|  |  |  |
| --- | --- | --- |
| **Academic Year:** 2024-25 | **Year:** Second Year | **Semester:** II |
| **PRN No.: 1012412079** | **Name: Ratnajeet Patil** | |
| **Subject:** Database Management System | | |
| **Assignment No.**: 5 |  | |
| **Date:** |  | |

**Lab Assignment: 05**

**Title:** Creating Basic Views:

Create a view that displays only selected columns from a table in your database application.

For example, in an employee database, create a view showing only employee names, job titles, and salaries.

1. Defining indexes to optimize query performance on large datasets.

**Theory:**

**What is View?**

A View in SQL is a virtual table that is based on a result set of a SQL query. It does not store data physically but provides a way to access and manipulate data from one or more tables. Views help in simplifying complex queries, enhancing security, and improving data abstraction.

**Differentiate Table and View**

|  |  |  |
| --- | --- | --- |
| Feature | Table | View |
| Definition | Stores actual data physically in the database | A virtual table that does not store data physically |
| Storage | Occupies storage space | Does not occupy storage space (only query definition is stored) |
| Data Modification | Can be modified (INSERT, UPDATE, DELETE) | Can be modified if it is based on a single table without aggregations or joins |
| Performance | Faster for large datasets | Slower compared to tables as it executes the underlying query each time |
| Security | Provides full access to data | Can restrict access by showing only selected columns/rows |

**Syntax of View Creation**

1. CREATE VIEW view\_name AS

2. SELECT column1, column2, ...

3. FROM table\_name

4. WHERE condition;

5.

**Static View Versus Dynamic View**

|  |  |  |
| --- | --- | --- |
| Type | Static View | Dynamic View |
| Definition | A view that does not change unless explicitly updated | A view that reflects real-time changes in the underlying table |
| Data Update | Data is fixed at the time of creation | Always up-to-date with the latest changes |
| Use Case | Used when a snapshot of data is required | Used when real-time data access is needed |

**What is Index?**

An Index in SQL is a database object that improves the speed of data retrieval operations on a table. It functions similarly to an index in a book, helping the database find records quickly without scanning the entire table.

**How Indexing is useful in Query optimization?**

* Faster Data Retrieval: Indexing reduces the number of rows scanned, making queries faster.
* Efficient Sorting: Indexed columns allow sorting operations (ORDER BY) to execute quickly.
* Improved Search Performance: WHERE clause filters work efficiently on indexed columns.
* Better Join Performance: Indexes improve the speed of JOIN operations between tables.
* Avoiding Full Table Scans: Instead of scanning all rows, the index helps in directly accessing the required data.

**Show Query Execution Screenshots for:**

* Create Basic View from single Table
* Create Basic View from two Tables
* Create Complex View with Sub Queries
* Create View from Calculated Fields

**Example:**

CREATE VIEW my\_transaction AS

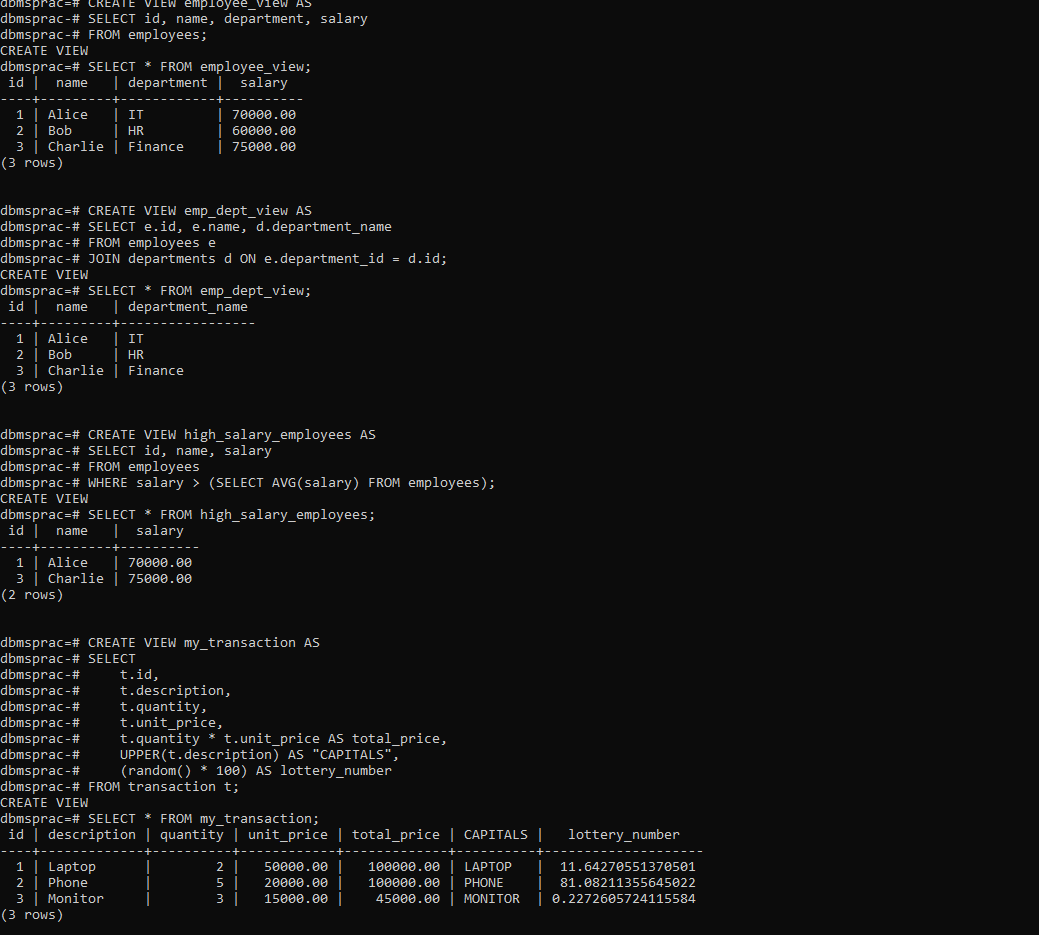
SELECT t.id , t.description , t.quantity , t.unit\_price, t.quantity \* t.unit\_price AS total\_price

, upper(t.description) AS "CAPITALS"

, (random()\* 100) AS lottery\_number

FROM transaction t ;

* Update View and Verify changes in Table
* Update Table and verify changes on created view
* Drop View
* Create Index on Single Attribute
* Create index on multiple attributes
* Create Unique Index
* Display index details
* Drop created index





**FAQs:**

**1. What is the purpose of an index in MySQL?**

An **index** in MySQL is used to improve the performance of queries by allowing faster data retrieval. It works like an index in a book, enabling the database engine to locate data quickly without scanning the entire table.

**2. What are the advantages of using views?**

* **Data Abstraction:** Simplifies complex queries by hiding details.
* **Security:** Restricts access to certain columns or rows without giving access to the whole table.
* **Reusability:** Allows reusable query structures.
* **Simplified Queries:** Reduces repetitive query writing.
* **Logical Data Independence:** Helps in modifying table structures without affecting applications.

**3. What are the limitations of views?**

* **Performance Overhead:** Views don’t store data, so querying complex views may be slower.
* **No Indexing on Views:** Views cannot have indexes directly (except indexed views in some databases).
* **Limited Update Support:** Updating data through views is restricted if they include joins, aggregates, or derived columns.
* **Dependent on Base Table:** If the base table changes, the view may become invalid.

**4. What are the types of indexes in MySQL?**

* **Primary Index:** Automatically created for primary key columns.
* **Unique Index:** Ensures that all values in the indexed column are unique.
* **Composite Index:** An index on multiple columns to speed up queries using those columns together.
* **Full-Text Index:** Optimized for searching text data.
* **Spatial Index:** Used for geographic (GIS) data.
* **Clustered Index (InnoDB only):** Determines how rows are stored physically.

**5. When should you use an index?**

* When querying **large datasets** for faster searches.
* When filtering using **WHERE, GROUP BY, ORDER BY** frequently.
* When using **JOINs** on large tables.
* When enforcing **uniqueness** in columns (e.g., emails, usernames).
* When using **Full-Text Search** on text-heavy columns.