Cogitate

Digital Insurance Accelerated

TASK DOCUMENT

DATA SCIENCE POSITION

TASK 1

Description: You are given three tables: Students, Friends and Packages. Students contains two columns: ID and Name. Friends contains two columns: ID and Friend_ID (ID of the ONLY best friend). Packages contain two columns: ID and Salary (offered salary in $ thousands per month).

Students	
ID	**Name**
1	Ashley
2	Samantha
3	Julia
4	Scarlet

Friends	
ID	**Friend_ID**
1	2
2	3
3	4
4	1

Packages	
ID	**Salary**
1	15.2
2	10.06
3	11.55
4	12.12

Provide solution and approach for the following:

Write a query to output the names of those students whose friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that neither of the two students got the same salary offer.

To solve this problem

- First of all we have to `create tables` named `Students, Friends, Packages`
- `Insert` The Following Details As `Shown In Table` Because To Write A Query For The Following Output . We Have To Create The Tables And Insert The Values From The Given Table
- We need to `Join` The tables `Students and Friends` on the condition that the `ID` of the `students matches` with the `ID of the Friends table` from there `we join` the table with `packages`
- We compare the salaries of each student with the salary of best friends
- Finally we arrange them in a Higher order

Output

ID	Name
1	Ashley
2	Samantha
3	Julia
4	Scarlet

ID	Friend_ID
1	2
2	3
3	4
4	1

ID	Salary
1	15.2
2	10.06
3	11.55
4	12.12

```
CREATE TABLE Students(ID int,Name varchar(50));  -- Creating a Students Table

insert into Students(ID,Name)                    -- inserting values in Students table

values (1,'Ashley'),(2,'Samantha'),(3,'Julia'),(4,'Scarlet');

select * from Students;                          -- Verify the Students Table
-- ------------------------------------------------------------------------------------

create table Friends(ID int, Friend_ID int);     -- Creating a Friends Table

insert into Friends(ID,Friend_ID)                -- inserting values in Friends table

values (1,2),(2,3),(3,4),(4,1);
select * from Friends;                           -- Verify the Friends Table
-- ------------------------------------------------------------------------------------
create table Packages(ID int, Salary float);     -- Creating a Packages Table

insert into Packages(ID,Salary)                  -- inserting values in Packages table
values (1,15.2),(2,10.06),(3,11.55),(4,12.12);

select * from Packages;                          -- Verify the Packages Table
```

Final Query & Output

```sql
select S1.Name as Student_Name, -- Selecting the student's name
       S2.Name as Friend_Name,  -- Selecting the friend's name
       P2.Salary as Friend_Salary -- Selecting the friend's salary
from Friends F
join Students S1 on F.ID = S1.ID -- Joining Friends table with Students table to get the student's name
join Students S2 on F.Friend_ID = S2.ID -- Joining Friends table with Students table again to get the friend's name
join Packages P1 on S1.ID = P1.ID -- Joining the result with Packages table to get the student's salary
join Packages P2 on S2.ID = P2.ID -- Joining the result with Packages table again to get the friend's salary
where P2.Salary > P1.Salary -- Filtering the result to get only those rows where the friend's salary is higher
order by P2.Salary; -- Ordering the result by the friend's salary
```

Student_Name	Friend_Name	Friend_Salary
Samantha	Julia	11.55
Julia	Scarlet	12.12
Scarlet	Ashley	15.2

Explanation Of The Query Task 1

1. `select S1.Name AS Student_Name, S2.Name AS Friend_Name, P2.Salary AS Friend_Salary`:
   - This selects the student's name (S1.Name), friend's name (S2.Name), and friend's salary (P2.Salary) from the tables.

2. `from Friends F`:
   - This starts by selecting from the Friends table (F), which connects students with their best friends.

3. `join Students S1 on F.ID = S1.ID`:
   - Joins the Friends table (F) with the Students table (S1) on the student's ID (F.ID = S1.ID), retrieving the student's name.

4. `join Students S2 on F.Friend_ID = S2.ID`:
   - Joins the Friends table (F) again with the Students table (S2) on the friend's ID (F.Friend_ID = S2.ID), retrieving the friend's name.

5. `join Packages P1 on S1.ID = P1.ID`:
   - Joins the result with the Packages table (P1) to get the salary of the student (P1.Salary).

6. `join Packages P2 on S2.ID = P2.ID`:
   - Joins the result with the Packages table (P2) again to get the salary of the friend (P2.Salary).

7. `where P2.Salary > P1.Salary`:
   - Filters the result to only include rows where the friend's salary (P2.Salary) is greater than the student's salary (P1.Salary), as required by the question.

8. `order by P2.Salary`:
   - Orders the result set by the friend's salary (P2.Salary) in ascending order.

TASK 2

Description: You are working with a student management system that stores student information in a table. The table has the following schema: ID, Name, Marks

Input

ID	Name	Marks
1	Alice	85
2	Bob	90
3	Carol	75
4	Dave	80
5	Eve	70
6	Frank	95

Expected Output

ID	Name	Marks
1	Bob	90
2	Alice	85
3	Dave	80
4	Carol	75
5	Frank	95
6	Eve	70

Provide solution and approach for the following:

You are required to write a SQL query to swap the Name and Marks values where Id is odd with even Id. The Id values must remain unchanged, ensuring that only the Name and Marks values are swapped between these two specific rows.

What is swapping ? And How It Works In Terms Of SQL

Swapping means to exchange their position or values of two items. In the context of data swapping refers to exchanging the value of two variables or table columns

Example: If we have 2 variables a & b and their respective values are 3 & 4 if we use the swapping the values of a & b will be 4 & 3

In SQL swapping refers to exchanging the values between columns and rows in a table. This operation is useful in various scenarios like reorganizing data , correcting errors or Updating the records

Id	Name	Family
1	jame	Ander
2	Mary	Simson

Id	Name	Family
1	Mary	Simson
2	jame	Ander

Example: From the above table swapping is used. Swapping is done on rows exchanging the rows of name and Family.
As you see the table
id 1 jame is Swap by the Mary & same Mary Swap by jame

To solve this problem

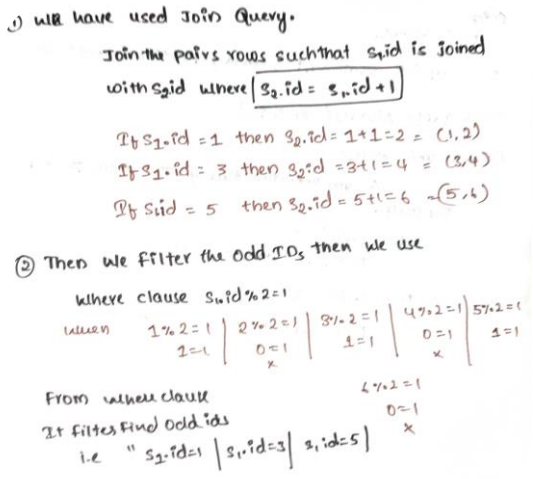
- First Of All We Have To Create Tables Named `Student_Management_System`
- Insert The Following Details As Shown In Table Because To Write A Query For The Following Output . We Have To Create The Tables And Insert The Values From The Given Table
- We Swap The `Name & Marks` Values Between Rows Where the ID is Odd and Even . We can use a combination of self-joins and conditional logic In SQL
- Identify The Pairs Of Rows Where The First Row Has An Odd ID And Next Row Has An Even ID
- Swap Them According To Given Condition Name & Marks

```sql
create database if not exists Task_2;
use Task_2;
-- Creating a Table
 create table if not exists student_management_system( id int,Name varchar(50),Marks Int);

insert into student_management_system() -- inserting values in to the student_management_system
values
(1,'Alice',85), (2,'Bob',90), (3,'Carol',75),
 (4,'Dave',80), (5,'Eve',70), (6,'Frank',95);

select * from student_management_system
```

id	Name	Marks
1	Alice	85
2	Bob	90
3	Carol	75
4	Dave	80
5	Eve	70
6	Frank	95

Final Query & Output

```sql
set SQL_SAFE_UPDATES = 0;
update student_management_system as s1
join student_management_system as s2 on s1.id = s2.id - 1
set
    s1.Name = s2.Name,
    s2.Name = s1.Name,
    s1.Marks = s2.Marks,
    s2.Marks = s1.Marks
where s1.id % 2 = 1; -- Ensures the swap occurs only for rows where ID is odd

select * from student_management_system
```



Expected Output

ID	Name	Marks
1	Bob	90
2	Alice	85
3	Dave	80
4	Carol	75
5	Frank	95
6	Eve	70

id	Name	Marks
1	Bob	90
2	Alice	85
3	Dave	80
4	Carol	75
5	Frank	95
6	Eve	70

Explanation Of The Query Task 2

1. **`update student_management_system as s1`**
   - It Starts the `update statement` and `Aliases` the table `student_management_system as S1` .

2. **`Table Aliases (s1 s2)`**
   - `student management system as S1` : The main table is `aliased as s1`
   - `student_management_system as S2` : The same table is joined and `aliased as s2`

3. **`join student_management_system  as s2 on s1.id=s2.id-1`**
   - In this Query we `join the student_management system` table with `itself` i.e `s2 alias` .
     - The condition used in the join statement is `s1.id=s2.id-1` represents that the row of s1 based on their ID. Says That `s1 with odd ID` Rows and `s2 with even ID` Row

4. **`set`**
   - `Swaps Name and Marks` values between `s1` and `s2` :
     - `s1.Name = s2.Name` : Sets s1's.Name to s2's.Name.
     - `s2.Name = s1.Name` : Sets s2's Name to s1's original Name
     - `Similarly` , same `swaps` is done on `Marks values` it set the `s1's Marks to s2's Marks` and `s2's Marks to s1's original Marks` .

5. **`Finally we pass the where Query`**
   - Where we have passed the condition `s1.id%2=1`
   - This condition says that only rows where `s1.id is odd` are considered for the `update` . The `modulus operation` % checks if `s1.id divided by 2` has a `remainder of 1` , which indicates that the `id is odd` .

Task 4

Python Requests Module

The `requests` library in Python is a `popular HTTP library` for `making HTTP requests`. It simplifies sending HTTP requests and handling responses, making it easier to interact with web services and APIs. The requests library is widely used due to its `simplicity and versatility` in handling HTTP requests and responses in Python applications.

BeautifulSoup

The `BeautifulSoup` is a Python library used for parsing `HTML and XML` documents, `primarily used for web scraping tasks`. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse HTML and retrieve tag attributes and `text content` extract `text from entir document` and also Supports different parsers (`html.parser, lxml, html5lib`).

`Robot.txt file`

A `robots. txt file` contains instructions for bots that tell them which `webpages they can and cannot access`. Robots. txt files are most relevant for `web crawlers` from search engines like Google.

The `User-Agent` header identifies the software making an HTTP request, `like a browser or web scraper`, while `Accept-Language tells` servers the preferred language for content. These headers help `mimic human browsing`, `reducing bot detection` and ensuring relevant content retrieval in the desired language.

To Avoid From `Access Denied` We Use `User_agent` i.e `user_agent = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36",'Accept-Language':'en-US,en;q=0.5'}`

In [1]:
```
import requests
from bs4 import BeautifulSoup
```

In [2]:
```
url = 'https://www.cogitate.us/'
headers = {
    'User-Agent':
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'
}
```

In [3]:
```
web_page=requests.get(url,headers=headers).text
```

In [4]:
```
soup =BeautifulSoup(web_page,'html.parser')
```

In [5]:
```
#extracting a logo
```

In [6]:
```
logo=soup.find('a',class_='logo')
```

In [7]:
```
logo
```

Out[7]:
```
<a class="logo" href="https://www.cogitate.us/"><img alt="Cogitate Technology Solutions" height="48" src="https://uatapps.cogit
ate.us/cogitatecms/images/logo.svg" width="162"/></a>
```

In [8]:
```
logo_url=logo.find('img')['src']
```

In [9]:
```
logo_url
```

Out[9]:
```
'https://uatapps.cogitate.us/cogitatecms/images/logo.svg'
```

Using the above link to see logo

Cogitate

In [10]:
```
# or we can find directly by using img tag
logo_url1=soup.find('img')['src']
logo_url1
```

Out[10]:
```
'https://uatapps.cogitate.us/cogitatecms/images/logo.svg'
```

In [11]:
```
print(logo_url)
print(logo_url1)
```

```
https://uatapps.cogitate.us/cogitatecms/images/logo.svg
https://uatapps.cogitate.us/cogitatecms/images/logo.svg
```

In [17]:
```
soup.find_all('a',class_='primaryBtn my-3')
```

Out[17]:
```
[<a class="primaryBtn my-3" href="#deforinsurance">Digital<span class="text-italic">Edge</span> for Insurance</a>,
 <a class="primaryBtn my-3" href="https://www.cogitate.us/digitalEdge-policy/">LEARN MORE<span class="screen-reader-text">Details</span></a>,
 <a class="primaryBtn my-3" href="https://www.cogitate.us/digitalEdge-billing/">LEARN MORE<span class="screen-reader-text">Details</span></a>,
 <a class="primaryBtn my-3" href="https://www.cogitate.us/digitaledge-claims/">LEARN MORE<span class="screen-reader-text">Details</span></a>,
 <a class="primaryBtn my-3" href="https://www.cogitate.us/why-cogitate/">LEARN MORE<span class="screen-reader-text">Details</span></a>]
```

In [20]:
```
soup.find_all('style',class_='primaryBtn my-3')
```

Out[20]:
```
[]
```

`Task Update` : I successfully extracted the logo from the provided URL. However, I encountered challenges with identifying and extracting the primary colors and button colors from the website. As I am not familiar with this process, I apologize for the inconvenience.