SQL Complete Summary Sheet (All in One)

Author: *Loganathan Sengottaiyan*

□ 1️⃣ What is SQL?

SQL (Structured Query Language) is used to **store, manipulate, and retrieve** data in a **relational database** (RDBMS).

⚙ 2️⃣ Types of SQL Commands

Type	Description	Examples
DDL	Define structure	`CREATE`, `ALTER`, `DROP`, `TRUNCATE`
DML	Manage data	`INSERT`, `UPDATE`, `DELETE`
DQL	Query data	`SELECT`
DCL	Access control	`GRANT`, `REVOKE`
TCL	Manage transactions	`COMMIT`, `ROLLBACK`, `SAVEPOINT`

🔑 3️⃣ Keys in SQL

Key	Description
Primary Key	Unique, Not Null
Foreign Key	References another table
Unique Key	Unique but can be Null
Composite Key	Combination of multiple columns
Candidate Key	Possible Primary Keys

□ 4️⃣ Constraints

Constraint	Description
NOT NULL	Field cannot be null
UNIQUE	No duplicate values
PRIMARY KEY	Unique + Not null
FOREIGN KEY	Links tables
CHECK	Restrict value range
DEFAULT	Default value if not given

□ 5️⃣ Clauses

Clause	Use
`WHERE`	Filter rows
`ORDER BY`	Sort results
`GROUP BY`	Group similar rows
`HAVING`	Filter groups
`LIMIT/TOP/FETCH`	Restrict number of rows

🔄 6️⃣ JOINS

Join Type	Description
INNER JOIN	Matching records from both tables
LEFT JOIN	All left + matching right
RIGHT JOIN	All right + matching left
FULL JOIN	All from both
SELF JOIN	Table joins itself

📃 Example:

```
SELECT e.EmpName, d.DeptName
FROM Employee e
INNER JOIN Department d
ON e.DeptID = d.DeptID;
```

🔍 7️⃣ ORDER BY / Sorting

Keyword	Use	Example
`ASC`	Ascending	`ORDER BY Salary ASC`
`DESC`	Descending	`ORDER BY Salary DESC`

💰 8️⃣ Highest & 2nd Highest Salary

MySQL / PostgreSQL

```
-- Highest
SELECT * FROM Employee ORDER BY Salary DESC LIMIT 1;

-- 2nd Highest
SELECT * FROM Employee ORDER BY Salary DESC LIMIT 1 OFFSET 1;
```

SQL Server

```
SELECT TOP 1 * FROM Employee ORDER BY Salary DESC; -- Highest
SELECT TOP 1 * FROM Employee WHERE Salary < (SELECT MAX(Salary) FROM
Employee);
```

Oracle

```
SELECT * FROM Employee ORDER BY Salary DESC OFFSET 1 ROW FETCH NEXT 1 ROW
ONLY;
```

□ 9 Functions

◆ Aggregate:

`COUNT(), SUM(), AVG(), MIN(), MAX()`

◆ String:

`LENGTH(), UPPER(), LOWER(), SUBSTRING(), CONCAT()`

◆ Date:

`NOW(), CURDATE(), SYSDATE()`

□ 10 Subqueries

Query inside another query.

📄 Example:

```
SELECT EmpName
FROM Employee
WHERE Salary > (SELECT AVG(Salary) FROM Employee);
```

□ 11 Set Operators

Operator	Description
`UNION`	Combine unique rows
`UNION ALL`	Combine all rows (duplicates allowed)
`INTERSECT`	Common rows
`MINUS`	Difference

📊 12 GROUP BY & HAVING

```
SELECT DeptID, COUNT(*) AS EmpCount
FROM Employee
GROUP BY DeptID
HAVING COUNT(*) > 5;
```

🔢 13 Ranking Functions

Function	Description
`RANK()`	Skips duplicate ranks
`DENSE_RANK()`	No skip in rank
`ROW_NUMBER()`	Unique sequence number

📄 Example:

```
SELECT EmpName, Salary,
RANK() OVER (ORDER BY Salary DESC) AS RankPos
FROM Employee;
```

☐ 14 TRUNCATE vs DELETE vs DROP

Command	Type	Deletes Data	Deletes Structure	Rollback	WHERE	Speed
DELETE	DML	✔	✖	✔	✔	Slow
TRUNCATE	DDL	✔	✖	✖	✖	Fast
DROP	DDL	✔	✔	✖	✖	Fastest

📄 Examples:

```
DELETE FROM Employee WHERE DeptID = 10;
TRUNCATE TABLE Employee;
DROP TABLE Employee;
```

⚙ 15 Transactions

Command	Purpose
`COMMIT`	Save changes
`ROLLBACK`	Undo changes
`SAVEPOINT`	Create partial rollback point

☐ 16 Indexes

Used to speed up searches on columns.

```
CREATE INDEX idx_salary ON Employee(Salary);
```

💡 17 Views

Virtual table based on query.

```
CREATE VIEW HighSalaryEmp AS
SELECT EmpName, Salary FROM Employee WHERE Salary > 90000;
```

---

18 Performance Optimization Tips

✓ Use `EXISTS` instead of `IN` for large sets
✓ Avoid `SELECT *`
✓ Use `JOIN` instead of subquery where possible
✓ Apply `WHERE` filters before grouping
✓ Index columns used in conditions

🔎 19 Real-Time Queries

Scenario	SQL Query
Employees above average salary	`SELECT * FROM Employee WHERE Salary > (SELECT AVG(Salary) FROM Employee);`
Duplicate Names	`SELECT EmpName, COUNT(*) FROM Employee GROUP BY EmpName HAVING COUNT(*)>1;`
Employee without Dept	`SELECT * FROM Employee WHERE DeptID IS NULL;`
Top 3 Highest Salaries	`SELECT * FROM Employee ORDER BY Salary DESC LIMIT 3;`

---

20 LIMIT / TOP / OFFSET — Comparison

Database	Syntax	Example	Purpose
MySQL	`LIMIT [N] OFFSET [M]`	`LIMIT 1 OFFSET 1`	Skip 1, show next
SQL Server	`TOP [N]`	`TOP 2`	Fetch top rows
Oracle	`OFFSET / FETCH`	`OFFSET 1 ROW FETCH NEXT 1 ROW ONLY`	Pagination

✅ 21 Most Asked Interview Queries

Query	Purpose
`SELECT MAX(Salary)`	Highest salary
`SELECT MAX(Salary) WHERE Salary < (MAX(Salary))`	2nd highest
`SELECT * ORDER BY Salary DESC LIMIT 3`	Top 3
`SELECT COUNT(DISTINCT DeptID)`	Unique departments
`SELECT DeptID, SUM(Salary)`	Department-wise total

🎯 Summary Taglines

- **DDL** → Structure
- **DML** → Data Manipulation
- **DQL** → Data Query
- **DCL** → Access
- **TCL** → Transaction
- **RANK()** → Skips duplicates
- **DENSE_RANK()** → Continuous
- **ROW_NUMBER()** → Sequential
- **DELETE vs TRUNCATE vs DROP** → Remove data/table differently
- **LIMIT / OFFSET / TOP** → For pagination & Nth records

SQL ORDER BY, LIMIT, TOP, OFFSET — Complete with Examples

☐ Sample Table — `Employee`

```
CREATE TABLE Employee (
  EmpID INT PRIMARY KEY,
  EmpName VARCHAR(50),
  Salary DECIMAL(10,2)
);

INSERT INTO Employee VALUES
(101, 'Logan', 80000),
(102, 'Raj', 95000),
(103, 'Alice', 70000),
(104, 'John', 95000),
(105, 'Ravi', 85000);
```

EmpID	EmpName	Salary
101	Logan	80000

EmpID	EmpName	Salary
102	Raj	95000
103	Alice	70000
104	John	95000
105	Ravi	85000

□ 1 ORDER BY — ASC & DESC

◆ Ascending (Lowest → Highest)

```
SELECT * FROM Employee ORDER BY Salary ASC;
```

Output:

EmpName	Salary
Alice	70000
Logan	80000
Ravi	85000
Raj	95000
John	95000

◆ Descending (Highest → Lowest)

```
SELECT * FROM Employee ORDER BY Salary DESC;
```

Output:

EmpName	Salary
Raj	95000
John	95000
Ravi	85000
Logan	80000
Alice	70000

□ 2 Highest Salary

□ Using ORDER BY (MySQL / SQL Server / Oracle)

```
SELECT * FROM Employee
ORDER BY Salary DESC
LIMIT 1;    -- MySQL / PostgreSQL
```

📃 or

```
SELECT TOP 1 * FROM Employee
ORDER BY Salary DESC;   -- SQL Server
```

📃 or

```
SELECT * FROM Employee
ORDER BY Salary DESC
FETCH FIRST 1 ROWS ONLY;  -- Oracle
```

✅ Output:

EmpName	Salary
Raj	95000

💎 3️⃣ Second Highest Salary

◆ (a) MySQL / PostgreSQL — Using `LIMIT` + `OFFSET`

```
SELECT * FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET 1;
```

➡ Explanation:

- `LIMIT 1` → fetch 1 record
- `OFFSET 1` → skip the first (highest) record

✅ Output:

EmpName	Salary
Ravi	85000

◆ (b) SQL Server — Using `TOP`

```
SELECT TOP 1 * FROM Employee
WHERE Salary < (SELECT MAX(Salary) FROM Employee)
ORDER BY Salary DESC;
```

✅ Output:

EmpName	Salary
Ravi	85000

◆ (c) Oracle / ANSI — Using FETCH & OFFSET

```
SELECT * FROM Employee
ORDER BY Salary DESC
OFFSET 1 ROW
FETCH NEXT 1 ROW ONLY;
```

✓ Output:

EmpName	Salary
Ravi	85000

□ 4 Nth Highest Salary (General Formula)

For MySQL/PostgreSQL:

```
SELECT * FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET N-1;
```

➡ Example: 3rd highest salary → `OFFSET 2`

🔢 5 DISTINCT Salary — Avoid Duplicates

If multiple employees share the same salary:

```
SELECT DISTINCT Salary
FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET 1;
```

✓ Returns the **2nd unique salary** (not the 2nd row).

□ 6 Using Subquery — Works in All Databases

```
SELECT MAX(Salary)
FROM Employee
WHERE Salary < (SELECT MAX(Salary) FROM Employee);
```

✓ Output → 85000

⚙ 7 TOP vs LIMIT vs OFFSET — Difference Table

Database	Syntax	Keyword	Example	Description
MySQL	LIMIT [N] OFFSET [M]	`LIMIT`	`LIMIT 1 OFFSET 1`	Skip 1st record and show next
PostgreSQL	LIMIT / OFFSET	`LIMIT`	Same as MySQL	Pagination
SQL Server	TOP [N]	`TOP`	`SELECT TOP 2 *`	Get top N rows
Oracle (12c+)	OFFSET / FETCH NEXT	`FETCH`	`OFFSET 1 ROW FETCH NEXT 1 ROW ONLY`	Pagination (modern syntax)

💡 8 Real-Life Use Cases

Top 3 Highest Salaries

```
SELECT * FROM Employee
ORDER BY Salary DESC
LIMIT 3;
```

Lowest 2 Salaries

```
SELECT * FROM Employee
ORDER BY Salary ASC
LIMIT 2;
```

5th Highest Salary (Generic)

```
SELECT * FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET 4;
```

9 Bonus — Using RANK() and DENSE_RANK()

```
SELECT EmpName, Salary,
       RANK() OVER (ORDER BY Salary DESC) AS RankPosition
FROM Employee
WHERE RankPosition = 2;
```

✓ Gives **2nd highest salary (skipping duplicates)**.
For continuous ranking use `DENSE_RANK()`.

✓ Summary Table

Query	Method	Database	Returns
Highest Salary	`ORDER BY DESC LIMIT 1`	MySQL	1st highest
2nd Highest Salary	`LIMIT 1 OFFSET 1`	MySQL	2nd highest
3rd Highest Salary	`LIMIT 1 OFFSET 2`	MySQL	3rd highest
Highest Salary	`TOP 1`	SQL Server	1st
2nd Highest	`TOP 1 WHERE Salary < MAX()`	SQL Server	2nd
2nd Highest	`OFFSET 1 FETCH NEXT 1 ROW`	Oracle	2nd