***Mysql***

**Q1. What is a database? Differentiate between SQL and NoSQL databases.**

**A database is a collection of organized data that is stored and accessed electronically. It can be thought of as an electronic filing system where data is organized in tables, columns, and rows for efficient management and retrieval. Databases are used in a wide variety of applications, including business, education, healthcare, and more.**

**SQL (Structured Query Language) and NoSQL (Not Only SQL) are two different types of database systems that differ in their data model, structure, and query language. Here are the main differences between the two:**

1. **Data Model: SQL databases are based on a relational model, where data is stored in tables with rows and columns. NoSQL databases, on the other hand, can use a variety of data models, including document-based, graph-based, key-value, and column-family.**
2. **Schema: SQL databases have a pre-defined schema that determines the structure of the data. NoSQL databases have a flexible schema that allows for more dynamic and evolving data structures.**
3. **Query Language: SQL databases use SQL, a standard query language, to access and manipulate data. NoSQL databases use a variety of query languages, including MongoDB Query Language, Cassandra Query Language, and GraphQL.**
4. **Scalability: NoSQL databases are designed to scale horizontally, meaning that they can handle a larger volume of data by adding more servers to the database cluster. SQL databases are designed to scale vertically, meaning that they can handle a larger volume of data by upgrading the hardware resources of the server.**
5. **ACID Compliance: SQL databases are generally ACID-compliant, meaning that they ensure data consistency, durability, and isolation. NoSQL databases can sacrifice some of these properties for performance and scalability.**

**In summary, SQL databases are ideal for applications that require complex queries, transactions, and a well-defined schema. NoSQL databases are suitable for applications that require flexible data structures, high scalability, and fast performance.**

**Q2. What is DDL? Explain why CREATE, DROP, ALTER, and TRUNCATE are used with an example.**

DDL stands for Data Definition Language, which is a set of SQL commands used to define and modify the structure of a database. DDL commands are used to create, modify, and delete database objects such as tables, indexes, views, and procedures.

Here are some examples of DDL commands and how they are used:

1. CREATE: The CREATE command is used to create a new database object. For example, to create a new table named "employees" with columns for name, age, and salary, you would use the following SQL command:

**CREATE TABLE employees (**

**name VARCHAR(50),**

**age INT,**

**salary DECIMAL(10,2)**

**);**

1. DROP: The DROP command is used to delete an existing database object. For example, to delete the "employees" table, you would use the following SQL command:

**DROP TABLE employees;**

1. ALTER: The ALTER command is used to modify the structure of an existing database object. For example, to add a new column named "department" to the "employees" table, you would use the following SQL command:

**ALTER TABLE employees ADD COLUMN department VARCHAR(50);**

1. TRUNCATE: The TRUNCATE command is used to delete all data from an existing table. For example, to delete all data from the "employees" table, you would use the following SQL command:

In summary, DDL commands are used to define and modify the structure of a database. CREATE is used to create new objects, DROP is used to delete existing objects, ALTER is used to modify existing objects, and TRUNCATE is used to delete data from existing objects.

**Q3. What is DML? Explain INSERT, UPDATE, and DELETE with an example.**

DML stands for Data Manipulation Language, which is a set of SQL commands used to manipulate the data within a database. DML commands are used to insert, update, and delete data in database tables.

Here are some examples of DML commands and how they are used:

1. INSERT: The INSERT command is used to add new data to a table. For example, to add a new employee named "John Doe" to the "employees" table with a salary of $50,000, you would use the following SQL command:

**INSERT INTO employees (name, salary) VALUES ('John Doe', 50000);**

1. UPDATE: The UPDATE command is used to modify existing data in a table. For example, to update the salary of the employee named "John Doe" to $60,000, you would use the following SQL command:

**UPDATE employees SET salary = 60000 WHERE name = 'John Doe';**

1. DELETE: The DELETE command is used to remove data from a table. For example, to delete the employee named "John Doe" from the "employees" table, you would use the following SQL command:

**DELETE FROM employees WHERE name = 'John Doe';**

In summary, DML commands are used to manipulate the data within a database. INSERT is used to add new data to a table, UPDATE is used to modify existing data in a table, and DELETE is used to remove data from a table.

**Q4. What is DQL? Explain SELECT with an example.**

DQL stands for Data Query Language, which is a set of SQL commands used to retrieve data from a database. DQL commands are used to query and retrieve data from one or more tables.

Here is an example of a DQL command and how it is used:

1. SELECT: The SELECT command is used to retrieve data from one or more tables. For example, to retrieve the names and salaries of all employees from the "employees" table, you would use the following SQL command:

**SELECT name, salary FROM employees;**

This command retrieves the "name" and "salary" columns from the "employees" table. The result of this query would be a table with two columns: "name" and "salary", and as many rows as there are employees in the table.

You can also use the SELECT command to retrieve data based on certain conditions using the WHERE clause. For example, to retrieve the names and salaries of all employees who have a salary greater than $50,000, you would use the following SQL command:

**SELECT name, salary FROM employees WHERE salary > 50000;**

This command retrieves the "name" and "salary" columns from the "employees" table where the "salary" is greater than $50,000. The result of this query would be a table with two columns: "name" and "salary", and only the rows where the salary is greater than $50,000.

In summary, DQL commands are used to retrieve data from a database. SELECT is used to retrieve data from one or more tables, and can be used to retrieve all columns or specific columns, as well as to filter data based on certain conditions using the WHERE clause.

**Q5. Explain Primary Key and Foreign Key.**

Primary Key and Foreign Key are two important concepts in relational databases.

A Primary Key is a column or set of columns in a table that uniquely identifies each row in the table. Primary Keys are used to enforce data integrity and ensure that each row in a table is unique. Primary Keys can be composed of one or more columns, and are usually defined when a table is created. Primary Keys must be unique and cannot contain null values. An example of a primary key is an employee ID number in a table of employee data.

A Foreign Key is a column or set of columns in a table that refers to the Primary Key of another table. The purpose of a Foreign Key is to create a relationship between two tables. This relationship is used to ensure referential integrity, which means that the values in the Foreign Key column must match the values in the Primary Key column of the related table. Foreign Keys are used to enforce data integrity and ensure that data is consistent across related tables. An example of a foreign key is a column in a table of orders that refers to the primary key of a table of customers.

In summary, a Primary Key uniquely identifies each row in a table, while a Foreign Key is used to create a relationship between two tables by referring to the Primary Key of another table. The Primary Key and Foreign Key are important concepts in relational databases and are used to ensure data integrity and consistency across related tables.

**Q6. Write a python code to connect MySQL to python. Explain the cursor() and execute() method.**

To connect to MySQL from Python, you need to install the mysql-connector-python library. You can do this by running the following command in your terminal:

**pip install mysql-connector-python**

Once the library is installed, you can use the following code to connect to a MySQL database from Python:

**import mysql.connector**

**mydb = mysql.connector.connect(**

**host="localhost",**

**user="yourusername",**

**password="yourpassword",**

**database="yourdatabase"**

**)**

**mycursor = mydb.cursor()**

**mycursor.execute("SELECT \* FROM yourtable")**

**for x in mycursor:**

**print(x)**

In this code, we first import the mysql.connector library. We then use the mysql.connector.connect() method to connect to the MySQL database. We pass the host name, user name, password, and database name as parameters to the connect() method.

Next, we create a cursor() object using the mydb.cursor() method. The cursor is used to execute SQL statements and fetch results.

We then execute a SQL statement using the execute() method of the cursor object. In this example, we execute a SELECT statement to retrieve all records from a table.

Finally, we loop through the result set and print each record.

The cursor() method creates a cursor object, which is used to execute SQL statements and fetch results from the database. The execute() method is used to execute a SQL statement. The SQL statement is passed as a parameter to the execute() method. The execute() method can execute any SQL statement, such as SELECT, INSERT, UPDATE, and DELETE. The execute() method returns the number of rows affected by the SQL statement.

**Q7. Give the order of execution of SQL clauses in an SQL query.**

The order of execution of SQL clauses in an SQL query is as follows:

1. FROM clause: The FROM clause specifies the table or tables from which the data will be retrieved. This is the first clause in a SQL query.
2. WHERE clause: The WHERE clause is used to filter the rows returned by the query. This clause is executed after the FROM clause.
3. GROUP BY clause: The GROUP BY clause is used to group the rows returned by the query based on one or more columns. This clause is executed after the WHERE clause.
4. HAVING clause: The HAVING clause is used to filter the groups returned by the GROUP BY clause. This clause is executed after the GROUP BY clause.
5. SELECT clause: The SELECT clause is used to select the columns that will be returned by the query. This clause is executed after the HAVING clause.
6. ORDER BY clause: The ORDER BY clause is used to sort the rows returned by the query based on one or more columns. This clause is executed after the SELECT clause.
7. LIMIT clause: The LIMIT clause is used to limit the number of rows returned by the query. This clause is executed after the ORDER BY clause.

It is important to note that not all SQL queries will contain all of these clauses, and the order of the clauses may vary depending on the specific query. However, in general, the above order of execution is followed.