Ex No: 4 Date: 17/9/25	Building and Automating Pipeline in Databricks for an Healthcare data

Problem Statement: The healthcare industry generates vast amounts of transactional and operational data, ranging from patient consultations to lab tests, pharmacy sales, and diagnostic services. Analyzing this data in near real-time is crucial for improving patient care, optimizing resource allocation, and monitoring revenue trends across service categories and locations.In this project, the goal is to design and implement a data pipeline on Databricks using the healthcare_orders.csv dataset (1000 rows). The pipeline will demonstrate the medallion architecture (Bronze → Silver → Gold), with the following stages:

STEP 1: Create Tasks in databricks

Task 1:

```
from pyspark.sql.functions import col, to_date

# Load the existing table into a dataframe
df_raw = spark.table(
 "workspace.default.silver_ecommerce_orders"
)
# Clean the data
df_cleaned = (
 df_raw.withColumn(
 "order_date",
 to_date(col("order_date"), "yyyy-MM-dd")
 ).withColumn(
 "total_value",
 col("quantity") * col("price")
 )
)
# Create a temp view for SQL
df_cleaned.createOrReplaceTempView(
 "tmp_healthcare_orders_cleaned"
)
# Save as a managed Silver table
spark.sql("""
CREATE OR REPLACE TABLE silver_healthcare_orders AS
SELECT * FROM tmp_healthcare_orders_cleaned
""")
```

Task 2:

```
from pyspark.sql.functions import sum as spark_sum

# Load the cleaned Silver table (from previous step)
df_cleaned = spark.table("silver_healthcare_orders")

# ---- Revenue by product category ----
df_category_sales = (
 df_cleaned.groupBy("product_category") # make sure this column exists
.agg(spark_sum("total_value").alias("total_revenue"))
)
df_category_sales.createOrReplaceTempView("tmp_healthcare_category_sales")

spark.sql("""
CREATE OR REPLACE TABLE gold_healthcare_category_sales AS
SELECT * FROM tmp_healthcare_category_sales
""")
# ---- Revenue by order date ----
df_daily_sales = (
 df_cleaned.groupBy("order_date")
```

```
 .agg(spark_sum("total_value").alias("daily_revenue"))
)
df_daily_sales.createOrReplaceTempView("tmp_healthcare_daily_sales")

spark.sql("""
CREATE OR REPLACE TABLE gold_healthcare_daily_sales AS
SELECT * FROM tmp_healthcare_daily_sales
""")
```

Task 3: html file

Link: https://dbc-0c5a7fba-77bc.cloud.databricks.com/editor/notebooks/2996621244090647?o=3390479664952876

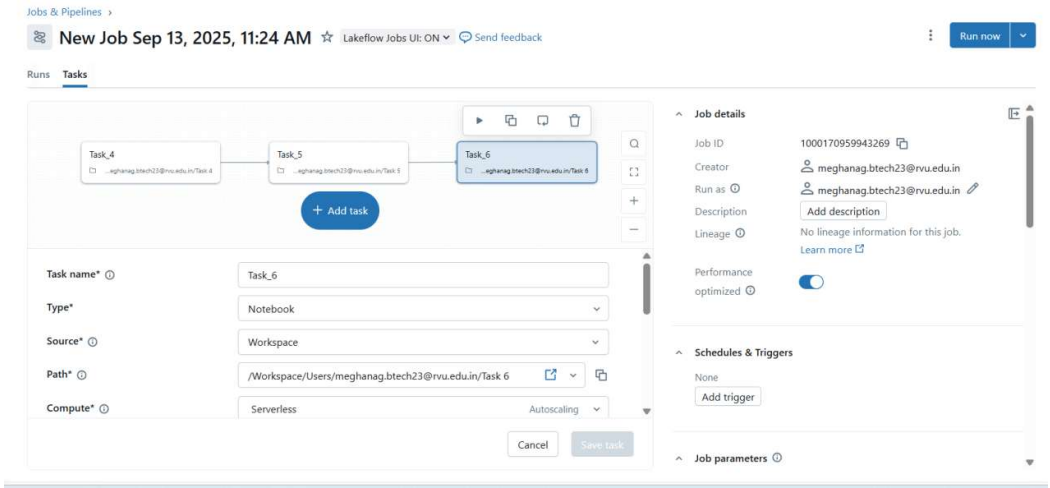
```
# ------------------------------------
# Dashboard: Healthcare Orders Analysis
# ------------------------------------

# Cell 1: Imports
import io
import base64
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Helper to convert matplotlib figure → base64 HTML <img>
def fig_to_base64(fig, fmt="png"):
 buf = io.BytesIO()
 fig.savefig(buf, format=fmt, bbox_inches="tight")
 buf.seek(0)
 img_b64 = base64.b64encode(buf.read()).decode("utf-8")
 buf.close()
 plt.close(fig)
 return f"{img_b64}"

# ------------------------------------
# Cell 2: Load Gold Tables
# ------------------------------------
df_cat = spark.table("gold_healthcare_category_sales").toPandas()
df_daily = spark.table("gold_healthcare_daily_sales").toPandas()

# Ensure proper types
if "order_date" in df_daily.columns:
 df_daily["order_date"] = pd.to_datetime(df_daily["order_date"])

# ------------------------------------
# Cell 3: Create Charts
# ------------------------------------

# Chart 1: Bar chart → Revenue by Product Category
fig1, ax1 = plt.subplots(figsize=(7,4))
ax1.bar(df_cat["product_category"].astype(str), df_cat["total_revenue"],
color="skyblue") ax1.set_title("Revenue by Product Category")
ax1.set_xlabel("Product Category")
ax1.set_ylabel("Total Revenue")
ax1.tick_params(axis='x', rotation=30)
fig1_b64 = fig_to_base64(fig1)
# Chart 2: Line chart → Daily Revenue Trend
fig2, ax2 = plt.subplots(figsize=(8,4))
ax2.plot(df_daily["order_date"], df_daily["daily_revenue"], marker="o", color="green")  ax2.set_title("Daily
Revenue Trend")
ax2.set_xlabel("Order Date")
ax2.set_ylabel("Daily Revenue")
fig2.autofmt_xdate()
fig2_b64 = fig_to_base64(fig2)

# ------------------------------------
```

```
# Cell 4: KPIs
# ----------------------------------
total_revenue = df_cat["total_revenue"].sum() if not df_cat.empty else 0
top_category = (
 df_cat.sort_values("total_revenue", ascending=False).iloc[0]["product_category"]  if not df_cat.empty
else "N/A"
)
last_update = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

# ----------------------------------
# Cell 5: Render Dashboard
# ----------------------------------
html_template = f"""
<div style="font-family:Arial; padding:16px;">
 <h2>Healthcare Orders Dashboard</h2>
 <div style="color:#666; margin-bottom:12px;">Last Updated: {last_update}</div>

 <!-- KPI cards -->
 <div style="display:flex; gap:20px; margin-bottom:20px;">
 <div style="flex:1; background:#f9f9f9; padding:12px; border-radius:8px; box-shadow:0 1px 3px  rgba(0,0,0,0.1);">
 <div style="font-size:12px; color:#777;">Total Revenue</div>
 <div style="font-size:22px; font-weight:700;">₹ {total_revenue:,.2f}</div>  </div>
 <div style="flex:1; background:#f9f9f9; padding:12px; border-radius:8px; box-shadow:0 1px 3px  rgba(0,0,0,0.1);">
 <div style="font-size:12px; color:#777;">Top Category</div>
 <div style="font-size:22px; font-weight:700;">{top_category}</div>
 </div>
 </div>

 <!-- Charts -->
 <div style="display:flex; gap:20px;">
 <div style="flex:1;">
 <h4>Revenue by Category</h4>
 <img src="{fig1_b64}" style="width:100%; border-radius:6px;"/>
 </div>
 <div style="flex:1;">
 <h4>Daily Revenue Trend</h4>
 <img src="{fig2_b64}" style="width:100%; border-radius:6px;"/>
 </div>
 </div>
</div>
"""

displayHTML(html_template)
```

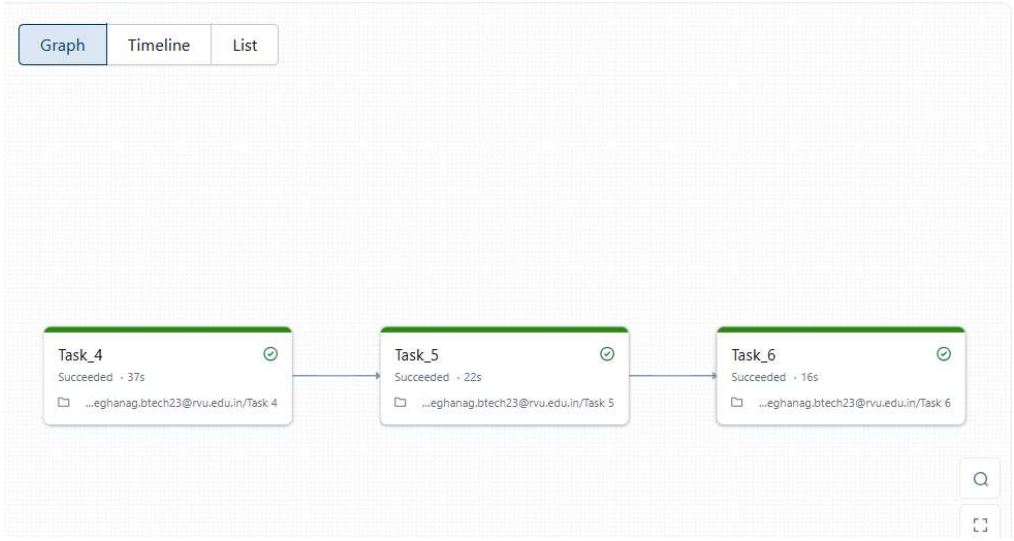
Save all the tasks in databricks notebook and run.

STEP 2: Create job and run the tasks

- Go to Job and pipelines in databricks and click on create job.
- Next type a Task name Task 1 and select the path of Task 1 notebook and save task.
- Next add another two tasks (Task 2 and Task 3).

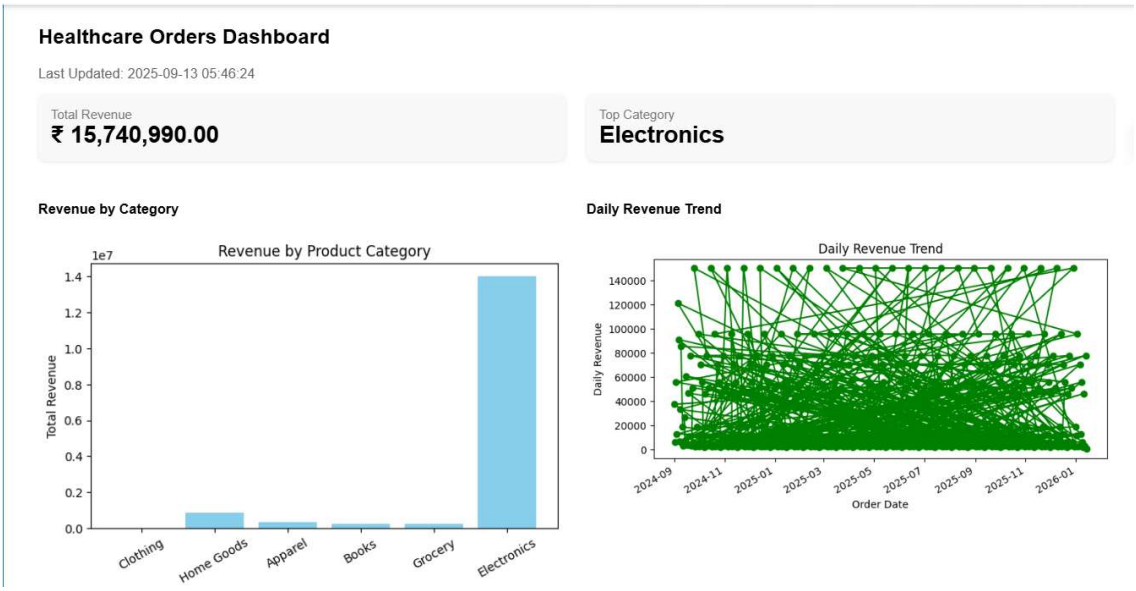


- After saving all task go to job notification , add the email and select start and success button and save
- Now run the job.



- Now all task run and job is succeeded.

STEP 3: Output



Job runs successful

Github link: https://github.com/meghana1653/Data-Engineering/blob/main/LAB_4.zip