End-to-End

Data Engineering

Solution for HR Analytics

Midterm Report

CSIS 4495 – 050

*Jay Clark Bermudez | 300380540 (Team Leader)*

*Bruno do Nascimento Beserra | 300392300*

*Matheus Filipe Figueiredo | 300389657*

October 23, 2025

## Contents

[1 Introduction 3](#_Toc212135374)

[2 Proposed Research Project 4](#_Toc212135375)

[2.1 Project Goal 4](#_Toc212135376)

[2.2 Methodology 4](#_Toc212135377)

[2.3 Technical Requirements 6](#_Toc212135378)

[3 Project Planning and Timeline 7](#_Toc212135379)

[3.1 Phase Overview and Milestones 7](#_Toc212135380)

[3.1.1 Requirement Analysis and Planning 7](#_Toc212135381)

[3.1.2 System Design and Modeling 7](#_Toc212135382)

[3.1.3 Implementation and Development 8](#_Toc212135383)

[3.1.4 Testing and Validation 8](#_Toc212135384)

[3.1.5 Deployment, Visualization, and Documentation 8](#_Toc212135385)

[3.2 Weekly Work Plan and Schedule 8](#_Toc212135386)

[3.3 Roles and Responsibilities 9](#_Toc212135387)

[4 Implementation of the Dataset 10](#_Toc212135388)

[4.1 Introduction to the dataset creation 10](#_Toc212135389)

[4.2 Details of Implementation 10](#_Toc212135390)

[5 Implementation of Data Engineering Pipeline 12](#_Toc212135391)

[5.1 Introduction of the Data Engineering Pipeline 12](#_Toc212135392)

[5.2 Details of Implementation 12](#_Toc212135393)

[5.2.1 Bronze Layer 12](#_Toc212135394)

[5.2.2 Silver Layer 12](#_Toc212135395)

[5.2.3 Gold Layer 12](#_Toc212135396)

[6 Implementation of Power BI 14](#_Toc212135397)

[6.1 Introduction/Overview of the feature Power BI 14](#_Toc212135398)

[6.2 Details of Implementation 14](#_Toc212135399)

[7 Implementation of Web Application 16](#_Toc212135400)

[7.1 Overview of Web Application 16](#_Toc212135401)

[7.2 Shift from Django to Dash Framework 16](#_Toc212135402)

[7.3 Initial Web Application using Dash and Plotly 16](#_Toc212135403)

[7.4 Future Updates 17](#_Toc212135404)

[8 Work Hours 18](#_Toc212135405)

[9 Acknowledgement 22](#_Toc212135406)

[10 References 23](#_Toc212135407)

[11 Appendix 25](#_Toc212135408)

[11.1 Screenshots of Datasets Implementation 25](#_Toc212135409)

[11.2 Screenshots of Data Engineering Pipeline Implementation 26](#_Toc212135410)

[11.3 Screenshots of Power BI Implementation 28](#_Toc212135411)

[11.4 Screenshots of Web Application Implementation 31](#_Toc212135412)

## 1 Introduction

This project looks at a challenge with Dayforce, a SaaS platform used to manage HR data like employee information and payroll. The problem is that Dayforce does not keep historical records. When an employee leaves, their data is deleted, and when updates are made, older records are replaced. This makes it hard to do historical analysis, track workforce trends, or study issues like employee turnover (Dayforce, 2024). This is not just a Dayforce issue, but a common limitation with HR SaaS platforms (Solutions, 2025).

Platform3 Solutions (Solutions, 2025) notes that not keeping payroll and HR records can lead to compliance issues, problems during audits, and even legal trouble. They stress that companies need a clear plan for keeping and archiving data so it stays available when needed and costs stay under control.

Research shows that without strong historical archives, companies struggle with workforce planning and decision-making (Madden, 2025). Using data engineering techniques like Slowly Changing Dimensions Type 2 and platforms like Databricks and Delta Lake can fix this problem by allowing data to be captured, stored, and analyzed over time (WJARR, 2025).

To solve this problem, the project will build a data pipeline that automatically collects, processes, and saves historical HR data. The pipeline will run in Databricks, using Python and PySpark for transformations and Delta Lake for reliable storage. On top of this, a web app will be built with Django and React to show the results of the analysis.

The finished system will help organizations keep and study HR history in a more efficient way. It will make storage use better, allow faster queries, support long-term workforce analysis, and improve decision-making by giving insights that are not possible with the current setup.

## 

## 2 Proposed Research Project

### 2.1 Project Goal

The goal of this project is to design and implement an end-to-end data engineering solution that preserves and enables analysis of historical HR data. The project addresses a key limitation in current HR data management, where employee records are deleted after termination or overwritten when changes occur, making historical analysis impossible.

To solve this, the team will develop a data pipeline that ingests daily data from Dayforce and applies data engineering techniques such as Slowly Changing Dimensions (Type 2) to track changes over time, the medallion architecture to structure data into quality layers, and Kimball data modelling to reduce redundancy and simplify queries.

The solution will be built in Databricks using Python, PySpark, and Delta Lake, and will be paired with a Power BI dashboard that highlights key workforce metrics. Together, these components will demonstrate how historical HR records can be effectively preserved, organized, and analyzed to support long-term workforce insights.

### 2.2 Methodology

As outlined in the project goal, this study will develop an end-to-end data engineering solution to preserve and manage historical HR data. In the corporate world, businesses typically obtain their HR data from platforms such as Dayforce or other human resource management systems. When reports are needed, this data is often exported as a CSV file and analyzed using visualization tools. Since our team does not have direct access to such corporate data, we will use a Kaggle dataset containing information on 2 million employees (Kaggle, 2025), from which we will extract a subset as our sample dataset for this project. This data will be updated monthly, allowing us to track trends over a period of seven years. The pipeline will follow the medallion architecture (Databricks, 2020), with Bronze, Silver, and Gold layers ensuring both quality and traceability.

Figure 2.1 System Architecture

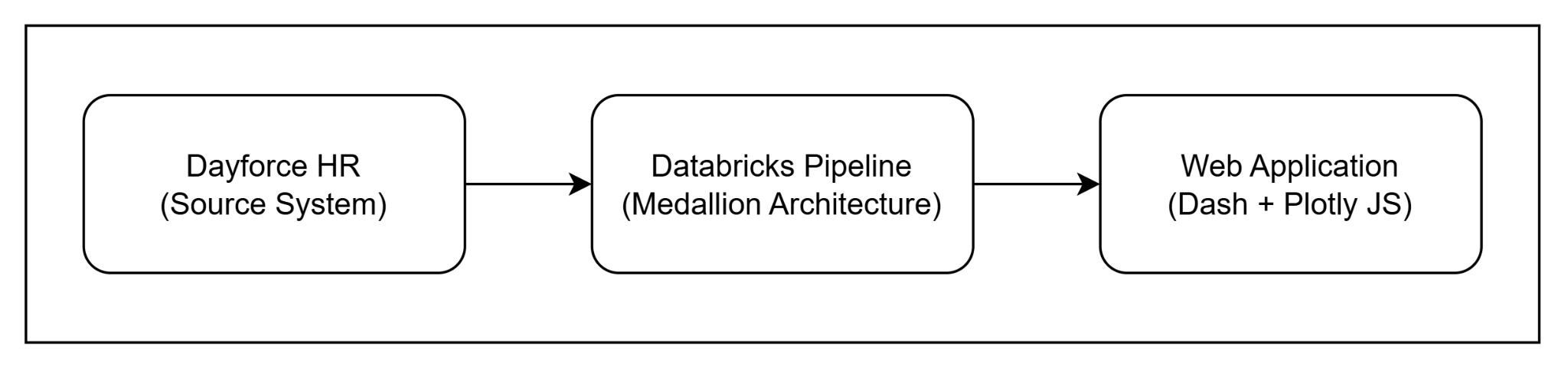


Figure 2.1 shows the overall system architecture. There are three (3) major stages. The raw data from Dayforce will be processed through a Databricks pipeline. Finally, one of the outputs of the system is a CSV file, which will then go through the web application for visualization.

Figure 2.2 Data Engineering Pipeline



Figure 2.2 shows the data pipeline utilizing Databricks. The process begins with the ingestion of raw employee data from Dayforce (Dayforce, 2020) into the Bronze layer, capturing daily snapshots of the system. Data is then stored in a parquet format. The Silver layer will apply Slowly Changing Dimensions Type 2 (Asanka, 2021) to track historical changes, such as promotions, transfers, or terminations. Data cleaning will address missing values, duplicates, and inconsistent formats, while verifying key identifiers such as employee IDs. Each record will include timestamps and active/inactive flags to maintain historical accuracy.

The Gold layer will structure data for analysis using Kimball modelling (Nguyen, Pham, and Chin, 2020), creating fact and dimension tables to reduce redundancy and simplify queries. Implementation will be carried out in Databricks notebooks with Delta Lake features such as incremental ingestion, schema enforcement, and merge operations to ensure consistency.

Additionally, the output from Gold layer can be readily used for any analytics reports that can be utilized by Tableau, PowerBI, other business needs, and the custom web application developed by the team. This web application will serve as the primary interface for end users to access and explore the visuals.

Figure 2.3 Web Application Architecture

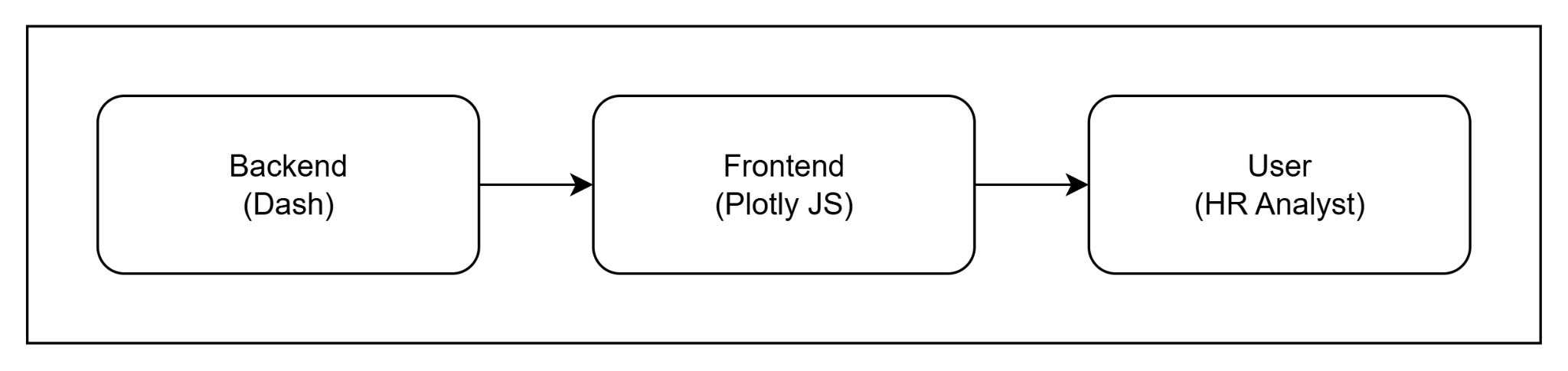


Figure 2.3 presents the architecture of the updated web application that works alongside the output from the data engineering pipeline. The initial design used Django for the backend and ReactJS for the frontend. However, the team decided to transition to Dash, which integrates both backend and frontend functionalities within a single Python framework. This change simplifies development, reduces the need for multiple technologies, and allows smoother interaction with the datasets produced by the Gold layer of the pipeline.

The new application will be developed using Dash and Plotly, which together support interactive, data-driven visualization. Dash manages data handling and user interaction, while Plotly generates dynamic charts and graphs directly from the processed CSV files of the Gold layer. This setup makes it possible to display real-time analytics without relying on separate web technologies.

The dashboards will present key HR metrics such as employee turnover, salary trends by department, experience versus performance, and departmental headcount distribution. Users will be able to filter results, compare values, and explore data through interactive charts, helping managers and analysts identify workforce patterns and make evidence-based decisions.

Development will follow best practices such as version control in Git (Microsoft, 2022) and thorough documentation. This methodology ensures the solution is robust, maintainable, and scalable, providing a framework that supports both current and future HR data analysis needs.

### 2.3 Technical Requirements

* **Data Platform:** Databricks
* **Data Pipeline and Processing:** Delta Lake, Python PySpark
* **Data Storage:** Delta Lake Bronze, Silver, Gold layers
* **Data Modelling:** Kimball dimensional modelling, Slowly Changing Dimensions (Type 2)
* **Backend:** Dash, Python
* **Frontend:** Plotly JS
* **Version Control:** Git

## 3 Project Planning and Timeline

### 3.1 Phase Overview and Milestones

Figure 3.1 Gantt Chart

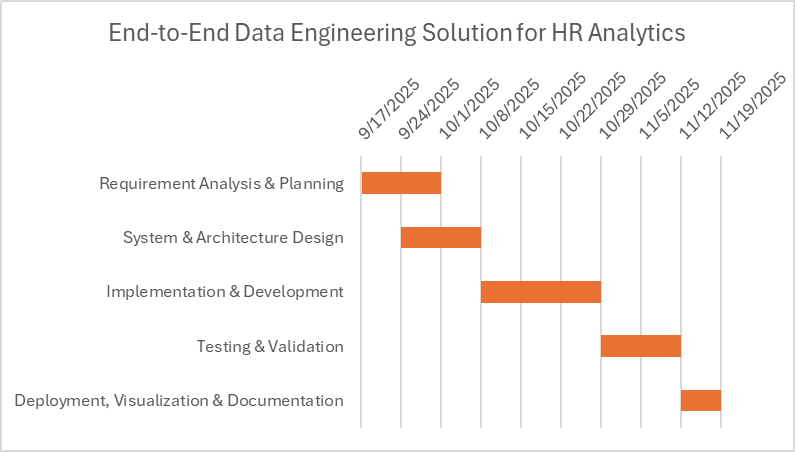


Figure 3.1 shows the Gantt chart, which outlines the five phases of the project: Requirement Analysis, System Design, Implementation, Testing, and Deployment, along with their timelines and dependencies. It shows how the earlier phases provide the foundation for development and testing, which ensures readiness for final deployment.

#### 3.1.1 Requirement Analysis and Planning

The team will assess Dayforce’s data limitations, focusing on record loss after updates or employee exits. From these observations, requirements such as daily ingestion, historical record preservation, and trend analysis will be defined. The scope will outline Databricks, PySpark, Delta Lake, and Django as core tools, with acceptance criteria based on accurate ingestion, retention, and longitudinal querying.

#### 3.1.2 System Design and Modeling

The system will be designed using the medallion architecture with Bronze, Silver, and Gold layers. SCD Type 2 logic will be specified to capture employee history, and Kimball-style modelling will define fact and dimension tables for analysis. Deliverables include schema definitions and a system design document. Additionally, the web application design will be planned, with Dash handling backend processes and Plotly JS enabling interactive dashboards for visualization.

#### 3.1.3 Implementation and Development

A Databricks pipeline will be built to ingest simulated Dayforce data into the Bronze layer, transform and apply SCD Type 2 in the Silver layer, and organize fact/dimension tables in the Gold layer. Delta Lake features such as incremental ingestion, schema enforcement, and merge operations will ensure reliability. The web application will be developed with the backend processing data and the frontend displaying HR metrics such as headcount trends and payroll history.

#### 3.1.4 Testing and Validation

Testing will cover both the pipeline and the web application. Unit tests will confirm that ingestion and transformation processes work correctly, while integration tests will ensure data flows smoothly across all layers. User acceptance testing (UAT) will check that historical records are preserved accurately and that analysis tasks return correct results. The web application will also be tested for responsiveness and accuracy of visualizations.

#### 3.1.5 Deployment, Visualization, and Documentation

The pipeline will be deployed in Databricks and connected to a custom web application for dashboards displaying metrics such as workforce trends, employee retention and turnover, and other key metrics. Final deliverables include technical documentation, versioned code in Git, a project report, and a presentation of the solution.

### 3.2 Weekly Work Plan and Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Duration | Due Date | Key Deliverables |
| Requirement Analysis & Planning | 2 weeks | Oct 1 | * Approved project proposal * Draft use case diagram * Initial dataset collection and exploration |
| System Design & Modeling | 2 weeks | Oct 8 | * System architecture design * Data engineering pipeline design * Web application architecture design * UI/UX design * Draft test cases * Set up of GitHub repository * Set up Databricks workspace |
| Implementation & Development | 3 weeks | Oct 29 | * Databricks ingestion and transformation pipeline * Lakehouse configuration for Bronze, Silver, Gold layers * Backend development * Frontend development * Unit testing of the pipeline and components |
| Testing & Validation | 2 weeks | Nov 12 | * Test execution * Regression testing for pipeline and web app * Bug tracking and resolution * Test summary report |
| Deployment, Visualization & Documentation | 1 week | Nov 19 | * Final pipeline deployment * Web application deployment * Project documentation * Final project presentation and demo |

### 3.3 Roles and Responsibilities

Jay Clark Bermudez - Project Manager & Back-end Developer

* Coordinate team meetings, progress, and ensure deadlines are met
* Serve as the main point of communication between the team, instructor, and other stakeholders
* Design and implement the backend system and APIs for the web application

Bruno do Nascimento Beserra - Data Engineer

* Extract, clean, and transform raw HR data for pipeline integration
* Implement the medallion architecture
* Collaborate with the backend developer to align data processing with the system requirements

Matheus Filipe Figueiredo - Front-end Developer & QA/Documentation

* Build the user interface for visualizing reports and analytics
* Develop and execute test cases to validate functionality and performance
* Track and document issues, prepare a user guide, and finalize project documentation

## 

## 4 Implementation of the Dataset

### 4.1 Introduction to the dataset creation

In order to showcase the result of our pipeline, we needed an effective way to present it. Since we did not have access to a dataset containing historical data over time, we decided to design a realistic simulation environment that captures key workforce dynamics throughout multiple years. This approach allows us to demonstrate our data pipeline solution built with state-of-the-art techniques.

Initially, we considered performing an analysis using past data. However, after start developing the code, we realized that this approach would add unnecessary complexity to the project, as it would require making hiring dates pass through a complex historical logic. To simplify the process while maintaining realism, we instead created a simulation that represents the evolution of a mid-sized company with 5,000 employees over seven years.

Throughout these seven years, employees may experience promotions, change teams, or leave the company. In parallel, the organization continuously hires new employees, using information from the main dataset to maintain a dynamic workforce.

To build this simulation, we defined a set of key metrics to guide the algorithm responsible for generating time-based snapshots. The final output is a structured folder hierarchy following the pattern **“data/year/month/file.csv”**, effectively mimicking an automated process that fetches daily data from a source.

### 

### 4.2 Details of Implementation

To build this part of the project, we defined a series of steps to complete the process. First, we imported the necessary libraries and the dataset. From there, we began by examining its basic information through an Exploratory Data Analysis (EDA), checking aspects such as data types, null value counts, shape, and statistical summaries. Once we gained a clear understanding of the dataset, we proceeded to the data cleaning stage, where we dropped unnecessary columns, corrected unrealistic values in the annual\_salary column, and fixed the data types across the dataset.

After cleaning the data, we created the first snapshot by generating a random sample of 5,000 rows. We used a fixed random seed to ensure reproducibility and added a new column, Job Level, based on each employee’s years of experience. Before finalizing this version of the dataset, we verified that the distribution of employees by job level followed a realistic organizational structure, with fewer employees in higher-level positions, similar to a real-world company.

To generate the historical snapshots, we defined several key metrics representing possible events over time within the company. These included salary raise percentages, monthly hiring rates, and promotion milestones to capture scenarios where an employee changes departments, leaves the company, earns a promotion, or receives a performance-based bonus. Using these parameters, we generated 84 monthly snapshots of the dataset, simulating seven consecutive years of company history, from October 2025 to September 2032.

## 

## 5 Implementation of Data Engineering Pipeline

### 5.1 Introduction of the Data Engineering Pipeline

The data engineering pipeline was implemented using state-of-the-art techniques. To build it, we used Databricks, a widely recognized and commonly used data platform in the industry. We took this project as an opportunity to learn how to use the platform effectively. However, the free version of Databricks presented some limitations, such as not allowing multiple users to collaborate within the same workspace. To address this constraint, each team member worked in a separate workspace, and we synchronized our progress by updating the files through GitHub.

To ensure our data pipeline was robust, we applied several modern techniques, including the Medallion Architecture and Kimball Architecture, with plans to implement Slowly Changing Dimension Type 2 (SCD Type 2) for the final presentation. Based on our research, these methodologies provide data professionals with a clearer and more stable workspace, where data is organized in layers and enriched with metadata to track changes over time. This structured approach allows updates to occur only when necessary, minimizing data duplication and maintaining a clean, consistent environment.

### 5.2 Details of Implementation

Based on the Medallion Architecture, we divided our pipeline into three main folders, which will be described in more detail in the following sections.

#### 5.2.1 Bronze Layer

The bronze layer is where the pipeline starts; its main goal is to fetch the data from the source and read it to import into the bronze layer. Depending on the dynamic of our source, it has a different form to access the data. For our case, we are mimicking it coming from some folder that updates every day, so we run through the CSV files, read them, and append them to our created table.

#### 5.2.2 Silver Layer

In the silver layer, we make our pipeline in a way that we first check our silver data to check what the last data available in it is, and then it only reads the data that it still doesn’t have from bronze. Then, we do a data cleaning in it, renaming columns, checking null values or inconsistencies in the data and implementing the SCD type 2 (in the future). In our pipeline, we already implemented the metadata columns for the dataset.

#### 5.2.3 Gold Layer

Here, we want the data to look exactly like what’s used in our reports and analytics tools. The data has already been cleaned and standardized in the Silver layer; the only step left is to create dimension and fact tables. This makes the data intuitive for business users to understand and query, while also making it easier to add new data marts in the future using the same structured and organized information. The Kimball architecture also helps the data load faster and improves analytics performance. Over time, it’s increasingly adopted because it reduces reliance on IT for every new report or dashboard.

## 

## 6 Implementation of Power BI

### 6.1 Introduction/Overview of the feature Power BI

For the midterm, we implemented an interactive HR analytics dashboard using Power BI. This feature was designed to visualize and analyze employee data sourced from our gold layer, which was modelled using Kimball’s dimensional architecture. The main objective was to provide a clear and intuitive interface for exploring key workforce metrics and trends.

The dashboard includes multiple layers of analysis, starting with a general overview of the organization’s employee distribution, average salary, and experience. It then progresses into more detailed trend-based visualizations, such as performance ratings over time, hiring patterns by year, and salary progression based on experience. These insights are made accessible through interactive filters and slicers that allow users to drill down by department, job level, and work mode.

This implementation marks a shift from static reporting toward dynamic, user-driven exploration. It reflects our focus on usability and business relevance, ensuring that HR stakeholders can quickly access and interpret the data they need to support strategic decisions.

### 6.2 Details of Implementation

The implementation process began with importing and cleaning the HR dataset directly into Power BI. This step involved organizing key fields such as job level, department, salary, experience, and hire year to ensure consistency and accuracy across all visuals.

But we decided to change after we got the Gold Layer working, so we created a new POWER BI with basically the same visualizations but now with more accurate stuff.

That said, the data was sourced from the gold layer, which had been structured using Kimball’s dimensional modelling approach, allowing for efficient relationships between fact and dimension tables.

Once the data model was established, a series of visuals was created to represent different aspects of the workforce. These included bar charts to show employee distribution by job level and department, line charts to track salary trends over time, pie charts to illustrate employment status breakdowns, and column charts to analyze performance ratings and hire year patterns. Each visual was designed to highlight specific HR metrics while maintaining a clean and readable layout.

Some filters and slicers were added to enable users to explore the data dynamically. These controls allowed for quick, making it easier to uncover targeted insights.

We also decided to use a custom blue color scheme throughout the dashboard to enhance visual consistency and improve readability.

The result is a professional and user-friendly dashboard that supports HR teams in analyzing workforce trends, identifying patterns, and making informed decisions with minimal effort.

## 

## 7 Implementation of Web Application

### 7.1 Overview of Web Application

The web application serves as the visualization layer of the HR data pipeline, allowing users to interactively explore workforce insights such as employee turnover, salary distribution, and performance trends. It functions as a prototype dashboard that supports data-driven decision-making through dynamic filtering and visual analytics.

Initially, the team planned to use Power BI for the visualization component. However, considering the project’s scope and the objective of demonstrating end-to-end development, the team chose to build a custom web application instead. This approach provided greater flexibility, deeper integration with the data pipeline, and enhanced control over functionality and design.

### 7.2 Shift from Django to Dash Framework

During the project proposal, the team initially planned to implement the web application using the Django framework for backend development and ReactJS with D3.js for frontend visualization. However, after evaluating the project’s requirements and overall scope, the team concluded that using Django would be excessive for the intended functionality. As a result, the team decided to shift to Dash, a Python-based framework designed for building interactive web applications and dashboards.

Additionally, Dash is highly compatible with PlotlyJS, which our professor strongly recommended and which the team chose to use instead of D3.js for creating data visualizations. Dash also integrates smoothly with Jupyter and other Python notebooks, enabling seamless incorporation into the team’s existing data analysis workflow.

This transition aligned with the project’s main objective of emphasizing data visualization and analysis rather than full-stack web development, making Dash a more efficient, flexible, and suitable framework for the project’s goals.

### 7.3 Initial Web Application using Dash and Plotly

To evaluate the functionality and layout of the web application, the team is currently using a sample HR dataset instead of connecting directly to Databricks. This dataset includes key employee attributes such as department, job title, salary, experience, and performance rating, closely resembling the structure of the processed data from the data pipeline.

The application was fully developed within Jupyter Notebook using several core Dash and Plotly components:

* Plotly Express - to create interactive visualizations such as bar charts, line graphs, and scatter plots.
* Dash Core Components and Dash HTML Components - to design the application layout and interactivity.
* Callbacks - to enable real-time updates and dynamic filtering based on user input.

The initial version of the dashboard included the following visualizations:

1. Employee Turnover Rate by Department
2. Salary Trend by Department
3. Experience vs. Performance Relationship

These visuals were designed to test and validate the dashboard’s structure, interactivity, and usability before integrating it with live data from Databricks. More dashboards will be added on the later updates.

### 7.4 Future Updates

Future iterations of the web application will connect directly to Databricks, enabling real-time visualization of HR metrics stored in the Gold layer of the Medallion Architecture. This integration will replace static CSV files with dynamic data sources, ensuring that the dashboard always reflects the most recent workforce analytics.

However, integrating Dash with Databricks introduces several technical challenges:

1. Environment Limitations - Databricks notebooks have restricted support for running interactive web servers such as Dash.
2. Port Accessibility - the platform does not natively allow the open HTTP ports required for hosting Dash applications externally.
3. Authentication and Security - establishing secure connections to Databricks SQL Warehouses or REST APIs requires proper handling of authentication tokens and cluster permissions.

To address these challenges, future development may involve deploying Dash externally (i.e. on a local device) and connecting it to Databricks through the Databricks SQL Connector. This setup would create a scalable architecture in which Databricks handles data processing while the Dash application functions as an independent visualization service.

## 

## 8 Work Hours

**Student Name: Bruno do Nascimento Beserra**

|  |  |  |
| --- | --- | --- |
| **Date** | **Number of Hours** | **Description of Work Done** |
| 09/26/2025 | 4 | Study General Databricks Concepts:  ● Platform  ● Workspaces  ● Notebook  ● Delta Lake |
| 09/27/2025 | 2.5 | Study General Databricks Concepts:  ● Clusters  ● Job  ● ABFSS Paths |
| 09/28/2025 | 3.5 | Create Folders and Delta Lakes in Databricks Workspace |
| 09/29/2025 | 1.5 | Review ETL and ELT Concepts for the project’s Data Flow |
| 10/01/2025 | 1.5 | Review Medallion Architecture concepts |
| 10/01/2025 | 1 | Set Databricks workspace to comply with Medallion Architecture |
| 10/03/2025 | 2.5 | Start working with Notebook for the project sample dataset in Databricks |
| 10/04/2025 | 1.5 | Finish creation of the first dataset and decide parameters for historical analysis |
| 10/05/2025 | 0.5 | Export Dataset from Databricks to Github |
| 10/09/2025 | 0.5 | Export Python Notebook of dataset creation from Databricks to Github |
| 10/09/2025 | 2.5 | Create Progress Report 2 Document |
| 10/09/2025 | 0.5 | Uploaded code of initial dataset creation |
| 10/11/2025 | 2 | Started working on snapshots logic |
| 10/12/2025 | 4 | Finished snapshots logic and created 84 snapshots of data |
| 10/15/2025 | 2.5 | Created coding for bronze layer of the data engineering pipeline |
| 10/17/2025 | 2 | Started coding for silver layer of the data engineering pipeline |
| 10/18/2025 | 2.5 | Finished working in the silver layer of the data engineering pipeline |
| 10/19/2025 | 0.5 | Updated description of bronze and Silver notebooks |
| 10/19/2025 | 1.5 | Helped Matheus developing the gold layer of the data engineering pipeline |
| 10/22/2025 | 3.5 | Add and update Dataset Creation and Data Engineering parts for Bronze and Silver in Midterm Report |

**Student Name: Jay Clark Bermudez**

|  |  |  |
| --- | --- | --- |
| **Date** | **Number of Hours** | **Description of Work Done** |
| 09/27/2025 | 0.5 | Update System Architecture to changes |
| 09/27/2025 | 2.5 | Create use case, context and data flow diagrams |
| 09/28/2025 | 2 | Initial data cleaning and exploration for the sample dataset, create sample visuals, and discover additional visuals that our group will implement |
| 09/30/2025 | 3 | Learn and explore Plotly Dash (new framework to be used instead of Django) through Linkedin Learning, YouTube, and community forums |
| 10/3/2025 | 4 | Continue learning about Dash from Udemy Course (Sections 1, 2, 3) |
| 10/4/2025 | 1 | Setup Jupyter and Dash in personal device |
| 10/4/2025 | 3 | Generate sample visuals for our dataset in Dash |
| 10/6/2025 | 2.5 | Continue learning about Dash from Udemy Course (Sections 4, 5) |
| 10/10/2025 | 2 | Continue learning about Dash from Udemy Course (Sections 6) |
| 10/10/2025 | 3 | Create initial dashboards:   * Bar Chart of Turnover Rate by Department * Line Chart Average Salary by Department * Scatterplot of Experience vs Performance |
| 10/13/2025 | 2.5 | Continue learning about Dash from Udemy Course (Sections 7) |
| 10/18/2025 | 2 | Update dashboard design to be interactive |
| 10/19/2025 | 2 | Learn Databricks from a YouTube video, crash courses, and Databricks documentation |
| 10/22/2025 | 1.5 | Set up Databricks workspace and initial attempt to connect Databricks to Notebook |
| 10/22/2025 | 2 | Add and update Web Application parts in the Midterm Report |
| 10/23/2025 | 2 | Double-check and finalize the Midterm documentation for submission |

**Student Name: Matheus Filipe Figueiredo**

|  |  |  |
| --- | --- | --- |
| **Date** | **Number of Hours** | **Description of Work Done** |
| 10/05/2025 | 0.5 | Initial look on the dataset, set up the POWER BI. |
| 10/06/2025 | 1.5 | Searched how to work with POWER BI. |
| 10/06/2025 | 1.25 | Initial work on the POWER BI (some minor labs for practice) |
| 10/07/2025 | 0.25 | Created the README for the POWER B |
| 10/08/2025 | 3.5 | Created the first version of the POWER BI Presentation |
| 10/08/2025 | 0.25 | Uploaded the initial POWER BI file and the README into GitHub. |
| 10/08/20250 | 0.10 | Made some minor changes in the POWER BI |
| 10/09/2025 | 0.25 | Prepared the Progress Report number 2 |
| 10/09/2025 | 0.20 | Uploaded the Progress Report number 2 to the GitHub |
| 10/12/2025 | 2.5 | Initial Study on the Databricks logic and website. |
| 10/18/2025 | 1.5 | Created the Report file, adding the sections based on the template on BlackBoard |
| 10/18/2025 | 2.5 | Created the Notebook to create the dataset for the gold layer |
| 10/19/2025 | 3.5 | Updated the Notebook to create the dataset for the gold layer |
| 10/19/2025 | 2.5 | Created a New Power BI using the new dataset |
| 10/21/2025 | 1 | Edited the POWER BI section (4.0) in the Midterm Report |
| 10/22/2025 | 1 | Add and update Data Engineering part for Gold Layer in Midterm Report |

## 

## 9 Acknowledgement

We would like to extend our heartfelt appreciation to **Dr. Bambang Sarif** for his unwavering support, encouragement, and thoughtful guidance throughout our project journey. His feedback and mentorship greatly helped us refine our ideas, strengthen our proposals, and stay motivated from the initial concept to the midterm updates.

## 

## 10 References

Asanka, D. (2021). *Implementing Slowly Changing Dimensions (SCDs) in Data Warehouses.* SQLShack.<https://www.sqlshack.com/implementing-slowly-changing-dimensions-scds-in-data-warehouses/>

Databricks. (2020). *Medallion Architecture.* Databricks.<https://www.databricks.com/glossary/medallion-architecture>

Dayforce. (2020). *Dayforce Main Website.*<https://www.dayforce.com/>

Dayforce. (2024). *Personal Data Retention Policies.*<https://help.dayforce.com/r/ImplementationGuide/Dayforce-Implementation-Guide/Personal-Data-Retention-Policies>

D3 by Observable. (2020). *What Is D3?* Observable.<https://d3js.org/what-is-d3>

Django Software Foundation. (2018). *Django: The Web Framework for Perfectionists With Deadlines.*<https://www.djangoproject.com/>

HRBrain. (2024). *HR SaaS Platform Analytics and Reporting.*<https://hrbrain.ai/blog/hr-saas-platform-analytics-and-reporting/>

Kaggle. (2025). *HR Dataset (Multinational Company).* Kaggle.<https://www.kaggle.com/datasets/rohitgrewal/hr-data-mnc>

Madden, S. (2025). *Combating Turnover Using People Analytics.* ScottMadden.<https://www.scottmadden.com/insight/combating-turnover-using-people-analytics/>

Microsoft. (2022). *What Is Git?* Microsoft Learn.<https://learn.microsoft.com/en-us/devops/develop/git/what-is-git>

Nguyen, H., Pham, H., & Chin, C. (2020). *The Analytics Setup Guidebook.* Holistics.<https://www.holistics.io/books/setup-analytics/>

Platform3 Solutions. (2025). *Payroll Data Retention Management: How to Do It Right.*<https://platform3solutions.com/blog/solution-for-keeping-payroll-records-data-retention-compliance/>

W3Schools. (2017). *React Introduction.* W3Schools.<https://www.w3schools.com/react/react_intro.asp>

World Journal of Advanced Research and Reviews (WJARR). (2025). *Developing Scalable HR Analytics Platforms for SMEs With SCD2.*<https://journalwjarr.com/sites/default/files/fulltext_pdf/WJARR-2025-2920.pdf>

## 

## 11 Appendix

### 11.1 Screenshots of Datasets Implementation

Figure 11.1.1 Sample of the dataset from Kaggle

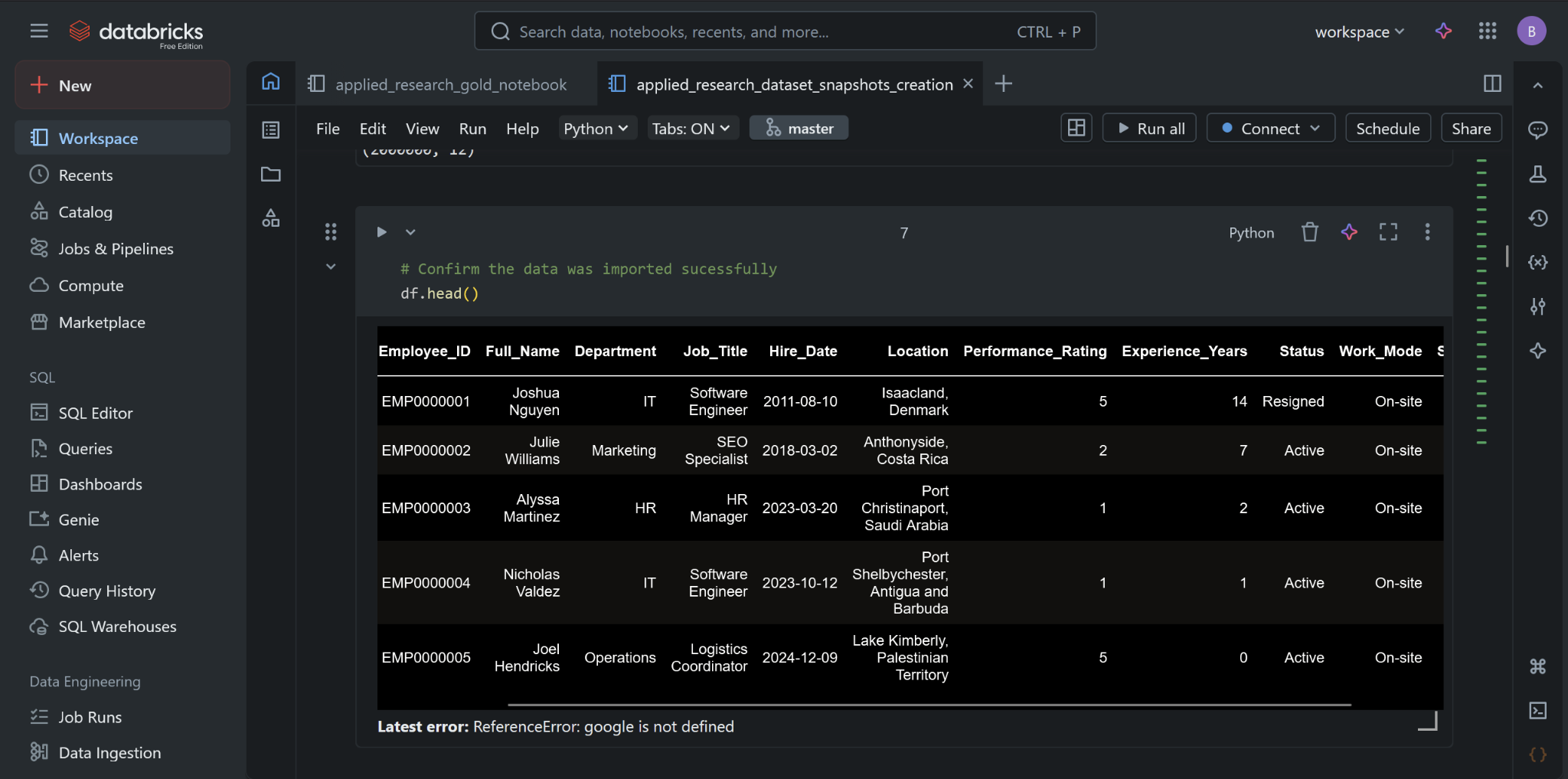


Figure 11.1.1 shows a sample of the dataset we download from Kaggle, the dataset was not cleaned at first and some columns had weird values.

Figure 11.1.2 Creation of Job Level inside the company based in years of experience

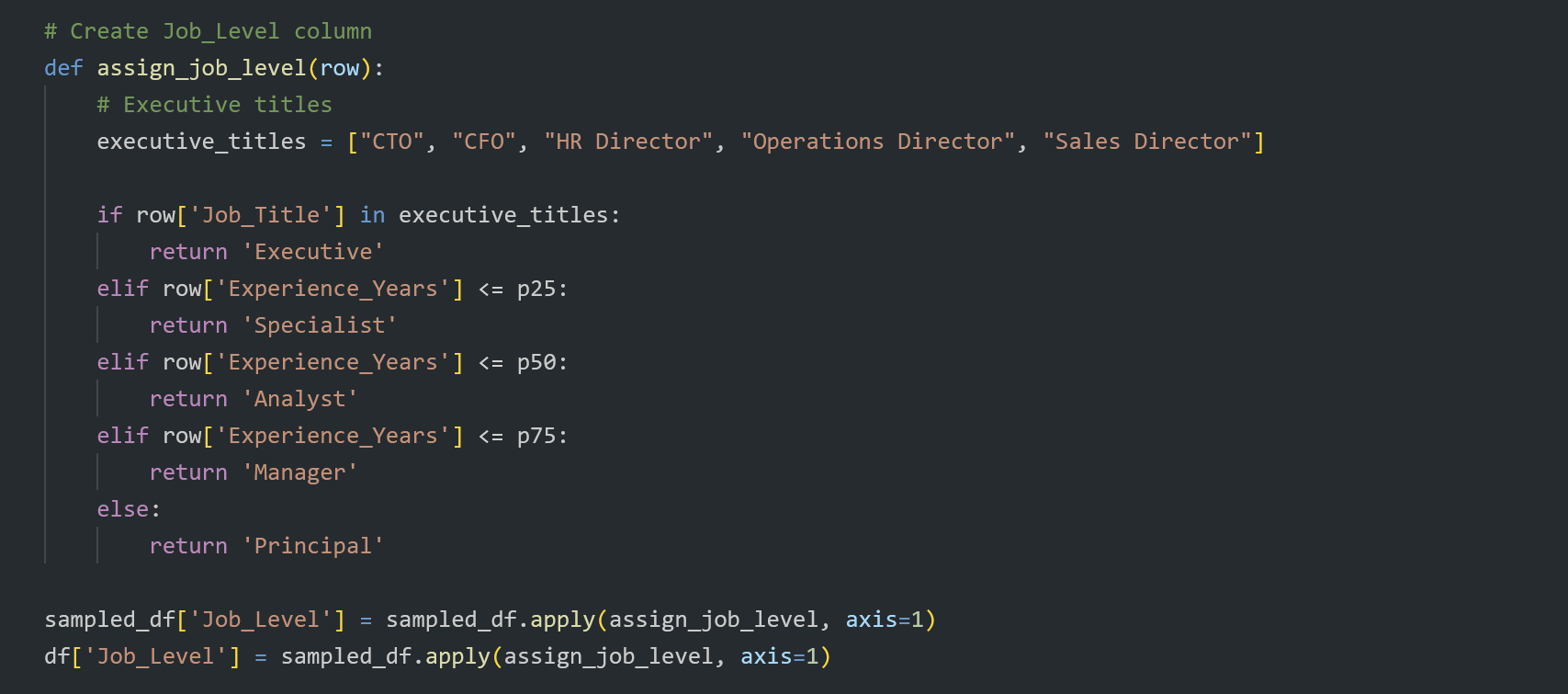


Figure 11.1.2 shows an attribute we created to help us to split employees into role level buckets, to create that we splitted them based on years of experience, these values were added based in the quartile values of the whole data.

Figure 11.1.3 Creation of metrics for historical snapshots of data

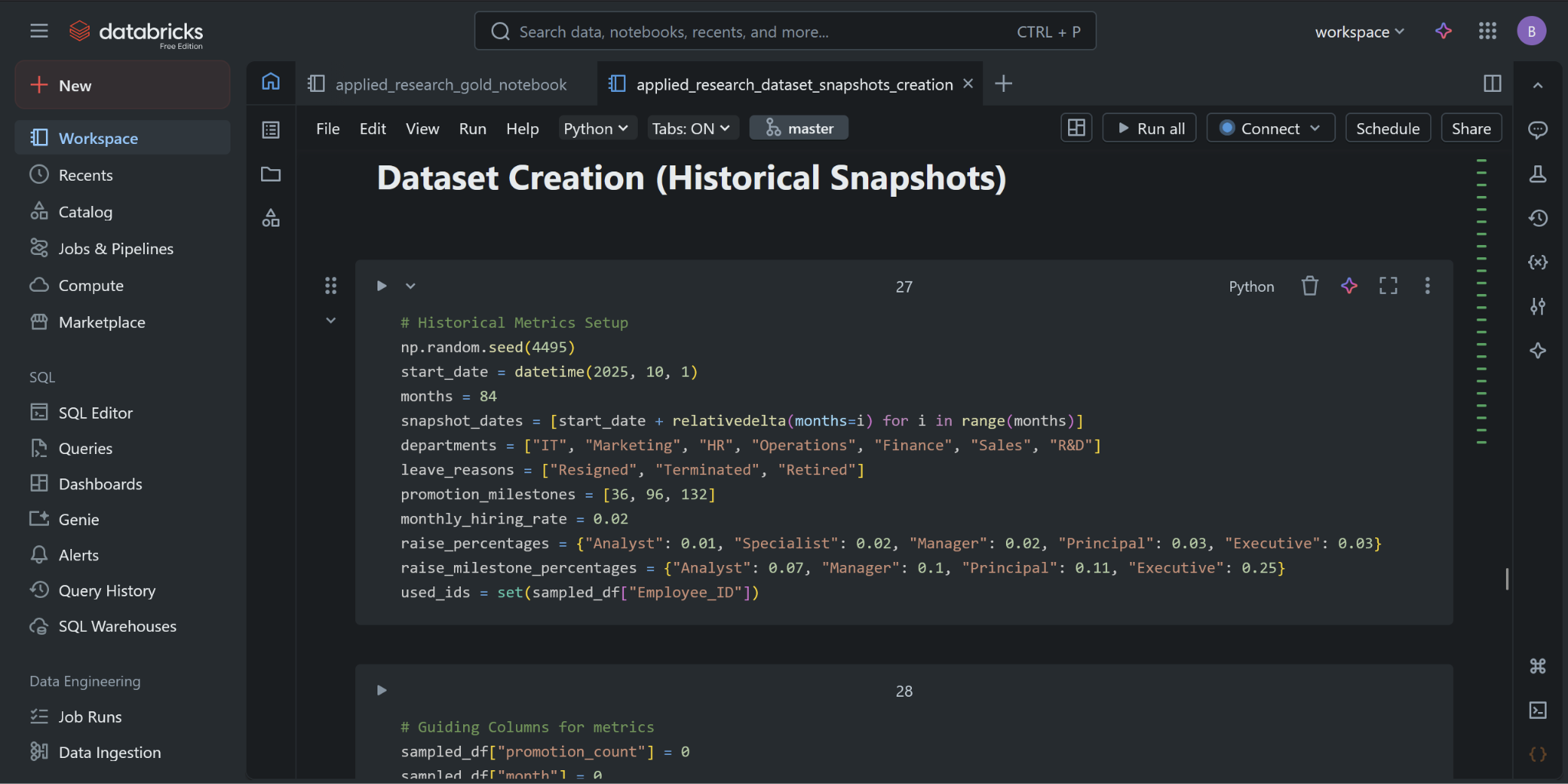


Figure 11.1.3 displays the metrics we used to create our historical scenario, the departments of our company, milestones for promotion, hiring rate, and salary increase percentages.

### 11.2 Screenshots of Data Engineering Pipeline Implementation

Figure 11.2.1 Databricks Workspace

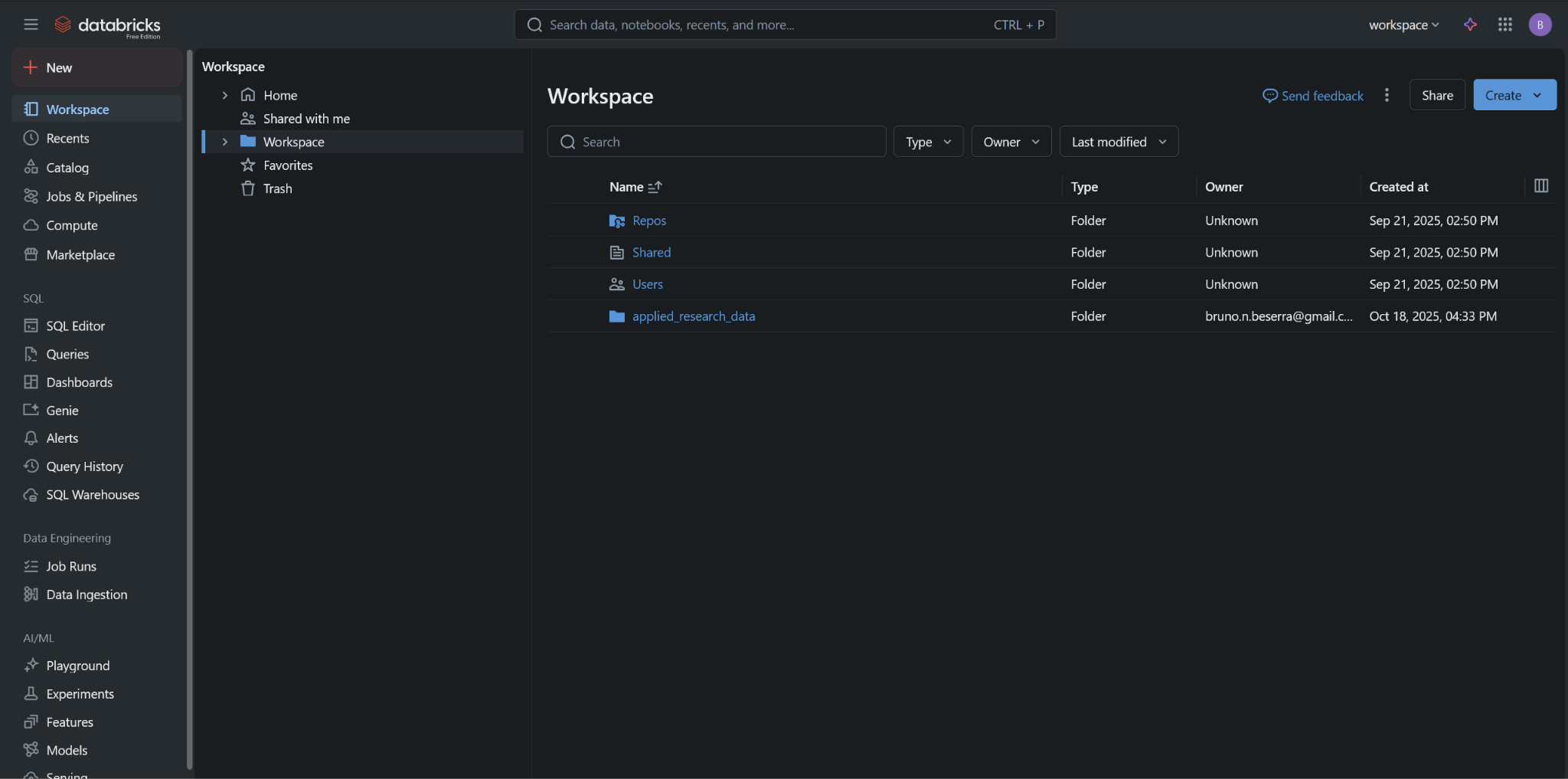


Figure 11.2.1 shows the main page of databricks platform that we used as part of our project, there we could sync it to github, and create folders inside the workspace for our pipeline. In Figure 11.2.2 we demonstrate the folder with the notebooks and Figure 11.2.3 displays the SQL database we are using to store our data inside databricks and export for both PowerBI and the web application.

Figure 11.2.2 Notebooks used for Pipelines designed using Medallion Architecture

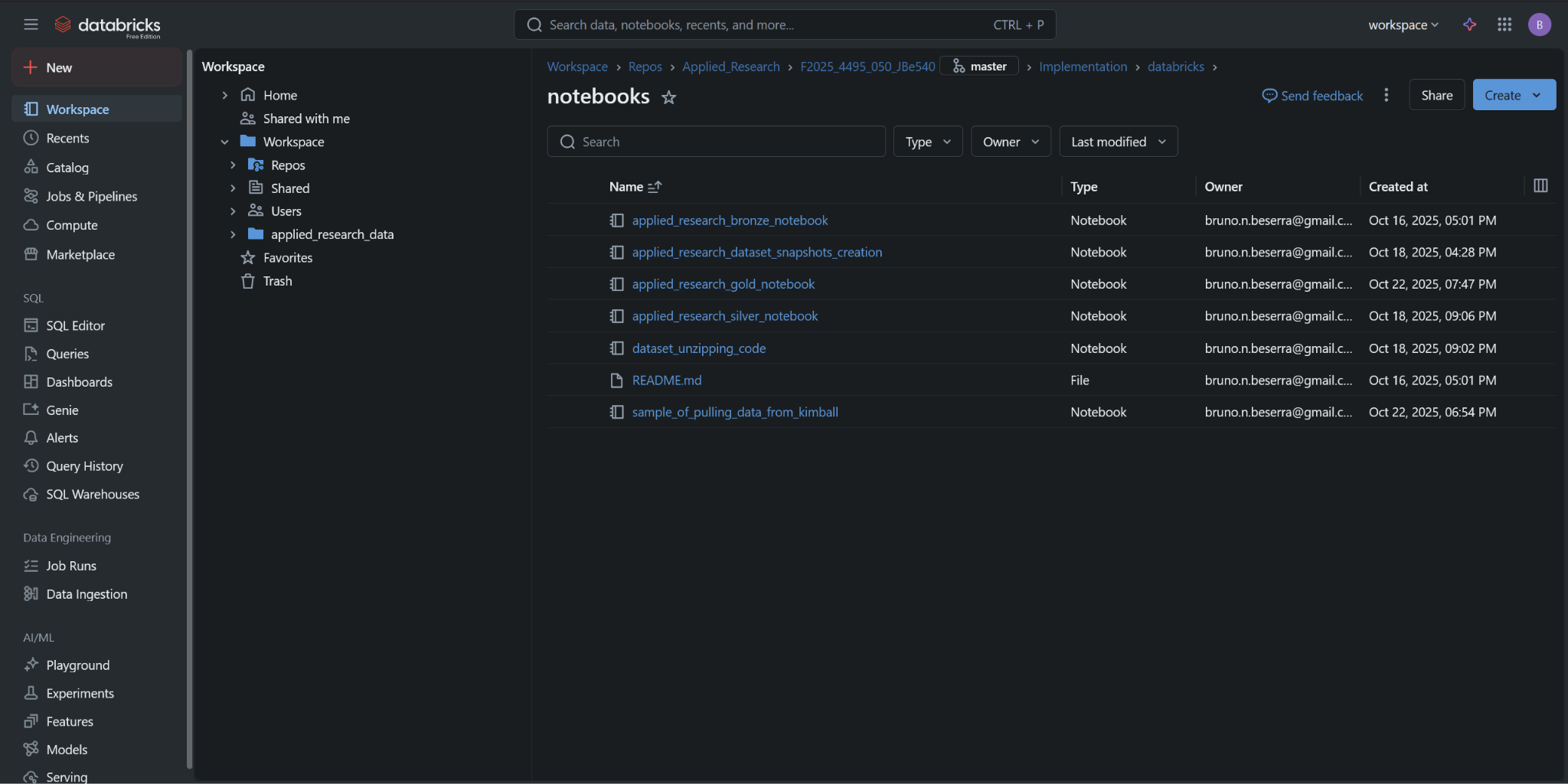
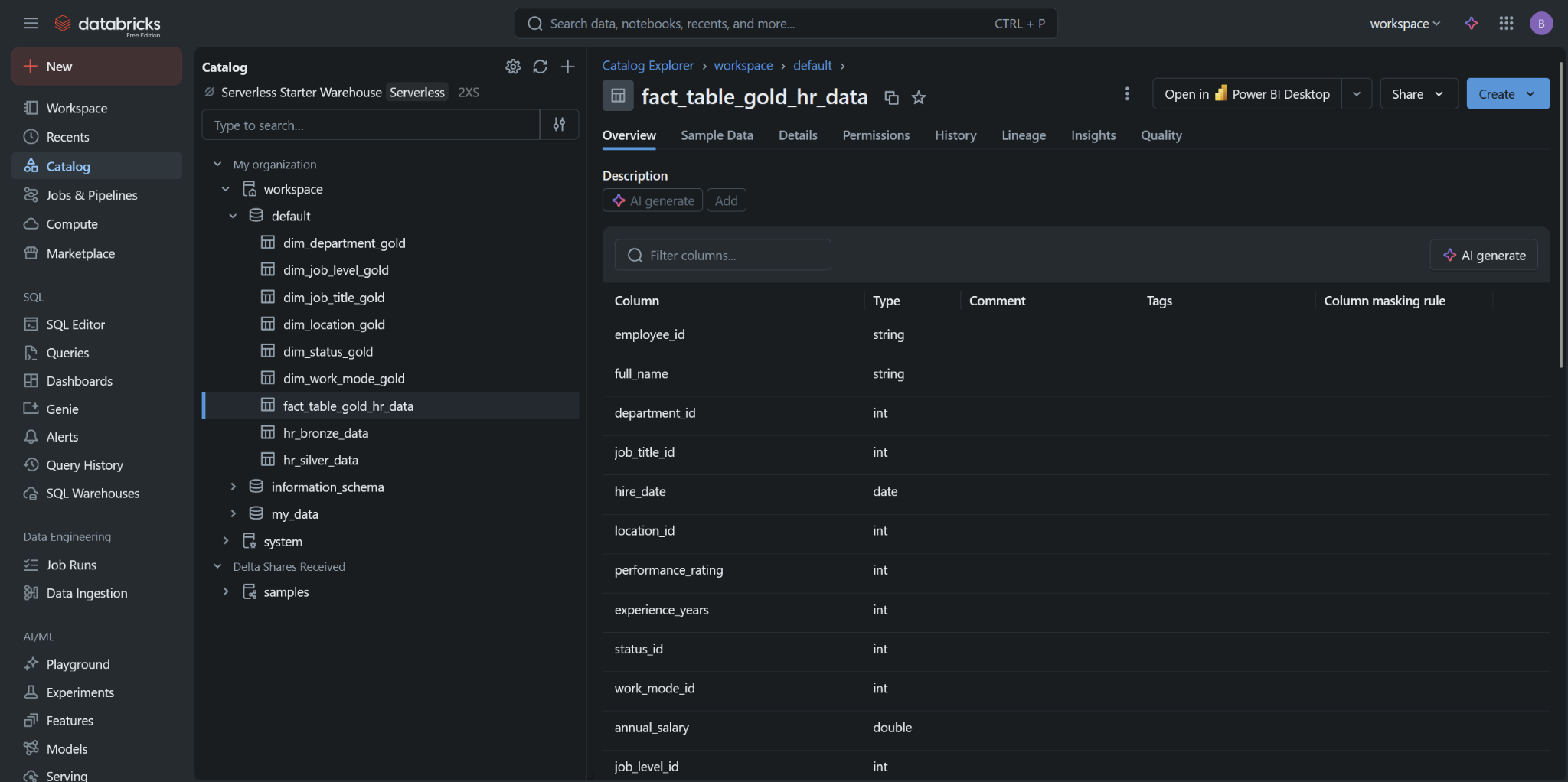


Figure 11.2.3 SQL Server inside Databricks with our data from all layers



### 11.3 Screenshots of Power BI Implementation

Figure 11.3.1 Screenshot of Power BI Dashboard

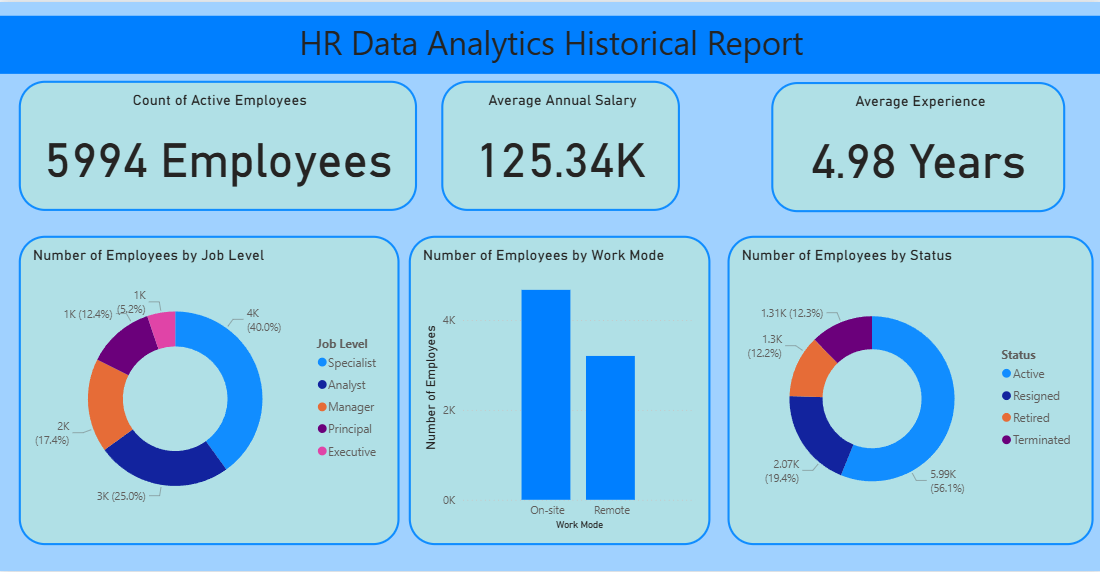


Figure 11.3.1 show the first page of Power BI, the team created a small report showing some interesting and important facts about our data. We focused on the number of employees for this first visualization.

Figure 11.3.2 Screenshot of Power BI Page 1

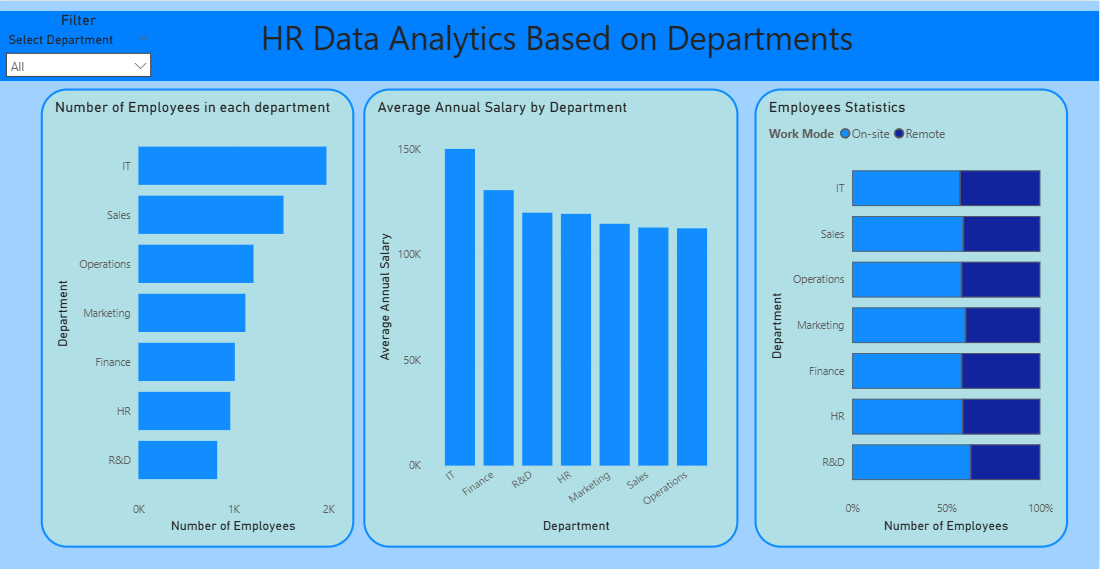


Figure 11.3.2 shows the second page, where the team decided to create several plots focused mainly on departments. In this case, we also added a slicer for departments, allowing you to view data for a specific department if desired.

Figure 11.3.3 Screenshot of Power BI Page 2

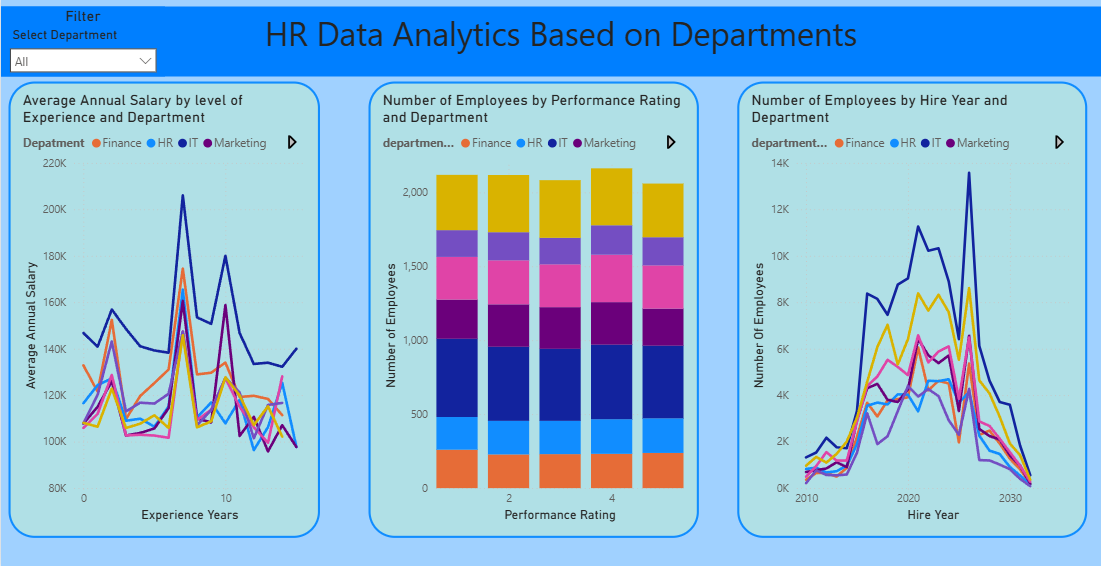


Figure 11.3.3 shows the third page, where the team created several plots based on departments. However, this time we focused on showing department trends rather than just static information. The page also includes a slicer, which is synced with the slicer on the previous page.

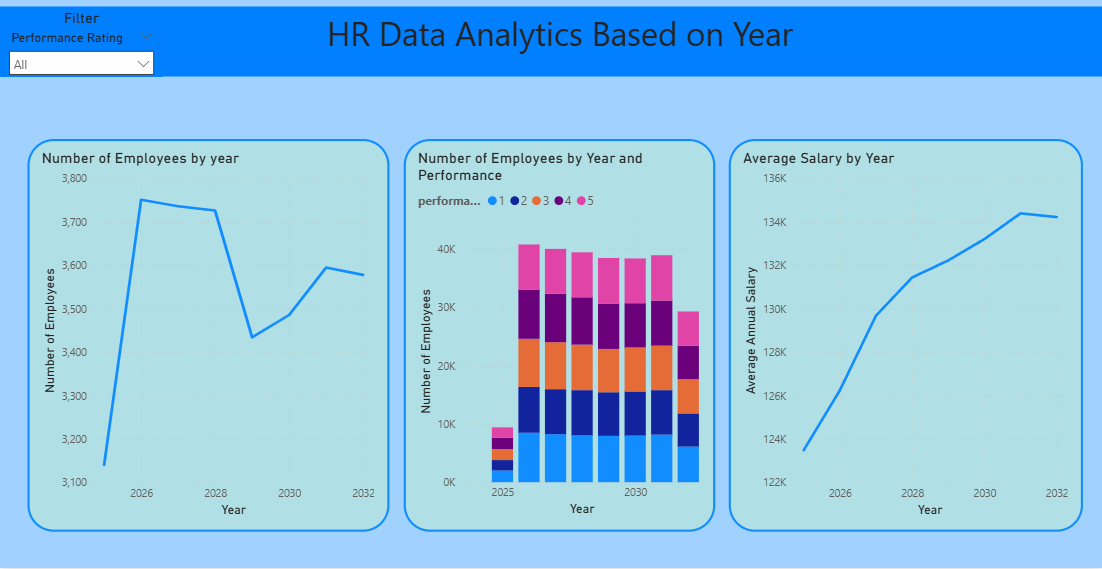
Figure 11.3.4 Screenshot of Power BI Page 4

Figure 11.3.4 is the last visualization; the team designed plots similar to the previous ones, but this time focusing on year-based trends instead of departments. Consequently, the slicer on this page filters by Performance Rating rather than by department.

### 

### 11.4 Screenshots of Web Application Implementation

Figure 11.4.1 Code Snapshot

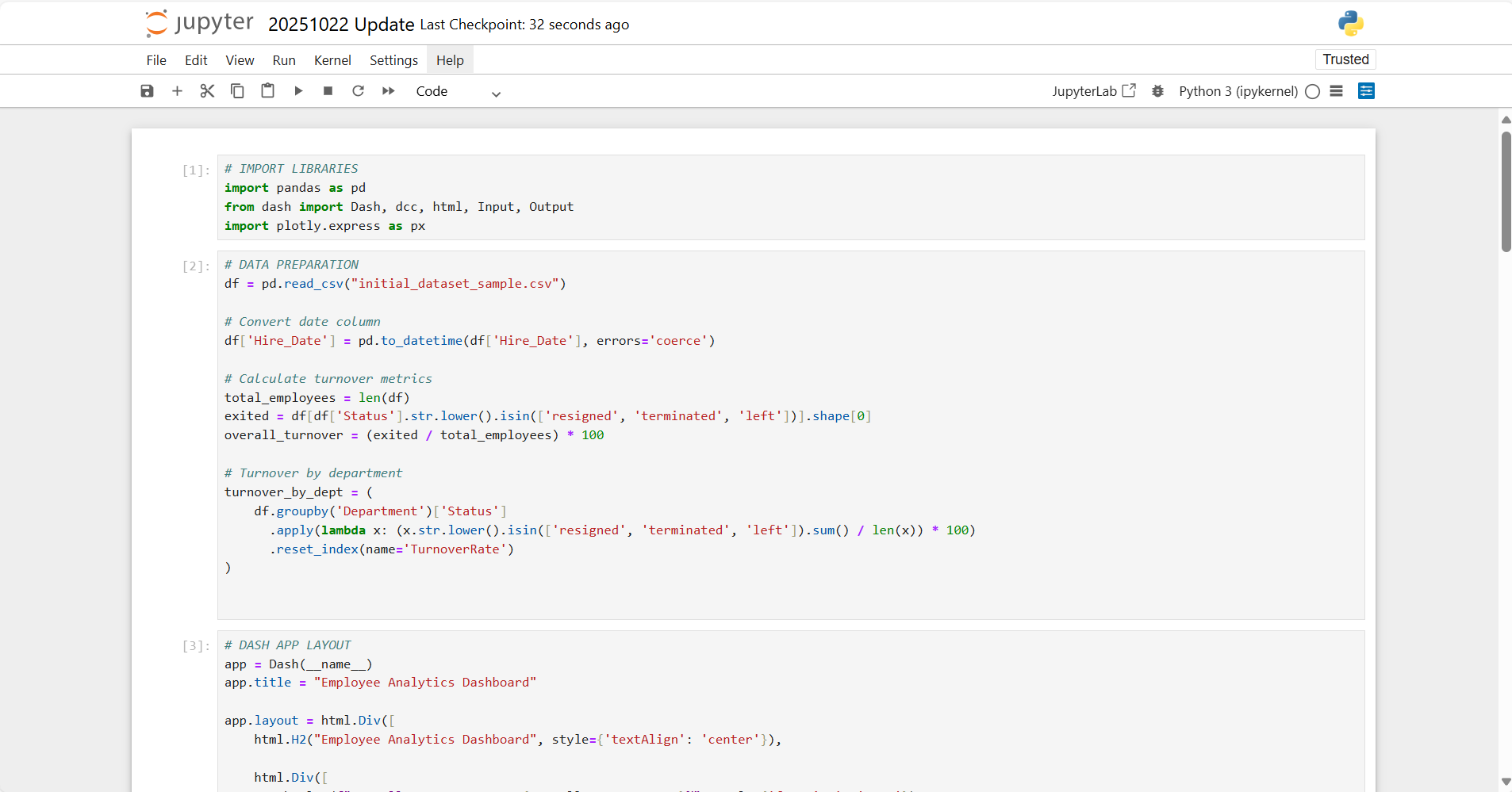


Figure 11.4.1 shows the code that defines three interactive charts in a Dash app, all updating based on the selected department. The first creates a turnover bar chart by job level or department. The second generates a salary chart that switches between a line chart for trends and a bar chart for averages. The third produces a scatter plot showing experience versus performance, with color indicating department or job level and point size representing average salary.

Figure 11.4.2 Snapshot of Initial Web Application: Turnover Rate

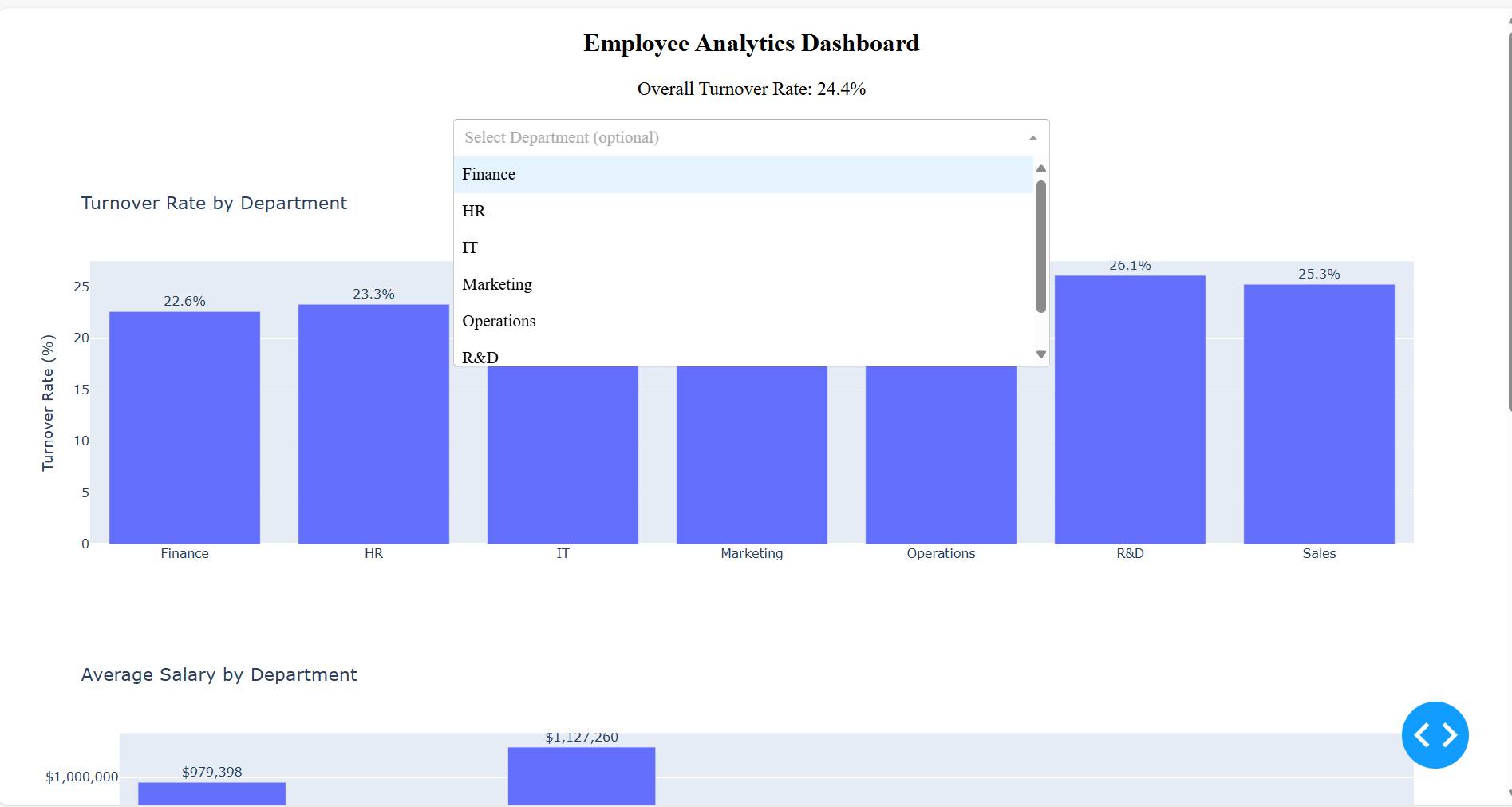


Figure 11.4.3 Average Salary

Figure 11.4.4 Experience vs Performance



The dashboard features three interactive charts that update in response to department selection. Figure 11.4.2 is the turnover chart that shows employee turnover rates either by department or by job level within a selected department. Figure 11.4.3 is the salary chart that displays salary trends over time for a chosen department or average salaries across all departments. Lastly, Figure 11.4.4 shows the experience versus performance chart that uses a scatter plot to show how experience and performance relate, with point size representing average salary. Together, these charts highlight key HR insights on turnover, compensation, and performance patterns.