Task 1: Customer Data Ingestion

Sample CSV Data

CustomerID	TransactionDate	TransactionAmount	ProductCategory
C001	2024-01-15	250.75	Electronics
C002	2024-01-16	125.50	Groceries
C003	2024-01-17	90.00	Clothing
C004	2024-01-18	300.00	Electronics
C005	2024-01-19	50.00	Groceries

dbutils.fs.cp("file:/Workspace/Shared/customer_transactions.csv","dbfs:/FileStore/customer_transactions.csv")

1. **Load the CSV data into a Delta table in Databricks.**

```
from pyspark.sql import SparkSession

from delta.tables import DeltaTable

import logging

spark = SparkSession.builder.appName("Customer Data Ingestion").getOrCreate()

file_path = " dbfs:/FileStore/customer_transactions.csv "

logging.basicConfig(level=logging.INFO)

try:

    customer_df = spark.read.format("csv").option("header", "true").option("inferSchema",
      "true").load(file_path)

    logging.info(f"File loaded successfully from {file_path}")

    customer_df.write.format("delta").mode("overwrite").save("/mnt/delta/customer_transactions
      _delta")

    logging.info("Data ingested and saved to Delta table.")

except Exception as e:

    logging.error(f"Error loading file: {e}")
```

Task 2: Data Cleaning

Notebook 2: To clean the ingested customer data

```
delta_table = DeltaTable.forPath(spark, "/mnt/delta/customer_transactions_delta")

cleaned_df = delta_table.toDF().dropDuplicates().na.drop(subset=["TransactionAmount"])

cleaned_df.write.format("delta").mode("overwrite").save("/mnt/delta/cleaned_customer_transactio
ns_delta")
```

Task 3: Data Aggregation

```
cleaned_data_df =
    spark.read.format("delta").load("/mnt/delta/cleaned_customer_transactions_delta")

aggregated_df = cleaned_data_df.groupBy("ProductCategory").sum("TransactionAmount")/

    .withColumnRenamed("sum(TransactionAmount)", "TotalTransactionAmount")

aggregated_df.write.format("delta").mode("overwrite")/

            .save("/mnt/delta/aggregated_transactions_delta")
```

Task 4: Pipeline Creation

```
import logging

logging.basicConfig(level=logging.INFO)

def ingest_data():

  try:

    customer_df = spark.read.format("csv").option("header", "true")/

          .option("inferSchema", "true").load(file_path)

    logging.info(f"File loaded successfully from {file_path}")

    customer_df.write.format("delta").mode("overwrite")/

          .save("/mnt/delta/customer_transactions_delta")

  except Exception as e:

    logging.error(f"Error in data ingestion: {e}")


def clean_data():

  try:

    delta_table = DeltaTable.forPath(spark, "/mnt/delta/customer_transactions_delta")

    cleaned_df = delta_table.toDF().dropDuplicates().na.drop(subset=["TransactionAmount"])

    cleaned_df.write.format("delta").mode("overwrite")/

          .save("/mnt/delta/cleaned_customer_transactions_delta")

  except Exception as e:

    logging.error(f"Error in data cleaning: {e}")
```

```
def aggregate_data():
    try:
        cleaned_data_df = spark.read.format("delta")/
                .load("/mnt/delta/cleaned_customer_transactions_delta")

        aggregated_df = cleaned_data_df.groupBy("ProductCategory")/
                .sum("TransactionAmount")/
                .withColumnRenamed("sum(TransactionAmount)", "TotalTransactionAmount")



        aggregated_df.write.format("delta").mode("overwrite")/
                .save("/mnt/delta/aggregated_transactions_delta")
    except Exception as e:
        logging.error(f"Error in data aggregation: {e}")
ingest_data()
clean_data()
aggregate_data()
```

Task 5: Data Validation

```
def validate_data():
    try:
        cleaned_data_df =
        spark.read.format("delta").load("/mnt/delta/cleaned_customer_transactions_delta")
        aggregated_data_df =
        spark.read.format("delta").load("/mnt/delta/aggregated_transactions_delta")


        total_transaction_amount =
        cleaned_data_df.groupBy().sum("TransactionAmount").collect()[0][0]
        total_aggregated_amount =
        aggregated_data_df.groupBy().sum("TotalTransactionAmount").collect()[0][0]
```

```
        if total_transaction_amount == total_aggregated_amount:
            logging.info("Data validation successful: totals match.")
        else:
            logging.warning("Data validation failed: totals do not match.")
    except Exception as e:
        logging.error(f"Error in data validation: {e}")

validate_data()
```