MYSQL

DATABASE MANAGEMENT SYSTEM(DBMS)

- A database is a systematic way of storing information so data can be accessed, analyzed, transformed, updated and moved with efficiency.
- Organized in tables having rows and columns.
- A database management system (DBMS) is a software package we use to create and manage databases.
  - In other words, a DBMS is a user interface (UI) that makes it possible for users to actually interact with the database.

- SQL(Structured Query Language) is the language to perform operations on a DB.
- SQL commands are
  - DDL, SQL commands that can be used to define the database schema, not the data.
    - CREARE,DROP,ALTER,TRUNCATE,COMMENT,RENAME
  - DML, SQL commands that deal with the manipulation of data present in the database
    - INSERT,UPDATE,DELETE
  - DCL, includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.
    - GRANT,REVOKE

Python and MySQL

- Python can be used in database applications.
- One of the most popular databases is MySQL.
- To install MySQL:
  - download a MySQL database at https://www.mysql.com/downloads/.
  - Download and install "MySQL Connector".(MySQL driver to access the DB) using:

    python -m pip install mysql-connector-python
  - Test the connectivity, for successful installation, no errors on execution

    import mysql.connector

Create Connection

- import mysql.connector

```
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword“,database=“dbna
me”
)

print(mydb)
```

Creating a Database

- import mysql.connector

```
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)

mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE
mydatabase")
```

Python Mysql Connector Module Methods

1. connect(): This function is used for establishing a connection with the MySQL server. The following are the arguments that are used to initiate a connection:

- **user:** User name associated with the MySQL server used to authenticate the connection
- **password:** Password associated with the user name for authentication
- **database:** Data base in the MySQL for creating the Table

2. cursor(): MySQL cursor class instantiates objects (mycursor) that can execute operations such as SQL statements .Cursor is the workspace created in the system memory when the SQL command is executed. This memory is temporary and the cursor connection is bounded for the entire session/lifetime and the commands are executed

3. execute(): The execute function takes a SQL query as an argument and executes. A query is an SQL command which is used to create, insert, retrieve, update, delete etc.

Creating Tables

……

…..

```
mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customer
(name VARCHAR(255), address
VARCHAR(255))")
```

- mycursor.execute(“ ") //check the existence of tables.
- mycursor.execute("**CREATE TABLE** customers (**id INT AUTO_INCREMENT PRIMARY KEY**, name VARCHAR(255), address VARCHAR(255))")
  - Set the column **id** as primary key, which holds unique values.
- mycursor.execute("**ALTER TABLE** customers **ADD COLUMN** dept VARCHAR(255)") //

- Insert Into Table:
  sql = "**INSERT INTO** customers (name, address) **VALUES** (%s, %s)"
  val = ("John", "Highway 21")
  mycursor.**execute**(**sql, val**)
  mydb.commit()
  print(mycursor.rowcount, "record inserted.")
- **mydb.commit()** is required to make the changes, otherwise no changes are made to the table.
- OUTPUT: 1 row inserted

- Insert many rows to a table:

……(sql connection)…

sql = "**INSERT INTO** customers (name, address)
VALUES (%s, %s)"
val = [('Peter', ‘Addr1'),
  ('Amy', ’Addr2'),
  ('Hannah', ’Addr3’)]
mycursor.executemany(sql, val)
mydb.commit()
print(mycursor.rowcount, "was inserted.")

OUTPUT: 3 rows inserted

- Update Table

sql = "**UPDATE** customers **SET** address = 'Canyon 123' **WHERE** address = 'Valley 345'"
mycursor.execute(sql)
mydb.commit()

- WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

- Select From a Table

mycursor.execute("**SELECT** * FROM customers")// select all rows
myresult = mycursor.**fetchall**() // fetches all rows from the last executed statement.

- fetchone() Method: select a single row

mycursor.execute("SELECT * FROM customers")
myresult = mycursor.**fetchone()**

- Selected columns:

mycursor.execute("SELECT **name,dept** FROM customers")
myresult = mycursor.fetchall()

- Select With a Filter / Condition:

```
sql = "SELECT * FROM customers WHERE address
='Park Lane 38'"
mycursor.execute(sql)
myresult = mycursor.fetchall()
```

- Select records with similar words:

```
sql = "SELECT * FROM customers WHERE address
LIKE '%way%'"
mycursor.execute(sql)
myresult = mycursor.fetchall()
```

- Sort the Result:

```
sql = "SELECT * FROM customers ORDER BY name"
mycursor.execute(sql)
myresult = mycursor.fetchall()
```

- sql = "SELECT * FROM customers **ORDER BY** name **DESC**"

  ```
  mycursor.execute(sql)
  myresult = mycursor.fetchall()
  ```

- Delete Record

sql = "**DELETE FROM** customers **WHERE** address = 'Mountain 21'"
mycursor.execute(sql)
mydb.commit()

- mydb.commit() is required to make the changes, otherwise no changes are made to the table.
- WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted.

- Delete a Table

sql = "**DROP TABLE** customers"
mycursor.execute(sql)

- To avoid the error on deleting an already deleted table,

sql = "DROP TABLE **IF EXISTS** customers"
mycursor.execute(sql)

Data visualization

- An easier way of presenting the data, however complex it is, to analyze trends and relationships amongst variables with the help of pictorial representation.
- Easier representation of compels data
- Highlights good and bad performing areas
- Explores relationship between data points
- Identifies data patterns even for larger data points

- Python Libraries: ***matplotlib***, *vispy, bokeh,* ***seaborn***, *pygal, folium, plotly, cufflinks*, and *networkx* etc.
  - Boxplot
  - Histogram
  - Pie chart
  - Scatter plot
  - Correlation plot

```
import matplotlib.pyplot as plt
 # Creating dataset
cars = ['AUDI', 'BMW', 'FORD', 'TESLA', 'JAGUAR', 'MERCEDES']
data = [23, 17, 35, 29, 12, 41]

# Creating plot
fig = plt.figure(figsize=(10, 7))
plt.pie(data, labels=cars)

# Show plot
plt.show()
```