Module 6 Work-Laboratory: Basic Math & Data Filtering

Redesigned Workflow: UCI Online Retail Open Dataset & Interactive Jupyter Cell Architecture

Participant Name: _____ **Date:** _____

Section 1: The Jupyter Cell Memory Paradigm

Unlike flat, script-based environments, a **Jupyter Notebook** retains its programmatic state and data frames variables in active memory across isolated execution blocks. This architecture allows an analyst to ingest a massive open dataset *once* in a single cell, and systematically run slicing, filtering, and descriptive mathematical computations in separate cells below without incurring the performance costs of re-reading or parsing raw files from scratch.

Section 2: Interactive Ingestion & Exploration

In your notebook, create your first execution block to pull the official online dataset stream into memory:

```
In [1]:
```

```
import pandas as pd

# Stream live open UCI Online Retail dataset subset from a public endpoint
url = "https://raw.githubusercontent.com/datagy/data/main/online_retail_subset.csv"
df = pd.read_csv(url)

# Audit structural shapes and column parameters
print(f"Dataset Dimensions Loaded: {df.shape}")
print(df.dtypes)
```

Section 3: Isolating Clusters & Filtering Outliers

To avoid polluting analytical aggregations, vector data filtering maps boolean criteria over individual arrays to eliminate returns, cancellations (negative values), or empty records:

```
In [2]:
```

```
# Remove transactions containing cancellations (Quantity <= 0) or unpriced events
df_clean = df[(df['Quantity'] > 0) & (df['UnitPrice'] > 0)].copy()

# Add a calculated descriptive metric vector: Line Total Revenue
df_clean['LineTotal'] = df_clean['Quantity'] * df_clean['UnitPrice']

print(f"Filtered clean dataframe row count: {len(df_clean)}")
```

Section 4: Group Descriptive Mathematics

Using Jupyter's cached `df_clean` frame, execute categorical data segmentation and calculate group math metrics:

In [3]:

```
# Segment records by Country and isolate transaction line metrics
country_analysis = df_clean.groupby('Country')['LineTotal'].agg(['sum', 'mean', 'count'])

# Sort and output the top 5 highest-volume international clusters
print(country_analysis.sort_values(by='sum', ascending=False).head(5))
```

In [4]:

```
# Filter specifically down to United Kingdom transactions for isolated trend evaluation
df_uk = df_clean[df_clean['Country'] == 'United Kingdom']
print(f"Mean order value in United Kingdom: ${df_uk['LineTotal'].mean():.2f}")
```

Section 5: Applied Technical Verification

Execute the cells above sequentially inside your active Jupyter session, evaluate the returned memory states, and complete the validation log entries below.

Question 1: Cell State Execution Continuity

If you re-run cell `In [3]` multiple times without restarting your notebook environment, does it re-download the global file from the web address or access the cached dataframe wrapper in active system memory? Explain why this design pattern is preferred.

Question 2: Scripting Conditional Complex Slices

Write the exact Python syntax to extract a new dataframe called `df_high_value` containing transactions from `df_clean` where the `Quantity` is greater than 50 **AND** the `UnitPrice` is greater than 10.00.

Question 3: Open Data Metric Interpretation

Look closely at the execution output from cell `In [4]`. Write down the calculated mean transaction line value for the United Kingdom. How does this mathematical vector result compare to other international groups?
