Cheat Sheet for Data Analysis #3Exploratory Data Analysis & Visualization

Basic Setup and Data Loading

Import essential libraries

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

Meaning: Imports necessary libraries for data analysis and visualization

Load a CSV file into a DataFrame

```
df = pd.read_csv("Diamond.csv")
```

Meaning: Reads data from a CSV file and stores it in a variable called `df`

View basic dataset information

```
print("Dataset shape:", df.shape)
df.head()
```

Meaning: Displays the dimensions of the dataset and the first few rows for quick inspection

---

Univariate Analysis (Single Variable)

Histogram for numerical variables

```
plt.figure(figsize=(8, 5))
sns.histplot(df['carat'], bins=20, kde=False)
plt.title('Histogram of Carat')
plt.xlabel('Carat')
plt.ylabel('Frequency')
plt.show()
```

Meaning: Shows the distribution of a numerical variable; reveals shape, central tendency, spread, and potential outliers

Density plot for numerical variables

```
plt.figure(figsize=(8, 5))
sns.kdeplot(df['price'], fill=True)
plt.title('Density Plot of Price')
```

```
plt.xlabel('Price')
plt.show()
```

Meaning: Smoothed representation of the distribution; better for comparing distributions across groups

Box/Whisker plot for numerical variables

```
plt.figure(figsize=(8, 5))
sns.boxplot(y=df['price'])
plt.title('Box Plot of Price')
plt.ylabel('Price')
plt.show()
```

Meaning: Visualizes five-number summary (min, Q1, median, Q3, max); excellent for identifying outliers and comparing distributions

Bar chart for categorical variables

```
plt.figure(figsize=(10, 6))
df['colour'].value_counts().plot(kind='bar')
plt.title('Bar Chart of Colour')
plt.xlabel('Colour')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```

Meaning: Shows frequency distribution of categorical variables; reveals most/least common categories

---

Bivariate Analysis (Two Variables)

Scatter plot (Numerical vs. Numerical)

```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='carat', y='price')
plt.title('Scatter Plot: Carat vs Price')
plt.xlabel('Carat')
plt.ylabel('Price')
plt.show()
```

Meaning: Visualizes relationship between two numerical variables; reveals patterns, trends, correlations, and outliers

Grouped box plots (Categorical vs. Numerical)

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='colour', y='price')
```

```
plt.title('Price Distribution by Colour')
plt.xlabel('Colour')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.show()
```

Meaning: Compares distribution of a numerical variable across categories; reveals differences in central tendency and spread

Violin plots (Categorical vs. Numerical)

```
plt.figure(figsize=(10, 6))
sns.violinplot(data=df, x='clarity', y='price')
plt.title('Price Distribution by Clarity (Violin Plot)')
plt.xlabel('Clarity')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.show()
```

Meaning: Combines box plot with kernel density estimate; shows distribution shape within each category

Stacked bar chart (Categorical vs. Categorical)

```
contingency_table = pd.crosstab(df['certification'], df['clarity'])
contingency_table.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Stacked Bar Chart: Certification vs Clarity')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.legend(title='Clarity', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Meaning: Shows joint frequency distribution of two categorical variables; reveals associations between categories

Heatmap of contingency table

```
plt.figure(figsize=(8, 6))
sns.heatmap(pd.crosstab(df['colour'], df['certification']),
            annot=True, fmt='d', cmap='Blues')
plt.title('Heatmap: Colour vs Certification')
plt.show()
```

Meaning: Visualizes frequency counts in a color-coded matrix; makes patterns in categorical relationships immediately apparent

---

Multivariate Analysis (Three or More Variables)

Correlation heatmap

```
# Select numerical variables
num_df = df[['carat', 'price']]
corr_matrix = num_df.corr()

plt.figure(figsize=(6, 4))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

Meaning: Visualizes pairwise correlations between numerical variables; identifies strong linear relationships

Pair plot (Scatterplot Matrix - SPLOM)

```
sns.pairplot(df, hue='certification', vars=['carat', 'price'])
plt.suptitle('Pair Plot: Carat and Price, colored by Certification', y=1.02)
plt.show()
```

Meaning: Creates a matrix of scatter plots for multiple numerical variables; color-coding by categorical variable reveals patterns across groups

Grouped scatter plot with hue

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='carat', y='price', hue='clarity')
plt.title('Carat vs Price Colored by Clarity')
plt.show()
```

Meaning: Adds a third dimension to scatter plots using color; reveals how relationships vary across categories

---

Advanced Visualization Tips

Subplots for multiple visualizations

```
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
# Histogram for carat
sns.histplot(df['carat'], kde=False, ax=axes[0,0], bins=20)
axes[0,0].set_title('Histogram of Carat')
# Density plot for carat
sns.kdeplot(df['carat'], ax=axes[0,1])
axes[0,1].set_title('Density Plot of Carat')
# Continue for other plots...
```

```
plt.tight_layout()
plt.show()
```

Meaning: Creates a grid of visualizations for comparative analysis; efficient use of space

Advanced Visualization Parameters Explained

Figure Size and Layout

```
plt.figure(figsize=(10, 6))
```

- **`figsize=(width, height)`**: Controls overall dimensions of the figure in inches
  - *Example*: `(10, 6)` creates a wider landscape-oriented plot
  - *Tip*: Larger figures improve readability for presentations

```
plt.tight_layout()
```

- **`tight_layout()`**: Automatically adjusts padding between subplots to prevent overlap
  - *Use when*: You have multiple subplots or long axis labels

```
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
```

- **`subplots(rows, cols)`**: Creates a grid of subplots
  - *rows*: Number of rows in the grid
  - *cols*: Number of columns in the grid
  - *axes*: Array of axes objects to plot on individually

Title and Labels

```
plt.title('Price Distribution by Colour', fontsize=16, fontweight='bold')
```

- **`title('text')`**: Sets the main title of the plot
  - *fontsize*: Text size (default=12, larger for presentations)
  - *fontweight*: 'normal', 'bold', 'light' (use 'bold' for emphasis)
  - *color*: Text color (e.g., 'red', '#1f77b4')
  - *loc*: Alignment ('left', 'center', 'right')

```
plt.xlabel('Colour', fontsize=12)
plt.ylabel('Price', fontsize=12)
```

- **`xlabel()/ylabel()`**: Sets axis labels
  - *fontsize*: Typically slightly smaller than title
  - *labelpad*: Distance between label and axis (e.g., `labelpad=10`)
  - *rotation*: For x-axis labels, use `rotation=45` to prevent overlap

Tick Parameters

```
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
```

- `xticks()/yticks()`: Controls tick mark appearance
  - *rotation*: Angle to rotate tick labels (45° prevents overlap)
  - *fontsize*: Size of tick labels (smaller than axis labels)
  - *color*: Color of tick labels
  - *You can also specify exact tick positions*: `plt.xticks([0, 50, 100, 150])`

Grid and Background

```
plt.grid(True, linestyle='--', alpha=0.7)
```

- `grid()`: Adds grid lines to improve readability
  - *linestyle*: '–' (dashed), '-' (solid), ':' (dotted), '-.' (dash-dot)
  - *alpha*: Transparency (0-1, 0.7 is good balance)
  - *color*: Grid line color (gray is standard)

```
sns.set_style("whitegrid")
```

- `set_style()`: Controls overall background style
  - *Options*: 'white', 'dark', 'whitegrid', 'darkgrid', 'ticks'
  - *Use 'whitegrid'* for data-heavy visualizations (best for presentations)

Legend Parameters

```
plt.legend(title='Clarity', bbox_to_anchor=(1.05, 1), loc='upper left')
```

- `legend()`: Controls the appearance of the legend
  - *title*: Label for the legend
  - *loc*: Location ('best', 'upper right', 'lower left', etc.)
  - *bbox_to_anchor*: Precise positioning (e.g., (1.05, 1) places outside plot to the right)
  - *fontsize*: Text size in the legend
  - *frameon*: Whether to draw a box around the legend (True/False)

Heatmap-Specific Parameters

```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

- `heatmap()`: Specialized parameters for heatmaps
  - *annot=True*: Displays the actual data values on the heatmap
  - *fmt='d'/'g'/'0.2f'*: Format for the annotations (integer, general, 2 decimal places)
  - *cmap='viridis'/'coolwarm'/'Blues'*: Color map to use
  - *vmin/vmax*: Sets minimum/maximum values for color scaling
  - *center=0*: For diverging color maps, sets the center point

Plot Aesthetics

```
sns.set_palette("Set2")
```

- **`set_palette()`**: Controls the color scheme
  - *Qualitative*: 'Set1', 'Set2', 'Paired' (for categorical data)
  - *Sequential*: 'Blues', 'Greens' (for ordered data)
  - *Diverging*: 'coolwarm', 'RdBu' (for data with natural midpoint)

```
sns.scatterplot(data=df, x='carat', y='price', hue='clarity', size='carat')
```

- **Additional visualization parameters**:
  - *hue*: Variable for color encoding (categorical)
  - *size*: Variable for point size encoding (numerical)
  - *style*: Variable for marker style encoding
  - *alpha*: Transparency of points (0-1)
  - *s*: Fixed size of points (e.g., `s=50`)