Subject: Salary Data Processing & Analysis Statement of Work (SoW)

1. Project Overview

This project involves processing and analyzing salary data using **PySpark, Sql Server**, and **Power BI**, in line with the **Medallion Architecture**. The goal is to clean, transform, and aggregate salary data to generate valuable business insights.

The project will be completed **within 30 hours** using a **small dataset** stored in CSV files. The final output will be a **Power BI dashboard** displaying key salary metrics, such as average salary, salary distribution, and department-wise analysis.

2. Software & Tools Required

The intern will set up the following tools on their local machine:

Software	Purpose
PySpark	Data processing & transformation
Jupyter Notebook	Running PySpark scripts interactively
Sql Server	Storing transformed data & final aggregated
Power BI	Data visualization & reporting

3. Medallion Architecture Overview

The project follows the Medallion Architecture, which consists of three structured layers:

Layer	Purpose	Storage Format
Bronze	Stores raw data with minimal processing	Parquet files
Silver	Cleansed, transformed, and enriched data	Parquet files
Gold	Aggregated data for reporting & dashboards	Sql server tables

4. Data Sources & Schema

Raw Data (CSV Files) → Bronze Layer
The following CSV files will serve as raw input data, which will be ingested into the Bronze Layer (Parquet format):

salary.csv

Column Name	Type	Description
salary_id	int	Unique Salary ID (Primary Key)
employee_id	int	Foreign Key to Employee
department_id	int	Foreign Key to Department
salary_amount	float	Salary paid
salary_date	date	Date when the salary was paid
source_file	varchar	Name of the source file

employee.csv

Column Name	Type	Description
employee_id	int	Unique Employee ID (Primary Key)
employee_name	varchar	Name of the employee
job_title	varchar	Job title of the employee

department.csv

Column Name	Type	Description
department_id	int	Unique Department ID (Primary Key)
department_name	varchar	Name of the department

The CSV files will be loaded into the Bronze Layer as Parquet files for further processing.

5. Bronze Layer: Raw Data Storage in Parquet Format

The Bronze Layer stores raw CSV data in Parquet format for efficiency. Each bronze table contains the raw data plus audit columns (ingestion_date, source_file) to track data lineage.

Parquet File	Source CSV
salary_bronze.parquet	salary.csv
employee_bronze.parquet	employee.csv
department_bronze.parquet	department.csv

6. Silver Layer: Data Cleaning, Enrichment & Transformation Objective

The Silver Layer prepares cleaned, structured, and enriched data for analysis by performing:

- **Data Cleaning**: Removing records with null values in critical columns.
- **Joining Related Tables**: Adding employee names and department details.
- **Computing Additional Fields**: E.g., total_salary_value = salary_amount (salary details).
- **Storing Data in Both Parquet and MySQL for further aggregation.**

Silver Tables & Schema

1. **salary_silver.parquet → MySQL Schema**: silver_db.salary_silver
    - Operations:
        - Joining with employee_bronze and department_bronze to add meaningful descriptions.
        - Computing total_salary_value = salary_amount.
        - Removing records where salary_amount is missing.

Column Name	Type	Description
salary_id	int	Unique Salary ID
employee_name	varchar	Name of the employee
job_title	varchar	Job title of the employee
department_name	varchar	Department name
salary_amount	float	Salary amount
total_salary_value	float	Calculated salary value
salary_date	date	Date when the salary was paid
processed_date	date	Date when data was transformed

2. **employee_silver.parquet → MySQL Schema**: silver_db.employee_silver

Column Name	Type	Description
employee_name	varchar	Name of the employee
job_title	varchar	Job title of the employee

3. **department_silver.parquet → MySQL Schema**: silver_db.department_silver

Column Name	Type	Description
department_name	varchar	Name of the department

7. Gold Layer: Data Aggregation for Reporting Objective

The Gold Layer will create a final aggregated dataset in MySQL for Power BI reports, computing key business metrics.

Operations Performed in MySQL (Using Silver Layer as Source):

- **Salary Aggregation by Department**:
  - SUM(total_salary_value)
  - AVG(salary_amount)
- **Employee-wise Salary Analysis**:
  - SUM(total_salary_value) by employee.
- **Time-based Salary Trends**:
  - Aggregating salary data for trend analysis.

Gold Table Schema in MySQL (gold_db.salary_gold)

Column Name	Type	Description
department_name	varchar	Name of the department
total_salary_value	float	Total salary paid per department
avg_salary	float	Average salary per department
report_date	date	Date for time-based analysis

Source for Gold Layer:

- Silver Tables from MySQL (silver_db.salary_silver, silver_db.employee_silver, silver_db.department_silver)

8. Final Deliverables

- Silver & Gold Layers in MySQL
- Power BI Dashboard with KPIs (Salary Overview, Employee Distribution, Department Analysis)

9. Power BI Insights and Visualization Examples

Once the Gold Layer is built, the data will be pulled into Power BI to generate insights through the following visualizations:

Example:

Bar Chart:

- **Department with Max salary**
- **Salary Distribution by Job Title**
- **Top 10 Highest Paid Employees**

Line Chart

- **Total Salary Paid Over Time** (Monthly/Yearly trend)

Pie Chart

- **Employee Count by Department**

Line Chart

- **Salary Trends Over Time** (By Month/Year)

10. Project Instructions :

1.Follow the **Medallion Architecture** for data processing.
2. Use proper **naming conventions** for all target files and SSMS database tables.
3. Maintain clean and consistent coding standards in **Python scripts and SQL queries** for the Gold table.
4. Automate the entire data pipeline from raw data ingestion to the final **Gold layer**.
5. Implement proper auditing at each stage of the pipeline to track **data processing.**
6. Ensure **clear documentation** of the entire project, including transformation steps and usage
guidelines.
7. Use **GIT** to store all code, SQL scripts, and relevant files.
8. Include the **GIT repository URL** in the final documentation for submission.