**Indexing:**

**Clustering Index :**

A clustered index defines the physical order of rows on disk.
The table data itself is stored inside the index structure.

Only ONE clustered index per table.

Primary key is always clustered key.

**if no PRIMARY KEY exists:**

1. Uses first **UNIQUE NOT NULL** index
2. Else creates a **hidden 6-byte row_id**

Helps sorting data from Primary Key.

If no primary key is there in table Create Index on different column name to primary key to get result fast.

- Primary key lookups
- Range scans (BETWEEN, <, >)
- Time-series data
- Join keys

**Non-Clustered Index**

A non-clustered index stores only index keys + a pointer to the actual row.
It does NOT change physical row order.

Filter data using non primary key

CREATE INDEX idx_customer ON orders(customer_id);

**EXPLAIN SELECT**

It shows how MySQL plans to compute it:

- How many rows will be read
- Which indexes are used
- Where filters, joins, sorting, grouping happen
- Whether MySQL will use extra memory or disk

Think of it as a cost breakdown of computation.

**Natural Key:**

A **natural key** is a column that **naturally exists in the business domain** and **uniquely identifies a record**—without being artificially created.

It comes from **real-world data**, not the database system.

**Surrogate Key**

A surrogate key is an artificial, system-generated identifier used to uniquely identify a row.
It has no business meaning and exists only inside the database.

It is used as a primary key to ensure stability, performance, and efficient joins, especially in transactional and analytical systems.

- **Maintenance Commands in MySQL**

Maintenance commands are used to keep tables healthy, performant, and consistent—they don't change business data, but they fix fragmentation, refresh statistics, and validate integrity.

**1.ANALYZE TABLE :**

- Recomputes **index cardinality & distribution**
- Helps the **optimizer choose better plans**

**When to use**

- After large INSERT/UPDATE/DELETE
- When EXPLAIN shows poor index choices

**2.OPTIMIZE TABLE**

- **Rebuilds the table and indexes**
- **Reclaims unused space**

**3.CHECK TABLE**

Check corruption in tables

- **Avoid large OFFSET in pagination**

**Efficient**

Use the last seen key instead of OFFSET.
**Inefficient**: LIMIT ... OFFSET ...

- **Optimization Techniques in MySQL**

  Used to reduce computation and execution time.

  - Avoid SELECT *
  - Filter early (WHERE)
  - Avoid functions on indexed columns
  - Avoid large OFFSET
  - Use keyset pagination