Assignment

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

Ans:Database:

A database is a structured collection of data that is organized in a way that makes it easy to manage, access, and update. It can be thought of as an electronic filing system where information is stored in tables, and relationships between the data are defined. Databases are used in various applications and systems to store and retrieve data efficiently.

SQL (Structured Query Language) Databases:

SQL databases are relational databases that use structured query language (SQL) for defining and manipulating the data. They are based on a table-based data model where data is organized into tables with rows and columns. Examples of SQL databases include MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. SQL databases are good for complex query-intensive operations and applications where the structure of the data is well-defined.

NoSQL Databases:

NoSQL databases, on the other hand, are non-relational or distributed databases. They are designed to handle large sets of unstructured or semi-structured data. NoSQL databases are more flexible than SQL databases and can handle different types of data, such as documents, key-value pairs, graphs, or wide-column stores. Examples of NoSQL databases include MongoDB, Cassandra, Redis, and Couchbase. NoSQL databases are often used in scenarios where scalability, high performance, and flexibility are crucial.

Differences:

Data Model:

- SQL databases use a structured, tabular data model.
- NoSQL databases use various data models, including document-oriented, key-value pairs, column-family, or graph formats.

Schema:

- SQL databases have a predefined schema, and any changes to the structure require altering the schema.
- NoSQL databases are schema-less, allowing for flexibility in adding or modifying fields without a predefined schema.

Scalability:

- SQL databases are generally vertically scalable, meaning you can increase the capacity of a single server.
- NoSQL databases are designed to be horizontally scalable, allowing for distribution of data across multiple servers.

Complexity of Queries:

- SQL databases are well-suited for complex queries and join operations.
- NoSQL databases may lack some of the advanced query capabilities but excel in simple queries and the retrieval of large volumes of data.

Use Cases:

- SQL databases are often used in scenarios where data has a well-defined structure, and ACID properties (Atomicity, Consistency, Isolation, Durability) are crucial, such as in financial applications.
- NoSQL databases are chosen for applications with rapidly changing data or where the scalability and flexibility of the database are critical, such as in big data and real-time web applications.

The choice between SQL and NoSQL depends on the specific requirements of the application and the nature of the data being handled.

Q2. What is DDL? Explain why CREATE, DROP, ALTER, and TRUNCATE are used with an example.
Ans:DDL (Data Definition Language):

DDL, or Data Definition Language, is a subset of SQL (Structured Query Language) that is used for defining and managing the structure of a relational database. DDL statements allow users to define, modify, and delete database objects such as tables, indexes, and schemas. The primary DDL commands include `CREATE`, `DROP`, `ALTER`, and `TRUNCATE`.

CREATE:

- The `CREATE` statement is used to create a new database object, such as a table, index, or view.
- Example: Creating a new table named "employees" with columns for employee ID, name, and salary.
- sql
- Copy code

```
CREATE TABLE
```

```
        INT PRIMARY
          VARCHAR 50
    DECIMAL 10  2
```

-

DROP:

- The `DROP` statement is used to delete existing database objects, such as tables or indexes.
- Example: Dropping the "employees" table.
- sql
- Copy code

```
DROP TABLE
```

-

ALTER:

- The `ALTER` statement is used to modify the structure of an existing database object, such as adding or removing columns from a table.
- Example: Adding a new column "department" to the "employees" table.
- sql
- Copy code

```
ALTER TABLE
ADD COLUMN            VARCHAR 50
```

-

TRUNCATE:

- The `TRUNCATE` statement is used to remove all rows from a table, effectively deleting all the data while keeping the table structure intact.
- Example: Truncating the "employees" table to remove all employee records.
- sql
- Copy code

```
TRUNCATE TABLE
```

-

Explanation:

- `CREATE`: This command is used when you want to define a new database object, such as a table, and specify its structure, including columns, data types, and constraints.
- `DROP`: This command is used when you want to delete an existing database object, such as a table. It removes the entire table and its data from the database.
- `ALTER`: This command is used when you want to modify the structure of an existing database object. For example, you can use it to add or remove columns from a table.
- `TRUNCATE`: This command is used when you want to remove all rows from a table but keep the table structure. It is faster than `DELETE` for removing all records, as it doesn't log individual row deletions.

These DDL commands are powerful tools for database administrators and developers to define and manage the structure of a database according to the evolving needs of an application.

Q3. What is DML? Explain INSERT, UPDATE, and DELETE with an example.
Ans:DML (Data Manipulation Language):

DML, or Data Manipulation Language, is a subset of SQL (Structured Query Language) used for managing data stored in a database. DML consists of commands such as `INSERT`, `UPDATE`, and `DELETE`, which allow users to insert, modify, and delete data in database tables.

INSERT:

- The `INSERT` statement is used to add new rows of data into a table.
- Example: Inserting a new employee into the "employees" table.
- sql
- Copy code

```
INSERT INTO
VALUES  1  'John Doe'  50000.00  'HR'
```

-

UPDATE:

- The `UPDATE` statement is used to modify existing data in a table based on a specified condition.
- Example: Updating the salary of an employee with ID 1.
- sql
- Copy code

```
UPDATE
SET          55000.00
WHERE                1
```

- DELETE:
  - The `DELETE` statement is used to remove one or more rows from a table based on a specified condition.
  - Example: Deleting an employee with ID 1 from the "employees" table.
  - sql
  - Copy code

```
DELETE FROM
WHERE                1
```

- Explanation:

- `INSERT`: This command is used to add new records (rows) to a table. The `VALUES` clause specifies the values for each column in the new row.
- `UPDATE`: This command is used to modify existing records in a table. The `SET` clause specifies the columns to be modified and their new values, and the `WHERE` clause specifies the condition that determines which rows will be updated.
- `DELETE`: This command is used to remove one or more records from a table. The `WHERE` clause specifies the condition that determines which rows will be deleted. If no `WHERE` clause is provided, all rows in the table will be deleted.

These DML commands are essential for managing the content of a database, allowing users to insert new data, update existing data, and remove unwanted data based on specified conditions. They are commonly used in applications to interact with and manipulate the data stored in database tables.

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

AnsDQL (Data Query Language):

DQL, or Data Query Language, is a subset of SQL (Structured Query Language) used for retrieving and querying data from a database. The primary DQL command is `SELECT`, which is used to specify the columns to be retrieved and the conditions for selecting rows.

SELECT:

The `SELECT` statement is used to retrieve data from one or more tables. It allows you to specify the columns you want to retrieve, apply filters, and even join multiple tables.

Example:

Suppose we have a table named "employees" with columns: `employee_id`, `employee_name`, `salary`, and `department`. Here are some examples of `SELECT` statements:

Selecting All Columns:

- To retrieve all columns for all employees:
- sql
- Copy code

```
SELECT   FROM
```

-

Selecting Specific Columns:

- To retrieve only specific columns (e.g., `employee_id` and `employee_name`):
- sql
- Copy code

```
SELECT                          FROM
```

-

Filtering Rows with WHERE Clause:

- To retrieve employees with a salary greater than 50000:
- sql
- Copy code

```
SELECT   FROM
WHERE          50000
```

-

Sorting Results with ORDER BY:

- To retrieve employees sorted by salary in descending order:
- sql
- Copy code

```
SELECT   FROM
ORDER BY        DESC
```

-

Aggregating Data with GROUP BY:

- To calculate the average salary for each department:
- sql
- Copy code

```
SELECT             AVG         as
FROM
GROUP BY
```

-

Joining Tables:

- To retrieve data from two related tables (e.g., "employees" and "departments"):
- sql
- Copy code

```
SELECT
FROM
INNER JOIN             ON
```

-

These examples demonstrate the flexibility of the `SELECT` statement in retrieving data based on specified criteria, sorting results, aggregating information, and joining multiple tables. DQL is a crucial component of SQL, enabling users to extract meaningful information from a database.

Q5. Explain Primary Key and Foreign Key.
Ans:Primary Key:

A primary key is a field or a set of fields in a database table that uniquely identifies each record in the table. It serves as a unique identifier for the records and ensures that each record can be

distinctly identified. The primary key must have a unique value for each record and cannot contain a null (empty) value.

Key characteristics of a primary key:

- Uniqueness: Each value in the primary key must be unique across all records in the table.
- Non-nullability: A primary key cannot have a null value. Each record must have a valid and unique identifier.
- Permanence: The values in a primary key should not change over time. If changes are necessary, they should be handled carefully to maintain data integrity.
- Single or Composite: A primary key can consist of a single column or a combination of multiple columns. In the case of a combination, it's called a composite primary key.

Commonly, primary keys are used as the basis for creating relationships between tables in a relational database.

Foreign Key:

A foreign key is a field or a set of fields in a database table that refers to the primary key of another table. It establishes a link or relationship between two tables by referencing the primary key in one table to the foreign key in another. The table containing the foreign key is called the referencing or child table, and the table containing the primary key being referenced is called the referenced or parent table.

Key characteristics of a foreign key:

- Referential Integrity: Foreign keys enforce referential integrity, ensuring that relationships between tables are valid. This means that values in the foreign key must match values in the primary key of the referenced table, or be null.
- No Unique Constraint: While the values in a foreign key must match values in the referenced primary key, they are not required to be unique within the foreign key column.
- Can Be Null: Depending on the design, a foreign key can contain null values. This is useful when a relationship is optional.
- Cascade Actions: Foreign key relationships often support actions like CASCADE UPDATE or CASCADE DELETE, which automatically propagate changes or deletions from the parent table to the child table.

In summary, a primary key uniquely identifies records within a table, while a foreign key establishes a relationship between two tables by referencing the primary key of another table. The use of primary and foreign keys is fundamental to maintaining data integrity and establishing meaningful connections in relational database systems.

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

Ans:To connect Python to MySQL, you can use the `mysql-connector` library. If you haven't installed it yet, you can do so by running:

bash

Copy code

Here's a simple example of connecting to MySQL using Python and explaining the `cursor()` and `execute()` methods:

python

Copy code

```
import


      "your_mysql_host"
      "your_mysql_user"
          "your_mysql_password"
          "your_mysql_database"




try
                  "SELECT * FROM your_table_name"



 for     in
```

```
 print

except                        as
 print f"Error: {err}"

finally
```

Explanation:

Connecting to MySQL:

python

Copy code

```
"your_mysql_host"
"your_mysql_user"
    "your_mysql_password"
    "your_mysql_database"
```

Replace the placeholders (`your_mysql_host`, `your_mysql_user`, `your_mysql_password`, `your_mysql_database`) with your actual MySQL server details.
Creating a Cursor:

python

Copy code

The `cursor()` method is used to create a cursor object. A cursor is essential for interacting with the database. It allows you to execute SQL queries and fetch results.
Executing a Query:

python

Copy code

```
"SELECT * FROM your_table_name"
```

The `execute()` method is used to execute a SQL query. In this example, a simple SELECT query is executed to retrieve all rows from a table.
Fetching Results:

python

Copy code

The `fetchall()` method retrieves all the rows of a query result set and returns a list of tuples.
Displaying Results:

python

Copy code

```
for     in
 print
```

Here, we loop through the result set and print each row. Adjust the printing format based on your specific use case.
Handling Errors:

python

Copy code

```
except                      as
 print f"Error: {err}"
```

It's good practice to handle potential errors that might occur during the execution of the code.
Closing Cursor and Connection:

python

Copy code

```
finally
```

Always close the cursor and connection to release resources after using them.

Remember to replace the placeholder values with your actual MySQL server details and adjust the SQL query based on your needs.

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

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

SELECT: The `SELECT` clause is used to specify the columns that you want to retrieve from the database.
FROM: The `FROM` clause specifies the table or tables from which to retrieve the data specified in the `SELECT` clause.
WHERE: The `WHERE` clause is used to filter records based on a specified condition. It is applied to the rows retrieved from the `FROM` clause.
GROUP BY: The `GROUP BY` clause is used to group rows that have the same values in specified columns into summary rows, like "total" or "average."
HAVING: The `HAVING` clause is used to filter the results of a `GROUP BY` clause based on a specified condition.
ORDER BY: The `ORDER BY` clause is used to sort the result set based on one or more columns. It is applied to the result set after all other clauses have been executed.
LIMIT / OFFSET: The `LIMIT` clause is used to limit the number of rows returned in a result set, and the `OFFSET` clause is used to skip a specific number of rows.

It's important to note that not all queries include every clause, and the order in which clauses appear in a query depends on the specific requirements of the query. Additionally, some databases might have variations or extensions to the standard SQL syntax that can affect the order of execution. However, the general sequence outlined above is a common guideline for understanding the logical flow of an SQL query.