“Exercise_3_4_HIS_EDA_Notebook.pdf” — covering **Exploratory Data Analysis (EDA)**

🧠 SUMMARY — Exploratory Data Analysis (EDA) for High-Integrity Systems

1. Overview

Exploratory Data Analysis (EDA) helps uncover patterns, anomalies, and relationships in system performance metrics.
It combines **statistics**, **visualization**, and **domain reasoning**.

---

2. Data Loading and Inspection

```
import pandas as pd
df = pd.read_csv("generated_his_system_metrics.csv", index_col=0)
df.info()
```

- Understand data structure: number of rows, columns, and data types.
- Identify **categorical** and **numerical** columns.

---

3. Classifying Data

```
categorical_cols = df.select_dtypes(include=['object', ‘category’]).columns.tolist()
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
```

- **Categorical (Nominal/Ordinal):** ‘mode’ (system state)
- **Numerical (Continuous/Discrete):** latency, CPU%, memory, errors, uptime, etc.

---

4. Histograms vs Bar Plots

- **Histogram** → Continuous/numerical data (shows frequency distribution)
- **Bar Plot** → Categorical data (shows counts per category)

```
df[numerical_cols].hist(figsize=(12,8))
```

---

5. Box Plots

- Useful for comparing **distributions** and spotting **outliers**.
- Can be **misleading** if:
  - Scale differences across variables
  - Combining units with different magnitudes

```
sns.boxplot(data=df[numerical_cols])
```

✅ Tip: Plot variables individually for clarity.

---

6. Descriptive Statistics

```
df.describe()
```

Measure	Meaning
mean	average
std	variability
25%, 50%, 75%	quartiles (Q1, Q2, Q3)
min/max	range of data

Use median for skewed data (less sensitive to outliers).
Use mean for symmetric distributions.

---

7. Measures of Variability

- **Range = max - min**
- **Standard Deviation (σ)** measures spread around the mean.
- **IQR (Interquartile Range)** measures middle 50% variability.

```
IQR = df['sensor_latency_ms'].quantile(0.75) - df['sensor_latency_ms'].quantile(0.25)
```

---

8. Skewness & Kurtosis

```
from scipy.stats import skew, kurtosis
skew(df['uptime_hours']), kurtosis(df['uptime_hours'])
```

- **Right-skewed (mean > median > mode)** → long tail on right.
- **Left-skewed (mean < median < mode)** → long tail on left.
- **Kurtosis** shows tail heaviness (outliers).

---

9. Quantiles and Outliers

```
q = df['uptime_hours'].quantile([0.01, 0.25, 0.5, 0.75, 0.99])
IQR = q[0.75] - q[0.25]
lower = q[0.25] - 1.5 * IQR
upper = q[0.75] + 1.5 * IQR
outliers = df[(df['uptime_hours'] < lower) | (df['uptime_hours'] > upper)]
```

- **Outliers** are data points outside the range `[Q1 - 1.5×IQR, Q3 + 1.5×IQR]`.
- High outlier counts may indicate **skewed** or **long-tailed** distributions.

---

10. Comparing Variables

```
df[['CPU_thermostat_stability', 'sensor_thermostat_stability']].agg(['mean', 'std'])
```

- Smaller **std → more stable system**.
- But be cautious — different units/scales can distort comparisons.

---

11. Scatter Plots for Relationships

```
sns.scatterplot(x='sensor_latency_ms', y='cpu_usage_percent', hue='mode', data=df)
```

- Detect **linear relationships** or **clusters**.
- Certain modes may be associated with higher CPU loads.

12. Correlation Matrix

```
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

- Shows strength & direction of relationships.
  - `+1` → strong positive
  - `-1` → strong negative
  - `0` → no correlation

```
corr.unstack().sort_values(ascending=False)
```

→ Identifies strongest correlations between metrics.

📘 CHEAT SHEET — Exam Quick Reference

Concept	Command/Formula	Interpretation
Load Data	`pd.read_csv()`	Import CSV file
Data Info	`df.info()`	Check column types
Categorical Columns	`select_dtypes(['object', 'category']) columns.tolist()`	Nominal/Ordinal variables
Numerical Columns	`select_dtypes(['float64','int64']) .columns.tolist()`	Continuous/Discrete
Histogram	`df[col].hist()`	Numeric data distribution
Box Plot	`sns.boxplot()`	Detect outliers & skew
Descriptive Stats	`df.describe()`	Summary metrics
IQR	`Q3 - Q1`	Spread of middle 50%
Outlier Rule	`[Q1−1.5×IQR, Q3+1.5×IQR]`	Extreme values
Range	`df[col].max() - df[col].min()`	Spread of data
Mean/Median Comparison	`mean > median > mode`	Right-skewed
Correlation	`df.corr()`	Relationships between vars
Scatter Plot	`sns.scatterplot(x, y)`	Visual relationship
Variability	`std()`	Consistency of metric
Percentiles	`df.quantile([.25,.5,.75])`	Quartiles
Skewness	`skew(df[col])`	Direction of tail
Kurtosis	`kurtosis(df[col])`	Tail heaviness
Compare Means	`df.agg(['mean','std'])`	Check consistency
Heatmap	`sns.heatmap(df.corr())`	Visual correlation map

⚙️ Common Exam Traps

- Don't compare raw variability of metrics with different units.
- Always verify **equal bin widths** in histograms.
- Ensure categorical plots (bar plots) don't start the Y-axis above zero.
- Confirm sample size adequacy before inferring trends.

Exercise 3–4: HIS EDA Notebook covers **Advanced Exploratory Data Analysis (EDA)** using **Pandas, Seaborn, and Matplotlib**, applied to a *High-Integrity Systems (HIS)* dataset.

🧠 Section 1: Exploring Distributions with Histograms

Concept:

- **Histogram** → Used for **numerical** data (continuous or discrete).
  It shows how values are distributed (frequency per range or "bin").
- **Bar Plot** → Used for **categorical** data (nominal or ordinal).

Example:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Example data
df = pd.DataFrame({
    'sensor_latency_ms': [12, 15, 20, 18, 25, 30, 22, 21],
    'mode': ['Normal', 'Normal', 'Emergency', 'Normal', 'Failure', 'Normal', 'Emergency',
'Failure']
})

# Histogram (numerical)
plt.hist(df['sensor_latency_ms'], bins=5, color='skyblue', edgecolor='black')
plt.title("Sensor Latency Distribution")
plt.xlabel("Latency (ms)")
plt.ylabel("Frequency")
plt.show()

# Bar plot (categorical)
sns.countplot(x='mode', data=df)
plt.title("System Mode Frequency")
plt.show()
```

⚠️ Misleading Bar Chart Example

If the maintenance engineer uses:

```
plt.ylim(40, None)
```

This **cuts the y-axis**, making differences seem **larger than they are** — misleading!

✅ Fix:

Remove `plt.ylim()` or start it from `0`:

```
sns.countplot(x='mode', data=df)
plt.ylim(0, None)
plt.title("System Mode Frequency - Corrected")
plt.show()
```

📦 Section 2: Box Plots

Concept:

A **box plot** summarizes:

- **Median (Q2)**
- **Quartiles (Q1, Q3)**
- **IQR (Q3−Q1)**
- **Outliers** (points outside 1.5×IQR range)

Use it to compare **distributions** or **spot anomalies**.

Example:

```
sns.boxplot(data=df[['sensor_latency_ms']])
plt.title("Boxplot of Sensor Latency")
plt.show()
```

🧩 **Misleading Boxplots**:

- When variables have **different scales**, putting all on one boxplot hides patterns.
  ✅ Fix: Normalize or plot separately:

```
df[['sensor_latency_ms', 'cpu_usage_percent']].boxplot()
```

📊 Section 3: Measures of Center & Variability

Measure	Description	Example Function
Mean	Average value	`df.mean()`
Median	Middle value	`df.median()`
Mode	Most frequent value	`df.mode()`
Std (σ)	Spread around mean	`df.std()`
Range	max - min	`df.max() - df.min()`

Example:

```
df.describe()
```

🧮 **IQR Example**:

```
Q1 = df['sensor_latency_ms'].quantile(0.25)
Q3 = df['sensor_latency_ms'].quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

📈 Section 4: Quantiles, Percentiles & Outliers

Concept:

Outliers are detected using **IQR rule**:

Lower bound} = Q1 - 1.5 × IQR
Upper bound} = Q3 + 1.5 × IQR

Example:

```
Q1 = df['sensor_latency_ms'].quantile(0.25)
Q3 = df['sensor_latency_ms'].quantile(0.75)
IQR = Q3 - Q1

lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR
outliers = df[(df['sensor_latency_ms'] < lower) | (df['sensor_latency_ms'] > upper)]

print(f"Outliers detected: {len(outliers)}")
```

🔥 Section 5: Comparing Variables

Example: Thermostat Stability

```
mean_cpu = df['CPU_thermostat_stability'].mean()
std_cpu = df['CPU_thermostat_stability'].std()

mean_sensor = df['sensor_thermostat_stability'].mean()
std_sensor = df['sensor_thermostat_stability'].std()

print("CPU:", mean_cpu, std_cpu)
print("Sensor:", mean_sensor, std_sensor)
```

⚠️ **Caution**:
Comparing only mean/std can be misleading — one variable may have outliers or be skewed.

🌐 Section 6: Scatter Plot Analysis

Concept:

Use **scatter plots** to visualize **relationships** (correlations or clusters).

Example:

```
sns.scatterplot(x='sensor_latency_ms', y='cpu_usage_percent', hue='mode', data=df)
plt.title("Latency vs CPU Usage by Mode")
plt.show()
```

- A linear trend → correlation
- Clusters → system modes or anomalies

📉 Section 7: Correlation Matrix

Concept:

Shows how strongly variables are related.

$$\rho = 1 \text{ (strong positive)}, \ \rho = -1 \text{ (strong negative)}$$

Example:

```
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix of System Metrics")
plt.show()

print("Strongest positive:",
corr.unstack().sort_values(ascending=False).drop_duplicates().head())
print("Strongest negative:", corr.unstack().sort_values().head())
```

🧾 Summary — Key Takeaways

Concept	Tool	Purpose
Histogram	`plt.hist`, `df.hist()`	Distribution (numerical)
Bar Plot	`sns.countplot`	Categorical frequency
Box Plot	`sns.boxplot`	Outliers & variability
IQR Rule	Quantiles	Outlier detection
Scatter Plot	`sns.scatterplot`	Correlation / clusters
Correlation Matrix	`sns.heatmap`	Strength of relationships

🧠 Exploratory Data Analysis (EDA) with Pandas, Matplotlib & Seaborn

Exercise 3–4: HIS Dataset Analysis

📦 1. Import Libraries and Load Data

```
# Importing core libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Display settings
pd.set_option('display.max_columns', None)
sns.set_style("whitegrid")

# Example dataset (you can replace with your HIS dataset)
data = {
    'mode': ['Normal', 'Emergency', 'Failure', 'Normal', 'Failure', 'Normal', 'Emergency',
'Normal'],
    'sensor_latency_ms': [15, 28, 35, 18, 40, 20, 30, 17],
    'cpu_usage_percent': [60, 85, 95, 70, 90, 68, 88, 72],
    'thermostat_stability': [0.9, 0.7, 0.4, 0.95, 0.45, 0.93, 0.65, 0.92]
}

df = pd.DataFrame(data)
df
```

📊 2. Understanding Distributions

Histogram (numerical data)

```
plt.figure(figsize=(7, 4))
plt.hist(df['sensor_latency_ms'], bins=5, color='skyblue', edgecolor='black')
plt.title("Sensor Latency Distribution")
plt.xlabel("Latency (ms)")
plt.ylabel("Frequency")
plt.show()
```

Bar Plot (categorical data)

```
plt.figure(figsize=(6, 4))
sns.countplot(x='mode', data=df, palette='pastel')
plt.title("System Mode Frequency")
plt.show()
```

⚠️ Warning about misleading charts

```
# Misleading version (y-axis cut off)
sns.countplot(x='mode', data=df)
plt.ylim(40, None)
plt.title("⚠️ Misleading Chart: Y-axis Cut Off!")
plt.show()

# Corrected version
sns.countplot(x='mode', data=df)
plt.ylim(0, None)
```

```
plt.title("✅ Correct Chart: Full Y-axis Shown")
plt.show()
```

📦 3. Boxplots — Detecting Outliers

```
plt.figure(figsize=(6, 4))
sns.boxplot(y='sensor_latency_ms', data=df)
plt.title("Boxplot: Sensor Latency")
plt.show()
```

Side-by-side boxplots:

```
plt.figure(figsize=(7, 5))
sns.boxplot(x='mode', y='cpu_usage_percent', data=df)
plt.title("CPU Usage by System Mode")
plt.show()
```

📈 4. Descriptive Statistics

```
df.describe()
```

IQR-based outlier detection:

```
Q1 = df['sensor_latency_ms'].quantile(0.25)
Q3 = df['sensor_latency_ms'].quantile(0.75)
IQR = Q3 - Q1

lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

outliers = df[(df['sensor_latency_ms'] < lower) | (df['sensor_latency_ms'] > upper)]
print("Outliers detected:\n", outliers)
```

🧮 5. Measures of Center & Spread

```
print("Mean latency:", df['sensor_latency_ms'].mean())
print("Median latency:", df['sensor_latency_ms'].median())
print("Standard deviation:", df['sensor_latency_ms'].std())
print("Range:", df['sensor_latency_ms'].max() - df['sensor_latency_ms'].min())
```

🔥 6. Comparing Variables

```
mean_cpu = df['cpu_usage_percent'].mean()
std_cpu = df['cpu_usage_percent'].std()

mean_stab = df['thermostat_stability'].mean()
std_stab = df['thermostat_stability'].std()

print(f"CPU Usage: Mean={mean_cpu:.2f}, Std={std_cpu:.2f}")
print(f"Thermostat Stability: Mean={mean_stab:.2f}, Std={std_stab:.2f}")
```

🌐 7. Relationship Analysis — Scatterplots

```
plt.figure(figsize=(7, 5))
sns.scatterplot(x='sensor_latency_ms', y='cpu_usage_percent', hue='mode', data=df, s=100)
plt.title("Latency vs CPU Usage by Mode")
plt.show()
```

🧩 8. Correlation Matrix

```
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()

print("Strongest correlations:")
print(corr.unstack().sort_values(ascending=False).drop_duplicates().head())
```

✅ 9. Key Takeaways

Concept	Function	Purpose
`plt.hist()`	Histogram	Distribution of numeric data
`sns.countplot()`	Bar plot	Frequency of categories
`sns.boxplot()`	Box plot	Outliers & variability
`.describe()`	Descriptive stats	Quick numeric summary
`.corr()` + `sns.heatmap()`	Correlation	Relationships between variables
`sns.scatterplot()`	Scatter plot	Visualize trends & clusters

🧠 Practice Challenge:

Try answering these:

1. Which system mode shows the highest CPU usage on average?
2. Is there any correlation between latency and stability?
3. Are there outliers in thermostat stability?
4. Create a violin plot for `cpu_usage_percent` by `mode`.
