

SQL Practical Exam Notes

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;
```