mtcars.csv

Read the dataset

```
import pandas as pd

# Assuming the dataset 'mtcars.csv' is in the current directory
mtcars = pd.read_csv('mtcars.csv');
```

Find the head of the dataset

```
# To examine the first few rows of the dataset:
print(mtcars.head())
```

Find the Datatype of Dataset (each column)

```
# To determine the datatype of each column:
print(mtcars.dtypes)
```

Plot a histogram of the 'mpg' variable

```
# To visualize the frequency distribution of the 'mpg' (Miles per gallon) variable using a
histogram:
import matplotlib.pyplot as plt
import pandas as pd

mtcars = pd.read_csv('mtcars.csv')
plt.hist(mtcars['mpg'], bins=10, edgecolor='black')
plt.xlabel('Miles per gallon (mpg)')
plt.ylabel('Frequency')
plt.title('Histogram of Miles per gallon (mpg)')
plt.show()
```

Find the highest frequency of interval

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

# Generate some example data
data = np.random.normal(loc=0, scale=1, size=1000)

# Create a histogram
plt.hist(data, bins=10, edgecolor='black')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Histogram of Data')
plt.grid(True)
```

```python
# Display the plot
plt.show()

# Get the histogram data
counts, bins, _ = plt.hist(data, bins=10, edgecolor='black')

# Find the bin with the maximum frequency
max_frequency = np.max(counts)
max_index = np.argmax(counts)
max_interval = (bins[max_index], bins[max_index + 1])
print(f"The interval with the highest frequency is {max_interval} with
frequency {max_frequency}.")
```

Inference from scatter plot of 'mpg' vs 'wt'

```python
plt.scatter(mtcars['wt'], mtcars['mpg'], color='blue')
plt.xlabel('Weight of car (wt)')
plt.ylabel('Miles per gallon (mpg)')
plt.title('Scatter plot of Miles per gallon vs Weight of car')
plt.show()
```

Churn_DataDescription.csv

Number of duplicate records based on CustomerID column

```python
import pandas as pd

# Load the dataset
churn = pd.read_csv('Churn_DataDescription.csv')

# Find duplicate records based on CustomerID column
duplicate_records = churn.duplicated(subset=['CustomerID']).sum()

print(f"Number of duplicate records based on CustomerID: {duplicate_records}")
```

Total number of missing values for the variable TotalCharges

```python
missing_total_charges = churn['TotalCharges'].isnull().sum()
print(f"Total number of missing values for TotalCharges {missing_total_charges}")
```

Average monthly charge paid by a customer

```python
**# Assuming the column representing monthly charges is named MonthlyCharges:**
**# Calculate average monthly charge**
average_monthly_charge = churn['MonthlyCharges'].mean()
print(f"Average monthly charge paid by a customer: {average_monthly_charge}")
```

Number of records in the Dependents column that have "1@#"

```
**# Count records in Dependents column that have "1@#"**
dependents_count = (churn['Dependents'] == '1@#').sum()
print(f"Number of records in Dependents column with '1@#': {dependents_count}")
```

Find the data type of the variable tenure from the churn dataframe

```
# Check the data type of the variable 'tenure'
tenure_dtype = churn['tenure'].dtype print(f"Data type of the variable 'tenure':
{tenure_dtype}")
```

Diamond_DataDescription.csv

Load data set

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

# Load the dataset
diamonds = pd.read_csv('Diamonds_DataDescription.csv')
```

Plot a boxplot for “price” vs “cut” from the dataset “diamond.csv”. Which of the categories under “cut” have the highest median price?

```
# Boxplot for price vs cut
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.boxplot(x='cut', y='price', data=diamonds, order=['Fair', 'Good', 'Very Good',
'Premium', 'Ideal'])

plt.title('Boxplot of Price vs Cut')
plt.xlabel('Cut')
plt.ylabel('Price')
plt.show()
```

Create a frequency table (one-way table) for the variable “cut” from the dataset “diamond.csv”. What is the frequency for the cut type “Ideal”?

```
# Frequency table for cut
cut_frequency = diamonds['cut'].value_counts()
print(cut_frequency)

# Frequency for cut type 'Ideal'
frequency_ideal = cut_frequency['Ideal']
print(frequency_ideal)
```

Show the subplot of diamond depth distribution

```
plt.tight_layout()
plt.show()

# Subplot of diamond depth distribution
plt.subplot(1, 2, 2)
plt.hist(diamonds['depth'], bins=30, color='lightgreen', edgecolor='black')
plt.title('Diamond Depth Distribution')
plt.xlabel('Depth')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

Build the Model using linear regression and find the accuracy

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Extract features and target
X = diamonds[['carat', 'depth']]
y = diamonds['price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize Linear Regression model
model = LinearRegression()

# Fit the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Calculate R-squared
accuracy = r2_score(y_test, y_pred)
print(f'Accuracy (R-squared): {accuracy}')plt.tight_layout()
plt.show()
```

Charm case.csv

Which of the variables have missing values

```
import pandas as pd
# Load the dataset

df = pd.read_csv("People Charm case.csv")

# Check for missing values
missing_values = df.isnull().sum()
variables_with_missing = missing_values[missing_values > 0].index.tolist()
```

```
print("Variables with missing values:")
print(variables_with_missing)
```

What is the third quartile value for the variable “lastEvaluvation”?

```
# Assuming 'lastEvaluvation' is a typo and should be 'lastEvaluation'
third_quartile = df['lastEvaluation'].quantile(0.75)
print("Third quartile value for lastEvaluation:", third_quartile)
```

Construct a Crosstable for the variables ‘dept’ and “salary” and find out which department has highest frequency value in the category low salary.

```
# Crosstab and finding the department with highest frequency of low salary
cross_table = pd.crosstab(df['dept'], df['salary'])

# Find department with highest frequency of low salary
dept_with_highest_low_salary = cross_table['low'].idxmax()
highest_freq_low_salary = cross_table.loc[dept_with_highest_low_salary, 'low']
print("Department with highest frequency of low salary:", dept_with_highest_low_salary)
print("Frequency of low salary:", highest_freq_low_salary)
```

Generate a boxplot for the variable “numberOfProjects” and get the median value for the number of projects where the employees have worked on.

```
import matplotlib.pyplot as plt

# Boxplot for numberOfProjects
plt.figure(figsize=(8, 6))
plt.boxplot(df['numberOfProjects'])
plt.title('Boxplot of numberOfProjects')
plt.ylabel('Number of Projects')
plt.show()

# Median value
median_projects = df['numberOfProjects'].median()
print("Median number of projects:", median_projects)
```

Plot a histogram using the variable “avgMonthlyHours” and find the range in which the number of employees worked for 150 hours per month?

```
# Histogram for avgMonthlyHours
plt.figure(figsize=(8, 6))
plt.hist(df['avgMonthlyHours'], bins=20, edgecolor='black')
plt.title('Histogram of avgMonthlyHours')
plt.xlabel('Average Monthly Hours')
plt.ylabel('Frequency')
plt.show()

# Range for 150 hours
```

```
num_employees_150_hours = ((df['avgMonthlyHours'] >= 150) & (df['avgMonthlyHours'] <
160)).sum()
print("Number of employees worked 150 hours per month:", num_employees_150_hours)
```

Generate a boxplot for the variables “lastEvaluation” and “numberOfProjects”

```
import seaborn as sns

# Boxplot for lastEvaluation and numberOfProjects
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['lastEvaluation', 'numberOfProjects']])
plt.title('Boxplot of lastEvaluation and numberOfProjects')
plt.ylabel('Value')
plt.show()
```