1. Air Quality Analysis: Inbuilt dataset: seaborn.load_dataset('mpg') in Python

A. Analyze missing values in the dataset and impute them appropriately.

B. Find the average ozone levels per month

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

df = sns.load_dataset('mpg')

print(df.isnull().sum())
df['horsepower'] = df['horsepower'].fillna(df['horsepower'].mean())

avg_mpg_by_year = df.groupby('model_year')['mpg'].mean().reset_index()
print(avg_mpg_by_year)
```

```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
name            0
dtype: int64
    model_year        mpg
0           70  17.689655
1           71  21.250000
2           72  18.714286
3           73  17.100000
4           74  22.703704
5           75  20.266667
6           76  21.573529
7           77  23.375000
8           78  24.061111
9           79  25.093103
10          80  33.696552
11          81  30.334483
12          82  31.709677
```

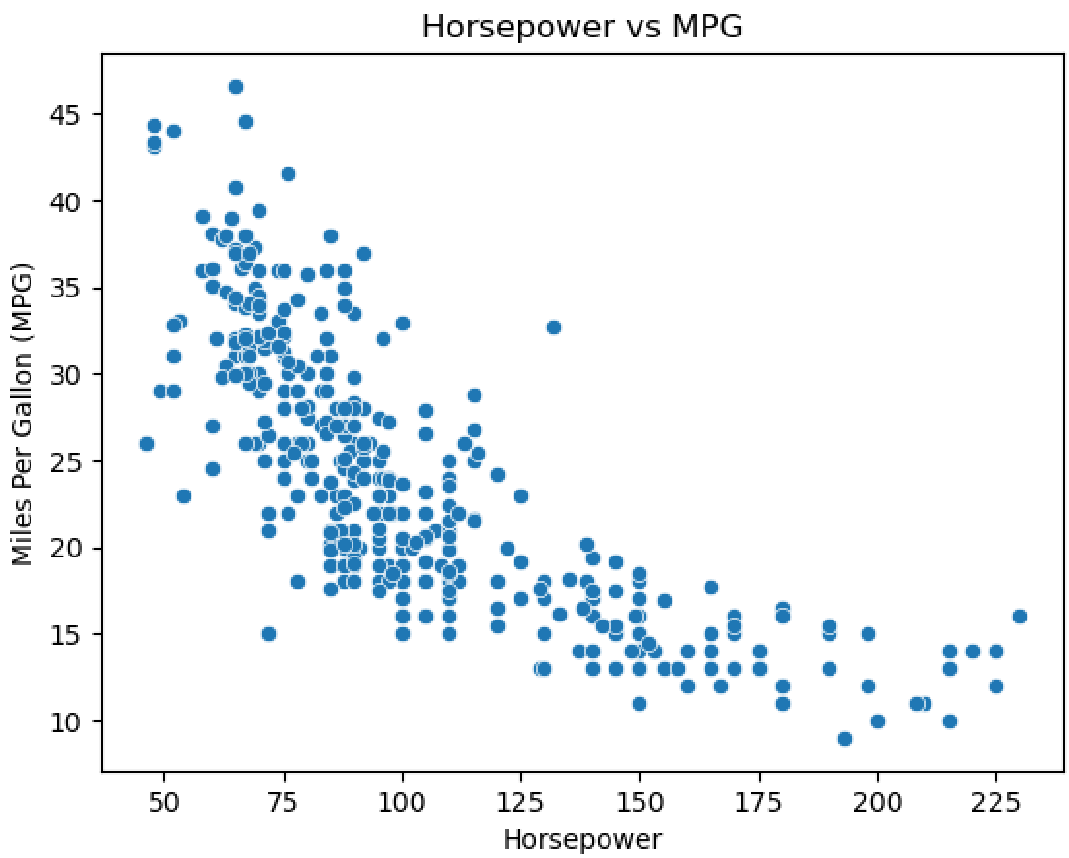
1. Car Performance Analysis: Inbuilt dataset: seaborn.load_dataset('mpg')

- Display the first 5 rows of the dataset.
- How many rows and columns does the dataset have?
- What are the names of all the columns in the dataset?
- Find the average miles per gallon (mpg) for each number of cylinders.

- Create a scatter plot to show the relationship between horsepower and mpg.

```
# dataset
df = sns.load_dataset('mpg')

print(df.head())

print(df.shape)  # (rows, columns)

print(df.columns.tolist())

avg_mpg_by_cyl = df.groupby('cylinders')['mpg'].mean()
print(avg_mpg_by_cyl)

sns.scatterplot(data=df, x='horsepower', y='mpg')
plt.title('Horsepower vs MPG')
plt.xlabel('Horsepower')
plt.ylabel('Miles Per Gallon (MPG)')
plt.show()
```

```
    mpg  cylinders  displacement  horsepower  weight  acceleration  \
0  18.0          8         307.0       130.0    3504          12.0
1  15.0          8         350.0       165.0    3693          11.5
2  18.0          8         318.0       150.0    3436          11.0
3  16.0          8         304.0       150.0    3433          12.0
4  17.0          8         302.0       140.0    3449          10.5

   model_year origin                       name
0          70    usa  chevrolet chevelle malibu
1          70    usa          buick skylark 320
2          70    usa         plymouth satellite
3          70    usa              amc rebel sst
4          70    usa                ford torino
(398, 9)
['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
'acceleration', 'model_year', 'origin', 'name']
cylinders
3    20.550000
4    29.286765
5    27.366667
6    19.985714
8    14.963107
Name: mpg, dtype: float64
```



1. . Titanic Survival Analysis: Inbuilt Dataset: seaborn.load_dataset('titanic') in Python

A. Compute the survival rate grouped by gender (sex) and passenger class (class).

B. Filter and display records of passengers who:

- Were in 1st class,
- Are female, and
- Had a fare greater than 50.

```
df = sns.load_dataset('titanic')

survival_rate = df.groupby(['sex', 'class'])
['survived'].mean().reset_index();
print(survival_rate)

filtered_passengers = df[(df['sex'] == 'female') &
                         (df['class'] == 'First') &
                         (df['fare'] > 50)]

print(filtered_passengers[['sex', 'class', 'fare']]);

      sex   class  survived
0  female   First  0.968085
```

```
1  female  Second  0.921053
2  female   Third  0.500000
3    male   First  0.368852
4    male  Second  0.157407
5    male   Third  0.135447
        sex  class      fare
1    female  First   71.2833
3    female  First   53.1000
31   female  First  146.5208
52   female  First   76.7292
61   female  First   80.0000
..      ...    ...       ...
835  female  First   83.1583
849  female  First   89.1042
856  female  First  164.8667
871  female  First   52.5542
879  female  First   83.1583

[82 rows x 3 columns]

/var/folders/kx/41v1tt6j1yx79h_8wk_hkl8w0000gn/T/
ipykernel_4467/1975499764.py:4: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  survival_rate = df.groupby(['sex', 'class'])
['survived'].mean().reset_index();
```

1. Iris Flower Classification: Inbuilt Dataset : iris in Python

A.

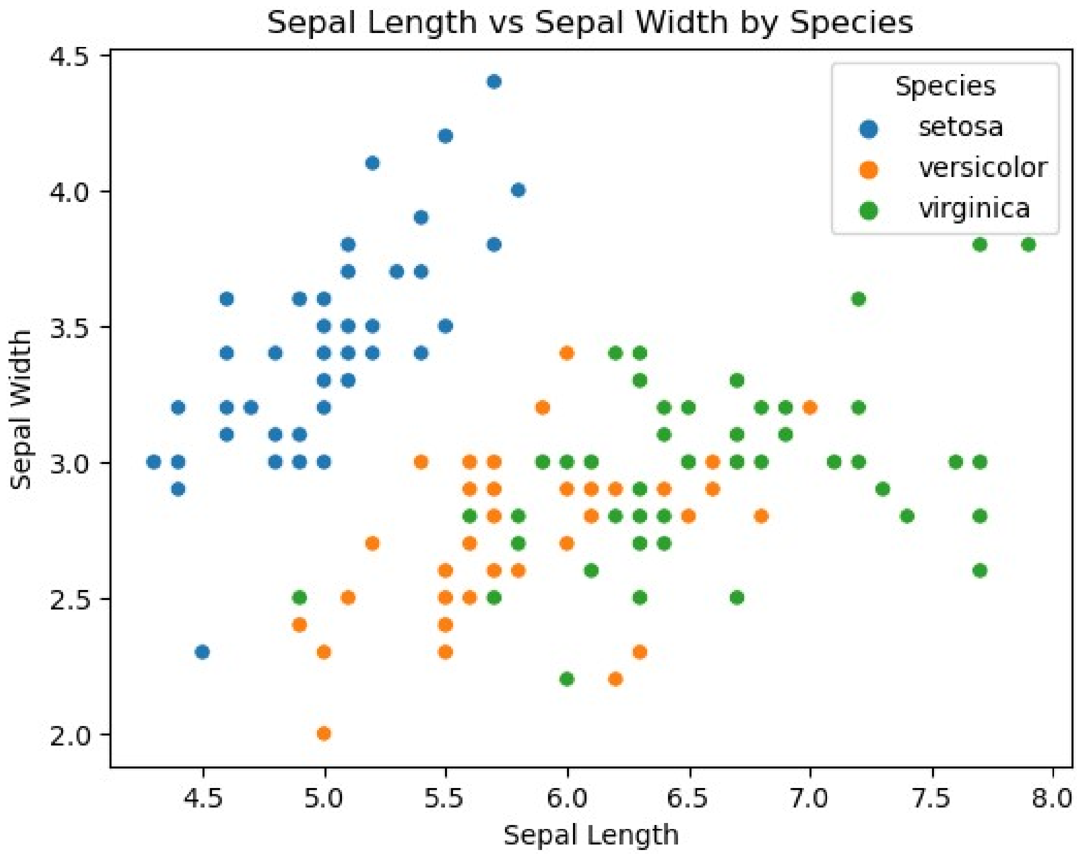
- Display basic information and summary statistics of the dataset.
- Check for missing values in each column.

B. Create a scatter plot of sepal length vs. sepal width, colored by species.

```
df = sns.load_dataset('iris')

print(df.info())

print(df.describe())

sns.scatterplot(data=df, x='sepal_length', y='sepal_width',
hue='species')
plt.title('Sepal Length vs Sepal Width by Species')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
```

```python
plt.legend(title='Species')
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

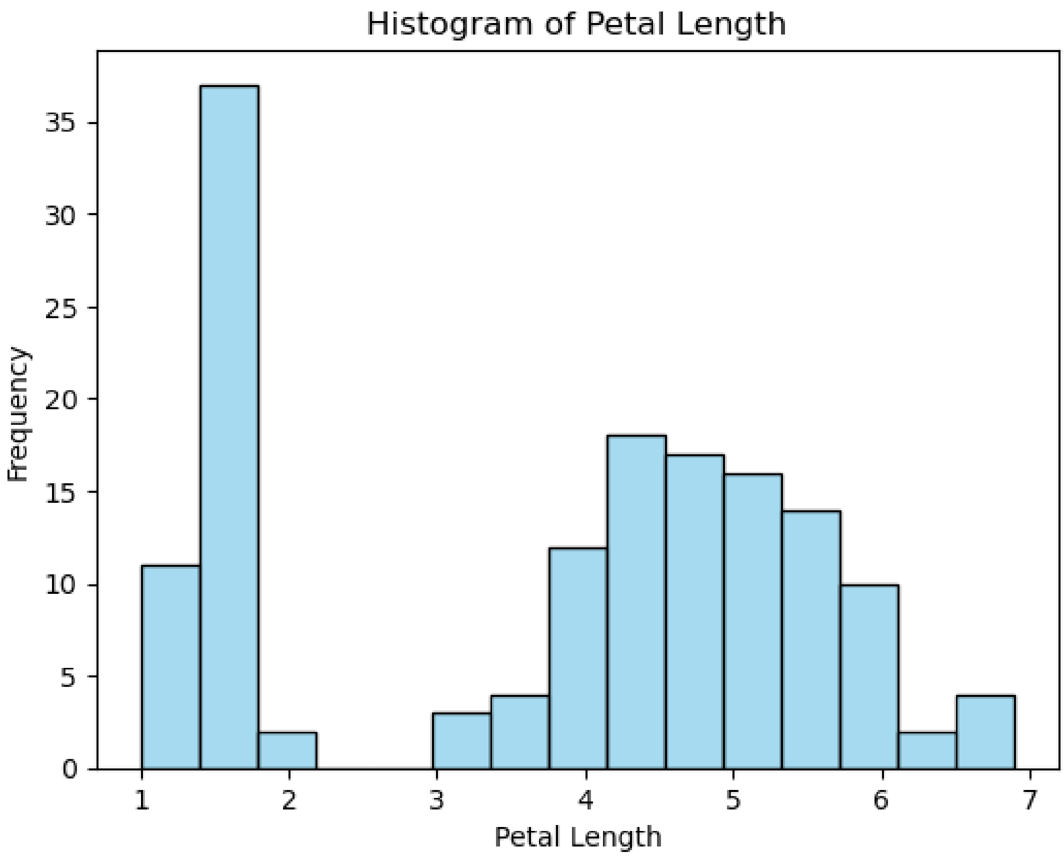


1. Distribution of Petal Length: Inbuilt dataset: iris in Python

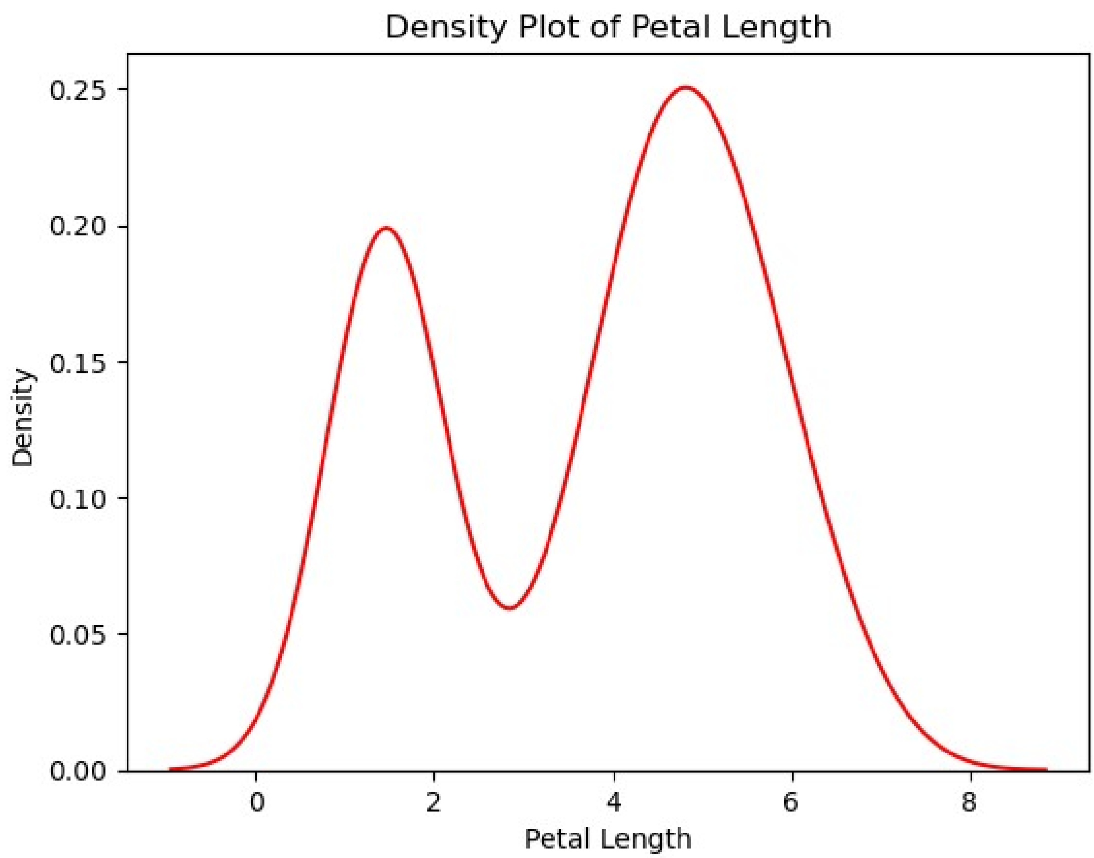
Use histograms and density plots to visualize petal length distribution.

```
df = sns.load_dataset('iris')

# Histogram of petal length

sns.histplot(df['petal_length'], bins=15, color='skyblue')
plt.title('Histogram of Petal Length')
plt.xlabel('Petal Length')
plt.ylabel('Frequency')
plt.show()

# Density plot of petal length
sns.kdeplot(df['petal_length'], color='red')
plt.title('Density Plot of Petal Length')
plt.xlabel('Petal Length')
plt.ylabel('Density')
plt.show()

/Users/mac/anaconda3/lib/python3.11/site-packages/seaborn/
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
```

```
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
/Users/mac/anaconda3/lib/python3.11/site-packages/seaborn/
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

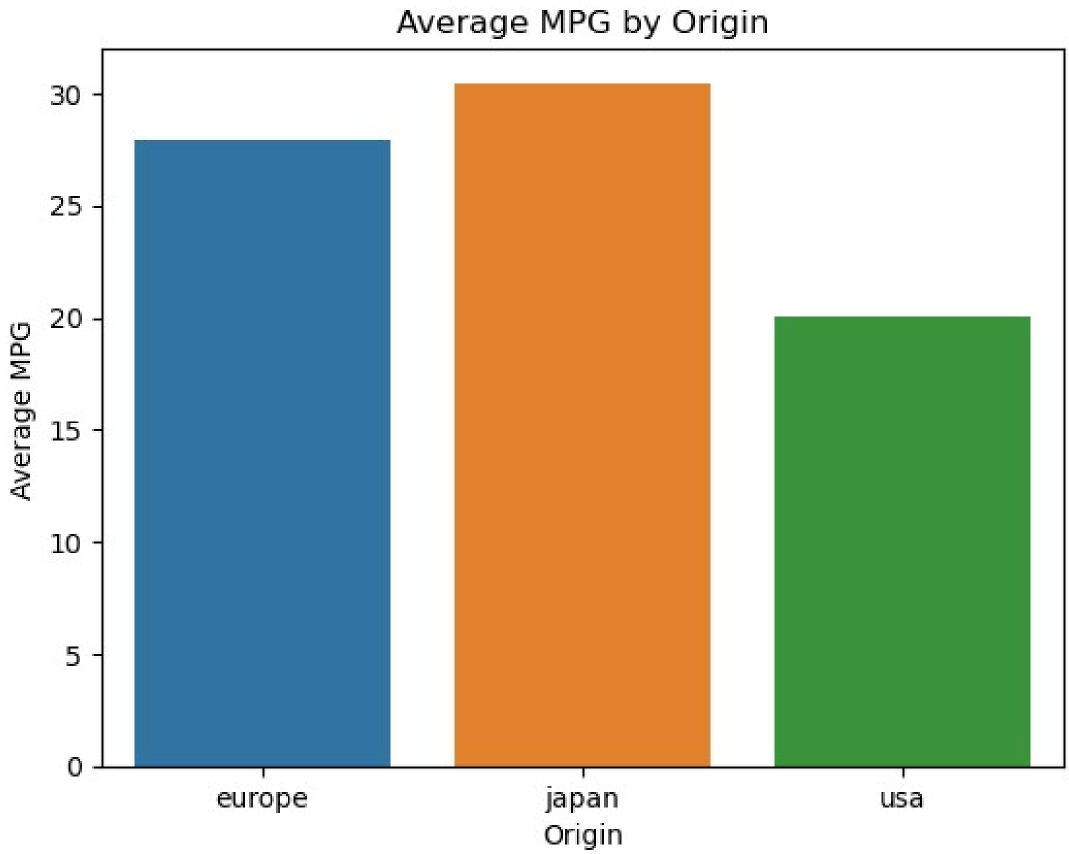


1. Ozone Levels Over Time: Inbuilt dataset: seaborn.load_dataset('mpg') in Python

A. find the number of unique car origins.

B. create a bar plot showing the average mpg for each origin.

```
df = sns.load_dataset('mpg')

unique_origins = df['origin'].unique()
print("Unique origins:", unique_origins)

avg_mpg = df.groupby('origin')['mpg'].mean().reset_index()

sns.barplot(data=avg_mpg, x='origin', y='mpg')
plt.title('Average MPG by Origin')
plt.xlabel('Origin')
plt.ylabel('Average MPG')
plt.show()

Unique origins: ['usa' 'japan' 'europe']
```



1. Inbuilt dataset: seaborn.load_dataset('diamonds') in Python

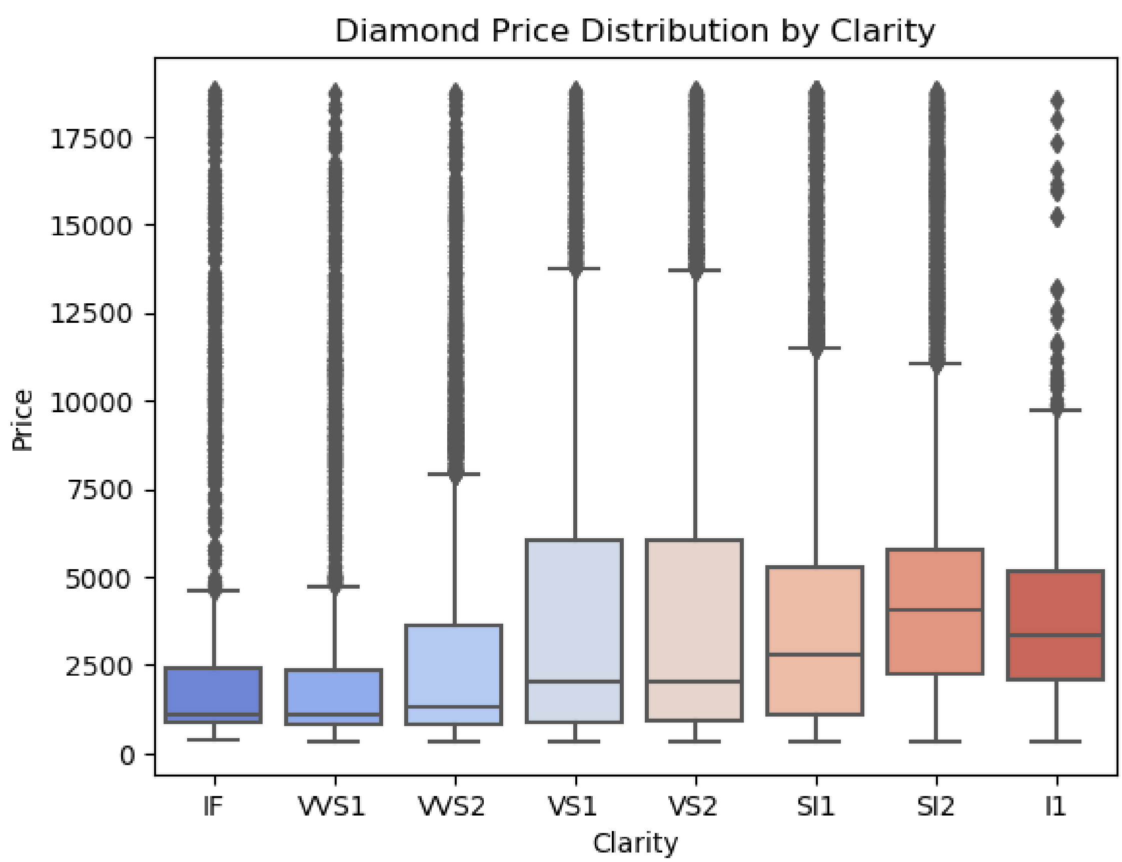
A. Analyze how the average price of diamonds varies with the cut quality (e.g., Fair, Good, Ideal, etc.).

B. Create a box plot to visualize the distribution of diamond prices for each clarity level.

```
df = sns.load_dataset('diamonds')

# A. Average price by cut quality
avg_price_by_cut = df.groupby('cut')['price'].mean().reset_index()
print(avg_price_by_cut)

# B. Box plot: Price distribution by clarity
sns.boxplot(data=df, x='clarity', y='price', palette='coolwarm')
plt.title('Diamond Price Distribution by Clarity')
plt.xlabel('Clarity')
plt.ylabel('Price')
plt.show()
```

	cut	price
0	Ideal	3457.541970
1	Premium	4584.257704

```
2  Very Good  3981.759891
3       Good  3928.864452
4       Fair  4358.757764

/var/folders/kx/41v1tt6j1yx79h_8wk_hkl8w0000gn/T/
ipykernel_4467/1318471352.py:4: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  avg_price_by_cut = df.groupby('cut')['price'].mean().reset_index()
/Users/mac/anaconda3/lib/python3.11/site-packages/seaborn/categorical.
py:641: FutureWarning: The default of observed=False is deprecated and
will be changed to True in a future version of pandas. Pass
observed=False to retain current behavior or observed=True to adopt
the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



1. A supermarket chain has collected sales data but has missing values and incorrect entries. The dataset is given below:

import pandas as pd

sales_data = pd.DataFrame({

```
"Transaction_ID": [101, 102, 103, 104],

"Date": pd.to_datetime(["2024-03-01", "2024-03-02", "2024-03-03",
"2024-03-04"]),

"Product": ["Apples", "Bread", "Milk", "Cheese"],

"Category": ["Fruits", "Bakery", "Dairy", "Dairy"],

"Quantity": [2, None, -1, 1],

"Price": [1.5, 2.0, 3.0, 5.0],

"Total_Sales": [3.0, None, -3.0, 5.0]
```

})

Write the code in Python for below problems

- Identify and handle missing values in Quantity and Total_Sales.
- Correct the incorrect Quantity values (negative values).
- Compute Total_Sales where missing.
- Summarize total sales per category.

```
sales_data = pd.DataFrame({
    "Transaction_ID": [101, 102, 103, 104],
    "Date": pd.to_datetime(["2024-03-01", "2024-03-02", "2024-03-03",
"2024-03-04"]),
    "Product": ["Apples", "Bread", "Milk", "Cheese"],
    "Category": ["Fruits", "Bakery", "Dairy", "Dairy"],
    "Quantity": [2, None, -1, 1],
    "Price": [1.5, 2.0, 3.0, 5.0],
    "Total_Sales": [3.0, None, -3.0, 5.0]
})

sales_data['Quantity'].fillna(0, inplace=True)
sales_data['Total_Sales'].fillna(0, inplace=True)

sales_data['Quantity'] = sales_data['Quantity'].apply(lambda x: abs(x)
if x < 0 else x)

sales_data['Total_Sales'] = sales_data['Quantity'] *
sales_data['Price']

category_sales = sales_data.groupby('Category')
['Total_Sales'].sum().reset_index()

print(sales_data)
print( category_sales)
```

```
   Transaction_ID       Date Product Category  Quantity  Price
Total_Sales
0             101 2024-03-01  Apples   Fruits       2.0    1.5
3.0
1             102 2024-03-02   Bread   Bakery       0.0    2.0
0.0
2             103 2024-03-03    Milk    Dairy       1.0    3.0
3.0
3             104 2024-03-04  Cheese    Dairy       1.0    5.0
5.0
  Category  Total_Sales
0   Bakery          0.0
1    Dairy          8.0
2   Fruits          3.0
```

1. Write the code in Python for below questions

import pandas as pd

df = pd.DataFrame({

```
'Order_ID': [101, 102, 103, 103, 104, 105, 105],

'Customer': ['Alice', 'Bob', None, None, 'Eve', 'Frank', 'Frank'],

'Product': ['Laptop', 'Phone', 'Tablet', 'Tablet', 'Monitor', None,
'Keyboard'],

'Price': [1000, 500, 300, 300, 200, 150, 100],

'Quantity': [2, None, 1, 1, 3, 2, 1]
```

})

** Identify and fill missing values:

- Fill missing Customer names with "Guest".
- Fill missing Quantity values with the median quantity.
- Fill missing Product values with "Unknown".

1. Remove duplicate Order_ID records, keeping the first occurrence
2. Add a new column called "Total Amount" = Price * Quantity

```
df = pd.DataFrame({
    'Order_ID': [101, 102, 103, 103, 104, 105, 105],
    'Customer': ['Alice', 'Bob', None, None, 'Eve', 'Frank', 'Frank'],
    'Product': ['Laptop', 'Phone', 'Tablet', 'Tablet', 'Monitor',
None, 'Keyboard'],
    'Price': [1000, 500, 300, 300, 200, 150, 100],
    'Quantity': [2, None, 1, 1, 3, 2, 1]
```

```
})

# Fill missing values
df['Customer'].fillna('Guest', inplace=True)
df['Quantity'].fillna(df['Quantity'].median(), inplace=True)
df['Product'].fillna('Unknown', inplace=True)

# Remove duplicate Order_ID records, keeping the first
df_unique = df.drop_duplicates(subset='Order_ID', keep='first');

# Add "Total Amount" column
df_unique['Total Amount'] = df_unique['Price'] *
df_unique['Quantity'];
print(df_unique);

   Order_ID Customer  Product  Price  Quantity  Total Amount
0       101    Alice   Laptop   1000       2.0        2000.0
1       102      Bob    Phone    500       1.5         750.0
2       103    Guest   Tablet    300       1.0         300.0
4       104      Eve  Monitor    200       3.0         600.0
5       105    Frank  Unknown    150       2.0         300.0

/var/folders/kx/41v1tt6j1yx79h_8wk_hkl8w0000gn/T/
ipykernel_4467/2724297360.py:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_unique['Total Amount'] = df_unique['Price'] *
df_unique['Quantity'];
```

1. Write the code in Python for below questions

df = pd.DataFrame({

```
'Transaction_ID': [1001, 1002, 1003, 1003, 1004, 1005],

'Customer': ['Alice', 'Bob', None, None, 'Eve', 'Frank'],

'Amount': [250, 400, None, 150, 700, 900],

'Discount': [10, 15, None, 5, None, 20]
```

})

1. Fill missing values:
- Customer → "Guest"
- Amount → mean of non-missing values

- Discount → replace None with 0

1. Remove duplicate Transaction_IDs.
2. Add a new column "Final Amount", calculated as Amount - (Amount * Discount / 100)

```
df = pd.DataFrame({
    'Transaction_ID': [1001, 1002, 1003, 1003, 1004, 1005],
    'Customer': ['Alice', 'Bob', None, None, 'Eve', 'Frank'],
    'Amount': [250, 400, None, 150, 700, 900],
    'Discount': [10, 15, None, 5, None, 20]
})

# 1. Fill missing Customer with "Guest"
df['Customer'].fillna('Guest', inplace=True)

# 2. Fill missing Amount with the mean of non-missing values
mean_amount = df['Amount'].mean()
df['Amount'].fillna(mean_amount, inplace=True)

# 3. Replace missing Discount values with 0
df['Discount'].fillna(0, inplace=True)

# 4. Remove duplicate Transaction_IDs, keeping the first
df = df.drop_duplicates(subset='Transaction_ID', keep='first')

# 5. Add "Final Amount" = Amount - (Amount * Discount / 100)
df['Final Amount'] = df['Amount'] - (df['Amount'] * df['Discount'] /
100)


print(df)
```

```
   Transaction_ID Customer  Amount  Discount  Final Amount
0            1001    Alice   250.0      10.0         225.0
1            1002      Bob   400.0      15.0         340.0
2            1003    Guest   480.0       0.0         480.0
4            1004      Eve   700.0       0.0         700.0
5            1005    Frank   900.0      20.0         720.0
```

1. Write the code in Python for below questions

df = pd.DataFrame({

```
'Product_ID': [101, 102, 103, 103, 104, 105],

'Product_Name': ['Laptop', None, 'Tablet', 'Tablet', 'Monitor',
'Keyboard'],

'Stock': [50, None, 30, 30, 20, None],

'Price': [1000, 500, 300, 300, 200, 150]
```

})

1. Fill missing values:
- Product_Name → "Unknown"
- Stock → median of non-missing stock values
1. Remove duplicate Product_IDs.
2. Add a column "Stock Value", calculated as Stock * Price.

```
df = pd.DataFrame({
    'Product_ID': [101, 102, 103, 103, 104, 105],
    'Product_Name': ['Laptop', None, 'Tablet', 'Tablet', 'Monitor',
'Keyboard'],
    'Stock': [50, None, 30, 30, 20, None],
    'Price': [1000, 500, 300, 300, 200, 150]
})

# 1. Fill missing Product_Name with "Unknown"
df['Product_Name'].fillna('Unknown', inplace=True)

# 2. Fill missing Stock values with the median of non-missing stock
values
median_stock = df['Stock'].median()
df['Stock'].fillna(median_stock, inplace=True)

# 3. Remove duplicate Product_IDs, keeping the first
df = df.drop_duplicates(subset='Product_ID', keep='first')

# 4. Add "Stock Value" column = Stock * Price
df['Stock Value'] = df['Stock'] * df['Price']

print(df)
```

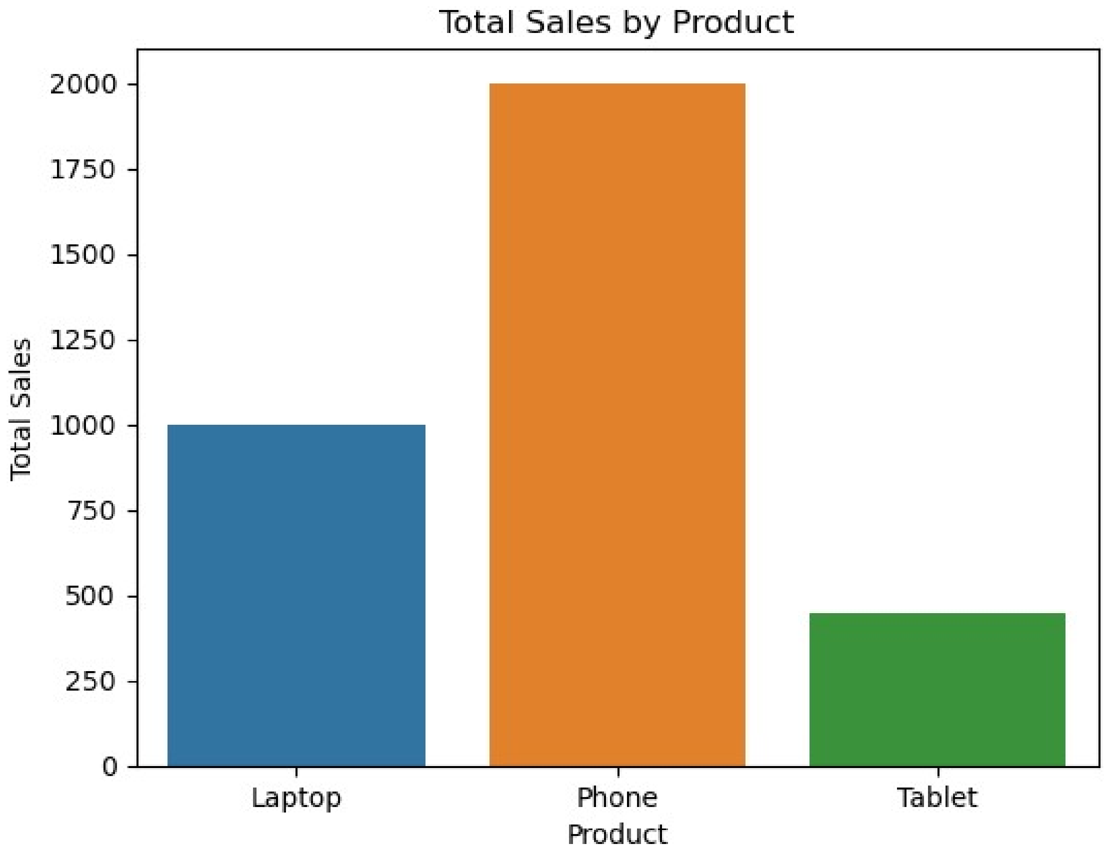
```
   Product_ID Product_Name  Stock  Price  Stock Value
0         101       Laptop   50.0   1000      50000.0
1         102      Unknown   30.0    500      15000.0
2         103       Tablet   30.0    300       9000.0
4         104      Monitor   20.0    200       4000.0
5         105     Keyboard   30.0    150       4500.0
```

Golden question

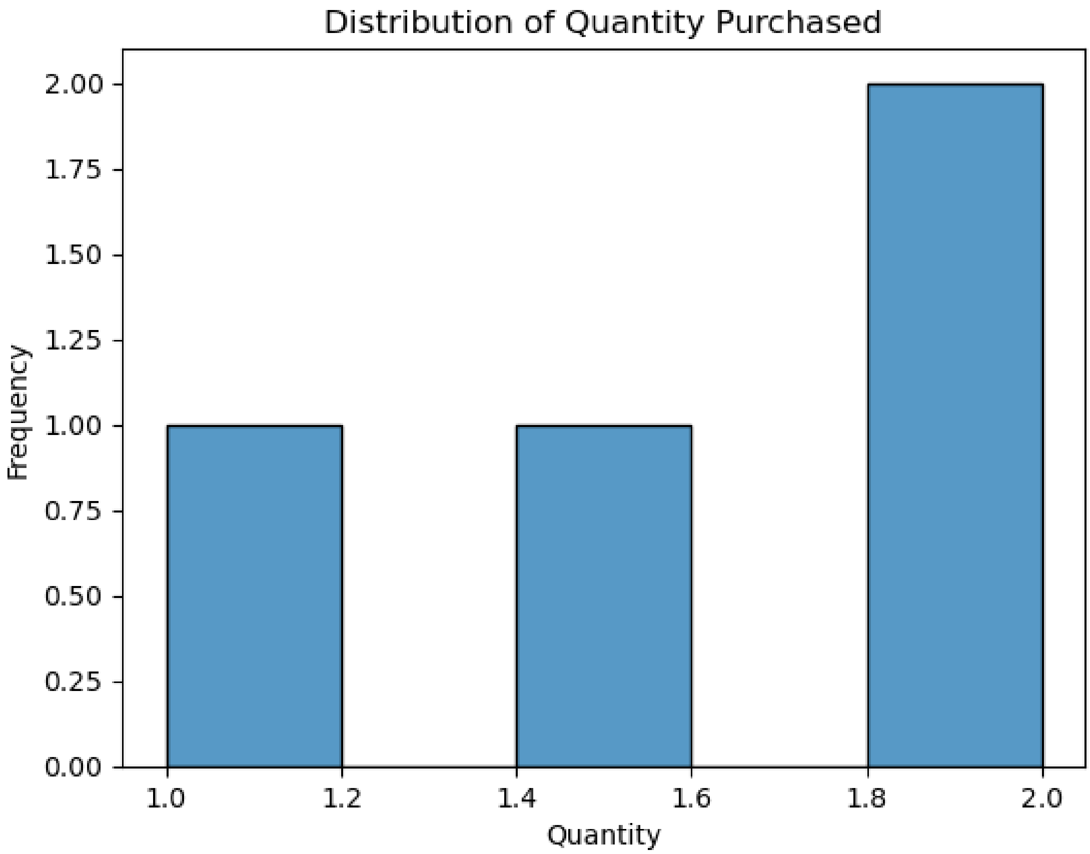
1. Create a Python dataframe with at least 4 columns and 5 rows (you can generate a dataset of your choice). Perform the following tasks in Python :
- Identify and handle missing values in the dataset.
- Remove duplicate rows if any.
- Add a new column based on existing data.
- Generate at least two visualizations using Matplotlib or Seaborn to analyze trends or distributions in the dataset.

```python
data = pd.DataFrame({
    'Customer': ['Alice', 'Bob', 'Charlie', 'Alice', np.nan],
    'Product': ['Laptop', 'Phone', 'Tablet', 'Laptop', 'Phone'],
    'Quantity': [1, 2, np.nan, 1, 2],
    'Price': [1000, 500, 300, 1000, 500]
})

print(data)

data['Customer'].fillna('Guest', inplace=True)
data['Quantity'].fillna(data['Quantity'].median(), inplace=True)

data.drop_duplicates(inplace=True)

data['Total'] = data['Quantity'] * data['Price']
print(data)

sns.barplot(data=data, x='Product', y='Total', estimator=sum)
plt.title("Total Sales by Product")
plt.ylabel("Total Sales")
plt.xlabel("Product")
plt.show()

sns.histplot(data['Quantity'], bins=5)
plt.title("Distribution of Quantity Purchased")
plt.xlabel("Quantity")
plt.ylabel("Frequency")
plt.show()
```

```
  Customer Product  Quantity  Price
0    Alice  Laptop       1.0   1000
1      Bob   Phone       2.0    500
2  Charlie  Tablet       NaN    300
3    Alice  Laptop       1.0   1000
4      NaN   Phone       2.0    500
  Customer Product  Quantity  Price   Total
0    Alice  Laptop       1.0   1000  1000.0
1      Bob   Phone       2.0    500  1000.0
2  Charlie  Tablet       1.5    300   450.0
4    Guest   Phone       2.0    500  1000.0
```




```
/Users/mac/anaconda3/lib/python3.11/site-packages/seaborn/
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Distribution of Quantity Purchased
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Frequency
1.0
1.2
1.4
1.6
1.8
2.0
Quantity

```r
# 1.  Air Quality Analysis: Inbuilt dataset: airquality in R
# A. Filter the records for the month of July.
# B. Group the data by Month and calculate the average Ozone.
# C. Use a pipe operator to fetch records where Ozone > 50.

library(dplyr)

data("airquality")

# A. Filter records for July (Month = 7)
july_data <- airquality %>%
  filter(Month == 7)

print(july_data)

# B. Group by Month and calculate average Ozone
ozone_avg <- airquality %>%
  group_by(Month) %>%
  summarise(Avg_Ozone = mean(Ozone), na.rm = TRUE)

print(ozone_avg)

# C. Use pipe to fetch records with Ozone > 50
high_ozone <- airquality %>%
  filter(Ozone > 50)

print(high_ozone)
```

```r
# 3. Car Performance Analysis: Inbuilt dataset: mtcars in R
# A. Compare the fuel efficiency (mpg) of automatic vs. manual transmission cars.
# B. Identify the relationship between horsepower (hp) and fuel consumption.

library(dplyr)
library(ggplot2)

# Add a readable label for transmission
mtcars$Transmission <- ifelse(mtcars$am == 0, "Automatic", "Manual")

# Calculate average mpg by transmission
avg_mpg <- mtcars %>%
  group_by(Transmission) %>%
  summarise(Average_MPG = mean(mpg))

print(avg_mpg)

# Bar plot for comparison
ggplot(avg_mpg, aes(x = Transmission, y = Average_MPG, fill = Transmission)) +
  geom_bar(stat = "identity") +
  labs(title = "Fuel Efficiency by Transmission Type",
       x = "Transmission Type",
```

```
       y = "Average MPG") +
  theme_minimal()


  # 5. Titanic Survival Analysis: Inbuilt Dataset: Titanic in R

# A. Compute the total number of passengers by gender and class.

# B. Calculate the percentage of passengers who survived, grouped by class.


library(titanic)
library(dplyr)

data <- titanic_train

# A. Total number of passengers by gender and class
passenger_counts <- data %>%
  group_by(Sex, Pclass) %>%
  summarise(Total_Passengers = n())

print(passenger_counts)

# B. Percentage of passengers who survived, grouped by class
survival_by_class <- data %>%
  group_by(Pclass) %>%
  summarise(Survival_Rate = mean(Survived) * 100)

print(survival_by_class)


# 5. Dataset: PlantGrowth (inbuilt in R)

# A. Compute the average weight of plants in each treatment group.
# B. Create a bar chart to visualize the average plant weights per group.


library(dplyr)
library(ggplot2)

data("PlantGrowth")

# A. Compute average weight by group
avg_weight <- PlantGrowth %>%
  group_by(group) %>%
  summarise(Avg_Weight = mean(weight))

print(avg_weight)

# B. Bar chart of average weight per group
ggplot(avg_weight, aes(x = group, y = Avg_Weight, fill = group)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Plant Weight by Group",
       x = "Treatment Group",
```

```r
      y = "Average Weight") +
  theme_minimal()

# 7. Iris Flower Classification: Inbuilt Dataset : iris in R

# A. Calculate the average petal length and petal width for each species.
# B. Create a scatter plot of Sepal.Length vs Sepal.Width colored by species


library(dplyr)
library(ggplot2)

data("iris")

# A. Average Petal.Length and Petal.Width by Species
avg_petal <- iris %>%
  group_by(Species) %>%
  summarise(
    Avg_Petal_Length = mean(Petal.Length),
    Avg_Petal_Width = mean(Petal.Width)
  )

print(avg_petal)

B. Scatter plot of Sepal.Length vs Sepal.Width by Species
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point(size = 3) +
  labs(title = "Sepal Dimensions by Species",
       x = "Sepal Length",
       y = "Sepal Width") +
  theme_minimal()


#  9. Distribution of Petal Length:  Inbuilt dataset: iris in R

# Use histograms and density plots to visualize petal length distribution.

library(ggplot2)

data("iris")

# Histogram of Petal Length
ggplot(iris, aes(x = Petal.Length)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black") +
  labs(title = "Histogram of Petal Length",
       x = "Petal Length",
       y = "Frequency") +
  theme_minimal()

# Density plot of Petal Length
ggplot(iris, aes(x = Petal.Length)) +
  geom_density(fill = "lightgreen", alpha = 0.6) +
  labs(title = "Density Plot of Petal Length",
       x = "Petal Length",
```

```r
       y = "Density") +
  theme_minimal()


# 11. Dataset: mtcars (inbuilt in R)
# A.  Filter and show details of cars with horsepower (hp) greater than 150.
# B. Create a scatter plot showing the relationship between horsepower (hp) and fuel efficienc

library(ggplot2)
library(dplyr)

# Load dataset
data("mtcars")


high_hp_cars <- mtcars %>% filter(hp > 150)

print(high_hp_cars)

# Scatter plot
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = "steelblue", size = 3) +
  labs(title = "Horsepower vs. Fuel Efficiency",
       x = "Horsepower (hp)",
       y = "Miles per Gallon (mpg)") +
  theme_minimal()




# 13. CO2 Emissions : Inbuilt dataset: CO2 in R

# A. Compare CO2 uptake between different treatment groups.
# B. Analyze which factors significantly affect CO2 levels.

library(dplyr)
library(ggplot2)

data("CO2")

# A. Average CO2 uptake by Treatment group
avg_uptake <- CO2 %>%
  group_by(Treatment) %>%
  summarise(Avg_Uptake = mean(uptake))

print(avg_uptake)

# B. Scatter plot: CO2 uptake vs. concentration, colored by Plant Type
ggplot(CO2, aes(x = conc, y = uptake, color = Type)) +
  geom_point(size = 3) +
  labs(title = "CO2 Uptake by Concentration and Plant Type",
       x = "CO2 Concentration (ppm)",
       y = "CO2 Uptake",
       color = "Plant Type") +
  theme_minimal()
```

```
# 15. A supermarket chain has collected sales data but has missing values and incorrect entrie

# sales_data <- data.frame(

#   Transaction_ID = c(101, 102, 103, 104),

#   Date = as.Date(c("2024-03-01", "2024-03-02", "2024-03-03", "2024-03-04")),

#   Product = c("Apples", "Bread", "Milk", "Cheese"),

#   Category = c("Fruits", "Bakery", "Dairy", "Dairy"),

#   Quantity = c(2, NA, -1, 1),

#   Price = c(1.5, 2.0, 3.0, 5.0),

#   Total_Sales = c(3.0, NA, -3.0, 5.0)

# )

# Write the code in R for below problems:

# Identify and handle missing values in Quantity and Total_Sales.
# Correct the incorrect Quantity values (negative values).
# Compute Total_Sales where missing.
# Summarize total sales per category.

sales_data <- data.frame(
  Transaction_ID = c(101, 102, 103, 104),
  Date = as.Date(c("2024-03-01", "2024-03-02", "2024-03-03", "2024-03-04")),
  Product = c("Apples", "Bread", "Milk", "Cheese"),
  Category = c("Fruits", "Bakery", "Dairy", "Dairy"),
  Quantity = c(2, NA, -1, 1),
  Price = c(1.5, 2.0, 3.0, 5.0),
  Total_Sales = c(3.0, NA, -3.0, 5.0)
)

# 1. Handle missing values in Quantity and Total_Sales
# Replace missing Quantity with the median
sales_data$Quantity[is.na(sales_data$Quantity)] <- median(sales_data$Quantity, na.rm = TRUE)

# Replace missing Total_Sales with 0
sales_data$Total_Sales[is.na(sales_data$Total_Sales)] <- 0

# 2. Correct negative Quantity values
sales_data$Quantity[sales_data$Quantity < 0] <- abs(sales_data$Quantity[sales_data$Quantity <

# 3. Recompute Total_Sales where it's 0 or wrong
sales_data$Total_Sales <- sales_data$Quantity * sales_data$Price

# 4. Summarize total sales per category
library(dplyr)
```

```
category_summary <- sales_data %>%
  group_by(Category) %>%
  summarise(Total_Sales_Sum = sum(Total_Sales))

print(category_summary)



# Golden Question

# 2. Using any built-in dataset in R, perform the following tasks:

# Data Manipulation using dplyr:

# Select relevant columns for analysis.
# Filter the dataset based on a meaningful condition.
# Create a new derived column using existing data.
# Group the data and compute summary statistics.
# Arrange the dataset meaningfully (e.g., in ascending or descending order).
# Data Visualization using ggplot2:

# Create at least two visualizations to explore trends or distributions in the dataset
# Use appropriate aesthetics such as color, size, and facets.
# Add clear axis labels, a title, and a legend where necessary.



library(dplyr)
library(ggplot2)


head(mtcars)

#  Data Manipulation
manipulated_data <- mtcars %>%
  select(mpg, cyl, hp, gear) %>%
  filter(hp > 100) %>%
  mutate(Efficiency = mpg / cyl) %>%
  group_by(gear) %>%
  summarise(
    Avg_MPG = mean(mpg),
    Avg_HP = mean(hp),
    Count = n()
  ) %>%
  arrange(desc(Avg_MPG))

print(manipulated_data)

#  Scatter Plot - HP vs MPG
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(size = 3) +
  labs(
    title = "Horsepower vs MPG",
    x = "Horsepower (hp)",
```

```
    y = "Miles Per Gallon (mpg)",
    color = "Cylinders"
  ) +
  theme_minimal()

#  Boxplot - MPG by Gear
ggplot(mtcars, aes(x = factor(gear), y = mpg)) +
  geom_boxplot() +
  labs(
    title = "Distribution of MPG by Number of Gears",
    x = "Number of Gears",
    y = "Miles Per Gallon (mpg)"
  ) +
  theme_minimal()
```