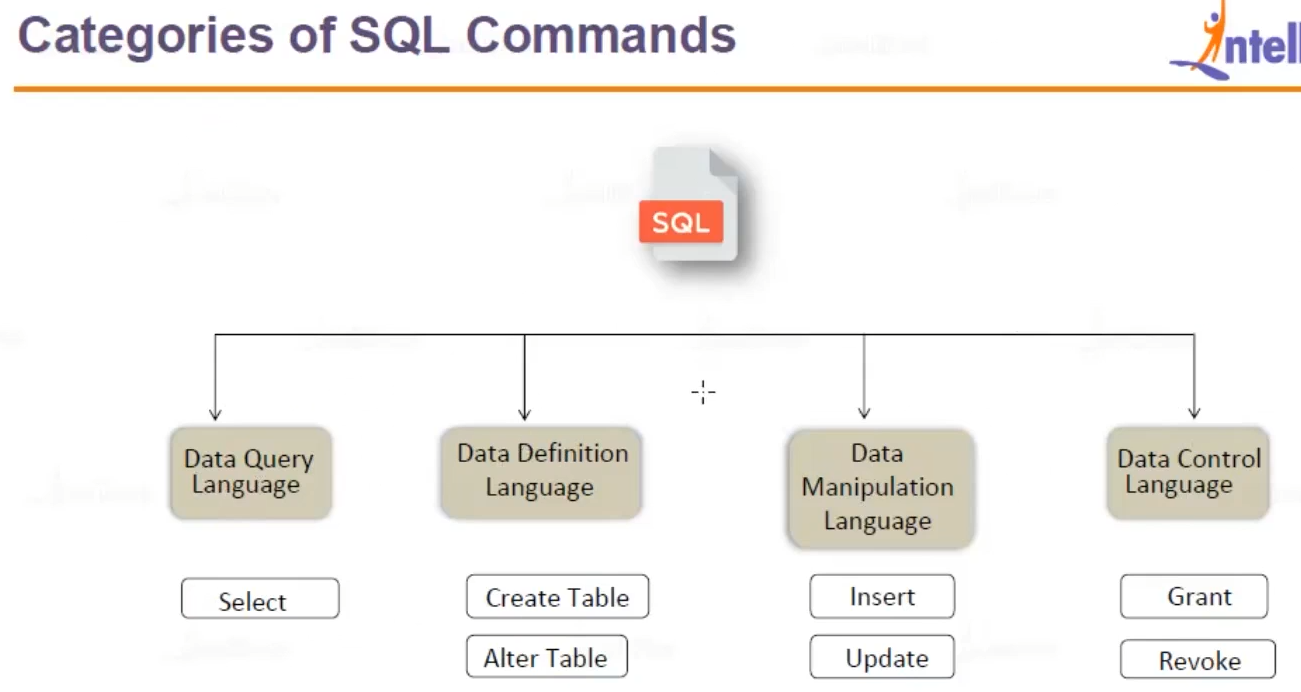
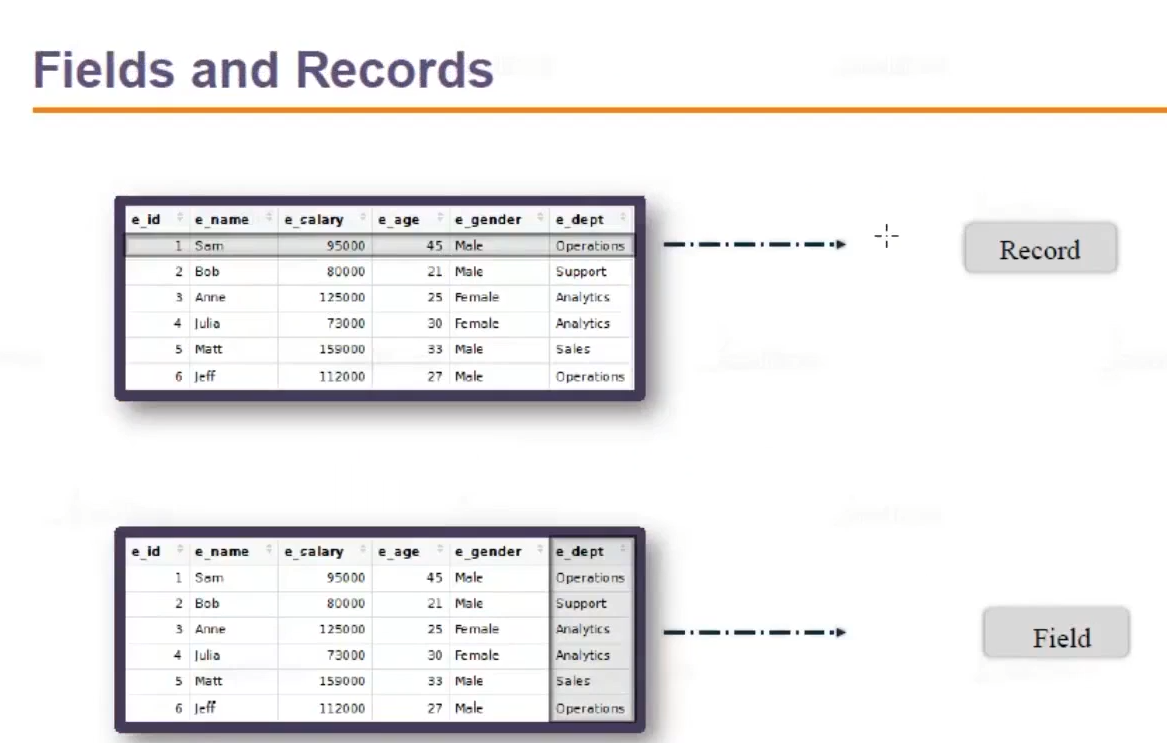
**SQL**

[**https://drive.google.com/drive/folders/1v7ZZcQVoawDTSpPrY2vq6Jo1dN2\_rlMl**](https://drive.google.com/drive/folders/1v7ZZcQVoawDTSpPrY2vq6Jo1dN2_rlMl)

**SQL Chapter 1:**



Firstly we create a DB (Data Base/also called container) in the server followed by creating Tables



The Rows are referred as “Record” and Column are referred as “Field”

Data Base – Tables

1. To create a Data Base :

*Create database Netflix*

Syntax : create database Data base name

1. Connect to the DB:

*Use Netflix*

Syntax : Use Netflix(DB name)

Data Types:

Difference between Data Types.

Data types define the **kind of data** a column can hold in a table.

**🎯 Main Categories:**

| **Category** | **Examples** | **Used For** |
| --- | --- | --- |
| **String** | CHAR, VARCHAR, TEXT | Names, addresses, any text data |
| **Numeric** | INT, FLOAT, DECIMAL | Numbers, prices, quantities |
| **Date/Time** | DATE, TIME, DATETIME | Timestamps, birthdays, durations |
| **Boolean** | BOOLEAN / BIT | True/False flags |
| **Binary** | BLOB, VARBINARY | Storing files or binary data |

**CHAR vs VARCHAR — What's the Difference?**

**✅ CHAR (Fixed-Length String)**

* Stores **fixed-length** text.
* Always uses the **full defined space**, even if the value is shorter (extra spaces are padded).
* Slightly **faster** for fixed-size data because storage is predictable.

**VARCHAR (Variable-Length String)**

* Stores **variable-length** text.
* Only uses space for actual content + 1-2 bytes for length.
* **More space-efficient**, especially for varying lengths of text.

| **Feature** | **CHAR** | **VARCHAR** |
| --- | --- | --- |
| Length | Fixed | Variable |
| Storage | Pads extra space | Stores only input length |
| Speed | Slightly faster for fixed | Slightly slower |
| Use Case | Fixed codes (e.g., PINs) | Names, emails, descriptions |

**Real Interview Question Example**

**Q: When would you use CHAR instead of VARCHAR?**

🧠 **Answer**:

I’d use CHAR when the values are of consistent, fixed length — like country codes (USA, IND) or gender (M, F). It provides slightly better performance. For variable-length fields like names or emails, VARCHAR is better due to space efficiency.

CREATE TABLE char\_vs\_varchar (

char\_col CHAR(5),

varchar\_col VARCHAR(5)

);

INSERT INTO char\_vs\_varchar VALUES ('Hi', 'Hi');

SELECT

char\_col,

LENGTH(char\_col) AS char\_length,

varchar\_col,

LENGTH(varchar\_col) AS varchar\_length

FROM char\_vs\_varchar;

Class On 31/05/2025:

Alias:

**What is an Alias in SQL?**

An **alias** is a temporary name **given to a table or a column** in an SQL query. It’s useful for:

* Making query results **more readable**
* **Renaming columns** in output
* **Shortening table names** for easier joins
* Improving clarity in **complex queries**

🧠 Syntax:

SELECT column\_name AS alias\_name

FROM table\_name;

SELECT t.column\_name

FROM table\_name AS t;

🔹 Note: AS is optional in many databases — SELECT column\_name alias\_name also works.

🧪 Example 1: **Column Alias**

SELECT first\_name AS name, salary AS monthly\_salary

FROM employees;

name | monthly\_salary

-----------|----------------

Alice | 5000

**🧪 Example 2: Table Alias**

SELECT e.first\_name, d.department\_name

FROM employees AS e

JOIN departments AS d ON e.dept\_id = d.id;

Here:

* e is an alias for employees
* d is an alias for departments

✅ Makes joins cleaner and easier to read.

**💡 Real-World Use Case:**

SELECT c.customer\_name, o.order\_date, p.product\_name

FROM customers AS c

JOIN orders AS o ON c.customer\_id = o.customer\_id

JOIN products AS p ON o.product\_id = p.product\_id;

🔹 Without aliases, the query would be long, repetitive, and hard to read.

Clauses:

**Where clause**: Used to **filter** the table before aggrigationg and grpupping.

**What is the WHERE Clause in SQL?**

The WHERE clause is used to **filter rows** before any grouping or aggregation happens. It tells SQL which rows to include in the query results based on a condition.

**✅ Syntax:**

SELECT column1, column2

FROM table\_name

WHERE condition;

Where clause is commonly used with:

| **Condition Type** | **Example** |
| --- | --- |
| Equality / Inequality | salary = 50000, department != 'HR' |
| Comparison | salary > 40000, age <= 30 |
| Logical | AND, OR, NOT |
| Pattern matching | name LIKE 'A%' |
| NULL checks | email IS NOT NULL |
| IN list | department IN ('HR', 'IT') |
| BETWEEN | salary BETWEEN 30000 AND 60000 |

**All About SQL Clauses:**

Clauses are **building blocks of SQL queries** — each clause serves a specific purpose like filtering, grouping, ordering, or limiting data.

| **Clause** | **Purpose** |
| --- | --- |
| SELECT | Choose which ***columns*** to display |
| FROM | Choose which ***table*** to query |
| WHERE | Filter rows ***before grouping*** |
| GROUP BY | Group ***rows*** for aggregation |
| HAVING | ***Filter groups (after aggregation)*** |
| ORDER BY | Sort the results |
| **LIMIT / TOP** | Limit the number of rows returned |
| JOIN | Combine rows from ***multiple tables*** |
| UNION | Combine results of ***multiple queries*** |
| **OFFSET** | Skip a number of rows |
| DISTINCT | Remove duplicate rows |

Execution order of SQL Clauses:

🔸 Execution Order of SQL Clauses (Important for Interviews!)

| **Order** | **Clause** | **Description** |
| --- | --- | --- |
| 1 | FROM | Select the source table |
| 2 | WHERE | Filter rows |
| 3 | GROUP BY | Group the filtered rows |
| 4 | HAVING | Filter the groups |
| 5 | SELECT | Choose the columns to return |
| 6 | ORDER BY | Sort the result |
| 7 | LIMIT | Limit the number of rows returned |

🔸 WHERE vs HAVING – Key Difference

| **Feature** | **WHERE** | **HAVING** |
| --- | --- | --- |
| Applies to | Individual rows | Groups |
| Used with | Any SELECT query | Must be used with GROUP BY |
| Example | WHERE salary > 50000 | HAVING SUM(salary) > 50000 |

Order of writing the SQL Queries:

**SQL QUARIES AND FUNCTIONS USED:**

“—” Used to commend

Eg: --create DB

syntax to create Database.

“create database Netflix(data base name)”

syntax to connect to the database

“use Netflix (data base name)”

syntax for creating a Table

“create table employee (Id int , Name varchar(255), age int, Gender varchar(255), Salary int,

Department varchar (255),city varchar(255)”

Select table whole table

“Select\* from employee”