Week-2

Session 6:

```
DELIMITER //

CREATE TRIGGER trg_commission_audit
BEFORE UPDATE ON Salespeople
FOR EACH ROW
BEGIN
   -- 1. Validation Logic
   IF NEW.commission_rate < 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Commission cannot be negative';
   END IF;

   -- 2. Audit Logging Logic
   INSERT INTO employee_audit(salesperson_id, old_comm, new_comm)
   VALUES (OLD.salesperson_id, OLD.commission_rate, NEW.commission_rate);
END //

DELIMITER ;
```

```
mysql> DELIMITER ;
mysql> UPDATE Salespeople SET commission_rate = -5.00 WHERE salesperson_id = 101;
ERROR 1644 (45000): Commission cannot be negative
mysql> -- Update with a valid value
mysql> UPDATE Salespeople SET commission_rate = 15.00 WHERE salesperson_id = 101;
Query OK, 0 rows affected (0.01 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql>
mysql> -- Check if the change was recorded in the audit table
mysql> SELECT * FROM employee_audit;
+----------+----------------+----------+----------+---------------------+
| audit_id | salesperson_id | old_comm | new_comm | changed_at          |
+----------+----------------+----------+----------+---------------------+
|        1 |            101 |    10.00 |    15.00 | 2026-02-03 10:22:14 |
|        2 |            101 |    15.00 |    15.00 | 2026-02-04 09:36:07 |
+----------+----------------+----------+----------+---------------------+
2 rows in set (0.00 sec)
```

Session 7:

```
-- 1. Find number of days between start_date (hire_date) and today (end_date)
SELECT
   first_name,
   hire_date AS start_date,
   CURDATE() AS end_date,
   DATEDIFF(CURDATE(), hire_date) AS days_difference
FROM Salespeople;

-- 2. Find the expiry date after 30 days (e.g., Warranty Expiry for a sale)
SELECT
   sale_id,
   sale_date,
   DATE_ADD(sale_date, INTERVAL 30 DAY) AS expiry_date
FROM Sales;

-- 3. Get Date before seven days from today
SELECT
   CURDATE() AS today,
   DATE_SUB(CURDATE(), INTERVAL 7 DAY) AS seven_days_ago;

-- 4. Get the records created on weekend dates (Saturday and Sunday)
-- Note: DAYOFWEEK returns 1 for Sunday and 7 for Saturday
SELECT * FROM Sales
WHERE DAYOFWEEK(sale_date) IN (1, 7);

-- 5. Read input string, convert to 'DD-MM-YYYY', and display Month Name
-- We use STR_TO_DATE to parse the string and MONTHNAME to extract the name
SELECT
   '25-12-2025' AS input_string,
   STR_TO_DATE('25-12-2025', '%d-%m-%Y') AS converted_date,
   MONTHNAME(STR_TO_DATE('25-12-2025', '%d-%m-%Y')) AS month_display;
```

```
mysql> SELECT
    ->     first_name,
    ->     hire_date AS start_date,
    ->     CURDATE() AS end_date,
    ->     DATEDIFF(CURDATE(), hire_date) AS days_difference
    -> FROM Salespeople;
+------------+------------+------------+-----------------+
| first_name | start_date | end_date   | days_difference |
+------------+------------+------------+-----------------+
| John       | 2020-01-15 | 2026-02-04 |            2212 |
| Jane       | 2021-03-20 | 2026-02-04 |            1782 |
| Michael    | 2022-06-10 | 2026-02-04 |            1335 |
| Sarah      | 2019-11-05 | 2026-02-04 |            2283 |
| David      | 2023-01-12 | 2026-02-04 |            1119 |
+------------+------------+------------+-----------------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT
    ->     sale_id,
    ->     sale_date,
    ->     DATE_ADD(sale_date, INTERVAL 30 DAY) AS expiry_date
    -> FROM Sales;
+---------+------------+-------------+
| sale_id | sale_date  | expiry_date |
+---------+------------+-------------+
|    5001 | 2026-01-25 | 2026-02-24  |
|    5002 | 2026-01-26 | 2026-02-25  |
|    5003 | 2026-01-27 | 2026-02-26  |
|    5004 | 2026-01-28 | 2026-02-27  |
|    5005 | 2026-01-29 | 2026-02-28  |
+---------+------------+-------------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT
    ->     CURDATE() AS today,
    ->     DATE_SUB(CURDATE(), INTERVAL 7 DAY) AS seven_days_ago;
+------------+----------------+
| today      | seven_days_ago |
+------------+----------------+
| 2026-02-04 | 2026-01-28     |
+------------+----------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Sales
    -> WHERE DAYOFWEEK(sale_date) IN (1, 7);
+---------+------------+---------+-------------+----------------+-------------+
| sale_id | sale_date  | vin     | customer_id | salesperson_id | final_price |
+---------+------------+---------+-------------+----------------+-------------+
|    5001 | 2026-01-25 | VIN1001 |           1 |            101 |    34500.00 |
+---------+------------+---------+-------------+----------------+-------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT
    ->     '25-12-2025' AS input_string,
    ->     STR_TO_DATE('25-12-2025', '%d-%m-%Y') AS converted_date,
    ->     MONTHNAME(STR_TO_DATE('25-12-2025', '%d-%m-%Y')) AS month_display;
+--------------+----------------+---------------+
| input_string | converted_date | month_display |
+--------------+----------------+---------------+
| 25-12-2025   | 2025-12-25     | December      |
+--------------+----------------+---------------+
1 row in set (0.00 sec)
```

Session 8:

-- 1. Improve search performance on email column
CREATE INDEX idx_customer_email ON Customers(email);

-- 2. Create UNIQUE INDEX for username
CREATE UNIQUE INDEX idx_unique_license ON Customers(driver_license);

-- 3. Find and remove an unused index
-- First, list the indexes to find the name
SHOW INDEX FROM Customers;
-- Then, remove the index (Example: removing the email index we just created)
DROP INDEX idx_customer_email ON Customers;

-- 4. Fix slow Aadhar query using an index
CREATE INDEX idx_aadhar ON Customers(driver_license); -- Applying logic to your unique ID

-- 5. Create a composite index (status, sale_date)
CREATE INDEX idx_status_date ON Sales(vin, sale_date);

```
mysql> EXPLAIN SELECT * FROM Customers WHERE driver_license = 'DL12345';
+----+-------------+-----------+------------+-------+-----------------------------+--------------------+---------+-------+------+----------+-------+
| id | select_type | table     | partitions | type  | possible_keys               | key                | key_len | ref   | rows | filtered | Extra |
+----+-------------+-----------+------------+-------+-----------------------------+--------------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | Customers | NULL       | const | idx_unique_license,idx_aadhar | idx_unique_license | 83      | const |    1 |   100.00 | NULL  |
+----+-------------+-----------+------------+-------+-----------------------------+--------------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> CREATE INDEX idx_customer_email ON Customers(email);
ERROR 1061 (42000): Duplicate key name 'idx_customer_email'
mysql> CREATE UNIQUE INDEX idx_unique_license ON Customers(driver_license);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM Customers;
+-----------+------------+--------------------+--------------+---------------+-----------+-------------+----------+--------+------+-
----------+---------+--------------+---------+------------+
| Table     | Non_unique | Key_name           | Seq_in_index | Column_name    | Collation | Cardinality | Sub_part | Packed | Null |
Index_type | Comment | Index_comment | Visible | Expression |
+-----------+------------+--------------------+--------------+---------------+-----------+-------------+----------+--------+------+-
----------+---------+--------------+---------+------------+
| customers |          0 | PRIMARY            |            1 | customer_id    | A         |           5 |     NULL |   NULL |      |
BTREE      |         |              | YES     | NULL       |
| customers |          0 | idx_unique_license |            1 | driver_license | A         |           5 |     NULL |   NULL | YES  |
BTREE      |         |              | YES     | NULL       |
| customers |          1 | idx_customer_email |            1 | email          | A         |           5 |     NULL |   NULL | YES  |
BTREE      |         |              | YES     | NULL       |
+-----------+------------+--------------------+--------------+---------------+-----------+-------------+----------+--------+------+-
----------+---------+--------------+---------+------------+
3 rows in set (0.02 sec)

mysql> DROP INDEX idx_customer_email ON Customers;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

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

mysql> CREATE INDEX idx_aadhar ON Customers(driver_license);
ERROR 1061 (42000): Duplicate key name 'idx_aadhar'
mysql>
```

Session 9:

-- 1. Create a default password (Username before @ + '123')

```
SELECT
   email,
   CONCAT(SUBSTRING_INDEX(email, '@', 1), '123') AS default_password
FROM Customers;
```

```
mysql> SELECT
    ->     email,
    ->     CONCAT(SUBSTRING_INDEX(email, '@', 1), '123') AS default_password
    -> FROM Customers;
+--------------------+------------------+
| email              | default_password |
+--------------------+------------------+
| amit.s@email.com   | amit.s123        |
| priya.v@email.com  | priya.v123       |
| rahul.n@email.com  | rahul.n123       |
| sonia.g@email.com  | sonia.g123       |
| vikram.s@email.com | vikram.s123      |
| NULL               | NULL             |
+--------------------+------------------+
6 rows in set (0.00 sec)
```

-- 2. Clean contact numbers (Remove spaces, dashes, and symbols)

```
SELECT
   phone AS original_phone,
   REGEXP_REPLACE(phone, '[^0-9]', '') AS cleaned_phone
FROM Customers;
```

```
mysql> SELECT
    ->     phone AS original_phone,
    ->     REGEXP_REPLACE(phone, '[^0-9]', '') AS cleaned_phone
    -> FROM Customers;
+----------------+---------------+
| original_phone | cleaned_phone |
+----------------+---------------+
| 9876543210     | 9876543210    |
| 8765432109     | 8765432109    |
| 7654321098     | 7654321098    |
| 9123456789     | 9123456789    |
| 8123456780     | 8123456780    |
| 9988776655     | 9988776655    |
+----------------+---------------+
6 rows in set (0.01 sec)
```

-- 3. Normalize Employee names (Mixed case to proper Upper Case)
SELECT
  CONCAT(UPPER(first_name), ' ', UPPER(last_name)) AS normalized_name
FROM Salespeople;

```
mysql> SELECT
    ->     CONCAT(UPPER(first_name), ' ', UPPER(last_name)) AS normalized_name
    -> FROM Salespeople;
+-----------------+
| normalized_name |
+-----------------+
| JOHN DOE        |
| JANE SMITH      |
| MICHAEL BROWN   |
| SARAH WILSON    |
| DAVID LEE       |
+-----------------+
5 rows in set (0.00 sec)
```

-- 4. Handle multiple fallback columns (Priority: Mobile -> Phone -> Email)
SELECT
  first_name,
  COALESCE(phone, email, 'No Contact Available') AS primary_contact
FROM Customers;

```
mysql> SELECT
    ->     first_name,
    ->     COALESCE(phone, email, 'No Contact Available') AS primary_contact
    -> FROM Customers;
+------------+-----------------+
| first_name | primary_contact |
+------------+-----------------+
| Amit       | 9876543210      |
| Priya      | 8765432109      |
| Rahul      | 7654321098      |
| Sonia      | 9123456789      |
| Vikram     | 8123456780      |
| Robert     | 9988776655      |
+------------+-----------------+
6 rows in set (0.00 sec)
```

-- 5. Find average experience per department
-- Note: Since your schema is small, we group by 'showroom_location' as the department
SELECT
    v.showroom_location AS department,
    AVG(TIMESTAMPDIFF(YEAR, s.hire_date, CURDATE())) AS avg_years_experience
FROM Salespeople s
JOIN Sales sa ON s.salesperson_id = sa.salesperson_id
JOIN Vehicles v ON sa.vin = v.vin
GROUP BY v.showroom_location;

```
mysql> SELECT
    ->     v.showroom_location AS department,
    ->     AVG(TIMESTAMPDIFF(YEAR, s.hire_date, CURDATE())) AS avg_years_experience
    -> FROM Salespeople s
    -> JOIN Sales sa ON s.salesperson_id = sa.salesperson_id
    -> JOIN Vehicles v ON sa.vin = v.vin
    -> GROUP BY v.showroom_location;
+--------------------+----------------------+
| department         | avg_years_experience |
+--------------------+----------------------+
| Main Floor - North |               6.0000 |
| Main Floor - South |               4.0000 |
| EV Wing            |               3.0000 |
| Performance Section |               6.0000 |
| Luxury Lounge      |               3.0000 |
+--------------------+----------------------+
5 rows in set (0.01 sec)
```

Session 10:

1.DELIMITER //

```
CREATE FUNCTION Get_Net_Salary(gross_salary DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
   DECLARE net_salary DECIMAL(10,2);
   -- Subtracting 10%
   SET net_salary = gross_salary * 0.90;
   RETURN net_salary;
END //

DELIMITER ;
```

```
mysql> SELECT first_name, Get_Net_Salary(30000) AS net_pay FROM Salespeople;
+------------+----------+
| first_name | net_pay  |
+------------+----------+
| John       | 27000.00 |
| Jane       | 27000.00 |
| Michael    | 27000.00 |
| Sarah      | 27000.00 |
| David      | 27000.00 |
+------------+----------+
5 rows in set (0.00 sec)
```

2.ALTER TABLE Customers ADD COLUMN last_login DATE;

```
UPDATE Customers SET last_login = '2025-12-30' WHERE customer_id = 1;
DELIMITER //
CREATE FUNCTION Check_User_Activity(last_login_date DATE)
RETURNS VARCHAR(10)
DETERMINISTIC
BEGIN
   DECLARE activity_status VARCHAR(10);
     IF DATEDIFF(CURDATE(), last_login_date) <= 30 THEN
       SET activity_status = 'Active';
   ELSE
       SET activity_status = 'Inactive';
   END IF;
   RETURN activity_status;
 END //

DELIMITER ;
```

```
mysql> DELIMITER ;
mysql> SELECT first_name, last_login, Check_User_Activity(last_login) AS status FROM Customers;
+------------+------------+----------+
| first_name | last_login | status   |
+------------+------------+----------+
| Amit       | 2025-12-30 | Inactive |
| Priya      | NULL       | Inactive |
| Rahul      | NULL       | Inactive |
| Sonia      | NULL       | Inactive |
| Vikram     | NULL       | Inactive |
| Robert     | NULL       | Inactive |
+------------+------------+----------+
6 rows in set (0.00 sec)
```

3.DELIMITER //

```
CREATE FUNCTION Calculate_Dynamic_Tax_Slab(income DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE tax_amount DECIMAL(10,2);

    SET tax_amount = CASE
        WHEN income <= 300000 THEN 0
        WHEN income BETWEEN 300001 AND 600000 THEN income * 0.10
        WHEN income BETWEEN 600001 AND 1000000 THEN income * 0.20
        ELSE income * 0.30
    END;

    RETURN tax_amount;
END //

DELIMITER ;
```

```
mysql> SELECT make, model, price, Calculate_Dynamic_Tax_Slab(price) AS tax_due FROM Vehicles;
+--------+---------+----------+---------+
| make   | model   | price    | tax_due |
+--------+---------+----------+---------+
| Toyota | Camry   | 35000.00 |    0.00 |
| Honda  | Civic   | 28000.00 |    0.00 |
| Tesla  | Model 3 | 45000.00 |    0.00 |
| Ford   | Mustang | 55000.00 |    0.00 |
| BMW    | X5      | 65000.00 |    0.00 |
+--------+---------+----------+---------+
5 rows in set (0.00 sec)
```

4.CREATE FUNCTION Get_Experience_Category(h_date DATE)

```
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE yrs INT;
    DECLARE category VARCHAR(20);
```

```
    -- Calculate difference in years
    SET yrs = TIMESTAMPDIFF(YEAR, h_date, CURDATE());

    -- Categorization logic
    SET category = CASE
        WHEN yrs < 2 THEN 'Fresher'
        WHEN yrs BETWEEN 2 AND 5 THEN 'Junior'
        WHEN yrs BETWEEN 6 AND 10 THEN 'Mid'
        ELSE 'Senior'
    END;

    RETURN category;
END //
```

```
mysql> SELECT
    ->     first_name,
    ->     hire_date,
    ->     Get_Experience_Category(hire_date) AS designation
    -> FROM Salespeople;
+------------+------------+-------------+
| first_name | hire_date  | designation |
+------------+------------+-------------+
| John       | 2020-01-15 | Mid         |
| Jane       | 2021-03-20 | Junior      |
| Michael    | 2022-06-10 | Junior      |
| Sarah      | 2019-11-05 | Mid         |
| David      | 2023-01-12 | Junior      |
+------------+------------+-------------+
5 rows in set (0.00 sec)

mysql> -- Test 1: 5 days late (5 * 50 = 250)
```

5.
```
CREATE FUNCTION Calculate_Late_Fee(due_date DATE, return_date DATE)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE days_late INT;
    DECLARE fee DECIMAL(10,2);
    -- Calculate the number of days late
    SET days_late = DATEDIFF(return_date, due_date);
    -- Logic: ₹50/day, if days_late is positive, else 0
    IF days_late > 0 THEN
```

```
    SET fee = days_late * 50;
  ELSE
    SET fee = 0;
  END IF;
   -- Limit the maximum amount to ₹1000
  IF fee > 1000 THEN
    SET fee = 1000;
  END IF;

  RETURN fee;
END //

DELIMITER ;
```

```
mysql> SELECT Calculate_Late_Fee('2025-01-01', '2025-01-06') AS fee_amount;
+------------+
| fee_amount |
+------------+
|     250.00 |
+------------+
1 row in set (0.00 sec)

mysql>
mysql> -- Test 2: 30 days late (30 * 50 = 1500, but capped at 1000)
mysql> SELECT Calculate_Late_Fee('2025-01-01', '2025-01-31') AS fee_amount;
+------------+
| fee_amount |
+------------+
|    1000.00 |
+------------+
1 row in set (0.00 sec)
```