

## ***Database Concepts – POC (Proof of Concept)***

*Concepts Covered: Triggers, Views, Indexes, Multiple Key Constraints*

*Technology: MySQL*

*Prepared by: Priyanshu Kumar*

### **1. Introduction**

*SQL (Structured Query Language) is used to manage and manipulate relational databases.*

*This document demonstrates key database concepts such as Triggers, Views, Indexes, and Key Constraints, along with simple Proof of Concepts (POCs) for each.*

### **2. Database and Table Setup**

`CREATE DATABASE newsql;`

*Creates a new database named newsql.*

### **3. Use Database**

`USE newsql;`

*Sets newsql as the active database.*

## 4.Create Users Table

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    gender ENUM('Male', 'Female', 'Other'),
    date_of_birth DATE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

*Creates a table with primary key, unique constraint, enum, and timestamp.*

## 5.Insert Sample Data

```
INSERT INTO users (name, email, gender, date_of_birth)
VALUES ('Priyanshu', 'prnshjh@gmail.com', 'Male', '2004-02-19');
```

*Inserts a sample record into the table.*

## 6.View Records

```
SELECT * FROM users;
```

*Displays all records from the table.*

**7. TRIGGERS** – A trigger is an automatic action that executes when an INSERT, UPDATE, or DELETE occurs on a table.

```
> CREATE TABLE user_audit (
    action VARCHAR(50),
    action_time TIMESTAMP
);
```

*Stores audit logs.*

```
> CREATE TRIGGER after_user_insert
AFTER INSERT ON users
FOR EACH ROW
INSERT INTO user_audit (action, action_time)
VALUES ('USER INSERTED', NOW());
```

*Automatically inserts a log entry whenever a new user is added.*

### Test Trigger

```
INSERT INTO users (name, email) VALUES ("Priyanshu",
'prnsh@gmail.com');
```

```
SELECT * FROM user_audit;
```

*Confirms trigger execution.*

## **8. VIEWS – A view is a virtual table created using a SQL query.**

[Create View](#)

```
CREATE VIEW active_users AS  
SELECT id, name, email FROM users;
```

*Creates a reusable virtual table.*

```
SELECT * FROM active_users;
```

*Fetches data using the view.*

**Why Views:** *Simplifies complex queries, Improves security, Enhances readability*

**9. INDEXES** – Indexes improve query performance by speeding up data retrieval.

`CREATE INDEX idx_email ON users(email);`

*Creates an index on the email column.*

`EXPLAIN SELECT * FROM users WHERE email = 'prnshjh@gmail.com';`

*Shows how index improves query execution.*

**Why Indexes:** *Faster searches, Optimized performance, Used on frequently searched columns*

**10. MULTIPLE KEY CONSTRAINTS** – Constraints ensure data integrity in relational databases.

### Create Parent Table

```
CREATE TABLE departments (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(100)
);
```

### Create Child Table with Foreign Key

```
CREATE TABLE employees (
    emp_id INT,
    dept_id INT,
    emp_name VARCHAR(100),
    PRIMARY KEY (emp_id, dept_id),
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```

*Composite Primary Key, Foreign Key relationship*

### Insert Data

```
INSERT INTO departments VALUES (1, 'Engineering');
INSERT INTO employees VALUES (101, 1, 'John');
```

**Why Constraints:** *Prevent duplicate data, Maintain referential integrity, Enforce business rules*

## **11. Conclusion**

*This document demonstrates practical implementations of core database concepts using SQL.*

*Each POC highlights how these features are used in real-world backend systems to ensure data integrity, performance optimization.*