CREATE CONNECTION

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


mydb=mysql.connector.connect(

    host='localhost',
    user="root",
    password="Kanha@8144",


)

cursor=mydb.cursor()
```

In [2]:
```
cursor.execute("SHOW DATABASES")
cursor.fetchall()
```

Out[2]:
```
[('imdb',),
 ('information_schema',),
 ('mysql',),
 ('performance_schema',),
 ('practiceDB',),
 ('sys',),
 ('testDB',),
 ('walmartsales',)]
```

In [4]:
```
cursor.execute("USE PracticeDB")
```

In [5]:
```
cursor.execute("SHOW TABLES")
cursor.fetchall()
```

Out[5]:
```
[('customers',),
 ('departments',),
 ('employees',),
 ('orders',),
 ('student',),
 ('table1',),
 ('table2',)]
```

CRUD Operation in Python using MySQL

In this article, we will be seeing how to perform CRUD (CREATE, READ, UPDATE and DELETE) operations in Python using MySQL.

CREATING A TABLE

In [7]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)


cursor = mydb.cursor()


query="""
CREATE TABLE studentss (
    name VARCHAR(20) NOT NULL,
    branch VARCHAR(50),
    roll INT NOT NULL PRIMARY KEY,
    section VARCHAR(10),
    age INT
)
"""
cursor.execute(query)
```

INSERT INTO TABLES

Inserting single row

In [8]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)


cursor = mydb.cursor()



query="""
INSERT INTO studentss (name,branch,roll,section,age)
VALUES (%s,%s,%s,%s,%s)
"""

val= ("Ram", "CSE", 85, "B", 19)
cursor.execute(query,val)
mydb.commit()
```

In [9]:
```
cursor.execute("SELECT * FROM studentss")
cursor.fetchall()
```

Out[9]:
```
[('Amit', 'ECE', 24, 'A', 21),
 ('Rohan', 'MAE', 43, 'B', 20),
 ('Anil', 'MAE', 45, 'B', 20),
 ('Megha', 'ECE', 55, 'A', 22),
 ('Ram', 'CSE', 85, 'B', 19),
 ('Sita', 'CSE', 95, 'A', 19),
 ('Nikhil', 'CSE', 98, 'A', 18),
 ('Nisha', 'CSE', 99, 'A', 18)]
```

Inserting multipe rows

In [6]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()

query = """
INSERT INTO studentss (name, branch, roll, section, age)
VALUES (%s, %s, %s, %s, %s)
"""

val = [
    ("Nikhil", "CSE", 98, "A", 18),
    ("Nisha", "CSE", 99, "A", 18),
    ("Rohan", "MAE", 43, "B", 20),
    ("Amit", "ECE", 24, "A", 21),
    ("Anil", "MAE", 45, "B", 20),
    ("Megha", "ECE", 55, "A", 22),
    ("Sita", "CSE", 95, "A", 19)
]

cursor.executemany(query, val)  # Use 'executemany' to insert multiple rows
mydb.commit()  # Commit the changes to the database
```

In [10]:
```
cursor.execute("SELECT * FROM studentss")
cursor.fetchall()
```

Out[10]:
```
[('Amit', 'ECE', 24, 'A', 21),
 ('Rohan', 'MAE', 43, 'B', 20),
 ('Anil', 'MAE', 45, 'B', 20),
 ('Megha', 'ECE', 55, 'A', 22),
 ('Ram', 'CSE', 85, 'B', 19),
 ('Sita', 'CSE', 95, 'A', 19),
 ('Nikhil', 'CSE', 98, 'A', 18),
 ('Nisha', 'CSE', 99, 'A', 18)]
```

FETCHING DATA

In [12]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()
```

```
query=" SELECT name,roll FROM studentss "

cursor.execute(query)
result=cursor.fetchall()

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

```
('Amit', 24)
('Rohan', 43)
('Anil', 45)
('Megha', 55)
('Ram', 85)
('Sita', 95)
('Nikhil', 98)
('Nisha', 99)
```

In [ ]:

In [ ]:

WHERE Clause

```
The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified
condition.
```

In [13]:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()

query=" SELECT * FROM student "

cursor.execute(query)
cursor.fetchall()
```

Out[13]:

```
[(1, 'prasad', 'ctc', 40),
 (2, 'prasad', 'ctc', 40),
 (3, 'Peter', 'Lowstreet 4', 40),
 (4, 'Amy', 'Apple st 652', 60),
 (5, 'Hannah', 'Mountain 21', 90),
 (6, 'Michael', 'Valley 345', 20),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (8, 'Betty', 'Green Grass 1', 90),
 (9, 'Richard', 'Sky st 331', 45),
 (10, 'Susan', 'One way 98', 86),
 (11, 'Vicky', 'Yellow Garden 2', 93),
 (12, 'Ben', 'Park Lane 38', 48),
 (13, 'William', 'Central st 954', 89),
 (14, 'Chuck', 'Main Road 989', 67),
 (15, 'Viola', 'Sideway 1633', 93)]
```

In [35]:

```
query="SELECT * FROM student WHERE id=4 "
cursor.execute(query)
cursor.fetchall()
```

Out[35]:
```
[(4, 'Amy', 'Apple st 652', 60)]
```

In [37]:
```
query="SELECT * FROM student WHERE id BETWEEN 5 AND 10 "
cursor.execute(query)
cursor.fetchall()
```

Out[37]:
```
[(5, 'Hannah', 'Mountain 21', 90),
 (6, 'Michael', 'Valley 345', 20),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (8, 'Betty', 'Green Grass 1', 90),
 (9, 'Richard', 'Sky st 331', 45),
 (10, 'Susan', 'One way 98', 86)]
```

In [43]:
```
query="SELECT * FROM student WHERE Name LIKE '%dy' "
cursor.execute(query)
cursor.fetchall()
```

Out[43]:
```
[(7, 'Sandy', 'Ocean blvd 2', 80)]
```

In [53]:
```
query="SELECT * FROM student WHERE id>3 AND Mark >= 40 "
cursor.execute(query)
cursor.fetchall()
```

Out[53]:
```
[(4, 'Amy', 'Apple st 652', 60),
 (5, 'Hannah', 'Mountain 21', 90),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (8, 'Betty', 'Green Grass 1', 90),
 (9, 'Richard', 'Sky st 331', 45),
 (10, 'Susan', 'One way 98', 86),
 (11, 'Vicky', 'Yellow Garden 2', 93),
 (12, 'Ben', 'Park Lane 38', 48),
 (13, 'William', 'Central st 954', 89),
 (14, 'Chuck', 'Main Road 989', 67),
 (15, 'Viola', 'Sideway 1633', 93)]
```

In [44]:
```
query="SELECT * FROM student WHERE id IN (1,2,3,4) "
cursor.execute(query)
cursor.fetchall()
```

Out[44]:
```
[(1, 'prasad', 'ctc', 40),
 (2, 'prasad', 'ctc', 40),
 (3, 'Peter', 'Lowstreet 4', 40),
 (4, 'Amy', 'Apple st 652', 60)]
```

In [54]:
```
query="SELECT * FROM student WHERE id>3 AND Mark IN (30,40,50) "
cursor.execute(query)
cursor.fetchall()
```

Out[54]:
```
[]
```

ORDER BY clause

OrderBy is used to arrange the result set in either ascending or descending order. By default, it is always in ascending order unless “DESC” is mentioned, which arranges it in descending order. “ASC” can also be used to explicitly arrange it in ascending order. But, it is generally not done this way since default already does that.

Syntax:

SELECT column1, column2

FROM table_name

ORDER BY column_name ASC|DESC;

In [46]:
```
query="SELECT * FROM student ORDER BY Mark DESC "
cursor.execute(query)
cursor.fetchall()
```

Out[46]:
```
[(11, 'Vicky', 'Yellow Garden 2', 93),
 (15, 'Viola', 'Sideway 1633', 93),
 (5, 'Hannah', 'Mountain 21', 90),
 (8, 'Betty', 'Green Grass 1', 90),
 (13, 'William', 'Central st 954', 89),
 (10, 'Susan', 'One way 98', 86),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (14, 'Chuck', 'Main Road 989', 67),
 (4, 'Amy', 'Apple st 652', 60),
 (12, 'Ben', 'Park Lane 38', 48),
 (9, 'Richard', 'Sky st 331', 45),
 (1, 'prasad', 'ctc', 40),
 (2, 'prasad', 'ctc', 40),
 (3, 'Peter', 'Lowstreet 4', 40),
 (6, 'Michael', 'Valley 345', 20)]
```

In [47]:
```
query="SELECT * FROM student ORDER BY Mark  "
cursor.execute(query)
cursor.fetchall()
```

Out[47]:
```
[(6, 'Michael', 'Valley 345', 20),
 (1, 'prasad', 'ctc', 40),
 (2, 'prasad', 'ctc', 40),
 (3, 'Peter', 'Lowstreet 4', 40),
 (9, 'Richard', 'Sky st 331', 45),
 (12, 'Ben', 'Park Lane 38', 48),
 (4, 'Amy', 'Apple st 652', 60),
 (14, 'Chuck', 'Main Road 989', 67),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (10, 'Susan', 'One way 98', 86),
 (13, 'William', 'Central st 954', 89),
 (5, 'Hannah', 'Mountain 21', 90),
 (8, 'Betty', 'Green Grass 1', 90),
 (11, 'Vicky', 'Yellow Garden 2', 93),
 (15, 'Viola', 'Sideway 1633', 93)]
```

In [50]:
```
query="SELECT * FROM student ORDER BY Name,Mark  "
cursor.execute(query)
cursor.fetchall()
```

Out[50]:
```
[(4, 'Amy', 'Apple st 652', 60),
 (12, 'Ben', 'Park Lane 38', 48),
 (8, 'Betty', 'Green Grass 1', 90),
 (14, 'Chuck', 'Main Road 989', 67),
 (5, 'Hannah', 'Mountain 21', 90),
 (6, 'Michael', 'Valley 345', 20),
 (3, 'Peter', 'Lowstreet 4', 40),
 (1, 'prasad', 'ctc', 40),
 (2, 'prasad', 'ctc', 40),
 (9, 'Richard', 'Sky st 331', 45),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (10, 'Susan', 'One way 98', 86),
 (11, 'Vicky', 'Yellow Garden 2', 93),
 (15, 'Viola', 'Sideway 1633', 93),
 (13, 'William', 'Central st 954', 89)]
```

In [52]:
```
query="SELECT * FROM student ORDER BY Name ASC , Mark DESC "
cursor.execute(query)
cursor.fetchall()
```

Out[52]:
```
[(4, 'Amy', 'Apple st 652', 60),
 (12, 'Ben', 'Park Lane 38', 48),
 (8, 'Betty', 'Green Grass 1', 90),
 (14, 'Chuck', 'Main Road 989', 67),
 (5, 'Hannah', 'Mountain 21', 90),
 (6, 'Michael', 'Valley 345', 20),
 (3, 'Peter', 'Lowstreet 4', 40),
 (1, 'prasad', 'ctc', 40),
 (2, 'prasad', 'ctc', 40),
 (9, 'Richard', 'Sky st 331', 45),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (10, 'Susan', 'One way 98', 86),
 (11, 'Vicky', 'Yellow Garden 2', 93),
 (15, 'Viola', 'Sideway 1633', 93),
 (13, 'William', 'Central st 954', 89)]
```

Limit Clause

```
   The Limit clause is used in SQL to control or limit the number of
   records in the result set returned from the query generated. By default,
   SQL gives out the required number of records starting from the top but
   it allows the use of OFFSET keyword. OFFSET allows you to start from a
   custom row and get the required number of result rows.
```

Syntax:

SELECT * FROM tablename LIMIT limit;

SELECT * FROM tablename LIMIT limit OFFSET offset;

In [14]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()

query = "SELECT * FROM STUDENT LIMIT 2 OFFSET 1"
cursor.execute(query)

myresult = cursor.fetchall()

for x in myresult:
    print(x)
```

```
(2, 'prasad', 'ctc', 40)
(3, 'Peter', 'Lowstreet 4', 40)
```

In [16]:
```
import mysql.connector
```

```
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()

query = "SELECT * FROM STUDENT LIMIT 5 "
cursor.execute(query)

myresult = cursor.fetchall()

for x in myresult:
    print(x)
```

```
(1, 'prasad', 'ctc', 40)
(2, 'prasad', 'ctc', 40)
(3, 'Peter', 'Lowstreet 4', 40)
(4, 'Amy', 'Apple st 652', 60)
(5, 'Hannah', 'Mountain 21', 90)
```

Update Data

```
The update query is used to change the existing values in a database. By
using update a specific value can be corrected or updated. It only
affects the data and not the structure of the table. The basic advantage
provided by this command is that it keeps the table accurate.

Syntax:

UPDATE tablename
SET ="new value"
WHERE ="old value";
```

In [17]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()

query = "UPDATE studentss SET age = 23 WHERE Name ='Ram'"
cursor.execute(query)
mydb.commit()
```

In [18]:
```
query = "SELECT * FROM studentss WHERE Name ='Ram' "
cursor.execute(query)
cursor.fetchall()
```

Out[18]:
```
[('Ram', 'CSE', 85, 'B', 23)]
```

Delete Data from Table

We can use the Delete query to delete data from the table in MySQL.

Syntax:

DELETE FROM TABLE_NAME
WHERE ATTRIBUTE_NAME = ATTRIBUTE_VALUE

In [19]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()

query = "DELETE FROM studentss WHERE Name ='Ram' "
cursor.execute(query)
mydb.commit()
```

In [20]:
```
query = "SELECT * FROM studentss  "
cursor.execute(query)
cursor.fetchall()
```

Out[20]:
```
[('Amit', 'ECE', 24, 'A', 21),
 ('Rohan', 'MAE', 43, 'B', 20),
 ('Anil', 'MAE', 45, 'B', 20),
 ('Megha', 'ECE', 55, 'A', 22),
 ('Sita', 'CSE', 95, 'A', 19),
 ('Nikhil', 'CSE', 98, 'A', 18),
 ('Nisha', 'CSE', 99, 'A', 18)]
```

Drop Tables

Drop command affects the structure of the table and not data. It is used to delete an already existing table. For cases where you are not sure if the table to be dropped exists or not DROP TABLE IF EXISTS command is used. Both cases will be dealt with in the following examples.

Syntax:

DROP TABLE tablename;
DROP TABLE IF EXISTS tablename;

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

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Kanha@8144",
  database="practiceDB"
)

cursor = mydb.cursor()

query = "DROP TABLE IF EXISTS studentss  "
cursor.execute(query)
mydb.commit()
```

```
In [3]: query = "SHOW TABLES  "
        cursor.execute(query)
        cursor.fetchall()
```

```
Out[3]: [('customers',),
         ('departments',),
         ('employees',),
         ('orders',),
         ('student',),
         ('table1',),
         ('table2',)]
```

```
In [ ]:
```

```
In [ ]:
```

SELECT Statement

    The SELECT statement is used to select data from a database.

    The data returned is stored in a result table, called the result-set.

```
In [56]: #select all columns
         query="SELECT * FROM student"
         cursor.execute(query)
         cursor.fetchall()
```

```
Out[56]: [(1, 'prasad', 'ctc', 40),
          (2, 'prasad', 'ctc', 40),
          (3, 'Peter', 'Lowstreet 4', 40),
          (4, 'Amy', 'Apple st 652', 60),
          (5, 'Hannah', 'Mountain 21', 90),
          (6, 'Michael', 'Valley 345', 20),
          (7, 'Sandy', 'Ocean blvd 2', 80),
          (8, 'Betty', 'Green Grass 1', 90),
          (9, 'Richard', 'Sky st 331', 45),
          (10, 'Susan', 'One way 98', 86),
          (11, 'Vicky', 'Yellow Garden 2', 93),
          (12, 'Ben', 'Park Lane 38', 48),
          (13, 'William', 'Central st 954', 89),
          (14, 'Chuck', 'Main Road 989', 67),
          (15, 'Viola', 'Sideway 1633', 93)]
```

```
In [57]: #select specific columns
         query="SELECT Name FROM student"
         cursor.execute(query)
         cursor.fetchall()
```

```
Out[57]: [('prasad',),
          ('prasad',),
          ('Peter',),
          ('Amy',),
          ('Hannah',),
          ('Michael',),
          ('Sandy',),
          ('Betty',),
          ('Richard',),
          ('Susan',),
          ('Vicky',),
          ('Ben',),
          ('William',),
          ('Chuck',),
          ('Viola',)]
```

In [55]:
```
QUERY="SELECT TOP 3 * FROM student"
cursor.execute(query)
cursor.fetchall()
```
Out[55]:
```
[]
```
In [60]:
```
#The SELECT DISTINCT statement is used to return only distinct (different) values.

query="SELECT DISTINCT Mark FROM student"
cursor.execute(query)
cursor.fetchall()
```
Out[60]:
```
[(40,), (60,), (90,), (20,), (80,), (45,), (86,), (93,), (48,), (89,), (67,)]
```
In [64]:
```
#total number of distnict marks

query="SELECT count(DISTINCT Mark) FROM student"
cursor.execute(query)
cursor.fetchall()
```
Out[64]:
```
[(11,)]
```
In [66]:
```
query="SELECT MIN(Mark) AS min_mark FROM student"
cursor.execute(query)
cursor.fetchall()
```
Out[66]:
```
[(20,)]
```
In [ ]:
```

```
In [2]:
```
import mysql.connector


connection=mysql.connector.connect(

    host='localhost',
    user='root',
    password='Kanha@8144',

)


cursor=connection.cursor()

cursor.execute("SHOW DATABASES")
cursor.fetchall()
```
Out[2]:
```
[('imdb',),
 ('information_schema',),
 ('mysql',),
 ('performance_schema',),
 ('practiceDB',),
 ('sys',),
 ('testDB',)]
```
In [3]:
```
import mysql.connector


connection=mysql.connector.connect(

    host='localhost',
    user='root',
    password='Kanha@8144',
    database="practiceDB"

)


cursor=connection.cursor()



cursor.execute("SHOW TABLES")
cursor.fetchall()
```

Out[3]:
```
[('student',)]
```

In [4]:
```
cursor.execute("SELECT * FROM student")
cursor.fetchall()
```

Out[4]:
```
[(1, 'prasad', 'ctc', 40),
 (2, 'prasad', 'ctc', 40),
 (3, 'Peter', 'Lowstreet 4', 40),
 (4, 'Amy', 'Apple st 652', 60),
 (5, 'Hannah', 'Mountain 21', 90),
 (6, 'Michael', 'Valley 345', 20),
 (7, 'Sandy', 'Ocean blvd 2', 80),
 (8, 'Betty', 'Green Grass 1', 90),
 (9, 'Richard', 'Sky st 331', 45),
 (10, 'Susan', 'One way 98', 86),
 (11, 'Vicky', 'Yellow Garden 2', 93),
 (12, 'Ben', 'Park Lane 38', 48),
 (13, 'William', 'Central st 954', 89),
 (14, 'Chuck', 'Main Road 989', 67),
 (15, 'Viola', 'Sideway 1633', 93)]
```

In [5]:
```
#alias for columns

cursor.execute("SELECT id AS STUDENT_ID , Name as student_name FROM student")
cursor.fetchall()
```

Out[5]:
```
[(1, 'prasad'),
 (2, 'prasad'),
 (3, 'Peter'),
 (4, 'Amy'),
 (5, 'Hannah'),
 (6, 'Michael'),
 (7, 'Sandy'),
 (8, 'Betty'),
 (9, 'Richard'),
 (10, 'Susan'),
 (11, 'Vicky'),
 (12, 'Ben'),
 (13, 'William'),
 (14, 'Chuck'),
 (15, 'Viola')]
```

In [7]:
```
# Single or double quotation marks are required if the alias name contains spaces:


cursor.execute("SELECT id AS 'STUDENT ID' , Name as student_name FROM student")
cursor.fetchall()
```

Out[7]:
```
[(1, 'prasad'),
 (2, 'prasad'),
 (3, 'Peter'),
 (4, 'Amy'),
 (5, 'Hannah'),
 (6, 'Michael'),
 (7, 'Sandy'),
 (8, 'Betty'),
 (9, 'Richard'),
 (10, 'Susan'),
 (11, 'Vicky'),
 (12, 'Ben'),
 (13, 'William'),
 (14, 'Chuck'),
 (15, 'Viola')]
```

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


connection=mysql.connector.connect(

    host='localhost',
    user='root',
    password='Kanha@8144',
    database="practiceDB"

)


cursor=connection.cursor()



cursor.execute("SHOW TABLES")
cursor.fetchall()
```

Out[3]:
```
[('student',)]
```

In [4]:
```
# creating customers table

query= """CREATE TABLE customers (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    customername VARCHAR(250),
    contactname VARCHAR(250),
    address VARCHAR(250),
    city VARCHAR(250),
    postalcode INT,
    country VARCHAR(250)
)
"""



cursor.execute(query)
```

In [6]:
```
query="INSERT INTO customers (customername, contactname, address, city, postalcode, coun


VAL=[
    ('Customer 1', 'Contact 1', 'Address 1', 'City 1', 100001, 'Country 1'),
    ('Customer 2', 'Contact 2', 'Address 2', 'City 2', 100002, 'Country 2'),
    ('Customer 3', 'Contact 3', 'Address 3', 'City 3', 100003, 'Country 3'),
    ('Customer 4', 'Contact 4', 'Address 4', 'City 4', 100004, 'Country 4'),
    ('Customer 5', 'Contact 5', 'Address 5', 'City 5', 100005, 'Country 5'),
    ('Customer 6', 'Contact 6', 'Address 6', 'City 6', 100006, 'Country 6'),
    ('Customer 7', 'Contact 7', 'Address 7', 'City 7', 100007, 'Country 7'),
    ('Customer 8', 'Contact 8', 'Address 8', 'City 8', 100008, 'Country 8'),
    ('Customer 9', 'Contact 9', 'Address 9', 'City 9', 100009, 'Country 9'),
    ('Customer 10', 'Contact 10', 'Address 10', 'City 10', 100010, 'Country 10')
]
```

```
cursor.executemany(query,VAL)
connection.commit()
```

In [7]:
```
# creating orders table

query="""
CREATE TABLE orders (
    OrderID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT,
    orderdate DATE,
    totalamount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES customers(CustomerID)
)
"""

cursor.execute(query)
```

In [ ]:
```

```

In [11]:
```
query="""
INSERT INTO orders (CustomerID, orderdate, totalamount)
VALUES
    (1, '2023-01-01', 100.00),
    (2, '2023-01-02', 150.00),
    (3, '2023-01-03', 200.00),
    (4, '2023-01-04', 75.00),
    (5, '2023-01-05', 120.00),
    (6, '2023-01-06', 80.00),
    (7, '2023-01-07', 130.00),
    (8, '2023-01-08', 160.00),
    (9, '2023-01-09', 110.00),
    (10, '2023-01-10', 90.00)

"""

cursor.execute(query)
connection.commit()
```

In [14]:
```
import mysql.connector


connection=mysql.connector.connect(

    host='localhost',
    user='root',
    password='Kanha@8144',
    database="practiceDB"

)


cursor=connection.cursor()




cursor.execute("SHOW TABLES")
cursor.fetchall()
```

Out[14]:
```
[('customers',), ('orders',), ('student',)]
```

In [15]:
```
cursor.execute("SELECT * FROM customers")
cursor.fetchall()
```

```
Out[15]: [(1, 'Customer 1', 'Contact 1', 'Address 1', 'City 1', 100001, 'Country 1'),
          (2, 'Customer 2', 'Contact 2', 'Address 2', 'City 2', 100002, 'Country 2'),
          (3, 'Customer 3', 'Contact 3', 'Address 3', 'City 3', 100003, 'Country 3'),
          (4, 'Customer 4', 'Contact 4', 'Address 4', 'City 4', 100004, 'Country 4'),
          (5, 'Customer 5', 'Contact 5', 'Address 5', 'City 5', 100005, 'Country 5'),
          (6, 'Customer 6', 'Contact 6', 'Address 6', 'City 6', 100006, 'Country 6'),
          (7, 'Customer 7', 'Contact 7', 'Address 7', 'City 7', 100007, 'Country 7'),
          (8, 'Customer 8', 'Contact 8', 'Address 8', 'City 8', 100008, 'Country 8'),
          (9, 'Customer 9', 'Contact 9', 'Address 9', 'City 9', 100009, 'Country 9'),
          (10,
           'Customer 10',
           'Contact 10',
           'Address 10',
           'City 10',
           100010,
           'Country 10'),
          (11, 'Customer 1', 'Contact 1', 'Address 1', 'City 1', 100001, 'Country 1'),
          (12, 'Customer 2', 'Contact 2', 'Address 2', 'City 2', 100002, 'Country 2'),
          (13, 'Customer 3', 'Contact 3', 'Address 3', 'City 3', 100003, 'Country 3'),
          (14, 'Customer 4', 'Contact 4', 'Address 4', 'City 4', 100004, 'Country 4'),
          (15, 'Customer 5', 'Contact 5', 'Address 5', 'City 5', 100005, 'Country 5'),
          (16, 'Customer 6', 'Contact 6', 'Address 6', 'City 6', 100006, 'Country 6'),
          (17, 'Customer 7', 'Contact 7', 'Address 7', 'City 7', 100007, 'Country 7'),
          (18, 'Customer 8', 'Contact 8', 'Address 8', 'City 8', 100008, 'Country 8'),
          (19, 'Customer 9', 'Contact 9', 'Address 9', 'City 9', 100009, 'Country 9'),
          (20,
           'Customer 10',
           'Contact 10',
           'Address 10',
           'City 10',
           100010,
           'Country 10')]
```

```
In [16]: cursor.execute("SELECT * FROM orders")
         cursor.fetchall()
```

```
Out[16]: [(1, 1, datetime.date(2023, 1, 1), Decimal('100.00')),
          (2, 2, datetime.date(2023, 1, 2), Decimal('150.00')),
          (3, 3, datetime.date(2023, 1, 3), Decimal('200.00')),
          (4, 4, datetime.date(2023, 1, 4), Decimal('75.00')),
          (5, 5, datetime.date(2023, 1, 5), Decimal('120.00')),
          (6, 6, datetime.date(2023, 1, 6), Decimal('80.00')),
          (7, 7, datetime.date(2023, 1, 7), Decimal('130.00')),
          (8, 8, datetime.date(2023, 1, 8), Decimal('160.00')),
          (9, 9, datetime.date(2023, 1, 9), Decimal('110.00')),
          (10, 10, datetime.date(2023, 1, 10), Decimal('90.00'))]
```

MySQL Aliases

Aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

Alias for Columns

```
In [19]: import mysql.connector
```

```
connection=mysql.connector.connect(

    host='localhost',
    user='root',
    password='Kanha@8144',
    database="practiceDB"

)


cursor=connection.cursor()



query="SELECT CustomerID AS ID ,CustomerName AS Customer FROM customers "
cursor.execute(query)
cursor.fetchall()
```

Out[19]:
```
[(1, 'Customer 1'),
 (2, 'Customer 2'),
 (3, 'Customer 3'),
 (4, 'Customer 4'),
 (5, 'Customer 5'),
 (6, 'Customer 6'),
 (7, 'Customer 7'),
 (8, 'Customer 8'),
 (9, 'Customer 9'),
 (10, 'Customer 10'),
 (11, 'Customer 1'),
 (12, 'Customer 2'),
 (13, 'Customer 3'),
 (14, 'Customer 4'),
 (15, 'Customer 5'),
 (16, 'Customer 6'),
 (17, 'Customer 7'),
 (18, 'Customer 8'),
 (19, 'Customer 9'),
 (20, 'Customer 10')]
```

In [ ]:
```
## Note: Single or double quotation marks are required if the alias name contains spaces
```

In [21]:
```
query="SELECT CustomerName AS Customer,ContactName AS 'Contact Peraon' FROM Customers"
cursor.execute(query)
cursor.fetchall()
```

Out[21]:
```
[('Customer 1', 'Contact 1'),
 ('Customer 2', 'Contact 2'),
 ('Customer 3', 'Contact 3'),
 ('Customer 4', 'Contact 4'),
 ('Customer 5', 'Contact 5'),
 ('Customer 6', 'Contact 6'),
 ('Customer 7', 'Contact 7'),
 ('Customer 8', 'Contact 8'),
 ('Customer 9', 'Contact 9'),
 ('Customer 10', 'Contact 10'),
 ('Customer 1', 'Contact 1'),
 ('Customer 2', 'Contact 2'),
 ('Customer 3', 'Contact 3'),
 ('Customer 4', 'Contact 4'),
 ('Customer 5', 'Contact 5'),
 ('Customer 6', 'Contact 6'),
 ('Customer 7', 'Contact 7'),
 ('Customer 8', 'Contact 8'),
 ('Customer 9', 'Contact 9'),
 ('Customer 10', 'Contact 10')]
```

In [22]:
```python
query="SELECT CustomerName, CONCAT_WS(', ', Address, PostalCode, City, Country) AS Addre
cursor.execute(query)
cursor.fetchall()
```

Out[22]:
```
[('Customer 1', 'Address 1, 100001, City 1, Country 1'),
 ('Customer 2', 'Address 2, 100002, City 2, Country 2'),
 ('Customer 3', 'Address 3, 100003, City 3, Country 3'),
 ('Customer 4', 'Address 4, 100004, City 4, Country 4'),
 ('Customer 5', 'Address 5, 100005, City 5, Country 5'),
 ('Customer 6', 'Address 6, 100006, City 6, Country 6'),
 ('Customer 7', 'Address 7, 100007, City 7, Country 7'),
 ('Customer 8', 'Address 8, 100008, City 8, Country 8'),
 ('Customer 9', 'Address 9, 100009, City 9, Country 9'),
 ('Customer 10', 'Address 10, 100010, City 10, Country 10'),
 ('Customer 1', 'Address 1, 100001, City 1, Country 1'),
 ('Customer 2', 'Address 2, 100002, City 2, Country 2'),
 ('Customer 3', 'Address 3, 100003, City 3, Country 3'),
 ('Customer 4', 'Address 4, 100004, City 4, Country 4'),
 ('Customer 5', 'Address 5, 100005, City 5, Country 5'),
 ('Customer 6', 'Address 6, 100006, City 6, Country 6'),
 ('Customer 7', 'Address 7, 100007, City 7, Country 7'),
 ('Customer 8', 'Address 8, 100008, City 8, Country 8'),
 ('Customer 9', 'Address 9, 100009, City 9, Country 9'),
 ('Customer 10', 'Address 10, 100010, City 10, Country 10')]
```

Alias for Tables

In [27]:
```python
import mysql.connector


connection=mysql.connector.connect(

    host='localhost',
    user='root',
    password='Kanha@8144',
    database="practiceDB"

)


cursor=connection.cursor()


query="""SELECT o.OrderID, o.OrderDate, c.CustomerName
         FROM Customers AS c, Orders AS o
         WHERE c.CustomerName='Customer 6' AND c.CustomerID=o.CustomerID
"""

cursor.execute(query)
cursor.fetchall()
```

Out[27]:
```
[(6, datetime.date(2023, 1, 6), 'Customer 6')]
```

In [28]:
```python
## same as above but without alias

query="""
SELECT Orders.OrderID,Orders.OrderDate,Customers.CustomerName
FROM Customers,Orders
WHERE Customers.CustomerName='Customer 3'
AND Customers.CustomerID=Orders.CustomerID
```

```
"""

cursor.execute(query)
cursor.fetchall()
```

Out[28]:
```
[(3, datetime.date(2023, 1, 3), 'Customer 3')]
```

In [17]:
```
# Define aliases for the tables
customers_alias = "c"
orders_alias = "o"

# Construct a query with aliases
query = f"""
SELECT {customers_alias}.customername AS CustomerName,
       {orders_alias}.orderdate AS OrderDate
FROM customers AS {customers_alias}
INNER JOIN orders AS {orders_alias}
ON {customers_alias}.CustomerID = {orders_alias}.CustomerID;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    customer_name, order_date = row
    print(f"Customer: {customer_name}, Order Date: {order_date}")

# Close the cursor and the database connection
cursor.close()
connection.close()
```

```
Customer: Customer 1, Order Date: 2023-01-01
Customer: Customer 2, Order Date: 2023-01-02
Customer: Customer 3, Order Date: 2023-01-03
Customer: Customer 4, Order Date: 2023-01-04
Customer: Customer 5, Order Date: 2023-01-05
Customer: Customer 6, Order Date: 2023-01-06
Customer: Customer 7, Order Date: 2023-01-07
Customer: Customer 8, Order Date: 2023-01-08
Customer: Customer 9, Order Date: 2023-01-09
Customer: Customer 10, Order Date: 2023-01-10
```

MySQL JOIN

```
    A JOIN clause is used to combine rows from two or more tables, based
on a related column between them.

Let's look at a selection from the "Orders" table:
```

In [29]:
```
import mysql.connector

# Establish a database connection
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Kanha@8144",
    database="practiceDB"
)
```

```
# Create a cursor object
cursor = connection.cursor()

# Create the 'employees' table
cursor.execute("""
    CREATE TABLE employees (
        EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
        FirstName VARCHAR(255),
        LastName VARCHAR(255),
        DepartmentID INT,
        HireDate DATE
    )
""")


connection.commit()
```

In [30]:
```
# Insert data into the 'employees' table
employees_data = [
    (1, 'John', 'Doe', 3, '2023-01-10'),
    (2, 'Jane', 'Smith', 2, '2023-02-15'),
    (3, 'Bob', 'Johnson', 3, '2023-03-20'),
    (4, 'Alice', 'Williams', 1, '2023-04-25'),
    (5, 'Charlie', 'Brown', 3, '2023-05-30'),
    (6, 'Ella', 'Davis', 2, '2023-06-05'),
    (7, 'Frank', 'Lee', 4, '2023-07-10'),
    (8, 'Grace', 'Clark', 2, '2023-08-15'),
    (9, 'David', 'Hall', 3, '2023-09-20'),
    (10, 'Helen', 'Martin', 1, '2023-10-25')
]

insert_employees_query = "INSERT INTO employees (EmployeeID, FirstName, LastName, Depart
cursor.executemany(insert_employees_query, employees_data)
connection.commit()
```

In [31]:
```
# Create the 'departments' table
cursor.execute("""
    CREATE TABLE departments (
        DepartmentID INT AUTO_INCREMENT PRIMARY KEY,
        DepartmentName VARCHAR(255)
    )
""")


connection.commit()
```

In [32]:
```
# Insert data into the 'departments' table
departments_data = [
    (1, 'HR'),
    (2, 'Sales'),
    (3, 'Engineering'),
    (4, 'Marketing'),
    (5, 'Finance')
]

insert_departments_query = "INSERT INTO departments (DepartmentID, DepartmentName) VALUE
cursor.executemany(insert_departments_query, departments_data)
connection.commit()
```

INNER JOIN:

The INNER JOIN returns only the rows that have matching values in both tables.

If there is no match for a row in one of the tables, that row will not be included in the result.

It is used to retrieve records that exist in both tables.
Syntax: SELECT ... FROM table1 INNER JOIN table2 ON table1.column = table2.column

In [33]:
```python
import mysql.connector

# Establish a database connection
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Kanha@8144",
    database="practiceDB"
)

# Create a cursor object
cursor = connection.cursor()

# Perform an INNER JOIN on the 'employees' and 'departments' tables
query = """
SELECT e.FirstName, e.LastName, d.DepartmentName
FROM employees AS e
INNER JOIN departments AS d
ON e.DepartmentID = d.DepartmentID;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    first_name, last_name, department_name = row
    print(f"Employee: {first_name} {last_name}, Department: {department_name}")
```

```
Employee: John Doe, Department: Engineering
Employee: Jane Smith, Department: Sales
Employee: Bob Johnson, Department: Engineering
Employee: Alice Williams, Department: HR
Employee: Charlie Brown, Department: Engineering
Employee: Ella Davis, Department: Sales
Employee: Frank Lee, Department: Marketing
Employee: Grace Clark, Department: Sales
Employee: David Hall, Department: Engineering
Employee: Helen Martin, Department: HR
```

LEFT JOIN (or LEFT OUTER JOIN):

The LEFT JOIN returns all rows from the left table and the matched rows from the right table.

If there is no match in the right table for a row in the left table, NULL values will be included for columns from the right table.

It is used to retrieve all records from the left table and related records from the right table, if they exist.

Syntax: SELECT ... FROM table1 LEFT JOIN table2 ON table1.column = table2.column

In [34]:
```
import mysql.connector

# Establish a database connection
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Kanha@8144",
    database="practiceDB"
)

# Create a cursor object
cursor = connection.cursor()

# Perform a LEFT JOIN to retrieve all employees and their departments (if available)
query = """
SELECT e.EmployeeID, e.FirstName, e.LastName, d.DepartmentName
FROM employees e
LEFT JOIN departments d ON e.DepartmentID = d.DepartmentID;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    employee_id, first_name, last_name, department_name = row
    print(f"EmployeeID: {employee_id}, Name: {first_name} {last_name}, Department: {depa
```

```
EmployeeID: 1, Name: John Doe, Department: Engineering
EmployeeID: 2, Name: Jane Smith, Department: Sales
EmployeeID: 3, Name: Bob Johnson, Department: Engineering
EmployeeID: 4, Name: Alice Williams, Department: HR
EmployeeID: 5, Name: Charlie Brown, Department: Engineering
EmployeeID: 6, Name: Ella Davis, Department: Sales
EmployeeID: 7, Name: Frank Lee, Department: Marketing
EmployeeID: 8, Name: Grace Clark, Department: Sales
EmployeeID: 9, Name: David Hall, Department: Engineering
EmployeeID: 10, Name: Helen Martin, Department: HR
```

RIGHT JOIN (or RIGHT OUTER JOIN):

The RIGHT JOIN is the reverse of the LEFT JOIN. It returns all rows from the right table and the matched rows from the left table.

If there is no match in the left table for a row in the right table, NULL values will be included for columns from the left table.

It is used to retrieve all records from the right table and related records from the left table, if they exist.
Syntax: SELECT ... FROM table1 RIGHT JOIN table2 ON table1.column = table2.column

In [35]:
```
import mysql.connector

# Establish a database connection
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Kanha@8144",
    database="practiceDB"
)

# Create a cursor object
cursor = connection.cursor()

# Perform a RIGHT JOIN to retrieve all departments and their employees (if available)
query = """
SELECT e.EmployeeID, e.FirstName, e.LastName, d.DepartmentName
FROM employees e
RIGHT JOIN departments d ON e.DepartmentID = d.DepartmentID;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    employee_id, first_name, last_name, department_name = row
    print(f"EmployeeID: {employee_id}, Name: {first_name} {last_name}, Department: {depa
```

```
EmployeeID: 10, Name: Helen Martin, Department: HR
EmployeeID: 4, Name: Alice Williams, Department: HR
EmployeeID: 8, Name: Grace Clark, Department: Sales
EmployeeID: 6, Name: Ella Davis, Department: Sales
EmployeeID: 2, Name: Jane Smith, Department: Sales
EmployeeID: 9, Name: David Hall, Department: Engineering
EmployeeID: 5, Name: Charlie Brown, Department: Engineering
EmployeeID: 3, Name: Bob Johnson, Department: Engineering
EmployeeID: 1, Name: John Doe, Department: Engineering
EmployeeID: 7, Name: Frank Lee, Department: Marketing
EmployeeID: None, Name: None None, Department: Finance
```

SELF JOIN:

```
A SELF JOIN is used to join a table with itself. It is often used when a
table contains hierarchical or related data.
It allows you to create relationships within a single table, for
example, to find all employees and their managers.
```

Syntax: SELECT ... FROM table t1 INNER JOIN table t2 ON t1.column = t2.column

In [36]:
```
import mysql.connector

# Establish a database connection
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Kanha@8144",
    database="practiceDB"
)

# Create a cursor object
cursor = connection.cursor()

# Perform a self-join to find the manager for each employee
query = """
SELECT e1.EmployeeID AS EmployeeID, e1.FirstName AS EmployeeFirstName, e1.LastName AS Em
```

```
       e2.EmployeeID AS ManagerID, e2.FirstName AS ManagerFirstName, e2.LastName AS Mana
FROM employees e1
LEFT JOIN employees e2 ON e1.DepartmentID = e2.DepartmentID
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    employee_id, employee_first_name, employee_last_name, manager_id, manager_first_name
    print(f"EmployeeID: {employee_id}, Employee: {employee_first_name} {employee_last_na
```

```
EmployeeID: 1, Employee: John Doe, ManagerID: 9, Manager: David Hall
EmployeeID: 1, Employee: John Doe, ManagerID: 5, Manager: Charlie Brown
EmployeeID: 1, Employee: John Doe, ManagerID: 3, Manager: Bob Johnson
EmployeeID: 1, Employee: John Doe, ManagerID: 1, Manager: John Doe
EmployeeID: 2, Employee: Jane Smith, ManagerID: 8, Manager: Grace Clark
EmployeeID: 2, Employee: Jane Smith, ManagerID: 6, Manager: Ella Davis
EmployeeID: 2, Employee: Jane Smith, ManagerID: 2, Manager: Jane Smith
EmployeeID: 3, Employee: Bob Johnson, ManagerID: 9, Manager: David Hall
EmployeeID: 3, Employee: Bob Johnson, ManagerID: 5, Manager: Charlie Brown
EmployeeID: 3, Employee: Bob Johnson, ManagerID: 3, Manager: Bob Johnson
EmployeeID: 3, Employee: Bob Johnson, ManagerID: 1, Manager: John Doe
EmployeeID: 4, Employee: Alice Williams, ManagerID: 10, Manager: Helen Martin
EmployeeID: 4, Employee: Alice Williams, ManagerID: 4, Manager: Alice Williams
EmployeeID: 5, Employee: Charlie Brown, ManagerID: 9, Manager: David Hall
EmployeeID: 5, Employee: Charlie Brown, ManagerID: 5, Manager: Charlie Brown
EmployeeID: 5, Employee: Charlie Brown, ManagerID: 3, Manager: Bob Johnson
EmployeeID: 5, Employee: Charlie Brown, ManagerID: 1, Manager: John Doe
EmployeeID: 6, Employee: Ella Davis, ManagerID: 8, Manager: Grace Clark
EmployeeID: 6, Employee: Ella Davis, ManagerID: 6, Manager: Ella Davis
EmployeeID: 6, Employee: Ella Davis, ManagerID: 2, Manager: Jane Smith
EmployeeID: 7, Employee: Frank Lee, ManagerID: 7, Manager: Frank Lee
EmployeeID: 8, Employee: Grace Clark, ManagerID: 8, Manager: Grace Clark
EmployeeID: 8, Employee: Grace Clark, ManagerID: 6, Manager: Ella Davis
EmployeeID: 8, Employee: Grace Clark, ManagerID: 2, Manager: Jane Smith
EmployeeID: 9, Employee: David Hall, ManagerID: 9, Manager: David Hall
EmployeeID: 9, Employee: David Hall, ManagerID: 5, Manager: Charlie Brown
EmployeeID: 9, Employee: David Hall, ManagerID: 3, Manager: Bob Johnson
EmployeeID: 9, Employee: David Hall, ManagerID: 1, Manager: John Doe
EmployeeID: 10, Employee: Helen Martin, ManagerID: 10, Manager: Helen Martin
EmployeeID: 10, Employee: Helen Martin, ManagerID: 4, Manager: Alice Williams
```

FULL OUTER JOIN (or FULL JOIN):

```
The FULL OUTER JOIN returns all rows when there is a match in either the
left or the right table.

If there is no match in one of the tables, NULL values will be included
for columns from that table.

It is used to retrieve all records from both tables and combine them.
```

Syntax: SELECT ... FROM table1 FULL JOIN table2 ON table1.column = table2.column

In [37]:
```
import mysql.connector


# Establish a database connection
```

```
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Kanha@8144",
    database="practiceDB"
)

# Create a cursor object
cursor = connection.cursor()

# Perform a FULL OUTER JOIN using a combination of LEFT JOIN and RIGHT JOIN
# and then UNION the results
query = """
SELECT e.EmployeeID, e.FirstName, e.LastName, d.DepartmentName
FROM employees e
LEFT JOIN departments d ON e.DepartmentID = d.DepartmentID
UNION
SELECT e.EmployeeID, e.FirstName, e.LastName, d.DepartmentName
FROM employees e
RIGHT JOIN departments d ON e.DepartmentID = d.DepartmentID;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    employee_id, first_name, last_name, department_name = row
    print(f"EmployeeID: {employee_id}, Name: {first_name} {last_name}, Department: {depa
```

```
EmployeeID: 1, Name: John Doe, Department: Engineering
EmployeeID: 2, Name: Jane Smith, Department: Sales
EmployeeID: 3, Name: Bob Johnson, Department: Engineering
EmployeeID: 4, Name: Alice Williams, Department: HR
EmployeeID: 5, Name: Charlie Brown, Department: Engineering
EmployeeID: 6, Name: Ella Davis, Department: Sales
EmployeeID: 7, Name: Frank Lee, Department: Marketing
EmployeeID: 8, Name: Grace Clark, Department: Sales
EmployeeID: 9, Name: David Hall, Department: Engineering
EmployeeID: 10, Name: Helen Martin, Department: HR
EmployeeID: None, Name: None None, Department: Finance
```

UNION Operator

    The UNION operator is used to combine the result-set of two or more
    SELECT statements.

    Every SELECT statement within UNION must have the same number of columns

    The columns must also have similar data types

    The columns in every SELECT statement must also be in the same order

To perform a UNION operation in MySQL , you need two tables with similar structures. Here's an example of creating two sample tables, table1 and table2, and then performing a UNION operation to combine their data:

Create the 'table1' Table:

In [38]:
```
import mysql.connector
```

```python
# Establish a database connection
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Kanha@8144",
    database="practiceDB"
)

# Create a cursor object
cursor = connection.cursor()

# Create the 'table1' table
cursor.execute("""
    CREATE TABLE table1 (
        ID INT AUTO_INCREMENT PRIMARY KEY,
        Name VARCHAR(255),
        City VARCHAR(255)
    )
""")


connection.commit()
```

In [39]:
```python
# Create the 'table2' table
cursor.execute("""
    CREATE TABLE table2 (
        ID INT AUTO_INCREMENT PRIMARY KEY,
        Name VARCHAR(255),
        City VARCHAR(255)
    )
""")


connection.commit()
```

In [40]:
```python
# Insert data into 'table1'
insert_data_table1 = [
    (1, 'Alice', 'New York'),
    (2, 'Bob', 'Los Angeles'),
    (3, 'Charlie', 'Chicago')
]
insert_query_table1 = "INSERT INTO table1 (ID, Name, City) VALUES (%s, %s, %s)"
cursor.executemany(insert_query_table1, insert_data_table1)

# Insert data into 'table2'
insert_data_table2 = [
    (1, 'David', 'San Francisco'),
    (2, 'Ella', 'Houston'),
    (3, 'Frank', 'Miami')
]
insert_query_table2 = "INSERT INTO table2 (ID, Name, City) VALUES (%s, %s, %s)"
cursor.executemany(insert_query_table2, insert_data_table2)

# Commit the changes
connection.commit()
```

In [41]:
```python
# Perform a UNION operation to combine data from 'table1' and 'table2'
query = """
SELECT Name, City FROM table1
UNION
SELECT Name, City FROM table2;
"""
```

```
# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    name, city = row
    print(f"Name: {name}, City: {city}")
```

```
Name: Alice, City: New York
Name: Bob, City: Los Angeles
Name: Charlie, City: Chicago
Name: David, City: San Francisco
Name: Ella, City: Houston
Name: Frank, City: Miami
```

In [42]:
```
query = """
SELECT  City FROM table1
UNION ALL
SELECT  City FROM table2;
"""

# Execute the query
cursor.execute(query)
cursor.fetchall()
```

Out[42]:
```
[('New York',),
 ('Los Angeles',),
 ('Chicago',),
 ('San Francisco',),
 ('Houston',),
 ('Miami',)]
```

In [43]:
```
query = """
SELECT  City FROM table1
UNION
SELECT  City FROM table2;
"""

# Execute the query
cursor.execute(query)
cursor.fetchall()
```

Out[43]:
```
[('New York',),
 ('Los Angeles',),
 ('Chicago',),
 ('San Francisco',),
 ('Houston',),
 ('Miami',)]
```

In [44]:
```
# UNION WITH WHERE
query = """
SELECT  City,Name FROM table1
WHERE Name='Alice'
UNION
SELECT  City,Name FROM table2
WHERE Name='Frank'
"""

# Execute the query
cursor.execute(query)
cursor.fetchall()
```

Out[44]:
```
[('New York', 'Alice'), ('Miami', 'Frank')]
```

GROUP BY Statement

```
The GROUP BY statement groups rows that have the same values into
summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(),
MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more
columns.
```

In [45]:

```
# Perform a GROUP BY operation on 'table1'
query = """
SELECT City, COUNT(*) as Count
FROM table1
GROUP BY City;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
cursor.fetchall()
```

Out[45]:

```
[('New York', 1), ('Los Angeles', 1), ('Chicago', 1)]
```

In [46]:

```
# Perform a GROUP BY operation to count the number of people in each city
query = """
SELECT City, COUNT(*) as TotalPeople
FROM (
    SELECT City FROM table1
    UNION ALL
    SELECT City FROM table2
) combined_tables
GROUP BY City;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
    city, total_people = row
    print(f"City: {city}, Total People: {total_people}")
```

```
City: New York, Total People: 1
City: Los Angeles, Total People: 1
City: Chicago, Total People: 1
City: San Francisco, Total People: 1
City: Houston, Total People: 1
City: Miami, Total People: 1
```

In [ ]:

```
#!jupyter nbconvert --to webpdf --allow-chromium-download PYTHON_MYSQL.ipynb
```