SQL Practical Exam Notes - Complete Set

1. SQL Basic Queries

```
-- Select all columns
SELECT * FROM students;

-- Select specific columns
SELECT name, age FROM students;

-- Using WHERE clause
SELECT * FROM students WHERE age > 20;

-- ORDER BY
SELECT * FROM students ORDER BY age DESC;

-- DISTINCT
SELECT DISTINCT dept_id FROM employees;

-- BETWEEN, IN, LIKE
SELECT * FROM employees WHERE salary BETWEEN 50000 AND 70000;
SELECT * FROM employees WHERE dept_id IN (1, 2);
SELECT * FROM employees WHERE emp_name LIKE 'A%';
```

2. INSERT with Constraints

```
CREATE TABLE students (
    id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT CHECK (age >= 18),
    gender VARCHAR(10) DEFAULT 'Other',
    email VARCHAR(100) UNIQUE
);

-- Basic insert
INSERT INTO students (id, name, age, gender, email)
VALUES (1, 'Alice', 20, 'Female', 'alice@example.com');
```

```
-- Using DEFAULT
INSERT INTO students (id, name, age, email)
VALUES (2, 'Bob', 22, 'bob@example.com');

-- Violating NOT NULL, PRIMARY KEY, UNIQUE, CHECK
-- (Examples included in earlier chat)
```

3. FOREIGN KEY (Create and ALTER)

```
-- Parent table
CREATE TABLE departments (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(100)
);

-- With FK in create
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);

-- Using ALTER TABLE
ALTER TABLE employees
ADD CONSTRAINT fk_dept
FOREIGN KEY (dept_id) REFERENCES departments(dept_id);
```

4. Aggregate Functions with GROUP BY, HAVING, WHERE

```
-- COUNT, SUM, AVG, MIN, MAX
SELECT COUNT(*) FROM employees;
SELECT SUM(salary) FROM employees;
SELECT AVG(salary) FROM employees;
SELECT MIN(salary) FROM employees;
SELECT MAX(salary) FROM employees;
```

```
-- GROUP BY + HAVING + WHERE
SELECT dept_id, AVG(salary) FROM employees WHERE salary > 40000 GROUP BY dept_id
HAVING AVG(salary) > 50000;

-- Complex examples (based on 'sales' table) also included above.
```

5. SQL JOINS (All Types)

```
-- INNER JOIN
SELECT emp_name, dept_name FROM employees
INNER JOIN departments ON employees.dept_id = departments.dept_id;

-- LEFT JOIN
SELECT emp_name, dept_name FROM employees
LEFT JOIN departments ON employees.dept_id = departments.dept_id;

-- RIGHT JOIN
SELECT emp_name, dept_name FROM employees
RIGHT JOIN departments ON employees.dept_id = departments.dept_id;

-- FULL OUTER JOIN (if supported)
SELECT emp_name, dept_name FROM employees
FULL OUTER JOIN departments ON employees.dept_id = departments.dept_id;
```

6. JOIN + GROUP BY + Aggregation Practice Queries

```
-- Total sales per customer
SELECT customer_name, SUM(amount) AS total_sales
FROM sales GROUP BY customer_name;

-- Customers with total sales > 5000
SELECT customer_name, SUM(amount) AS total_sales
FROM sales GROUP BY customer_name HAVING SUM(amount) > 5000;

-- Avg order in February per region
SELECT region, AVG(amount)
FROM sales WHERE MONTH(order_date) = 2 GROUP BY region;
```

7. UPDATE and DELETE

```
-- UPDATE
UPDATE students SET age = 21 WHERE id = 1;

-- DELETE
DELETE FROM students WHERE age < 18;
```

8. ALIAS, ORDER BY, LIMIT

```
-- ALIAS
SELECT name AS student_name FROM students;

-- ORDER BY and LIMIT
SELECT * FROM employees ORDER BY salary DESC LIMIT 5;
```

9. SUBQUERIES

```
-- Scalar subquery
SELECT name FROM students WHERE age = (SELECT MAX(age) FROM students);

-- IN subquery
SELECT name FROM employees WHERE dept_id IN (SELECT dept_id FROM departments
WHERE dept_name = 'IT');
```

10. SET Operations (UNION, INTERSECT, EXCEPT)

```
-- UNION
SELECT name FROM students
UNION
SELECT emp_name FROM employees;

-- INTERSECT (Not supported in MySQL)
-- EXCEPT (Not supported in MySQL)
```

11. CREATE, DROP, ALTER

```
-- CREATE TABLE
CREATE TABLE test (id INT, name VARCHAR(100));

-- ALTER TABLE
ALTER TABLE test ADD email VARCHAR(100);

-- DROP TABLE
DROP TABLE test;
```

12. CONSTRAINTS (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT)

```
-- Already shown in earlier sections, here's a summary:

CREATE TABLE example (
    id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    age INT CHECK (age >= 18),
    gender VARCHAR(10) DEFAULT 'Other'
);
```

13. VIEWS

```
-- Creating a view
CREATE VIEW view_sales AS
SELECT customer_name, SUM(amount) AS total_sales
FROM sales
GROUP BY customer_name;

-- Using the view
SELECT * FROM view_sales;

-- Dropping a view
DROP VIEW view_sales;
```

14. INDEXES

```
-- Creating index
CREATE INDEX idx_name ON students(name);

-- Dropping index
DROP INDEX idx_name ON students;
```

15. CASE and IF

```
-- CASE
SELECT name,
       CASE
           WHEN age >= 18 THEN 'Adult'
           ELSE 'Minor'
       END AS age_group
FROM students;

-- IF (MySQL specific)
SELECT name, IF(age >= 18, 'Adult', 'Minor') AS age_group
FROM students;
```