Task 1: Product Inventory Data Ingestion

Sample CSV Data

ProductID	ProductName	StockQuantity	Price	LastRestocked
P001	Laptop	50	1500.00	2024-02-01
P002	Smartphone	200	800.00	2024-02-02
P003	Headphones	300	100.00	2024-01-29
P004	Tablet	150	600.00	2024-01-30
P005	Smartwatch	100	250.00	2024-02-03

dbutils.fs.cp('file:/Workspace/Shared/product_inventory.csv','dbfs:/FileStore/product_inventory.csv')

1. **Load CSV Data**:

```
from pyspark.sql import SparkSession
from pyspark.sql.utils import AnalysisException
import logging
spark = SparkSession.builder \
    .appName("Product Inventory Ingestion") \
    .getOrCreate()

file_path = 'dbfs:/FileStore/product_inventory.csv'
logging.basicConfig(level=logging.INFO)

try:
    product_df = spark.read.csv(file_path, header=True, inferSchema=True)
    product_df.write.format("delta").mode("overwrite").save("/delta/product_inventory")
    logging.info("Data ingested successfully.")

except FileNotFoundError:
    logging.error("File not found: %s", file_path)
except AnalysisException:
    logging.error("Error reading the CSV file, it may be corrupted.")
```

Task 2: Data Cleaning

```
cleaned_product_df = spark.read.format("delta").load("/delta/product_inventory")

cleaned_product_df = cleaned_product_df.filter(
    (cleaned_product_df.StockQuantity.isNotNull()) &
    (cleaned_product_df.Price.isNotNull()) &
    (cleaned_product_df.StockQuantity >= 0)
)

cleaned_product_df.write.format("delta").mode("overwrite")/
                .save("/delta/cleaned_product_inventory")

logging.info("Data cleaned and saved to new Delta table.")
```

Task 3: Inventory Analysis

```
inventory_df = spark.read.format("delta").load("/delta/cleaned_product_inventory")


inventory_df = inventory_df.withColumn("TotalStockValue", inventory_df.StockQuantity *
    inventory_df.Price)

restock_products_df = inventory_df.filter(inventory_df.StockQuantity < 100)


inventory_df.write.format("delta").mode("overwrite").save("/delta/inventory_analysis")

restock_products_df.write.format("delta").mode("overwrite").save("/delta/restock_products")

logging.info("Inventory analysis completed and results saved.")
```

Task 4: Build an Inventory Pipeline

```
from pyspark.sql import SparkSession

from pyspark.sql.utils import AnalysisException

import logging

spark = SparkSession.builder \
    .appName("Product Inventory Pipeline") \
    .getOrCreate()

logging.basicConfig(level=logging.INFO)

file_path = "dbfs:/FileStore/product_inventory.csv"
```

```
def run_pipeline():
    try:
        logging.info("Starting Task 1: Ingesting product inventory data...")

        product_df = spark.read.csv(file_path, header=True, inferSchema=True)

        product_df.write.format("delta").mode("overwrite").save("/delta/product_inventory")
        logging.info("Task 1 completed: Data ingested successfully.")

        logging.info("Starting Task 2: Cleaning data...")
        cleaned_product_df = spark.read.format("delta").load("/delta/product_inventory")
        cleaned_product_df = cleaned_product_df.filter(
            (cleaned_product_df.StockQuantity.isNotNull()) &
            (cleaned_product_df.Price.isNotNull()) &
            (cleaned_product_df.StockQuantity >= 0)
        )
        cleaned_product_df.write.format("delta").mode("overwrite")/
                .save("/delta/cleaned_product_inventory")
        logging.info("Task 2 completed: Data cleaned and saved to new Delta table.")

        logging.info("Starting Task 3: Performing inventory analysis...")
        inventory_df = spark.read.format("delta").load("/delta/cleaned_product_inventory")
        inventory_df = inventory_df.withColumn("TotalStockValue", inventory_df.StockQuantity
       * inventory_df.Price)
        restock_products_df = inventory_df.filter(inventory_df.StockQuantity < 100)

       inventory_df.write.format("delta").mode("overwrite").save("/delta/inventory_analysis")
        restock_products_df.write.format("delta").mode("overwrite")/
                .save("/delta/restock_products")
        logging.info("Task 3 completed: Inventory analysis completed and results saved.")

        logging.info("Pipeline executed successfully.")
```

```
    except FileNotFoundError:
        logging.error("File not found: %s", file_path)
    except AnalysisException as e:
        logging.error("Error reading or writing Delta table: %s", str(e))
    except Exception as e:
        logging.error("Pipeline execution failed: %s", str(e))

run_pipeline()
```

Task 5: Inventory Monitoring

```
restock_df = spark.read.format("delta").load("/delta/restock_products")
if restock_df.count() > 0:
    logging.warning("The following products need restocking:")
    restock_df.show()
else:
    logging.info("All products are sufficiently stocked.")
```