Salary Data Processing & Analysis
Statement of Work (SoW)

Documentation by -

Vaishnavi Sunil Pawar

Intern - 24

Table of Contents

1- INTODUCTION

The Salary Data Processing & Analysis project is a full-cycle data engineering and analytics solution built to demonstrate how raw organizational data , specifically, salary data can be transformed into powerful, actionable insights. The project applies industry-standard data processing techniques using tools such as Apache Spark (PySpark), SQL Server, Power BI, and follows the Medallion Architecture approach for data layering.

This end-to-end pipeline simulates real-world data processing workflows using raw CSV files as the primary data source. These files are ingested and processed through various stages of refinement, eventually feeding into a Power BI dashboard for reporting and business intelligence.

1.1 – PROJECT OBJECTIVE

The goal of this project is to:

- Ingest, clean, and transform raw salary data.
- Aggregate data to derive key salary-based insights.
- Create a robust, modular data pipeline that follows best practices in modern data architecture.
- Deliver interactive and visual dashboards using Power BI for decision-making.

The final outcome includes:

- Department-wise salary aggregations
- Average and total salary calculations
- Monthly trends in salary payments
- Top-paid employees and job role distributions

1.2 – PROBLEM STATEMENT

In many organizations, salary data is stored in multiple files and formats across systems. These files may contain duplicates, null values, inconsistent formats, and lack meaningful aggregations. Without a structured data pipeline, HR and Finance teams struggle to derive actionable insights from the data.

This project addresses these issues by:

- **Ingesting** data from a centralized source.
- **Processing** it in stages to improve data quality.
- **Structuring** it for analytical workloads.
- **Visualizing** trends, patterns, and anomalies that drive decision-making.

1.3 – SCOPE AND DURATION

- **Tools & Technologies Used:**
    - **PySpark:** Data ingestion and transformation
    - **SQL Server:** Storage of cleansed and aggregated data
    - **Power BI:** Data visualization and business intelligence reporting
    - **Git:** Source control for tracking code and pipeline versions
    -
- **Duration:** Completed within **30 hours**

- **Dataset:** Small-scale salary-related CSV files stored locally or in blob storage simulation.

1.4 – BASIC CONCEPTS

This project uses a combination of cloud storage, big data processing, and business intelligence tools to create an end-to-end data pipeline. Key concepts include:

ETL (Extract, Transform, Load): the process of extracting data from source systems, transforming and refining it to satisfy specific requirements, and finally sending it to a specified destination for further use.

Data pipeline: an automated operation that executes and coordinates data transformation and transportation across multiple systems seamlessly.

Parquet: Parquet is a columnar data format optimized for storing and accessing data efficiently, specifically tailored to handle large-scale data workloads effectively.

Azure: Microsoft's cloud platform offering cloud computing and storage services.

Databricks: a cloud computing and analytics platform. Databricks provisions compute for data processing, provides user interface for data operations.

Azure Data Factory (ADF): is one of the services in Azure cloud platform. ADF is used to extract data from source systems and orchestrate data pipelines.

Power BI: a business analytics tool developed by Microsoft that allows users to visualize data, share insights, and create interactive dashboards and reports.

PySpark: A python API for Apache Spark, enabling Python developers to leverage the power of Spark for large-scale distributed data processing, machine learning, and real-time analytics.

Azure Blob Storage-Cloud storage to store raw input CSV files, simulating the Bronze layer in the pipeline.

Azure Data Lake Storage (ADLS Gen2)-Provides a structured, scalable way to store data across Bronze, Silver, and Gold layers (simulated locally in this project).

Medallion Architecture-

Data is processed in three layers:

- Bronze: Raw data
- Silver: Cleaned and enriched data
- Gold: Aggregated data for reporting

Jupyter Notebook- Used for writing and executing PySpark code interactively.

Git- Version control system used to manage code and track changes throughout the project.

2- Medallion Architecture Overview

The Salary Data Processing & Analysis project is designed following the Medallion Architecture, a layered data engineering approach that improves data quality and readiness for business reporting through three main layers: Bronze, Silver, and Gold.

The pipeline ensures each transformation step is modular, auditable, and optimized for performance.

Data Flow Overview:

Raw CSV files are initially stored in the raw container of Azure Blob Storage. These files are then processed and written to Azure Data Lake Storage Gen2 (ADLS Gen2) in Parquet format, categorized into the Bronze layer. The data is cleaned, enriched, and written to the Silver layer in both Parquet and MySQL. Finally, the Gold layer in SQL Server holds aggregated data for reporting in Power BI.

Architecture Layers-

Layer	**Purpose**	**Storage Format**
Bronze	Stores raw data with minimal processing	Parquet (ADLS Gen2)
Silver	Cleansed, enriched, and transformed data	Parquet + MySQL
Gold	Aggregated data for dashboards and reporting	SQL Server Tables

Detailed Layer-wise Description

Bronze Layer:

- Ingests raw CSV data from the raw container of Azure Blob Storage
- Converts data into **Parquet format**
- Adds metadata like ingestion_date and source_file
- Stores the data into **bronze container** in ADLS Gen2
- No data cleaning is done at this stage

Silver Layer:

- Reads Parquet files from the Bronze layer
- Cleans null/missing data from important fields

- Performs transformations like joins with employee and department tables
- Adds business-relevant fields like employee_name, department_name, and total_salary_value
- Saves the output to **silver container** in Parquet format and **MySQL database** for analytics

Gold Layer:

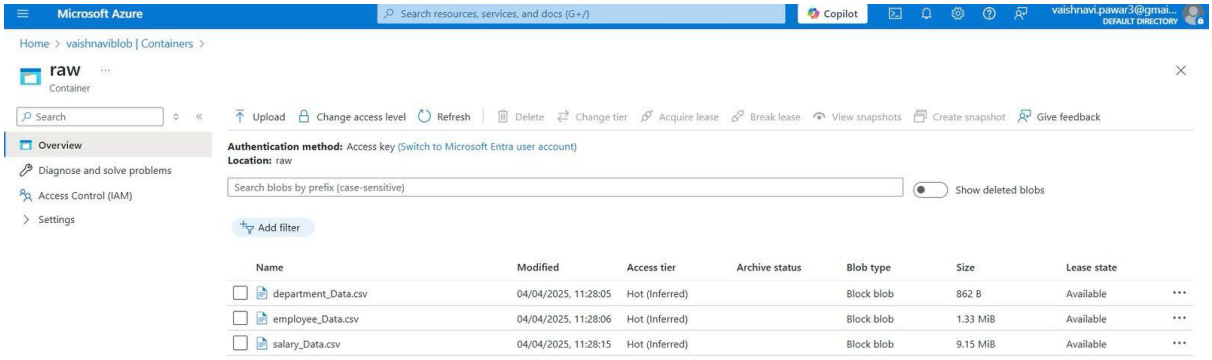
- Reads cleaned and enriched data from **MySQL Silver tables**
- Performs aggregations for reporting (e.g., average salary, total salary by department)
- Stores results into **MySQL tables under gold_db schema**
- Used directly by Power BI for dashboarding and reporting

Layer-Wise Flow Description:

Raw Layer (Azure Blob Storage - Raw Container)

- **Purpose**:
  - Initial storage of unprocessed CSV files from the source systems.
  - Acts as the landing zone for raw datasets.
- **Details**:
  - Stored in the **"raw" container** of Azure Blob Storage.
  - Files are in **CSV format**.
  - No transformation or validation is performed here.
- **Example Files**:
  - employee_data.csv
  - department_data.csv
  - salary_data.csv

Figure- Source data -- Azure blob storage [raw container with csv files]



Bronze Layer (Azure Data Lake Storage Gen2 - Bronze Container)

- **Purpose**:
  - Store minimally processed data.
  - Standardize file format for efficient processing.
- **Process**:
  - Raw CSV files are **read into PySpark**.
  - Basic schema validation and type casting is done.
  - Then the data is written to **Parquet format**, a columnar storage optimized for analytics.
- **Storage**:
  - Files are written to the **"bronze" container** in **Azure Data Lake Gen2**.

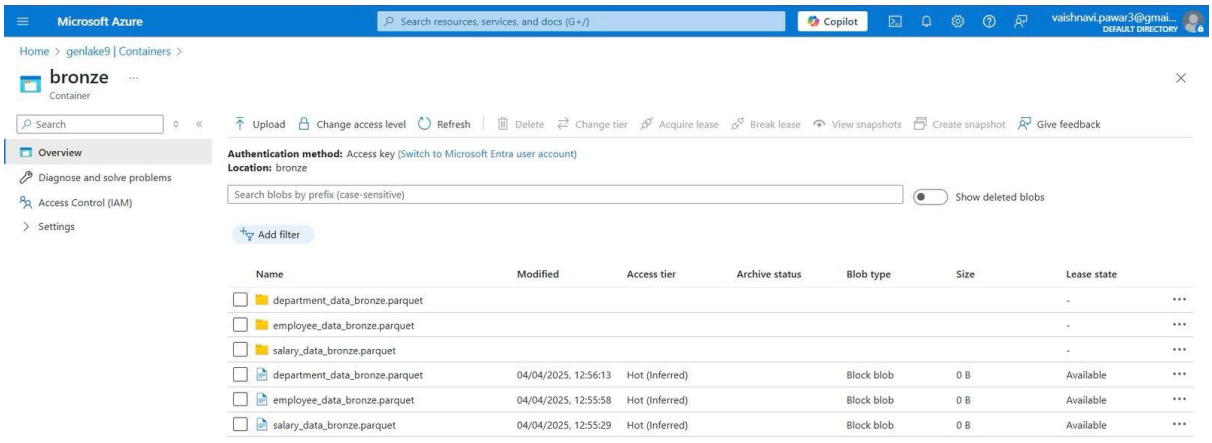


Figure – Bronze container

Silver Layer (Azure Data Lake Gen2 - Silver Container)

- **Purpose**:
  - Clean, filter, join, and enrich the data.
  - Remove duplicates, handle nulls, and standardize fields.
- **Process**:
  - Perform joins between salary, employee, and department tables.
  - Apply business logic like deduplication and filtering invalid/null values.
  - Cast columns to the appropriate data types.
  - Add derived columns (e.g., processed_date).
  - Identify latest salary record per employee using **Window functions**.
- **Storage**:
  - The resulting refined datasets are stored again in **Parquet format** in the **"silver" container** of Azure Data Lake.

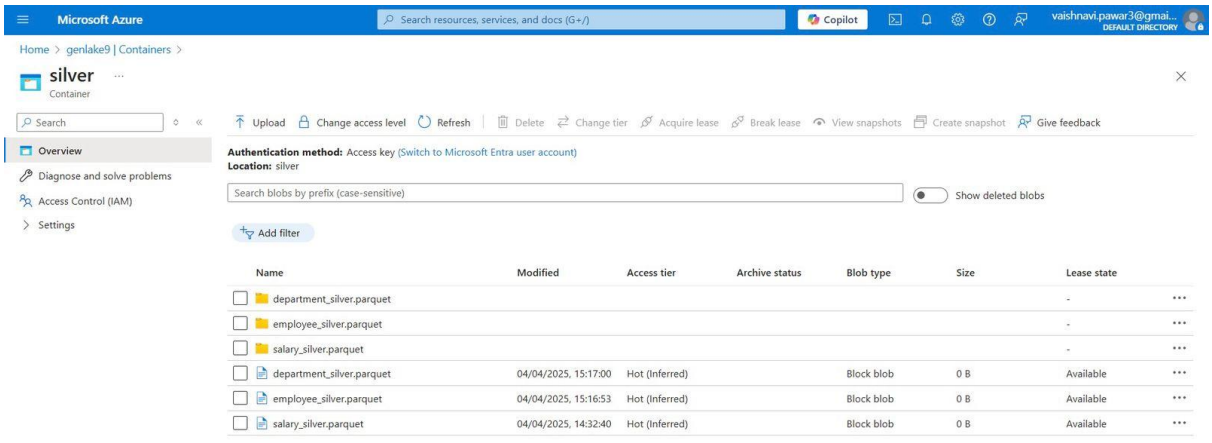
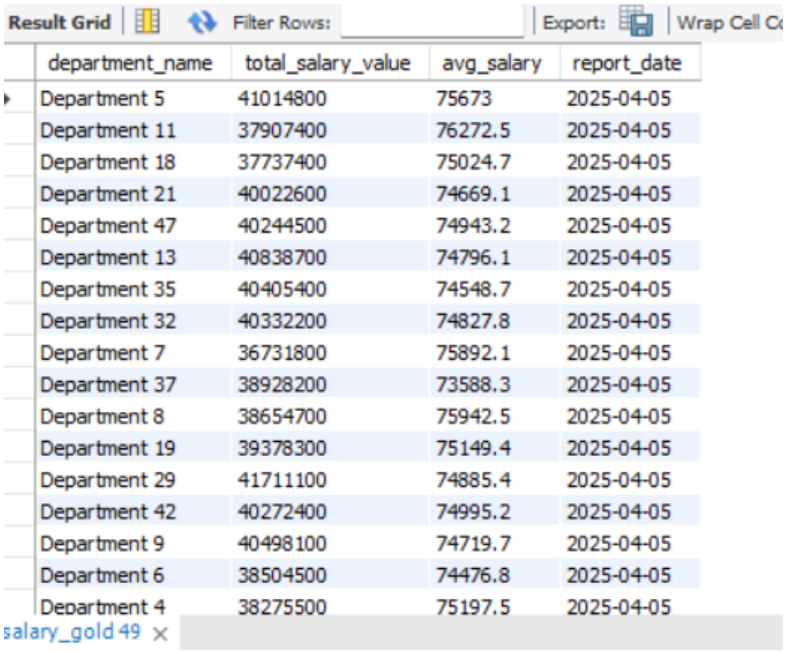


Figure - Silver container

Gold Layer (SQL Server / Azure SQL Database)

- **Purpose**:
  - Final aggregation and reporting-ready data.
  - Used directly for business dashboards.
- **Process**:
  - Load the final salary_silver DataFrame into SQL Server using JDBC.
  - Tables are structured for consumption by BI tools (like Power BI).
  - May include KPIs, department-wise salary summaries, monthly trends, etc.
- **Storage**:
  - Stored in relational tables in **SQL Server database**.

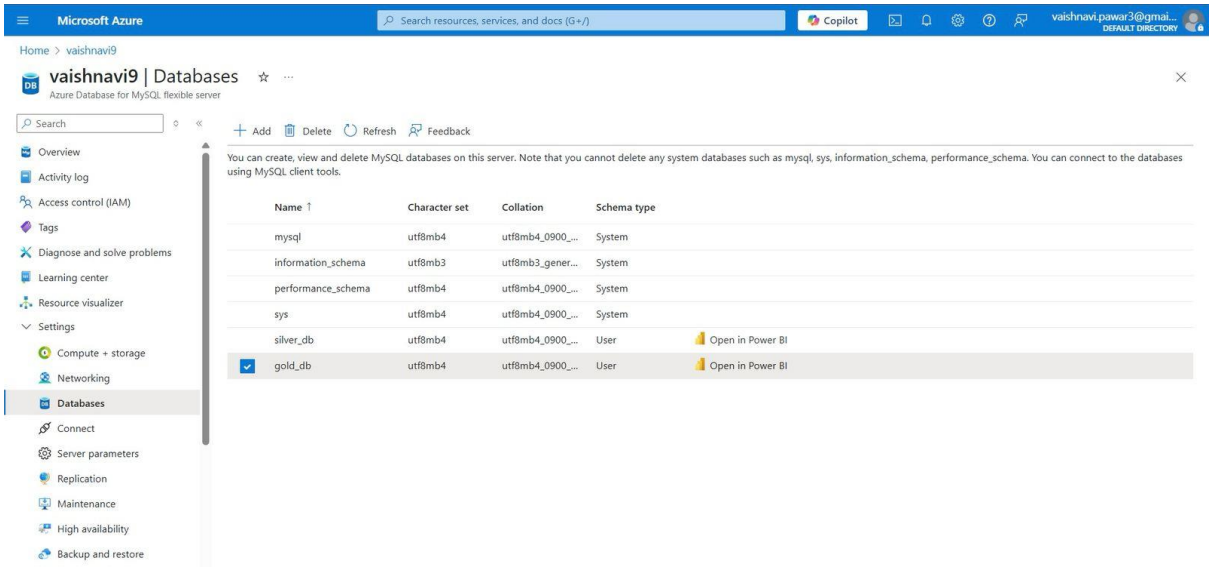

Result Grid | Filter Rows: | Export: | Wrap Cell C

department_name	total_salary_value	avg_salary	report_date
Department 5	41014800	75673	2025-04-05
Department 11	37907400	76272.5	2025-04-05
Department 18	37737400	75024.7	2025-04-05
Department 21	40022600	74669.1	2025-04-05
Department 47	40244500	74943.2	2025-04-05
Department 13	40838700	74796.1	2025-04-05
Department 35	40405400	74548.7	2025-04-05
Department 32	40332200	74827.8	2025-04-05
Department 7	36731800	75892.1	2025-04-05
Department 37	38928200	73588.3	2025-04-05
Department 8	38654700	75942.5	2025-04-05
Department 19	39378300	75149.4	2025-04-05
Department 29	41711100	74885.4	2025-04-05
Department 42	40272400	74995.2	2025-04-05
Department 9	40498100	74719.7	2025-04-05
Department 6	38504500	74476.8	2025-04-05
Department 4	38275500	75197.5	2025-04-05

salary_gold 49 ×

Figure - table salary_gold from database gold_db

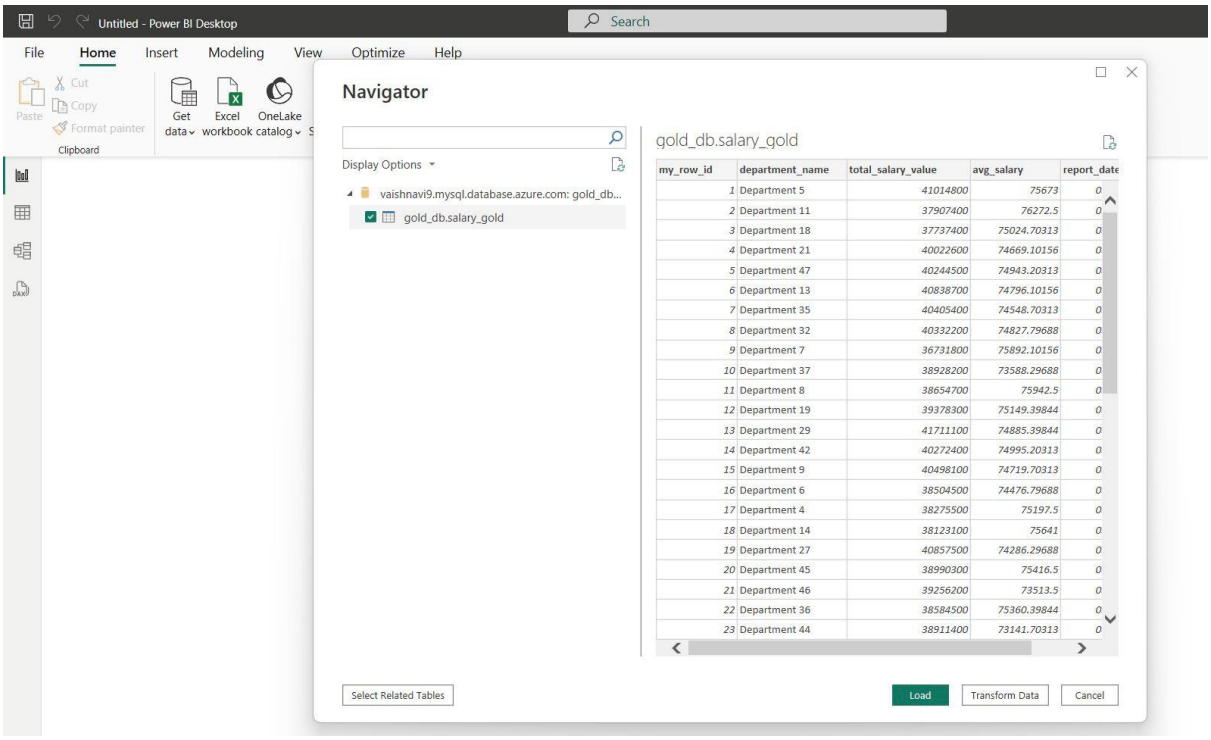
Figure - Downloading (.pbids) file of the database from Azure Databases for MySQL flexible servers



Reporting Layer (Power BI)

- **Purpose**:
  - Visualize key insights from the Gold layer.
- **Visuals in Power BI**:
  - Average Salary by Department
  - Salary Distribution Histogram
  - Salary Trends over Time
  - Top 10 Highest Paid Employees
  - Department-wise Headcount and Salary Analysis
- **Connection**:
  - Power BI connects to SQL Server (Gold Layer) using **ODBC or DirectQuery**.

Figure - The downloaded gold_db.pbids file loaded into power bi for visualization.



Storage Overview

Layer	Purpose	Location	Format
Raw	Store unprocessed data	Azure Blob Storage (raw container)	CSV
Bronze	Standardized raw data	Azure Data Lake Gen2 (bronze container)	Parquet
Silver	Cleaned and enriched data	Azure Data Lake Gen2 (silver container)	Parquet
Gold	Final aggregated, reporting data	SQL Server Database	Relational
Reports	Dashboard and business intelligence	Power BI	Visual/Live

3- Data Set Description [Data Sources & Schema]

This project utilizes three raw CSV files—salary.csv, employee.csv, and department.csv—as input data. These files are initially stored in the **raw container of Azure Blob Storage**. From there, they are ingested and stored in **Parquet format** within the **Bronze Layer** of **Azure Data Lake Storage Gen2 (ADLS Gen2)**, setting the foundation for further processing and transformation through the Medallion Architecture.

Data Files Description:

1. salary.csv

This file contains salary payment records and acts as a transactional dataset.

Column Name	Data Type	Description
salary_id	int	Unique identifier for each salary record
employee_id	int	References the employee receiving salary
department_id	int	References the department of the employee
salary_amount	float	Salary paid
salary_date	date	Date of salary payment
source_file	varchar	Name of the source file (for traceability)

2. employee.csv

This file provides employee details such as their names and job titles.

Column Name	Data Type	Description
employee_id	int	Unique identifier for employee
employee_name	varchar	Full name of the employee
job_title	varchar	Designation/job title

3. department.csv

This file holds information about various departments within the organization.

Column Name	Data Type	Description
department_id	int	Unique ID for each department
department_name	varchar	Name of the department

These files are transformed and cleaned through the Bronze to Silver layer process, which removes duplicates, fills missing values, and joins related data from different sources. The final enriched datasets are then aggregated and stored in the Gold Layer, making them ready for visualization and reporting via Power BI.

4- Final Solution Implementation (End-to-End)

The solution follows the Medallion Architecture using the Bronze, Silver, and Gold Layer concept. Each layer progressively transforms the data for quality, enrichment, and business-readiness, implemented using Azure Databricks, Azure Storage, Azure Data Factory, MySQL, and Power BI.

Step 1: Environment Setup and Azure Resource Provisioning

To begin with, multiple Azure services were configured and connected to establish the data pipeline ecosystem:

- **Azure Storage Accounts**:
    - **Storage Account 1 (Blob Storage):** Used for uploading and hosting raw .csv files.
    - **Storage Account 2 (Azure Data Lake Gen2):** Used for storing transformed parquet files (Bronze & Silver Layers).
- **Azure Databricks Workspace** was created and configured:
    - A compute **cluster** was initiated.
    - Databricks was **mounted** to both storage accounts to enable reading and writing datasets using secure access tokens.
- **Azure SQL Database** was provisioned to host **Silver and Gold layer tables** for reporting and aggregation.

- **Azure Data Factory (ADF)**:
    - Created an ADF **pipeline** to automate the orchestration of transformation workflows.
    - Two Databricks notebooks were integrated into this pipeline:
        - raw_to_bronze
        - bronze_to_silver
    - Pipeline was tested and validated for successful execution end-to-end.

Step 2: Bronze Layer – Raw Data Ingestion

The **Bronze Layer** was implemented to store raw .csv data in an optimized and queryable **Parquet format**.

- Each .csv file (salary, employee, department) was converted to a .parquet file with additional **audit columns** such as:
    - ingestion_date (to track when the data was ingested)
    - source_file (to trace the origin of the data)
- The objective of this layer was to preserve raw fidelity while enabling efficient reads and future transformations.

Step 3: Silver Layer – Data Cleaning, Deduplication, and Enrichment

The **Silver Layer** focused on cleaning, enriching, and structuring data for analysis. This included removing inconsistencies, deduplicating records, and preparing the data for business use.

- **Cleaning Operations**:
    - Removed records containing null values in critical columns such as salary amount.
- **Deduplication**:
    - Employee and department tables were deduplicated based on unique identifiers to ensure data quality.
- **Column Pruning**:
    - After deduplication, unnecessary identifier columns (e.g., employee_id, department_id) were dropped to simplify the schema for downstream use.

- **Enrichment and Joining**:
  - The salary dataset was enhanced by joining it with employee and department data to add employee names and department descriptions.
  - A new field total_salary_value was computed to reflect salary insights.

- **Output Storage**:
  - The cleaned and transformed data was stored in both **Parquet format** (in Data Lake) and **MySQL database** (for analytical querying and Power BI consumption).

Step 4: SQL Integration for Silver & Gold Layers

To enable structured querying and business reporting:

- **Databricks was connected to MySQL** using a **JDBC Maven driver**.
- The Silver layer datasets were written into MySQL using this JDBC connection.
- SQL Workbench was used to verify and query the inserted tables.
- This provided the foundation for aggregation and analysis in the **Gold Layer**.

Step 5: Gold Layer – Aggregation for Business Reporting

In the **Gold Layer**, business-level aggregations were performed using SQL queries on the Silver layer tables in MySQL.

- **Key Aggregations** included:
  - Total and average salary by department
  - Salary distribution by employee
  - Time-based salary trends for reporting
- The resulting table, salary_gold, was structured to include:
  - Department name
  - Total salary value
  - Average salary
  - Report date (used for time-based analysis)

This final dataset was structured and optimized for **dashboard creation in Power BI**.

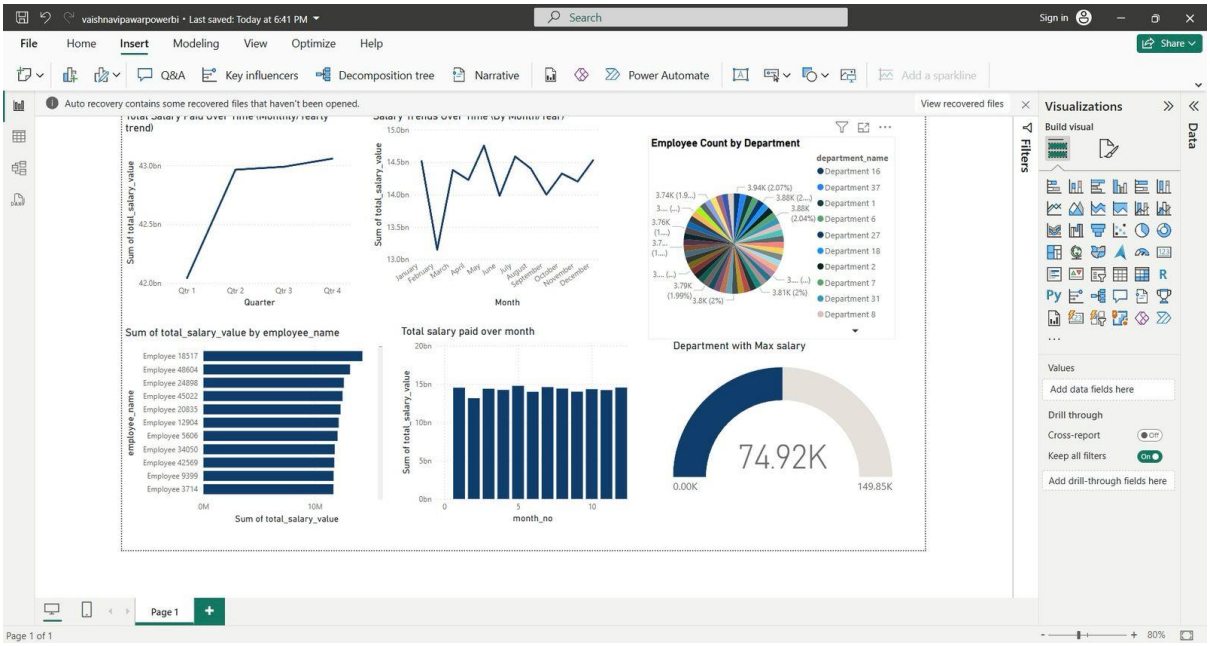
Step 6: Dashboard Development Using Power BI

Once the Gold Layer was created and stored in MySQL, the data was connected to Power BI using the MySQL connector. An interactive dashboard was developed to extract business insights and support strategic decision-making. The dashboard contains multiple KPIs and visuals that highlight key trends and patterns related to salary and employee data. Below are the key components included in the Power BI report:

The key performance indicators (KPIs) displayed include the **total salary paid**, **average salary per department**, **top paying department**, **employee count by department**, and the **top five highest paid employees**. These KPIs provide a quick snapshot of the organization's financial and workforce dynamics.

The report features several insightful visualizations. **Bar charts** are used to showcase the department with the highest total salary paid, salary distribution by job title, and the top 10 highest paid employees. **Line charts** illustrate salary trends over time, including total salary paid monthly or yearly, and department-wise salary growth. A **pie chart** displays the distribution of employees across various departments, offering a clear view of organizational structure. Additionally, **slicers and filters** are incorporated to allow users to dynamically explore the data by department, job title, or salary date. These visuals collectively support better decision-making by providing both a macro and micro view of organizational salary trends.

Figure- Final Dashboard with visuals



7. Pipeline Automation with Azure Data Factory (ADF)

To automate the data transformation flow from raw .csv files to business-ready Parquet and SQL tables, a **data pipeline** was created using **Azure Data Factory (ADF)**. The orchestration enables **end-to-end automation** of the data processing lifecycle across the **Bronze and Silver Layers**.

7.1 - Notebooks Used in Pipeline

Two Azure Databricks notebooks were created, developed, and integrated into the ADF pipeline to handle different stages of the data transformation process:

1. Notebook Name: raw_to_bronze

- **Function:** Ingest raw .csv files from Azure Blob Storage and convert them into .parquet format.
- **Input Source:** salary.csv, employee.csv, department.csv from **Blob Storage**.
- **Output Target: Bronze Layer Parquet files** stored in **Azure Data Lake Gen2**.
- **Additional Logic:** Added audit columns like ingestion_date and source_file to maintain data lineage and traceability.

2. Notebook Name: bronze_to_silver

- **Function:** Perform data cleaning, deduplication, enrichment, and join operations on Bronze data.
- **Input Source:** Bronze Layer Parquet files from **Azure Data Lake Gen2**.
- **Output Target:**
    - **Silver Layer Parquet files**
    - **MySQL Database** tables (salary_silver, employee_silver, department_silver)
- **Additional Logic:**
    - Removed null values from key fields.
    - Dropped identifier columns after deduplication.
    - Joined salary data with employee and department tables.
    - Calculated new field total_salary_value.

7.2 Azure Data Factory Pipeline Components

The pipeline consists of the following activities:

7.3 Pipeline Flow -

1. **Step 1**: raw_to_bronze notebook is called.
    - Reads raw .csv files from Blob Storage.
    - Converts and writes them as Parquet files to the Bronze layer in Data Lake.
2. **Step 2**: Upon successful completion, the bronze_to_silver notebook is triggered.
    - Reads Bronze Parquet files.
    - Cleans, deduplicates, joins, and enriches the data.
    - Writes final Silver Parquet files and inserts cleaned data into the MySQL database.
3. **Step 3**: Once Silver Layer is ready in MySQL, Gold aggregations are triggered via SQL for Power BI dashboard consumption.

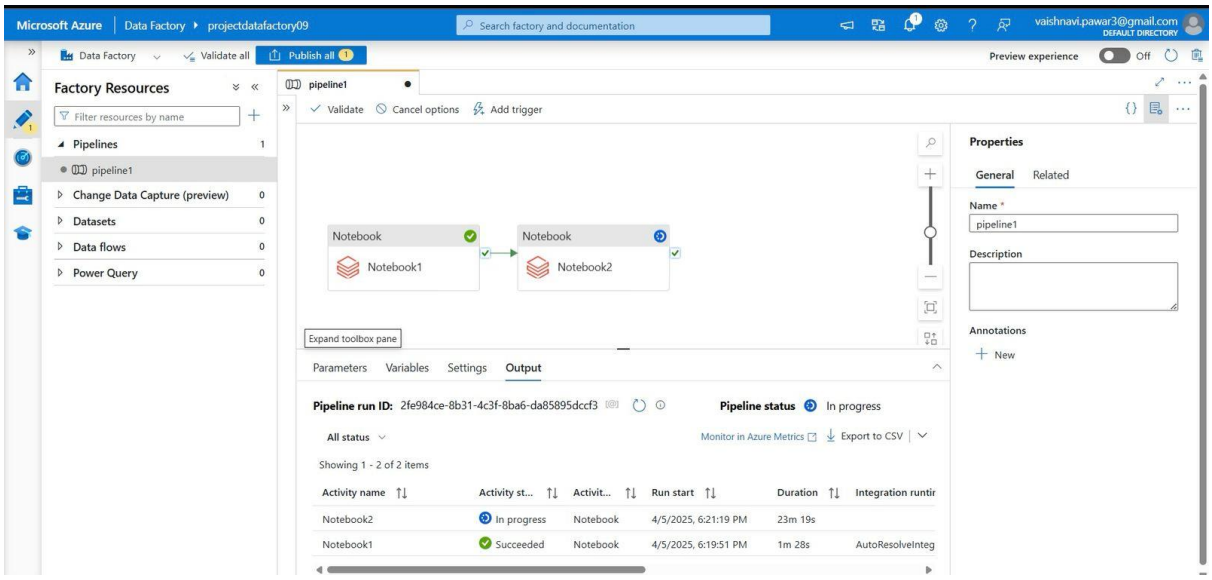


Figure: Orchestration Pipeline Using Azure Data Factory and Databricks Notebooks

5- Git Repository

The entire project implementation, including notebooks, configuration scripts, and documentation, is available on GitHub:

🔗 **Repository Name**: https://github.com/CodeByVaishnavi/Salary-Data-Processing-Analysis-Statement-of-Work-SoW-

6- Challenges & How I Solved Them

While working on this project, I came across several practical and technical challenges, especially since it involved working across cloud platforms, big data processing, and dashboard integration. Each obstacle was a great learning moment and pushed me to dig deeper and find effective solutions.

Some of the main challenges I faced:

- Figuring out which Azure regions actually support MySQL Flexible Server (surprisingly, not all do).
- Connecting MySQL Flexible Server to MySQL Workbench to run and verify queries.
- Encountering issues while writing data into MySQL due to missing JDBC connectors.
- Facing Power BI connection errors when trying to fetch data from MySQL.
- Data not showing up in the Gold layer as expected, especially salary records — due to incorrect joins or dropped columns during transformations.

How I tackled them:

- I went through updated Azure documentation and community forums to shortlist regions where MySQL Flexible Server was supported.
- Followed online tutorials and troubleshooting guides to successfully connect Workbench with the cloud-hosted MySQL instance.
- Installed the required MySQL JDBC driver manually on the Databricks cluster to resolve dependency issues.
- To fix the Power BI error, I installed the appropriate .NET connector to enable stable connectivity with MySQL.
- Spent time debugging the Spark joins and column filtering logic to ensure salary data appeared correctly in the Silver and Gold layers.

7- Learnings

This was one of my first real-world data engineering projects and it definitely helped me apply everything I've been learning. It gave me exposure to modern data architectures, cloud-based tooling, and the complete lifecycle of data — right from ingestion to final insights.

Key learnings:

- Understood the Bronze-Silver-Gold layered data architecture and how it simplifies handling raw to refined data.
- Got hands-on experience building PySpark pipelines for ingestion, transformation, and filtering.
- Learned how to connect Databricks with MySQL and write logic using Spark SQL and MySQL for generating business metrics.
- Realized the importance of keeping schemas and joins intact during data cleaning and transformation.
- Built an interactive Power BI dashboard to visualize KPIs from the Gold layer, and saw how dashboards can drive clarity in decision-making.

This project gave me both the technical exposure and confidence to approach real-life data scenarios with a structured mindset.

8- Conclusion

This project helped me bring together multiple tools and technologies to build an automated data pipeline based on the Medallion architecture. Starting with raw CSV data in Azure Blob Storage, I was able to transform, refine, and deliver business-ready insights through Databricks and Power BI.

8.1 What I Achieved

- Ingested employee, department, and salary data from Azure Blob to the Bronze layer.
- Performed cleaning, joining, and business logic transformation in Silver and Gold layers.
- Connected the Gold data to MySQL and used Power BI for dashboarding.
- Set up an end-to-end data pipeline that could be automated and reused.

8.2 Why It Matters

This solution can be applied in real-world HR or payroll use cases. For example:

- HR teams can track top-paid employees and salary distribution.
- Finance teams can review monthly or departmental salary trends.

- Leadership teams can use dashboards to get quick insights and take informed decisions based on real data.

8.3 My Takeaways

- I learned how to build and manage modern data pipelines on the cloud.
- Faced and solved actual integration challenges, which made the learning deeper.
- Saw the full journey of raw data becoming meaningful business intelligence.
- Understood how clear documentation, version control, and automation matter in any professional data project.

Overall, this project strengthened both my technical and problem-solving skills. More importantly, it taught me how to think like a data engineer and approach challenges with a calm, solution-oriented mindset.