MySQL — Week 2: Advanced Notes & Examples

Table of Contents

---

1. Subqueries

A **subquery** is a query inside another query. It is useful for breaking down complex problems.

Nested Subquery

Executed once and passed to the outer query.

```
SELECT name
FROM students
WHERE id IN (
  SELECT student_id
  FROM marks
```

```
  WHERE score > 80
);
```

Correlated Subquery

Depends on the outer query and executes for each row.

```
SELECT s.name, s.id
FROM students s
WHERE score > (
  SELECT AVG(score)
  FROM marks m
  WHERE m.student_id = s.id
);
```

---

2. UNION and UNION ALL

UNION

- Combines results of two queries.
- Removes duplicates.

```
SELECT city FROM customers
UNION
SELECT city FROM suppliers;
```

UNION ALL

- Combines results but keeps duplicates.

```
SELECT city FROM customers
UNION ALL
SELECT city FROM suppliers;
```

Key Notes:

- Both queries must have the same number of columns.
- Data types must be compatible.

---

3. Stored Procedures

A stored procedure is a reusable set of SQL statements stored in the database.

Syntax

```
DELIMITER $$
CREATE PROCEDURE procedure_name(parameter_list)
BEGIN
  -- SQL statements
END $$
DELIMITER ;
```

Example

```
DELIMITER $$
CREATE PROCEDURE get_students()
BEGIN
  SELECT * FROM students;
END $$
DELIMITER ;
```

Parameters

- **IN** → Input value.
- **OUT** → Return value.
- **INOUT** → Both input and output.

```
DELIMITER $$
CREATE PROCEDURE get_marks(IN student_id INT, OUT avg_marks FLOAT)
BEGIN
  SELECT AVG(score) INTO avg_marks
  FROM marks
  WHERE marks.student_id = student_id;
END $$
DELIMITER ;
```

---

4. Triggers

A trigger is executed automatically when an event occurs (INSERT, UPDATE, DELETE).

Syntax

```
DELIMITER $$
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name
FOR EACH ROW
BEGIN
  -- actions
END $$
DELIMITER ;
```

Example

```
DELIMITER $$
CREATE TRIGGER before_insert_students
BEFORE INSERT ON students
FOR EACH ROW
BEGIN
  SET NEW.created_at = NOW();
END $$
DELIMITER ;
```

- **NEW** → Refers to new row values.
- **OLD** → Refers to existing row values.

---

5. Functions

Functions return a single value and can be used in SQL queries.

Syntax

```
DELIMITER $$
CREATE FUNCTION function_name(parameters)
RETURNS datatype
DETERMINISTIC
BEGIN
  -- SQL statements
  RETURN value;
END $$
DELIMITER ;
```

Example

```
DELIMITER $$
CREATE FUNCTION get_total_marks(s_id INT) RETURNS INT
DETERMINISTIC
BEGIN
  DECLARE total INT;
  SELECT SUM(score) INTO total FROM marks WHERE student_id = s_id;
  RETURN total;
END $$
DELIMITER ;
```

Usage:

```
SELECT name, get_total_marks(id) AS total_score
FROM students;
```

---

6. Views

A **view** is a virtual table based on a query result.

Creating a View

```
CREATE VIEW high_scorers AS
SELECT s.name, m.score
FROM students s
JOIN marks m ON s.id = m.student_id
WHERE m.score > 80;
```

Using a View

```
SELECT * FROM high_scorers;
```

Updating Through Views

```
UPDATE high_scorers SET score = 95 WHERE name = 'John';
```

Dropping a View

```
DROP VIEW high_scorers;
```

✅ Summary

- **Subqueries**: Inner queries (Nested, Correlated).
- **UNION / UNION ALL**: Combine results, with or without duplicates.
- **Stored Procedures**: Reusable SQL blocks with parameters.
- **Triggers**: Automatic actions on table events.
- **Functions**: Return single values, usable in queries.
- **Views**: Virtual tables for simplicity and readability.