SQL Tutorial

Hands-on with MySQL Workbench

www.youtube.com/@DataXOdyssey

What is DATA?

- Data is raw information
- Just values, no meaning yet

- Example:
  - Names, Age, Marks, emp id, salary.
  - Is 50,000 salary good or bad?

What is FACT?

- A fact is a calculation or rule applied to data

- Example:
  - Highest salary is 80,000
  - Lowest salary is 50,000

FACT + DATA = Meaning

- Data becomes meaningful when we can compare each value against a fact.

- Example:
  - Employee A earns less
  - Employee B earns the most

What is a DATABASE?

- A database is an organized place where data is stored in a structured way so that it can be "searched, filtered, and analyzed efficiently"

- Example: University DB, Employee DB, Bank DB etc.

Types of DATABASE?

Relational Database (SQL)

- Data is stored in tables made of rows and columns. Tables are connected to each other using relationships
- Example – Employees table, Departments table (Employee belongs to department – that's the relationship)
- Tools: MySQL, PostgreSQL, Oracle
- Real-life Usage: Banks, Companies, Colleges

Non-Relational Database (NoSQL)

- Data is not stored in tables.
- It is stored in formats like documents or key-value pairs
- Example – json file { "name": "Aman", "skills": ["SQL", "Python", "Power BI"]}
- Tools: MongoDB, Cassandra
- Real-life Usage: Mobile apps, Social media apps

What is SQL?

- SQL stands for Structured Query Language
- It allows users to interact with databases to perform CRUD (Create, Read, Update, Delete) operations

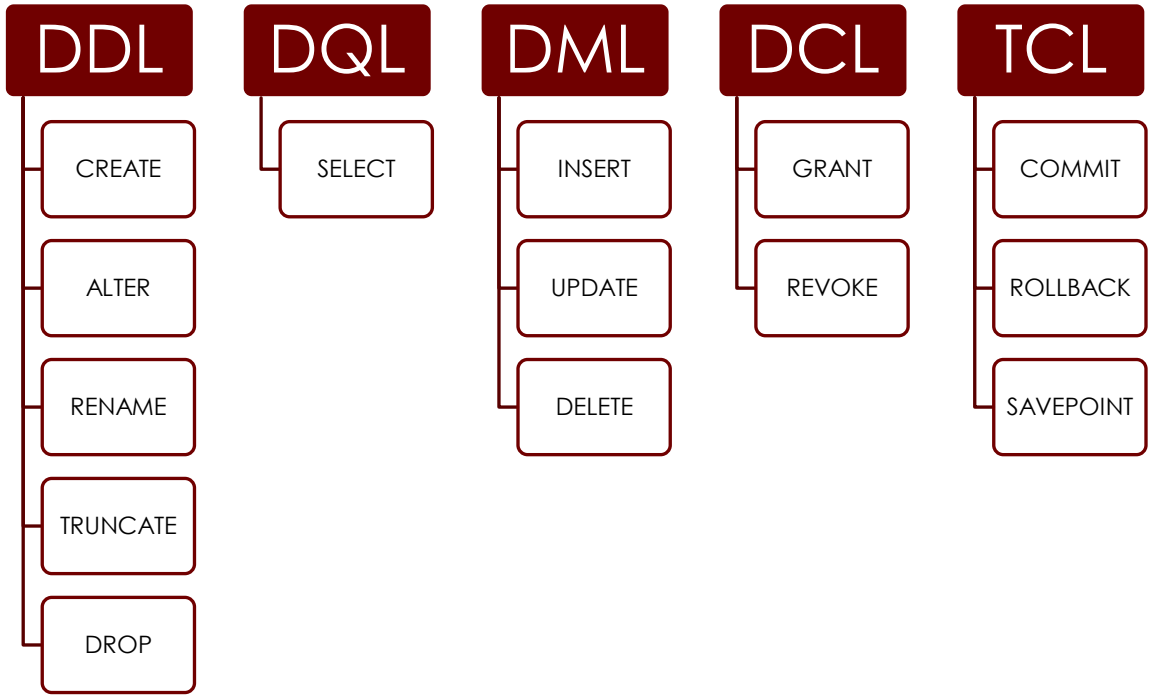
- Example: SELECT * FROM EMPLOYEES
- This command asks the database to show all employee data

PRO Tip:

- SQL is NOT a programming language like Python or Java. It is a query language.
- SQL was known as SEQUEL. Structured English Query Language

Types of SQL Commands

- DDL – Data Definition Language:
  - Creates or changes the structure of the tables
- DQL – Data Query Language:
  - To see/view data, not change it
- DML – Data Manipulation Language:
  - Adds, changes, or deletes data inside tables
- DCL – Data Control Language:
  - Controls who can access the database, and what commands to perform
- TCL – Transaction Control Language:
  - Decides when changes are permanent or when to undo them



www.youtube.com/@DataXOdyssey

PRO Tip:

- SHOW, DESC, USE are **Utility Commands** (not DDL/DML)

Creating or Deleting DATABASE

- Creating a database means creating a container where all our tables and data will live
  - Database = Folder
    - Tables = Files inside the folder

- Syntax:
  - CREATE DATABASE company;
  - CREATE DATABASE IF NOT EXISTS company;
  - DROP DATABASE company;
  - DROP DATABASE IF EXISTS company;

PRO Tip:

- We must create a database first. Without a database, we cannot create tables

Using DATABASE

- Creating a database does not mean we are using it
- We should activate it so that going forward, we can work inside it

- Syntax:
  - USE company;

Creating TABLE

- Creating a table means creating a structure to store data in rows and columns
  - Columns = Headings
  - Rows = Records

- Syntax:
  - CREATE TABLE table_name (

    column_name datatype constraint,

    column_name1 datatype constraint,

    column_name2 datatype constraint);
- Example:
  - CREATE TABLE EMPLOYEES(

    ID INT PRIMARY KEY,

    NAME VARCHAR(50) NOT NULL,

    DATE_OF_BIRTH DATE NOT NULL,

    DEPARTMENT VARCHAR(50),

    SALARY INT);

CONSTRAINTS

Constraints are rules applied on table's columns to make sure correct data is stored.

They help maintain data accuracy and integrity

1. NOT NULL: This constraint ensures that a column cannot have a NULL value.
2. UNIQUE: This constraint ensures that a column or a set of columns contains unique values across all rows in a table.
3. PRIMARY KEY: This constraint is a combination of NOT NULL and UNIQUE and is used to uniquely identify each row in a table.
4. FOREIGN KEY: This constraint ensures that a value in a column (or set of columns) matches a value in the primary key of another table, maintaining a relationship between tables.
5. CHECK: This constraint ensures that a column's value satisfies a specific condition or range of values.
6. DEFAULT: This constraint assigns a default value to a column if no value is provided during an INSERT operation.

Renaming TABLE

1. How to rename existing table
2. How to alter existing table

- Syntax:
  - RENAME TABLE existing_table_name TO new_table_name;
  - ALTER TABLE existing_table_name RENAME TO new_table_name;

- Example:
  - RENAME TABLE EMPLOYEES TO EMP;
  - ALTER TABLE EMP RENAME TO EMPLOYEES;

Altering TABLE Structure

- You can use ALTER TABLE to modify the structure of an existing table

1. Add a column
2. Add a column at specific position
3. Rename a column
4. Modify a column type
5. Drop a column

- Syntax:
  1. ALTER TABLE table_name ADD Col_name DATATYPE, CONSTRAINT;
  2. ALTER TABLE table_name ADD Col_name DATATYPE, CONSTRAINT AFTER col_name;
  3. ALTER TABLE table_name RENAME COLUMN old_Col_name TO new_col_name;
  4. ALTER TABLE table_name MODIFY col_name NEW_DATATYPE;
  5. ALTER TABLE table_name DROP Col_name;

- Example:
  1. ALTER TABLE EMPLOYEES ADD COUNTRY VARCHAR(15);
  2. ALTER TABLE EMPLOYEES ADD GENDER VARCHAR(15) AFTER NAME;
  3. ALTER TABLE EMPLOYEES RENAME COLUMN NAME TO EMP_NAME;
  4. ALTER TABLE EMPLOYEES MODIFY GENDER VARCHAR(10);
  5. ALTER TABLE EMPLOYEES DROP GENDER;

Inserting Data In a Table

1. Insert without specifying column names (Not recommended)
2. Insert by specifying column names (Best Practice)

- Syntax:
  1. INSERT INTO table_name VALUES (col1, col2, col3);
  2. INSERT INTO table_name (col1, col2..) VALUES (col1, col2…)

- Example:
  1. INSERT INTO EMPLOYEES VALUES (1, 'Amit Sharma', '1990-05-12', 'IT', 80000, 'India');
  2. INSERT INTO EMPLOYEES (ID, NAME, DATE_OF_BIRTH, DEPARTMENT, SALARY, COUNTRY)

     VALUES(1, 'Amit Sharma', '1990-05-12', 'IT', 80000, 'India'),

     (2, 'Neha Verma', '1992-08-20', 'HR', 60000, 'India'),

     (3, 'Rahul Singh', '1988-03-15', 'Finance', 90000, 'India');

Deleting Table or Data From a Table

1. DELETE: Remove rows (row-level operation)
2. TRUNCATE: Remove all rows (table-level operation)
3. Remove the table completely

- Syntax:
  1. DELETE FROM table_name WHERE condition;
  2. TRUNCATE TABLE table_name;
  3. DROP TABLE table_name
- Example:
  1. DELETE FROM EMPLOYEES WHERE DEPARTMENT = 'HR';
  2. TRUNCATE TABLE EMPLOYEES;
  3. DROP TABLE EMPLOYEES;

PRO Tip:

- Disable safe mode: SET SQL_SAFE_UPDATES = 0;
- Enable safe mode: SET SQL_SAFE_UPDATES = 1;

UPDATE Command

1. To modify existing data in a table.

- Syntax:
  1. UPDATE table_name SET column_name = value WHERE condition;
- Example:
  1. UPDATE EMPLOYEES SET SALARY = 85000 WHERE ID = 1;
  2. UPDATE EMPLOYEES SET DEPARTMENT = 'IT', COUNTRY = 'India' WHERE ID = 5;

PRO Tip:

- WHERE is optional — but **dangerous if skipped**

SELECT Command

1. View every column and row from employee table
2. View specific columns, lets say, name & department

- Syntax:
  - SELECT * FROM table_name;
  - SELECT col1, col2 FROM table_name;
- Example:
  - SELECT * FROM EMPLOYEES;
  - SELECT NAME, DEPARTMENT FROM EMPLOYEES;

DISTINCT Keyword

1. It is a keyword used to remove duplicate values from the result set.
2. Without DISTINCT → duplicates are shown
   With DISTINCT → duplicates are removed

- Syntax:
  - SELECT DISTINCT column FROM table_name;
  - SELECT DISTINCT column1, column2 FROM table_name;
- Example:
  - SELECT DISTINCT DEPARTMENT FROM EMPLOYEES;
  - SELECT DISTINCT DEPARTMENT, COUNTRY FROM EMPLOYEES;

WHERE Clause

- Filtering the data with where clause

- Syntax:
  - SELECT * FROM table_name WHERE conditions;
  - SELECT col1, col2 FROM table_name where conditions;

- Example:
  - SELECT * FROM EMPLOYEES WHERE DEPARTMENT = 'IT';
  - SELECT NAME, DEPARTMENT FROM EMPLOYEES WHERE SALARY > 80000;

WHERE Clause (with operators)

- Arithmetic Operators : +(addition) , -(subtraction), *(multiplication), /(division), %(modulus)
- Comparison Operators : =, != (not equal to), > , >=, <=
- Logical Operators : AND, OR , NOT, IN, BETWEEN, ALL, LIKE, REGEXP, ANY
- Bitwise Operators : & (Bitwise AND), | (Bitwise OR)

- AND Operator (both conditions must be true)

Q1. Employees from IT department AND salary more than 75,000

```
SELECT * FROM EMPLOYEES WHERE DEPARTMENT = 'IT'
AND SALARY > 75000;
```

- OR Operator (Any one conditions can be true)

Q2. Employees from HR OR Finance

```
SELECT * FROM EMPLOYEES WHERE DEPARTMENT = 'HR' OR
DEPARTMENT = 'Finance';
```

- LIKE Operator (Matches the pattern)

Q3. Employee names starting with A

```
SELECT * FROM EMPLOYEES WHERE NAME LIKE 'A%';
```

WHERE Clause (with operators)

- Arithmetic Operators : +(addition) , -(subtraction), *(multiplication), /(division), %(modulus)
- Comparison Operators : =, != (not equal to), > , >=, <=
- Logical Operators : AND, OR , NOT, IN, BETWEEN, ALL, LIKE, REGEXP, ANY
- Bitwise Operators : & (Bitwise AND), | (Bitwise OR)

- REGEXP Operator (Matches Pattern)

Q6. Employee names starting with A

```
SELECT * FROM EMPLOYEES WHERE NAME REGEXP '^[A-D]';
```

- != or <> (Not equal to) Operator

Q4. Employees NOT working in IT department

```
SELECT * FROM EMPLOYEES WHERE DEPARTMENT != 'IT';
```

- BETWEEN Operator (Checks range inclusive)

Q5. Employees with salary between 70,000 and 85,000

```
SELECT NAME, SALARY FROM EMPLOYEES WHERE SALARY BETWEEN 70000 AND 85000;
```

- NOT IN Operator (Exclude)

Q6. Employees not from India or USA

```
SELECT NAME, COUNTRY FROM EMPLOYEES WHERE COUNTRY NOT IN ('India', 'United States');
```

ORDER By Clause

- Sort the result based on one or more columns
- Sorting can be ascending (ASC) or descending (DESC)
- Default order is ASC if not specified

- Syntax:
  - SELECT column1, column2 FROM table_name ORDER BY column_name [ASC | DESC];
- Example:
  - SELECT NAME, SALARY FROM EMPLOYEES ORDER BY SALARY DESC;

LIMIT & OFFSET Clause

- LIMIT: How many rows to return
- OFFSET: How many rows to skip first

- Syntax:
  - SELECT * FROM table_name LIMIT number;
  - SELECT col1, col2 FROM table_name LIMIT number;
  - SELECT col1, col2 FROM table_name LIMIT number OFFSET number;
- Example:
  - SELECT * FROM EMPLOYEES LIMIT 3';
  - SELECT DEPARTMENT, SALARY FROM EMPLOYEES LIMIT 3;
  - SELECT SALARY FROM EMPLOYEES ORDER BY SALARY DESC LIMIT 1 OFFSET 1;

Functions In SQL (MySQL)

- MySQL has many built-in functions.

SINGLE ROW	MULTI ROW
• STRING • MATH • DATE • CONTROL FLOW • COMPARISON • CONVERSION	• AGGREGATE • WINDOW

PRO Tip:

- Single-row functions return one result per row, while multi-row functions operate on a set of rows

Aggregate Functions

- Aggregate functions perform calculations on a group of rows and return a single value
- They are commonly used with GROUP BY

1. COUNT(): Returns the number of rows
2. MAX(): Returns the highest value
3. MIN(): Returns the lowest value
4. SUM(): Returns the total of values in a column
5. AVG(): Returns the average (mean) value

- Syntax:
  1. COUNT(*), COUNT(1), COUNT('x'), COUNT(column_name)
  2. MAX(column_name)
  3. MIN(column_name)
  4. SUM(column_name)
  5. AVG(column_name)

- Example:
  1. SELECT COUNT(*) AS total_employees FROM EMPLOYEES;
  2. SELECT MAX(SALARY) AS highest_salary FROM EMPLOYEES;
  3. SELECT MIN(SALARY) AS lowest_salary FROM EMPLOYEES;
  4. SELECT SUM(SALARY) AS total_salary FROM EMPLOYEES;
  5. SELECT AVG(SALARY) AS average_salary FROM EMPLOYEES;

GROUP By Clause

- To group rows that have the same values in one or more columns and then apply aggregate functions (like COUNT, SUM, AVG, MIN, MAX) to each group

- Syntax:
  - SELECT column_name, AGGREGATE_FUNCTION(column_name) FROM table_name GROUP BY column_name;

- Example:
  - SELECT DEPARTMENT, COUNT(*) AS total_employees FROM EMPLOYEES GROUP BY DEPARTMENT;
  - SELECT DEPARTMENT, AVG(SALARY) AS avg_salary FROM EMPLOYEES GROUP BY DEPARTMENT;
  - SELECT COUNTRY, SUM(SALARY) AS total_salary FROM EMPLOYEES GROUP BY COUNTRY;

HAVING By Clause

- To filter grouped data after GROUP BY and after aggregate functions are applied
  - WHERE → filters rows.
  - HAVING → filters groups
- Use WHERE → before grouping
- Use HAVING → after grouping.

- Syntax:
  - SELECT column_name, AGGREGATE_FUNCTION(column) FROM table_name GROUP BY column_name HAVING AGGREGATE_FUNCTION(column) condition;
- Example:
  - SELECT DEPARTMENT, AVG(SALARY) AS avg_salary FROM EMPLOYEES GROUP BY DEPARTMENT HAVING AVG(SALARY) > 80000;
  - SELECT DEPARTMENT, COUNT(*) AS emp_count FROM EMPLOYEES WHERE COUNTRY = 'India' GROUP BY DEPARTMENT HAVING COUNT(*) >= 2;

PRO Tip:

- You cannot use aggregate functions in WHERE, but you can use them in HAVING

Logical Order (Query & Execution)

1. This is the logical / writing order used in queries
2. But it actually EXECUTES clauses in this order

- Syntax: (Writing Order)
  - SELECT column(s)
  - FROM table_name
  - WHERE condition
  - GROUP BY column(s)
  - HAVING condition
  - ORDER BY column(s) ASC | DESC
  - LIMIT n OFFSET m;
- Syntax: (Execute Order)
  - FROM
  - WHERE
  - GROUP BY
  - HAVING
  - SELECT
  - ORDER BY
  - LIMIT / OFFSET

PRO Tip:

- Knowing both makes you sound senior

NULL Handing

- NULL means missing / unknown value — not zero, not empty string.

1. Is NULL(): to check NULL values
2. Is NOT NULL(): to check Not NULL values
3. IFNULL(): Replaces NULL with a default value
4. COALESCE(): Returns the first NON-NULL value from a list (left to right)

- Syntax:
  1. column_name IS NULL
  2. column_name IS NOT NULL
  3. IFNULL(expression, replacement_value)
  4. COALESCE(value1, value2, value3, replacement_value)

- Example:
  1. SELECT NAME, COUNTRY FROM EMPLOYEES WHERE COUNTRY IS NULL;
  2. SELECT NAME, COUNTRY FROM EMPLOYEES WHERE COUNTRY IS NOT NULL;
  3. SELECT NAME, IFNULL(COUNTRY, 'Unknown') AS COUNTRY FROM EMPLOYEES;
  4. SELECT NAME, COALESCE(COUNTRY, DEPARTMENT, 'Unknown') AS LOCATION_INFO FROM EMPLOYEES;

PRO Tip:

- Never Use = NULL
- IFNULL replaces NULL using a single fallback value, whereas COALESCE returns the first non-NULL value from multiple options.

DCL Commands: GRANT

- GRANT: To give privileges to a database user. Privileges include:
  - SELECT, INSERT, UPDATE, DELETE ,ALL PRIVILEGES

- Syntax:
  - Creating the user:
    - CREATE USER 'user_name' IDENTIFIED BY 'password';
  - Grant permission to the user:
    - GRANT privilege_name ON database_name.table_name TO 'user_name'';

- Example:
  1. CREATE USER ‘adam' IDENTIFIED BY ‘adam';
  2. GRANT SELECT ON dataxodyssey.employees TO ‘adam';
  3. GRANT SELECT, INSERT, UPDATE ON company.employees TO ‘adam';
  - Verify permission to the user:
    - SHOW GRANTS FOR ‘adam’;
    - SELECT * FROM mysql.tables_priv;

PRO Tip:

- FLUSH PRIVILEGES; Ensures permissions are reloaded immediately

DCL Commands: REVOKE

- REVOKE: To remove previously granted privileges from a database user.

- Syntax:
  - REVOKE privilege_name ON database_name.table_name FROM 'user_name';

- Example:
  1. REVOKE SELECT ON company.employees FROM 'adam';
  2. REVOKE SELECT, INSERT, UPDATE ON company.employees FROM ‘adam';
  3. SHOW GRANTS FOR 'adam';

PRO Tip:

- FLUSH PRIVILEGES; Ensures privilege changes take effect immediately