SQL Practical Exam - Quick Revision Sheet

1. Create Table (DDL)

```
CREATE TABLE Student (
  Student_ID INT PRIMARY KEY,
  Name VARCHAR(50),
  Date_of_Birth DATE
);
```

2. Normalization Concepts

1NF: Atomic values only

2NF: No partial dependency

3NF: No transitive dependency

3. DML Operations

```
INSERT INTO Student VALUES (1, 'Sumit', '2003-05-10');
UPDATE Student SET Name = 'Kartik' WHERE Student_ID = 1;
DELETE FROM Student WHERE Student_ID = 1;
```

4. Relational Operators

```
SELECT * FROM Student WHERE Student_ID = 1;
SELECT * FROM Student WHERE Student_ID != 2;
```

5. Boolean Operators

```
SELECT * FROM Student WHERE Name = 'Sumit' OR Name = 'Kartik';
```

6. Pattern Matching

```
SELECT * FROM Student WHERE Name LIKE 'S%';

SELECT * FROM Student WHERE Name LIKE '%t';
```

7. Arithmetic & Built-in Functions

```
SELECT Student_ID + 5 FROM Student;

SELECT UPPER(Name), LENGTH(Name) FROM Student;
```

8. Add Column & Fill Age

```
ALTER TABLE Student ADD Age INT;

UPDATE Student SET Age = 21 WHERE Student_ID = 1;
```

9. Date & Time Functions

```
SELECT CURDATE();

SELECT TIMESTAMPDIFF(YEAR, Date_of_Birth, CURDATE()) AS Age FROM Student;
```

10. Set Operators

```
SELECT * FROM Student_A UNION SELECT * FROM Student_B;
```

11. Views

```
CREATE VIEW StudentView AS SELECT Student_ID, Name FROM Student;

UPDATE StudentView SET Name = 'Sagar' WHERE Student_ID = 1;
```

12. Stored Procedure

```
DELIMITER //

CREATE PROCEDURE AddStudentSimple()
```

```
BEGIN
  INSERT INTO Student VALUES (1, 'Sumit', '2003-05-10');
END //
DELIMITER ;
```

13. Function

```
DELIMITER //
CREATE FUNCTION GetAgeSimple()
RETURNS INT
BEGIN
  DECLARE age INT;
  SELECT TIMESTAMPDIFF(YEAR, Date_of_Birth, CURDATE()) INTO age
  FROM Student WHERE Student_ID = 1;
  RETURN age;
END //
DELIMITER ;
```

14. Triggers

Row-Level:

```
CREATE TRIGGER After_Student_Insert AFTER INSERT ON Student
FOR EACH ROW BEGIN
INSERT INTO Student_Log (Action_Performed) VALUES (CONCAT('Inserted student: ', NEW.Name));
END;
```

Statement-Level Simulation:

SQL Practical Exam - Quick Revision Sheet

```
CREATE TRIGGER Before_Student_Delete BEFORE DELETE ON Student

FOR EACH ROW BEGIN

IF (SELECT COUNT(*) FROM Student) = 1 THEN

INSERT INTO Delete_Log (Info) VALUES ('Last student being deleted!');

END IF;

END;
```

Viva Tips

Procedure = performs a task

Function = returns a value

Trigger = auto executes on action

Use LIKE, IN, BETWEEN, UNION