**16 feb MySQL Assignment**

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

 A database is a structured collection of data that is stored and organized in a way that allows for efficient retrieval, management, and updating of information. Databases are commonly used in a wide range of applications, from small-scale personal projects to large-scale enterprise systems.

There are two main types of databases:

SQL (Structured Query Language)

 NoSQL (Not only SQL).

Here are some differences between the two:

\* Data Model: SQL databases are based on the relational data model, where data is stored in tables with predefined relationships between them. NoSQL databases, on the other hand, use a variety of data models, including document, key-value, column-family, and graphi,image,video.

\* Schema: SQL databases have a rigid schema that defines the structure of the data in advance. This makes it easy to maintain consistency and enforce data integrity, but can also make it difficult to adapt to changing requirements. NoSQL databases, on the other hand, have a flexible schema that allows for dynamic changes to the data structure.

\* Scalability: SQL databases are vertically scalable, meaning that they can be scaled up by adding more resources to a single server. NoSQL databases, on the other hand, are horizontally scalable, meaning that they can be scaled out by adding more servers to a cluster.

\* Query Language: SQL databases use the SQL query language to manipulate data, which is a standard language used across all relational databases. NoSQL databases use their own query languages, which can vary between different types of NoSQL databases.

\* ACID compliance: SQL databases are typically ACID-compliant, meaning that they guarantee atomicity, consistency, isolation, and durability of transactions. NoSQL databases, on the other hand, are often designed to sacrifice some of these properties in order to achieve higher performance and scalability.

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

DDL (Data Definition Language) is a set of SQL commands used to define and manage the structure of a database.

Here's an explanation of each of the four main DDL commands and why they are used:

1. CREATE: This command is used to create new database objects, such as tables, views, indexes, and constraints. For example, the following SQL statement creates a new table named "my\_table" with columns for ID, First name, and Last Name:

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

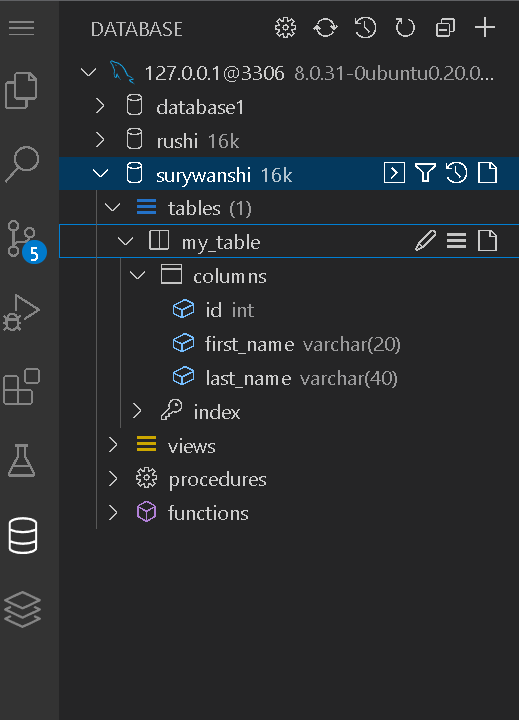
)

mycursor = mydb.cursor()

mycursor.execute("create database if not exists surywanshi")

mycursor.execute("CREATE TABLE if not exists surywanshi.my\_table(id INT, first\_name VARCHAR(20),last\_name VARCHAR(40) );")

mydb.close()



1. DROP: This command is used to delete existing database objects, such as tables, views, indexes, and constraints.
2. import mysql.connector
3. mydb = mysql.connector.connect(
4. host="localhost",
5. user="abc",
6. password="password"
7. )
8. mycursor = mydb.cursor()
9. mycursor.execute("DROP TABLE surywanshi.my\_table")
10. mydb.close()

Text

Description automatically generated with medium confidence

1. ALTER: This command is used to modify the structure of existing database objects, such as tables and views.

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE surywanshi.my\_table ADD COLUMN age INT")

mydb.close()

Text

Description automatically generated

1. TRUNCATE: This command is used to remove all data from a table, but keep the table structure intact. This is faster and more efficient than using the DELETE command to remove rows one by one.

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

mycursor = mydb.cursor()

mycursor.execute("truncate TABLE surywanshi.my\_table")

mydb.close()

Text

Description automatically generated

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

DML (Data Manipulation Language) is a set of SQL commands used to manipulate the data within a database. Here's an explanation of three common DML commands and examples of how they are used:

1. INSERT: This command is used to insert new data into a table

CODE

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

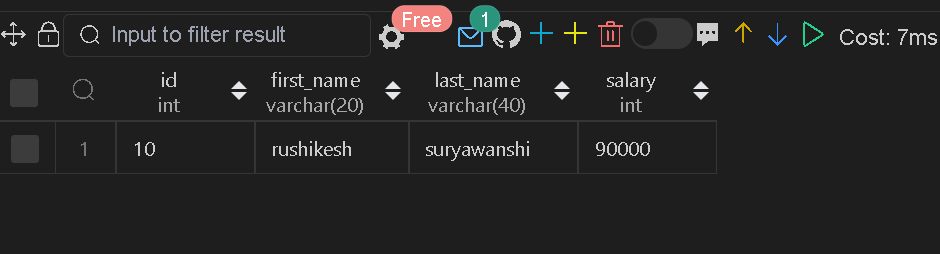
mycursor = mydb.cursor()

mycursor.execute("insert into db1.employee values(10 , 'rushikesh','suryawanshi',90000)")

mydb.commit()

mydb.close()

OUTPUT



1. UPDATE: This command is used to modify existing data in a table.

CODE

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

mycursor = mydb.cursor()

# mycursor.execute("create database if not exists surywanshi")

mycursor.execute("UPDATE db1.employee SET salary = 69999 WHERE id = 11;")

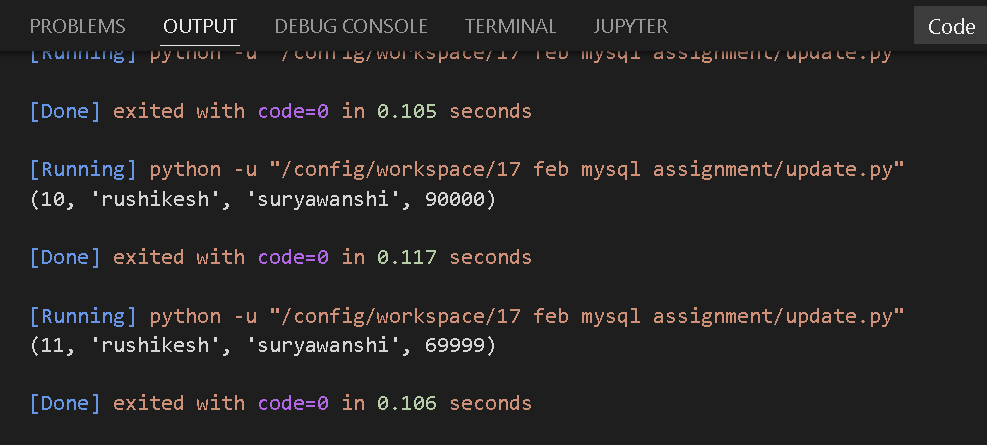
mycursor.execute("select \* from db1.employee")

for x in mycursor:

  print(x)

mydb.close()

OUTPUT



1. DELETE: This command is used to remove data from a table

CODE

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

mycursor = mydb.cursor()

# mycursor.execute("create database if not exists surywanshi")

mycursor.execute("DELETE FROM db1.employee where id = 1")

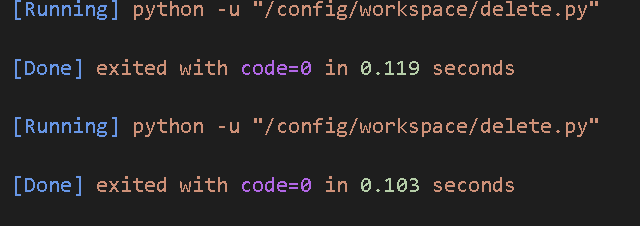
mycursor.execute("select \* from surywanshi.my\_table")

for x in mycursor:

  print(x)

mydb.close()

OUTPUT



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

DQL (Data Query Language) is a set of SQL commands used to retrieve data from a database. Here's an explanation of the main DQL command and an example of how it is used:

SELECT: This command is used to retrieve data from one or more tables in a database. For example, the following SQL statement retrieves all data from the "employees" table:

Code 1:

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

mycursor = mydb.cursor()

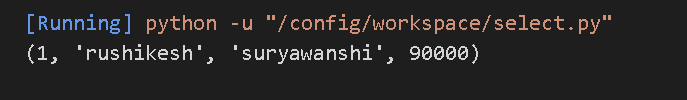
mycursor.execute("select \* from db1.employee")

for x in mycursor:

  print(x)

mydb.close()

OUTPUT



Code 2:

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

mycursor = mydb.cursor()

mycursor.execute("SELECT last\_name, salary FROM db1.employee WHERE salary > 50000;")

for x in mycursor:

  print(x)

mydb.close()

Text

Description automatically generated

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

* A Primary Key is a column or set of columns in a table that uniquely identifies each row. It must contain unique values and cannot contain NULL values. Typically, Primary Keys are used as a reference in other tables to create relationships between them. For example, in a database of employees and departments, the employee ID column might be used as the Primary Key in the employee table to ensure that each employee has a unique identifier.
* A Foreign Key is a column or set of columns in a table that refers to the Primary Key of another table. It creates a link between two tables by referencing a column that contains unique values. Foreign Keys ensure that the data in one table is related to the data in another table. For example, in the same employee and department database, the department ID column might be used as the Foreign Key in the employee table to establish a relationship between the employee and department tables.
* To establish a relationship between tables using Primary and Foreign Keys, the Foreign Key column in one table must reference the Primary Key column in another table. This creates a constraint that ensures that only valid values are allowed in the Foreign Key column. For example, if an employee record includes a department ID that does not exist in the department table, the database will prevent the record from being added or updated, ensuring the integrity of the relationship between the two tables.

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

Here's an example Python code to connect MySQL to Python using the mysql-connector module:

import mysql.connector

mydb = mysql.connector.connect(

  host="localhost",

  user="abc",

  password="password"

)

mycursor = mydb.cursor()

mycursor.execute("create database if not exists rushi")

mycursor.execute("CREATE TABLE if not exists rushi.my\_table(c1 INT, c2 VARCHAR(20),c3 INT, c4 FLOAT,c5 VARCHAR(40) );")

# create a cursor object

mycursor = mydb.cursor()

# execute a SQL query

mycursor.execute("SELECT \* FROM db1.employee")

# retrieve the results

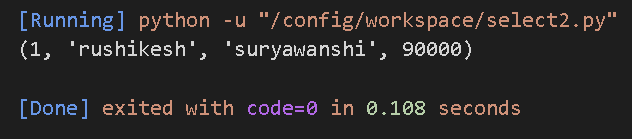
result = mycursor.fetchall()

# print the results

for row in result:

  print(row)

mydb.close()



The cursor() method creates a cursor object, which is used to execute SQL queries and retrieve the results. It provides methods such as execute(), fetchone(), and fetchall() to interact with the database.

The execute() method is used to execute a SQL query. It takes a string as a parameter, which should be a valid SQL statement. The method returns None and does not retrieve any results. To retrieve the results of a query, we must use the fetchone() or fetchall() method on the cursor object.

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

In an SQL query, the order of execution of SQL clauses is as follows:

1. FROM: This clause specifies the tables from which the data should be selected. It is the first clause that is executed.
2. WHERE: This clause filters the data based on a specified condition. It is executed after the FROM clause.
3. GROUP BY: This clause groups the data based on a specified column or set of columns. It is executed after the WHERE clause.
4. HAVING: This clause filters the grouped data based on a specified condition. It is executed after the GROUP BY clause.
5. SELECT: This clause selects the columns that should be returned in the result set. It is executed after the HAVING clause.
6. ORDER BY: This clause sorts the data based on a specified column or set of columns. It is executed after the SELECT clause.
7. LIMIT: This clause limits the number of rows returned in the result set. It is executed last, after all other clauses have been executed.