MySQL Concepts - Syntax, Theory, and Examples

1. TRIGGERS

Theory:

A trigger is a set of instructions that executes automatically before or after an event (INSERT, UPDATE, DELETE) on a table.

Syntax:

```
CREATE TRIGGER trigger_name
BEFORE | AFTER INSERT | UPDATE | DELETE
ON table_name
FOR EACH ROW
BEGIN
   -- statements
END;
```

Example:

```
CREATE TRIGGER before_insert_student
BEFORE INSERT ON Student
FOR EACH ROW
BEGIN
   SET NEW.Address = UPPER(NEW.Address);
END;
```

2. VIEWS

Theory:

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

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2 FROM table_name WHERE condition;
```

Example:

```
CREATE VIEW high_salary_instructors AS
SELECT name, salary FROM Instructor WHERE salary > 50000;
```

3. FUNCTIONS

Theory:

Functions return a single value and are used in queries.

Syntax:

```
CREATE FUNCTION function_name (param1 TYPE, ...)
RETURNS return_type
DETERMINISTIC
BEGIN
   RETURN value;
END;
```

Example:

```
CREATE FUNCTION getStatus(salary INT)
RETURNS VARCHAR(10)
DETERMINISTIC
```

```
BEGIN
  RETURN IF(salary > 50000, 'High', 'Low');
END;
```

4. PROCEDURES

Theory:

Procedures are blocks of SQL that may return multiple results using OUT or INOUT.

Syntax:

```
CREATE PROCEDURE procedure_name (IN param1 TYPE, OUT param2 TYPE)
BEGIN
  -- SQL statements
END;
```

Example:

```
CREATE PROCEDURE GetEmployeeCount (IN dept_id INT, OUT emp_count INT)
BEGIN
  SELECT COUNT(*) INTO emp_count FROM Employee WHERE dept_id = dept_id;
END;
```

5. CURSORS

Theory:

Cursors allow row-by-row processing of query results.

Syntax:

```
DECLARE cursor_name CURSOR FOR SELECT_query;
```

```
OPEN cursor_name;

FETCH cursor_name INTO variable;

CLOSE cursor_name;
```

Example:

```
CREATE PROCEDURE ShowAllNames()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE sname VARCHAR(50);
  DECLARE cur CURSOR FOR SELECT name FROM Student;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN cur;
  read_loop: LOOP
    FETCH cur INTO sname;
    IF done THEN LEAVE read_loop; END IF;
    SELECT sname;
  END LOOP;
  CLOSE cur;
END;
```

6. INDEXES

Theory:

Indexes enhance performance of SELECT queries.

Syntax:

CREATE INDEX index_name ON table_name(column_name);

Example:

CREATE INDEX idx_name ON Student(Name);

Storage:

- Indexes are stored in secondary memory (disk), optimized by the DBMS.

- InnoDB stores them in B-tree structure.

- PRIMARY KEY and UNIQUE constraints automatically create indexes.

Default Indexes:

- PRIMARY KEY: Always indexed.

- UNIQUE: Creates unique indexes.

- FOREIGN KEY: May also create indexes if not present.

Calling or Viewing Indexes:

SHOW INDEX FROM table_name;