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

Answer:

```
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 c.CityID = a.ID WHERE c.MotherID IN
(SELECT Id FROM Customer WHERE EyeColor = 'Brown');
```

---

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

Order Table

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

Answer:

```
SELECT u.FirstName, u.LastName FROM UserTable u
JOIN OrderTable o ON u.Id = o.UserID GROUP BY u.FirstName, u.LastName HAVING
COUNT(o.CourseID) >= 3;
```

---

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:

Amount

NULL

PYTHON Questions & Answers:

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

Python Tuple is a collection of objects separated by commas.
Tuples are immutable and ordered and allow duplicate values.
There are various ways by which you can create a tuple in Python. They are as follows:

- Using round brackets
- With one item
- Tuple Constructor

Difference between list and tuple is tuples can't be changed after they're created, but lists can be modified. Tuples use less memory than lists. They are also a bit faster, especially when you're just looking up values.

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

Local variables are defined within a function or block of code and are only accessible within that specific function or block.
They are created when the function is called and destroyed when the function exits.
No special keyword is required to declare a local variable.

Global variables are defined outside of any function or block and are accessible from anywhere within the program.
They exist throughout the program's execution.

No special keyword is required to declare a global variable, but you can use the global keyword to modify a global variable inside a function.

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

Python uses a parameter-passing mechanism called "Pass by Object Reference" or "Pass by Assignment".
When you pass something (like a variable) to a function in Python, you're not actually passing the thing itself. Instead, you're passing a reference to the object that the variable is pointing to. Think of it like giving someone the address of a house rather than giving them the house itself.
Now, whether the function can change that object depends on whether the object is mutable or immutable:

- Mutable objects (like lists, dictionaries, or sets) can be changed. So, if you pass a list to a function and the function modifies it, those changes will stick, even outside the function.
- Immutable objects (like integers, strings, or tuples) can't be changed. If you pass one of these to a function and try to modify it, Python will actually create a new object instead, leaving the original one untouched.

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

```
def display_file_content(filename):
with open(filename, 'r') as file:
 content = file.read()
print(content)
```

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]}

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

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

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

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

```
valList = [1, 2, 3]
result = {str(x): [x, x + 1] for x in valList}
print(result)
```