SQL Basics & Querying Data

CODEUS

SELECT

- **Purpose:** Specifies the columns (data) you want to retrieve from a table.
- **Basic Syntax:**

```
SQL
```

```sql
SELECT column1, column2, ...
FROM table_name;
```

- **Example:**

```
SQL
```

```sql
SELECT customer_name, city
FROM Customers;
```

  - Explanation: This will show you the `customer_name` and `city` for all entries in the `Customers` table.

- **Selecting Everything:** Use `*` to select all columns:

```
SQL
```

```sql
SELECT *
FROM Customers;
```

FROM

- **Purpose:** Indicates the table where the data you need is located.
- Always follows the `SELECT` statement.
- **Basic Syntax:**

SQL

```sql
SELECT ...
FROM table_name;
```

- **Example:**

SQL

```sql
SELECT product_name
FROM Products;
```

  - Explanation: The data is in the 'Products' table.

WHERE

- **Purpose:** Allows you to set conditions to filter which rows are returned.
- **Syntax:**

```sql
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- The `condition` can involve comparison operators and logical operators.

- `=` (Equal to)

```sql
SELECT order_id FROM Orders WHERE order_date = '2025-03-17';
```

- `>` (Greater than)

```sql
SELECT product_name FROM Products WHERE price > 50.00;
```

- `<` (Less than)

```sql
SELECT customer_name FROM Customers WHERE age < 30;
```

- `>=` (Greater than or equal to), `<=` (Less than or equal to), `!=` (Not equal to)

WHERE

- `BETWEEN` : Checks if a value is within a range.

```sql
SELECT * FROM Accounts WHERE balance BETWEEN 1000 AND 5000;
```

- `IN` : Checks if a value is one of a list of values.

```sql
SELECT * FROM Customers WHERE city IN ('Kyiv', 'Lviv');
```

- `LIKE` : Used for pattern matching (`%` - any sequence of characters, `_` - one any character).

```sql
SELECT customer_name FROM Customers WHERE customer_name LIKE 'An%';
```

- `IS NULL` / `IS NOT NULL` : Checks for the presence or absence of NULL values.

```sql
SELECT * FROM Customers WHERE contact_details IS NULL;
```

ORDER BY

- **Purpose:** Sorts the result set based on one or more columns.
- **Syntax:**

SQL

```sql
SELECT column1, column2, ...
FROM table_name
ORDER BY column_to_sort [ASC|DESC];
```

- `ASC` : Ascending order (default).
- `DESC` : Descending order.
- **Examples:**

SQL

```sql
SELECT product_name FROM Products ORDER BY product_name;
SELECT customer_name, registration_date FROM Customers ORDER BY registratio
SELECT city, customer_name FROM Customers ORDER BY city ASC, customer_name
```

Order of Writing Arguments (Clauses) in a SQL SELECT Query

- Typical Order of Writing:
  1. `SELECT` (What to select)
  2. `FROM` (From which table)
  3. `WHERE` (What filtering conditions)
  4. `GROUP BY` (How to group the data - we'll discuss later)
  5. `HAVING` (What filtering conditions for grouped data - we'll discuss later)
  6. `ORDER BY` (How to order the results)
  7. `LIMIT` (How many results to return - might be specific to the DBMS)

Example:

SQL

```sql
SELECT c.customer_name,
       b.branch_name,
       COUNT(t.transaction_id) AS total_transactions,
       AVG(t.amount) AS average_transaction_amount
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
JOIN Branches b ON c.city = b.location
JOIN Transactions t ON a.account_number = t.account_number
WHERE t.transaction_date >= '2023-01-01'
  AND t.transaction_date <= '2023-12-31'
  AND a.account_type = 'savings'
GROUP BY c.customer_name, b.branch_name
HAVING COUNT(t.transaction_id) > 2
ORDER BY average_transaction_amount DESC
LIMIT 10;
```

Order of Execution of Arguments (Clauses) at the DBMS Level

Typical Order of Execution:

1. `FROM` (Determines the tables from which to retrieve data)
2. `WHERE` (Filters rows based on the specified conditions)
3. `GROUP BY` (Groups rows that match the `WHERE` conditions)
4. `HAVING` (Filters groups created by `GROUP BY`)
5. `SELECT` (Selects the columns to display after all previous operations)
6. `DISTINCT` (Removes duplicate rows from the result set)
7. `ORDER BY` (Sorts the final result set)
8. `LIMIT` (Limits the number of rows returned)

```
SELECT customer_ID, SUM(total_amount) AS "Total"
FROM orders
WHERE order_date BETWEEN '2022-01-01' AND '2022-03-31'
AND customer_city = 'New York'
GROUP BY customer_id
ORDER BY Total DESC;
```

Order of Execution

CODEUS

Thank you

- Author: Denys Kuchmei
- My LinkedIn: https://www.linkedin.com/in/denys-kuchmei-7a8259208/
- Date: March 2025
- Join Codeus community in Discord
- Join Codeus community in LinkedIn