**Day - 6**

* Query Optimization (EXPLAIN plans)
* Window Functions (ROW\_NUMBER, RANK, LAG, LEAD)

✅ **What is Query Optimization?** (Very Easy Version)

Query Optimization means **making your SQL query run faster** by finding the **smartest way** to get the data.

**🎯 Real-Life Example:**

Imagine you want to find a book in a library:

* ❌ **Slow way**: Check every shelf one by one (full scan).
* ✅ **Fast way**: Use the library catalog to go directly to the shelf (using index).

The **SQL engine** also thinks like this. It checks:

* “Should I scan everything?”
* “Can I use an index?”
* “Which table should I read first?”

Then it **chooses the best plan** to run your query faster.

**🔍 Why it’s important:**

* Saves time ⏱️
* Reduces load on the database 💾
* Helps when working with **big data** 📊

**🤔 So Why Index If SQL Is Smart?**

Because…

✅ **Query Optimizer is smart**,  
❌ But **without Index**, it has no shortcut.

It’s like:

* You are Google Maps (optimizer)
* But if there is no road (index), you **still reach** — just **very slowly**

**Final Thought 💡:**

🔑 **Index = Shortcut**  
🧠 **Query Optimizer = Brain**

👉 The **brain needs the shortcut** to do the best job.

**💡 My Doubt:**

Without query optimization, will index not work?

**✅ Answer:**

Index **will still work** — but the **Query Optimizer** is the one who **decides** to use it or not.

**🧠 Think of it like this:**

* 🛣️ **Index** = A shortcut road
* 🧠 **Query Optimizer** = A smart GPS (Google Maps)

🚘 The Index is **just a road**  
🧠 But the Optimizer is what **chooses the best road**

**📌 So:**

* You **create** an index ➡️ Like building a shortcut
* SQL Engine **uses Query Optimization** to decide:
  + Should I use the shortcut (index)?
  + Or should I just go the long way (full table scan)?

**❗ Important:**

🔍 Just creating an index **doesn't guarantee** it will be used  
🧠 Query Optimizer **analyzes each query** and decides if the index is **helpful or not**

🔁 Example: -- You create an index

CREATE INDEX idx\_name ON Employees(Name);

-- Then run this query

SELECT \* FROM Employees WHERE Name = 'Paresh';

If the table is big, the optimizer will likely use the index.  
But if the table is very small, it may choose **not to use the index**, because a full scan is faster.

**✅ Final Line:**

🔑 Index = Shortcut  
🧠 Query Optimizer = Smart brain that decides whether to use shortcut or not  
🔄 Both works **together** to make SQL fast and smart.

**Window Functions (ROW\_NUMBER, RANK, LAG, LEAD)**

**Window Functions** are **SQL tools** that help you **analyze row-by-row** **with context**. Think of them as **smart tools** used for:

**✅ What Window Functions Can Do:**

| **🔍 Task** | **📘 Window Function Helps You...** |
| --- | --- |
| 🥇 **Ranking** students or employees | Use RANK() or ROW\_NUMBER() to know who's 1st, 2nd, etc. |
| 👁 **Look back or forward** | Use LAG() or LEAD() to compare this row with the previous or next row |
| ➕ **Running totals or averages** | Use SUM() or AVG() with OVER() to get totals row by row |
| 🧮 **Count or number rows** | Use COUNT() with OVER() to know how many rows are processed |
| 📊 **Reporting & Dashboard Logic** | Use Window Functions to prepare clean reports without complex joins or subqueries |

**💡 Easy Real-Life Examples:**

| **📌 Question** | **🛠️ Window Function** |
| --- | --- |
| Who came **1st, 2nd, 3rd** in marks? | RANK() |
| What was the **previous month’s sales**? | LAG() |
| What is the **difference from the last row**? | LAG() or math |
| What’s the **total marks so far** for each student? | SUM() OVER() |
| Who got **highest salary in each department**? | RANK() OVER(PARTITION BY department) |

**✅ So yes, Paresh:**

You can easily:

* Find **1st student** ✅
* Compare **with the previous student** ✅
* Add **totals and ranks** ✅
* Generate data for **reports and dashboards** ✅
* There are **many functions** you can use inside a **Window Function**, but they mostly fall into **5 main types**:

**🔹 1. Ranking Functions**

Used for giving ranks or positions within a partition (like: who came 1st, 2nd, 3rd).

| **Function** | **What it does** |
| --- | --- |
| ROW\_NUMBER() | Gives a unique number to each row |
| RANK() | Gives rank, but skips if there are ties |
| DENSE\_RANK() | Gives rank without skipping numbers |
| NTILE(n) | Divides rows into equal n parts (like quartiles) |

**🔹 2. Value Functions**

Used to access other rows (like previous, next, or a specific row).

| **Function** | **What it does** |
| --- | --- |
| LAG() | Shows value from previous row |
| LEAD() | Shows value from next row |
| FIRST\_VALUE() | Shows the first value in the window |
| LAST\_VALUE() | Shows the last value in the window |
| NTH\_VALUE(n) | Shows the nth value in the window |

**🔹 3. Aggregate Functions**

Used to calculate totals or averages across rows — **without grouping**.

| **Function** | **What it does** |
| --- | --- |
| SUM() | Running total |
| AVG() | Running average |
| MIN() / MAX() | Smallest or biggest value |
| COUNT() | Number of rows in the window |

**🔹 4. Analytical Functions *(Optional/Advanced)***

| **Function** | **What it does** |
| --- | --- |
| CUME\_DIST() | Cumulative distribution (percentile ranking) |
| PERCENT\_RANK() | Relative rank from 0 to 1 |
| RANK() & DENSE\_RANK() | Also belong here |

**🔹 5. Window Frame Control**

Not a function, but part of window function usage.

Example:

sql

CopyEdit

SUM(Sales) OVER (ORDER BY Date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)

This controls *how many rows* are in the “window.”

**✅ What is ROW\_NUMBER() in SQL?**

🔢 It gives a **unique number** (like 1, 2, 3...) to **each row**, based on the **order you define**.

It **starts from 1** and increases row by row.

🧾 Output:

| **Name** | **Marks** | **RowNum** |
| --- | --- | --- |
| Aakash | 95 | 1 |
| Charan | 95 | 2 |
| Bhavna | 92 | 3 |
| Deepika | 88 | 4 |

🔎 It just numbers each row after sorting by Marks in descending order.