

# PYTHON ANSWERS

## 2. 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)
```

## 4. 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()
```

## 6. . 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']]);
```

## 8. 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')  
plt.legend(title='Species')  
plt.show()
```

## 10. 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()
```

## 12. 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()
```

## 14. 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()
```

## 16. 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)

```

## 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".

2. Remove duplicate Order\_ID records, keeping the first occurrence
3. 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);
```

## 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]
})
```

1. Fill missing values:

\* Customer → "Guest"

\* Amount → mean of non-missing values

\* Discount → replace None with 0

2. Remove duplicate Transaction\_IDs.

3. Add a new column "Final Amount", calculated as  $\text{Amount} - (\text{Amount} * \text{Discount} / 100)$

```
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)
```

## 19. 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

2. Remove duplicate Product\_IDs.

3. 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)
```

### ### 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.

```
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()
```