**Project Title:** HR Data Analytics using Microsoft Fabric

**Project Overview**

**🎯 Objective:**

* Analyze and visualize HR data using Microsoft Fabric.
* Uncover insights on employee attrition, departmental trends, and termination date.
* Implement the **Medallion Architecture**: Staging → Bronze → Silver → Gold.

**🛠️ Tools & Technologies:**

* Microsoft Fabric
* Dataflow
* PySpark (Notebooks)
* Power BI

**📂 Data Source:**

* HR\_DATA.csv containing employee profile information (name, age, gender, department, etc.).

**🔁 Project Workflow:**

* Ingest data into Staging Layer.
* Clean and move to Bronze Layer.
* Transform/enrich in Silver Layer.
* Analyze and aggregate in Gold Layer.
* Build visual dashboards in Power BI.

**Staging & Bronze Layers**

**🪜 Staging Layer:**

* Uploaded CSV to OneLake within Fabric workspace.
* Viewed raw data in Lakehouse "Files" tab.
* No transformations—used for lineage and backup.

**🥉 Bronze Layer:**

* Promoted headers and inferred schema.
* Validated data types and structure.
* Stored as structured Bronze Table.

**✅ Sample Schema:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**Data Cleaning & Transformation (Silver Layer)**

**🧹 Cleaning Steps:**

* Removed nulls from key columns (e.g., ID, Department).
* Standardized text formats (e.g., Title casing).
* Renamed columns for clarity.
* Derived columns:Employee\_id.
* Removed duplicate records.

**📝 PySpark Snippet:**

# Create a new column 'id\_cleaned' by removing dashes from 'id'

from pyspark.sql.functions import \*

cleand\_id = df\_silver.withColumn("int\_id", regexp\_replace(col("id"), "-", ""))

silver\_df = cleand\_id.withColumn("int\_id", col("int\_id").cast("int"))

# silver\_df.select("id", "id\_cleaned").show(10)

display(silver\_df)

# Convert 'birthdate' and 'hire\_date' into proper date format

silver\_df\_birth\_date = silver\_df.withColumn("birthdate", to\_date(col("birthdate"), "M/d/yyyy"))

hr\_data\_silver = silver\_df\_birth\_date.withColumn("hire\_date",to\_date(col("hire\_date"),"M/d/yyyy"))

# display(silver\_df\_hire\_date)

# display(silver\_df\_birth\_date)

hr\_data\_silver.printSchema()

print(f"""

Silver Layer Processing Summary:

Total records before processing: {total\_records\_before}

Null values removed (except termdate): {nulls\_removed}

Duplicate records removed: {duplicates\_removed}

Total records after cleaning: {total\_records\_after}

""")

**Aggregations & Analysis (Gold Layer)**

**📊 KPIs & Aggregations:**

* Total employees
* Average age, salary, tenure
* Termination rate per department
* Gender distribution
* Location-wise attrition analysis

**🧮 Code Example:**

from pyspark.sql.functions import round, col, sum, concat\_ws, lit, collect\_list

from pyspark.sql.window import Window

# Step 1: Count genders per department

gender\_counts = df6.groupBy("department", "gender").count()

# Step 2: Calculate percentage per gender per department

gender\_percentages = gender\_counts.withColumn(

"percentage",

round((col("count") / sum("count").over(Window.partitionBy("department"))) \* 100, 2)

)

# Step 3: Format as "gender:xx%"

gender\_percentages = gender\_percentages.withColumn(

"gender\_percentage",

concat\_ws("", col("gender"), lit(":"), col("percentage").cast("string"), lit("%"))

)

# Step 4: Aggregate all gender\_percentage strings into a single string per department

df\_7 = gender\_percentages.groupBy("department").agg(

concat\_ws(", ", collect\_list("gender\_percentage")).alias("gender\_distribution")

))

**✨ Result:** Gold Layer tables ready for dashboarding.

**Fact-Dimension Model & Business Logic**

**📐 Fact Table:**

DimEmployee: Employee-specific data

DimLocation: Location-specific data

DimDepartment: Department-specific data

FactEmployee: Aggregated metrics and analytics for employees and departments

**📁 Dimension Tables:**

* Employee
* Location
* Department

**📈 Measures Created:**

* Attrition Rate = Terminated / Total Employees
* Average Tenure = Total Tenure / Count(Emp)
* Filter-based metrics: Gender-wise, Location-wise

**Page 7: Dashboard Design in Power BI**

**📌 Key Visuals:**

* KPI Cards: Total Employees, Avg turnover\_rate,
* Column Charts: Total employee by quater
* Line Charts: Total employee by termination
* Filters: Department, Gender

**Column Mapping & Data Dictionary**

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Data Type |
| id | Unique Employee ID | Int |
| first\_name | First Name | String |
| last\_name | Last Name | String |
| birthdate | Date of Birth | Date |
| gender | Gender | String |
| race | Ethnic Group | String |
| department | Business Department | String |
| jobtitle | Employee Designation | String |
| location | Combined Location (Deprecated) | String |
| hire\_date | Date of Joining | Date |
| termdate | Termination Date (if any) | Date |
| location\_city | City of Office | String |
| location\_state | State of Office | String |
| Age | Age (calculated) | Integer |
| AgeGroup | Grouped Age Band | String |

**Quality Checks & Validation**

**✅ Performed:**

* Null value handling
* Data type checks
* Deduplication
* Logical validity

**🛠️ Fixes Applied:**

* Date format issues fixed
* Incomplete gender values standardized
* Outliers in age

**Reusability & Scalability**

**♻️ Benefits with Fabric:**

* Centralized data storage
* Modular pipeline (Staging → Gold)
* Flexible for scaling up new HR datasets

**Deployment & Sharing**

**🚀 Deployed To:**

* Microsoft Fabric Workspace

**🔐 Access Control:**

* View/Edit rights controlled via roles

**🧾 Version Control:**

* Notebooks and reports tagged by version/date

**Learnings & Takeaways**

**🧠 Technical Learnings:**

* Microsoft Fabric
* PySpark & Power BI
* HR dataset transformations

**Page 15: Conclusion & Future Scope**

**🔚 Conclusion:**

* Successfully implemented HR analytics pipeline using Microsoft Fabric.
* Delivered actionable insights on employee

**Prepared By:** Ruchita Urkude   
**Role:** Intern