R Question

```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)
```

# 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.

```r
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.

```r
library(dplyr)
library(ggplot2)

data("PlantGrowth")
```

# A. Compute average weight by group

```r
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",
       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

```r
library(dplyr)
library(ggplot2)
```

```r
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.

```r
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",
       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 efficiency (mpg).

```
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 entries. The dataset is given below:

```
# 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 < 0])
```

# 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:

```r
# 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()
```

Python

Q.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 average mpg per model year

Ans:-

```
import seaborn as sns
import pandas as pd

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

```
# Preview the dataset
df.head()
```

A:-

```
df.isnull().sum()
# Impute numerical column
df['horsepower'].fillna(df['horsepower'].median(), inplace=True)

# Example: if origin had missing values
if df['origin'].isnull().sum() > 0:
    df['origin'].fillna(df['origin'].mode()[0], inplace=True)

# Confirm no missing values
df.isnull().sum()
```

B:-

```
# Group by model year and compute average mpg
avg_mpg_per_year = df.groupby('model_year')['mpg'].mean().reset_index()

# Display result
print(avg_mpg_per_year)

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(avg_mpg_per_year['model_year'], avg_mpg_per_year['mpg'], marker='o',
color='teal')
plt.title('Average MPG per Model Year')
plt.xlabel('Model Year')
plt.ylabel('Average MPG')

plt.grid(True)

plt.show()
```

Q.2Car 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.

Ans:-

1. Load the dataset and display the first 5 rows

python

CopyEdit

```
import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt


# Load the dataset

df = sns.load_dataset('mpg')


# Display first 5 rows

df.head()
```

✅ 2. How many rows and columns does the dataset have?

python

CopyEdit

df.shape # Returns (rows, columns)

✅ **3. What are the names of all the columns in the dataset?**

python

CopyEdit

```
df.columns.tolist()
```

Expected columns include:

plaintext

CopyEdit

```
['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
 'acceleration', 'model_year', 'origin', 'name']
```

✅ **4. Find the average miles per gallon (mpg) for each number of cylinders**

python

CopyEdit

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


# Display the result
print(avg_mpg_by_cyl)
```

✅ **5. Create a scatter plot to show the relationship between horsepower and mpg**

Before plotting, handle any missing values in horsepower or mpg.

python

CopyEdit

```
# Drop rows with missing values in relevant columns
df_clean = df.dropna(subset=['horsepower', 'mpg'])


# Scatter plot
```

```
plt.figure(figsize=(8, 6))

sns.scatterplot(data=df_clean, x='horsepower', y='mpg', hue='cylinders',
palette='viridis')

plt.title('Horsepower vs. MPG')

plt.xlabel('Horsepower')

plt.ylabel('Miles per Gallon (MPG)')

plt.grid(True)

plt.show()
```

Q.3.:- 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.**

Ans:-

```
import seaborn as sns

import pandas as pd


# Load the Titanic dataset

df = sns.load_dataset('titanic')


# Preview the first few rows

df.head()
```

A:-

```
# Group by 'sex' and 'class' and calculate the mean of 'survived' column

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

```
# Rename the column for clarity
survival_rate.rename(columns={'survived': 'survival_rate'}, inplace=True)

# Display the result
print(survival_rate)
```

B:-

```
# Apply all conditions using boolean filtering
filtered_passengers = df[
    (df['class'] == 'First') &
    (df['sex'] == 'female') &
    (df['fare'] > 50)
]

# Display the filtered result
filtered_passengers
```

Q.4:- Iris Flower Classification: Inbuilt Dataset : iris in Python

� Display basic information and summary statistics of the dataset.

� Check for missing values in each column.

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

Ans:-

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

# Load the Iris dataset
df = sns.load_dataset('iris')
```

# Preview the dataset

df.head()

1. **2. Display basic information and summary statistics**

python

CopyEdit

# Basic info (data types, non-null counts, etc.)

df.info()

# Summary statistics

df.describe()

✅ 3. Check for missing values in each column

python

CopyEdit

# Check for missing values

df.isnull().sum()

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

python

CopyEdit

plt.figure(figsize=(8, 6))

sns.scatterplot(data=df, x='sepal_length', y='sepal_width', hue='species', palette='Set1')

plt.title('Sepal Length vs Sepal Width by Species')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Sepal Width (cm)')

```
plt.grid(True)
plt.show()
```

Q.5:- Distribution of Petal Length: Inbuilt dataset: iris in Python

Use histograms and density plots to visualize petal length distribution.

Ans:-

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


# Load the iris dataset
df = sns.load_dataset('iris')
plt.figure(figsize=(8, 5))
sns.histplot(data=df, x='petal_length', bins=20, color='teal')
plt.title('Histogram of Petal Length')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()


plt.figure(figsize=(8, 5))
sns.kdeplot(data=df, x='petal_length', fill=True, color='darkorange')
plt.title('Density Plot of Petal Length')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Density')
plt.grid(True)
plt.show()
plt.figure(figsize=(8, 5))
sns.kdeplot(data=df, x='petal_length', hue='species', fill=True)
```

```
plt.title('Petal Length Density Plot by Species')

plt.xlabel('Petal Length (cm)')

plt.ylabel('Density')

plt.grid(True)

plt.show()
```

Q.6:- 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.

Ans:-

```
import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt


# Load the dataset

df = sns.load_dataset('mpg')


# Preview the data

df.head()

unique_origins = df['origin'].nunique()

print("Number of unique car origins:", unique_origins)

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

# Group by origin and calculate average mpg

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


# Plot

plt.figure(figsize=(8,5))
```

```
sns.barplot(data=avg_mpg_by_origin, x='origin', y='mpg', palette='pastel')
plt.title('Average MPG by Car Origin')
plt.xlabel('Origin')
plt.ylabel('Average Miles Per Gallon (MPG)')
plt.grid(axis='y')
plt.show()
```

Q.7. 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.

Ans:-

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


# Load the dataset
df = sns.load_dataset('diamonds')


# Preview the data
df.head()
```

A:-

```
# Group by 'cut' and calculate average price
avg_price_by_cut = df.groupby('cut')['price'].mean().reset_index()


# Display the result
print(avg_price_by_cut)
plt.figure(figsize=(8, 5))
sns.barplot(data=avg_price_by_cut, x='cut', y='price', palette='coolwarm')
```

```
plt.title('Average Diamond Price by Cut Quality')

plt.xlabel('Cut Quality')

plt.ylabel('Average Price (USD)')

plt.grid(axis='y')

plt.show()
```

B:-

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

sns.boxplot(data=df, x='clarity', y='price', palette='Set3')

plt.title('Diamond Price Distribution by Clarity')

plt.xlabel('Clarity Level')

plt.ylabel('Price (USD)')

plt.xticks(rotation=45)

plt.grid(True)

plt.show()
```

Q.8:- . 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.

Ans:-

```
import pandas as pd


# Create the DataFrame

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]

})
```

Step 2: Identify and handle missing values in Quantity and Total_Sales

```
# Show missing values

print("Missing values:\n", sales_data[['Quantity', 'Total_Sales']].isnull().sum())


# Fill missing Quantity with 0 (or any other logic like average if needed)

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


# Fill missing Total_Sales temporarily with NaN; we'll recalculate it

sales_data['Total_Sales'] = sales_data['Total_Sales'].fillna(0)
```

✅ **Step 3: Correct negative Quantity values**

# Replace negative Quantity values with their absolute value

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

✅ Step 4: Recompute Total_Sales where it's zero (previously missing or incorrect)

# Recalculate Total_Sales where it was originally missing or negative

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

✅ Step 5: Summarize total sales per category

# Group by Category and sum Total_Sales

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

# Display the summary

print("Total Sales per Category:\n", category_sales_summary)

✅ Final cleaned dataset preview

print("Cleaned Sales Data:\n", sales_data)

Q.8:- 17. 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".

 Remove duplicate **Order_ID** records, keeping the first occurrence.

 Add a new column **"Total Amount"**, calculated as Price * Quantity.

Ans:-

```
import pandas as pd


# Create the DataFrame
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]
})
```

1.-

```
df['Customer'].fillna('Guest', inplace=True)
median_quantity = df['Quantity'].median()
df['Quantity'].fillna(median_quantity, inplace=True)
df['Product'].fillna('Unknown', inplace=True)
```

2.-

```
df = df.drop_duplicates(subset='Order_ID', keep='first')
```

3.-

```
df['Total Amount'] = df['Price'] * df['Quantity']
```

4.=

```
print(df)
```

Q.9:- 18. 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]
})
```

 Fill missing values:

- **Customer** → "Guest"
- **Amount** → mean of non-missing values
- **Discount** → replace None with 0

 Remove duplicate **Transaction_IDs**.

 Add a new column **"Final Amount"**, calculated as Amount - (Amount * Discount / 100)

Ans:-

```
import pandas as pd

# Create the DataFrame
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]
})

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

# 2. Fill missing Amount values with the mean of non-missing values

df['Amount'].fillna(df['Amount'].mean(), inplace=True)

# 3. Fill missing Discount values with 0

df['Discount'].fillna(0, inplace=True)

# Remove duplicate Transaction_IDs, keeping the first occurrence

df = df.drop_duplicates(subset='Transaction_ID', keep='first')

# Add a new column "Final Amount" calculated as Amount - (Amount * Discount / 100)

df['Final Amount'] = df['Amount'] - (df['Amount'] * df['Discount'] / 100)

# Show the cleaned DataFrame

print(df)

Q:-10 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]
})
```

 Fill missing values:

- **Product_Name** → "Unknown"
- **Stock** → median of non-missing stock values

 Remove duplicate **Product_IDs**.

□ Add a column **"Stock Value"**, calculated as Stock * Price.

Ans:-

```
import pandas as pd

# Create the DataFrame
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 the non-missing stock values
df['Stock'].fillna(df['Stock'].median(), inplace=True)

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

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

# Display the cleaned DataFrame
print(df)
```