In [2]:
```
pip install mysql-connector-python
```

```
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.0.0-cp39-cp39-win_amd64.whl (14.3 MB)
     ---------------------------------------- 14.3/14.3 MB 4.9 MB/s eta 0:0
0:00
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.0.0
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import mysql.connector
from mysql.connector import errorcode
```

In [4]:
```
import pandas as pd
```

In [7]:
```
mydb= mysql.connector.connect(
      host='localhost',
      user='root',
      password='SiKu17@26R'
)
```

In [8]:
```
# Check if the connection was successful
if mydb.is_connected():
    print("Connected to the database")
```

```
Connected to the database
```

create database

In [19]:
```
cur=mydb.cursor()
cur.execute("create database db2")
```

CREATE TABLE

In [20]:
```
mydb= mysql.connector.connect(
      host='localhost',
      user='root',
      password='SiKu17@26R',
      database='db2'
)

cur = mydb.cursor()
```

In [21]:
```
s="CREATE TABLE Book(bookid integer(6), title varchar(20), price float(5,2))
cur.execute(s)
```

for checking the database if its create or not and also for checking the table of the database is creates or not use= use database-name; then show tables;

In [23]:
```
s="INSERT INTO BOOK(bookid, title, price) values(%s, %s, %s)"
b1=(1, 'gods palne', 123)
cur.execute(s,b1)
mydb.commit()
```

INSERT MULTIPLE RECORDES

In [25]:
```
s="INSERT INTO BOOK(bookid, title, price) values(%s, %s, %s)"
books=[(2,'php',300),(3,'math',400 )]
cur.executemany(s,books)
mydb.commit()
```

Selecting Data

In [32]:
```
cur.execute("select * from book")
result=cur.fetchall()

for row in result:
    print(row)
```

```
(2, 'php', 23.0)
(3, 'math', 400.0)
```

fetchall(): Retrieves all rows that satisfy the query and returns them as a list of tuples.

Updating Data

In [27]:
```
cur.execute("update book set price=23 where bookid= 2")
mydb.commit()
```

Deleting Data

In [29]:
```
cur.execute("delete from book where bookid= 1")
mydb.commit()
```

Prepared Statements

In [31]:
```
s="select * from book where price> %s"
val=(20, )
cur.execute(s,val)
rows=cur.fetchall()
```

%s is a placeholder (similar to a parameter) that will be replaced by the actual value when the query is executed.

This clause filters the results to include only rows where the value of the price column is greater than the specified value.

The %s is a placeholder for the actual value that will be passed dynamically.

The comma in (20,) ensures that this is a tuple, even if it has only one element. Tuples in Python require a trailing comma when they have only one element.

(20,): The tuple contains the number 20, meaning we want to select all rows where the price column is greater than 20.

In [ ]:
```

```

Error Handling

In [33]:
```
import mysql.connector
from mysql.connector import Error

try:
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="SiKu17@26R"
    )
except Error as e:
    print(f"Error connecting to database: {e}")
finally:
    if mydb.is_connected():
        cur.close()
        mydb.close()
```

connection closed

In [34]:
```
cur.close()
mydb.close()
```

In [ ]:
```

```

In [1]:
```
import mysql.connector
from mysql.connector import errorcode
```

In [2]:
```
import pandas as pd
```

In [3]:
```
from mysql.connector import Error
```

In [4]:
```
def create_server_connection(host_name, user_name, user_password):
    connection=None
    try:
        connection=mysql.connector.connect(
         host=host_name,  # The address of the MySQL server (usually "localh
        user=user_name,  #the username to log into the mysql server (eg. 'ro
        password=user_password  # The password for the specified user by the
        )
        print("Mysql database connection succesfully:")
    except Error as err:
        print(f"Error: {err}")    # If any error occurs during connection, i
    return connection

#put our mysql connection password

pw= 'SiKu17@26R'

#database name
db="sidd320"

connection=create_server_connection("localhost", "root", pw)
```

```
Mysql database connection succesfully:
```

In [5]:
```
#create mysql_python

def Create_database(connection, quary):
    cursor=connection.cursor()
    try:
        cursor.execute(quary)
        print("database create succesfully")
    except Error as err:
         print(f"Error: {err}")
Create_database_quary="Create database mysql_pythonwala"
Create_database(connection,Create_database_quary )
```

```
Error: 1007 (HY000): Can't create database 'mysql_pythonwala'; database exis
ts
```

CONNECT DATABASE TO SERVER

In [6]:
```
def create_db_connection(host_name,user_name, user_password, db_name):
    connection= None
    try:
        connection=mysql.connector.connect(
```

```
        host=host_name,
        user=user_name,
        passwd=user_password,
        database=db_name)
        print("mysql database connection successfully")
    except Error as err:
         print(f"Error: {err}")
    return connection
```

EXECUTE SQL QUARIES

In [7]:
```
def execute_quary(connection, quary):
    cursor=connection.cursor()
    try:
        cursor.execute(quary)
        connection.commit()
        print("quary execute successfully")
    except Error as err:
            print(f"Error: {err}")
```

In [8]:
```
create_washer_table= """
create table washer(
order_id int primary key,
customer_name varchar(30) not null,
product_name varchar(20) not null,
date_ordered date,
quantity int,
unit_price float,
phone_number varchar(20))
"""
```

In [9]:
```
# Connect to the database
db = "mysql_python"  # Replace with your database name
pw = "SiKu17@26R"       # Replace with your MySQL root password
connection = create_db_connection("localhost", "root", pw, db)
```

```
mysql database connection successfully
```

In [10]:
```
# Execute the query to create the oluu table
if connection:
    execute_quary(connection, create_washer_table)
else:
    print("Failed to connect to the database")
```

```
Error: 1050 (42S01): Table 'washer' already exists
```

INSERT DATA

In [11]:
```
data_washer="""
insert into washer table
(101, 'steve','Laptop','2008-06-12',2,800,'6003489678')
(102, 'bhalu','Mouse','2008-05-15',5,12,'6003484568')
```

```
(103, 'lalu','keyboard','2008-04-22',6,50,'6003481238')
"""
```

In [12]:
```
data_washer = """
INSERT INTO washer (order_id, customer_name, product_name, date_ordered, qua
VALUES
(101, 'Steve', 'Laptop', '2008-06-12', 2,800, '6003489678'),
(102, 'Bhalu', 'Mouse', '2008-05-15', 10,12, '6003484568'),
(103, 'Lalu', 'Keyboard', '2008-04-22', 5,50, '6003481238');
"""
```

In [13]:
```
# Connect to the database
db = "mysql_python"  # Replace with your database name
pw = "SiKu17@26R"       # Replace with your MySQL root password
connection = create_db_connection("localhost", "root", pw, db)
```

```
mysql database connection successfully
```

In [14]:
```
# Execute the insert query
if connection:
    execute_quary(connection, data_washer)
else:
    print("Failed to connect to the database")
```

```
Error: 1062 (23000): Duplicate entry '101' for key 'washer.PRIMARY'
```

In [15]:
```
def read_quary(connection, quary):
    cursor=connection.cursor()
    result=None
    try:
        cursor.execute(quary)
        result=cursor.fetchall()
        return result
    except Error as err:
        print(f"Error: {err}")
```

In [16]:
```
#using the select statament
q1= """

select * from washer;
"""
connection = create_db_connection("localhost", "root", pw, db)
results= read_quary(connection, q1)
```

```
mysql database connection successfully
```

In [17]:
```
# Check if results are not None, then print each result
if results:
    for result in results:
        print(result)
else:
    print("No results found or an error occurred.")
```

```
(101, 'Steve', 'Laptop', datetime.date(2008, 6, 12), 2, 800.0, '6003489678')
(102, 'Bhalu', 'Mouse', datetime.date(2008, 5, 15), 10, 12.0, '6003484568')
(103, 'Lalu', 'Keyboard', datetime.date(2008, 4, 22), 5, 145.0, '600348123
8')
(104, 'Mona', 'Headphones', datetime.date(2023, 9, 20), 3, 300.0, '987654321
0')
(105, 'John', 'Speakers', datetime.date(2023, 9, 21), 1, 450.0, '987654321
1')
```

In [18]:
```python
#using the select statament
q2= """

select customer_name,product_name from washer;
"""
connection = create_db_connection("localhost", "root", pw, db)
results= read_quary(connection, q2)
```

```
mysql database connection successfully
```

In [19]:
```python
# Check if results are not None, then print each result
if results:
    for result in results:
        print(result)
else:
    print("No results found or an error occurred.")
```

```
('Steve', 'Laptop')
('Bhalu', 'Mouse')
('Lalu', 'Keyboard')
('Mona', 'Headphones')
('John', 'Speakers')
```

In [20]:
```python
#using the select statament
q3= """

select distinct year(date_ordered) from washer;
"""
connection = create_db_connection("localhost", "root", pw, db)
results= read_quary(connection, q3)
```

```
mysql database connection successfully
```

In [21]:
```python
# Check if results are not None, then print each result
if results:
    for result in results:
        print(result)
else:
    print("No results found or an error occurred.")
```

```
(2008,)
(2023,)
```

In [22]:
```python
#using the select statament
q5= """

select * from washer order by unit_price;
"""
```

```
connection = create_db_connection("localhost", "root", pw, db)
results= read_quary(connection, q5)
```

```
mysql database connection successfully
```

In [23]:
```
# Check if results are not None, then print each result
if results:
    for result in results:
        print(result)
else:
    print("No results found or an error occurred.")
```

```
(102, 'Bhalu', 'Mouse', datetime.date(2008, 5, 15), 10, 12.0, '6003484568')
(103, 'Lalu', 'Keyboard', datetime.date(2008, 4, 22), 5, 145.0, '600348123
8')
(104, 'Mona', 'Headphones', datetime.date(2023, 9, 20), 3, 300.0, '987654321
0')
(105, 'John', 'Speakers', datetime.date(2023, 9, 21), 1, 450.0, '987654321
1')
(101, 'Steve', 'Laptop', datetime.date(2008, 6, 12), 2, 800.0, '6003489678')
```

In [24]:
```
#using the select statament
q6= """

select * from washer where order_id=103;
"""
connection = create_db_connection("localhost", "root", pw, db)
results= read_quary(connection, q6)
```

```
mysql database connection successfully
```

In [25]:
```
# Check if results are not None, then print each result
if results:
    for result in results:
        print(result)
else:
    print("No results found or an error occurred.")
```

```
(103, 'Lalu', 'Keyboard', datetime.date(2008, 4, 22), 5, 145.0, '600348123
8')
```

CREATE A DATAFRAME

In [33]:
```
from_db= []

for result in results:
    result=list(result)
    from_db.append(result)

columns=["order_id",
"customer_name",
"product_name",
"date_ordered",
"quantity",
"unit_price",
"phone_number"]
```

```
df=pd.DataFrame(from_db, columns=columns)

display(df)
```

	order_id	customer_name	product_name	date_ordered	quantity	unit_price
0	101	Steve	Laptop	2008-06-12	2	800.0
1	102	Bhalu	Mouse	2008-05-15	10	12.0
2	103	Lalu	Keyboard	2008-04-22	5	145.0
3	104	Mona	Headphones	2023-09-20	3	300.0
4	105	John	Speakers	2023-09-21	1	450.0

UPDATE COMMAND

In [27]:
```
update="""

update washer
set unit_price= 145
where order_id=103
"""
connection = create_db_connection("localhost", "root", pw, db)
execute_quary(connection, update)
```

```
mysql database connection successfully
quary execute successfully
```

ANOTHER FORMNAT TO CREAT A DTAFRAME USING THIS

In [28]:
```
import pandas as pd
df = pd.read_sql("SELECT * FROM  washer", connection)
print(df)
```

```
   order_id customer_name product_name date_ordered  quantity  unit_price  \
0       101         Steve       Laptop   2008-06-12         2       800.0
1       102         Bhalu        Mouse   2008-05-15        10        12.0
2       103          Lalu     Keyboard   2008-04-22         5       145.0
3       104          Mona   Headphones   2023-09-20         3       300.0
4       105          John     Speakers   2023-09-21         1       450.0

  phone_number
0   6003489678
1   6003484568
2   6003481238
3   9876543210
4   9876543211
```

```
C:\Users\shaw3\anaconda3\lib\site-packages\pandas\io\sql.py:762: UserWarnin
g: pandas only support SQLAlchemy connectable(engine/connection) ordatabase
string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested,
please consider using SQLAlchemy
  warnings.warn(
```

DELETE TABLE

In [29]:
```
#delete_order_query = """
#DELETE FROM washer
#WHERE order_id = 102;
#"""

#execute_query(connection, delete_order_query)
```

Batch Insertions

In [30]:
```
def execute_quary(quary):
    cursor = connection.cursor()  # Cursor should be created here
    cursor.execute(quary)
    return cursor.fetchall()

# Call the function and pass the query
quary = "SELECT * FROM washer"
results = execute_quary(quary)

for row in results:
    print(row)
```

```
(101, 'Steve', 'Laptop', datetime.date(2008, 6, 12), 2, 800.0, '6003489678')
(102, 'Bhalu', 'Mouse', datetime.date(2008, 5, 15), 10, 12.0, '6003484568')
(103, 'Lalu', 'Keyboard', datetime.date(2008, 4, 22), 5, 145.0, '600348123
8')
(104, 'Mona', 'Headphones', datetime.date(2023, 9, 20), 3, 300.0, '987654321
0')
(105, 'John', 'Speakers', datetime.date(2023, 9, 21), 1, 450.0, '987654321
1')
```

In [31]:
```
data = [
    (104, 'Mona', 'Headphones', '2023-09-20', 3, 300.00, '9876543210'),
    (105, 'John', 'Speakers', '2023-09-21', 1, 450.00, '9876543211')
]

insert_many_quary = """
INSERT INTO washer (order_id, customer_name, product_name, date_ordered, qua
VALUES (%s, %s, %s, %s, %s, %s, %s)
"""
```

In [32]:
```
# Step 3: Check if the connection is successful and create a cursor
if connection:
    try:
        # Create a cursor object
```

```
        cursor = connection.cursor()

        # Step 4: Execute the `executemany()` function to insert multiple ro
        cursor.executemany(insert_many_quary, data)

        # Step 5: Commit the changes to save them to the database
        connection.commit()
        print("Data inserted successfully.")

    except Error as err:
        print(f"Error: '{err}'")

    finally:
        # Close the cursor and connection
        cursor.close()
        connection.close()
else:
    print("Failed to connect to the database.")
```

```
Error: '1062 (23000): Duplicate entry '104' for key 'washer.PRIMARY''
```

In [ ]:

Handling Large Result Sets

In [ ]:
```
query = "SELECT * FROM large_table"
cursor.execute(query)

while True:
    rows = cursor.fetchmany(100)  # Fetch 100 rows at a time
    if not rows:
        break
    for row in rows:
        print(row)
```

This approach reduces memory usage when working with large datasets.

In [ ]:

In [ ]: