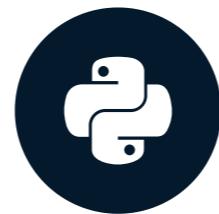


# Considerations for categorical data

EXPLORATORY DATA ANALYSIS IN PYTHON



George Boorman

Curriculum Manager, DataCamp

# Why perform EDA?

- Detecting patterns and relationships
- Generating questions, or **hypotheses**
- Preparing data for machine learning



<sup>1</sup> Image credit: <https://unsplash.com/@simonesecchi>

# Representative data

- Sample represents the population

For example:

- Education versus income in USA
  - Can't use data from France

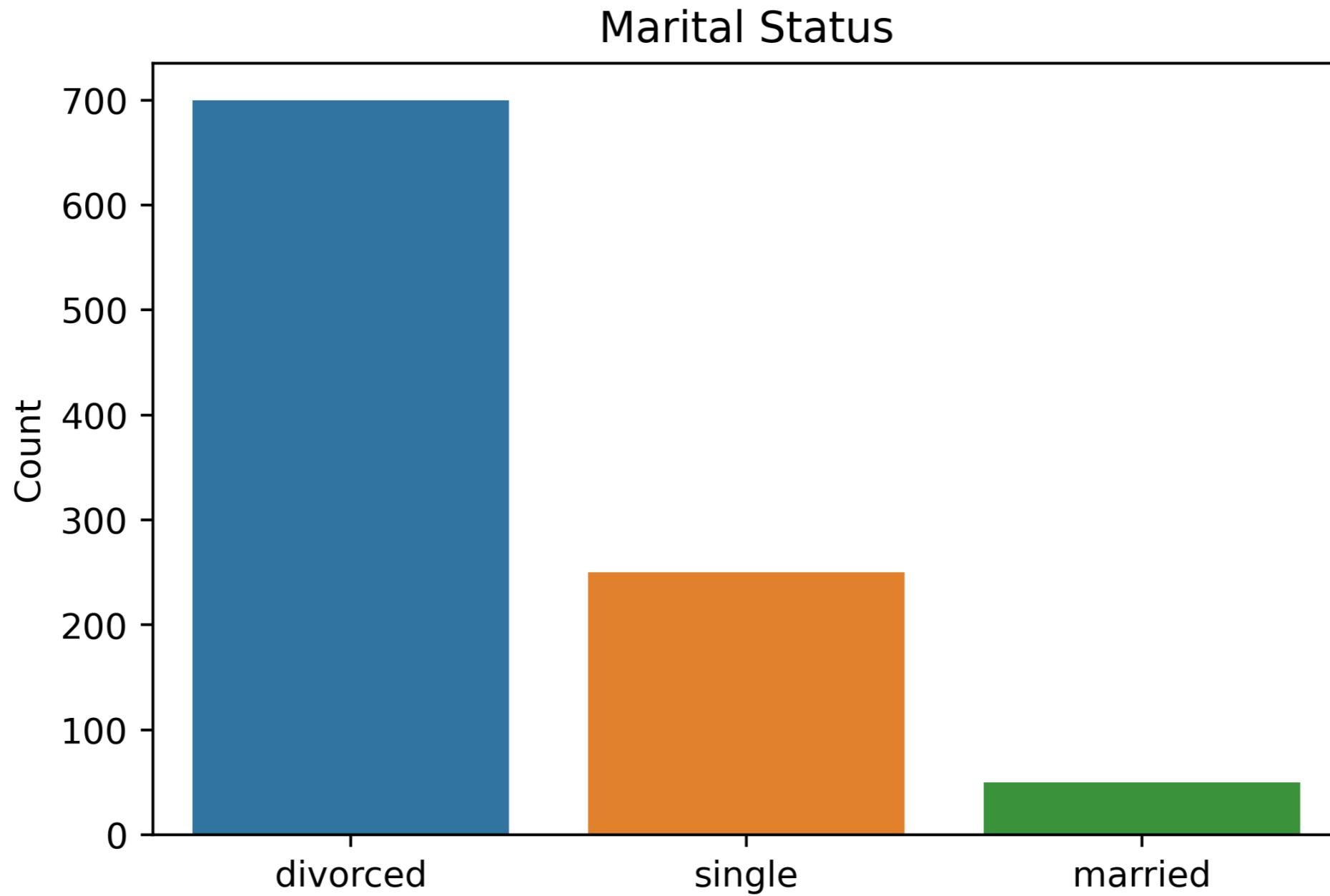


<sup>1</sup> Image credits: [https://unsplash.com/@cristina\\_glebova](https://unsplash.com/@cristina_glebova); [https://unsplash.com/@nimbus\\_vulpis](https://unsplash.com/@nimbus_vulpis)

# Categorical classes

- Classes = labels
- Survey people's attitudes towards marriage
  - Marital status
    - Single
    - Married
    - Divorced

# Class imbalance



# Class frequency

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

```
Cochin      4391  
Banglore    2773  
Delhi       1219  
New Delhi   888  
Hyderabad   673  
Kolkata     369
```

```
Name: Destination, dtype: int64
```

# Relative class frequency

- 40% of internal Indian flights have a destination of Delhi

```
planes["Destination"].value_counts(normalize=True)
```

```
Cochin      0.425773
Banglore    0.268884
Delhi       0.118200
New Delhi   0.086105
Hyderabad   0.065257
Kolkata    0.035780
Name: Destination, dtype: float64
```

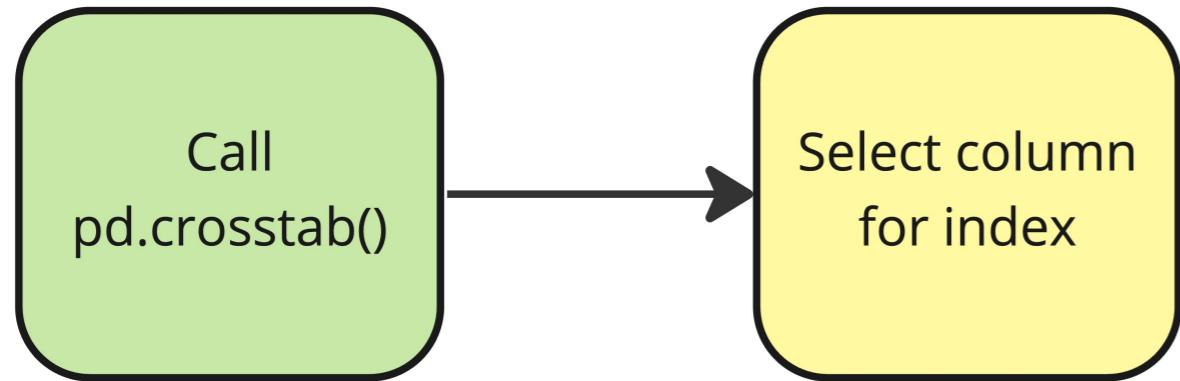
- Is our sample representative of the population (Indian internal flights)?

# Cross-tabulation

Call  
pd.crosstab()

pd.crosstab(

# Select index



```
pd.crosstab(planes["Source"],
```

# Select columns



```
pd.crosstab(planes["Source"], planes["Destination"])
```

# Cross-tabulation

Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
Source						
Banglore	0	0	1199	0	0	868
Chennai	0	0	0	0	364	0
Delhi	0	4318	0	0	0	0
Kolkata	2720	0	0	0	0	0
Mumbai	0	0	0	662	0	0

# Extending cross-tabulation

Source	Destination	Median Price (IDR)
Banglore	Delhi	4232.21
Banglore	New Delhi	12114.56
Chennai	Kolkata	3859.76
Delhi	Cochin	9987.63
Kolkata	Banglore	9654.21
Mumbai	Hyderabad	3431.97

# Aggregated values with pd.crosstab()

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

Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
Source						
Banglore		NaN	4823.0	NaN	NaN	10976.5
Chennai		NaN	NaN	NaN	3850.0	NaN
Delhi		NaN	10262.0	NaN	NaN	NaN
Kolkata	9345.0		NaN	NaN	NaN	NaN
Mumbai		NaN	NaN	3342.0	NaN	NaN

# Comparing sample to population

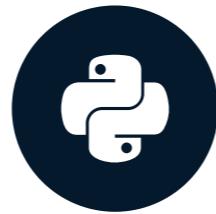
Source	Destination	Median Price (IDR)	Median Price (dataset)
Banglore	Delhi	4232.21	4823.0
Banglore	New Delhi	12114.56	10976.50
Chennai	Kolkata	3859.76	3850.0
Delhi	Cochin	9987.63	10260.0
Kolkata	Banglore	9654.21	9345.0
Mumbai	Hyderabad	3431.97	3342.0

# **Let's practice!**

**EXPLORATORY DATA ANALYSIS IN PYTHON**

# Generating new features

EXPLORATORY DATA ANALYSIS IN PYTHON

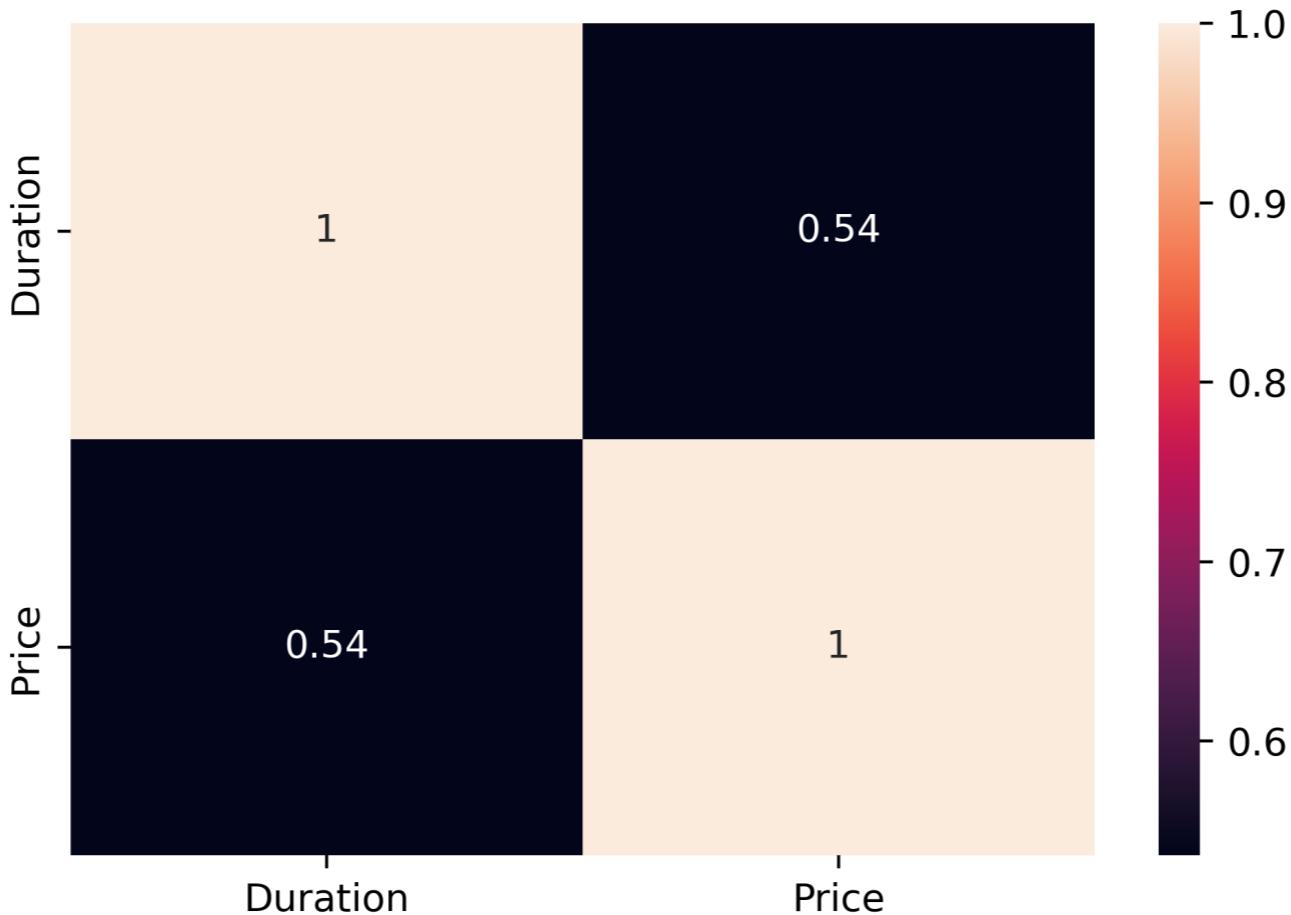


George Boorman

Curriculum Manager, DataCamp

# Correlation

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



# Viewing data types

```
print(planes.dtypes)
```

```
Airline          object  
Date_of_Journey    datetime64[ns]  
Source           object  
Destination       object  
Route             object  
Dep_Time         datetime64[ns]  
Arrival_Time      datetime64[ns]  
Duration          float64  
Total_Stops        object  
Additional_Info     object  
Price             float64  
dtype: object
```

# Total stops

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

```
1 stop        4107  
non-stop     2584  
2 stops      1127  
3 stops       29  
4 stops        1
```

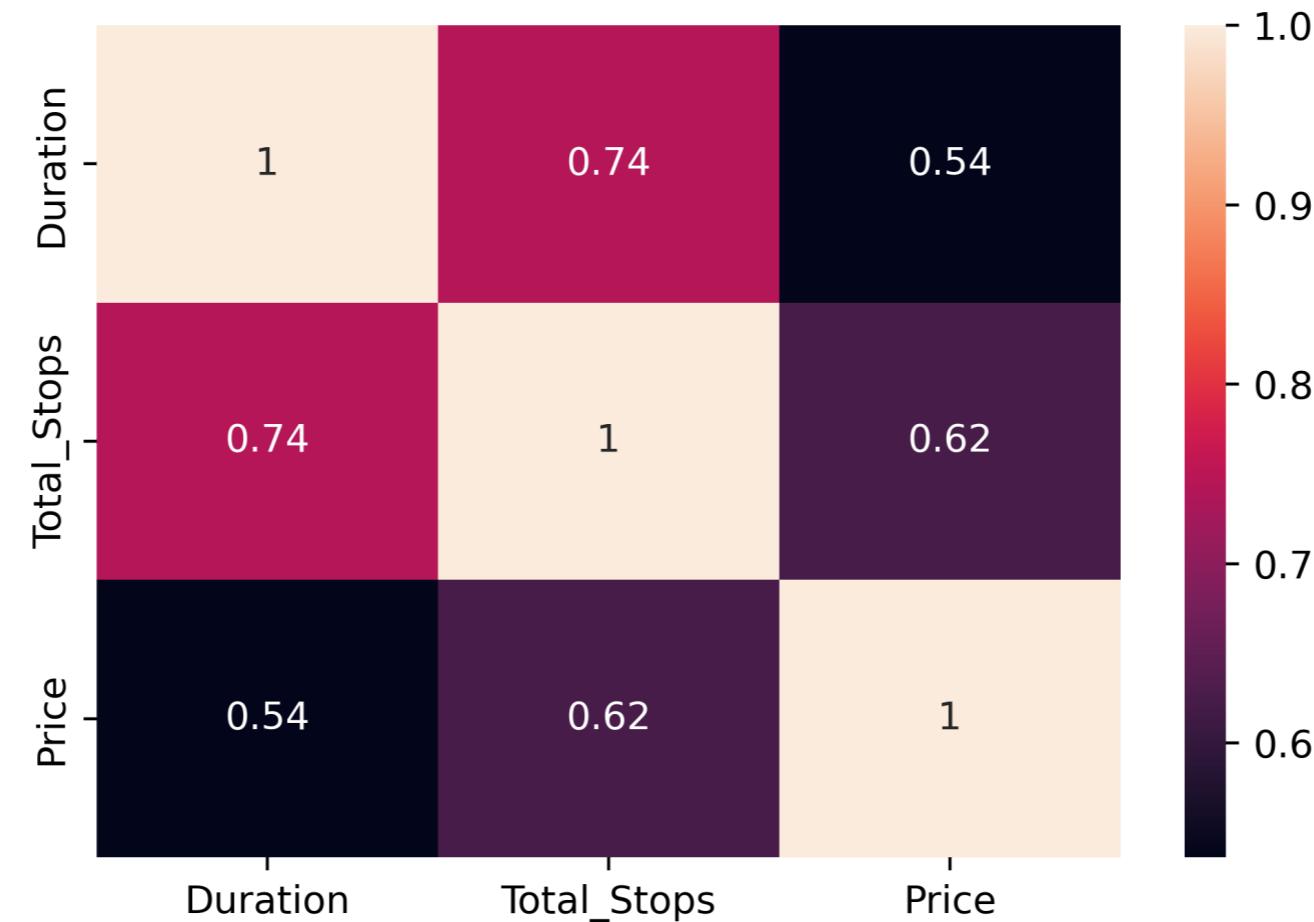
```
Name: Total_Stops, dtype: int64
```

# Cleaning total stops

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

# Correlation

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



# Dates

```
print(planes.dtypes)
```

```
Airline          object  
Date_of_Journey    datetime64[ns]  
Source           object  
Destination       object  
Route             object  
Dep_Time         datetime64[ns]  
Arrival_Time      datetime64[ns]  
Duration          float64  
Total_Stops        int64  
Additional_Info    object  
Price             float64  
dtype: object
```

# Extracting month and weekday

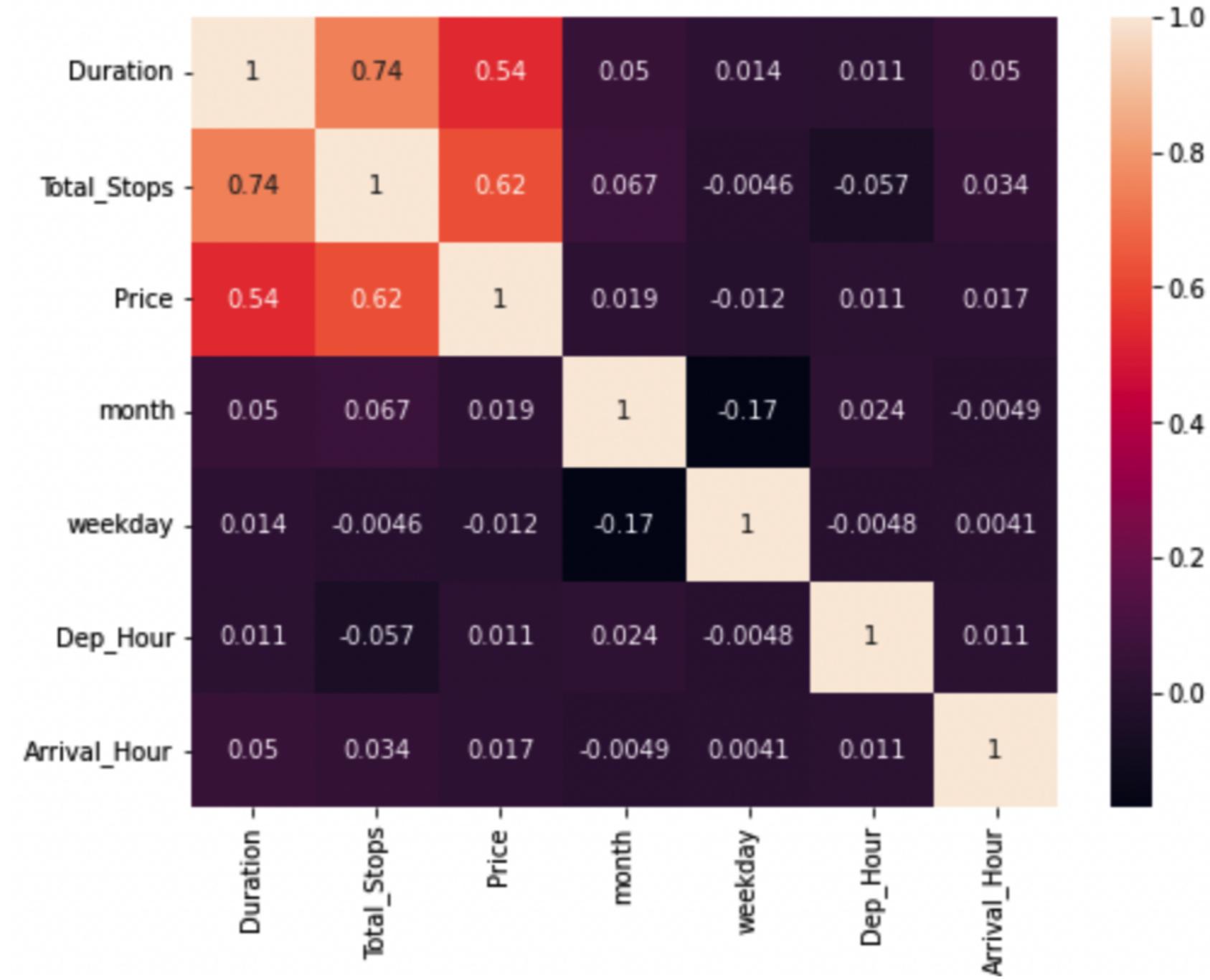
```
planes["month"] = planes["Date_of_Journey"].dt.month  
planes["weekday"] = planes["Date_of_Journey"].dt.weekday  
print(planes[["month", "weekday", "Date_of_Journey"]].head())
```

	month	weekday	Date_of_Journey
0	9	4	2019-09-06
1	12	3	2019-12-05
2	1	3	2019-01-03
3	6	0	2019-06-24
4	12	1	2019-12-03

# Departure and arrival times

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

# Correlation



# Creating categories

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

```
count      7848.000000
mean      9035.413609
std       4429.822081
min      1759.000000
25%      5228.000000
50%      8355.000000
75%     12373.000000
max     54826.000000
Name: Price, dtype: float64
```

Range	Ticket Type
<= 5228	Economy
> 5228 <= 8355	Premium Economy
> 8355 <= 12373	Business Class
> 12373	First Class

# Descriptive statistics

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

# Labels and bins

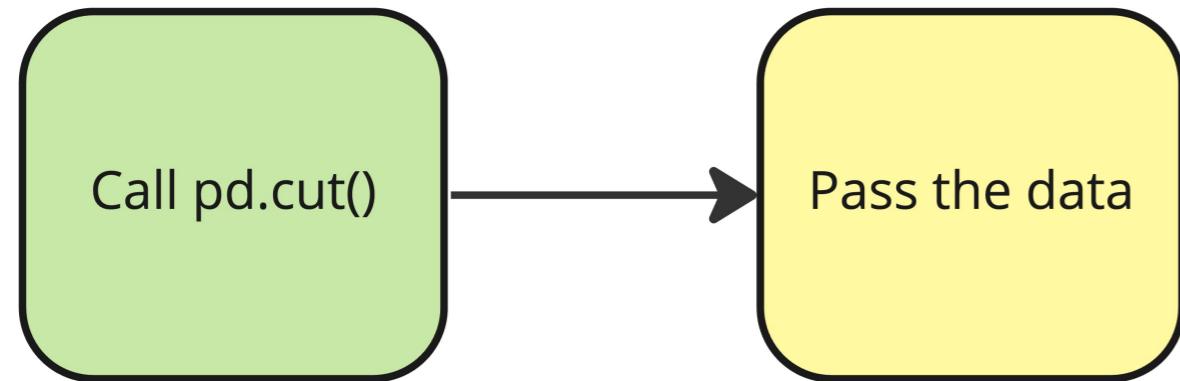
```
labels = ["Economy", "Premium Economy", "Business Class", "First Class"]  
bins = [0, twenty_fifth, median, seventy_fifth, maximum]
```

# pd.cut()

Call pd.cut()

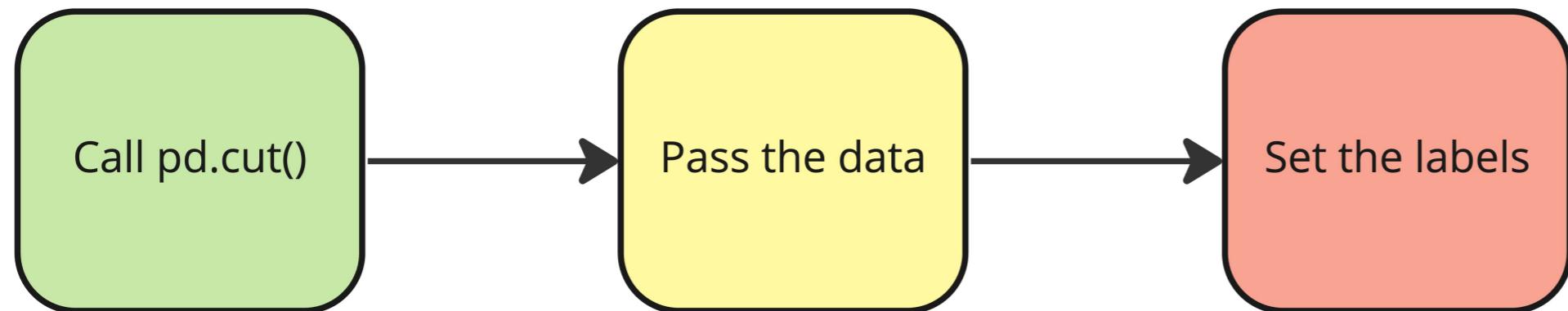
```
planes["Price_Category"] = pd.cut(
```

# pd.cut()



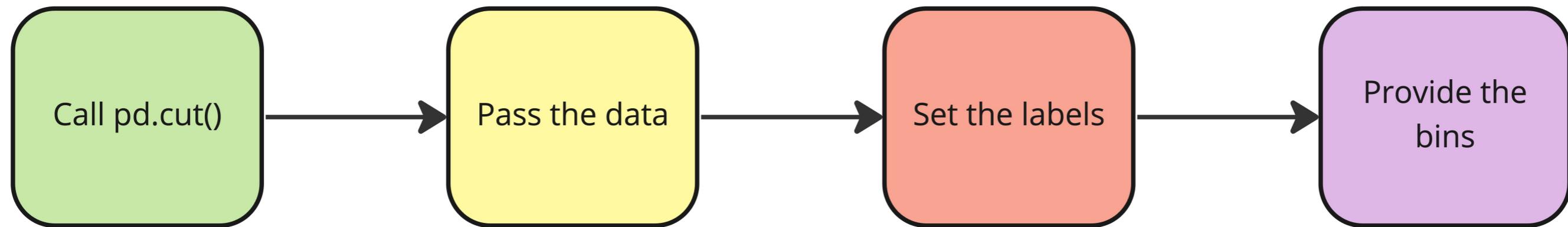
```
planes["Price_Category"] = pd.cut(planes["Price"],
```

# pd.cut()



```
planes["Price_Category"] = pd.cut(planes["Price"],  
                                 labels=labels,
```

# pd.cut()



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

# Price categories

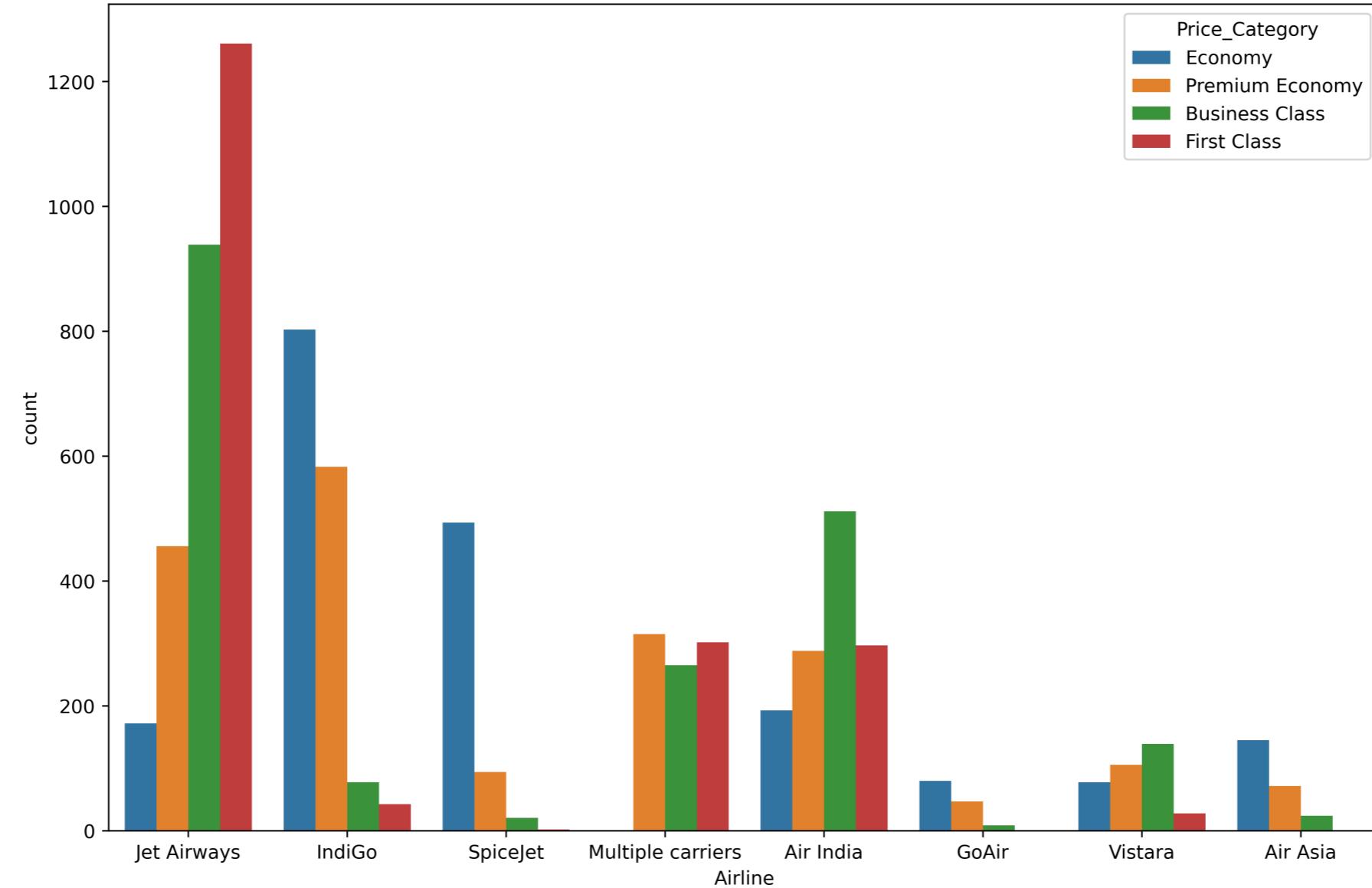
```
print(planes[["Price", "Price_Category"]].head())
```

```
   Price  Price_Category
0  13882.0      First Class
1   6218.0  Premium Economy
2  13302.0      First Class
3   3873.0        Economy
4  11087.0    Business Class
```

# Price category by airline

```
sns.countplot(data=planes, x="Airline", hue="Price_Category")  
plt.show()
```

# Price category by airline

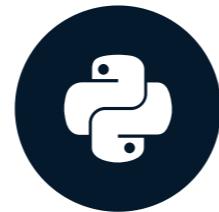


# **Let's practice!**

**EXPLORATORY DATA ANALYSIS IN PYTHON**

# Generating hypotheses

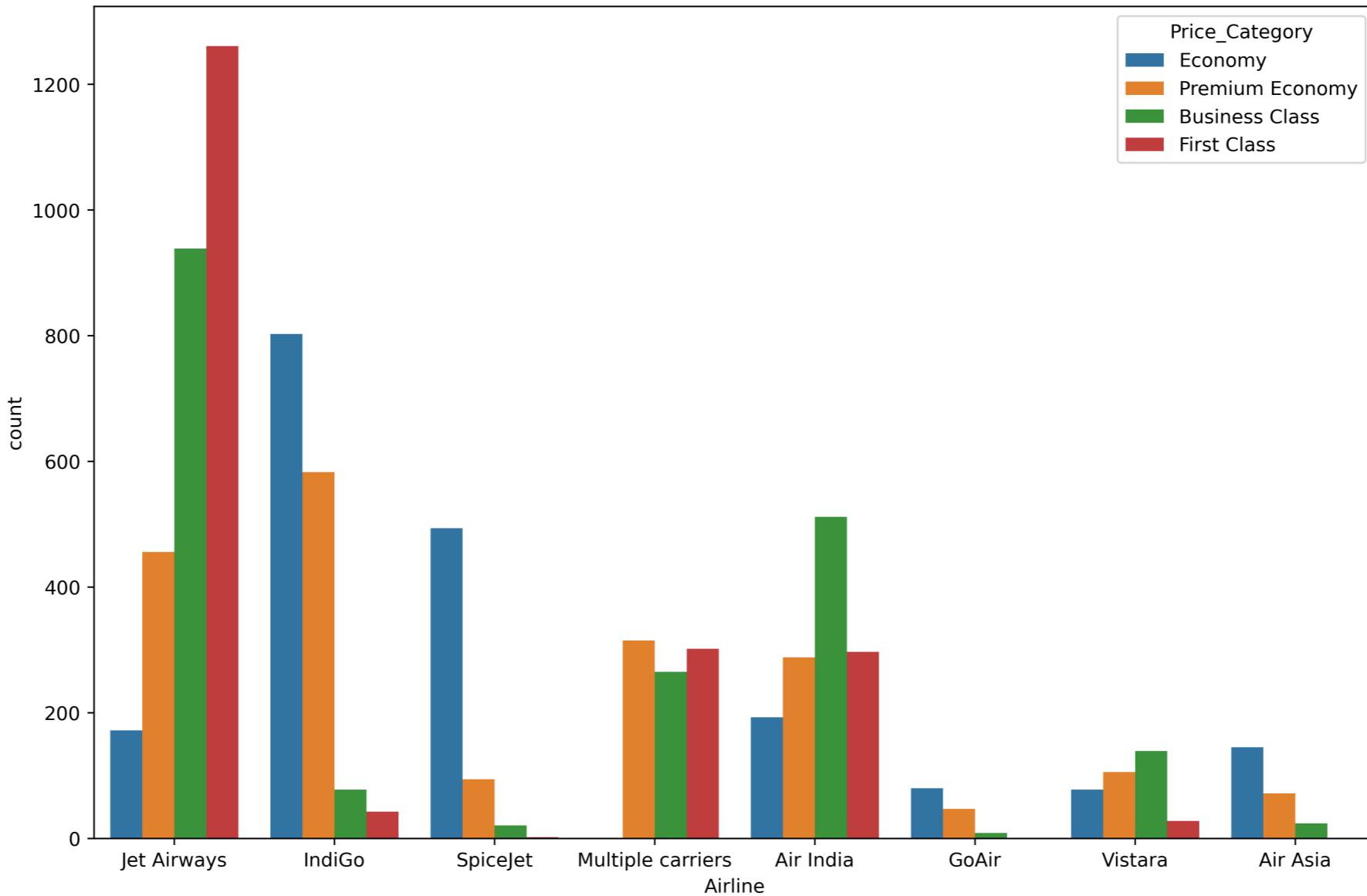
EXPLORATORY DATA ANALYSIS IN PYTHON



George Boorman

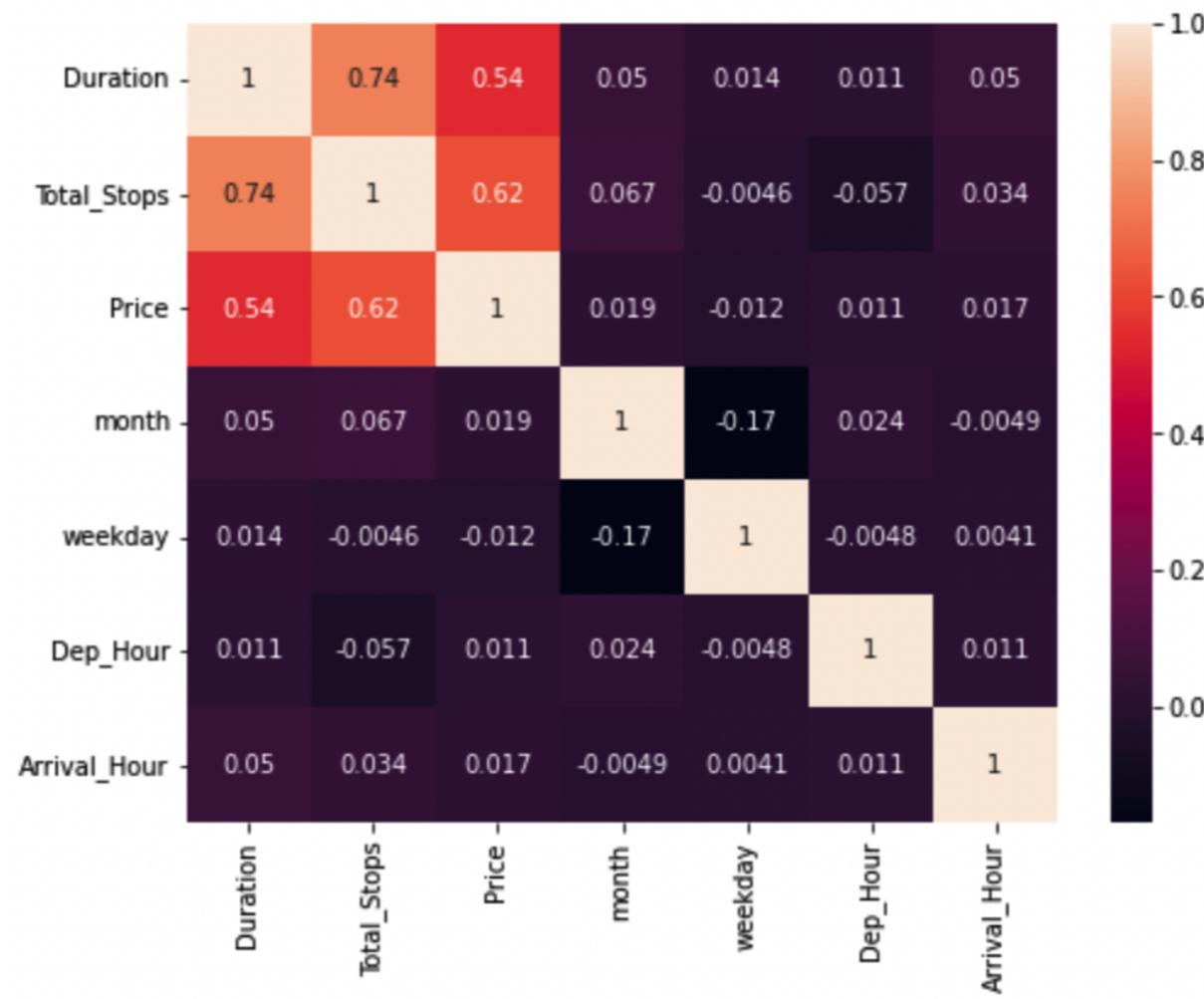
Curriculum Manager, DataCamp

# What do we know?



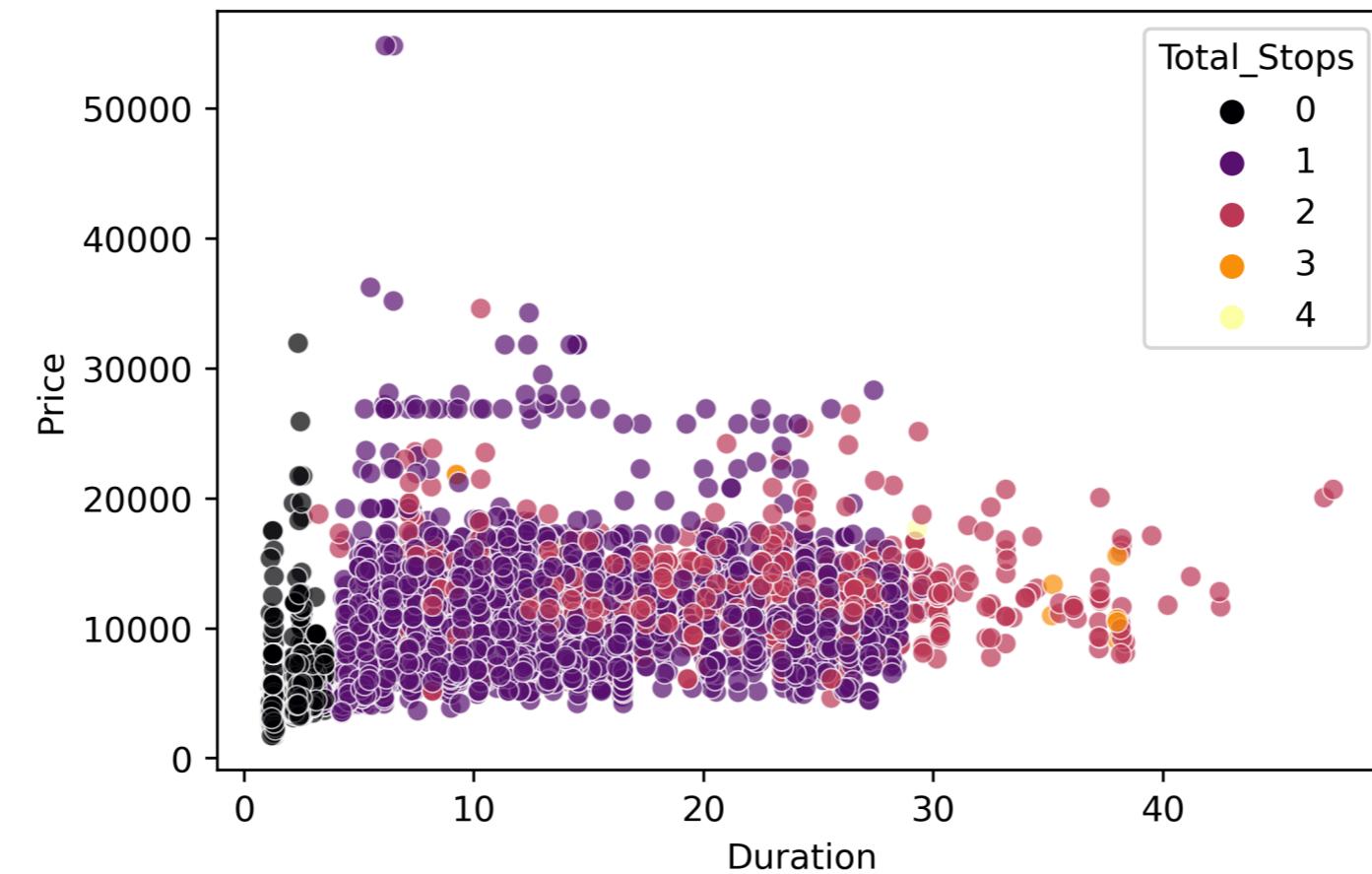
# What do we know?

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

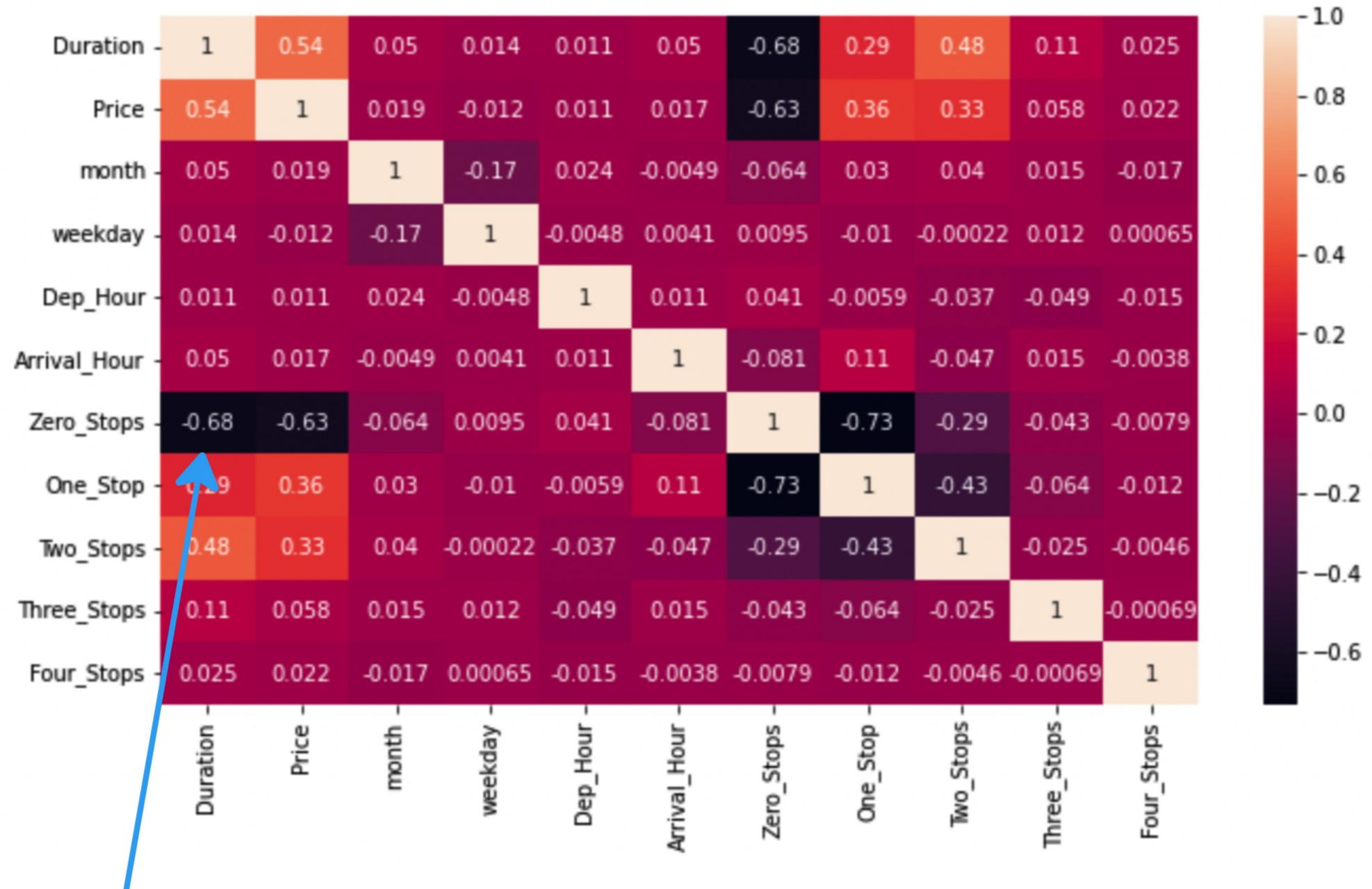


# Spurious correlation

```
sns.scatterplot(data=planes, x="Duration", y="Price", hue="Total_Stops")  
plt.show()
```



# How do we know?



# What is true?



- Would data from a different time give the same results?
- Detecting relationships, differences, and patterns:
  - We use **Hypothesis Testing**
- Hypothesis testing requires, prior to data collection:
  - Generating a hypothesis or question
  - A decision on what statistical test to use

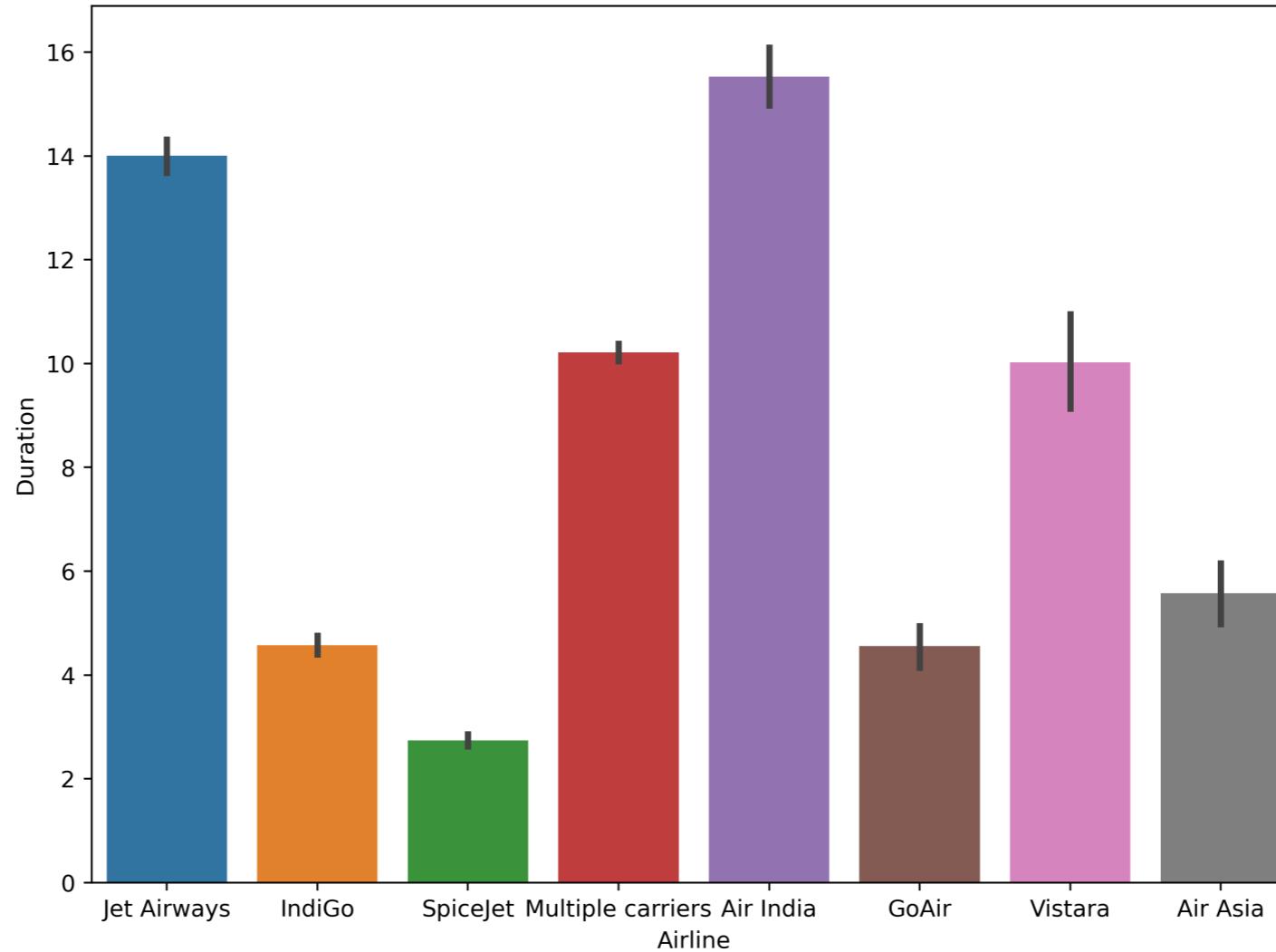
<sup>1</sup> Image credit: <https://unsplash.com/@markuswinkler>

# Data snooping



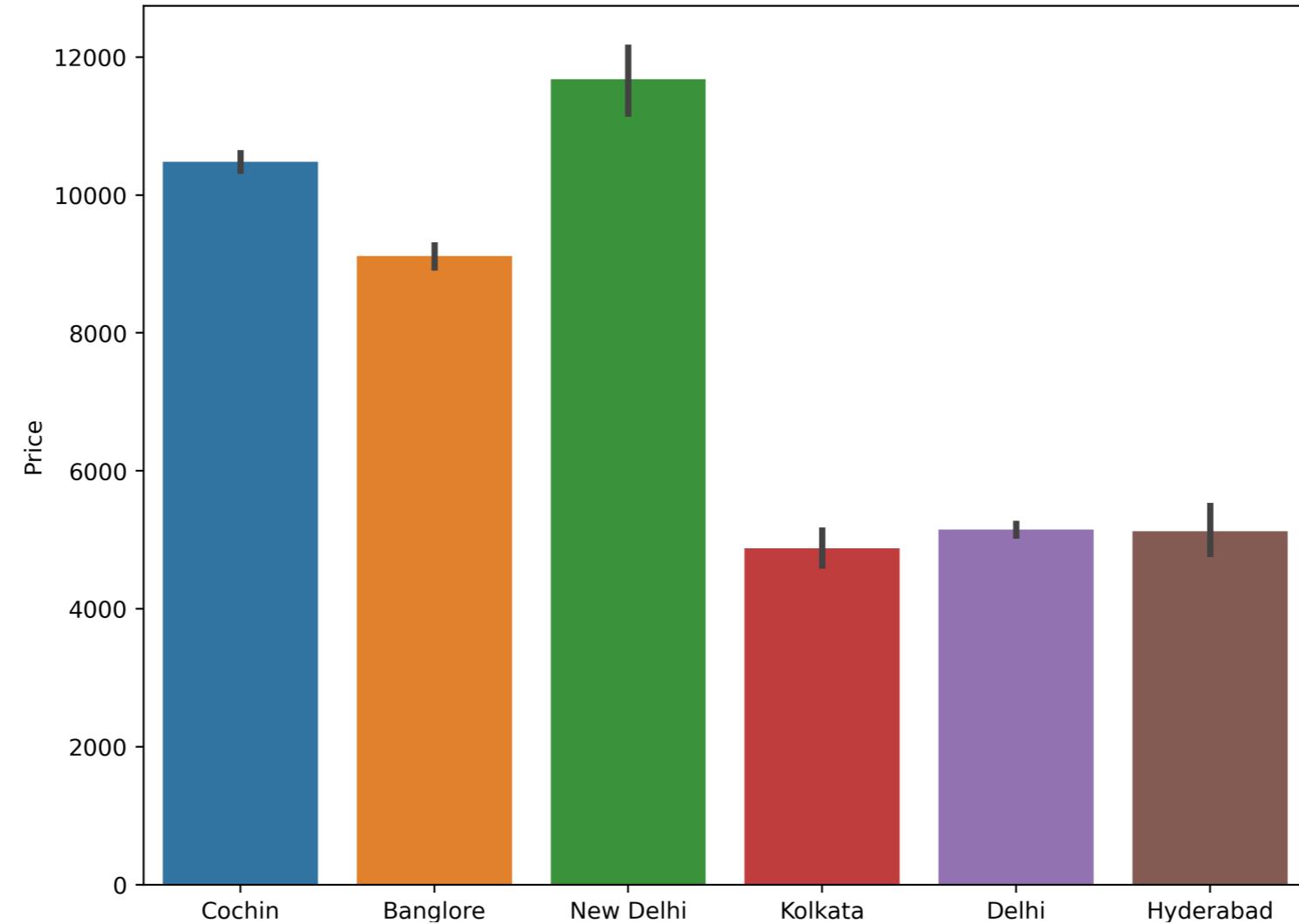
# Generating hypotheses

```
sns.barplot(data=planes, x="Airline", y="Duration")  
plt.show()
```



# Generating hypotheses

```
sns.barplot(data=planes, x="Destination", y="Price")  
plt.show()
```



# Next steps

- Design our experiment
- Involves steps such as:
  - Choosing a sample
  - Calculating how many data points we need
  - Deciding what statistical test to run

# **Let's practice!**

**EXPLORATORY DATA ANALYSIS IN PYTHON**

# Congratulations

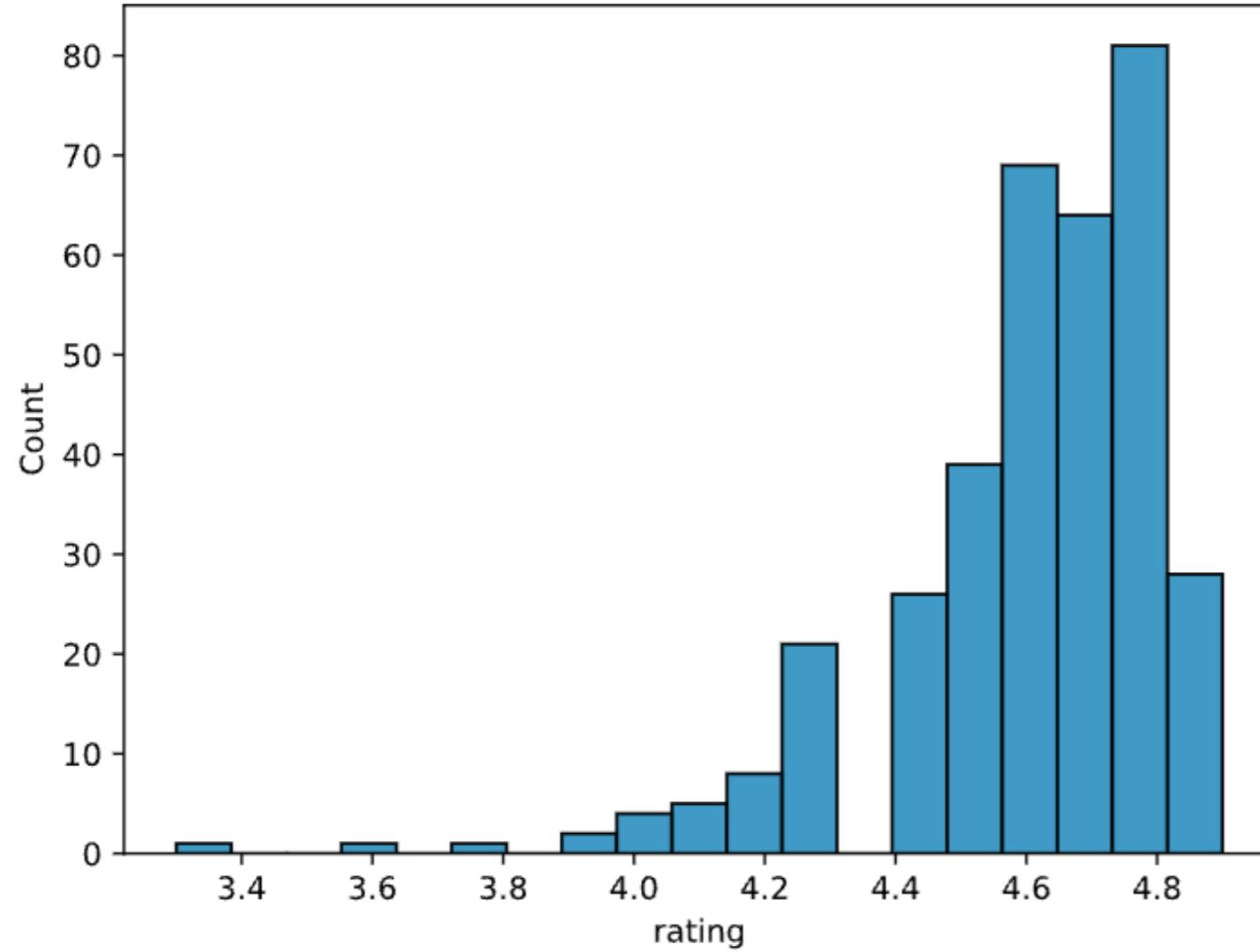
EXPLORATORY DATA ANALYSIS IN PYTHON



George Boorman

Curriculum Manager, DataCamp

# Inspection and validation



```
books["year"] = books["year"].astype(int)  
books.dtypes
```

```
name          object  
author        object  
rating       float64  
year         int64  
genre        object  
dtype: object
```

# Aggregation

```
books.groupby("genre").agg(  
    mean_rating=("rating", "mean"),  
    std_rating=("rating", "std"),  
    median_year=("year", "median"))  
)
```

genre	mean_rating	std_rating	median_year
Childrens	4.780000	0.122370	2015.0
Fiction	4.570229	0.281123	2013.0
Non Fiction	4.598324	0.179411	2013.0

# Address missing data

```
print(salaries.isna().sum())
```

```
Working_Year           12
Designation            27
Experience             33
Employment_Status      31
Employee_Location      28
Company_Size            40
Remote_Working_Ratio    24
Salary_USD              60
dtype: int64
```

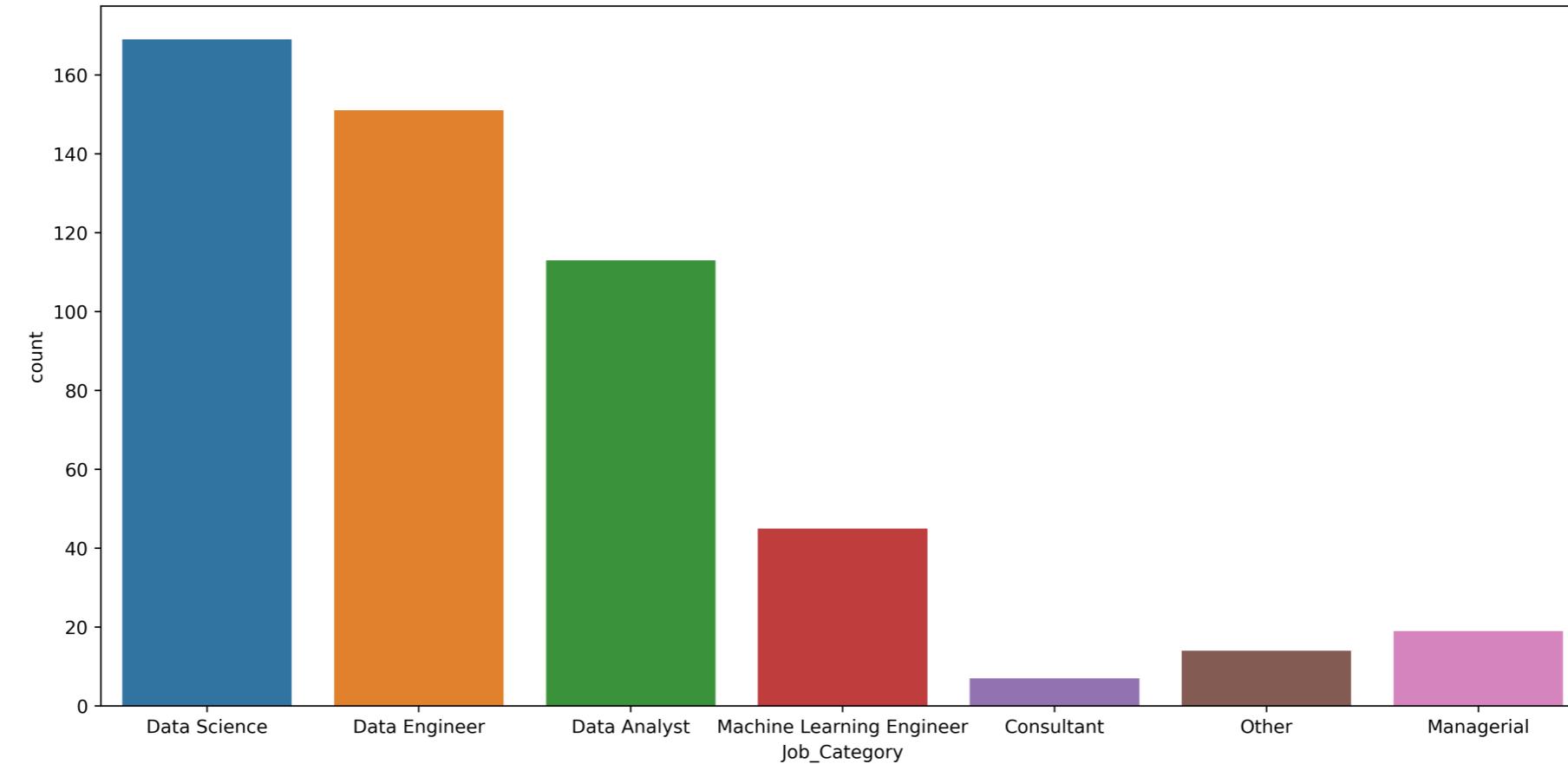
# Address missing data

- Drop missing values
- Impute mean, median, mode
- Impute by sub-group

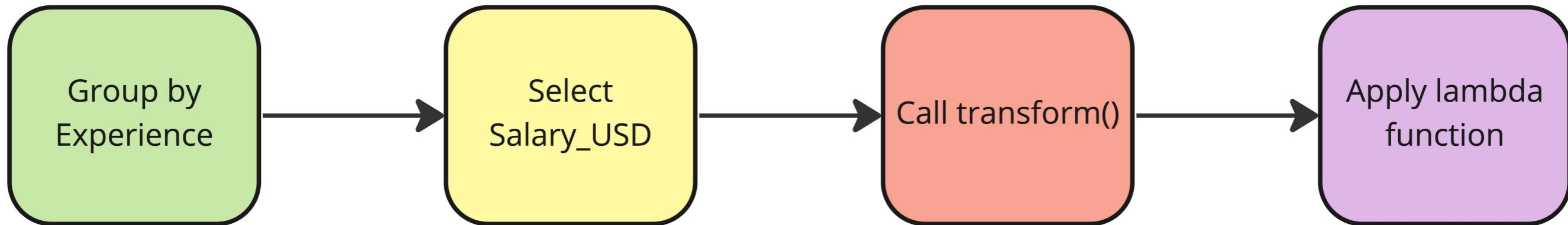
```
salaries_dict = salaries.groupby("Experience")["Salary_USD"].median().to_dict()  
salaries["Salary_USD"] = salaries["Salary_USD"].fillna(salaries["Experience"].map(salaries_dict))
```

# Analyze categorical data

```
salaries["Job_Category"] = np.select(conditions,  
                                    job_categories,  
                                    default="Other")
```



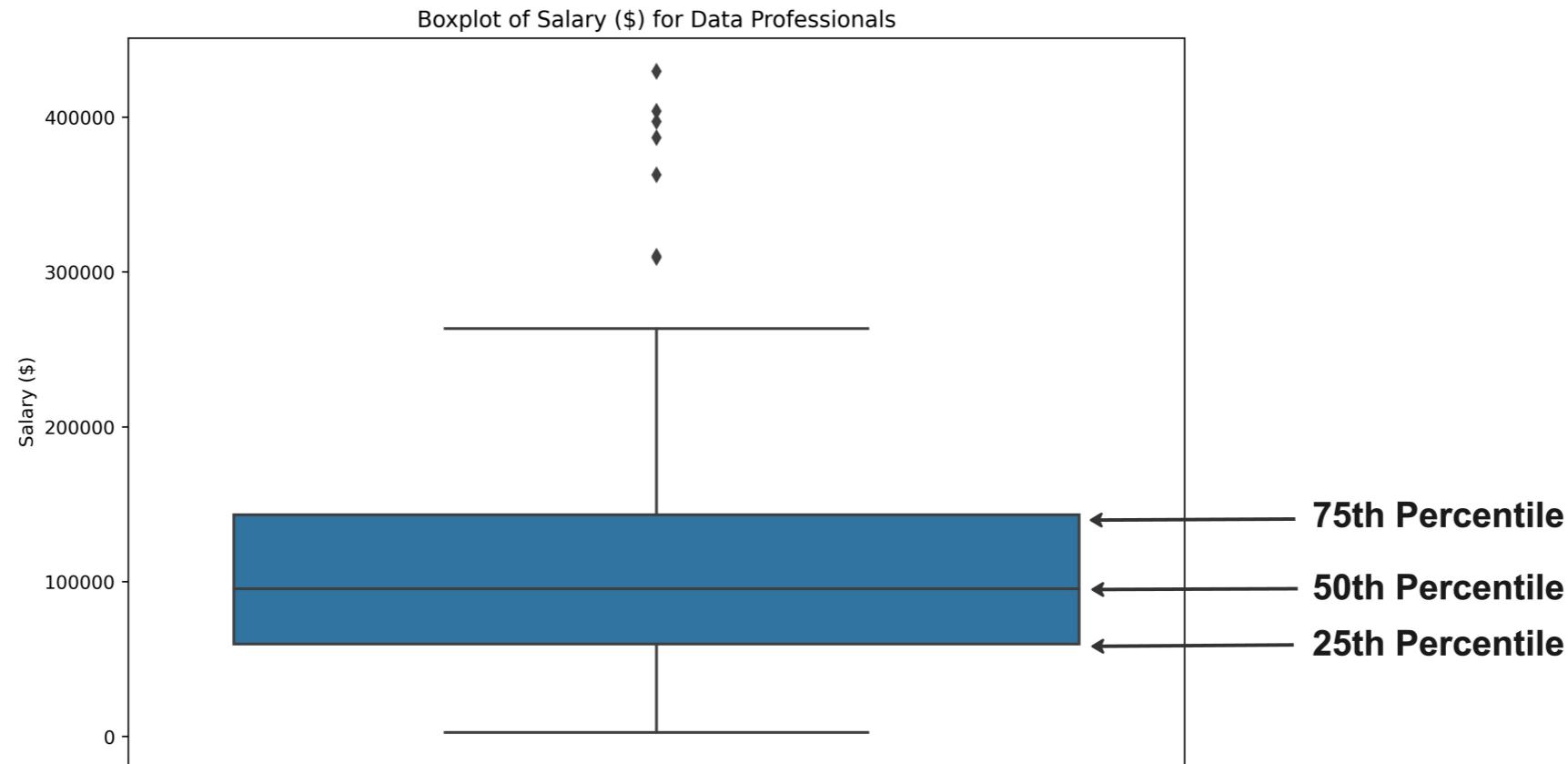
# Apply lambda functions



```
salaries["std_dev"] = salaries.groupby("Experience")["Salary_USD"].transform(lambda x: x.std())
```

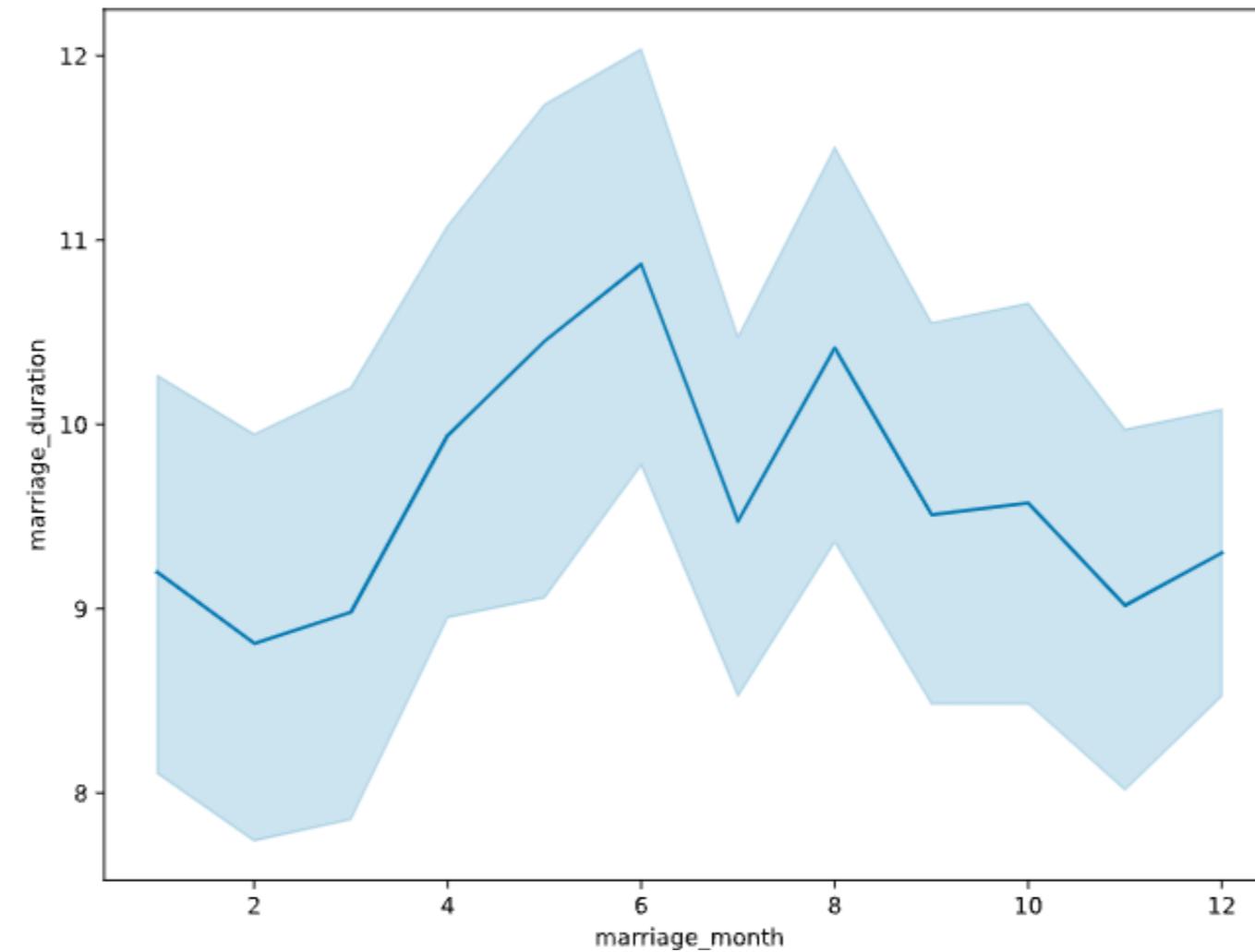
# Handle outliers

```
sns.boxplot(data=salaries,  
             y="Salary_USD")  
plt.show()
```



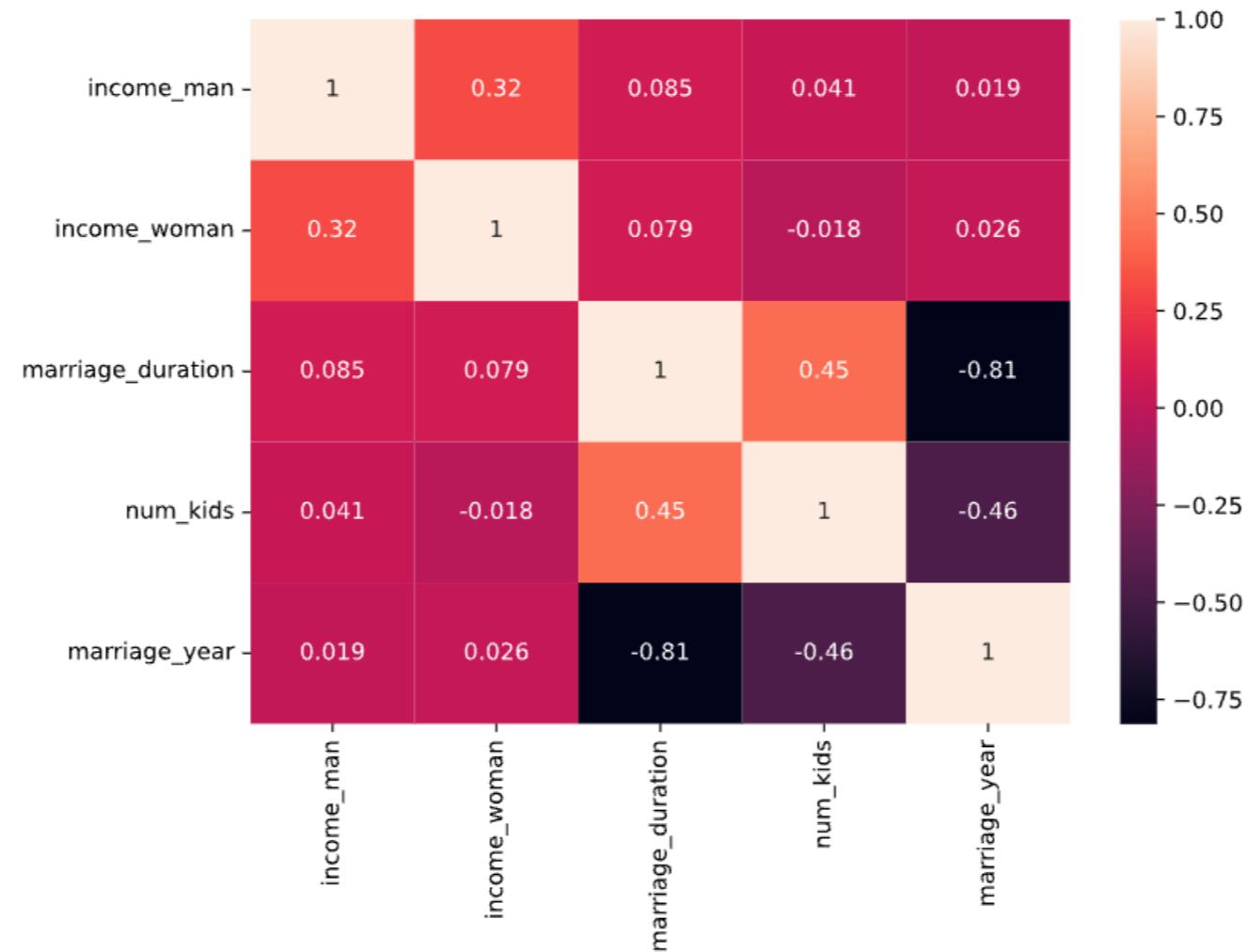
# Patterns over time

```
sns.lineplot(data=divorce, x="marriage_month", y="marriage_duration")  
plt.show()
```



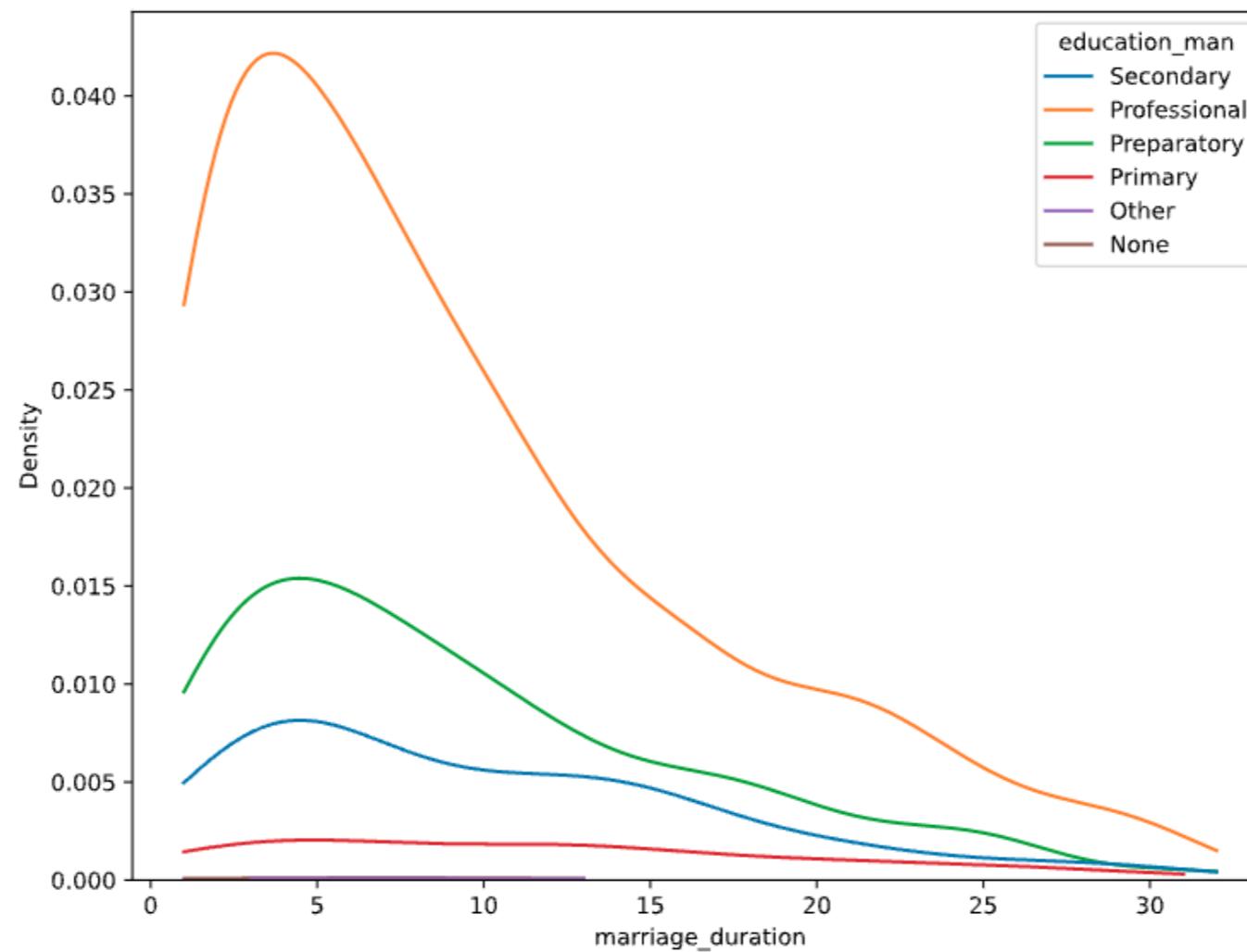
# Correlation

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



# Distributions

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

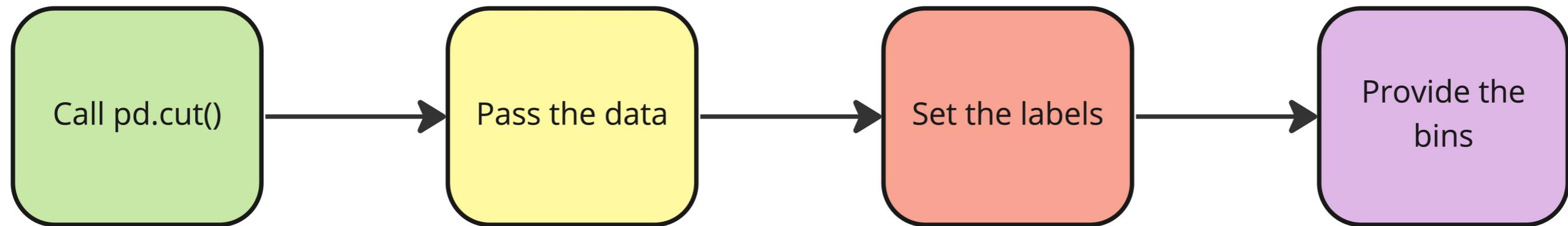


# Cross-tabulation

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

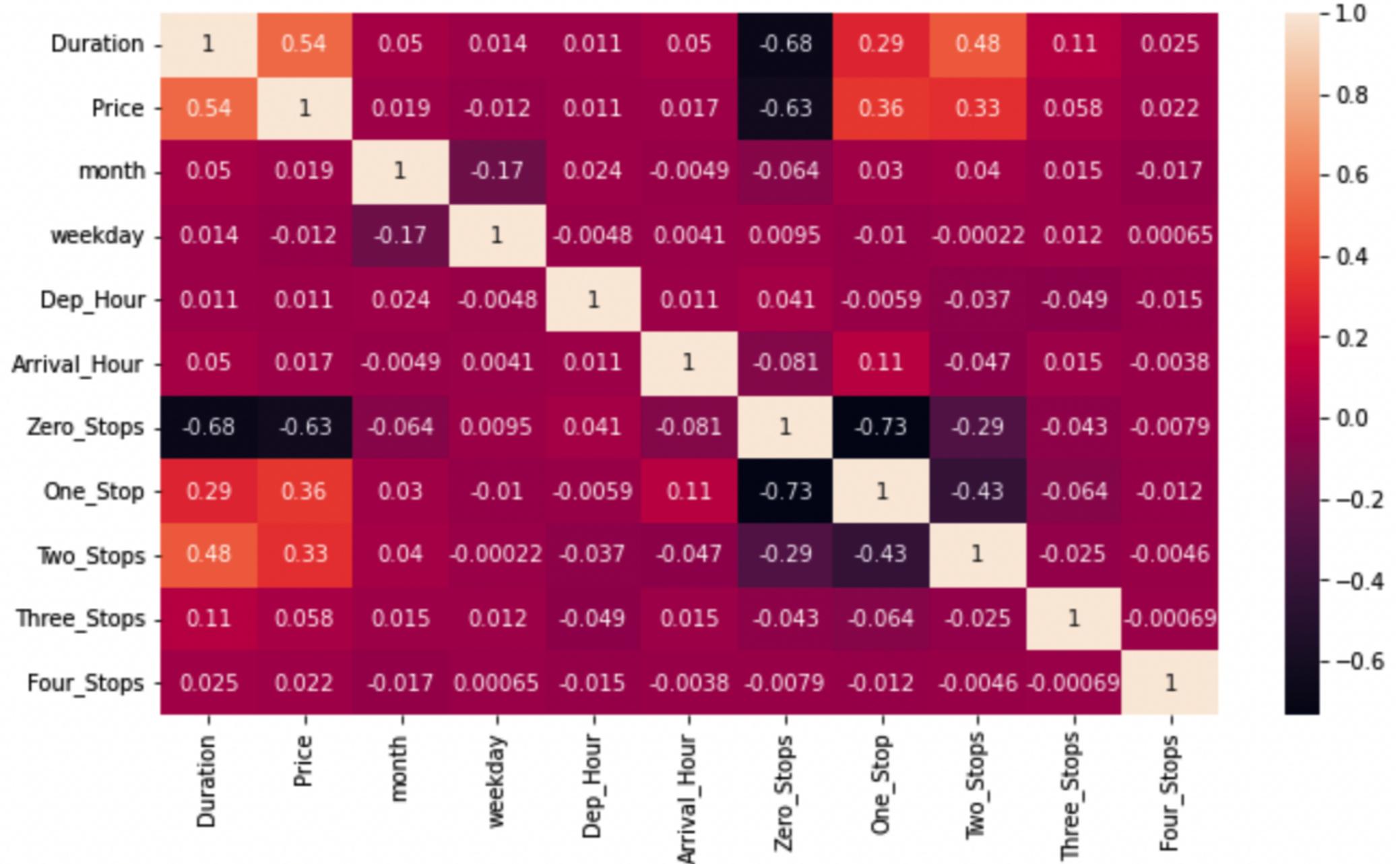
Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
Source						
Banglore		NaN	4823.0	NaN	NaN	10976.5
Chennai		NaN	NaN	NaN	3850.0	NaN
Delhi		NaN	10262.0	NaN	NaN	NaN
Kolkata	9345.0		NaN	NaN	NaN	NaN
Mumbai		NaN	NaN	3342.0	NaN	NaN

# pd.cut()



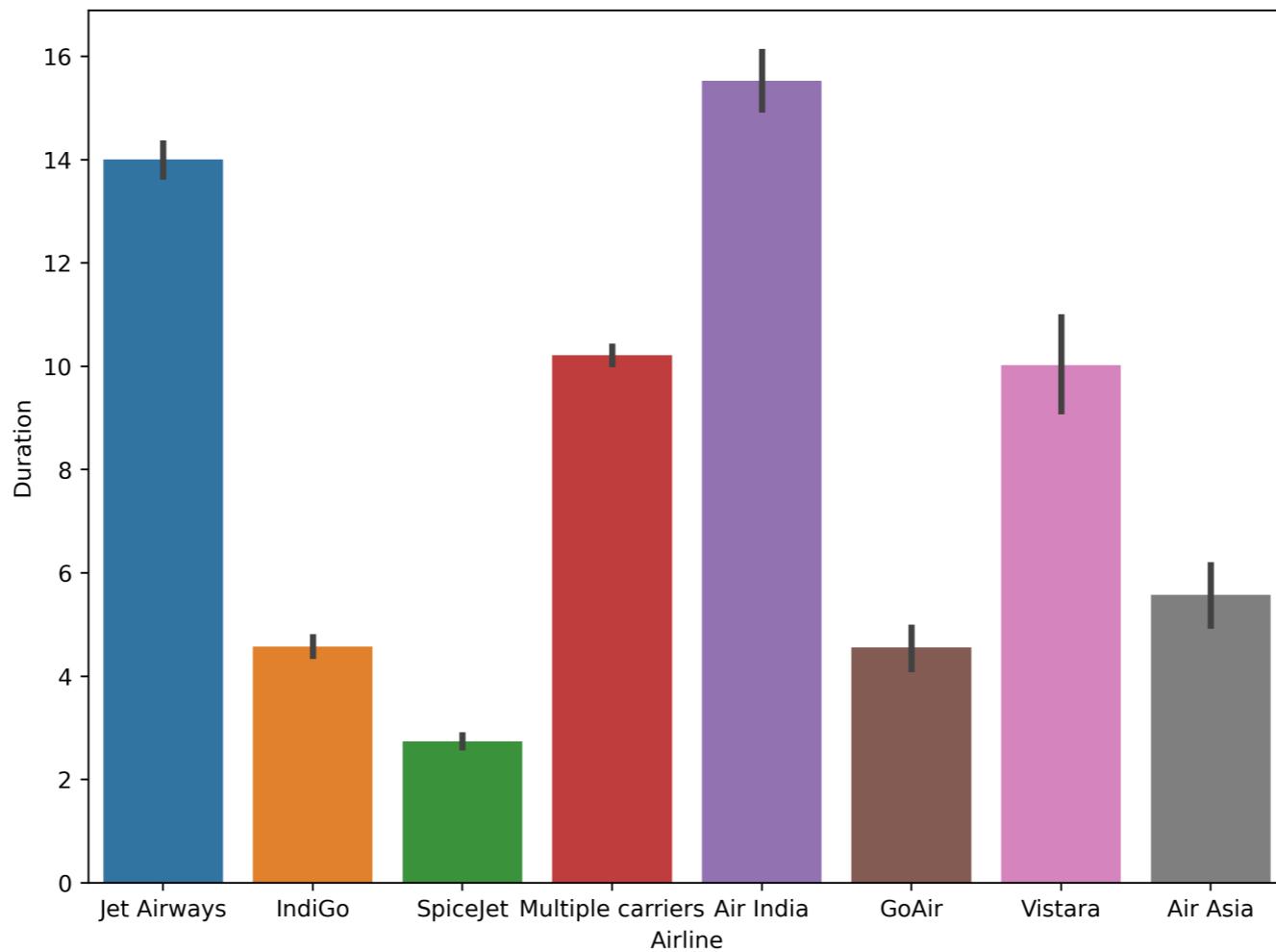
```
planes["Price_Category"] = pd.cut(planes["Price"],  
                                 labels=labels,  
                                 bins=bins)
```

# Data snooping



# Generating hypotheses

```
sns.barplot(data=planes, x="Airline", y="Duration")  
plt.show()
```



# Next steps

- **Sampling in Python**
- **Hypothesis Testing in Python**
- **Supervised Learning with scikit-learn**

# Congratulations!

EXPLORATORY DATA ANALYSIS IN PYTHON