CoGrammar

Welcome to this session:

Task Walkthrough - Task 11 - 15

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Data Science

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: Questions

Skills Bootcamp Data Science

- For all **non-academic questions**, please submit a query: ***www.hyperiondev.com/support***

- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***

- We would love your feedback on lectures: ***Feedback on Lectures***

- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

- **Load and explore datasets using pandas** to understand their structure and contents.
- **Perform basic data manipulations** on DataFrames, such as filtering, sorting, and summarizing data.
- **Create visualizations using matplotlib and seaborn** to identify trends, patterns, and relationships in data.
- **Combine DataFrame operations with visualizations** to generate comprehensive data analysis reports.

Task Walkthrough

As a data scientist for a retail company, your goal is to analyze, clean, and visualize sales data to uncover trends in customer behavior. Using pandas for data manipulation, seaborn and matplotlib for visualization, and exploratory data analysis (EDA) techniques, you will transform raw sales data into actionable insights.

This task will guide you through:

- ❖ Loading and exploring the dataset using pandas
- ❖ Cleaning and preprocessing data for consistency and accuracy
- ❖ Performing exploratory data analysis (EDA) to summarize key trends
- ❖ Visualizing customer and sales patterns with matplotlib and seaborn

Which seaborn visualization is best for understanding the relationship between numerical variables?

A. Bar plot
B. Line plot
C. Heatmap
D. Histogram

CoGrammar

What pandas function is used to fill missing values in a categorical column with the most frequent value?

A. .fillna(value=0)
B. .mode()[0]
C. .dropna()
D. .astype(str)

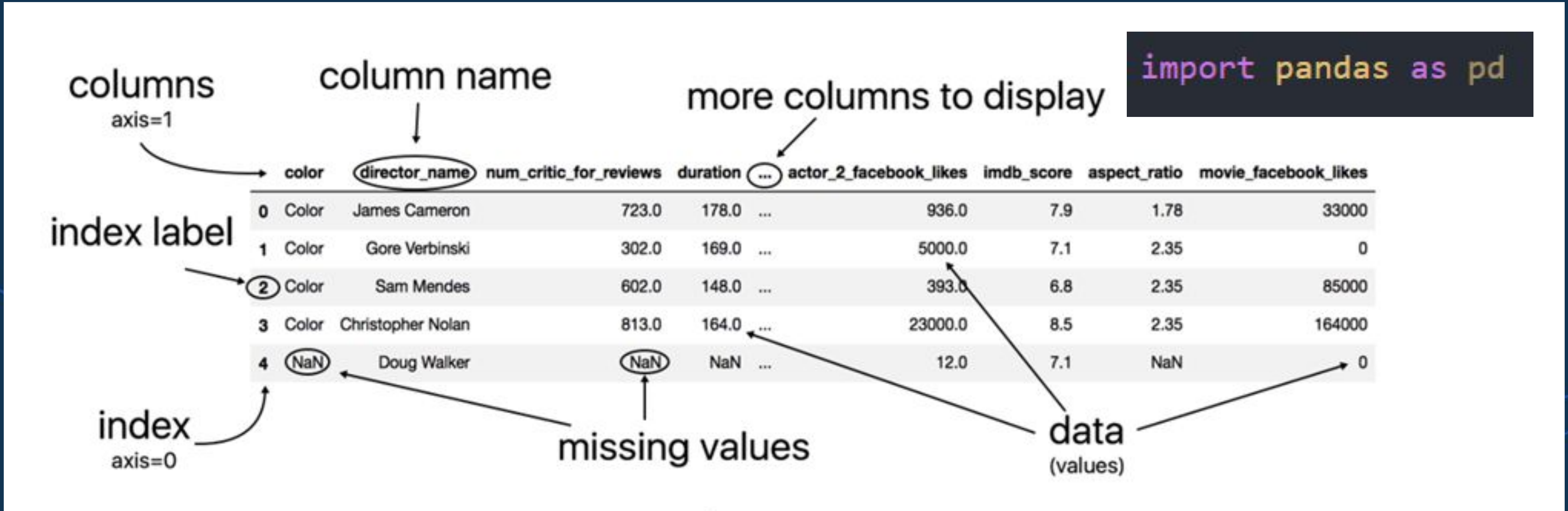
CoGrammar

Pandas DataFrame

HyperionDev

Pandas DataFrame

- The pandas' library documentation defines a DataFrame as a "two-dimensional, size-mutable, with labelled rows and columns."



	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

Anatomy of a DataFrame

Pandas DataFrame

- Pandas provides functions like **pd.read_csv()**, **pd.read_excel()**, **pd.read_sql()**, to bring your data directly into your coding environment as DataFrames.

- This is where you start turning your raw data into something easily workable.

```python
import pandas as pd
```

```python
# url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv'
# df = pd.read_csv(url)

iris = datasets.load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
```

Exploring datasets

- **df.head(), df.tail()**: Peek at the top and bottom rows for initial understanding

```
df.head()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Exploring datasets

- **df.head(), df.tail()**: Peek at the top and bottom rows for initial understanding

```
df.tail()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

Exploring datasets

- **df.shape:** Tells you the dimensions (rows, columns) of your data.

```
df.shape
✓ 0.0s
(150, 5)
```

Exploring datasets

❖ **df.info():** Gives the data types of each column, and if columns have missing values

```
df.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

Exploring datasets

- **df.describe():** Quick summary statistics for numerical columns.

```
df.describe()
```

✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

Manipulating Data

HyperionDev

Manipulating Data

- **Selecting Columns**: You often work with a **subset of features**.

- Using df[['column1', 'column2']] gets you only specific columns.

```
df.columns
```

✓ 0.0s

```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'species'],
      dtype='object')
```

```
# Select specific columns
df_selected = df[['species', 'petal length (cm)', 'petal width (cm)']]
```

✓ 0.0s

Manipulating Data

- **Filtering Rows:** Focus on specific subsets meeting certain conditions, e.g., df[df['species'] == 'setosa’]

```python
# Filter by flower species
df_setosa = df[df['species'] == 'setosa']
```

✓ 0.0s

Manipulating Data

- **Creating New Columns:** Derived features, e.g., calculating area from length and width.

```python
# Create a new calculated column
df['petal area (cm^2)'] = df['petal length (cm)'] * df['petal width (cm)']
```

✓ 0.0s

Manipulating Data

- **Renaming/Dropping:** Improve clarity or get rid of unneeded data.

```
# Rename a column
df = df.rename(columns={'sepal length (cm)': 'sepal_len'})
✓ 0.0s
```

- Data manipulation gives you a **highly customized DataFrame** focused on your exact analysis needs.

Built-in Methods

- Pandas offers a toolbox of functions for calculations:
  - **mean()** - Computes the mean for each column.
  - **min()** - Computes the minimum for each column.
  - **max()** - Computes the maximum for each column.
  - **std()** - Computes the standard deviation for each column.
  - **var()** - Computes the variance for each column.
  - **unique()** - Computes the number of unique values in each column.

- This is the start of understanding the characteristics of your data.

Grouping and Aggregation

- df.groupby(): Divide your data **based on categories** in a column (e.g., group by species).

```
print(df['petal area (cm^2)'].mean())
print(df['species'].nunique())
print(df.groupby('species')['petal length (cm)'].std())
```

✓ 0.0s

```
5.794066666666667
3
species
0    0.173664
1    0.469911
2    0.551895
Name: petal length (cm), dtype: float64
```

HyperionDev

Grouping and Aggregation

❖ **.agg()**: Apply calculations within each group (e.g., average length, maximum width).

```
df.groupby('species').agg(
    mean_petal_length=('petal length (cm)', 'mean'),
    max_sepal_width=('sepal width (cm)', 'max')
)
```

✓ 0.0s

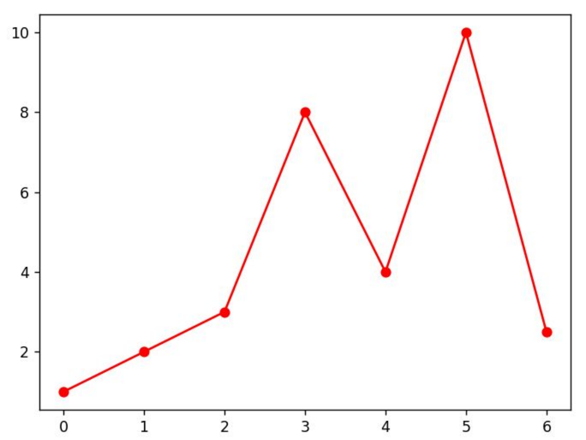
	mean_petal_length	max_sepal_width
species		
0	1.462	4.4
1	4.260	3.4
2	5.552	3.8

Matplotlib

HyperionDev

Matplotlib

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([1, 2, 3, 8, 4, 10, 2.5])

plt.plot(ypoints, 'o-r')

plt.show()
```



Markers
o = Circle, * = Star,
. = Point, x = Cross,
s = Square, d = diamond

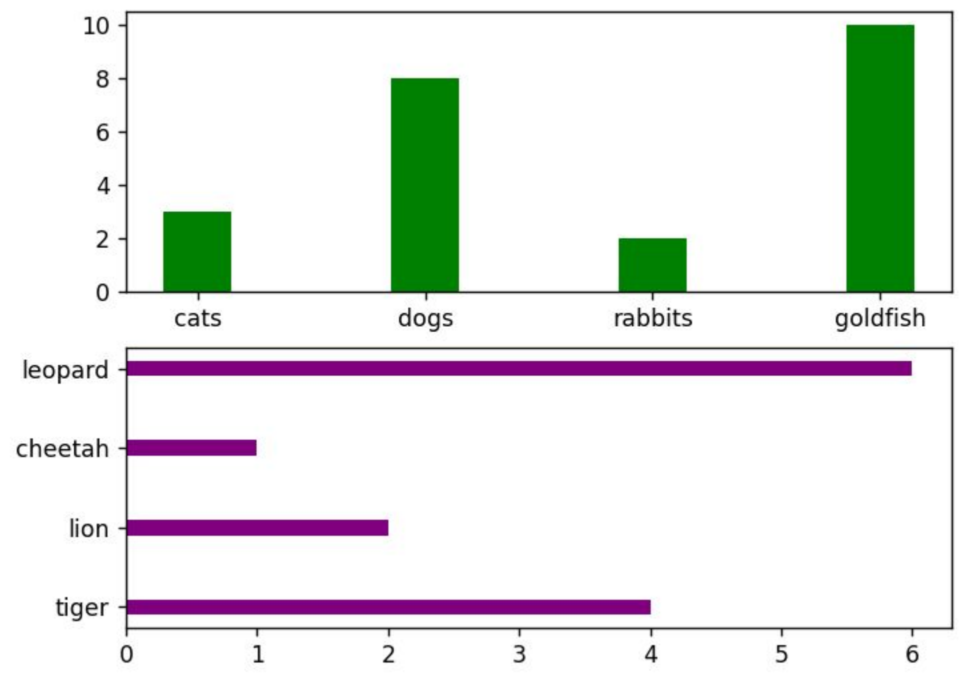
Linestyle ls
- Solid line
: Dotted line
-- Dashed line
-. Dashed/dotted line

Colour ***List of colours***
b = Blue, r = Red, g = Green,
c = Cyan, m = Magenta, y =
Yellow, k = Black, w = White

HyperionDev

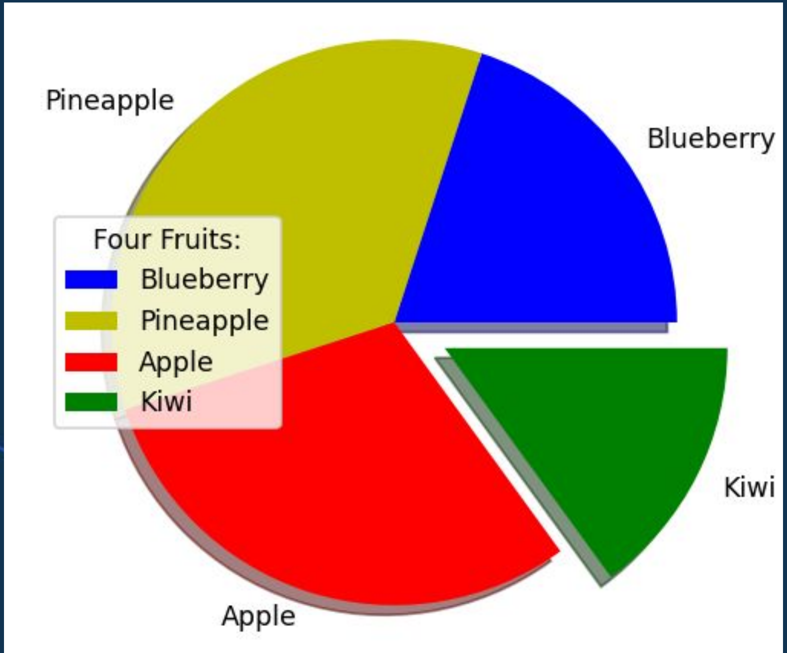
Matplotlib: Barplot

```python
import matplotlib.pyplot as plt
import numpy as np

#x,y for 1st and 2nd barplots
x1 = np.array(["cats", "dogs", "rabbits", "goldfish"])
y1 = np.array([3, 8, 2, 10])
x2 = np.array(["tiger", "lion", "cheetah", "leopard"])
y2 = np.array([4, 2, 1, 6])

#Subplot, parameters(rows, columns, index of current plot)
plt.subplot(2,1,1)
plt.bar(x1, y1, color = 'g', width=0.3)
plt.subplot(2,1,2)
plt.barh(x2, y2, color = '#800080', height=0.2)
plt.show()
```



HyperionDev

Matplotlib: Pie chart



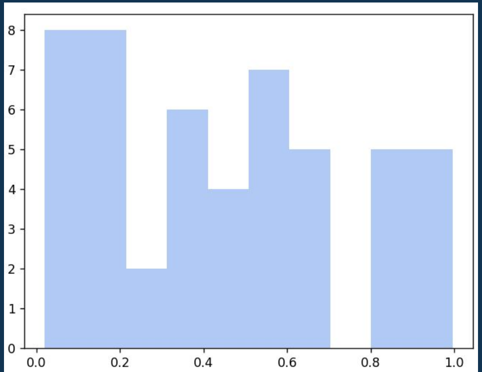

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([20, 35, 30, 15])
mylabels = ['Blueberry', 'Pineapple','Apple', 'Kiwi']
mycolors = ['b', 'y', 'r', 'g']
myexplode = [0, 0, 0, 0.2]

plt.pie(y, labels = mylabels, colors = mycolors,
        explode = myexplode, shadow = True)
plt.legend(loc='center left',title = "Four Fruits:")
plt.show()
```

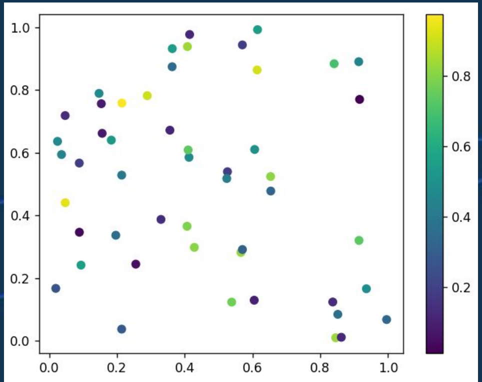
HyperionDev

Histograms & Scatterplots

A graph showing frequency distributions, the number of observations within each given interval.

```
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)

plt.hist(x, color='cornflowerblue', alpha=0.5)
plt.show()

plt.scatter(x, y, c=colors)
plt.colorbar()
plt.show()
```





HyperionDev

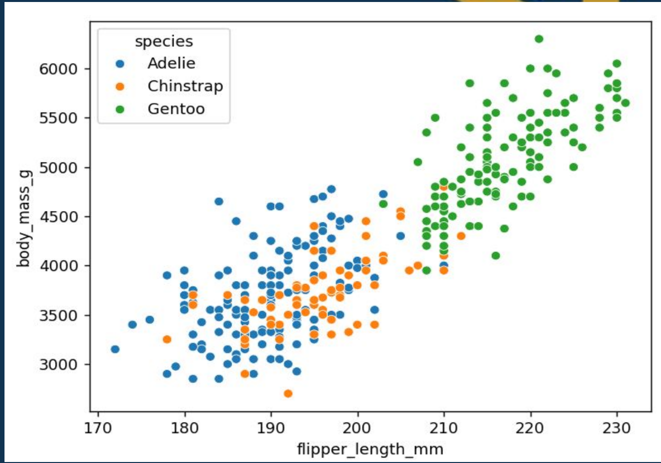
Seaborn

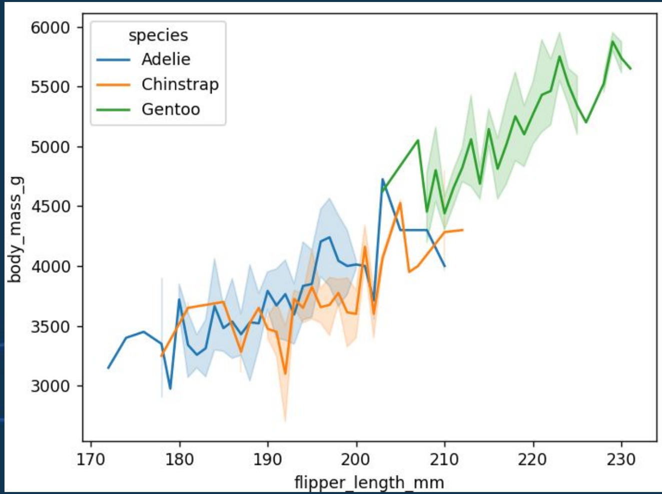
HyperionDev

Seaborn plots

Scatter and Line plots

```
#Scatterplot
sns.scatterplot(x="flipper_length_mm", y="body_mass_g",
                data=peng_df, hue="species")
plt.show()

#Lineplot
sns.lineplot(x="flipper_length_mm", y="body_mass_g",
             data=peng_df, hue="species")
plt.show()
```





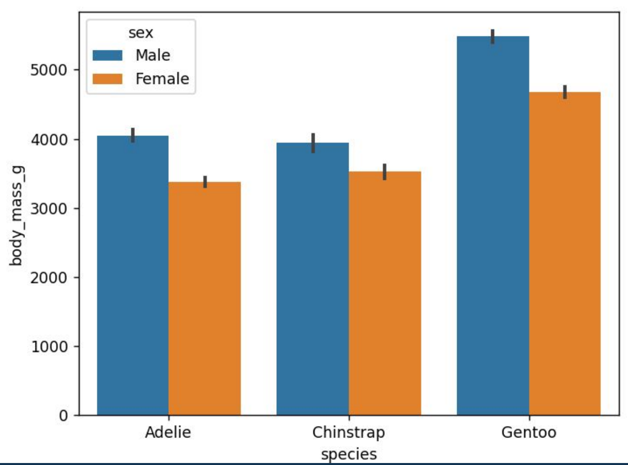
❖ If not in Jupyter or IPython notebook, explicitly call matplotlib.pyplot for displaying the plot

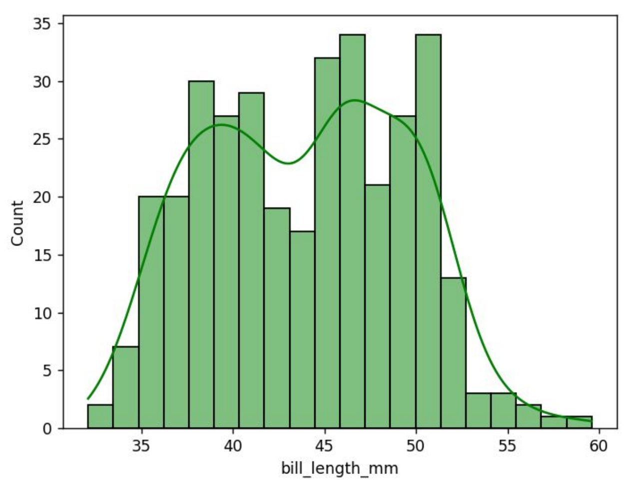
HyperionDev

Seaborn plots

Bar plots and histogram

```
#Barplot
sns.barplot(x="species", y="body_mass_g",
            hue="sex", data=peng_df)
plt.show()

#Histogram
sns.histplot(x="bill_length_mm", data=peng_df,
             bins=20, kde=True, color="green")
plt.show()
```



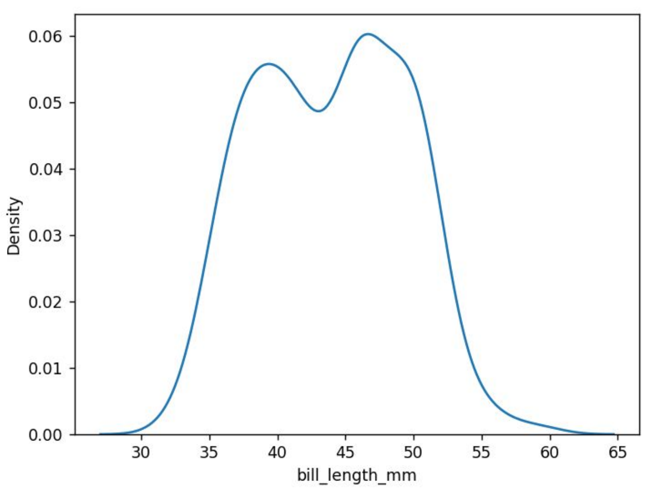


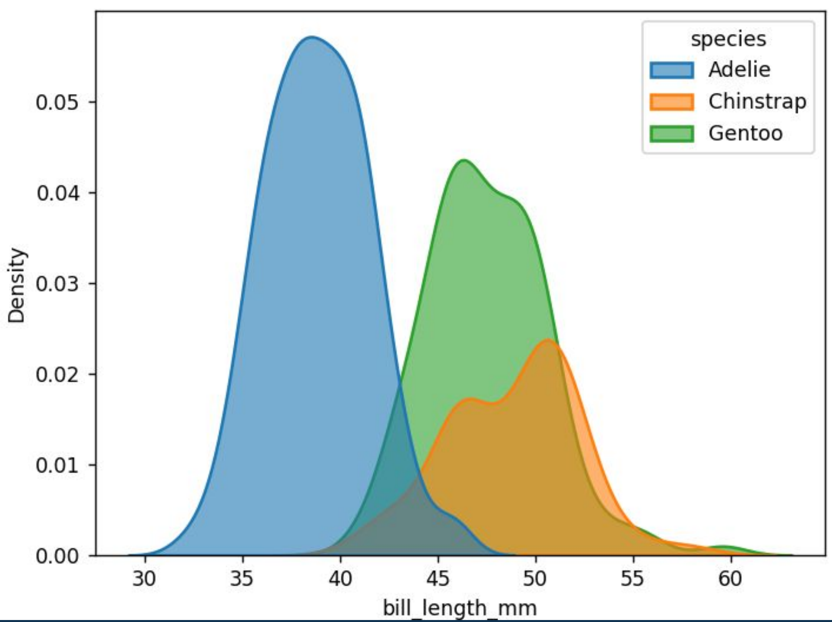
Seaborn plots

```
#Kernel density plot
sns.kdeplot(data=peng_df, x="bill_length_mm")
plt.show()
sns.kdeplot(data=peng_df, x="bill_length_mm",
            hue="species", fill=True, alpha=0.6,
            linewidth=1.5)
plt.show()
```

Kernel density plots:

Smooth curves representing density of data points, great for comparing distributions of several groups.

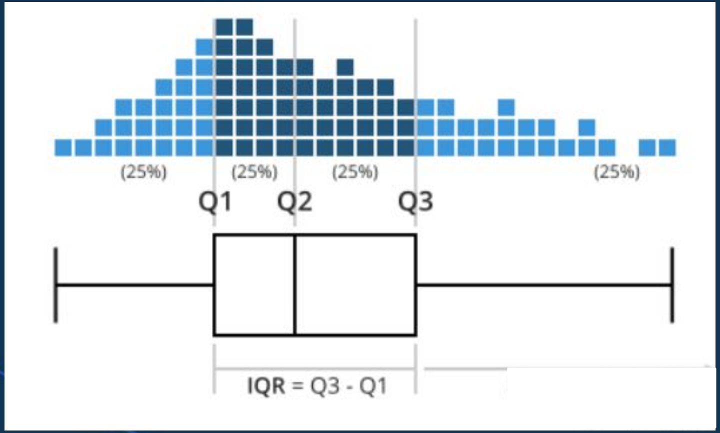


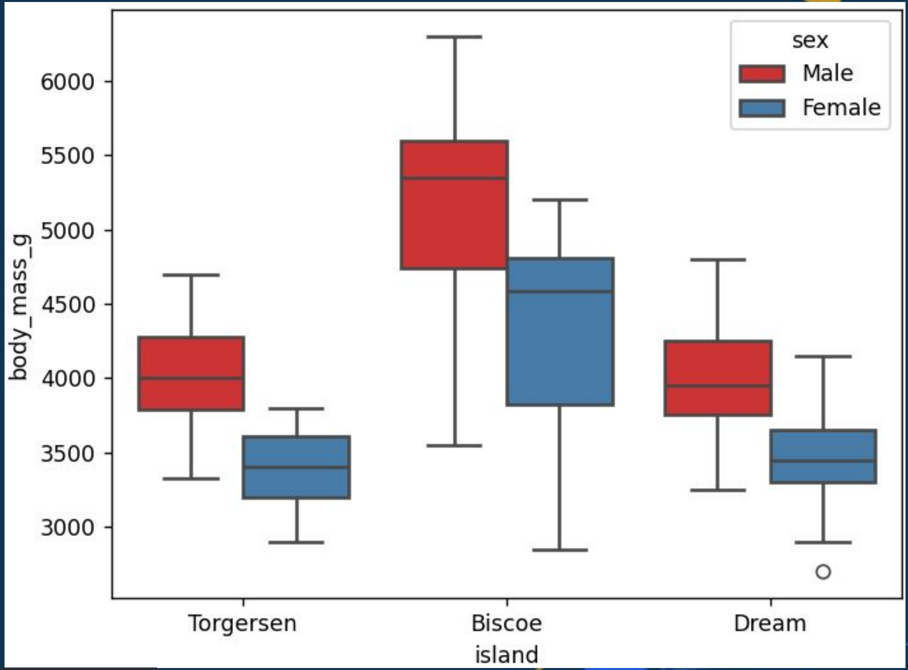


HyperionDe

Seaborn plots

Box plot



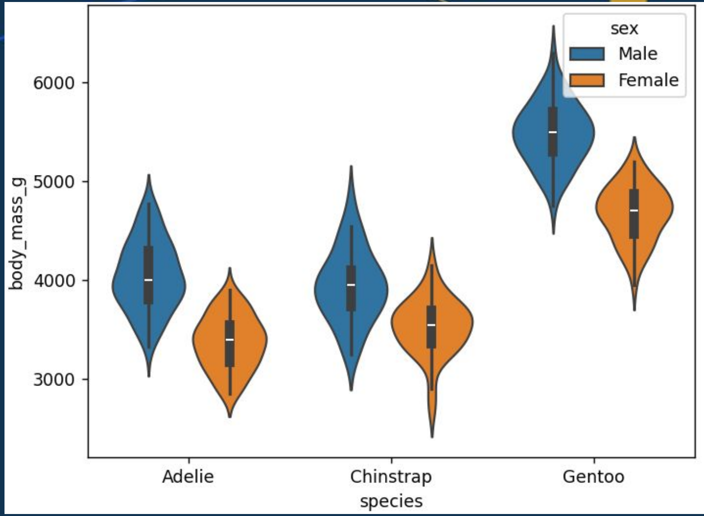



```python
#Boxplot
sns.boxplot(x="island", y="body_mass_g", hue="sex",
            data=peng_df, palette="Set1", linewidth=1.5)
plt.show()
```

Seaborn plots

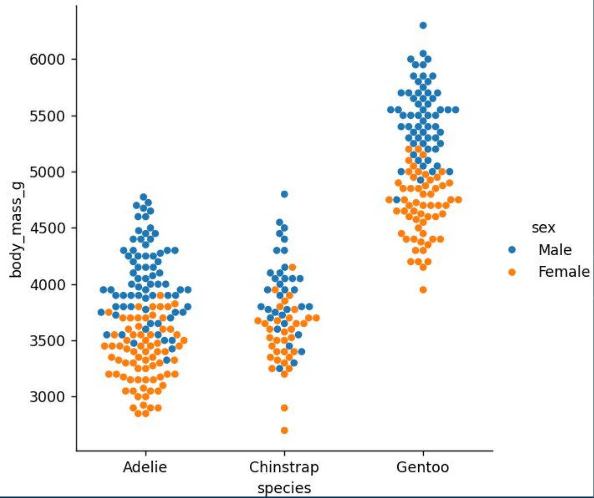
Violin plot: Combine aspects of KDEs and boxplots, ideal for showing density alongside summary statistics.

```
#Violinplot
sns.violinplot(x="species", y="body_mass_g",
                hue="sex", data=peng_df)
plt.show()
```



Categorical plot

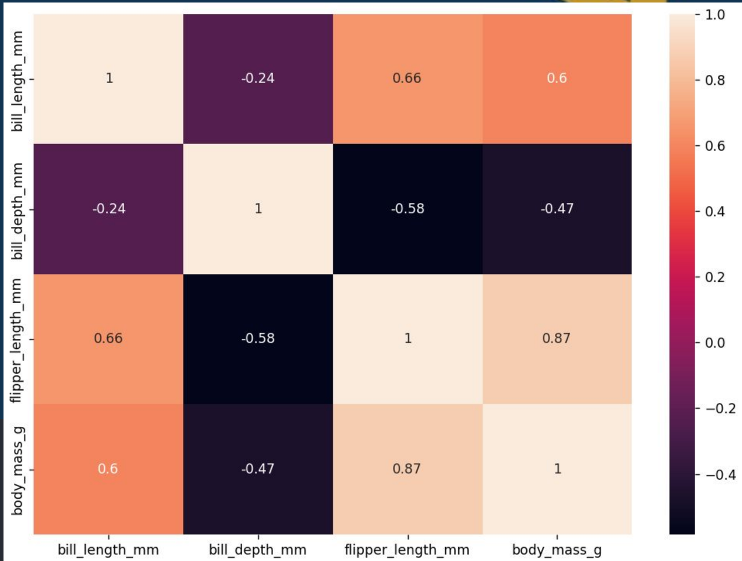
```
#Categorical plot
sns.catplot(data=peng_df, kind="swarm",
            x="species", y="body_mass_g", hue="sex")
plt.show()
```



Seaborn plots

Heatmap

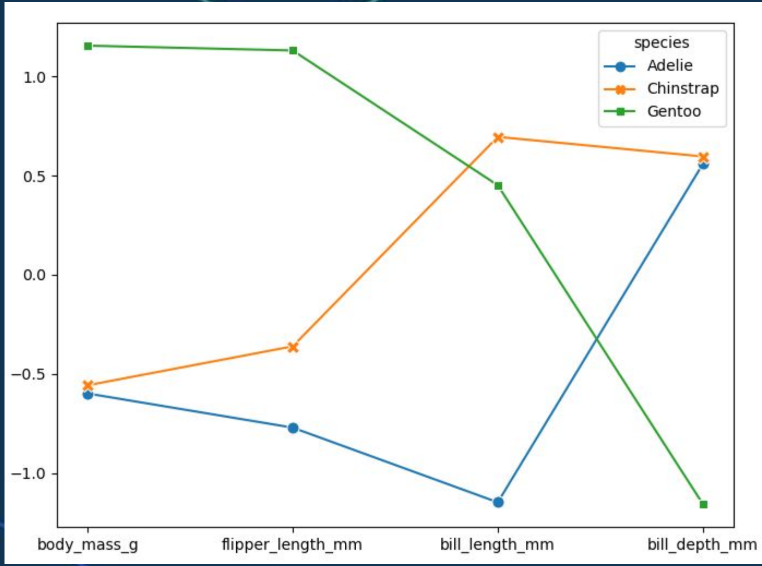
Color-coded matrices excellent for **revealing structure, highlighting correlations, and identifying clusters**.




```
#Heatmap
# Create correlation between variables (floats only)
print(peng_df.dtypes)
peng_float = peng_df.select_dtypes(include=[np.float64])
corr = peng_float.corr()

#Plot
plt.figure(figsize=(10,7))
sns.heatmap(corr, annot=True)
plt.show()
```

Customization in heatmaps is key for optimal interpretation.

Parallel coordinates



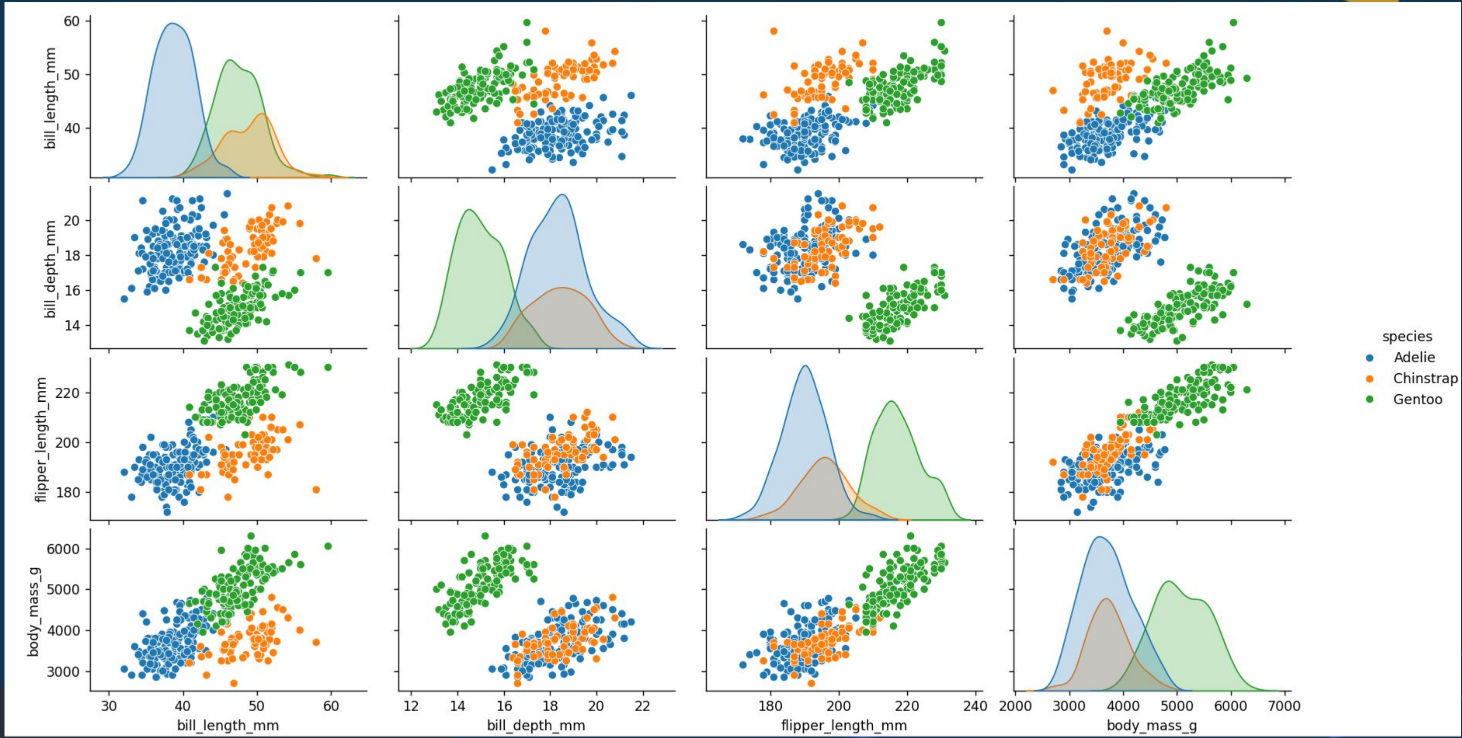
- Show many dimensions on the same plot
- Each feature has a vertical axis, data points become lines crossing them.
- Ideal when you have many interrelated variables and need to spot outliers or group characteristics.

```
#Parallel coordinates
# Calculate the average values for each continent
average_data = peng_df.groupby('species')[['body_mass_g', 'flipper_length_mm',
                                           'bill_length_mm','bill_depth_mm']].mean()

# Normalize the data for better visualization
normalized_data = (average_data - average_data.mean()) / average_data.std()

# Create parallel plot
plt.figure(figsize=(8, 6))
parallel_plot = sns.lineplot(data=normalized_data.transpose(),
                             dashes=False,
                             markers=True,
                             markersize=8)
plt.show()
```

HyperionDev

Seaborn plots

Pairplot



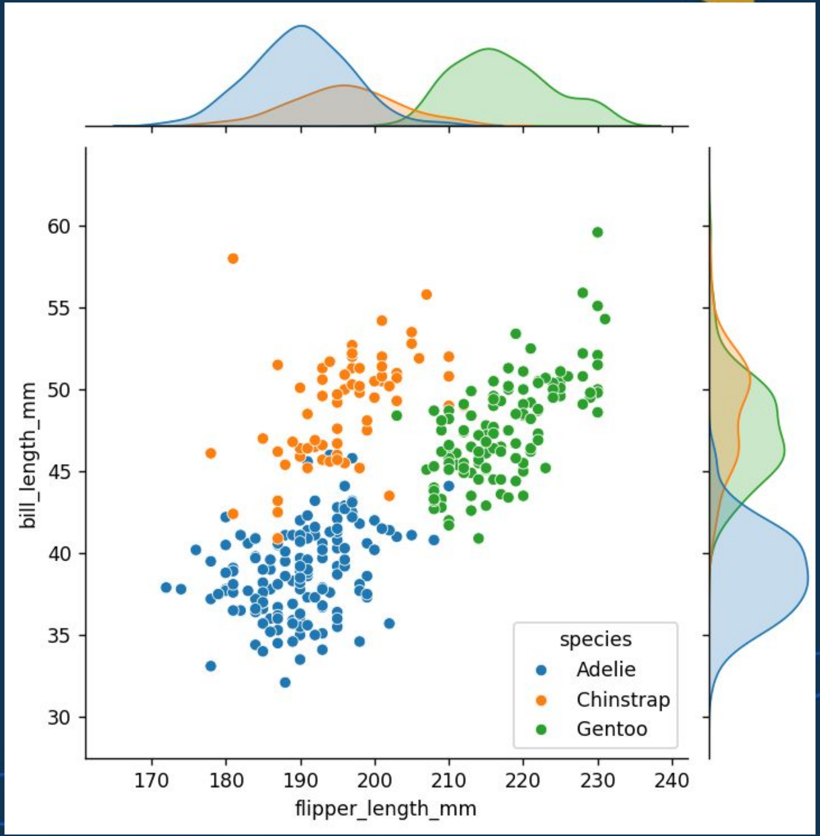

```python
#Pairplot
sns.pairplot(peng_df, hue="species")
plt.show()
```

Compact way to depict pairwise relationships between several variables simultaneously.

Seaborn plots

Jointplot

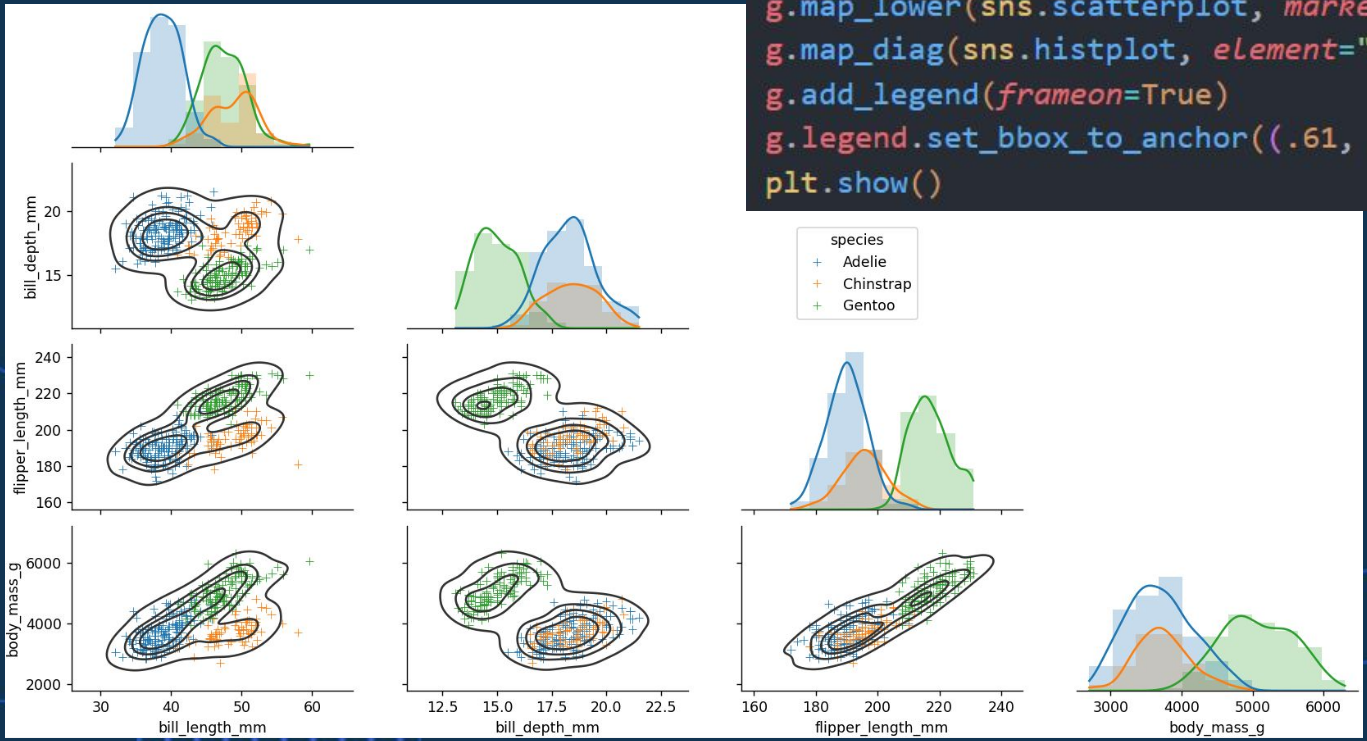
```
#Jointplot
sns.jointplot(data=peng_df, x="flipper_length_mm",
              y="bill_length_mm", hue="species")
plt.show()
```



HyperionDev

Combination plots

```python
#Combination plots
g = sns.PairGrid(peng_df, hue="species", corner=True)
g.map_lower(sns.kdeplot, hue=None, levels=5, color=".2")
g.map_lower(sns.scatterplot, marker="+")
g.map_diag(sns.histplot, element="step", linewidth=0, kde=True)
g.add_legend(frameon=True)
g.legend.set_bbox_to_anchor((.61, .6))
plt.show()
```



Task Walkthrough

As a data scientist for a retail company, your goal is to analyze, clean, and visualize sales data to uncover trends in customer behavior. Using pandas for data manipulation, seaborn and matplotlib for visualization, and exploratory data analysis (EDA) techniques, you will transform raw sales data into actionable insights.

<u>This task will guide you through:</u>

- ❖ Loading and exploring the dataset using pandas
- ❖ Cleaning and preprocessing data for consistency and accuracy
- ❖ Performing exploratory data analysis (EDA) to summarize key trends
- ❖ Visualizing customer and sales patterns with matplotlib and seaborn

What is the purpose of converting the "Date" column to datetime format?

A. To calculate the total revenue per transaction
B. To perform time-based analysis and extract trends
C. To remove missing values
D. To count the number of unique customers

What is the purpose of a box plot in exploratory data analysis (EDA)?

A. To visualize correlations between numerical variables

B. To show the distribution and outliers of a numerical variable

C. To compare categorical data

D. To display seasonal trends

Summary

★ **Datasets & DataFrames**
Load and explore datasets with pandas
Inspect and clean missing data
Standardize categorical values

★ **Data Cleaning & Preprocessing**
Convert Date to datetime format
Fill missing values appropriately
Compute new calculated fields

★ **Exploratory Data Analysis (EDA)**
Identify trends in customer demographics and spending behavior
Compute aggregated revenue and quantity metrics

★ **Data Visualization with Matplotlib & Seaborn**
Bar charts for category-wise revenue
Histograms for customer age distribution
Box plots for spending patterns
Line charts for monthly sales trends
Heatmaps for correlation analysis

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education