# OnlineSales.Ai Assignment

**Task 1 SQL Query:-** Fetch the top 3 departments along with their name and average monthly salary. Below is the format of the report.

**Query:-** SELECT  d.NAME AS DEPT_NAME, AVG(s.AMT) AS AVG_MONTHLY_SALARY
FROM departments d
JOIN employees e ON d.ID = e.DEPT_ID
JOIN salaries s ON e.ID = s.EMP_ID
GROUP BY d.ID, d.NAME
ORDER BY AVG_MONTHLY_SALARY DESC
LIMIT 3;

**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

**Information**:- I used **Python** for this task. The code is written below

```
# create Index.py and run the test after reading CSV File

import csv

# Function to calculate average salary
def calculate_average_salary(salaries):
    total_salary = sum(salaries)
    average_salary = total_salary / len(salaries)
    return round(average_salary, 2)
```

```python
# Read departments from departments.csv
departments = {}
with open('departments.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        department_id = int(row[0])
        department_name = row[1]
        departments[department_id] = department_name


# Read employees from employees.csv
employees = {}
with open('employees.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        employee_id = int(row[0])
        employee_name = row[1]
        department_id = int(row[2])
        employees[employee_id] = (employee_name, department_id)


# Read salaries from salaries.csv
salaries = {}
with open('salaries.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        employee_id = int(row[0])
        month = row[1]
        salary = int(row[2])
        if employee_id in salaries:
            salaries[employee_id].append(salary)
        else:
            salaries[employee_id] = [salary]
```

```
# Calculate average monthly salary for each department
department_salaries = {}
for employee_id, (employee_name, department_id) in employees.items():
    if department_id in departments:  # Check if department ID is valid
        if department_id in department_salaries:

department_salaries[department_id].extend(salaries[employee_id])
        else:
            department_salaries[department_id] = salaries[employee_id]
    else:
        print(f"Invalid department ID for employee {employee_id}")

# Generate the report
print("DEPT_NAME ", " AVG_MONTHLY_SALARY (USD)")
# print("AVG_MONTHLY_SALARY (USD)")
print()

# Sort departments by average salary in descending order
sorted_departments = sorted(department_salaries.items(), key=lambda x:
calculate_average_salary(x[1]), reverse=True)

# Display top 3 departments
for i in range(3):
    department_id, salaries = sorted_departments[i]
    department_name = departments.get(department_id, "Unknown")
    average_salary = calculate_average_salary(salaries)
    print(department_name," ", average_salary)
    # print(average_salary)
    print()
```

**Task-3 Debugging:-** Given below is a Bash / Python script that performs the following computation on an integer input (n):

1. If n is less than 10: Calculate its Square
    a. Example: 4 => 16
2. If n is between 10 and 20: Calculate the factorial of (n-10)

    a. Example: 15 => 120
3. If n is greater than 20: Calculate the sum of all integers between 1 and (n-20)
    a. 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.

**Information**:- I used **Python** for this task. The code is written below

**Details**:-The bugs in the script can be fixed as follows:

- In the second part of the script, when calculating the factorial, the range should start from 1, not from 0. This is because the factorial of 0 is 1.

- In the third part of the script, the sum of integers between 1 and (n-20) should be calculated using the formula for the sum of an arithmetic series, which is (n * (n + 1)) / 2, not by squaring and subtracting lim.

```python
def compute(n):
    if n < 10:
        out = n ** 2
    elif n < 20:
        out = 1
        for i in range(1, n - 9):  # Fixed the range starting from 1
            out *= i
    else:
        lim = n - 20
        out = (lim * (lim + 1)) // 2  # Fixed the calculation for sum of integers
    print(out)


n = int(input("Enter an integer: "))
compute(n)
```