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

divorce = pd.read_csv('divorce.csv')
divorce.dtypes

# divorce["marriage_date"] = pd.to_datetime(divorce[["month", "day",
"year"]])
# divorce.head(2)

# divorce["marriage_date"] = pd.to_datetime(divorce["marriage_date"])
# Same as parse_dates
# divorce.dtypes

# divorce["marriage_month"] = divorce["marriage_date"].dt.month
# divorce.head()

# sns.lineplot(data=divorce, x='marriage_month',
y='marriage_duration')
# plt.show()

divorce = pd.read_csv('divorce.csv', parse_dates=["divorce_date",
"dob_man", "dob_woman"])
divorce.dtypes

divorce["marriage_date"] = pd.to_datetime(divorce["marriage_date"])

divorce["marriage_year"] = divorce["marriage_date"].dt.year

sns.lineplot(data=divorce, x='marriage_year', y='num_kids')
plt.show()

divorce.corr(numeric_only=True)

sns.heatmap(divorce.corr(numeric_only=True), annot=True)
plt.show()

sns.scatterplot(data=divorce, x='income_man', y='income_woman')
plt.show()

sns.pairplot(data=divorce)
plt.show()

sns.pairplot(data=divorce, vars=["income_man", "income_woman",
"marriage_duration"])
plt.show()

print("Before: ", divorce.isna().sum())
divorce["num_kids_temp"] = divorce["num_kids"].fillna(0)
print("After: ", divorce.isna().sum())
```

```
sns.scatterplot(data=divorce, x='marriage_duration',
y='num_kids_temp')
plt.show()

sns.pairplot(data=divorce, vars=["income_woman", "marriage_duration"])

divorce['education_man'].value_counts()

sns.histplot(data=divorce, x='marriage_duration', binwidth=1)
plt.show()

sns.histplot(data=divorce, x='marriage_duration', hue='education_man',
binwidth=1)
plt.show()

sns.kdeplot(data=divorce, x='marriage_duration', hue='education_man')
plt.show()

sns.kdeplot(data=divorce, x='marriage_duration', hue='education_man',
cut=0)
plt.show()

sns.kdeplot(data=divorce, x='marriage_duration', hue='education_man',
cut=0, cumulative=True)
plt.show()

divorce["man_age_marriage"] = divorce["marriage_year"] -
divorce["dob_man"].dt.year
divorce["woman_age_marriage"] = divorce["marriage_year"] -
divorce["dob_woman"].dt.year

sns.scatterplot(data=divorce, x='man_age_marriage',
y='woman_age_marriage', hue='education_man')
plt.show()

sns.scatterplot(data=divorce, x='woman_age_marriage',
y='income_woman', hue='education_woman')
plt.show()

planes = pd.read_csv("Airlines_unclean.csv", index_col=0)

print(planes["Destination"].value_counts())

pd.crosstab(planes["Source"], planes["Destination"],
values=planes["Price"], aggfunc="median")

salaries = pd.read_csv("Salary_Rupee_USD.csv", index_col=0)

relative_frequency =
salaries["Job_Category"].value_counts(normalize=True)
relative_frequency
```

```
pd.crosstab(index=salaries["Company_Size"],
columns=salaries["Experience"])

pd.crosstab(index=salaries["Job_Category"],
columns=salaries["Company_Size"])

pd.crosstab(index=salaries["Job_Category"],
columns=salaries["Company_Size"], values=salaries["Salary_USD"],
aggfunc="mean")

planes = pd.read_csv('Airlines_unclean.csv', index_col = 0,
parse_dates=['Date_of_Journey','Dep_Time','Arrival_Time'], date_format
= "%d/%m/%Y" )

# Remove the string character
planes["Duration"] = planes["Duration"].str.replace("h", ".")
planes["Duration"] = planes["Duration"].str.replace("m", "")
planes["Duration"] = planes["Duration"].str.replace(" ", "")

# Convert to float data type
planes["Duration"] = planes["Duration"].astype(float)
print(planes.info())
ax = sns.heatmap(planes.corr(numeric_only=True), annot=True)
ax.set_ylim([0,2])
plt.show()

#remove Nan values
threshold = len(planes) * 0.05
print(threshold)

# 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())

#planes = planes.drop(columns = ['Additional_Info'])
# Calculate median plane ticket prices by Airline
airline_prices = planes.groupby("Airline")["Price"].median()
print(airline_prices)
print('==========================================')
# Convert to a dictionary
prices_dict = airline_prices.to_dict()
print(prices_dict)
```

```
print('==========================================')
# 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())

print(planes["Total_Stops"].value_counts())

planes["Total_Stops"] = planes["Total_Stops"].str.replace("non-stop",
"0")
planes["Total_Stops"] = planes["Total_Stops"].str.replace(" stops",
"")
planes["Total_Stops"] = planes["Total_Stops"].str.replace(" stop", "")
planes["Total_Stops"] = planes["Total_Stops"].astype(int)

print(planes["Total_Stops"].value_counts())

sns.heatmap(planes.corr(numeric_only=True), annot=True)
plt.show()

print(planes.dtypes)

planes["month"] = planes["Date_of_Journey"].dt.month
planes["weekday"] = planes["Date_of_Journey"].dt.weekday
print(planes[["month", "weekday", "Date_of_Journey"]].head())

planes["Dep_Time"] = pd.to_datetime(planes["Dep_Time"],
format='mixed')
planes["Arrival_Time"] = pd.to_datetime(planes["Arrival_Time"],
format='mixed')

planes["Dep_Hour"] = planes["Dep_Time"].dt.hour
planes["Arrival_Hour"] = planes["Arrival_Time"].dt.hour

sns.heatmap(planes.corr(numeric_only=True), annot=True)
plt.show()

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

twenty_fifth = planes["Price"].quantile(0.25)
median = planes["Price"].median()
seventy_fifth = planes["Price"].quantile(0.75)
maximum = planes["Price"].max()

labels = ["Economy", "Premium Economy", "Business Class", "First
Class"]
bins = [0, twenty_fifth, median, seventy_fifth, maximum]

planes["Price Category"] = pd.cut(planes["Price"], bins=bins,
labels=labels)
planes[["Price", "Price Category"]].head()
```

```
sns.countplot(data=planes,x="Airline" , hue="Price Category")
plt.xticks(rotation=45)
plt.show()
```

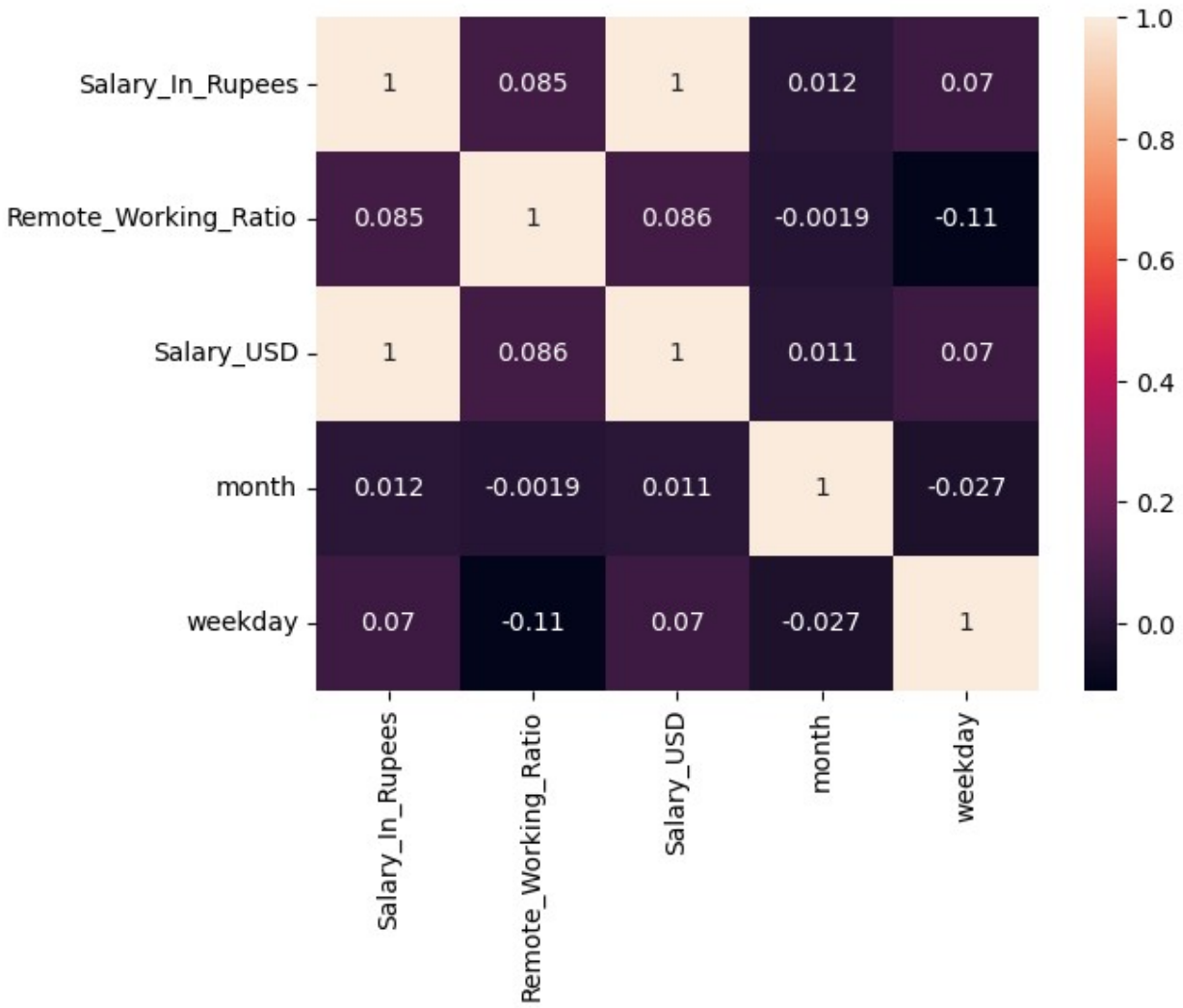
```
salaries = pd.read_csv('Salaries_with_date_of_response.csv',
index_col=0)
salaries["date_of_response"] =
pd.to_datetime(salaries["date_of_response"], format='%d/%m/%Y')
```

```
salaries.dtypes
```

```
Designation                    object
date_of_response       datetime64[ns]
Experience                     object
Employment_Status              object
Salary_In_Rupees              float64
Employee_Location              object
Company_Location               object
Company_Size                   object
Remote_Working_Ratio            int64
Salary_USD                    float64
Job_Category                   object
dtype: object
```

```
salaries["month"] = salaries["date_of_response"].dt.month
salaries["weekday"] = salaries["date_of_response"].dt.weekday
```

```
sns.heatmap(salaries.corr(numeric_only=True), annot=True)
plt.show()
```

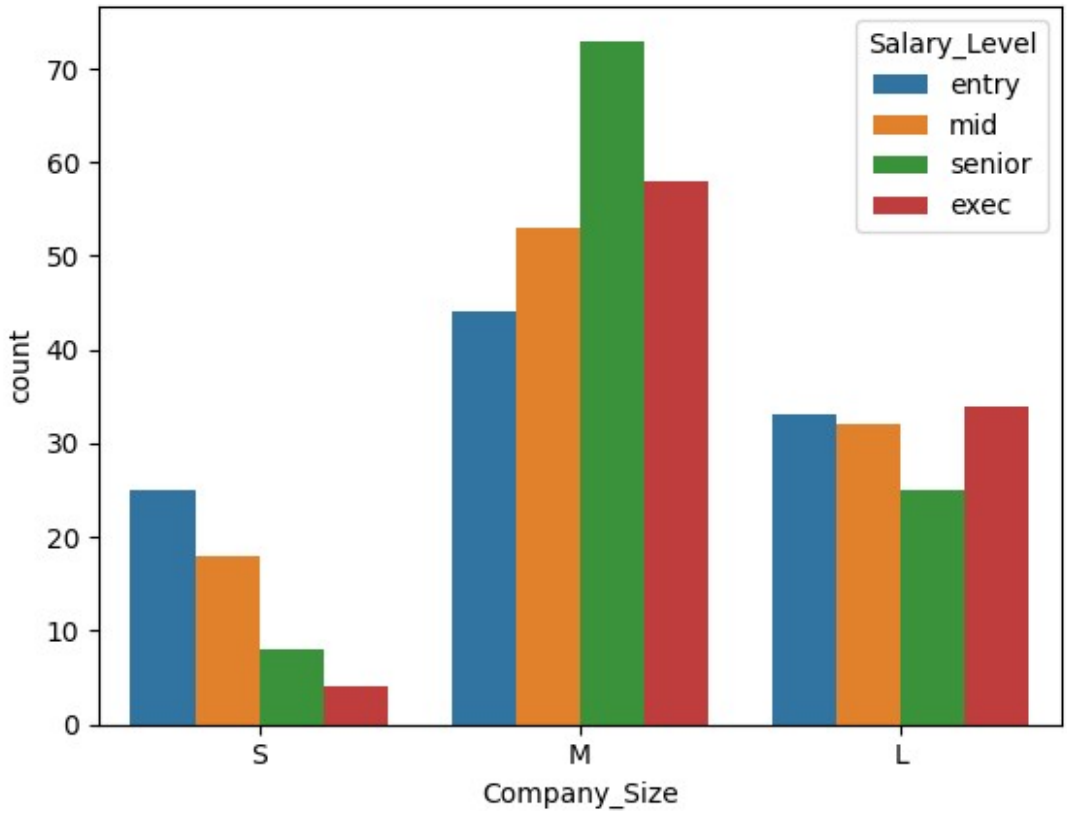



```
twenty_fifth = salaries["Salary_USD"].quantile(0.25)
salaries_median = salaries["Salary_USD"].median()
seventy_fifth = salaries["Salary_USD"].quantile(0.75)

salary_labels = ["entry", "mid", "senior", "exec"]
salary_ranges = [0, twenty_fifth, salaries_median, seventy_fifth,
salaries["Salary_USD"].max()]

salaries["Salary_Level"] = pd.cut(salaries["Salary_USD"],
bins=salary_ranges, labels=salary_labels)

sns.countplot(data=salaries, x="Company_Size", hue="Salary_Level")
plt.show()
```

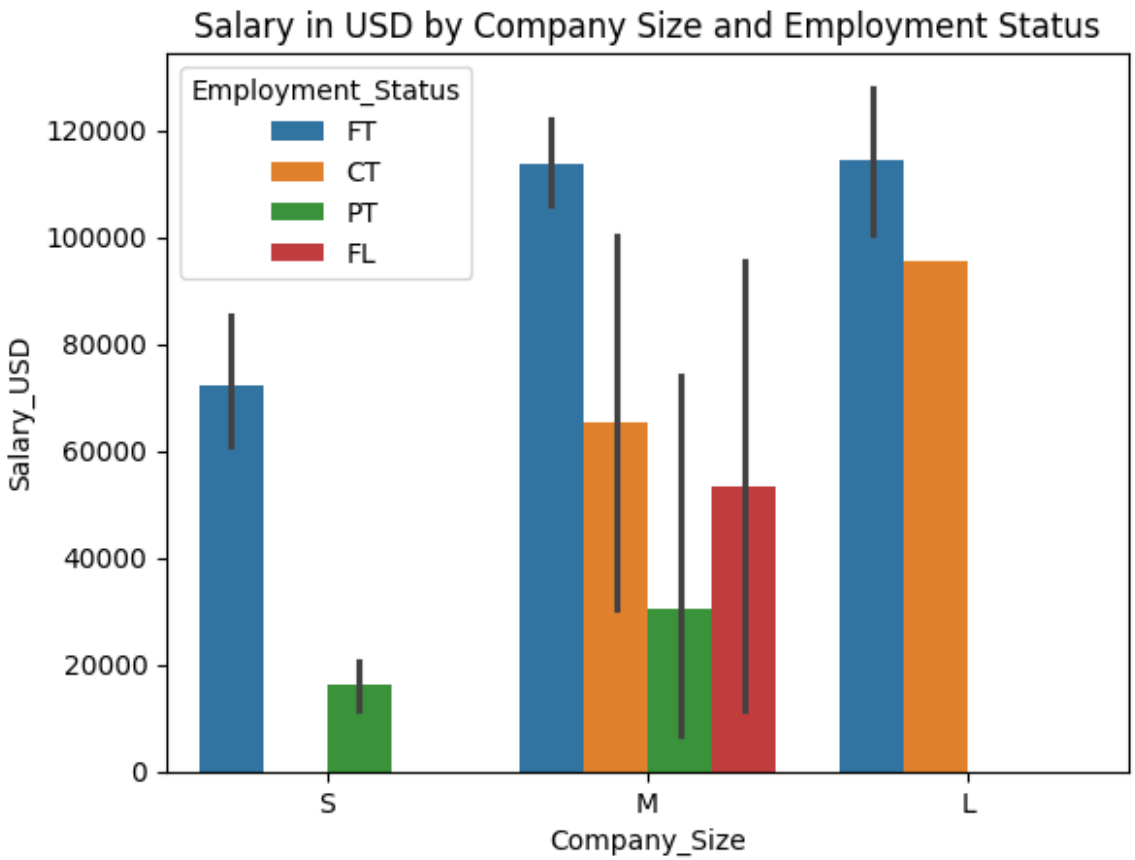



```
usa_and_gb = salaries[salaries["Employee_Location"].isin(["US",
"GB"])]

sns.barplot(data=usa_and_gb, x="Employee_Location", y="Salary_USD")
plt.title("Salary in USD by Employee Location")
plt.show()

sns.barplot(data=salaries, x="Company_Size", y="Salary_USD",
hue="Employment_Status")
plt.title("Salary in USD by Company Size and Employment Status")
plt.show()
```


Salary in USD by Employee Location
140000
120000
100000
80000
60000
40000
20000
0
Salary_USD
GB
US
Employee_Location


Salary in USD by Company Size and Employment Status
Employment_Status
FT
CT
PT
FL
120000
100000
80000
60000
40000
20000
0
Salary_USD
S
M
L
Company_Size