## *Input (Data)*

[ASDE\_ASSIGNMENT](https://docs.google.com/spreadsheets/d/1lHYAKwfWEONislVTkkHu_zMoAWCOG1MlAt8nPoW0grY/edit#gid=0)

## *Output (Report)*

Fetch top 3 departments along with their name and average monthly salary. Below is the format of the report.

| **DEPT\_NAME** | **AVG\_MONTHLY\_SALARY (USD)** |
| --- | --- |

## 

## **Task-1 SQL**

In the attachment above, use each worksheet as a table in a relational database and write an SQL query that generates the output report

**Steps to be followed:**

1) Create a new schema: : Launch MySQL Workbench and connect to your MySQL server.

If you don't have an existing schema to import the data into, create a new schema in MySQL Workbench. Right-click on the "SCHEMAS" tab and choose "Create Schema." Enter a name for the schema and click "Apply" to create it.

2) Import the CSV file: In MySQL Workbench, go to the "Server" menu and choose "Data Import."Select". Import File" and click the "..." button to browse and select the CSV file you saved earlier. Choose the appropriate options for the import, such as the target schema and table name. Make sure the column names and data types match your data. Click "Next" to proceed.

3) Verify the imported data: In MySQL Workbench, navigate to the schema and table where you imported the data. Right-click on the table and choose "Select Rows - Limit 1000" to view a sample of the imported data. Verify that the data is correctly imported and aligned with the original Excel file.

4) Create SQL file: Create SQL file to write query to fetch top 3 departments along with their name and average monthly salary

5) First ‘SELECT’ columns to be retrieved from tables.The ‘FROM’ specifies ‘employees’ as E.Then we move forward to ‘LEFT JOIN’ to join ‘departments’ & ‘employees’ using dept\_id(‘ID’) in departments & dept\_id(‘DEPT\_ID’) in ‘employees’.

6) Again ‘LEFT JOIN’ is performed for joining ‘salaries’ table with ‘employees’ table based on EMP\_ID column & ID column in ‘departments’ table.

7) Then ‘GROUP BY’ results with department name & ‘ORDER BY’ for sorting the result based on average salary in descending order by setting the ‘LIMIT’ as 3.

8) Click here to view Submitted SQL query :[**task1.sql**](https://drive.google.com/file/d/1zBCMy-1GPJ5b21nwjyIcpd_-Eu6FDxdW/view?usp=share_link)

## **Task-2 Scripting**

With the same attachment, use each worksheet as a CSV file and write a script (Bash or Python) that generates the same report. Data is to be read from the CSV files not from a database.

**Steps to be followed:**

1) Initial step is to read data from CSV files. For that purpose we have defined 3 methods as below:

**read\_department\_data(filename)**

**read\_employee\_data(filename)**

**read\_salary\_data(filename)**

Then provide appropriate paths to CSV files as per the function name.

2) Next step is to get the employees list for each department. For this we have created a method **get\_emplist(department)** that takes the department as a parameter.

3) For getting the average department salaries using employees list and department we have created a method:

**get\_avg\_dept\_salaries(employees\_list, salaries)**

4) So final step is to get the top 3 departments with average monthly salary we have created method:

**get\_top\_departments(avg\_dept\_sal\_with\_names)**

The output of this method is a dictionary which is being inverted further to get the desired format i.e dept\_name and average salary.

5) Then finally we need to generate a report using the top\_departments we got in the desired manner. For this we have created method:

**generate\_report(top\_departments)**

6) Click here to view Submitted Script with corrections:[**task2.py**](https://drive.google.com/file/d/1Cl1SgjsLRt6pIp_byT7KjC7gtyMw4TEP/view?usp=share_link)

## **Task-3 Debugging**

Given below is a Bash / Python script that performs following computation on an integer input (n):

1. If n is less than 10: Calculate its Square
   1. Example: 4 => 16
2. If n is between 10 and 20: Calculate the factorial of (n-10)
   1. Example: 15 => 120
3. If n is greater than 20: Calculate the sum of all integers between 1 and (n-20)
   1. Example: 25 => 15

The task is to identify the bugs in the script, fix them and share the new script. Only one of the two scripts required Bash or Python. **Hint**: You can correct the script by only changing 3-4 characters.

**Steps to be followed to debug the given code:**

1) In the given python script the range for the second elif condition is given as range(1,n-10) which is incorrect,that must be corrected as **range(1,n-9)**.

2) Similarly, for else condition the range (n-20) is incorrect which must be corrected as **(n-19)**

3) Also, the integer division operator is used which must be corrected as floor division operator.

4) Test Cases for above script would be:

i) If n is less than 10: Calculate its Square i.e (n\*n)

Example: If n = 4, then

n \* n = 4 \* 4 = 16

4 => 16

ii) If n is between 10 and 20: Calculate the factorial of (n-10)

Example: 15 => 120

iii) If n is greater than 20: Calculate the sum of all integers between 1 and (n-20),

We have logic as:

lim = n -19

out = lim \* lim

out = out - lim

out = out // 2

Example: If n=25,then

lim = 25 -19 = 6

sum = 6 \* 6 = 36

sum = 36 - 6 = 30

sum = 30 // 2 = 15

Sum = 25 => 15

5) Click here to view Submitted Script with corrections:[**task3.py**](https://drive.google.com/file/d/1hjMBJp0QpPw0wNvfNpWPpJCQWA6dsfMH/view?usp=share_link)

Command to run script :

>> **python3 task3.py**