Practical No. 4

Aim:- Create and manage indexes and views in a MySQL database.

High Level Task: Customer Orders Database

Create multiple indexes on a large customer orders table to optimize queries by **customer ID**, **order date**, and **product category**. Write views for frequent **sales reports** and **customer activity summaries**.

Process:

1. **Understand the Scenario:**

- A large e-commerce database where reports and queries are frequently run on orders.
- Need to optimize for searching by customer, date, and product category.

2. **Identify Key Entity:**

- Customer Orders

3. **Define Attributes:**

- order_id, customer_id, order_date, product_category, total_amount

4. **Apply Naming Standards:**

- Tables: tbl_customer_orders
- Columns: col_order_id, col_customer_id, col_order_date, col_product_category, col_total_amount

5. **Establish Relationships:**

- Can later connect to customers and products if needed.

6. **Normalize:**

- Ensure order table is in 3NF (atomic values, no redundancy).

7. **Design Indexes:**
   CREATE INDEX idx_customer_id ON tbl_customer_orders(col_customer_id);
   CREATE INDEX idx_order_date ON tbl_customer_orders(col_order_date);
   CREATE INDEX idx_product_category ON tbl_customer_orders(col_product_category);

8. **Create Views:**

- **Sales Report View**
  CREATE VIEW vw_sales_report AS
  SELECT col_product_category,
  SUM(col_total_amount) AS total_sales,
  COUNT(col_order_id) AS total_orders
  FROM tbl_customer_orders
  GROUP BY col_product_category

- **Customer Activity View**

```
CREATE VIEW vw_customer_activity AS
SELECT col_customer_id,
COUNT(col_order_id) AS total_orders,
SUM(col_total_amount) AS total_spent,
MIN(col_order_date) AS first_order,
MAX(col_order_date) AS last_order
FROM tbl_customer_orders
GROUP BY col_customer_id;
```

9. **SQL Implementation Plan:**

   Create table → Insert data → Apply indexes → Create views.

10. **Verify Schema:**

    Use EXPLAIN to confirm indexes are being used.

Moderate Level Task: University Enrollments Database

Create an index on the student_id column of an enrollments table. Create a view showing student enrollment counts per course.

Process:

1. **Understand the Use Case:**
   - **University system managing course enrollments.**
2. **Entities:**
   - **Enrollments**
3. **Attributes:**
   - **enrollment_id, student_id, course_id**
4. **Naming Standards:**
   - **Table: tbl_enrollments**
   - **Columns: col_enrollment_id, col_student_id, col_course_id**
5. **Normalization:**
   - **Table is in 2NF (atomic values, no redundancy).**
6. **Primary Key / Foreign Keys:**
   - **col_enrollment_id → Primary Key**

- **col_student_id, col_course_id → Foreign Keys (to students and courses)**

7. **Create Index:**

```
CREATE INDEX idx_student_id ON tbl_enrollments(col_student_id);
```

8. **Create View:**

```
CREATE VIEW vw_course_enrollments AS
SELECT col_course_id, COUNT(col_student_id) AS total_students
FROM tbl_enrollments
GROUP BY col_course_id;
```

9. **Execution Plan:**

   Create table → Define constraints → Add index → Build view.

10. **Validation:**
    Run queries with EXPLAIN to check index usage.

Poor Level Task: Contacts Database

Create an index on a **name** column in a simple contacts table. Create a basic view listing all contacts with phone numbers.

Process:

1. **Scenario:**
   - A small contacts database where queries are simple.
2. **Entities:**
   - Contacts
3. **Attributes:**
   - contact_id, name, phone_number
4. **Naming Standards:**
   - Table: tbl_contacts
   - Columns: col_contact_id, col_name, col_phone_number
5. **Primary Key:**
   - col_contact_id
6. **Create Index (Not very useful):**
   sql
   CopyEdit
   CREATE INDEX idx_name ON tbl_contacts(col_name);
7. **Create View:**
   sql
   CopyEdit
   CREATE VIEW vw_contacts_with_phone AS

SELECT col_name, col_phone_number
FROM tbl_contacts;

8. **Normalization:**
    - 1NF satisfied (atomic fields).
9. **Execution:**
    - Create table → Insert records → Add index → Build simple view.
10. **Validation:**

- Ensure view displays correct records.

Level	Scenario	Expected Output
High	Customer Orders DB	- 1 main table (tbl_customer_orders)- 3 indexes (customer_id, order_date, product_category)- Views: Sales Report, Customer Activity- 3NF compliance- Naming conventions applied
Moderate	University Enrollments	- 1 table (tbl_enrollments)- 1 index (student_id)- View: Course-wise enrollment count- Normalized up to 2NF
Poor	Contacts Database	- 1 table (tbl_contacts)- 1 index (name, less useful)- View: Contacts with phone numbers- Only 1NF, minimal design

Viva Questions

- What is the purpose of an index in MySQL?
- How do views help in reporting?
- What are common naming conventions in databases?
- Why is indexing on name column considered a poor practice?
- How do you verify if an index is being used in a query?