

1. Sales Performance Analysis (Highest and Lowest Sales)

📌 A company records monthly sales in an array: `[5000, 7000, 8000, 6500, 7200, 9000, 8500]`. Find and print the highest and lowest sales values.

```
sales = np.array([5000, 7000, 8000, 6500, 7200, 9000, 8500])
highest_sales = np.max(sales)
lowest_sales = np.min(sales)

print("Highest Sales:", highest_sales)
print("Lowest Sales:", lowest_sales)
```

2. Reshape a 1D Array into a 2D Matrix

📌 A company's quarterly sales data is stored in a 1D NumPy array: `[1000, 1200, 1500, 1800, 2000, 2100, 2500, 2700]`. Convert it into a 2D array with 4 rows and 2 columns representing sales per quarter.

```
sales_data = np.array([1000, 1200, 1500, 1800, 2000, 2100, 2500, 2700])
reshaped_sales = sales_data.reshape(4, 2)
print("Reshaped Quarterly Sales Data:\n", reshaped_sales)
```

3. Calculate Total Revenue from Products Sold

📌 A store sells 4 types of products. The quantity sold and the price per unit are stored as NumPy arrays:

```
quantities = np.array([10, 15, 7, 20]) # Number of units sold
prices = np.array([50, 40, 100, 30])    # Price per unit
```

```
quantities = np.array([10, 15, 7, 20])
prices = np.array([50, 40, 100, 30])

revenue_per_product = quantities * prices
total_revenue = np.sum(revenue_per_product)

print("Revenue per product:", revenue_per_product)
print("Total Store Revenue:", total_revenue)
```

4. Find Students Who Passed (Marks ≥ 40)

📌 A class has students' marks stored in an array: `[35, 60, 42, 75, 29, 90, 55]`. Find and print the marks of students who passed (≥ 40).

```
marks = np.array([35, 60, 42, 75, 29, 90, 55])
passed_students = marks[marks >= 40]
print("Marks of students who passed:", passed_students)
```

5. Find the Most Frequent Element in an Array

📌 Given a NumPy array `[1, 2, 2, 3, 3, 3, 4, 4, 4, 4]`, determine which element appears the most times.

```
values = np.array([1, 2, 2, 3, 3, 3, 4, 4, 4, 4])
unique, counts = np.unique(values, return_counts=True)
most_frequent = unique[np.argmax(counts)]
print("Most frequent element:", most_frequent)
```

6. Normalize an Array (Scaling Values Between 0 and 1)

📌 You have an array of exam scores: `[50, 80, 90, 60, 75, 85]`. Normalize them between 0 and 1 using Min-Max Scaling.

```
scores = np.array([50, 80, 90, 60, 75, 85])
normalized_scores = (scores - np.min(scores)) / (np.max(scores) -
np.min(scores))
print("Normalized scores:", normalized_scores)
```

Dataset => [Social_Network_Ads.csv](https://mitu.co.in)
<https://mitu.co.in>

7. Load the Dataset and Display Basic Info

📌 You are given the `Social_Network_Ads.csv` dataset. Load it into a Pandas DataFrame and display the first 5 rows along with dataset summary information.

```
import pandas as pd

df = pd.read_csv("Social_Network_Ads.csv") # Load dataset
print(df.head()) # Display first 5 rows
print(df.info()) # Display dataset summary
```

8. Find the Number of Users by Gender

📌 How many male and female users are present in the dataset?

```
gender_counts = df["Gender"].value_counts()
print(gender_counts)
```

9. Find the Average Age of Users Who Purchased the Product

📌 Calculate the average age of users who have purchased the product (`Purchased == 1`).

```
average_age = df[df["Purchased"] == 1]["Age"].mean()
print("Average age of users who purchased:", average_age)
```

10. Identify High Earners (Above ₹100,000) Who Didn't Purchase

📌 Find all users who have an `EstimatedSalary` greater than ₹100,000 but didn't purchase the product (`Purchased == 0`).

```
high_earners_no_purchase = df[(df["EstimatedSalary"] > 100000) &
(df["Purchased"] == 0)]
print(high_earners_no_purchase)
```

11. Find the Gender of Users Who Made Purchases

📌 What is the count of male and female users who purchased the product?

```
gender_purchases = df[df["Purchased"] == 1]["Gender"].value_counts()
```

```
print(gender_purchases)
```

12. Find the Age Group with the Highest Purchase Rate

📌 Group users into age brackets (18-25, 26-35, 36-45, 46+) and find which age group has the highest purchase rate.

```
# Define age bins
bins = [18, 25, 35, 45, 100]
labels = ["18-25", "26-35", "36-45", "46+"]
df["Age Group"] = pd.cut(df["Age"], bins=bins, labels=labels, right=False)

# Calculate purchase rate per age group
age_group_purchases = df.groupby("Age Group")["Purchased"].mean()
print(age_group_purchases)
```

13. Calculate the Percentage of Users Who Purchased (Overall and By Gender)

📌 What percentage of total users purchased the product? Also, break it down by gender.

```
overall_purchase_rate = (df["Purchased"].sum() / len(df)) * 100
gender_wise_purchase_rate = df.groupby("Gender")["Purchased"].mean() * 100

print("Overall Purchase Rate:", overall_purchase_rate, "%")
print("Purchase Rate by Gender:\n", gender_wise_purchase_rate)
```

Dataset => iris.csv
<https://mitu.co.in>

14. Load the Dataset and Display Basic Information

📌 Load the dataset and display the first five rows, along with dataset summary statistics.

```
import pandas as pd
from sklearn.datasets import load_iris
```

```
# Load dataset using sklearn
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df["species"] = iris.target

# Mapping species to actual names
species_mapping = {0: "setosa", 1: "versicolor", 2: "virginica"}
df["species"] = df["species"].map(species_mapping)


print(df.head()) # First five rows
print(df.describe()) # Summary statistics
```

15. Find the Average Sepal Length for Each Species

 Compute the average sepal length for each species of iris flower.

```
average_sepal_length = df.groupby("species")["sepal length (cm)"].mean()
print(average_sepal_length)
```

16. Identify the Species with the Largest Petal Width

 Find the species that has the largest average petal width.

```
max_petal_width_species = df.groupby("species")["petal width (cm)"].mean().idxmax()
print("Species with largest petal width:", max_petal_width_species)
```

17. Find the Most Common Petal Length

 Find the most frequently occurring petal length value in the dataset.

```
most_common_petal_length = df["petal length (cm)"].mode()[0]
print("Most common petal length:", most_common_petal_length)
```

18. Count How Many Samples Have Sepal Width Between 2.5 cm and 3.5 cm

📌 Count how many iris samples have a *sepal width* between 2.5 cm and 3.5 cm.

```
sepal_width_range_count = df[(df["sepal width (cm)"] >= 2.5) & (df["sepal width (cm)"] <= 3.5)].shape[0]
print("Number of samples with sepal width between 2.5 cm and 3.5 cm:", sepal_width_range_count)
```

19. Add a Column Indicating Large or Small Petals

📌 Create a new column *Petal_Size* which labels samples as "Large" if the petal length is greater than 4 cm, otherwise "Small".

```
df["Petal_Size"] = df["petal length (cm)"].apply(lambda x: "Large" if x > 4 else "Small")
print(df.head())
```

20. Save the Updated Dataset

📌 Save the updated dataset with the *Petal_Size* column to a CSV file named *Updated_Iris_Dataset.csv*.

```
df.to_csv("Updated_Iris_Dataset.csv", index=False)
print("Updated dataset saved successfully!")
```

Dataset => iris.csv

<https://mitu.co.in>

21. Create a Scatter Plot of Sepal Length vs Sepal Width

📌 You are analyzing the relationship between sepal length and sepal width. Create a scatter plot with species as different colors.

```
# Load Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df["species"] = iris.target
```

```
species_mapping = {0: "setosa", 1: "versicolor", 2: "virginica"}
df["species"] = df["species"].map(species_mapping)

# Scatter Plot
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df["sepal length (cm)"], y=df["sepal width (cm)"],
hue=df["species"], palette="viridis")
plt.title("Sepal Length vs Sepal Width")
plt.xlabel("Sepal Length (cm)")
plt.ylabel("Sepal Width (cm)")
plt.legend(title="Species")
plt.show()
```

22. Create a Histogram of Petal Length

📌 You want to understand the distribution of petal length values. Plot a histogram.

```
plt.figure(figsize=(8, 5))
plt.hist(df["petal length (cm)"], bins=20, color="skyblue",
edgecolor="black")
plt.title("Distribution of Petal Length")
plt.xlabel("Petal Length (cm)")
plt.ylabel("Frequency")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

23. Create a Box Plot for Sepal Length for Each Species

📌 You need to compare the spread and distribution of *sepal length* among different species using a box plot.

```
plt.figure(figsize=(8, 5))
sns.boxplot(x="species", y="sepal length (cm)", data=df, palette="Set2")
plt.title("Sepal Length Distribution by Species")
plt.xlabel("Species")
plt.ylabel("Sepal Length (cm)")
plt.show()
```

24. Create a Pair Plot of the Iris Dataset

📌 You want to explore pairwise relationships between all features for different species using a pair plot.

```
sns.pairplot(df, hue="species", palette="husl", diag_kind="kde")  
plt.show()
```

25. Create a Count Plot of Different Species

📌 You need to visualize how many samples belong to each species.

```
plt.figure(figsize=(6, 5))  
sns.countplot(x="species", data=df, palette="pastel")  
plt.title("Count of Each Iris Species")  
plt.xlabel("Species")  
plt.ylabel("Count")  
plt.show()
```

26. Create a KDE Plot for Petal Width

📌 You want to visualize the probability distribution of petal width for each species using a KDE (Kernel Density Estimation) plot.

```
plt.figure(figsize=(8, 5))  
sns.kdeplot(data=df, x="petal width (cm)", hue="species", fill=True,  
palette="coolwarm")  
plt.title("Petal Width Distribution by Species")  
plt.xlabel("Petal Width (cm)")  
plt.ylabel("Density")  
plt.show()
```

27. Create a Heatmap of Correlation Between Features

📌 You want to analyze how strongly the features are correlated with each other using a heatmap.

```
plt.figure(figsize=(8, 6))  
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f",  
linewidths=0.5)  
plt.title("Feature Correlation Heatmap")
```



```
plt.show()
```

28. Create a Violin Plot for Petal Length

📌 You want to analyze the distribution and density of **petal length** for each species using a violin plot.

```
plt.figure(figsize=(8, 5))
sns.violinplot(x="species", y="petal length (cm)", data=df,
palette="muted")
plt.title("Petal Length Distribution by Species")
plt.xlabel("Species")
plt.ylabel("Petal Length (cm)")
plt.show()
```

29. Create a Subplot Grid Showing the Distribution of All Features with Different Plot Types

📌 You want to analyze the distribution of each numerical feature in the Iris dataset. Create a **2x2 subplot grid**, using:

- Histogram for **sepal length**
- Box plot for **sepal width**
- KDE plot for **petal length**
- Violin plot for **petal width**

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

# Histogram for Sepal Length
sns.histplot(df["sepal length (cm)"], bins=20, kde=True, color="darkblue",
ax=axes[0, 0])
axes[0, 0].set_title("Sepal Length Distribution")
axes[0, 0].set_xlabel("Sepal Length (cm)")

# Box Plot for Sepal Width
```

```

sns.boxplot(x=df["sepal width (cm)"], color="purple", ax=axes[0, 1])
axes[0, 1].set_title("Sepal Width Boxplot")
axes[0, 1].set_xlabel("Sepal Width (cm)")

# KDE Plot for Petal Length
sns.kdeplot(df["petal length (cm)"], fill=True, color="red", alpha=0.5,
ax=axes[1, 0])
axes[1, 0].set_title("Petal Length Density Plot")
axes[1, 0].set_xlabel("Petal Length (cm)")

# Violin Plot for Petal Width
sns.violinplot(x=df["species"], y=df["petal width (cm)"],
palette="coolwarm", ax=axes[1, 1])
axes[1, 1].set_title("Petal Width Violin Plot")
axes[1, 1].set_xlabel("Species")
axes[1, 1].set_ylabel("Petal Width (cm)")

plt.tight_layout()
plt.show()

```

30. Create a Custom Dual-Axis Line and Bar Chart Comparing Average Feature Values Across Species

📌 You need to compare the **average feature values** of each species using a combination of **bar plots and line plots on the same graph**. Implement:

- **Bars** to show average **sepal length** and **petal length**
- **A line plot** to show **sepal width** and **petal width** on a secondary y-axis
- **Custom colors, labels, and dual-axis formatting**

```

# Compute averages for each species
avg_values = df.groupby("species").mean()

# Create figure and axis
fig, ax1 = plt.subplots(figsize=(10, 6))

# Bar plot for Sepal Length and Petal Length
bar_width = 0.3

```

```
species = avg_values.index

ax1.bar(species, avg_values["sepal length (cm)"], width=bar_width,
color="blue", label="Sepal Length")
ax1.bar(species, avg_values["petal length (cm)"], width=bar_width,
color="orange", label="Petal Length", alpha=0.8, bottom=avg_values["sepal
length (cm)"])

ax1.set_xlabel("Species")
ax1.set_ylabel("Length (cm)")
ax1.set_title("Comparison of Average Feature Values Across Species")
ax1.legend(loc="upper left")

# Secondary axis for Sepal Width and Petal Width (Line Plot)
ax2 = ax1.twinx()
ax2.plot(species, avg_values["sepal width (cm)"], color="green",
marker="o", label="Sepal Width", linestyle="dashed")
ax2.plot(species, avg_values["petal width (cm)"], color="red", marker="s",
label="Petal Width", linestyle="dotted")

ax2.set_ylabel("Width (cm)")
ax2.legend(loc="upper right")

plt.show()
```