■ SQL Notes (Well-Formatted)

Section 01: Introduction to SQL

SQL = Structured Query Language to manage data stored in a relational DBMS.

Categories of SQL:

- **DDL** (Data Definition Language): CREATE, ALTER, DROP.
- **DML** (Data Manipulation Language): SELECT, INSERT, UPDATE, DELETE.
- **DCL** (Data Control Language): GRANT, REVOKE.

SQL was one of the first commercial database languages since the 1970s.

Section 02: Querying the Data

Basic SELECT syntax:

```
SELECT column1, column2 FROM table;
```

Filtering with WHERE, DISTINCT, LIMIT, FETCH.

Comparison operators: =, <>, >, <, >=, <=.

Logical operators: AND, OR, NOT, IN, BETWEEN, LIKE, IS NULL.

Section 03: Sorting Data

ORDER BY clause sorts the result set.

Default order: ASC (ascending).

Example:

```
SELECT first_name, salary FROM employees ORDER BY salary DESC;
```

Section 04: Conditional Expressions

CASE allows conditional logic in queries.

Example:

```
SELECT first_name, salary,
        CASE
            WHEN salary < 30000 THEN 'Low'
            WHEN salary BETWEEN 30000 AND 50000 THEN 'Average'
            ELSE 'High'
        END AS category
        FROM employees;
```

Section 05: Joins

INNER JOIN → Returns only matching rows.

LEFT JOIN → All rows from left table + matching right.

RIGHT JOIN → All rows from right table + matching left.

FULL OUTER JOIN → All rows from both tables.

CROSS JOIN → Cartesian product.

SELF JOIN → Table joined with itself.

Example:

```
SELECT e.first_name, d.department_name
        FROM employees e
        INNER JOIN departments d
        ON e.department_id = d.department_id;
```

Section 06: Aggregations & Grouping

Aggregate Functions: AVG, COUNT, SUM, MAX, MIN.

GROUP BY groups rows by column values.

HAVING filters groups after aggregation.

Example:

```
SELECT department_id, COUNT(*) AS total_employees
        FROM employees
        GROUP BY department_id
        HAVING COUNT(*) > 5;
```

Section 07: Set Operators

UNION → Combine result sets, removes duplicates.

UNION ALL → Combine result sets, keeps duplicates.

INTERSECT → Returns rows common to both queries.

MINUS / EXCEPT → Returns rows from first query not in second.

Example:

```
SELECT employee_id FROM employees
        INTERSECT
        SELECT manager_id FROM employees;
```

Section 08: Subqueries

Subqueries allow nesting queries inside another.

Correlated Subquery → Uses values from outer query.

Operators: EXISTS, ALL, ANY.

Example:

```
SELECT first_name, salary
        FROM employees
        WHERE salary > (SELECT AVG(salary) FROM employees);
```

Section 09: Modifying Data (DML)

```
INSERT → Add new rows.
UPDATE → Modify existing rows.
DELETE → Remove rows.
```

Examples:

```
INSERT INTO departments VALUES (10, 'HR');
UPDATE employees SET salary = salary * 1.1 WHERE department_id = 10;
DELETE FROM employees WHERE employee_id = 100;
```

Section 10: Working with Tables (DDL)

```
CREATE TABLE → Define new table.
```

ALTER TABLE → Modify table structure.

DROP TABLE → Remove a table.

TRUNCATE TABLE → Remove all rows quickly.

Example:

```
CREATE TABLE employees (
            id INT PRIMARY KEY,
            name VARCHAR(50),
            salary DECIMAL(10,2)
        );
```

Section 11: Constraints

PRIMARY KEY → Ensures uniqueness + not null.

FOREIGN KEY → References another table.

UNIQUE → Column must have unique values.

NOT NULL → Column cannot be null.

CHECK → Restrict values.

Example:

```
CREATE TABLE students (
            id INT PRIMARY KEY,
            age INT CHECK (age > 18)
        );
```

Section 12: Practice Queries

Find highest paid employee:

```
SELECT first_name, MAX(salary) FROM employees;
```

Find departments without employees:

```
SELECT d.department_name
        FROM departments d
        LEFT JOIN employees e
        ON d.department_id = e.department_id
        WHERE e.employee_id IS NULL;
```

Find employees earning more than their department average:

```
SELECT e.first_name, e.salary
        FROM employees e
        WHERE e.salary > (
            SELECT AVG(salary)
            FROM employees
            WHERE department_id = e.department_id
        );
```