Unit 1 – database management and web development basics

Week 2:

Session 6- SQL (Triggers)

1.Create a simple trigger BEFORE INSERT or update or delete trigger using the CREATE TRIGGER statement for driver schema assuming the otp for entry.

Step 1: Create

```
mysql> USE driver_db;
Database changed
mysql> CREATE TABLE driver (
    ->     driver_id INT PRIMARY KEY,
    ->     driver_name VARCHAR(50),
    ->     otp INT
    -> );
Query OK, 0 rows affected (0.17 sec)
```

Step 2: BEFORE INSERT Trigger

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER before_driver_insert
    -> BEFORE INSERT ON driver
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.otp IS NULL THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'OTP is required for driver entry';
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.03 sec)

mysql> DELIMITER ;
mysql> INSERT INTO driver VALUES (1, 'Ravi', NULL);
ERROR 1644 (45000): OTP is required for driver entry
mysql> INSERT INTO driver VALUES (1, 'Ravi', 1234);
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM driver;
+-----------+-------------+------+
| driver_id | driver_name | otp  |
+-----------+-------------+------+
|         1 | Ravi        | 1234 |
+-----------+-------------+------+
1 row in set (0.01 sec)
```

Step 3: BEFORE UPDATE Trigger

```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER before_driver_update
    -> BEFORE UPDATE ON driver
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.otp IS NULL THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'OTP is required to update driver details';
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> DELIMITER ;
mysql> UPDATE driver
    -> SET driver_name = 'Ravi Kumar', otp = NULL
    -> WHERE driver_id = 1;
ERROR 1644 (45000): OTP is required to update driver details
mysql> UPDATE driver
    -> SET driver_name = 'Ravi Kumar', otp = 5678
    -> WHERE driver_id = 1;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM driver;
+-----------+-------------+------+
| driver_id | driver_name | otp  |
+-----------+-------------+------+
|         1 | Ravi Kumar  | 5678 |
+-----------+-------------+------+
1 row in set (0.00 sec)
```

Step 4 : DELETE FROM driver

```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER before_driver_delete
    -> BEFORE DELETE ON driver
    -> FOR EACH ROW
    -> BEGIN
    ->     IF OLD.otp IS NULL THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'OTP is required to delete driver record';
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> DELETE FROM driver
    -> WHERE driver_id = 1 AND otp IS NULL;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM driver
    -> WHERE driver_id = 1;
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM driver;
Empty set (0.00 sec)
```

2. Create a simple trigger AFTER INSERT When a customer adds a dish to their order, the system must automatically calculate the price and add it to the order's running total.

Step 1: Create table – dish, orders, order_items

```
mysql> CREATE TABLE dish (
    ->     dish_id INT PRIMARY KEY,
    ->     dish_name VARCHAR(50),
    ->     price DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE orders (
    ->     order_id INT PRIMARY KEY,
    ->     total_amount DECIMAL(10,2) DEFAULT 0
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE order_items (
    ->     order_item_id INT PRIMARY KEY,
    ->     order_id INT,
    ->     dish_id INT,
    ->     quantity INT
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER after_insert_order_items
    -> AFTER INSERT ON order_items
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE dish_price DECIMAL(10,2);
    ->
    ->     -- Get price of the dish
    ->     SELECT price INTO dish_price
    ->     FROM dish
    ->     WHERE dish_id = NEW.dish_id;
    ->
    ->     -- Add price × quantity to order total
    ->     UPDATE orders
    ->     SET total_amount = total_amount + (dish_price * NEW.quantity)
    ->     WHERE order_id = NEW.order_id;
    -> END $$
Query OK, 0 rows affected (0.06 sec)
```

Step 2: After INSERT Trigger

```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER after_insert_order_items
    -> AFTER INSERT ON order_items
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE dish_price DECIMAL(10,2);
    ->
    ->     -- Get price of the dish
    ->     SELECT price INTO dish_price
    ->     FROM dish
    ->     WHERE dish_id = NEW.dish_id;
    ->
    ->     -- Add price × quantity to order total
    ->     UPDATE orders
    ->     SET total_amount = total_amount + (dish_price * NEW.quantity)
    ->     WHERE order_id = NEW.order_id;
    -> END $$
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> DELIMITER ;
mysql> INSERT INTO dish VALUES (1, 'Pizza', 250);
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO orders VALUES (101, 0);
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO order_items VALUES (1, 101, 1, 2);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM orders;
+----------+--------------+
| order_id | total_amount |
+----------+--------------+
|      101 |       500.00 |
+----------+--------------+
1 row in set (0.00 sec)
```

3. Automatically set created_at timestamp

Step 1: Create table

```
mysql> CREATE TABLE users (
    ->     id INT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     email VARCHAR(50),
    ->     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.10 sec)
```

Step 2: Insert

```
mysql> INSERT INTO users (id, name, email)
    -> VALUES (1, 'Amit', 'amit@example.com');
Query OK, 1 row affected (0.06 sec)
```

Step 3: Output

```
mysql> SELECT * FROM users;
+----+------+-----------------+---------------------+
| id | name | email           | created_at          |
+----+------+-----------------+---------------------+
|  1 | Amit | amit@example.com | 2026-02-03 09:50:50 |
+----+------+-----------------+---------------------+
1 row in set (0.00 sec)
```

4. Prevent negative salary entry by Stop inserting or updating an employee record if salary < 0.

Step 1: Create Employee table

```
mysql> CREATE TABLE employee (
    ->     emp_id INT PRIMARY KEY,
    ->     emp_name VARCHAR(50),
    ->     salary DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Step 2: Create BEFORE INSERT Trigger

```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER before_employee_update
    -> BEFORE UPDATE ON employee
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.salary < 0 THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'Salary cannot be negative';
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.06 sec)
```

Step 3: Create BEFORE UPDATE Trigger

```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER before_employee_update
    -> BEFORE UPDATE ON employee
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.salary < 0 THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'Salary cannot be negative';
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.06 sec)
```

Step 4: Test the Trigger

Insert

```
mysql> INSERT INTO employee VALUES (1, 'Ravi', -5000);
ERROR 1644 (45000): Salary cannot be negative
mysql> INSERT INTO employee VALUES (1, 'Ravi', 5000);
Query OK, 1 row affected (0.03 sec)
```

Update

```
mysql> UPDATE employee
    -> SET salary = -2000
    -> WHERE emp_id = 1;
ERROR 1644 (45000): Salary cannot be negative
mysql>
```

5.Maintain audit log table: Whenever an employee record is updated, store old values in an employee_audit table.

Step 1: Create Employee Table and Employee Audit Table

```
mysql> CREATE DATABASE employee_db;
Query OK, 1 row affected (0.06 sec)

mysql> USE employee_db;
Database changed
mysql> CREATE TABLE employee (
    ->     emp_id INT PRIMARY KEY,
    ->     emp_name VARCHAR(50),
    ->     salary DECIMAL(10,2),
    ->     department VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE employee_audit (
    ->     audit_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     emp_id INT,
    ->     old_name VARCHAR(50),
    ->     old_salary DECIMAL(10,2),
    ->     old_department VARCHAR(50),
    ->     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Step 3: Test the Trigger

```
mysql> INSERT INTO employee VALUES (1, 'Ravi', 5000, 'IT');
ERROR 1062 (23000): Duplicate entry '1' for key 'employee.PRIMARY'
mysql> UPDATE employee
    -> SET salary = 7000, department = 'HR'
    -> WHERE emp_id = 1;
Query OK, 0 rows affected (0.06 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> SELECT * FROM employee_audit;
+----------+--------+----------+------------+---------------+---------------------+
| audit_id | emp_id | old_name | old_salary | old_department | updated_at          |
+----------+--------+----------+------------+---------------+---------------------+
|        1 |      1 | Ravi     |    7000.00 | HR             | 2026-02-03 10:22:28 |
+----------+--------+----------+------------+---------------+---------------------+
1 row in set (0.00 sec)
```

SESSION 7 – SQL (Working with Date)

Step 1: create database

```
mysql> CREATE DATABASE date_functions_demo;
Query OK, 1 row affected (0.09 sec)

mysql> USE date_functions_demo;
Database changed
```

Step 2: create table

```
mysql> CREATE TABLE orders (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     customer_name VARCHAR(50),
    ->     start_date DATE,
    ->     end_date DATE,
    ->     created_date DATE
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Step 3: insert values

```
mysql> INSERT INTO orders (customer_name, start_date, end_date, created_date) VALUES
    -> ('Ravi',  '2026-01-01', '2026-01-10', '2026-01-04'),
    -> ('Anu',   '2026-01-05', '2026-01-20', '2026-01-10'),
    -> ('Kiran', '2026-01-08', '2026-01-25', '2026-01-11'),
    -> ('Meena', '2026-01-10', '2026-02-01', '2026-01-12');
Query OK, 4 rows affected (0.06 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Step 4: output

```
mysql> SELECT * FROM orders;
+----+---------------+------------+------------+--------------+
| id | customer_name | start_date | end_date   | created_date |
+----+---------------+------------+------------+--------------+
|  1 | Ravi          | 2026-01-01 | 2026-01-10 | 2026-01-04   |
|  2 | Anu           | 2026-01-05 | 2026-01-20 | 2026-01-10   |
|  3 | Kiran         | 2026-01-08 | 2026-01-25 | 2026-01-11   |
|  4 | Meena         | 2026-01-10 | 2026-02-01 | 2026-01-12   |
+----+---------------+------------+------------+--------------+
4 rows in set (0.05 sec)
```

1.Find number of days between start_date and end_date

```
mysql> SELECT DATEDIFF(end_date, start_date) AS total_days
    -> FROM orders;
+------------+
| total_days |
+------------+
|          9 |
|         15 |
|         17 |
|         22 |
+------------+
4 rows in set (0.01 sec)
```

2.Find the expiry date after 30 days

```
mysql> SELECT
    ->     id,
    ->     customer_name,
    ->     start_date,
    ->     DATE_ADD(start_date, INTERVAL 30 DAY) AS expiry_date
    -> FROM orders;
+----+---------------+------------+-------------+
| id | customer_name | start_date | expiry_date |
+----+---------------+------------+-------------+
|  1 | Ravi          | 2026-01-01 | 2026-01-31  |
|  2 | Anu           | 2026-01-05 | 2026-02-04  |
|  3 | Kiran         | 2026-01-08 | 2026-02-07  |
|  4 | Meena         | 2026-01-10 | 2026-02-09  |
+----+---------------+------------+-------------+
4 rows in set (0.00 sec)
```

3.Date before 7 days from today

```
mysql> SELECT CURDATE() AS today,
    ->        DATE_SUB(CURDATE(), INTERVAL 7 DAY) AS before_7_days;
+------------+---------------+
| today      | before_7_days |
+------------+---------------+
| 2026-02-03 | 2026-01-27    |
+------------+---------------+
1 row in set (0.00 sec)
```

4.Records created on weekend dates

```
mysql> SELECT *
    -> FROM orders
    -> WHERE DAYOFWEEK(created_date) IN (1, 7);
+----+---------------+------------+------------+--------------+
| id | customer_name | start_date | end_date   | created_date |
+----+---------------+------------+------------+--------------+
|  1 | Ravi          | 2026-01-01 | 2026-01-10 | 2026-01-04   |
|  2 | Anu           | 2026-01-05 | 2026-01-20 | 2026-01-10   |
|  3 | Kiran         | 2026-01-08 | 2026-01-25 | 2026-01-11   |
+----+---------------+------------+------------+--------------+
3 rows in set (0.01 sec)
```

5.Month name from input date string

```
mysql> SELECT
    ->     '15-02-2026' AS input_date,
    ->     MONTHNAME(STR_TO_DATE('15-02-2026', '%d-%m-%Y')) AS month_name;
+------------+------------+
| input_date | month_name |
+------------+------------+
| 15-02-2026 | February   |
+------------+------------+
1 row in set (0.00 sec)
```

SESSION 8 – SQL (Indexes and Views)

Create table for index and views :

```
mysql> CREATE TABLE users (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     username VARCHAR(50),
    ->     email VARCHAR(100),
    ->     aadhar VARCHAR(12),
    ->     status VARCHAR(20),
    ->     created_at DATETIME
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO users (username, email, aadhar, status, created_at) VALUES
    -> ('arun', 'arun@gmail.com', '111122223333', 'active', '2026-01-01 10:00:00'),
    -> ('kavi', 'kavi@gmail.com', '444455556666', 'inactive', '2026-01-02 11:30:00'),
    -> ('meena', 'meena@gmail.com', '777788889999', 'active', '2026-01-03 09:15:00');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM users;
+----+----------+-----------------+--------------+----------+---------------------+
| id | username | email           | aadhar       | status   | created_at          |
+----+----------+-----------------+--------------+----------+---------------------+
|  1 | arun     | arun@gmail.com  | 111122223333 | active   | 2026-01-01 10:00:00 |
|  2 | kavi     | kavi@gmail.com  | 444455556666 | inactive | 2026-01-02 11:30:00 |
|  3 | meena    | meena@gmail.com | 777788889999 | active   | 2026-01-03 09:15:00 |
+----+----------+-----------------+--------------+----------+---------------------+
3 rows in set (0.00 sec)
```

1.Improve search performance on email

```
mysql> CREATE INDEX idx_email ON users(email);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM users;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
--------+------------+
| Table | Non_unique | Key_name  | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
Visible | Expression |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
--------+------------+
| users |          0 | PRIMARY  |            1 | id          | A         |           3 |     NULL |   NULL |      | BTREE      |         |               |
YES     | NULL       |
| users |          1 | idx_email |            1 | email       | A         |           3 |     NULL |   NULL | YES  | BTREE      |         |               |
YES     | NULL       |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
--------+------------+
2 rows in set (0.02 sec)
```

2.Create UNIQUE INDEX on username

```
mysql> CREATE UNIQUE INDEX idx_unique_username ON users(username);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM users;
+-------+------------+--------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+-------
--------+---------+------------+
| Table | Non_unique | Key_name           | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_
comment | Visible | Expression |
+-------+------------+--------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+-------
--------+---------+------------+
| users |          0 | PRIMARY            |            1 | id          | A         |           3 |     NULL |   NULL |      | BTREE      |         |
        | YES     | NULL       |
| users |          0 | idx_unique_username |            1 | username    | A         |           3 |     NULL |   NULL | YES  | BTREE      |         |
        | YES     | NULL       |
| users |          1 | idx_email          |            1 | email       | A         |           3 |     NULL |   NULL | YES  | BTREE      |         |
        | YES     | NULL       |
+-------+------------+--------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+-------
--------+---------+------------+
3 rows in set (0.01 sec)
```

3.Find and remove unused index

```
      DROP INDEX idx_email ON users;
Query OK, 0 rows affected (0.03 sec)
Records
mysql> SHOW INDEX FROM users;
+-------+------------+--------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+-------
--------+---------+------------+
| Table | Non_unique | Key_name           | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_
comment | Visible | Expression |
+-------+------------+--------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+-------
--------+---------+------------+
| users |          0 | PRIMARY            |            1 | id          | A         |           3 |     NULL |   NULL |      | BTREE      |         |
        | YES     | NULL       |
| users |          0 | idx_unique_username |            1 | username    | A         |           3 |     NULL |   NULL | YES  | BTREE      |         |
        | YES     | NULL       |
+-------+------------+--------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+-------
--------+---------+------------+
2 rows in set (0.00 sec)
```

4.Fix slow aadhar query using index

```
mysql> CREATE INDEX idx_aadhar ON users(aadhar);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN SELECT * FROM users WHERE aadhar = '111122223333';
+----+-------------+-------+------------+------+---------------+------------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key        | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | users | NULL       | ref  | idx_aadhar    | idx_aadhar | 51      | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

5.Composite index (status, created_at)

```
mysql> EXPLAIN
    -> SELECT * FROM users
    -> WHERE status = 'active'
    -> ORDER BY created_at;
+----+-------------+-------+------------+------+-------------------+-------------------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys     | key               | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+------+-------------------+-------------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | users | NULL       | ref  | idx_status_created | idx_status_created | 83      | const |    2 |   100.00 | NULL  |
+----+-------------+-------+------------+------+-------------------+-------------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

View:

```
mysql> CREATE VIEW active_users AS
    -> SELECT id, username, email, status
    -> FROM users
    -> WHERE status = 'active';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM active_users;
+----+----------+-----------------+--------+
| id | username | email           | status |
+----+----------+-----------------+--------+
|  1 | arun     | arun@gmail.com  | active |
|  3 | meena    | meena@gmail.com | active |
+----+----------+-----------------+--------+
2 rows in set (0.00 sec)
```

SESSION 9 – SQL Built-in Functions

Create User and Employee table :

```
mysql> CREATE TABLE user_details (
    ->     id INT PRIMARY KEY AUTO_INCREMENT,
    ->     username VARCHAR(100),
    ->     contact VARCHAR(50),
    ->     mobile VARCHAR(15),
    ->     phone VARCHAR(15),
    ->     email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE employees (
    ->     emp_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     emp_name VARCHAR(100),
    ->     department VARCHAR(50),
    ->     join_date DATE
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Insert values :

```
mysql> INSERT INTO user_details (username, contact, mobile, phone, email) VALUES
    -> ('arun@gmail.com', '987-654-3210', NULL, '044 234567', 'arun@gmail.com'),
    -> ('meena@yahoo.com', '98 76 54 32', '9876543211', NULL, NULL),
    -> ('kavi@outlook.com', '(987)654-1111', NULL, NULL, 'kavi@outlook.com');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO employees (emp_name, department, join_date) VALUES
    -> ('raVi kuMar', 'IT', '2018-01-10'),
    -> ('MEENA devi', 'HR', '2016-03-15'),
    -> ('aRun sharma', 'IT', '2019-07-01'),
    -> ('kAVI', 'HR', '2020-05-20');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Output:

```
mysql> SELECT * FROM user_details;
+----+------------------+---------------+------------+------------+------------------+
| id | username         | contact       | mobile     | phone      | email            |
+----+------------------+---------------+------------+------------+------------------+
|  1 | arun@gmail.com   | 987-654-3210  | NULL       | 044 234567 | arun@gmail.com   |
|  2 | meena@yahoo.com  | 98 76 54 32   | 9876543211 | NULL       | NULL             |
|  3 | kavi@outlook.com | (987)654-1111 | NULL       | NULL       | kavi@outlook.com |
+----+------------------+---------------+------------+------------+------------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM employees;
+--------+-------------+------------+------------+
| emp_id | emp_name    | department | join_date  |
+--------+-------------+------------+------------+
|      1 | raVi kuMar  | IT         | 2018-01-10 |
|      2 | MEENA devi  | HR         | 2016-03-15 |
|      3 | aRun sharma | IT         | 2019-07-01 |
|      4 | kAVI        | HR         | 2020-05-20 |
+--------+-------------+------------+------------+
4 rows in set (0.00 sec)
```

1.Extract letters before @ and append 123

```
mysql> SELECT
    ->     username,
    ->     CONCAT(SUBSTRING_INDEX(username, '@', 1), '123') AS default_password
    -> FROM user_details;
+------------------+------------------+
| username         | default_password |
+------------------+------------------+
| arun@gmail.com   | arun123          |
| meena@yahoo.com  | meena123         |
| kavi@outlook.com | kavi123          |
+------------------+------------------+
3 rows in set (0.01 sec)
```

2.Correct invalid contact numbers (keep only digits)

```
mysql> SELECT
    ->     contact,
    ->     REGEXP_REPLACE(contact, '[^0-9]', '') AS valid_contact
    -> FROM user_details;
+---------------+---------------+
| contact       | valid_contact |
+---------------+---------------+
| 987-654-3210  | 9876543210    |
| 98 76 54 32   | 98765432      |
| (987)654-1111 | 9876541111    |
+---------------+---------------+
3 rows in set (0.05 sec)
```

3.Normalise employee names (Proper case)

```
mysql> SELECT
    ->     emp_name,
    ->     CONCAT(
    ->         UPPER(SUBSTRING(emp_name, 1, 1)),
    ->         LOWER(SUBSTRING(emp_name, 2))
    ->     ) AS normalised_name
    -> FROM employees;
+-------------+-----------------+
| emp_name    | normalised_name |
+-------------+-----------------+
| raVi kuMar  | Ravi kumar      |
| MEENA devi  | Meena devi      |
| aRun sharma | Arun sharma     |
| kAVI        | Kavi            |
+-------------+-----------------+
4 rows in set (0.00 sec)
```

4.Handle fallback contact priority (Mobile → Phone → Email)

```
mysql> SELECT
    ->     username,
    ->     COALESCE(mobile, phone, email) AS preferred_contact
    -> FROM user_details;
+------------------+-------------------+
| username         | preferred_contact |
+------------------+-------------------+
| arun@gmail.com   | 044 234567        |
| meena@yahoo.com  | 9876543211        |
| kavi@outlook.com | kavi@outlook.com  |
+------------------+-------------------+
3 rows in set (0.00 sec)
```

5.Average experience per department

```
mysql> SELECT
    ->     department,
    ->     AVG(TIMESTAMPDIFF(YEAR, join_date, CURDATE())) AS avg_experience
    -> FROM employees
    -> GROUP BY department;
+------------+----------------+
| department | avg_experience |
+------------+----------------+
| IT         |         7.0000 |
| HR         |         7.0000 |
+------------+----------------+
2 rows in set (0.01 sec)
```

SESSION 10 – SQL User-Defined Functions

Select database :

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
Query OK, 0 rows affected, 1 warning (0.04 sec)

mysql> USE date_functions_demo;
Database changed
```

1.Create a function to calculate net salary after 10% tax deduction

```
mysql> DELIMITER $$
mysql>
mysql> CREATE FUNCTION calc_net_salary(gross_salary DECIMAL(10,2))
    -> RETURNS DECIMAL(10,2)
    -> DETERMINISTIC
    -> BEGIN
    ->     RETURN gross_salary - (gross_salary * 0.10);
    -> END $$
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT calc_net_salary(50000) AS net_salary;
+------------+
| net_salary |
+------------+
|   45000.00 |
+------------+
1 row in set (0.10 sec)
```

2.Function to check user activity (Active / Inactive)

```
mysql> DELIMITER $$
mysql>
mysql> CREATE FUNCTION check_user_activity(last_login DATE)
    -> RETURNS VARCHAR(20)
    -> DETERMINISTIC
    -> BEGIN
    ->     IF DATEDIFF(CURDATE(), last_login) <= 30 THEN
    ->         RETURN 'Active';
    ->     ELSE
    ->         RETURN 'Inactive';
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT check_user_activity('2026-01-15') AS user_status;
+-------------+
| user_status |
+-------------+
| Active      |
+-------------+
```

3. Dynamic Tax Slab Calculation

Salary	Tax %
≤ 3,00,000	0
3,00,001 – 6,00,000	10
6,00,001 – 10,00,000	20
> 10,00,000	30

```
mysql> DELIMITER $$
mysql>
mysql> CREATE FUNCTION calc_tax(salary DECIMAL(10,2))
    -> RETURNS DECIMAL(10,2)
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE tax DECIMAL(10,2);
    ->
    ->     IF salary <= 300000 THEN
    ->         SET tax = 0;
    ->     ELSEIF salary <= 600000 THEN
    ->         SET tax = salary * 0.10;
    ->     ELSEIF salary <= 1000000 THEN
    ->         SET tax = salary * 0.20;
    ->     ELSE
    ->         SET tax = salary * 0.30;
    ->     END IF;
    ->
    ->     RETURN tax;
    -> END $$
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT calc_tax(750000) AS tax_amount;
+------------+
| tax_amount |
+------------+
|  150000.00 |
+------------+
1 row in set (0.00 sec)
```

4.Categorize employees based on experience

Years	Category
< 2	Fresher
2–5	Junior
6–10	Mid
> 10	Senior

```
mysql> DELIMITER $$
mysql>
mysql> CREATE FUNCTION emp_category(exp_years INT)
    -> RETURNS VARCHAR(20)
    -> DETERMINISTIC
    -> BEGIN
    ->     IF exp_years < 2 THEN
    ->         RETURN 'Fresher';
    ->     ELSEIF exp_years BETWEEN 2 AND 5 THEN
    ->         RETURN 'Junior';
    ->     ELSEIF exp_years BETWEEN 6 AND 10 THEN
    ->         RETURN 'Mid';
    ->     ELSE
    ->         RETURN 'Senior';
    ->     END IF;
    -> END $$
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT emp_category(7) AS category;
+----------+
| category |
+----------+
| Mid      |
+----------+
1 row in set (0.00 sec)
```

5.Late fee calculator

- ₹50 per day
- Max ₹1000

```
mysql> DELIMITER $$
mysql>
mysql> CREATE FUNCTION calc_late_fee(due_date DATE, payment_date DATE)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE late_days INT;
    ->     DECLARE fee INT;
    ->
    ->     SET late_days = DATEDIFF(payment_date, due_date);
    ->
    ->     IF late_days <= 0 THEN
    ->         SET fee = 0;
    ->     ELSE
    ->         SET fee = late_days * 50;
    ->         IF fee > 1000 THEN
    ->             SET fee = 1000;
    ->         END IF;
    ->     END IF;
    ->
    ->     RETURN fee;
    -> END $$
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT calc_late_fee('2026-01-01', '2026-01-30') AS late_fee;
+----------+
| late_fee |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```