# Import the required visualization libraries

import seaborn as sns

import matplotlib.pyplot as plt

# Create a histogram of 2021 unemployment; show a full percent in each bin

sns.histplot(data=unemployment,x="2021",binwidth=1)

plt.show()

# Update the data type of the 2019 column to a float

unemployment["2019"] = unemployment["2019"].astype(float)

# Print the dtypes to check your work

print(unemployment.dtypes)

# Define a Series describing whether each continent is outside of Oceania

not\_oceania = ~unemployment["continent"].isin(["Oceania"])

# Print unemployment without records related to countries in Oceania

print(unemployment[not\_oceania])

# Print the minimum and maximum unemployment rates during 2021

print(unemployment["2021"].min(), unemployment["2021"].max())

# Create a boxplot of 2021 unemployment rates, broken down by continent

sns.boxplot(data=unemployment,x="2021",y="continent")

plt.show()

continent\_summary = unemployment.groupby("continent").agg(

    # Create the mean\_rate\_2021 column

    mean\_rate\_2021=("2021","mean"),

    # Create the std\_rate\_2021 column

    std\_rate\_2021=("2021","std")

)

print(continent\_summary)

# Create a bar plot of continents and their average unemployment

sns.barplot(data=unemployment, x="continent",y="2021")

plt.show()

# Count the number of missing values in each column

print(planes.isna().sum())

# Find the five percent threshold

threshold = len(planes) \* 0.05

# Create a filter

cols\_to\_drop = planes.columns[planes.isna().sum() <= threshold]

# Drop missing values for columns below the threshold

planes.dropna(subset=cols\_to\_drop, inplace=True)

print(planes.isna().sum())

# Calculate median plane ticket prices by Airline

airline\_prices = planes.groupby("Airline")["Price"].median()

print(airline\_prices)

# Convert to a dictionary

prices\_dict = airline\_prices.to\_dict()

# Map the dictionary to missing values of Price by Airline

planes["Price"] = planes["Price"].fillna(planes["Airline"].map(prices\_dict))

# Check for missing values

print(planes.isna().sum())

# Filter the DataFrame for object columns

non\_numeric = planes.select\_dtypes('object')

# Loop through columns

for col in non\_numeric.columns:

# Mean Price by Destination

planes["price\_destination\_mean"] = planes.groupby("Destination")["Price"].transform(lambda x: x.mean())

print(planes[["Destination","price\_destination\_mean"]].value\_counts())

  # Print the number of unique values

  print(f"Number of unique values in {col} column: ", non\_numeric[col].nunique())

# Create conditions for values in flight\_categories to be created

conditions = [

    (planes["Duration"].str.contains(short\_flights)),

    (planes["Duration"].str.contains(medium\_flights)),

    (planes["Duration"].str.contains(long\_flights))

]

# Apply the conditions list to the flight\_categories

planes["Duration\_Category"] = np.select(conditions,

                                        flight\_categories,

                                        default="Extreme duration")

# Plot the counts of each category

sns.countplot(data=planes, x="Duration\_Category")

plt.show()

# Preview the column

print(planes["Duration"].head())

# Remove the string character

planes["Duration"] = planes["Duration"].str.replace("h", "")

# Convert to float data type

planes["Duration"] = planes["Duration"].astype(float)

# Plot a histogram

sns.histplot(data=planes, x="Duration")

plt.show()

# Find the 75th and 25th percentiles

price\_seventy\_fifth = planes["Price"].quantile(0.75)

price\_twenty\_fifth = planes["Price"].quantile(0.25)

# Calculate iqr

prices\_iqr = price\_seventy\_fifth - price\_twenty\_fifth

# Calculate the thresholds

upper = price\_seventy\_fifth + (1.5 \* prices\_iqr)

lower = price\_twenty\_fifth - (1.5 \* prices\_iqr)

# Subset the data

planes = planes[(planes["Price"] > lower) & (planes["Price"] < upper)]

print(planes["Price"].describe())

# Define the marriage\_year column

divorce["marriage\_year"] = divorce["marriage\_date"].dt.year

# Create a line plot showing the average number of kids by year

sns.lineplot(data=divorce, x="marriage\_year", y="num\_kids")

plt.show()

# Create the scatterplot

sns.scatterplot(data=divorce,x="marriage\_duration",y="num\_kids")

plt.show()

# Create a pairplot for income\_woman and marriage\_duration

sns.pairplot(data=divorce, vars=["income\_woman" , "marriage\_duration"])

plt.show()

# Create the scatter plot

sns.scatterplot(data=divorce,x="woman\_age\_marriage",y="income\_woman",hue="education\_woman")

plt.show()

# Update the KDE plot to show a cumulative distribution function

sns.kdeplot(data=divorce, x="marriage\_duration", hue="num\_kids", cut=0, cumulative=True)

plt.show()

# Cross-tabulate Job\_Category and Company\_Size

print(pd.crosstab(salaries["Job\_Category"], salaries["Company\_Size"],

            values=salaries["Salary\_USD"], aggfunc="mean"))

# Get the month of the response

salaries["month"] = salaries["date\_of\_response"].dt.month

# Extract the weekday of the response

salaries["weekday"] = salaries["date\_of\_response"].dt.weekday

# Create a heatmap

sns.heatmap(salaries.corr(), annot=True)

plt.show()

# Find the 25th percentile

twenty\_fifth = salaries["Salary\_USD"].quantile(0.25)

# Save the median

salaries\_median = salaries["Salary\_USD"].median()

# Gather the 75th percentile

seventy\_fifth = salaries["Salary\_USD"].quantile(0.75)

print(twenty\_fifth, salaries\_median, seventy\_fifth)

# Create salary labels

salary\_labels = ["entry", "mid", "senior", "exec"]

# Create the salary ranges list

salary\_ranges = [0, twenty\_fifth, salaries\_median, seventy\_fifth, salaries["Salary\_USD"].max()]

# Create salary labels

salary\_labels = ["entry", "mid", "senior", "exec"]

# Create the salary ranges list

salary\_ranges = [0, twenty\_fifth, salaries\_median, seventy\_fifth, salaries["Salary\_USD"].max()]

# Create salary\_level

salaries["salary\_level"] = pd.cut(salaries["Salary\_USD"],

                                  bins=salary\_ranges,

                                  labels=salary\_labels)

# Plot the count of salary levels at companies of different sizes

sns.countplot(data=salaries, x="Company\_Size", hue="salary\_level")

plt.show()

# Filter for employees in the US or GB

usa\_and\_gb = salaries[salaries["Employee\_Location"].isin(["US", "GB"])]

# Create a barplot of salaries by location

sns.barplot(data=usa\_and\_gb, x="Employee\_Location", y="Salary\_USD")

plt.show()