SQL Interview Questions - KirkYagami

Fields = Columns
Records = Rows

SQL VS MySQL

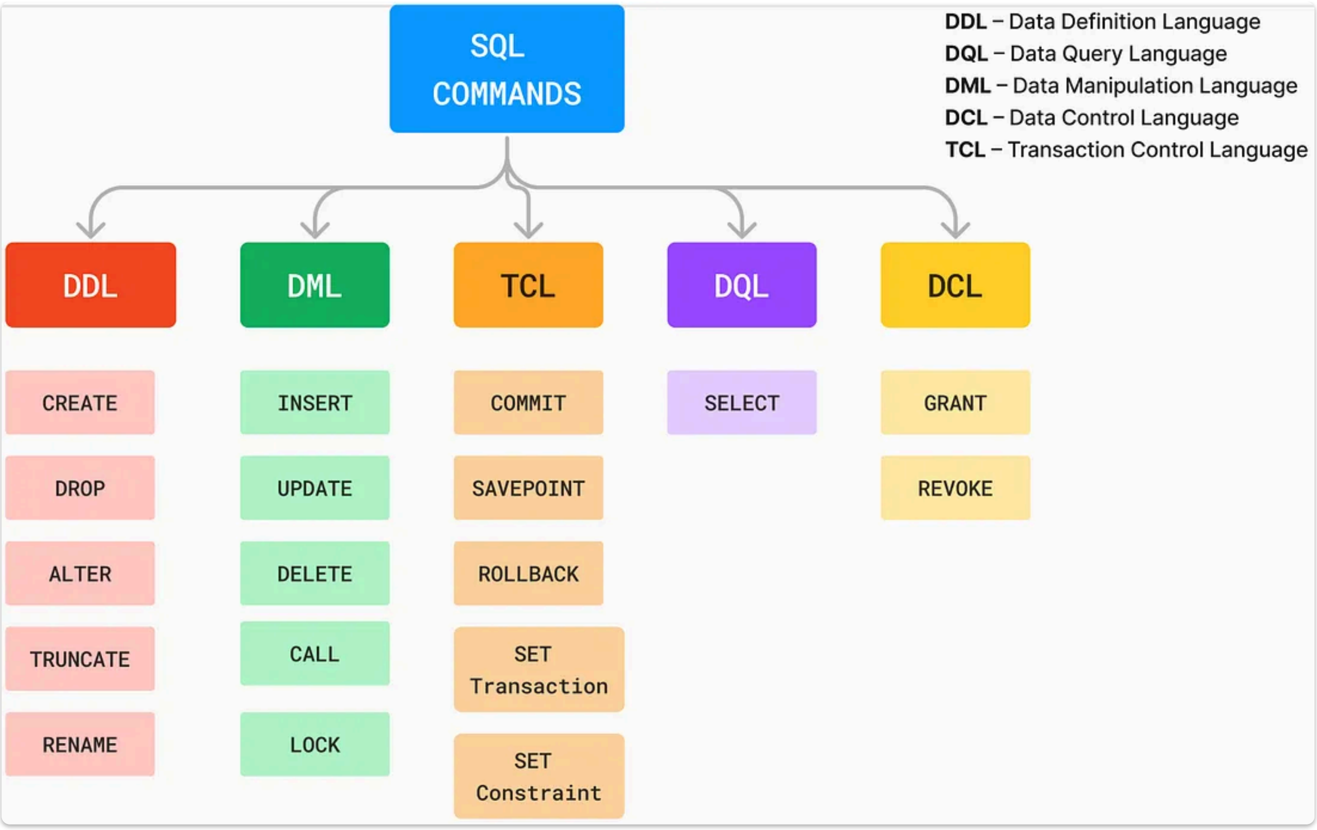
SQL (Structured Query Language):

- SQL is a standard language used for managing and manipulating relational databases.
- It provides a set of commands for querying, updating, and managing databases.
- SQL is not specific to any particular database system and can be used with various relational database management systems (RDBMS) such as MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server, etc.
- It follows ANSI/ISO standards, ensuring portability across different database platforms.
- SQL statements include commands like SELECT, INSERT, UPDATE, DELETE, CREATE TABLE, ALTER TABLE, etc.
- SQL can be used to create, modify, and delete databases, tables, and indexes, as well as perform data manipulation and retrieval operations.

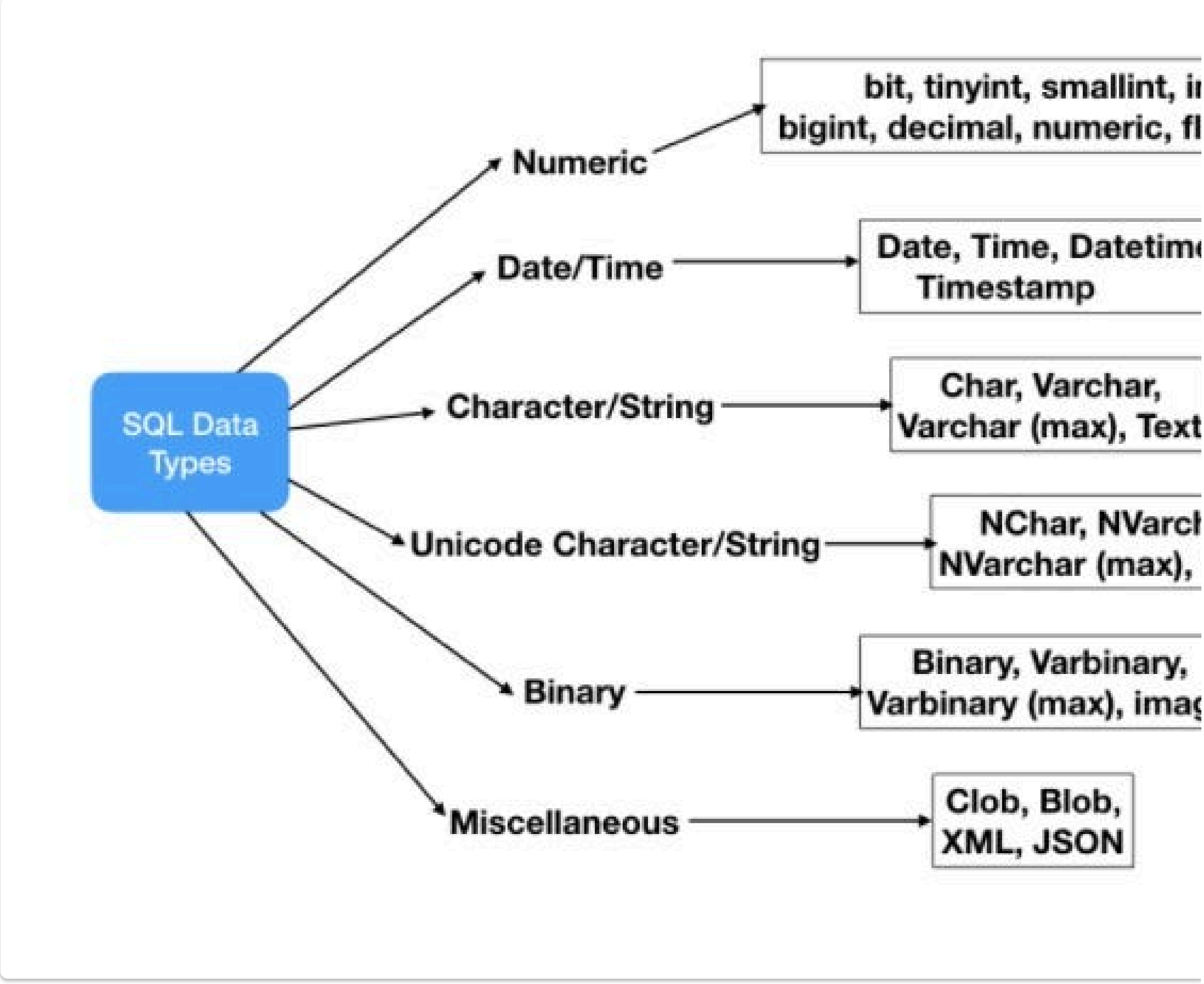
MySQL:

- MySQL is a popular open-source relational database management system (RDBMS).
- MySQL is known for its speed, reliability, and ease of use, making it a preferred choice for many web applications and small to medium-sized businesses.

Type of SQL commands



SQL Data types



ACID Properties

1. Atomicity:
    - Ensures that database transactions are either fully completed/successful or fully aborted/failed.
    - Transactions are considered as indivisible units of work, and if any part of a transaction fails, the entire transaction is rolled back to its original state.
2. Consistency:
    - Guarantees that the database remains in a consistent state before and after the execution of a transaction.
    - All constraints, rules, and relationships defined in the database schema must be maintained, ensuring data integrity.
3. Isolation:
    - Ensures that transactions are executed independently and securely, without interference from other concurrent transactions.
    - Transactions should appear to execute serially, even when multiple transactions are executing concurrently.

4. Durability:
    - Guarantees that once a transaction is committed, its effects are permanently stored in the database, even in the event of system failures.
    - Changes made by committed transactions should survive system crashes or power outages, ensuring data persistence.

SQL Queries for Interview 2nd file

1. What is Database?

Electronic Storage

A database is an organized collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modeling approaches.

2. What is DBMS?

DBMS stands for Database Management System. DBMS is a system software responsible for the creation, retrieval, updation, and management of the database. It ensures that our data is consistent, organized, and is easily accessible by serving as an interface between the database and its end-users or application software.

3. What are Tables and Fields?

A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

4. What are Constraints in SQL?

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command. The constraints are:

- NOT NULL - Restricts NULL value from being inserted into a column.
- CHECK - Verifies that all values in a field satisfy a condition. `age >18`
- DEFAULT - Automatically assigns a default value if no value has been specified for the field.
- UNIQUE - Ensures unique values to be inserted into the field.
- INDEX - Indexes a field providing faster retrieval of records.
- PRIMARY KEY - Uniquely identifies each record in a table.

- FOREIGN KEY - Ensures referential integrity for a record in another table.
  - It references the PK of another table.

5. What is a Primary Key?

- unique identifier in a table for each record (row).
- Must contain UNIQUE values and implicit NOT NULL constraint.
- Only one primary key per table
- This primary key can be comprised of single or multiple columns

```
CREATE TABLE Students (   /* Create table with a single field as primary key */
   ID INT NOT NULL
   Name VARCHAR(255)
   PRIMARY KEY (ID)
);

CREATE TABLE Students (   /* Create table with multiple fields as primary key */
   ID INT NOT NULL
   LastName VARCHAR(255)
   FirstName VARCHAR(255) NOT NULL,
   CONSTRAINT PK_Student  PRIMARY KEY (ID, FirstName)
);

ALTER TABLE Students   /* Set a column as primary key */ DDL
ADD PRIMARY KEY (ID);

ALTER TABLE Students   /* Set multiple columns as primary key */
ADD CONSTRAINT PK_Student   /*Naming a Primary Key*/
PRIMARY KEY (ID, FirstName);
```

- write a sql statement to add primary key 't_id' to the table 'teachers'.

```
ALTER TABLE teachers ADD PRIMARY KEY (t_id);
```

- Write a SQL statement to add primary key constraint 'pk_a' for table 'table_a' and fields 'col_b, col_c'.

```
ALTER TABLE table_a ADD CONSTRAINT pk_a PRIMARY KEY (col_b, col_c);
```

6. What is a UNIQUE constraint?

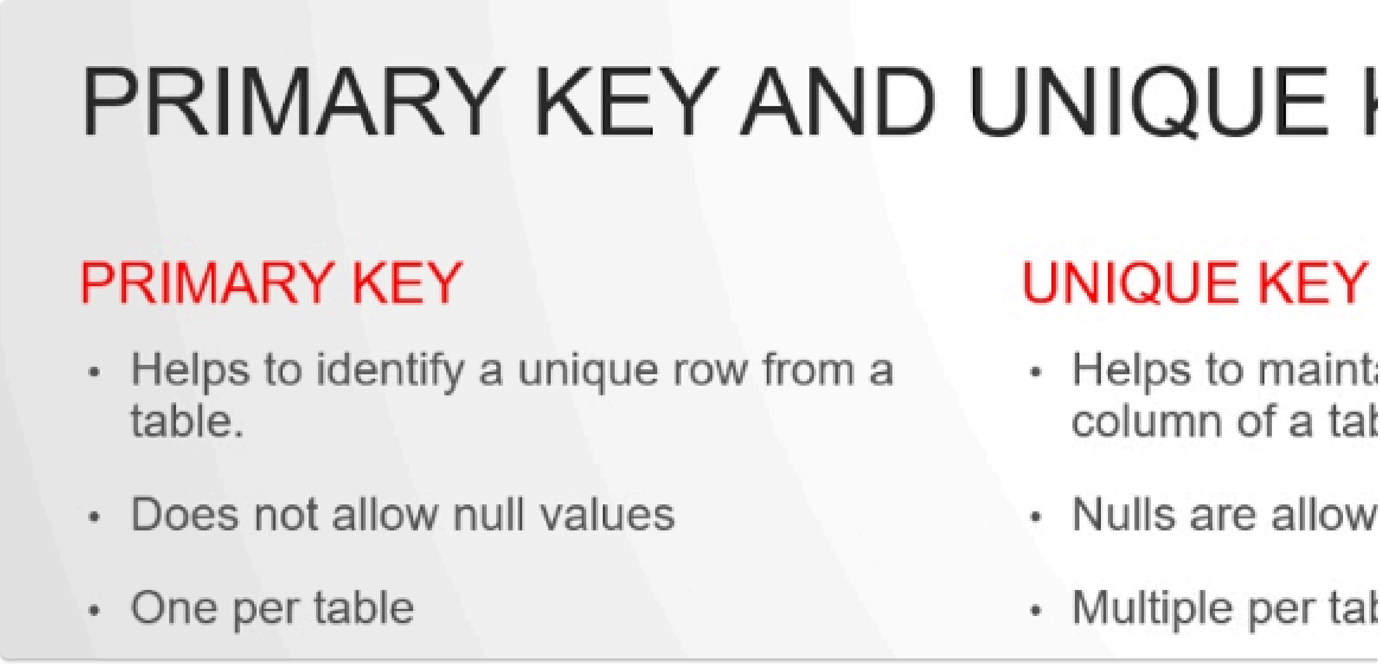
A UNIQUE constraint ensures that all values in a column are different. This provides uniqueness for the column(s) and helps identify each row uniquely. Unlike primary key, there can be multiple unique constraints defined per table. The code syntax for UNIQUE is quite similar to that of PRIMARY KEY and can be used interchangeably.

```
CREATE TABLE Students (   /* Create table with a single field as unique */
   ID INT NOT NULL UNIQUE
   Name VARCHAR(255)
);

CREATE TABLE Students (   /* Create table with multiple fields as unique */
   ID INT NOT NULL
   LastName VARCHAR(255)
   FirstName VARCHAR(255) NOT NULL
   CONSTRAINT PK_Student
   UNIQUE (ID, FirstName)
);

ALTER TABLE Students   /* Set a column as unique */
ADD UNIQUE (ID);

ALTER TABLE Students   /* Set multiple columns as unique */
ADD CONSTRAINT PK_Student   /* Naming a unique constraint */
UNIQUE (ID, FirstName);
```



PRIMARY KEY AND UNIQUE I

PRIMARY KEY

- Helps to identify a unique row from a table.
- Does not allow null values
- One per table

UNIQUE KEY

- Helps to maint column of a tal
- Nulls are allow
- Multiple per tal

7. What is a Foreign Key?

- comprises of single or more fields in a table.
- that essentially refers to the PRIMARY KEY in another table. Foreign key constraint ensures referential integrity in the relation between two tables.

The table with the foreign key constraint is labeled as the child table, and the table containing the candidate key is labeled as the referenced or parent table.

```
CREATE TABLE Students (   /* Create table with foreign key - Way 1 */
   ID INT NOT NULL
   Name VARCHAR(255)
   LibraryID INT
   PRIMARY KEY (ID)
   FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

CREATE TABLE Students (   /* Create table with foreign key - Way 2 */
   ID INT NOT NULL PRIMARY KEY
   Name VARCHAR(255)
   LibraryID INT FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

ALTER TABLE Students   /* Add a new foreign key */
ADD FOREIGN KEY (LibraryID)
REFERENCES Library (LibraryID);
```

8. What is a Join? List its different types.

The SQL Join clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.

There are four different types of JOINs in SQL:

- (INNER) JOIN: Retrieves records that have matching values in both tables involved in the join. This is the widely used join for queries.

```
SELECT *
FROM Table_A
JOIN Table_B;
SELECT *
FROM Table_A
INNER JOIN Table_B;
```

- LEFT (OUTER) JOIN: Retrieves all the records/rows from the left and the matched records/rows from the right table.

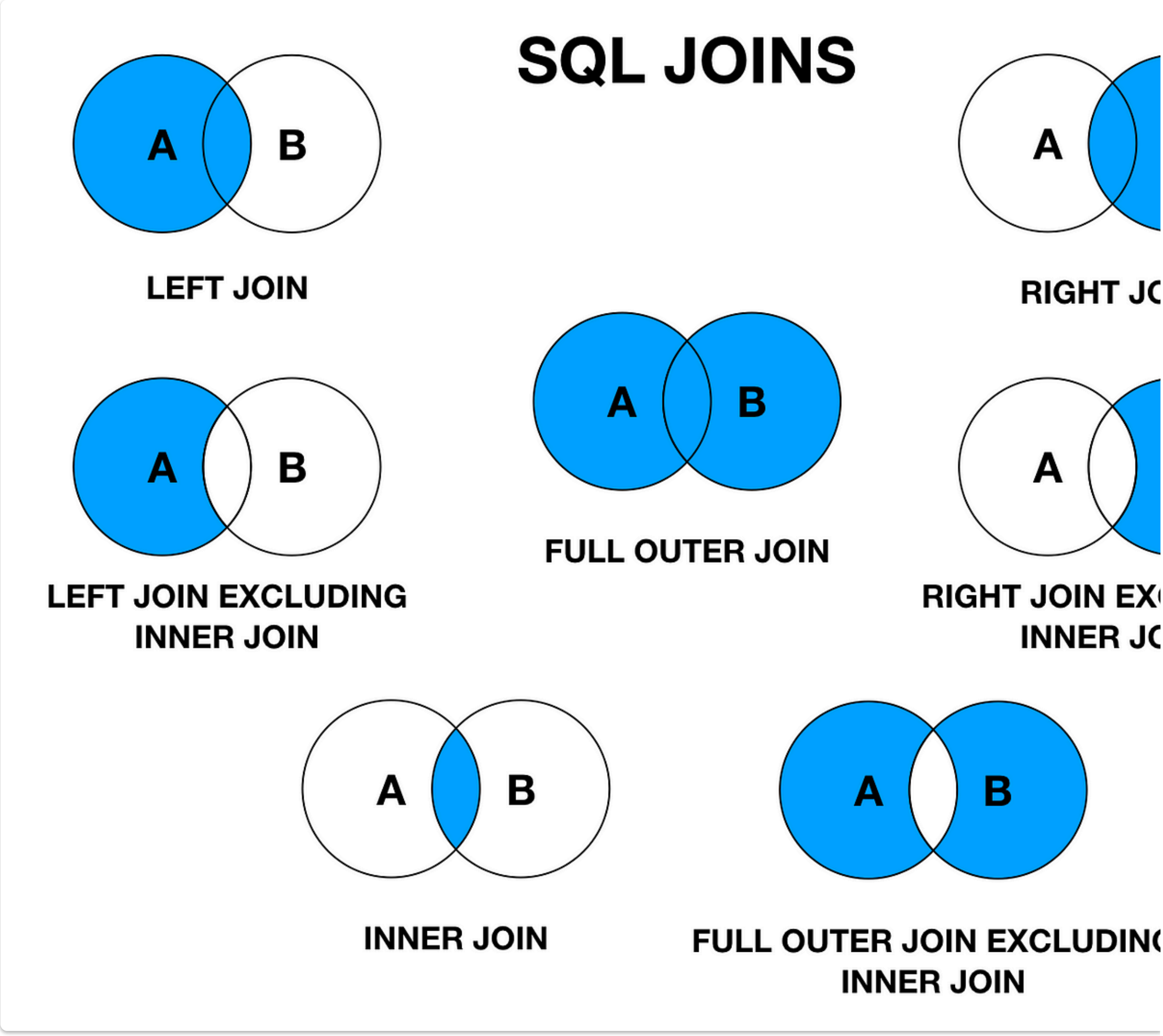
```
SELECT *
FROM Table_A A
LEFT JOIN Table_B B
ON A.col = B.col;
```

- RIGHT (OUTER) JOIN: Retrieves all the records/rows from the right and the matched records/rows from the left table.

```
SELECT *
FROM Table_A A
RIGHT JOIN Table_B B
ON A.col = B.col;
```

- FULL (OUTER) JOIN: Retrieves all the records where there is a match in either the left or right table.

```
SELECT *
FROM Table_A A
FULL JOIN Table_B B
ON A.col = B.col;
```



9. What is a Self-Join?

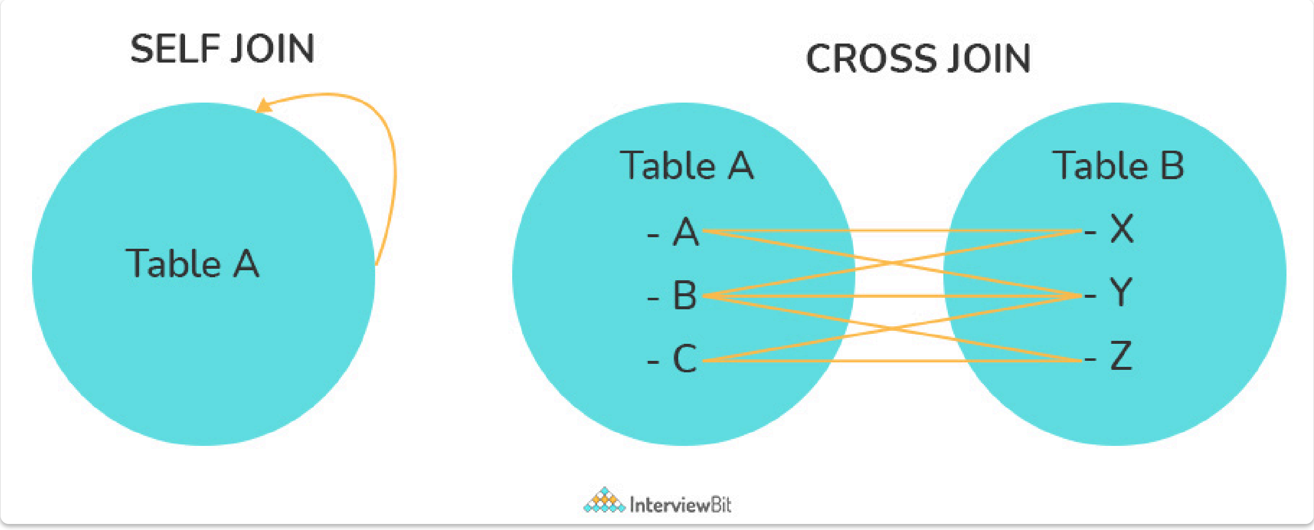
A self JOIN is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

```
SELECT A.emp_id AS "Emp_ID",A.emp_name AS "Employee",
B.emp_id AS "Sup_ID",B.emp_name AS "Supervisor"
FROM employee A, employee B
WHERE A.emp_sup = B.emp_id;
```

10. What is a Cross-Join?

Cross join can be defined as a cartesian product of the two tables included in the join. The table after join contains the same number of rows as in the cross-product of the number of rows in the two tables. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

```
SELECT stu.name, sub.subject
FROM students AS stu
CROSS JOIN subjects AS sub;
```



11. What is an Index? Explain its different types.

```
-- INDEX (BTree data structure)
-- Indexes are used to find values within a specific column more quickly | But it
uses your Storage
-- so use wisely
-- Indexes increase SELECT speed but reduces INSERT, UPDATE, DELETE.
-- Single column index
CREATE INDEX last_name_idx ON customers (last_name);
```

```sql
-- Multi column index
CREATE INDEX last_name_first_name_idx ON customers (last_name, first_name);
```

```sql
CREATE INDEX index_name   /* Create Index */
ON table_name (column_1, column_2);
DROP INDEX index_name;   /* Drop Index */
```

There are different types of indexes that can be created for different purposes:

- **Unique and Non-Unique Index:**

Unique indexes are indexes that help maintain data integrity by ensuring that no two rows of data in a table have identical key values. Once a unique index has been defined for a table, uniqueness is enforced whenever keys are added or changed within the index.

```sql
CREATE UNIQUE INDEX myIndex
ON students (enroll_no);
```

Non-unique indexes, on the other hand, are not used to enforce constraints on the tables with which they are associated. Instead, non-unique indexes are used solely to improve query performance by maintaining a sorted order of data values that are used frequently.

- **Clustered and Non-Clustered Index:**

Clustered indexes are indexes whose order of the rows in the database corresponds to the order of the rows in the index. This is why only one clustered index can exist in a given table, whereas, multiple non-clustered indexes can exist in the table.

The only difference between clustered and non-clustered indexes is that the database manager attempts to keep the data in the database in the same order as the corresponding keys appear in the clustered index.

12. What is a Query?

A query is a request for data or information from a database table or combination of tables. A database query can be either a select query or an action query.

```sql
SELECT fname, lname    /* select query */
FROM myDb.students
WHERE student_id = 1;
```

```
UPDATE myDB.students    /* action query */
SET fname = 'Captain', lname = 'America'
WHERE student_id = 1;
```

13. What is a Subquery? What are its types?

A subquery is a query within another query, also known as a nested query or inner query.
First the subquery is executed and its result is passed to the outer query.

```
SELECT name, email, mob, address
FROM myDb.contacts
WHERE roll_no IN (
                 SELECT roll_no
                 FROM myDb.students
                 WHERE subject = 'Maths'
                 );
```

14. What are some common clauses used with SELECT query in SQL?

Some common SQL clauses used in conjuction with a SELECT query are as follows:

- WHERE clause in SQL is used to filter records that are necessary, based on specific conditions.
- ORDER BY clause in SQL is used to sort the records based on some field(s) in ascending (ASC) or descending order (DESC).

```
SELECT *
FROM myDB.students
WHERE graduation_year = 2019
ORDER BY studentID DESC;
```

- GROUP BY clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
- HAVING clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

```
SELECT COUNT(studentId), country
FROM myDB.students
WHERE country != "INDIA"
```

```
GROUP BY country
HAVING COUNT(studentID) > 5;
```

15. What are UNION, MINUS and INTERSECT commands?

The UNION operator combines and returns the result-set retrieved by two or more SELECT statements.

The MINUS operator in SQL is used to remove duplicates from the result-set obtained by the second SELECT query from the result-set obtained by the first SELECT query and then return the filtered results from the first.
The INTERSECT clause in SQL combines the result-set fetched by the two SELECT statements where records from one match the other and then returns this intersection of result-sets.

Certain conditions need to be met before executing either of the above statements in SQL -

- Each SELECT statement within the clause must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement should necessarily have the same order

```
SELECT name FROM Students   /* Fetch the union of queries */
UNION
SELECT name FROM Contacts;

SELECT name FROM Students   /* Fetch the union of queries with duplicates*/
UNION ALL
SELECT name FROM Contacts;
```

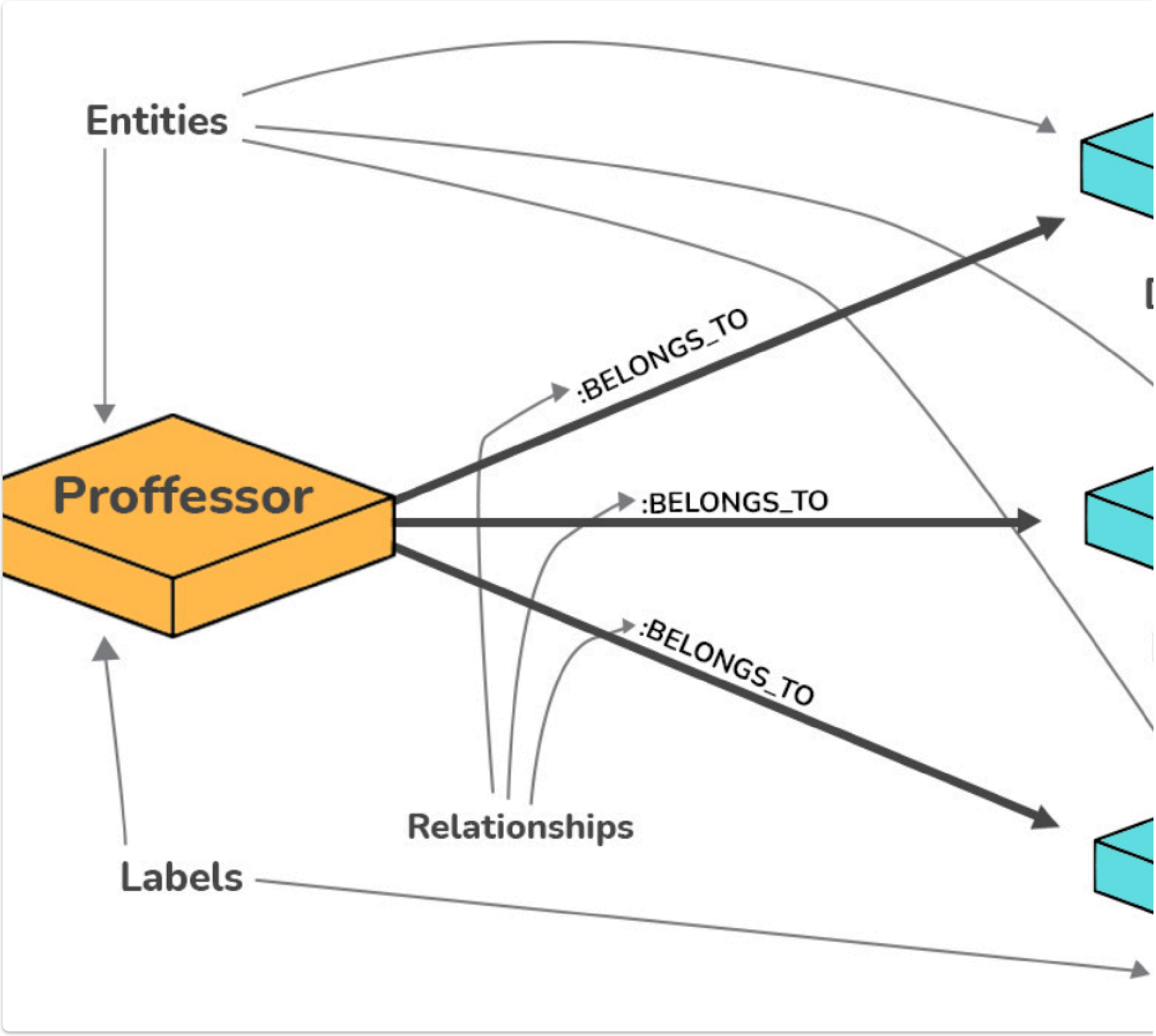
```
SELECT name FROM Students   /* Fetch names from students */
MINUS     /* that aren't present in contacts */
SELECT name FROM Contacts;
```

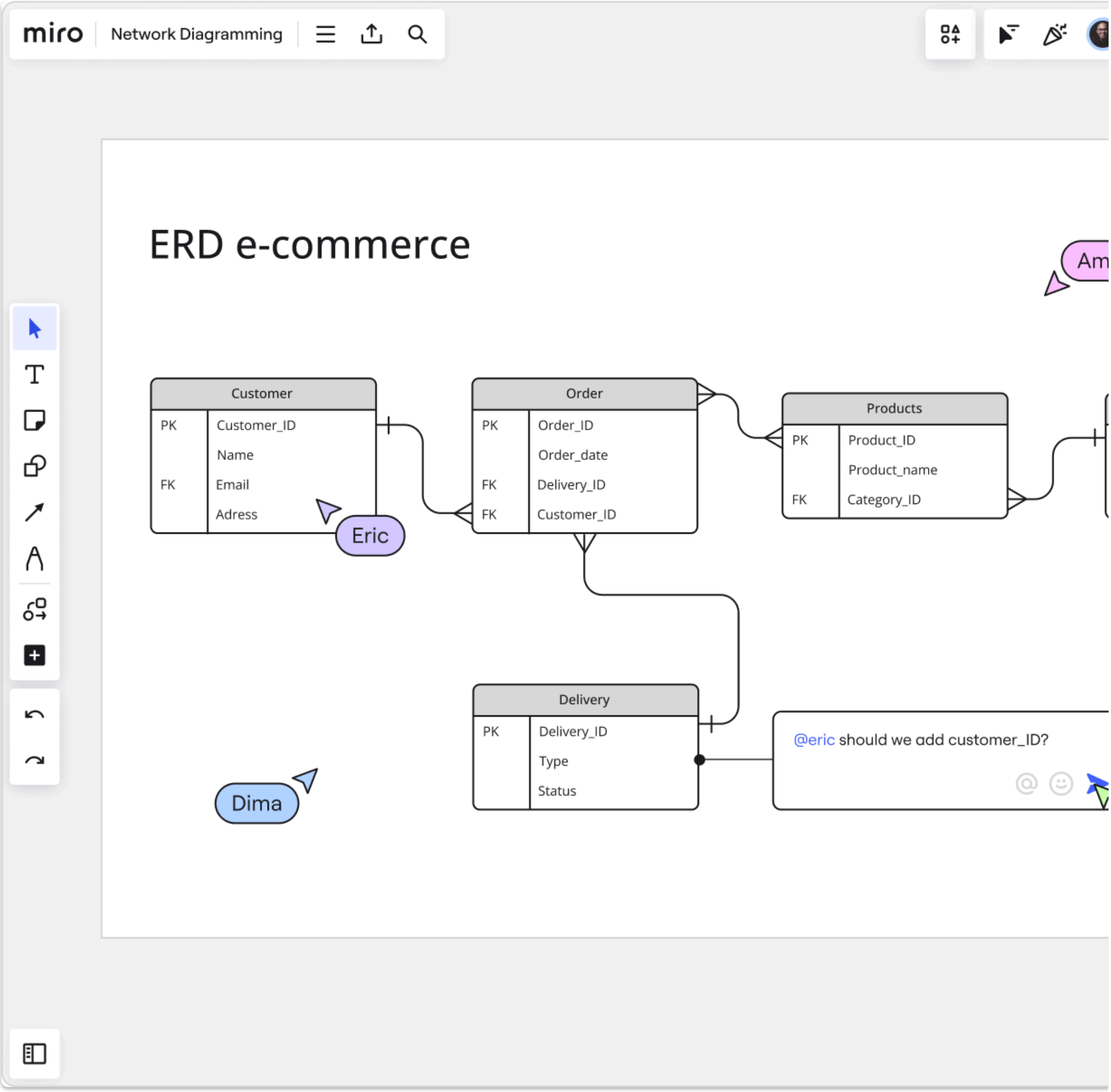
```
SELECT name FROM Students   /* Fetch names from students */
INTERSECT    /* that are present in contacts as well */
SELECT name FROM Contacts;
```

16. What are Entities and Relationships?

Entity: Real world objects that are identifiable For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

Relationships: Relations or links between entities that have something to do with each other. For example - The employee's table in a company's database can be associated with the salary table in the same database.





ERD e-commerce

Customer	
PK	Customer_ID
	Name
FK	Email
	Adress

Order	
PK	Order_ID
	Order_date
FK	Delivery_ID
FK	Customer_ID

Products	
PK	Product_ID
	Product_name
FK	Category_ID

Delivery	
PK	Delivery_ID
	Type
	Status

@eric should we add customer_ID?

17. List the different types of relationships in SQL.

- One-to-One - This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
- One-to-Many & Many-to-One - This is the most commonly used relationship where a record in a table is associated with multiple records in the other table.
- Many-to-Many - This is used in cases when multiple instances on both sides are needed for defining a relationship.
- Self-Referencing Relationships - This is used when a table needs to define a relationship with itself.

18. What is an Alias in SQL?

An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name.

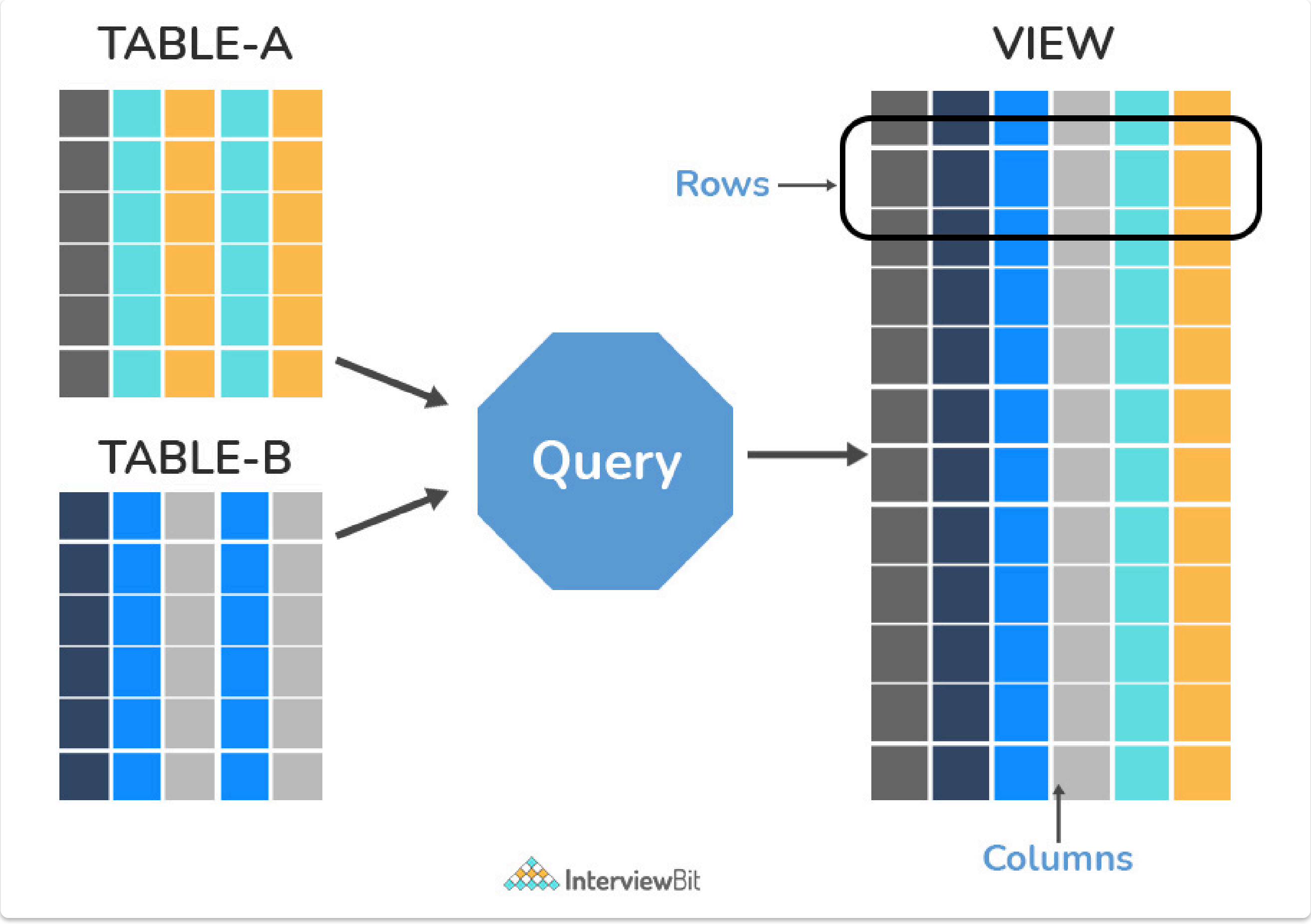
An alias is represented explicitly by the AS keyword but in some cases, the same can be performed without it as well. Nevertheless, using the AS keyword is always a good practice.

```
SELECT A.emp_name AS "Employee"  /* Alias using AS keyword */
B.emp_name AS "Supervisor"
FROM employee A, employee B   /* Alias without AS keyword */
WHERE A.emp_sup = B.emp_id;
```

Write an SQL statement to select all from table "Limited" with alias "Ltd".

19. What is a View?

A view in SQL is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.



20. What is Normalization?

https://youtu.be/EGEwkad_llA?t=70
Watch from where it starts.

21. What is Denormalization?

Denormalization is the process of intentionally introducing redundancy into a database schema to improve query performance by reducing the need for joins and speeding up data retrieval.

22. What is the difference between DELETE and TRUNCATE statements?

The TRUNCATE command is used to delete all the rows from the table and free the space containing the table.
The DELETE command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.
The DROP command erases the entire table data and its structure.

23. What are Aggregate and Scalar functions?

An aggregate function performs operations on a collection of values to return a single scalar value. Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

- AVG() - Calculates the mean of a collection of values.
- COUNT() - Counts the total number of records in a specific table or view.
- MIN() - Calculates the minimum of a collection of values.
- MAX() - Calculates the maximum of a collection of values.
- SUM() - Calculates the sum of a collection of values.
- FIRST() - Fetches the first element in a collection of values.
- LAST() - Fetches the last element in a collection of values.

Note: All aggregate functions described above ignore NULL values except for the COUNT function.

A scalar function returns a single value based on the input value. Following are the widely used SQL scalar functions:

- LEN() - Calculates the total length of the given field (column).

- UCASE() - Converts a collection of string values to uppercase characters.
- LCASE() - Converts a collection of string values to lowercase characters.
- MID() - Extracts substrings from a collection of string values in a table.
- CONCAT() - Concatenates two or more strings.
- RAND() - Generates a random collection of numbers of a given length.
- ROUND() - Calculates the round-off integer value for a numeric field (or decimal point values).
- NOW() - Returns the current date & time.
- FORMAT() - Sets the format to display a collection of values.

24. What is Pattern Matching in SQL?

SQL pattern matching provides for pattern search in data if you have no clue as to what that word should be. This kind of SQL query uses wildcards to match a string pattern, rather than writing the exact word. The LIKE operator is used in conjunction with SQL Wildcards to fetch the required information.

- Using the % wildcard to perform a simple search

The % wildcard matches zero or more characters of any type and can be used to define wildcards both before and after the pattern. Search a student in your database with first name beginning with the letter K:

```
SELECT *
FROM students
WHERE first_name LIKE 'K%'
```

- Omitting the patterns using the NOT keyword

Use the NOT keyword to select records that don't match the pattern. This query returns all students whose first name does not begin with K.

```
SELECT *
FROM students
WHERE first_name NOT LIKE 'K%'
```

- Matching a pattern anywhere using the % wildcard twice

Search for a student in the database where he/she has a K in his/her first name.

```
SELECT *
FROM students
WHERE first_name LIKE '%Q%'
```

- Using the _ wildcard to match pattern at a specific position

The _ wildcard matches exactly one character of any type. It can be used in conjunction with % wildcard. This query fetches all students with letter K at the third position in their first name.

```sql
SELECT *
FROM students
WHERE first_name LIKE '__K%'
```

- Matching patterns for a specific length

The _ wildcard plays an important role as a limitation when it matches exactly one character. It limits the length and position of the matched results. For example -

```sql
SELECT *   /* Matches first names with three or more letters */
FROM students
WHERE first_name LIKE '___%'

SELECT *   /* Matches first names with exactly four characters */
FROM students
WHERE first_name LIKE '____'
```

25. What is User-defined function? What are its various types?

The user-defined functions in SQL are like functions in any other programming language that accept parameters, perform complex calculations, and return a value. They are written to use the logic repetitively whenever required. There are two types of SQL user-defined functions:

- Scalar Function: As explained earlier, user-defined scalar functions return a single scalar value.
- Table-Valued Functions: User-defined table-valued functions return a table as output.
  - Inline: returns a table data type based on a single SELECT statement.
  - Multi-statement: returns a tabular result-set but, unlike inline, multiple SELECT statements can be used inside the function body.

```sql
-- Change the delimiter to enable defining the function
DELIMITER //

-- Create a user-defined function to calculate the factorial of a number
CREATE FUNCTION CalculateFactorial (n INT)
RETURNS INT
```

```
BEGIN
    DECLARE result INT DEFAULT 1;
    DECLARE i INT DEFAULT 1;

    WHILE i <= n DO
        SET result = result * i;
        SET i = i + 1;
    END WHILE;

    RETURN result;
END //

-- Reset the delimiter back to semicolon
DELIMITER ;
```

26. What is OLTP?

OLTP stands for Online Transaction Processing.

- Stores updated/current data
- Fast
- Write operations
  Examples: retail point-of-sale systems (eg. DMART), online banking systems, airline reservation systems, and order processing systems.

OLAP stands for Online Analytical Processing

- Stores historical Data
- Read operations
- Data is taken from the OLTP database and put in a warehouse for analysis.

27. What is a Stored Procedure?

- We will write the procedure once and we can use as many times as we need.
- No need to write the logic again and again.

```
DELIMITER $$
CREATE PROCEDURE FetchAllStudents()
BEGIN
SELECT *  FROM myDB.students;
END $$
DELIMITER ;
```

28. How to create empty tables with the same structure as another table?

Creating empty tables with the same structure can be done smartly by fetching the records of one table into a new table using the INTO operator while fixing a WHERE clause to be false for all records. Hence, SQL prepares the new table with a duplicate structure to accept the fetched records but since no records get fetched due to the WHERE clause in action, nothing is inserted into the new table.

```sql
SELECT * INTO Students_copy
FROM Students WHERE 1 = 2;

-- Only structure is copied and no data.
```