**Considerations for categorical data**

Let's see how we convert exploratory data analysis into action! We'll start by looking at class frequencies.

**Why perform EDA?**

Recall that EDA is performed for a variety of reasons, like detecting patterns and relationships in data, generating questions or hypotheses, or to prepare data for machine learning models.

1. 1 Image credit: https://unsplash.com/@simonesecci

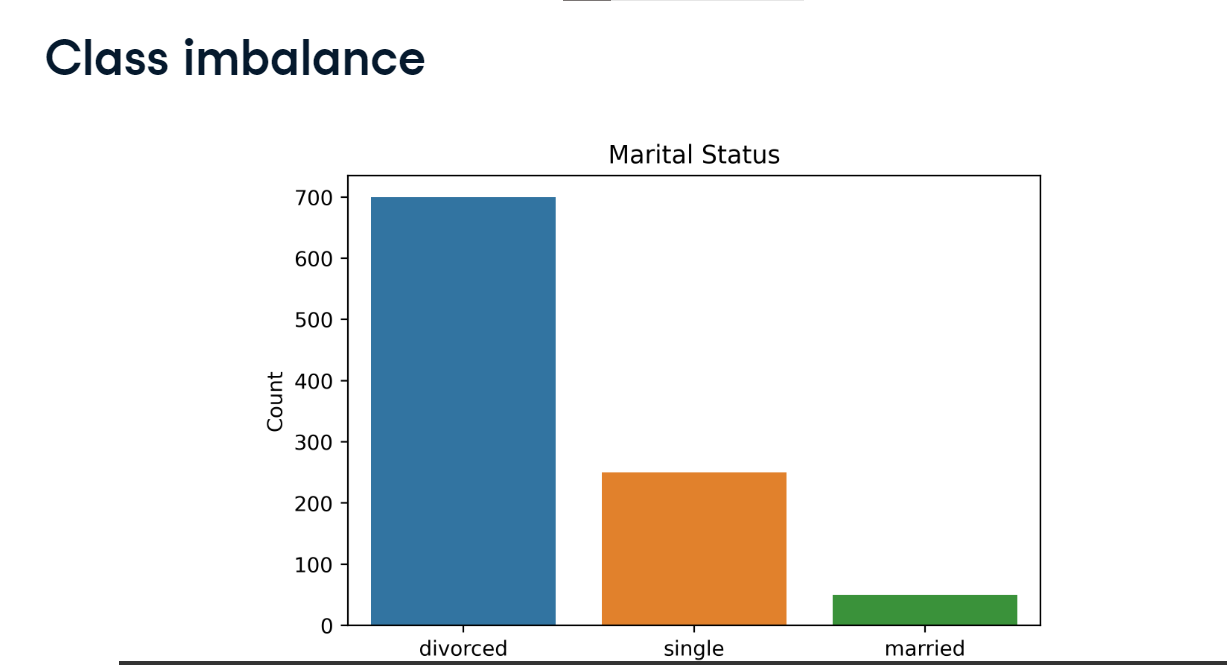
**Representative data**

There's one requirement our data must satisfy regardless of our plans after performing EDA - it must be representative of the population we wish to study. For example, if we collect data with the aim of analyzing the relationship between education level and income in the USA, then we would need to collect this data from adults residing in the USA, and can't rely on data from residents of France.

1. 1 Image credits: https://unsplash.com/@cristina\_glebova; https://unsplash.com/@nimbus\_vulpis

**Categorical classes**

With categorical data, one of the most important considerations is about the representation of classes, which is another term for labels. For example, say we collect data on people's attitudes to marriage. As part of our data collection we find out their marital status, with the classes including single, married, and divorced.



**Class imbalance**

When we perform EDA we realize only 50 people were married, while 700 were divorced and the remaining 250 were single. Do we think that this sample accurately represents the general public's opinion about marriage? Are divorced people more likely to have a negative view towards marriage? This is an example of class imbalance, where one class occurs more frequently than others. This can bias results, particularly if this class does not occur more frequently in the population.

**Class frequency**

We've been counting the number of observations per class using pandas dot-value\_counts, like here, where we see how many flights went to different destinations in our planes dataset.

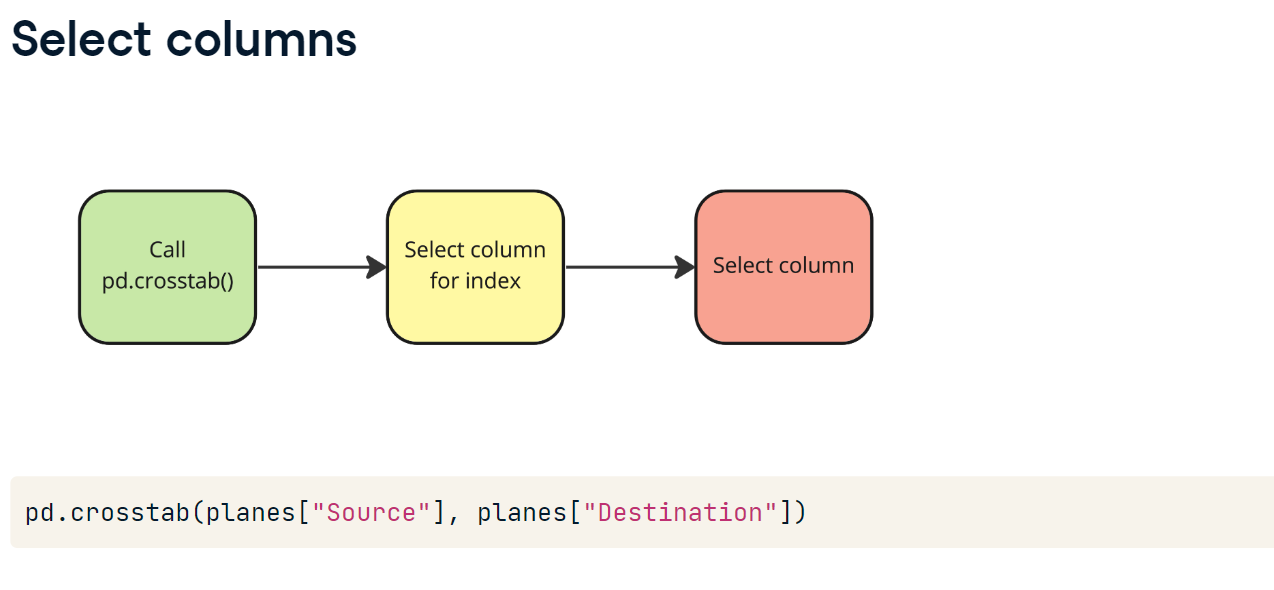
**Relative class frequency**

Say that we know 40 percent of internal Indian flights go to Delhi. We can use value\_counts method again, but this time set the normalize keyword argument equal to True. This returns the relative frequencies for each class, showing that Delhi only represents 11-point-eight-two percent of destinations in our dataset. Again, this could suggest that our data is not representative of the population - in this case, internal flights in India.



**Cross-tabulation**

Another method for looking at class frequency is cross-tabulation, which enables us to examine the frequency of combinations of classes. Let's look at flight route frequencies. We'll start by calling pandas-dot-crosstab function.



**Select index**

Next we select the column to use as the index for the table, in this case the Source.

**Select columns**

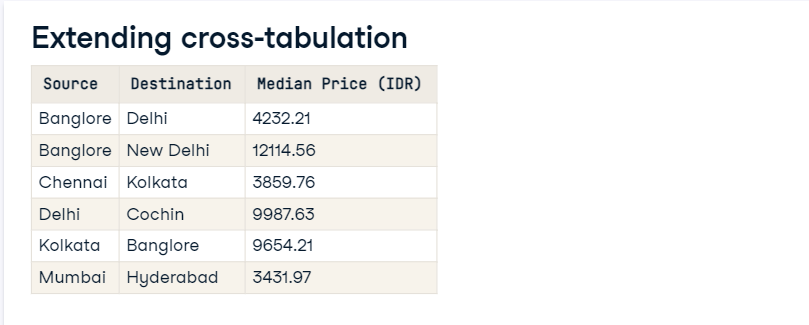
Lastly, we pass the Destination. Values in this column will become the names of the columns in the table, and the values will be the count of combined observations.

**Cross-tabulation**

We see the most popular route is from Delhi to Cochin, making up 4318 flights.

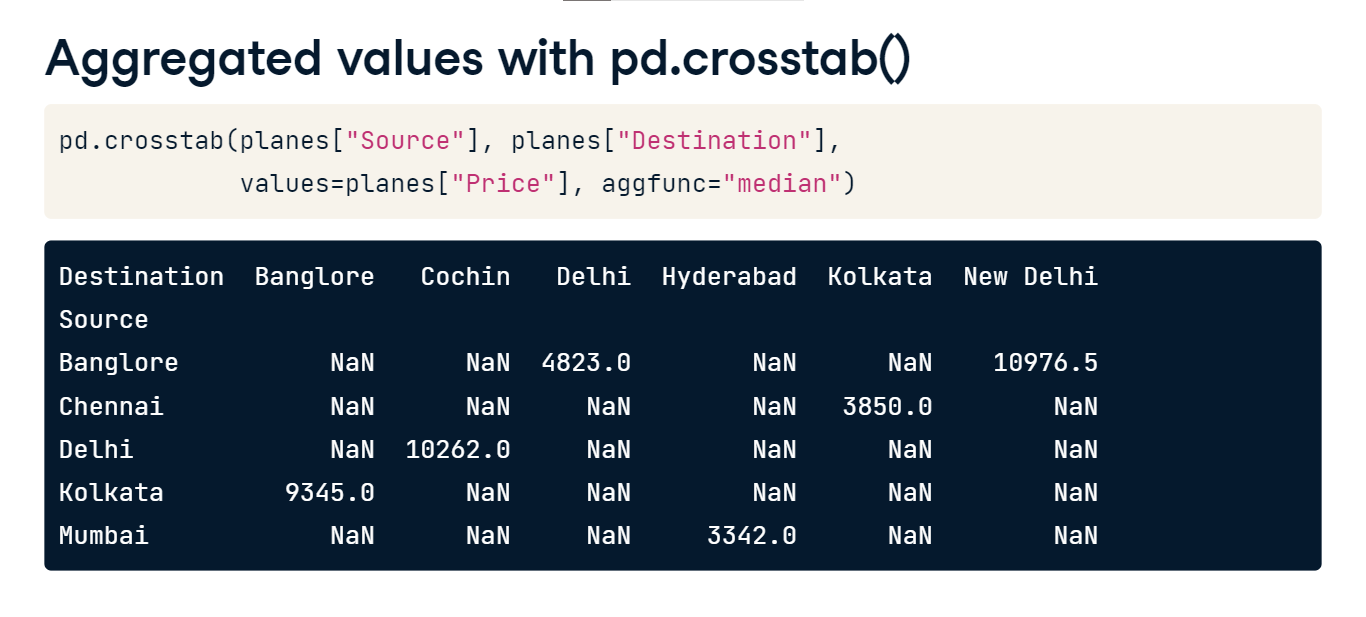
**Extending cross-tabulation**

Say we know the median price for all internal flight routes in India. Here they are for the routes in our dataset, measured in Indian Rupees. We can calculate the median price for these routes in our DataFrame, and compare the difference to these expected values.



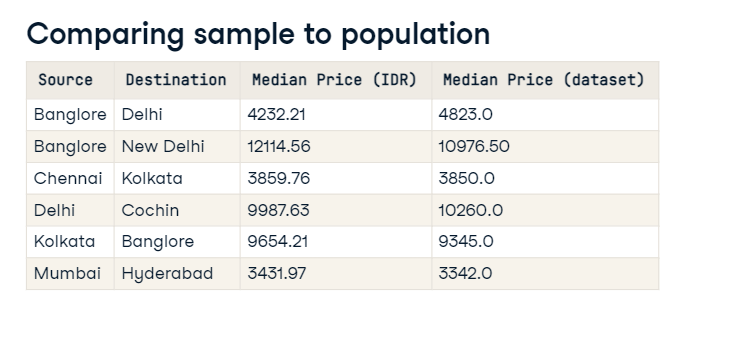
**Aggregated values with pd.crosstab()**

We do this by adding two keyword arguments to pd-dot-crosstab. We pass the Price column to the values argument, and use aggfunc to select what aggregated calculation we want to perform. We can pass a summary statistic as a string, in this case setting it equal to median. The results show median values for all possible routes in the dataset.



**Comparing sample to population**

Comparing our prices with the expected values, most are similar. However, routes from Banglore to Delhi and New Delhi are more expensive in our dataset, suggesting they aren't representative of the population.



**Generating new features**

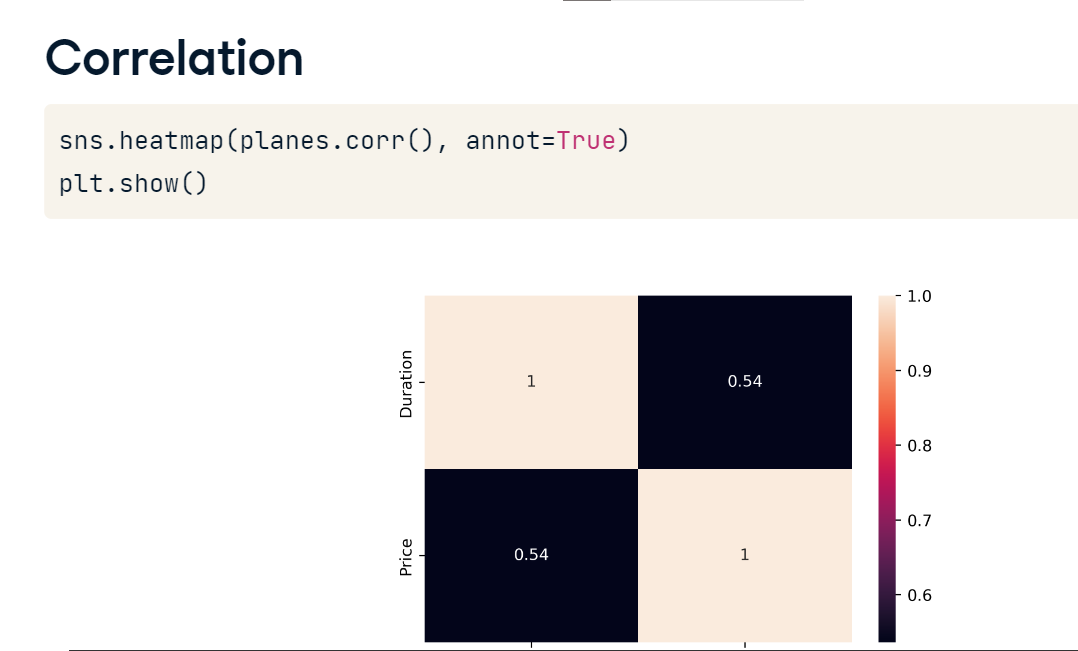
Sometimes the format of our data can limit our ability to detect relationships or inhibit the potential performance of machine learning models. One method to overcome these issues is to generate new features from our data!

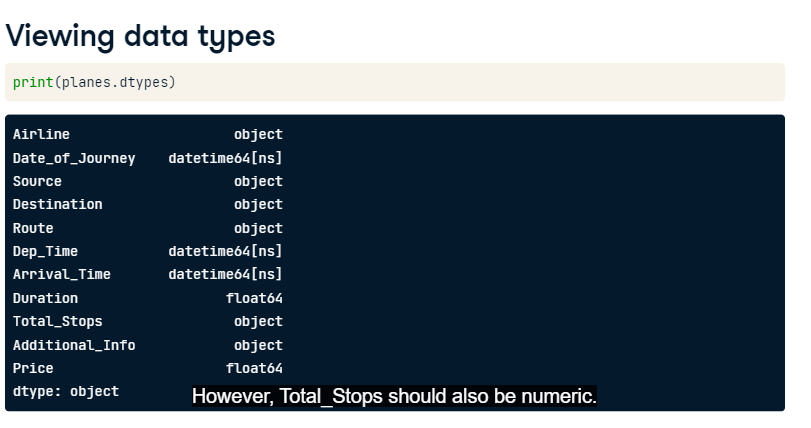
**Correlation**

Checking correlation with a heatmap, we see a moderate positive correlation between Price and Duration, but it looks like those are the only numeric variables in our dataset.

**Viewing data types**

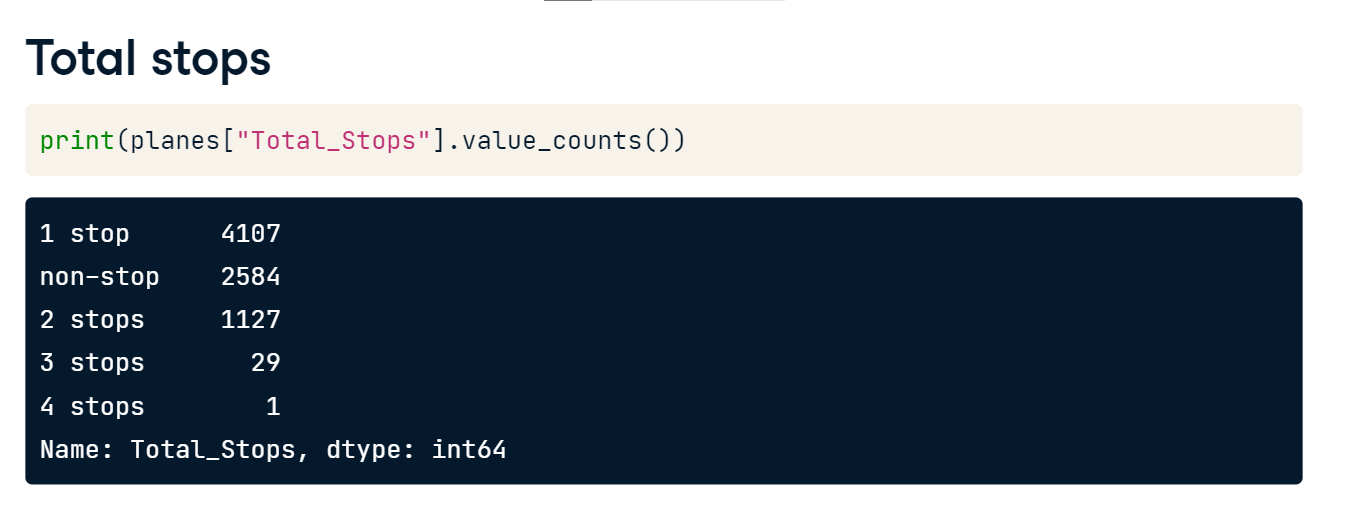
Viewing the data types confirms this is the case. However, Total\_Stops should also be numeric.



****

**Total stops**

Viewing the value\_counts, we see we need to remove string characters, and change non-stop to zero, before converting the data type to integer.



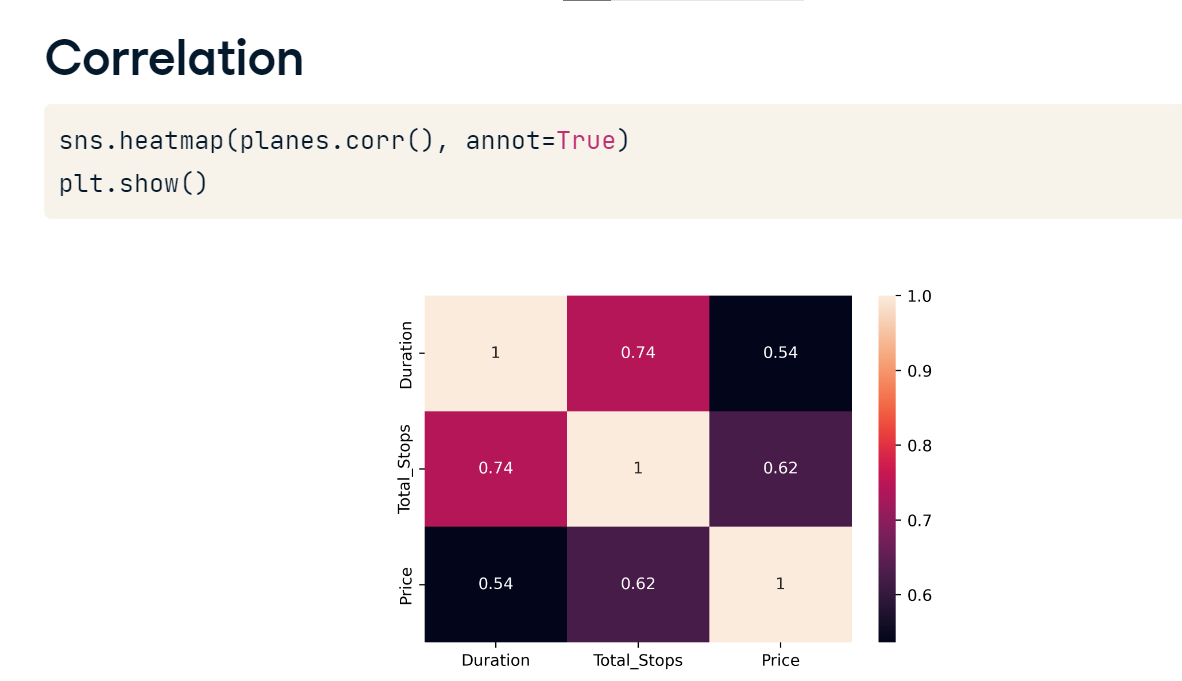
**Cleaning total stops**

We use the string-dot-replace method to first remove " stops", including the space, so that flights with two, three, or four stops are ready to convert. Next we clean flights with one stop. Lastly, we change "non-stop" to "0", then set the data type to integer.



**Correlation**

Unsurprisingly, Total\_Stops is strongly correlated with Duration. What is interesting is that Total\_Stops and Price are more strongly correlated than Duration is with Price! Let's see what else we can find out!

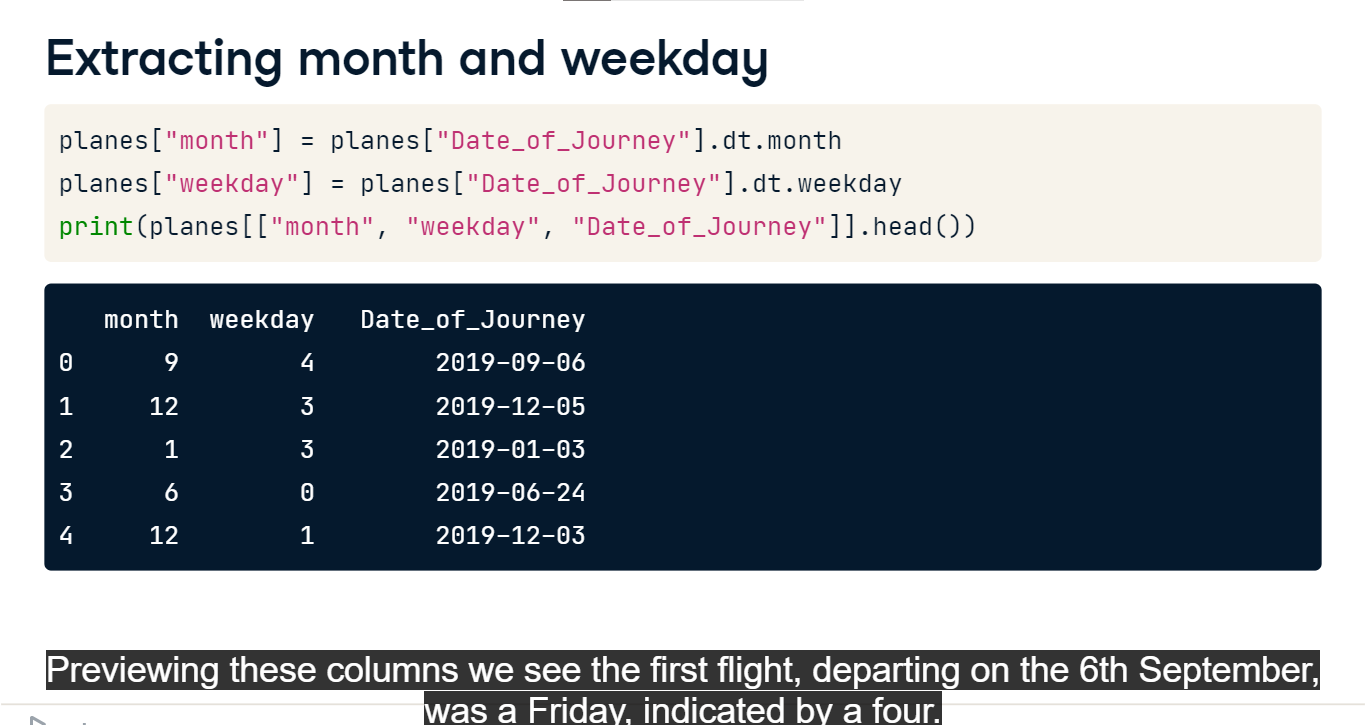


**Dates**

Rechecking our data types, notice that there are three datetime variables - Date\_of\_Journey, Dep\_Time, and Arrival\_Time.

**Extracting month and weekday**

We know how to extract attributes from datetime values, so we can see if these offer any insights into pricing. To start, let's look at Date\_of\_Journey. If we think prices vary per month, it's worth using this attribute - we create it as a column in our DataFrame. Perhaps prices might also differ depending on the day of the week? Let's grab that using the dt-dot-weekday attribute. It returns values of zero, representing Monday, through to seven, for Sunday. Previewing these columns we see the first flight, departing on the 6th September, was a Friday, indicated by a four.



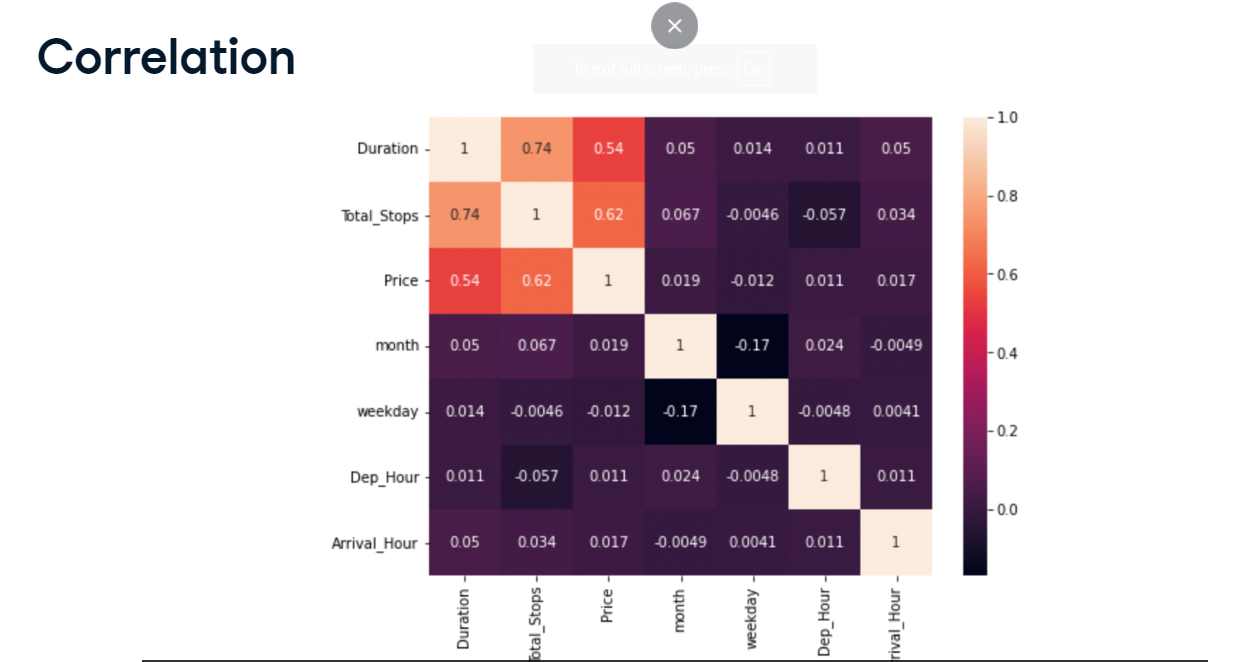
**Departure and arrival times**

We might wonder if people tend to pay more to depart or arrive at more convenient times. We extract the hour of departure and arrival from those respective columns too.



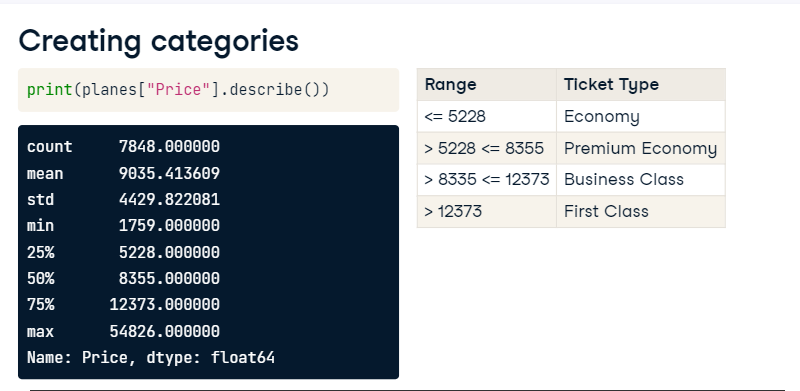
**Correlation**

Because they are numeric, we can calculate correlation between these new datetime features and other variables. Re-plotting our heatmap, unfortunately there aren't any new strong relationships. But we wouldn't have known this if we hadn't generated these features.



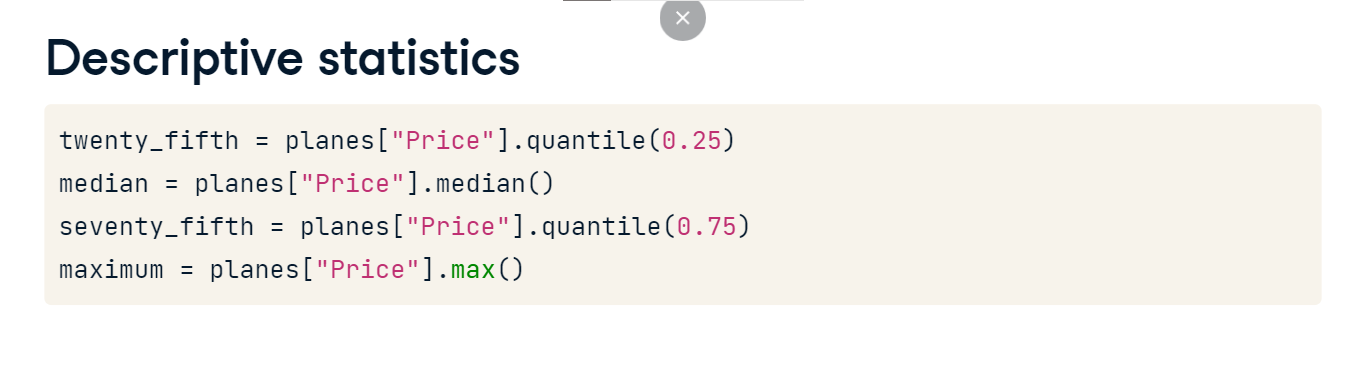
**Creating categories**

There's one more technique we can use to generate new features. We can group numeric data and label them as classes. For example, we don't have a column for ticket type. We could use descriptive statistics to label flights as economy, premium economy, business class, or first class, based on prices within specific ranges, or bins.



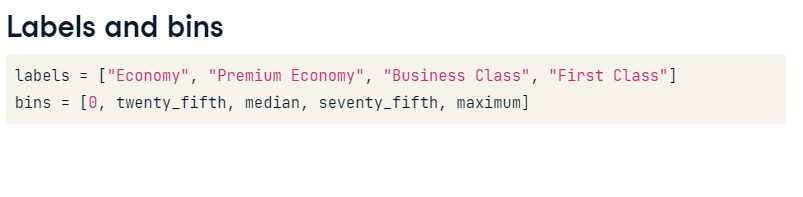
**Descriptive statistics**

We'll split equally across the price range using quartiles. We first store the 25th percentile using the quantile method. We get the 50th percentile by calling the median. Next we get the 75th percentile, and lastly, we store the maximum value.



**Labels and bins**

We create the labels, in this case our ticket types, and store as a list. Next, we create the bins, a list starting from zero and including our descriptive statistic variables.

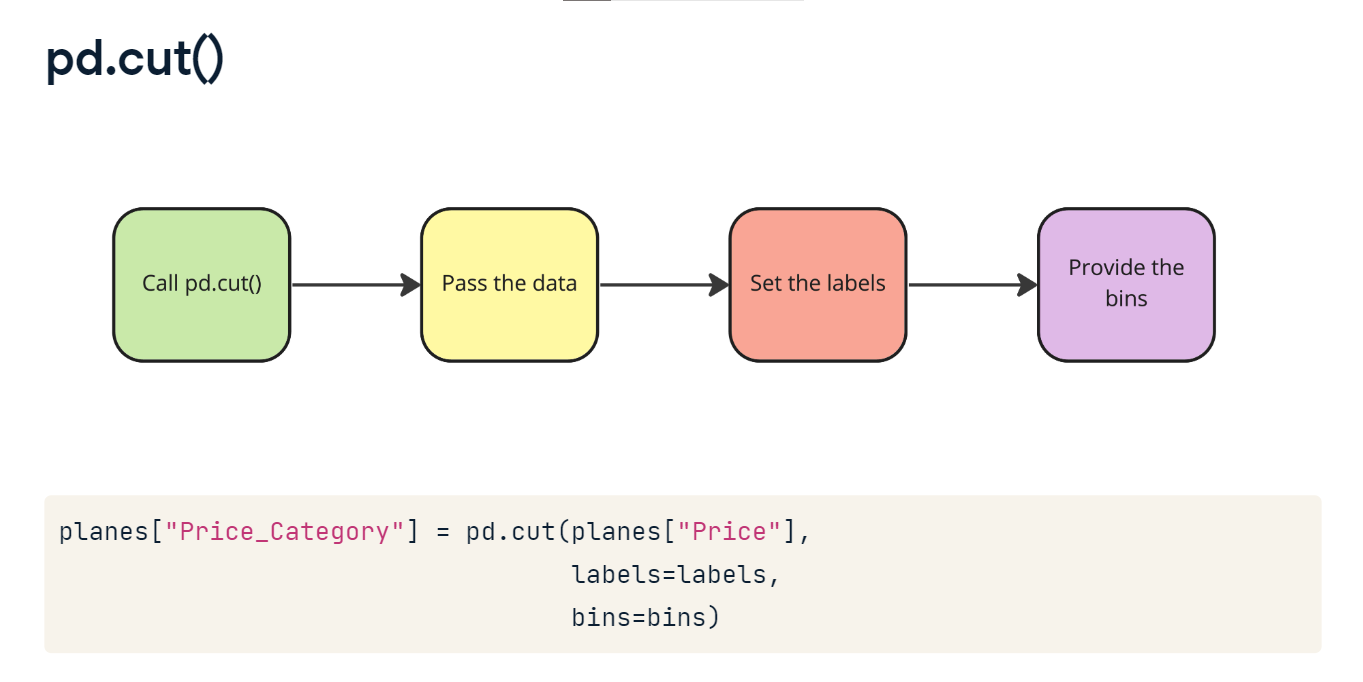


**pd.cut()**

We now call the pd-dot-cut function,

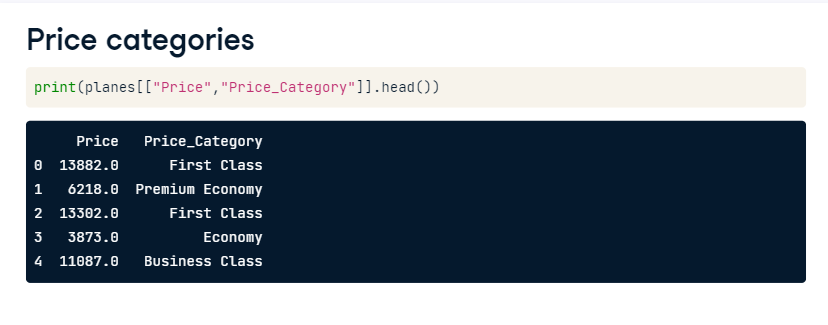
passing our Price column,

setting the labels argument equal to our labels variable,and the bins argument equal to our bins.



**Price categories**

Previewing the Price and Price\_Category columns, we see the mapping has been successfully applied!



**Price category by airline**

We can plot the count of flights in different categories per airline by passing our new column to the hue argument when calling sns-dot-countplot.

Looks like Jet Airways has the largest number of "First Class" tickets, while most of IndiGo and SpiceJet's flights are "Economy".



**1. Generating hypotheses**

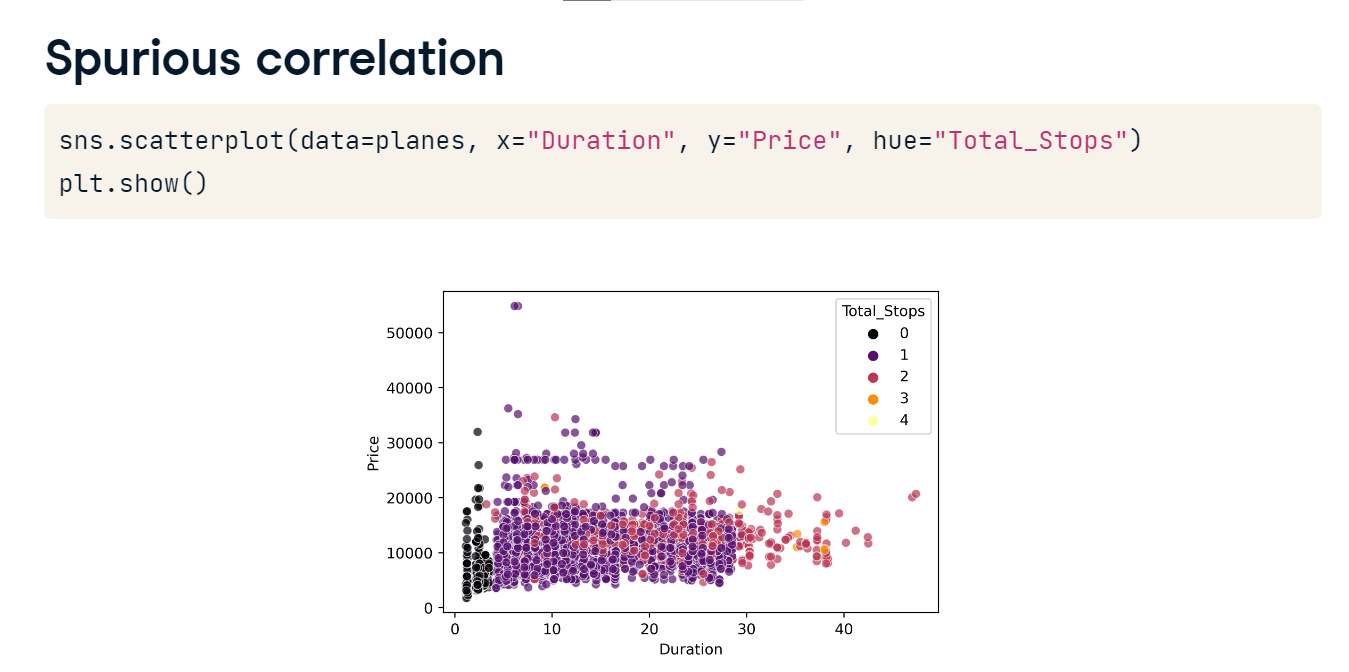
00:00 - 00:07

Generating hypotheses is a fundamental task for data scientists. Let's look at how and when this is done!

**2. What do we know?**

00:07 - 00:27

It's reasonable to feel like we have a good idea about our planes dataset at this point, right? We've explored our data extensively and even generated new features to get new insights! We know that a large proportion of Jet Airways' tickets are expensive, as we labeled them as First Class!



**3. What do we know?**

00:27 - 00:36

We also know that Duration, Total\_Stops, and Price are all moderately correlated, but no other meaningful relationships exist.

**4. Spurious correlation**

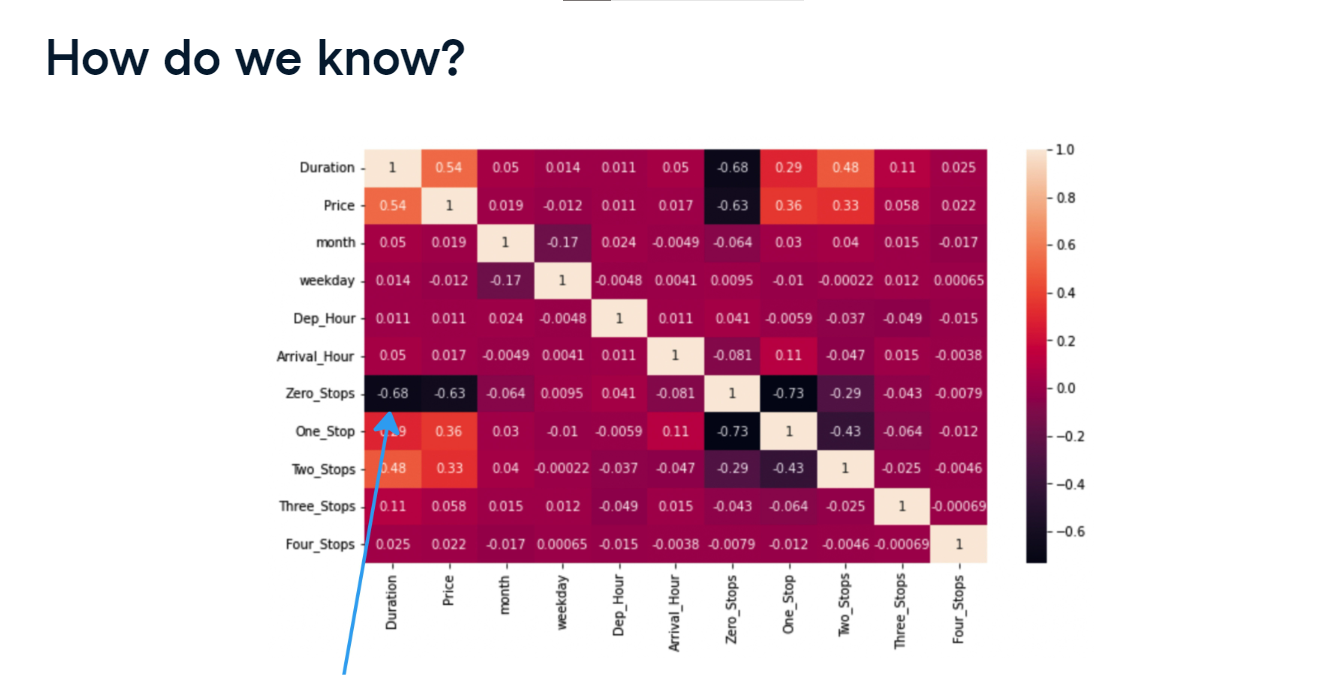
00:36 - 01:01

But if we generate a scatter plot of Price versus Duration, factoring Total\_Stops, it looks like Total\_Stops largely depend on Duration. This is an example of a spurious correlation - we might think that Total\_Stops is correlated with Price, but in fact its just Duration that is correlated and Total\_Stops mostly maps to Duration ranges!

**5. How do we know?**

01:01 - 01:16

Also, if we split out the number of stops to look at correlation individually, it looks like zero stops has a strong negative correlation with price, but there's no meaningful relationship for journeys with three of four stops!



**6. What is true?**

01:16 - 02:00

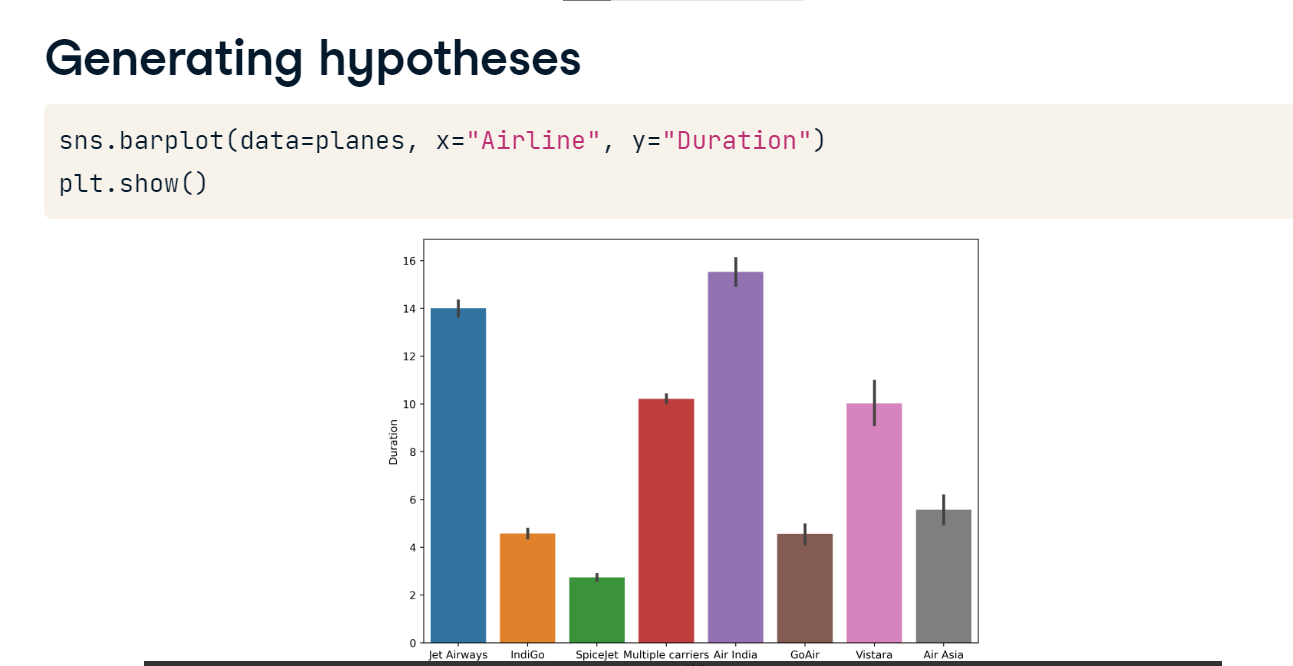
When performing EDA, the question we should ask is how do we know what we are observing is true? For example, if we collected new data on flights from a different time period, would we observe the same results? To make conclusions regarding relationships, differences, and patterns in our data, we need to use a branch of statistics called Hypothesis Testing. This involves the following steps before we even start collecting data: coming up with a hypothesis, or question, and specifying a statistical test that we will perform in order to reasonably conclude whether the hypothesis was true or false.

1. 1 Image credit: https://unsplash.com/@markuswinkler

**7. Data snooping**

02:00 - 03:04

Let's imagine we work for an agency regulating airlines, so we have our planes data available as part of our day-to-day work without any specific questions in mind. We might be thinking, well, we have all this data, so why not just come up with questions and run some tests now? But we didn't collect the data with the aim of answering these questions. Plus, we've already looked at the data extensively and generated new features, so we might be bias and generate hypotheses that we are confident exist to prove ourselves right! We could also be tempted to run lots of tests, since we have lots of data. The acts of excessive exploratory analysis, the generation of multiple hypotheses, and the execution of multiple statistical tests are collectively known as data snooping, or p-hacking. Chances are, if we look at enough data and run enough tests, we will find a significant result.



**8. Generating hypotheses**

03:04 - 03:21

So how do we generate hypotheses? We perform some EDA! Say we think that, on average, Jet Airways flights last longer than SpiceJet. We can create a bar plot, which shows us the mean duration per Airline.

**9. Generating hypotheses**

03:21 - 03:32

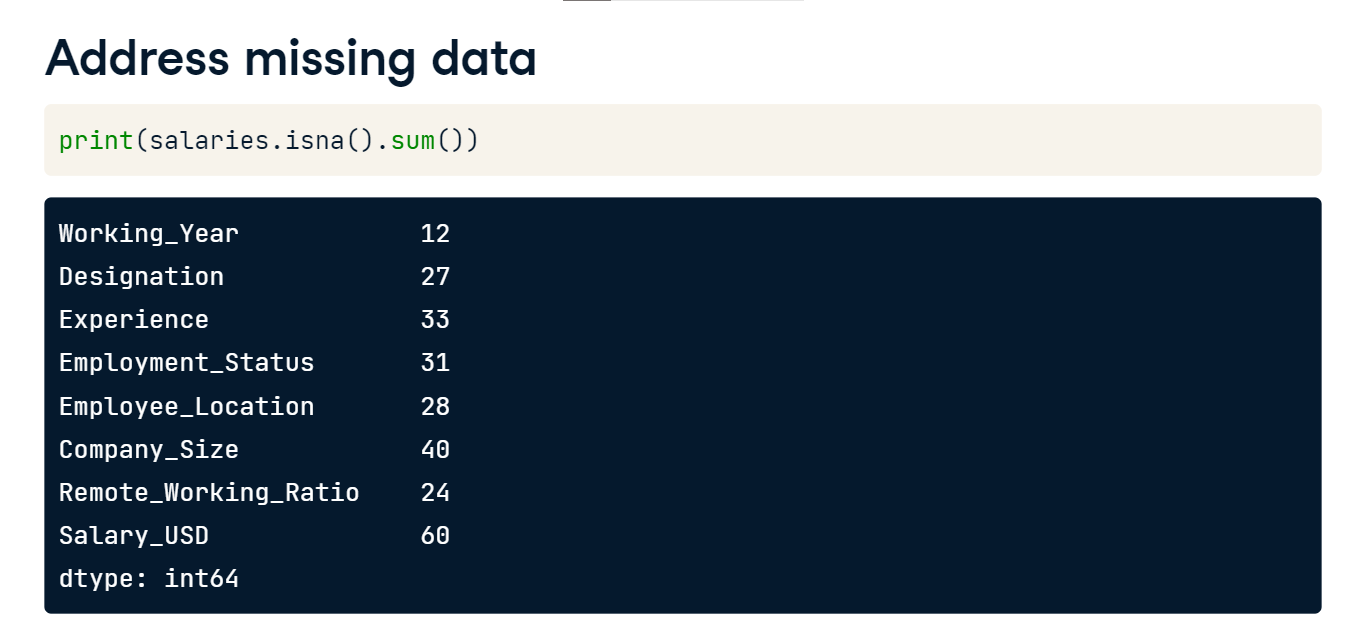
Or we might have a hunch that flights to New Delhi are more expensive than other destinations on average. Again, we can plot the data to see if this seems to be the case.

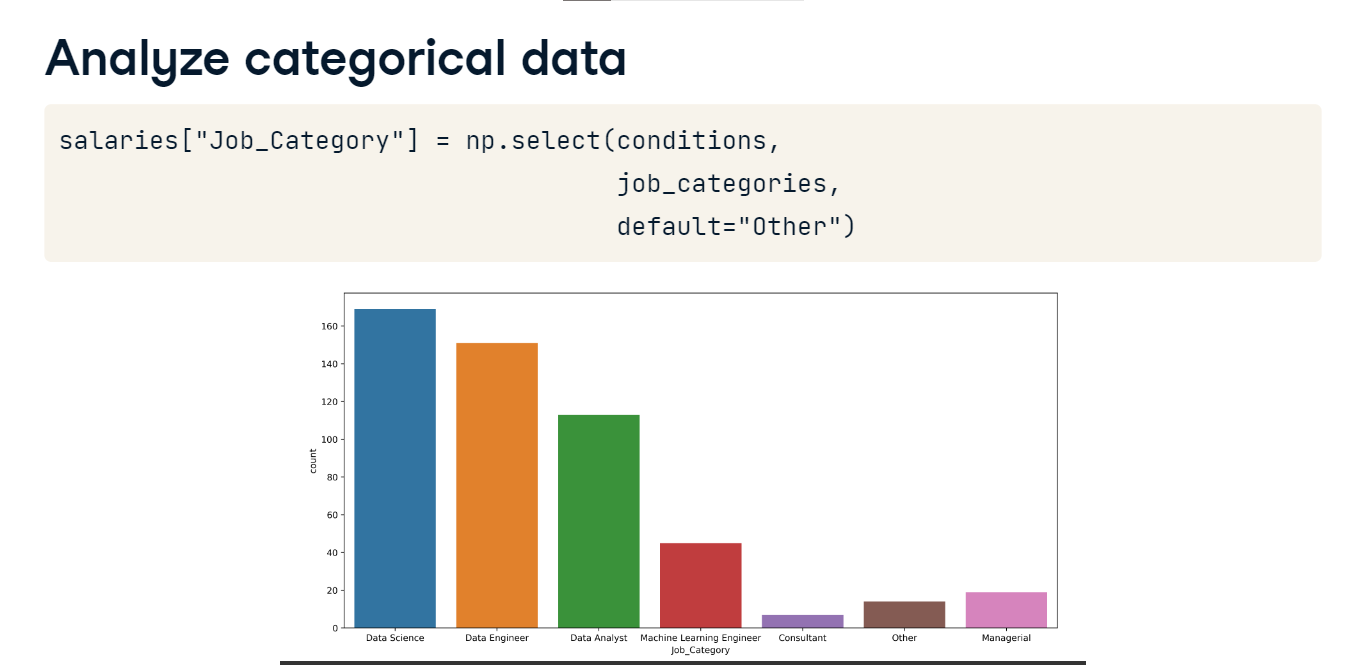
**10. Next steps**

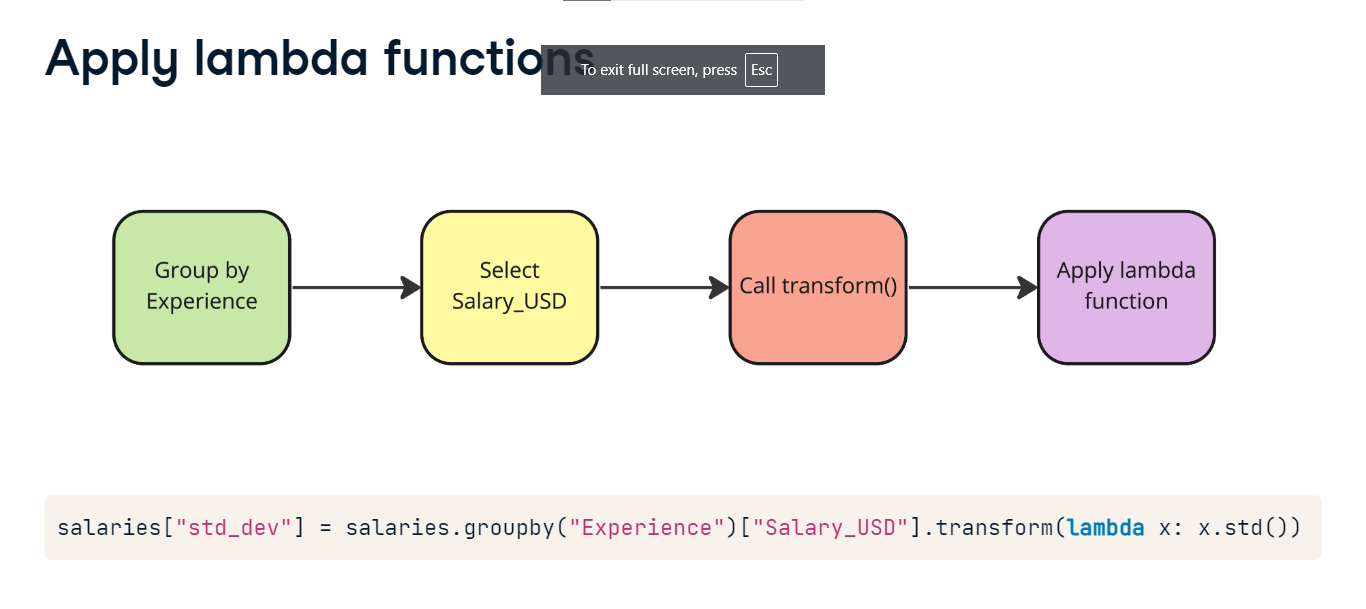
03:32 - 03:58

From there, we need to design our experiment. This involves many steps such as choosing a sample, calculating how many data points we need, and deciding what statistical test to run. The steps involved in this process are outside the scope of this course, but hopefully we now have a sense of the advantages, limitations, and overall remit of exploratory data analysis in a data science workflow!









**Congratulations**

Congratulations on completing the course, you've covered a lot!

**2. Inspection and validation**

00:05 - 00:10

You started off by learning how to inspect and validate data,

**3. Aggregation**

00:10 - 00:14

before performing aggregation and calculating summary statistics!

**4. Address missing data**

00:14 - 00:17

You saw how to check for missing values.

**5. Address missing data**

00:17 - 00:25

You then identified strategies to deal with it, including dropping missing values, and imputation!

**6. Analyze categorical data**

00:25 - 00:28

You discovered how to create categories from strings,

**7. Apply lambda functions**

00:28 - 00:37

use lambda functions to conditionally calculate summary statistics based on categories and add values into the original DataFrame,

**8. Handle outliers**

00:37 - 00:39

and deal with outliers!

**9. Patterns over time**

00:39 - 00:45

You progressed to examining relationships, including patterns over time,

**10. Correlation**

00:45 - 00:47

correlation between variables,

**11. Distributions**

00:47 - 00:49

and interpreting distributions!

**12. Cross-tabulation**

00:49 - 00:53

In the final chapter you learned the benefits of cross-tabulation,

**13. pd.cut()**

00:53 - 00:57

generated new features using pd-dot-cut,

**14. Data snooping**

00:57 - 01:00

and saw the impact of data snooping!

**15. Generating hypotheses**

01:00 - 01:08

You finished by identifying the limits of EDA and the next step of the data science workflow, hypothesis testing.

**16. Next steps**

01:08 - 01:21

Now you understand EDA, you may wish to explore some courses that build on the concepts in this course, such as the steps involved in hypothesis testing, or supervised learning, which is a form of machine learning!

