1. Write SQL statement select to display customer Full Name in one column, their City and Amount of sales. We need data only for customers whose mother has brown eyes.

Sales

ID	CustomerID	CityID	Amount
1	3	2	500
2	1	1	10000
3	4	4	800
4	2	3	600
5	3	1	10000
6	1	1	630
7	3	1	960

Customer

Id	Gender	FirstName	LastName	EyeColor	IDNumber	MotherIDNumber	FatherIDNumber
1	Male	Peter	Cina	Blue	SK-156-232	NULL	NULL
2	Female	Adela	Cinova	Brown	SK-216-897	NULL	NULL
3	Female	Petra	Atkinson	Blue	SK-258-321	SK-216-897	SK-156-232
4	Male	Andrej	Nowak	Brown	SK-244-221	SK-411-897	SK-226-233
5	Female	Andrea	Atkinson	Green	SK-411-897	NULL	NULL
6	Male	Jozef	Jovanovic	Green	SK-226-233	NULL	NULL

Address

ID	Country	City
1	Slovakia	Presov
2	England	London
3	Slovakia	Bratislava
4	Slovakia	Trnava

Solution:

```
select CONCAT(c.FirstName,' ',c.LastName)as FullName,a.City,s.Amount from sales s JOIN customer c
ON s.customerID=c.ID JOIN address a ON s.CityID = a.ID JOIN customer m ON c.MotherIDNumber = m.IDNumber
WHERE m.EyeColor = 'Brown' ;
```

Answers:-

FullName	City	Amount
Petra Atkinson	Presov	960
Petra Atkinson	Presov	10000
Petra Atkinson	London	500

2. Write SQL statement select to display First Name and Last Name of users which ordered 3 and more courses. Use tables from below.

Course Table

ID	Name
1	Course A
2	Course B
3	Course C
4	Course D

ID	UserID	CourseID
1	1	1
2	2	1
3	3	1
4	2	2
5	4	2
6	5	3
7	6	4
8	7	3
9	3	4
10	5	4
11	6	2
12	2	3

User Table

ID	FirstName	LastName
1	Peter	Jovanovic
2	Jozef	Djordjevic
3	Milan	Atkinson
4	Maria	Armstrong
5	Slavomir	Cina
6	Robert	Varga
7	Peter	Nowak

Solution :

```
select UserTable.FirstName,UserTable.lastName From UserTable JOIN OrderTable ON UserTable.ID =
OrderTable.UserID GROUP BY UserTable.ID, UserTable.FirstName, UserTable.lastName HAVING
COUNT(Ordertable.CourseID) >=3;
```

Answers:-

FirstName	lastName
Jozef	Djordjevic

3. What will be the result of the select below

```
SELECT

        SUM(p.Amount) AS Amount

FROM

      Payments p
      INNER JOIN Clients c ON p.ClientId = c.Id
      INNER JOIN Address a ON c.Id = a.ClientId

WHERE

        c.Name LIKE '%iro
```

Clients

Id	Name	Age
1	Fero	14
2	Jozo	16
3	Miro	22
4	David	10
5	Vlado	35

Payments

Id	DueDate	Amount	ClientId
1	2016-07-08	100	1
2	2016-07-25	200	1
3	2016-09-08	300	2
4	2016-07-11	400	2
5	2016-11-12	500	3

Address

Id	Line 1	City	IsPrimary	ClientId
1	Fucikova 1	Bratislava	0	1
2	Jesenskeho 2	Trnava	1	1
3	Odborarska 3	Senec	0	1
4	Bottova 4	Malacky	0	3
5	Holleho 5	Topolcany	1	3

Answer :- The Result for Given SQL Query is

| Amount |

| 500 |

PYTHON questions:

1. What is tuple in Python? What is the difference between list and tuple?

Answer:-

A tuple in Python is an immutable sequence type. This means once a tuple is created, its contents cannot be modified you cannot add, remove, or change elements. Tuples are defined by enclosing elements in parentheses '()'.

Example:-
```
mytuple = ("apple",1, "banana",2, "cherry",3)
print(mytuple)
```

Difference:-

List	Tuple
1.Lists are mutable, meaning you can change their contents after creation (we can adding or removing elements).	1. Tuples are immutable meaning you cannot change their contents after creation (we cannot adding or removing elements).
2. Lists use square brackets '[]'.	2. Tuples use parentheses '( )'.
3.Use Case**:** Lists are typically used for collections of items that might need to be changed or updated.	3. Use Case**:** while tuples are used for fixed collections of items or when you need to ensure that the data Cannot be modified.
4. Performance**:** List are generally Slower than tuples because of their mutability.	4. Performance**:** Tuples are generally faster than lists because of their immutability. .

2. What are the rules for a local and global variable in Python?

Answer:-

Local Variables:

- A variable declared inside a function or a block is a local variable.
- Local variables are only accessible within the function or block where they are defined.
- They are created when the function or block is executed and destroyed once the function or block completes execution.

Global Variables:

- A variable declared outside of all functions or blocks is a global variable.
- Global variables can be accessed from any function or block within the same module.
- To modify a global variable inside a function, you need to use the 'global' keyword.

Example:-

```
x = 10  # global variable

def func():
    global x
    x = 20  # modifies the global variable x
    y = 30  # local variable
func()
print(x)  # Output: 20
# print(y)  # This would raise an error because y is local to function()
```

3. What is Python's parameter passing mechanism? Name it and explain it.

Answer:-

Python uses a parameter passing mechanism known as **"pass-by-object-reference"**.

Explanation:

- When you pass a variable to a function in Python, you're passing a reference to the object, not the actual object itself.
- If you modify the object inside the function (e.g., if it's a mutable object like a list or dictionary), the changes will be reflected outside the function.
- However, if you reassign the variable to a new object within the function, this reassignment does not affect the original variable outside the function.

Example:-

```
def modify_list(lst):
    lst.append(4)  # This modifies the original list
    lst = [1, 2]  # This reassigns lst to a new list, not affecting the original list

my_list = [1, 2, 3]
modify_list(my_list)
print(my_list)
 # Output: [1, 2, 3, 4]
```

4. Write a method to open a text file and display its content?

Answer:-

```
def read_and_display_file(file_path):
    try:
        with open(file_path, 'r') as file:
            content = file.read()
            print(content)
    except FileNotFoundError:
        print("The file was not found.")
    except IOError:
            print("An error occurred while reading the file.")
```

This method uses a with statement to ensure the file is properly closed after reading. It also handles potential exceptions if the file does not exist or an I/O error occurs.

5. You have two lists: strList = ["Vishesh", "For", "Python"] and valList = [1, 2] for the first two tasks and one list valList = [1, 2, 3] for third task. Write the sintax so you will get these results:

1) {'key2': ['Vishesh', 'For', 'Python'], 'key1': [1, 2]}
2) {'key1': [1, 2, ['vishesh', 'For', 'python']]}
3) {'1': [1, 2], '3': [3, 4], '2': [2, 3]} # Creating a dictionary of lists using list comprehension.

Answer:-

1) {'key2': ['Vishesh', 'For', 'Python'], 'key1': [1, 2]}

Solution:-

```
strList = ["Vishesh", "For", "Python"]
valList = [1, 2]
result1 = {'key2': strList, 'key1': valList}
print(result1)

# Output: {'key2': ['Vishesh', 'For', 'Python'], 'key1': [1, 2]}
```

2) {'key1': [1, 2, ['vishesh', 'For', 'python']]}

Solution:-

```
strList = ["vishesh", "For", "python"]
valList = [1, 2, ]
result2 = {'key1': valList + [strList]}
print(result2)

# Output: {'key1': [1, 2, ['vishesh', 'For', 'python']]}
```

3) {'1': [1, 2], '3': [3, 4], '2': [2, 3]} # Creating a dictionary of lists using list comprehension.

Solution:-

```
valList = [1, 2, 3]
result3 = {str(v): [i, i+1] for v, i in zip(valList, range(len(valList)))}
print(result3)

# Output: {'1': [1, 2], '3': [3, 4],'2': [2, 3]}
```

In this last example, we use zip to combine valList with a range of indices and create dictionary entries where each key is a string representation of the original list values and each value is a list containing the original value and the next integer.