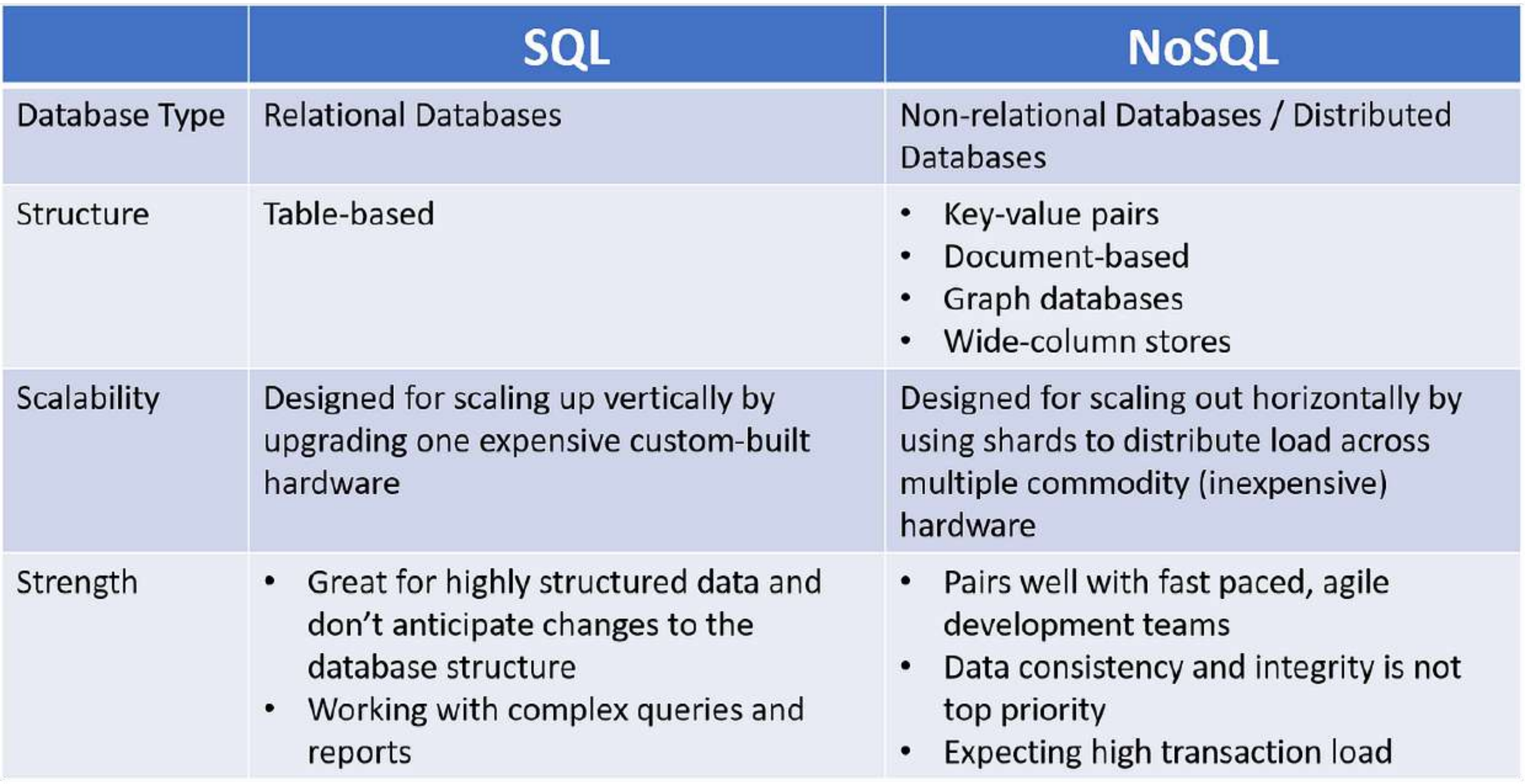
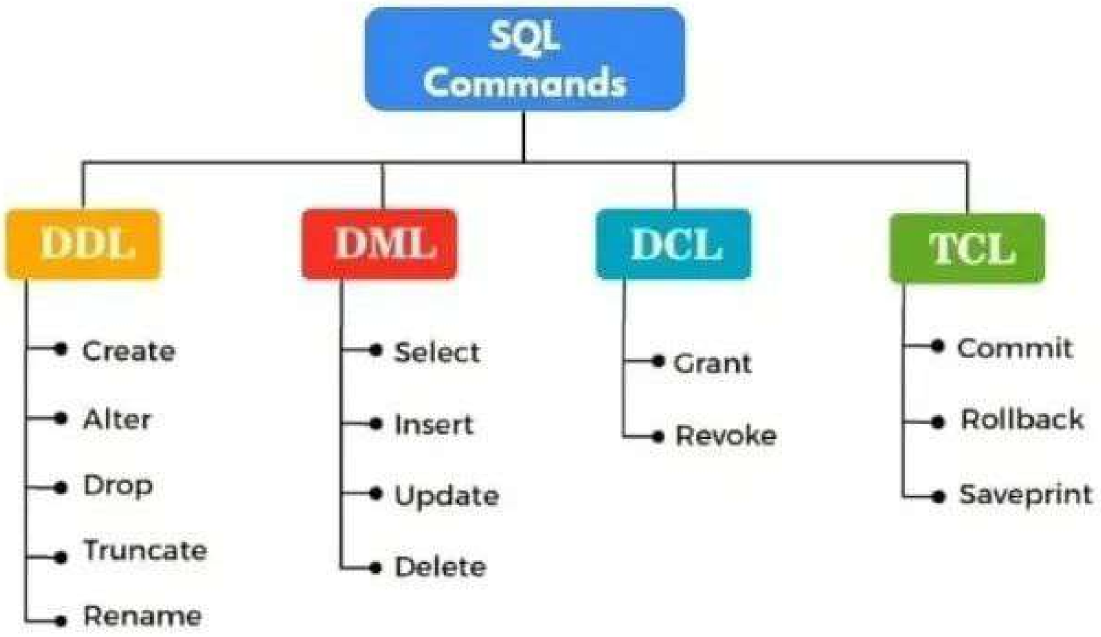


	SQL	NoSQL
Database Type	Relational Databases	Non-relational Databases / Distributed Databases
Structure	Table-based	• Key-value pairs • Document-based • Graph databases • Wide-column stores
Scalability	Designed for scaling up vertically by upgrading one expensive custom-built hardware	Designed for scaling out horizontally by using shards to distribute load across multiple commodity (inexpensive) hardware
Strength	• Great for highly structured data and don't anticipate changes to the database structure • Working with complex queries and reports	• Pairs well with fast paced, agile development teams • Data consistency and integrity is not top priority • Expecting high transaction load

What is SQL?

- sql is stand for structured query language.
- This database language is mainly designed for maintaining the data in relational database management systems.
- sql is standard language for accessing and manipulating database.

Types of SQL Commands:



1. What is SQL?

Answer: SQL stands for Structured Query Language, and it is a domain-specific language used for managing and manipulating relational databases.

2. What is a database?

Answer: A database is a structured collection of data that is organized and stored for efficient retrieval and manipulation.

3. Explain the difference between SQL and MySQL.

Answer: SQL is a language used to manage and manipulate relational databases, while MySQL is an open-source relational database management system (RDBMS) that uses SQL as its query language.

4. What is a primary key?

Answer: A primary key is a unique identifier for a record in a database table. It ensures that each record can be uniquely identified and helps maintain data integrity.

5. What is a foreign key?

Answer: A foreign key is a field in a database table that is used to establish a link between the data in two tables. It creates a relationship between the tables by referencing the primary key of another table.

6. Explain the types of SQL commands.

Answer: SQL commands can be broadly categorized into Data Definition Language (DDL), Data Manipulation Language (DML), Data Query Language (DQL), and Data Control Language (DCL).

7. What is the difference between CHAR and VARCHAR data types?

Answer: CHAR is a fixed-length character data type, while VARCHAR is a variable-length character data type. VARCHAR is more flexible as it only stores the characters entered, whereas CHAR pads the remaining spaces with blanks.

8. What is normalization?

Answer: Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller tables and defining relationships between them.

9. Explain the difference between INNER JOIN and LEFT JOIN.

Answer: INNER JOIN returns only the rows where there is a match in both tables, while LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.

10. What is the purpose of the GROUP BY clause?

Answer: The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows, like finding the total or average for each group.

11. Explain the difference between UNION and UNION ALL.

Answer: UNION combines the result sets of two queries, eliminating duplicate rows, while UNION ALL combines the result sets including duplicates.

12. What is an index, and why is it used?

Answer: An index is a data structure that improves the speed of data retrieval operations on a database table. It is used to quickly locate and access the rows in a table based on the values in one or more columns.

13. What is a stored procedure?

Answer: A stored procedure is a precompiled collection of one or more SQL statements that can be executed as a single unit. It is stored in the database and can be called by name.

14. Explain the ACID properties of a transaction.

Answer: ACID stands for Atomicity, Consistency, Isolation, and Durability. It is a set of properties that guarantee that database transactions are processed reliably.

15. What is the purpose of the HAVING clause?

Answer: The HAVING clause is used in combination with the GROUP BY clause to filter the results of a group based on a specified condition.

16. What is a view in SQL?

Answer: A view is a virtual table derived from one or more tables. It does not store the data itself but provides a way to represent the result of a stored query.

17. Explain the difference between a clustered and a non-clustered index.

Answer: A clustered index determines the physical order of data in a table, whereas a non-clustered index does not affect the physical order and creates a separate structure for faster data retrieval.

18. What is the purpose of the SQL ORDER BY clause?

Answer: The ORDER BY clause is used to sort the result set of a query in ascending or descending order based on one or more columns.

19. What is a trigger?

Answer: A trigger is a set of instructions that are automatically executed (or "triggered") in response to certain events on a particular table or view in a database.

20. Explain the difference between a candidate key, primary key, and a super key.

Answer: A super key is any set of columns that uniquely identifies a row, a candidate key is a minimal super key, and the primary key is the chosen candidate key for a table.

21. What is the purpose of the SQL LIKE statement?

Answer: The LIKE statement is used in a WHERE clause to search for a specified pattern in a column.

22. What is the difference between a view and a table?

Answer: A table is a physical storage structure that stores data, while a view is a virtual table derived from one or more tables, presenting a way to represent the result of a stored query.

23. What is a self-join?

Answer: A self-join is a regular join, but the table is joined with itself. It is often used when a table has a foreign key that references its own primary key.

24. Explain the purpose of the SQL IN operator.

Answer: The IN operator is used in a WHERE clause to filter the result set based on a specified list of values.

25. What is a subquery?

Answer: A subquery is a query nested inside another query. It can be used to retrieve data that will be used in the main query as a condition to further restrict the data to be retrieved.

26. What is the purpose of the SQL GROUP BY and COUNT() functions?

Answer: The GROUP BY clause is used to group rows based on specified columns, and COUNT() is an aggregate function that counts the number of rows in each group.

27. Explain the difference between a DELETE and TRUNCATE statement.

Answer: DELETE is used to remove rows from a table based on a condition, while TRUNCATE removes all rows from a table and is faster but cannot be rolled back.

28. What is a cross join?

Answer: A cross join, or Cartesian join, returns the Cartesian product of the sets of rows from the joined tables. It results in every combination of rows from both tables.

29. What is the purpose of the SQL BETWEEN operator?

Answer: The BETWEEN operator is used in a WHERE clause to filter the result set based on a range of values.

30. Explain the purpose of the SQL UPDATE statement.

Answer: The UPDATE statement is used to modify the existing records in a table. It allows you to update specific columns with new values.

31. What is a composite key?

Answer: A composite key is a combination of two or more columns in a table that uniquely identifies a record. Each column in the composite key contributes to the uniqueness of the key.

32. What is the difference between a view and a materialized view?

Answer: A view is a virtual table derived from one or more tables, while a materialized view is a physical copy of the result set of a query stored for faster data retrieval.

33. Explain the purpose of the SQL MAX() function.

Answer: The MAX() function is an aggregate function that returns the maximum value in a set of values.

34. What is the purpose of the SQL DISTINCT keyword?

Answer: The DISTINCT keyword is used in a SELECT statement to eliminate duplicate records from the result set.

35. Explain the purpose of the SQL JOIN statement.

Answer: The JOIN statement is used to combine rows from two or more tables based on a related column between them.

36. What is a cursor in SQL?

Answer: A cursor is a database object used to traverse the result set of a query. It allows you to process each row individually.

37. Explain the difference between a left outer join and a right outer join.

Answer: A left outer join returns all rows from the left table and the matched rows from the right table, while a right outer join returns all rows from the right table and the matched rows from the left table.

38. What is the purpose of the SQL AVG() function?

Answer: The AVG() function is an aggregate function that returns the average value of a numeric column.

39. Explain the purpose of the SQL CASE statement.

Answer: The CASE statement is used to perform conditional logic in SQL queries, similar to the "if-else" statements in programming languages.

40. What is the difference between a database and a schema?

Answer: A database is a collection of tables, while a schema is a collection of database objects, including tables, views, and stored procedures.

41. What is the purpose of the SQL COUNT() function?

Answer: The COUNT() function is an aggregate function that returns the number of rows in a result set or the number of non-null values in a column.

42. Explain the difference between the SQL WHERE and HAVING clauses.

Answer: The WHERE clause is used to filter rows before grouping in a query, while the HAVING clause is used to filter groups after they have been formed.

43. What is the purpose of the SQL ROLLBACK statement?

Answer: The ROLLBACK statement is used to undo transactions that have not been saved to the database. It is typically used in error-handling scenarios.

44. Explain the difference between UNION and JOIN.

Answer: UNION combines the result sets of two queries, while JOIN combines rows from two or more tables based on a related column.

45. What is the purpose of the SQL TRIGGER statement?

Answer: The TRIGGER statement is used to specify a set of actions that are automatically performed when a certain event occurs in a particular table or view.

46. Explain the purpose of the SQL SUM() function.

Answer: The SUM() function is an aggregate function that returns the sum of all values in a numeric column.

47. What is a natural join?

Answer: A natural join is a type of join that combines tables based on columns with the same name and automatically eliminates one of the duplicate columns.

48. What is the purpose of the SQL NULL value?

Answer: The NULL value in a database represents the absence of a value in a column. It is not the same as an empty string or zero.

49. Explain the difference between a unique key and a primary key.

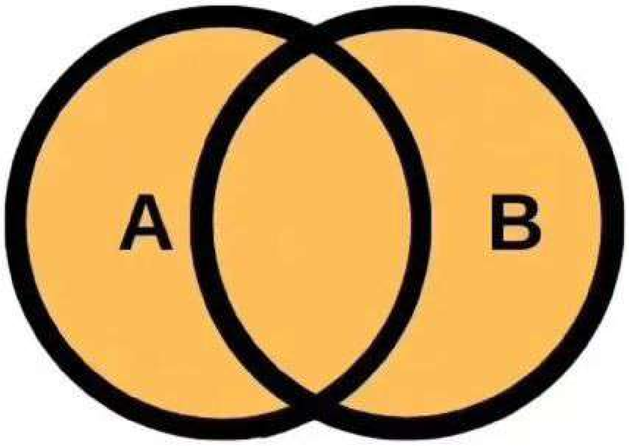
Answer: A unique key ensures that all values in a column are unique, while a primary key is a unique key that also serves as the main identifier for a record in a table.

50. What is the purpose of the SQL INSERT statement?

Answer: The INSERT statement is used to insert new records into a table. It allows you to specify the values for each column or use a subquery to select the values from another table.

- **Full Outer Join**

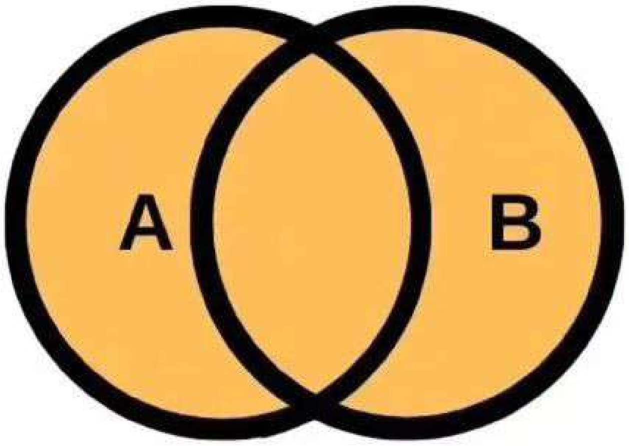
This type of join returns all rows from both tables, and any matching rows. If there is no match



```
SELECT *
FROM TableA
FULL OUTER JOIN TableB ON
TableA.ID = TableB.ID;
```

- **Full Outer Join**

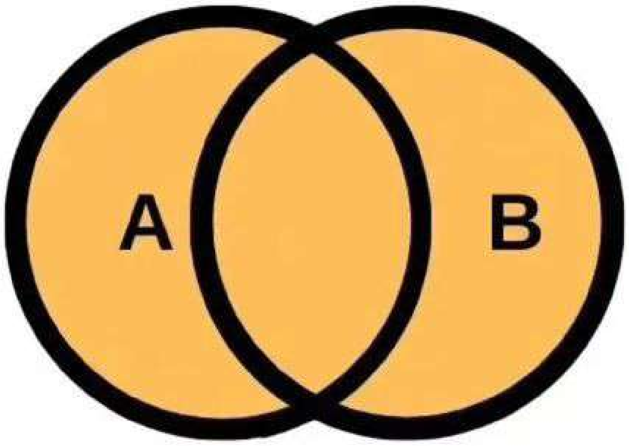
This type of join returns all rows from both tables, and any matching rows. If there is no match



```sql
SELECT *
FROM TableA
FULL OUTER JOIN TableB ON
TableA.ID = TableB.ID;
```

- **Full Outer Join**

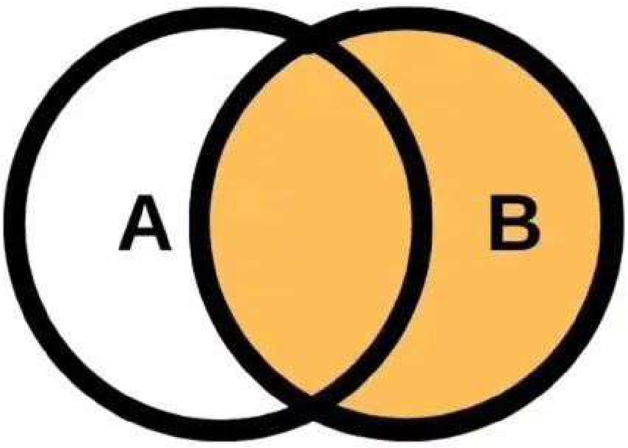
This type of join returns all rows from both tables, and any matching rows. If there is no match



```
SELECT *
FROM TableA
FULL OUTER JOIN TableB ON
TableA.ID = TableB.ID;
```

• Right Join

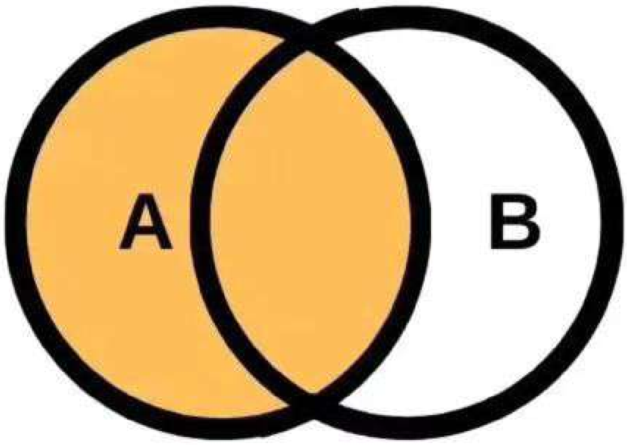
This type of join returns all rows from the right table (table B), and any matching rows from the left table (table A). If there is no match, NULL values will be returned for left table's columns



```
SELECT *
FROM TableA
RIGHT JOIN TableB ON
TableA.ID = TableB.ID;
```

- **Left join**

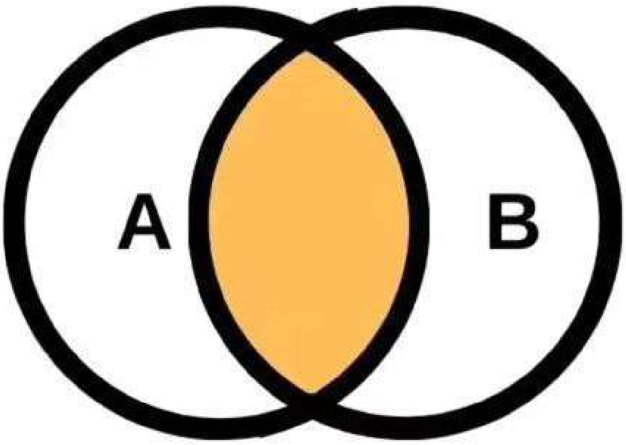
This type of join returns all rows from the left table (table A), and any matching rows from the right table (table B). If there is no match, NULL values will be returned for right table's columns.



```
SELECT *
FROM TableA
LEFT JOIN TableB ON
TableA.ID = TableB.ID;
```

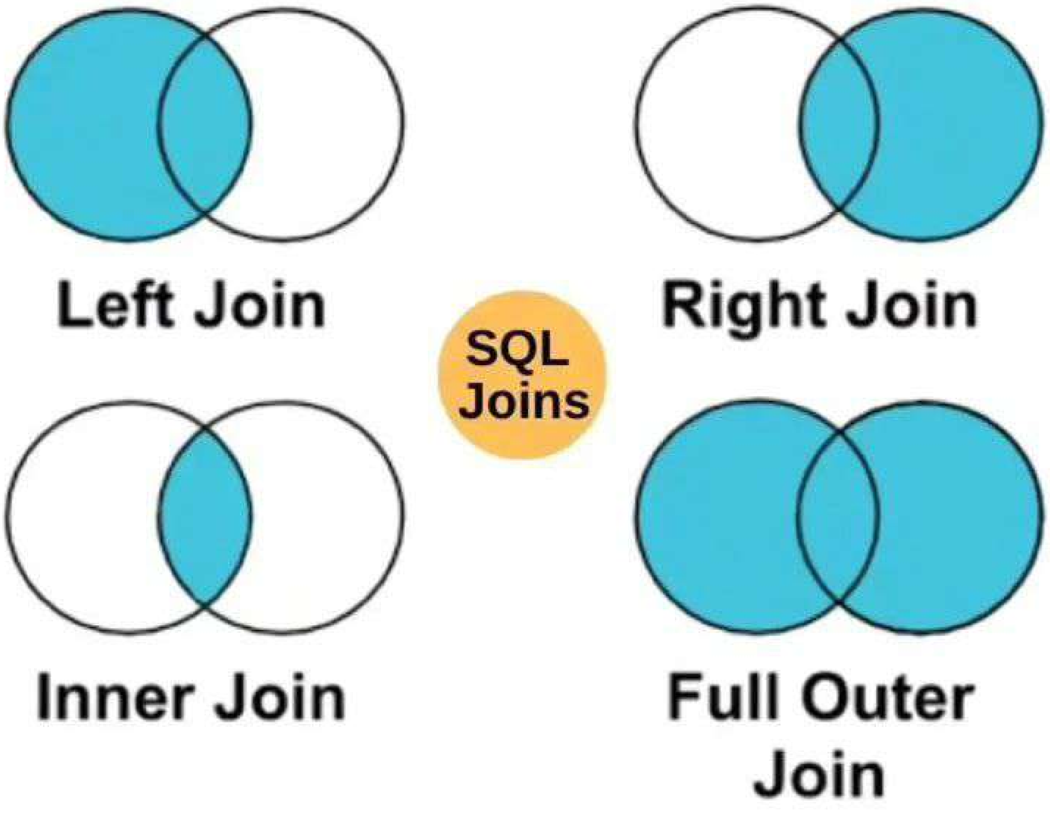
• Inner Join

This type of join returns only the rows that have matching values in both tables.

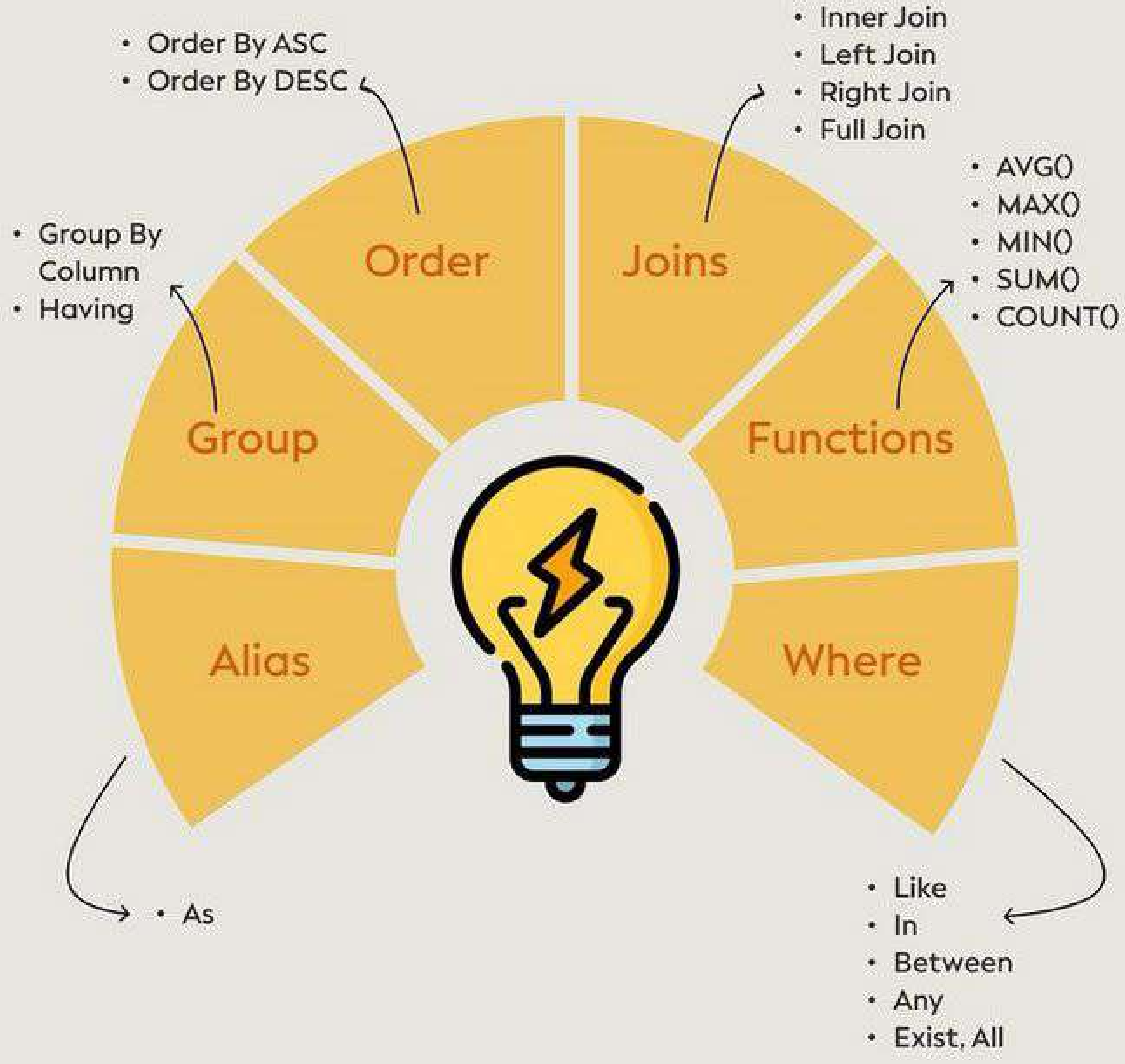


```
SELECT *
FROM TableA
JOIN TableB ON TableA.ID =
TableB.ID;
```

Joins in SQL



SQL CIRCLE

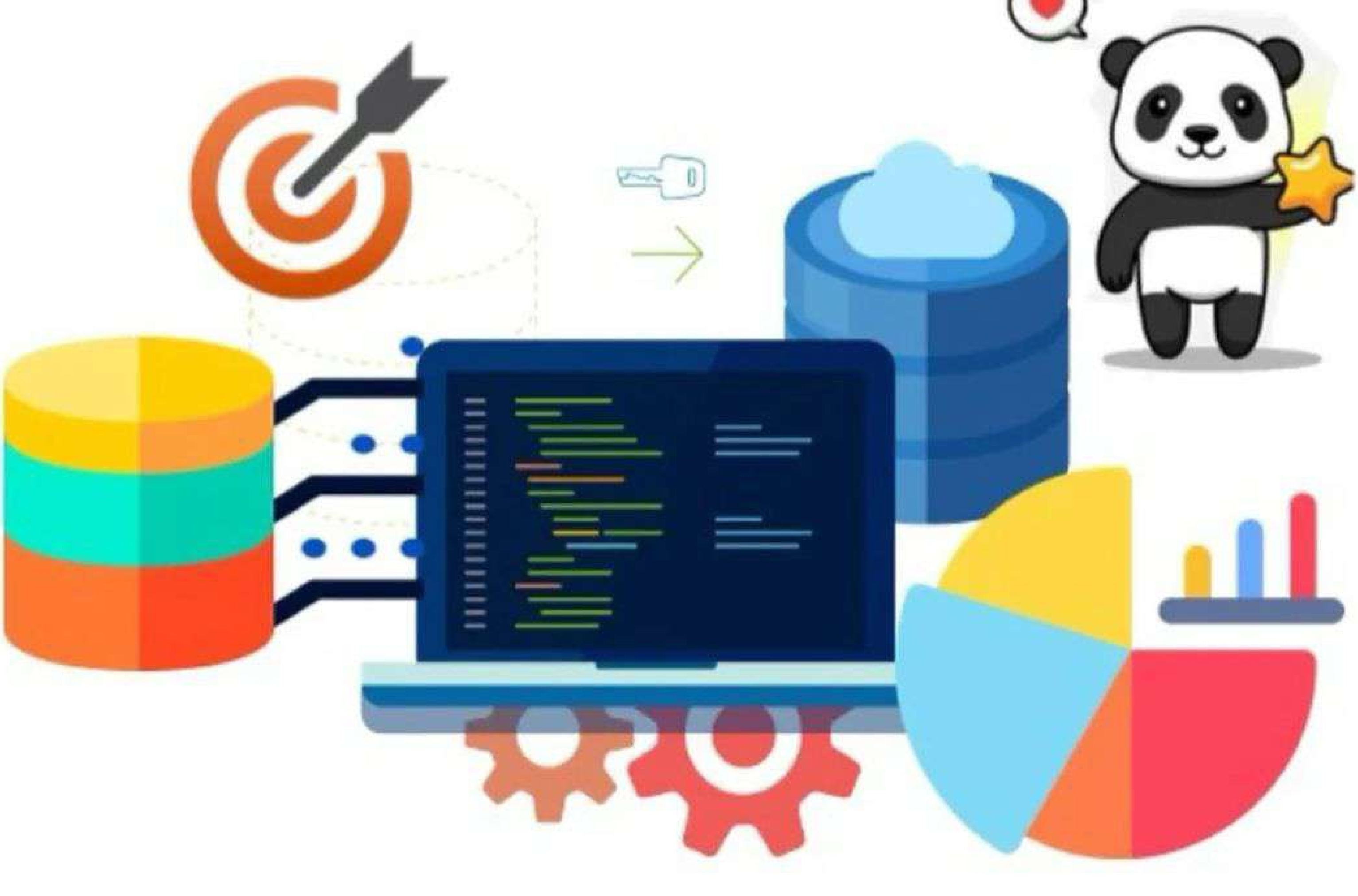


Order
• Order By ASC
• Order By DESC
Joins
• Inner Join
• Left Join
• Right Join
• Full Join
Group
• Group By Column
• Having
Functions
• AVG()
• MAX()
• MIN()
• SUM()
• COUNT()
Alias
• As
Where
• Like
• In
• Between
• Any
• Exist, All

SQL	NoSQL
Structured	Unstructured
Table with fixed rows and columns- Static schema	Dynamic schema
Structured Data	Unstructured Data
Suited for complex trasactions	Suited for Big Data Analytics
Best for E-Commerce Platforms	Best for Social Media Platforms
MySQL, PostgreSQL	MongoDB, Cassandra

VS

MySQL Cheat Sheet



BEGINNER'S SQL GUIDE

Rishabh Mishra

Save For Later

1. Database Operations

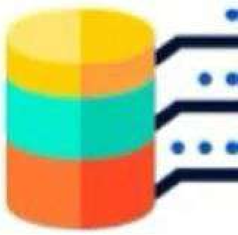
- **Create database:** CREATE DATABASE dbname;
- **Drop database:** DROP DATABASE dbname;
- **Select database:** USE dbname;
- **List all databases:** SHOW DATABASES;

2. Table Operations

- **Create table:** CREATE TABLE tablename (id INT, data VARCHAR(100));
- **Drop table:** DROP TABLE tablename;
- **Rename table:** RENAME TABLE oldname TO newname;
- **List all tables:** SHOW TABLES;
- **Describe table structure:** DESCRIBE tablename;
- **Truncate table:**TRUNCATE TABLE tablename;
- **Add column:** ALTER TABLE tablename ADD col_name datatype;
- **Drop column:** ALTER TABLE tablename DROP col_name;

- **Rename column:** ALTER TABLE tablename CHANGE old_col_name new_col_name datatype;
- **Modify column type:** ALTER TABLE tablename MODIFY col_name new_datatype;
- **Add a primary key:** ALTER TABLE tablename ADD PRIMARY KEY (col_name);
- **Drop a primary key:** ALTER TABLE tablename DROP PRIMARY KEY;
- **Add a unique constraint:** ALTER TABLE tablename ADD UNIQUE (col_name);
- **Add a foreign key:** ALTER TABLE tablename ADD CONSTRAINT fk_name FOREIGN KEY (col_name) REFERENCES other_table(col_name);
- **Drop a foreign key:** ALTER TABLE tablename DROP FOREIGN KEY fk_name;
- **Create an index:** CREATE INDEX index_name ON tablename (col_name);
- **Drop an index:** DROP INDEX index_name ON tablename;

3. Data Manipulation



- **Insert row:** INSERT INTO tablename (col1, col2) VALUES (value1, value2);
- **Update rows:** UPDATE tablename SET col1 = value1 WHERE condition;
- **Delete rows:** DELETE FROM tablename WHERE condition;
- **Select data:** SELECT * FROM tablename;
- **Select data with condition:** SELECT * FROM tablename WHERE condition;
- **Select and order data:** SELECT * FROM tablename ORDER BY col ASC/DESC;
- **Select distinct rows:** SELECT DISTINCT col FROM tablename;
- **Count rows:** SELECT COUNT(*) FROM tablename;
- **Sum:** SELECT SUM(col) FROM tablename;
- **Average:** SELECT AVG(col) FROM tablename;
- **Limit & Offset:** SELECT * FROM tablename LIMIT number OFFSET number;
- **Group data:** SELECT col, COUNT(*) FROM tablename GROUP BY col;
- **Having clause:** SELECT col1, col2, COUNT(*) FROM tablename GROUP BY col1, col2 HAVING COUNT(*) > 1;

4. JOINs

- **LEFT Join:** SELECT * FROM table1 LEFT JOIN table2 ON table1.col = table2.col;
- **RIGHT Join:** SELECT * FROM table1 RIGHT JOIN table2 ON table1.col = table2.col;
- **INNER Join:** SELECT * FROM table1 INNER JOIN table2 ON table1.col = table2.col;
- **FULL Join:** SELECT * FROM table1 LEFT JOIN table2 ON table1.col = table2.col UNION SELECT * FROM table1 RIGHT JOIN table2 ON table1.col = table2.col;
- **Cross Join:** SELECT * FROM table1 CROSS JOIN table2;
- **Self Join:** SELECT a.col, b.col FROM table a JOIN table b ON a.common_col = b.common_col WHERE condition;
- **Natural Join:** SELECT * FROM table1 NATURAL JOIN table2;

5. Subqueries

- **Scalar Subquery (Returns single value):** SELECT col_name FROM table_name WHERE col_name = (SELECT col_name FROM another_table WHERE condition);
- **Row Subquery (Returns single row):** SELECT col1, col2 FROM table_name WHERE (col1, col2) = (SELECT col1, col2 FROM another_table WHERE condition);
- **Col Subquery (Returns single col):** SELECT col_name FROM table_name WHERE col_name IN (SELECT col_name FROM another_table WHERE condition);
- **Table Subquery (Returns a table):** SELECT * FROM (SELECT col1, col2 FROM table_name WHERE condition) AS subquery_alias;
- **Correlated Subquery (Reference to a col from the outer query):** SELECT col_name FROM table_name outer_table_alias WHERE col_name_operator (SELECT col_name FROM another_table WHERE condition = outer_table_alias.col_name);
- **Exists Subquery (Checks for the existence of rows in a subquery):** SELECT col_name FROM table_name WHERE EXISTS (SELECT col_name FROM another_table WHERE condition);
- **NOT EXISTS Subquery (Checks for the non-existence of rows in a subquery):** SELECT col_name FROM table_name WHERE NOT EXISTS (SELECT col_name FROM another_table WHERE condition);

6. Text and String Functions

- **Concatenate string:** SELECT CONCAT(col1, ' ', col2) AS col12 FROM tablename;
- **Uppercase:** SELECT UPPER(col) FROM tablename;
- **Lowercase:** SELECT LOWER(col) FROM tablename;
- **Substring:** SELECT SUBSTRING(col, 1, 10) FROM tablename;
- **Replace text:** SELECT REPLACE(col, 'old', 'new') FROM tablename;
- **Length of a string:** SELECT LENGTH(col) FROM tablename;
- **Trim spaces:** SELECT TRIM(col) FROM tablename;
- **Find position of substring:** SELECT INSTR(col, 'substring') FROM tablename;

7. Numeric and Date Functions

- **Round number:** SELECT ROUND(col, decimals) FROM tablename;
- **Get current date:** SELECT CURDATE();
- **Get current time:** SELECT CURTIME();
- **Extract year from date:** SELECT YEAR(col) FROM tablename;
- **Extract month from date:** SELECT MONTH(col) FROM tablename;
- **Date difference:** SELECT DATEDIFF(date1, date2) FROM tablename;
- **Add days to a date:** SELECT DATE_ADD(col, INTERVAL 10 DAY) FROM tablename;
- **Format date:** SELECT DATE_FORMAT(col, '%Y-%m-%d') FROM tablename;

8. Set Operations

- **Union:** SELECT col FROM table1 UNION SELECT col FROM table2;
- **Union All:** SELECT col FROM table1 UNION ALL SELECT col FROM table2;
- **Except:** SELECT col FROM table1 WHERE NOT EXISTS (SELECT col FROM table2 WHERE table1.col = table2.col);

9. Aggregate Functions

- **Minimum value:** SELECT MIN(col) FROM tablename;
- **Maximum value:** SELECT MAX(col) FROM tablename;
- **Average:** SELECT AVG(col) FROM tablename;
- **Standard deviation:** SELECT STDDEV(col) FROM tablename;
- **Variance:** SELECT VARIANCE(col) FROM tablename;
- **Group concat:** GROUP_CONCAT(expression SEPARATOR 'separator');
- **Sum over:** SUM(expression) OVER (PARTITION BY col ORDER BY col);

10. Window Functions

- **ROW_NUMBER():** ROW_NUMBER() OVER (ORDER BY col_name)
- **RANK():** RANK() OVER (ORDER BY col_name)
- **DENSE_RANK():** DENSE_RANK() OVER (ORDER BY col_name)
- **NTILE():** NTILE(num_buckets) OVER (ORDER BY col_name)
- **LAG():** LAG(col_name, offset, default_value) OVER (ORDER BY col_name)
- **LEAD():** LEAD(col_name, offset, default_value) OVER (ORDER BY col_name)
- **FIRST_VALUE():** FIRST_VALUE(col_name) OVER (ORDER BY col_name)
- **LAST_VALUE():** LAST_VALUE(col_name) OVER (ORDER BY col_name)
- **CUME_DIST():** CUME_DIST() OVER (ORDER BY col_name)
- **PERCENT_RANK():** PERCENT_RANK() OVER (ORDER BY col_name)
- **PERCENTILE_CONT():** PERCENTILE_CONT(percent) WITHIN GROUP (ORDER BY col_name)
- **PERCENTILE_DISC():** PERCENTILE_DISC(percent) WITHIN GROUP (ORDER BY col_name)
- **NTH_VALUE():** NTH_VALUE(col_name, n) WITHIN GROUP (ORDER BY col_name)

11. Stored Procedure & CTE

- **Stored Procedure:** CREATE PROCEDURE procedure_name ([parameters])

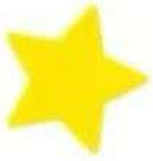
```
BEGIN
  -- SQL statements;
END;)
```

- **Common Table Expression (CTE):** WITH cte_name AS (

```
  -- CTE query here )
SELECT * FROM cte_name;
```

12. Conditional Expressions

- **IF function:** SELECT IF(condition, value_if_true, value_if_false) FROM tablename;
- **Simple CASE statement:** SELECT col, CASE WHEN value1 THEN result1 ELSE default_result END FROM tablename;
- **Searched CASE statement:** SELECT col, CASE WHEN condition1 THEN result1 ELSE default_result END FROM tablename;
- **COALESCE function:** SELECT COALESCE(col, 'default_value') FROM tablename;
- **NULLIF function:** SELECT NULLIF(col, 'default_value') FROM tablename;
- **IFNULL function:** SELECT IFNULL(col, 'default_value') FROM tablename;
- **NULLIFNULL function:** SELECT NULLIFNULL(col, 'default_value') FROM tablename;



13. User and Permissions

- **Create user:** CREATE USER 'user'@'host' IDENTIFIED BY 'password';
- **Grant permissions:** GRANT ALL PRIVILEGES ON dbname.* TO 'user'@'host';
- **Revoke permissions:** REVOKE ALL PRIVILEGES ON dbname.* FROM 'user'@'host';
- **Set password:** SET PASSWORD FOR 'user'@'host' = PASSWORD('newpassword');
- **Change password:** ALTER USER 'user'@'host' IDENTIFIED BY 'newpassword';
- **Show grants:** SHOW GRANTS FOR 'user'@'host';
- **Drop user:** DROP USER 'user'@'host';
- **Flush privileges:** FLUSH PRIVILEGES;

14. Backup and Recovery

- Backup a database: mysqldump -u username -p dbname > backupfile.sql;
- Restore a database: mysql -u username -p dbname < backupfile.sql;