`

NOT : `SELECT * FROM movie WHERE NOT genre='Animation';`

LIKE : `SELECT * FROM movie WHERE title LIKE 'The%';`

BETWEEN : `SELECT * FROM movie WHERE title BETWEEN 'A' AND 'J';`

IN : `SELECT * FROM movie WHERE genre IN ('Comedy', 'Drama', 'Romance');`

ORDER BY : `SELECT * FROM movie ORDER BY title;`

IS NULL : `SELECT * FROM movie WHERE release_year IS NULL;`

Practice

- [W3Schools SQL](#)

AND, OR, NOT, LIKE, BETWEEN, IN, ORDER BY, IS NULL, IS NOT NULL, LIMIT

- [SQL BOLT](#)

SQL Lesson 2: Queries with constraints (Pt. 1)

SQL Lesson 3: Queries with constraints (Pt. 2)

SQL Lesson 4: Filtering and sorting Query results

SQL Review: Simple SELECT Queries