System Requirements Specification Index

For

Pyspark Usecase

Employee Data processing analysis L2

1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Problem Statement : **Employee data processing**
Description : Use relevant methods operations toperform specified activities which are given in the instructions.

PYSPARK TASK -L2
A mid-sized company wants to analyze employee data stored in a MySQL database. The data includes employee details, their skills, salaries, years of experience, department information, and bonuses. The HR and management teams need insights to make strategic decisions related to employee compensation, performance, and department-level skill distribution.
The company's employee data is divided across different tables:

1. **Employee**: Contains basic details of employees such as EmployeeID, Name, and DepartmentID.
2. **Skills**: Tracks each employee's proficiency in various skills (e.g., Beginner, Intermediate, Expert).
3. **ExperienceSalary**: Holds employee salary, bonus, and years of experience information.
4. **Department**: Contains information about department names and their locations.

The company uses **PySpark** as a data processing engine for its ability to handle large datasets, and they want to perform the following tasks using PySpark and MySQL
Database to be created for the Pyspark code
To create the database you login to mysqlworkbench on the desktop where the usename and password to the msql workbench is mentioned on the desktop

To avoid connectivity issues make sure the mysql-connector-java-8.0.12.jar **is inside the folder of the environment**
Employee Table
The Employee table stores information about employees.

- **EmployeeID**: INT (Primary Key) – Unique identifier for each employee.
- **Name**: VARCHAR(50) – Employee's name (up to 50 characters).
- **DepartmentID**: INT – Foreign key referencing the Department table, identifying which department the employee belongs to.
- **HireDate**: VARCHAR(50) – The date the employee was hired (stored as a string).

Skills Table
The Skills table stores information about the skills employees possess.

- **SkillID**: INT (Primary Key) – Unique identifier for each skill.
- **SkillName**: VARCHAR(50) – The name of the skill.
- **EmployeeID**: INT – Foreign key referencing the Employee table, representing which employee has this skill.
- **ProficiencyLevel**: VARCHAR(20) – The level of proficiency the employee has in this skill (e.g., 'Expert', 'Intermediate').

ExperienceSalary Table

The ExperienceSalary table stores information about the years of experience and salary for each employee.

- **EmployeeID**: INT (Primary Key) – Unique identifier for each employee, referencing the Employee table.
- **YearsOfExperience**: INT – Number of years the employee has worked.
- **Salary**: DECIMAL(10, 2) – The employee's salary.
- **Bonus**: DECIMAL(10, 2) – The employee's bonus.

The Department table stores information about the departments within the company.

- **DepartmentID**: INT (Primary Key) – Unique identifier for each department.

- **DepartmentName**: VARCHAR(50) – Name of the department.
- **ManagerID**: INT – EmployeeID of the manager of the department, referencing the Employee table.
- **Location**: VARCHAR(50) – The location where the department is based.

Relationships

- The EmployeeID in the Skills and ExperienceSalary tables references the Employee table.
- The DepartmentID in the Employee table references the Department table.
- The ManagerID in the Department table references the Employee table, indicating the employee who manages the department.

Dataset to be used

EmployeeID	Name	DepartmentID	HireDate
1	Alice Smith	101	2020-01-15
2	Bob Johnson	102	2019-03-22
3	Carol Davis	103	2021-07-30
4	David Brown	101	2018-11-12
5	Eve White	104	2022-06-01
6	Frank Green	105	2019-08-17
7	Grace Taylor	102	2020-05-19
8	Hank Wilson	103	2021-09-09

Skills Table Data

SkillID	SkillName	EmployeeID	ProficiencyLevel
1	Python	1	Expert
2	SQL	2	Intermediate
3	JavaScript	3	Advanced
4	Java	4	Expert
5	Excel	5	Beginner
6	Cloud Computing	6	Advanced
7	Data Analysis	7	Intermediate
8	Cybersecurity	8	Expert

ExperienceSalary Table Data

EmployeeID	YearsOfExperience	Salary	Bonus
1	5	80000.00	5000.00
2	7	85000.00	6000.00
3	3	70000.00	4000.00
4	10	95000.00	8000.00
5	2	65000.00	3000.00
6	8	90000.00	7000.00
7	6	75000.00	4500.00
8	4	72000.00	3500.00

Department Table Data

DepartmentID	DepartmentName	ManagerID	Location
101	IT	1	New York
102	HR	2	San Francisco
103	Marketing	3	Chicago
104	Finance	4	Boston
105	Operations	5	Seattle
106	Sales	6	Austin
107	Legal	7	Denver
108	R&D	8	Miami

Steps to connect MYSQL
Create the connection to mysql
# MySQL database connection details
url = "jdbc:mysql://localhost:3306/employeedetails <--- your databasename?useSSL=false"
user = "your username " <---username is mentioned in the desktop in the instructions file
password = " your password"<---password is mentioned in the desktop in the instructions file
Create the Pyspark code for the questions

1. What is the maximum salary offered across the organization, as retrieved by the get_max_salary function from the ExperienceSalary table?
2. How does the get_mid_level_avg_salary function calculate the average salary for mid-level employees with 3-6 years of experience?
3. How many employees have "Expert" proficiency in any skill, as determined by the get_expert_count function from the Skills table?
4. Which employee has the highest salary, as identified by the get_highest_paid_employee function from the ExperienceSalary and Employee tables?
5. Who is the employee with the most years of experience, as determined by the get_most_experienced_employee function?

6. What is the average salary across all employees, as calculated by the get_average_salary function from the ExperienceSalary table?
7. What is the total bonus payout for all employees, as summed up by the get_total_bonus function?
8. How does the get_employee_skill_department function provide a combined view of employees, their skills, and department locations by joining the Employee, Skills, and Department tables?

System Execution Flow:

1. The PySpark application connects to a MySQL database using the **JDBC connector**.
2. Data from MySQL tables (Employee, Skills, ExperienceSalary, and Department) is **loaded into PySpark DataFrames** and cached for efficient memory usage.
3. The application performs various **aggregations, filtering, and joins** using PySpark's SQL-like functions to generate insights.

Execution Steps to Follow:

1. All actions like build, compile, running application,running test cases will be through Command Terminal.

2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal

3. This editor Auto Saves the code

4. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.

5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

6. To setup environment:

You can run the application without importing any packages

7. To launch application:

Pip install pyspark

Pip install requests

python3 main.py

8. To run Test cases:

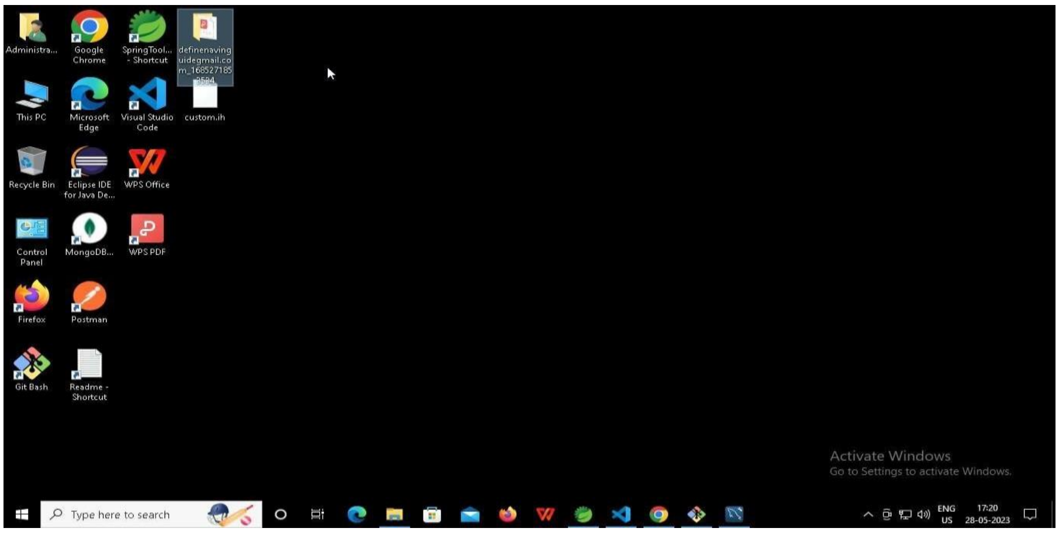
python3 -m unittest

9. Before Final Submission also, you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository for code

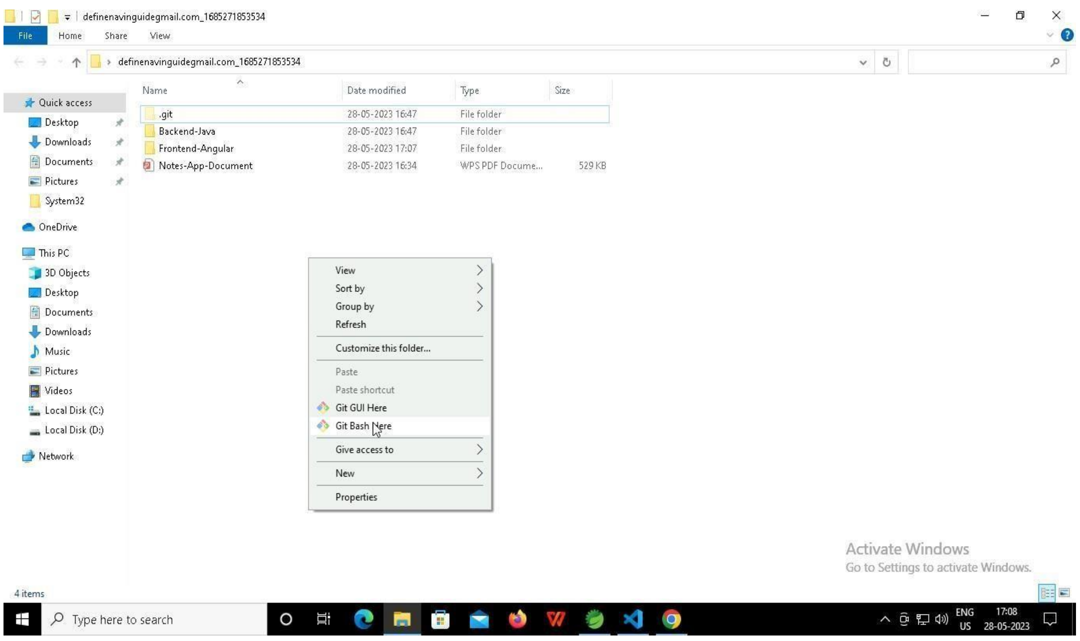
-----x-----

You can run test cases as many numbers of times and at any stage of Development, to check how many test cases are passed/failed and accordingly refactor your code.

1. **Make sure before final submission you commit all changes to git**. For that open the project folder available on desktop



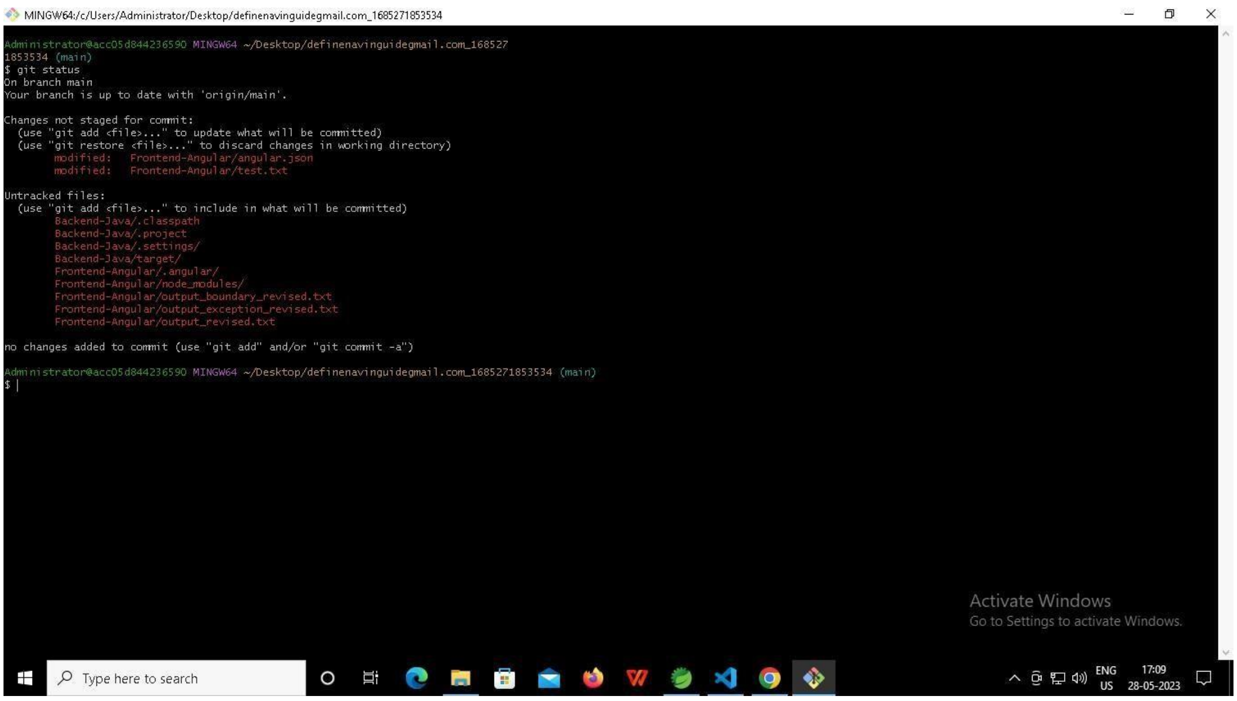
   a. Right click in folder and open Git Bash



Name	Date modified	Type	Size
.git	28-05-2023 16:47	File folder	
Backend-Java	28-05-2023 16:47	File folder	
Frontend-Angular	28-05-2023 17:07	File folder	
Notes-App-Document	28-05-2023 16:34	WPS PDF Docume...	529 KB

   b. In Git bash terminal, run following commands

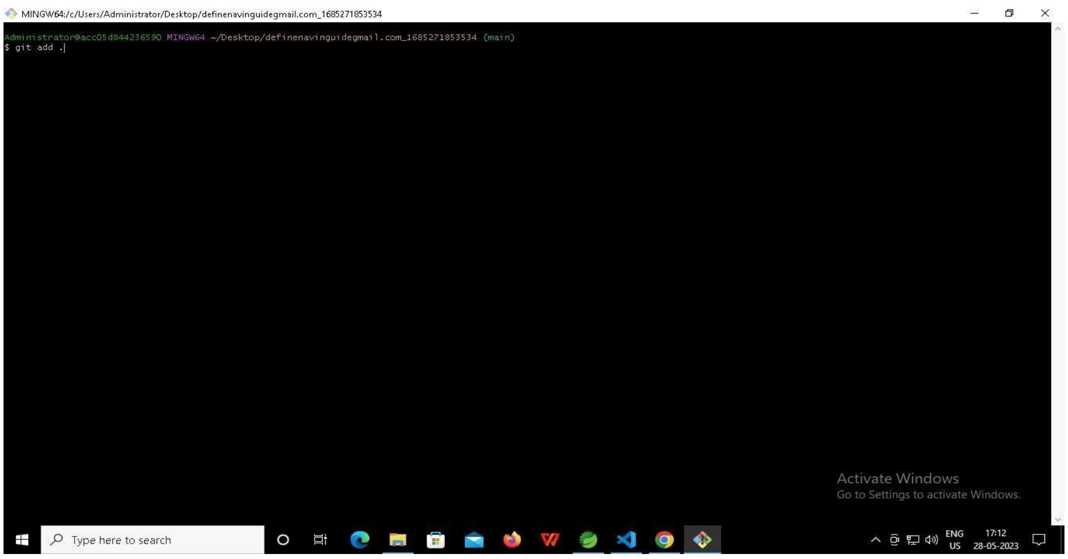
   c. git status



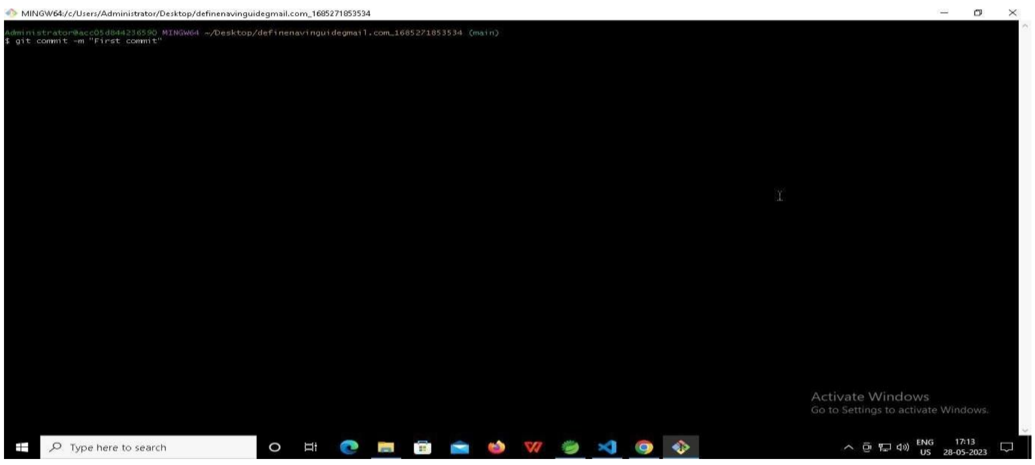
```
MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_168527
1853534 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Frontend-Angular/angular.json
        modified:   Frontend-Angular/test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Backend-Java/.classpath
        Backend-Java/.project
        Backend-Java/.settings/
        Backend-Java/target/
        Frontend-Angular/.angular/
        Frontend-Angular/node_modules/
        Frontend-Angular/output_boundary_revised.txt
        Frontend-Angular/output_exception_revised.txt
        Frontend-Angular/output_revised.txt

no changes added to commit (use "git add" and/or "git commit -a")

Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ |
```

d. git add .

e. git commit -m “First commit”
(You can provide any message every time you commit)

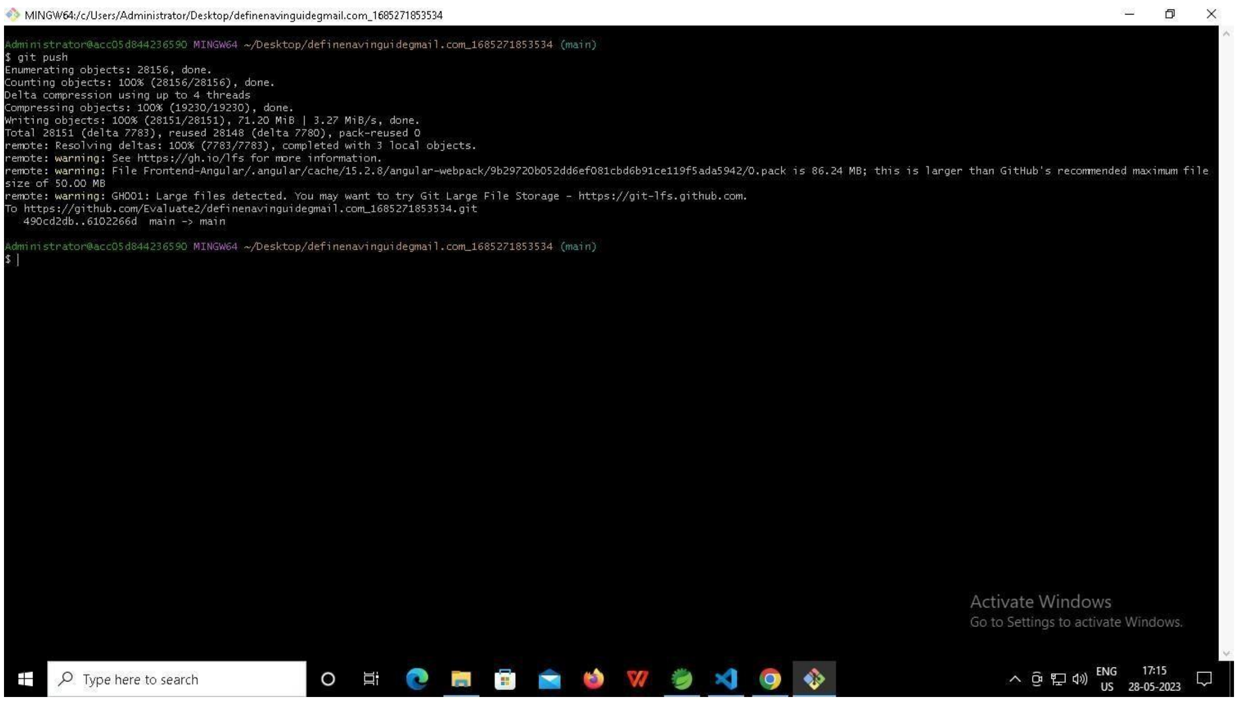


```
MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git add .
```



```
MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git commit -m "First commit"
```

f. git push



MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534

```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git push
Enumerating objects: 28156, done.
Counting objects: 100% (28156/28156), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19230/19230), done.
Writing objects: 100% (28151/28151), 71.20 MiB | 3.27 MiB/s, done.
Total 28151 (delta 7783), reused 28148 (delta 7780), pack-reused 0
remote: Resolving deltas: 100% (7783/7783), completed with 3 local objects.
remote: warning: See https://gh.io/lfs for more information.
remote: warning: File Frontend-Angular/.angular/cache/15.2.8/angular-webpack/9b29720b052dd6ef081cbd6b91ce119f5ada5942/0.pack is 86.24 MB; this is larger than GitHub's recommended maximum file
size of 50.00 MB
remote: warning: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To https://github.com/Evaluate2/definenavinguidegmail.com_1685271853534.git
   490cd2db..6102266d  main -> main

Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ |
```

-----x-----