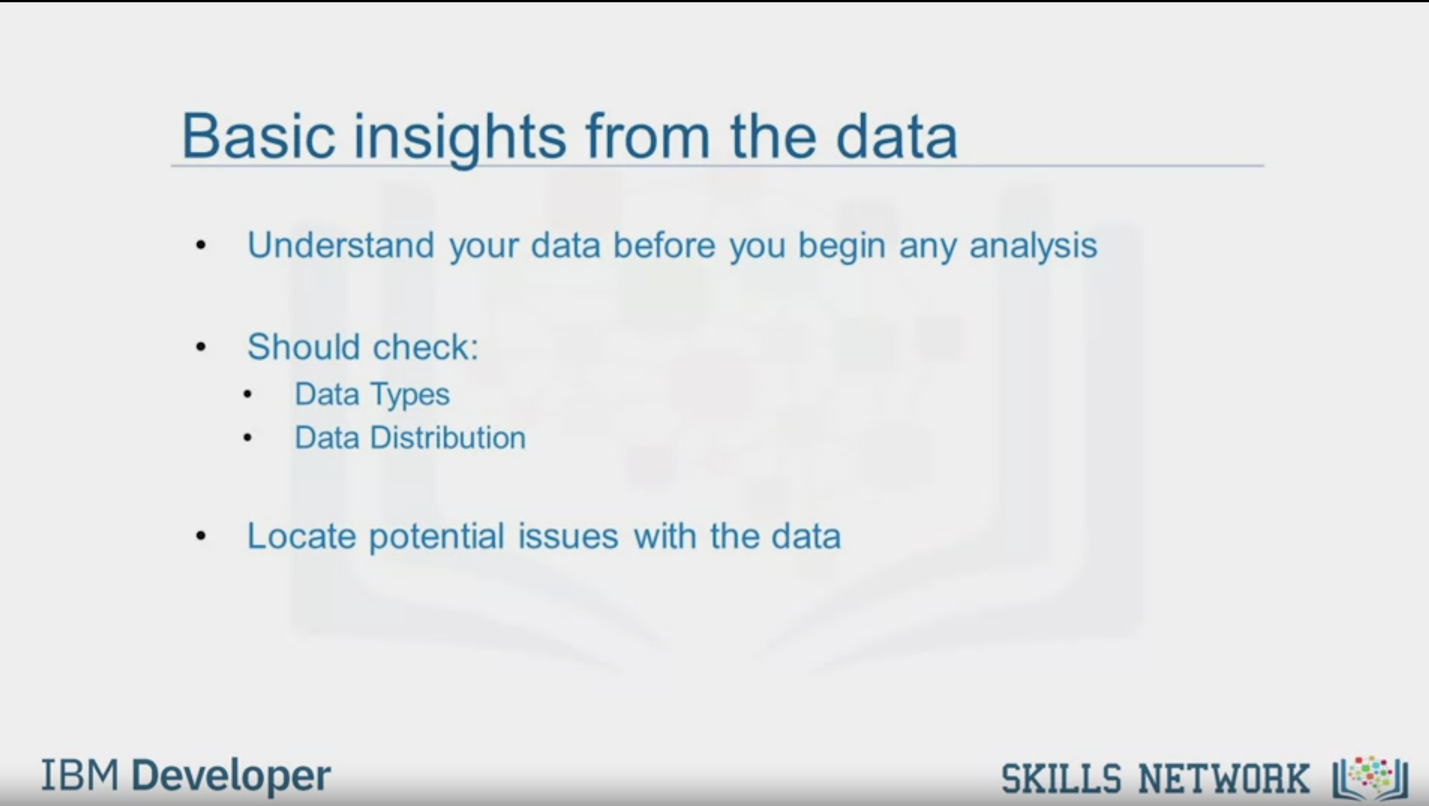


Getting Started Analyzing Data

in Python

IBM Developer

SKILLS NETWORK



Basic insights from the data

Understand your data before you begin any analysis

Should check:

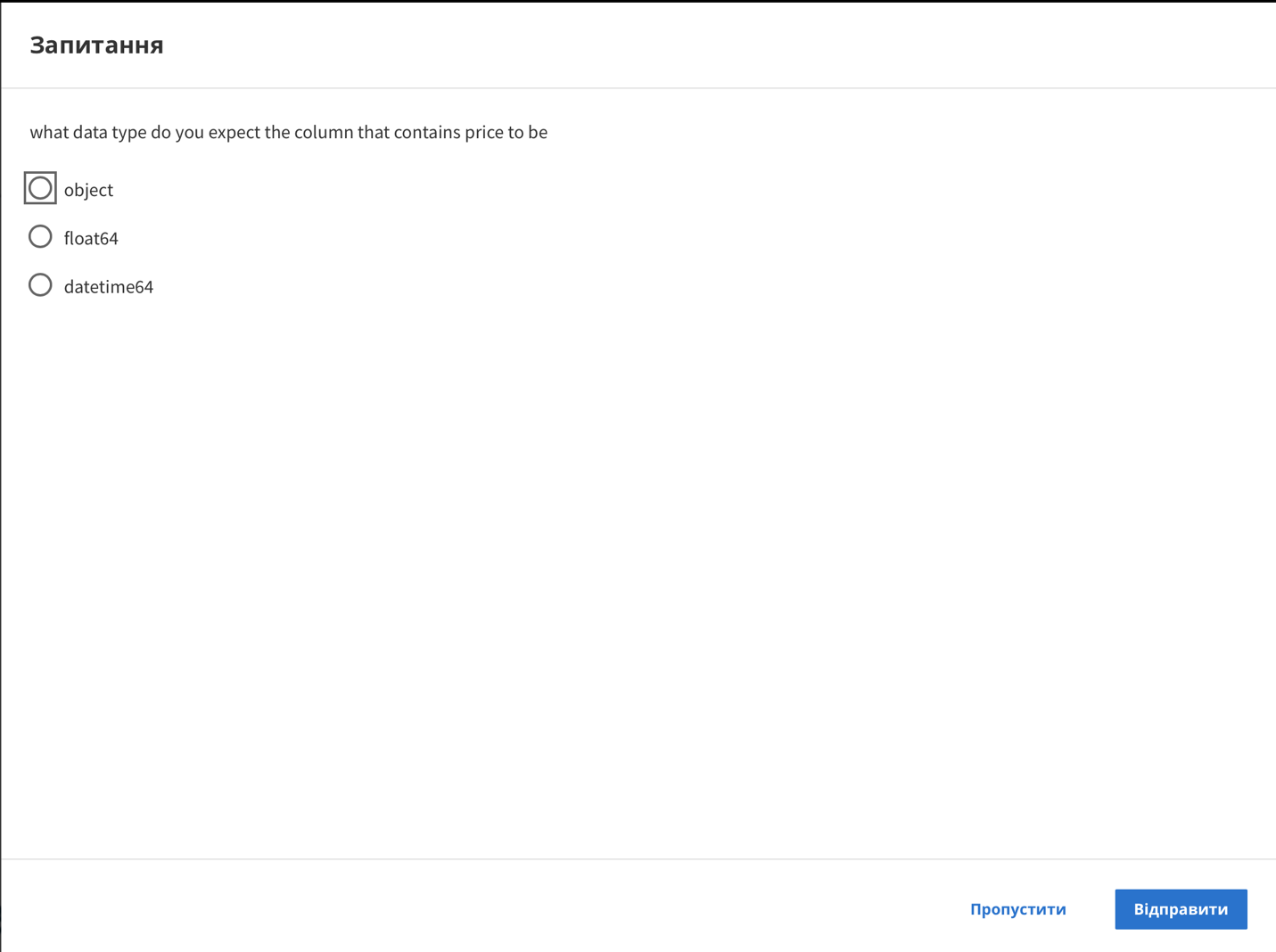
Data Types

Data Distribution

Locate potential issues with the data

IBM Developer

SKILLS NETWORK



Question

what data type do you expect the column that contains price to be

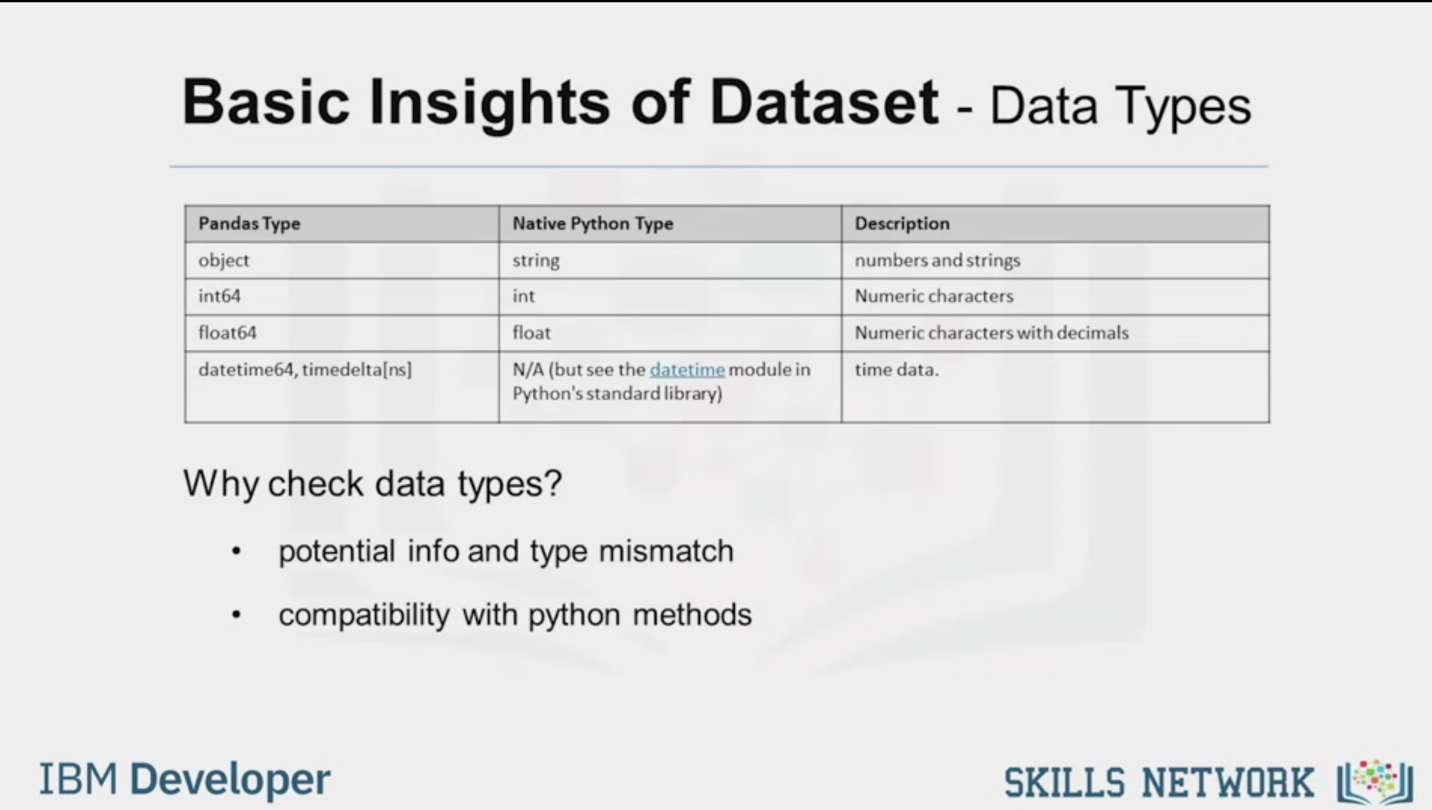
object

float64

datetime64

Skip

Send



Basic Insights of Dataset - Data Types

|  |  |  |
| --- | --- | --- |
| Pandas Type | Native Python Type | Description |
| object | string | numbers and strings |
| int64 | int | Numeric characters |
| float64 | float | Numeric characters with decimals |
| datetime64, timedelta[ns] | N/A (but see the datetime module in  Python's standard library) | time data. |

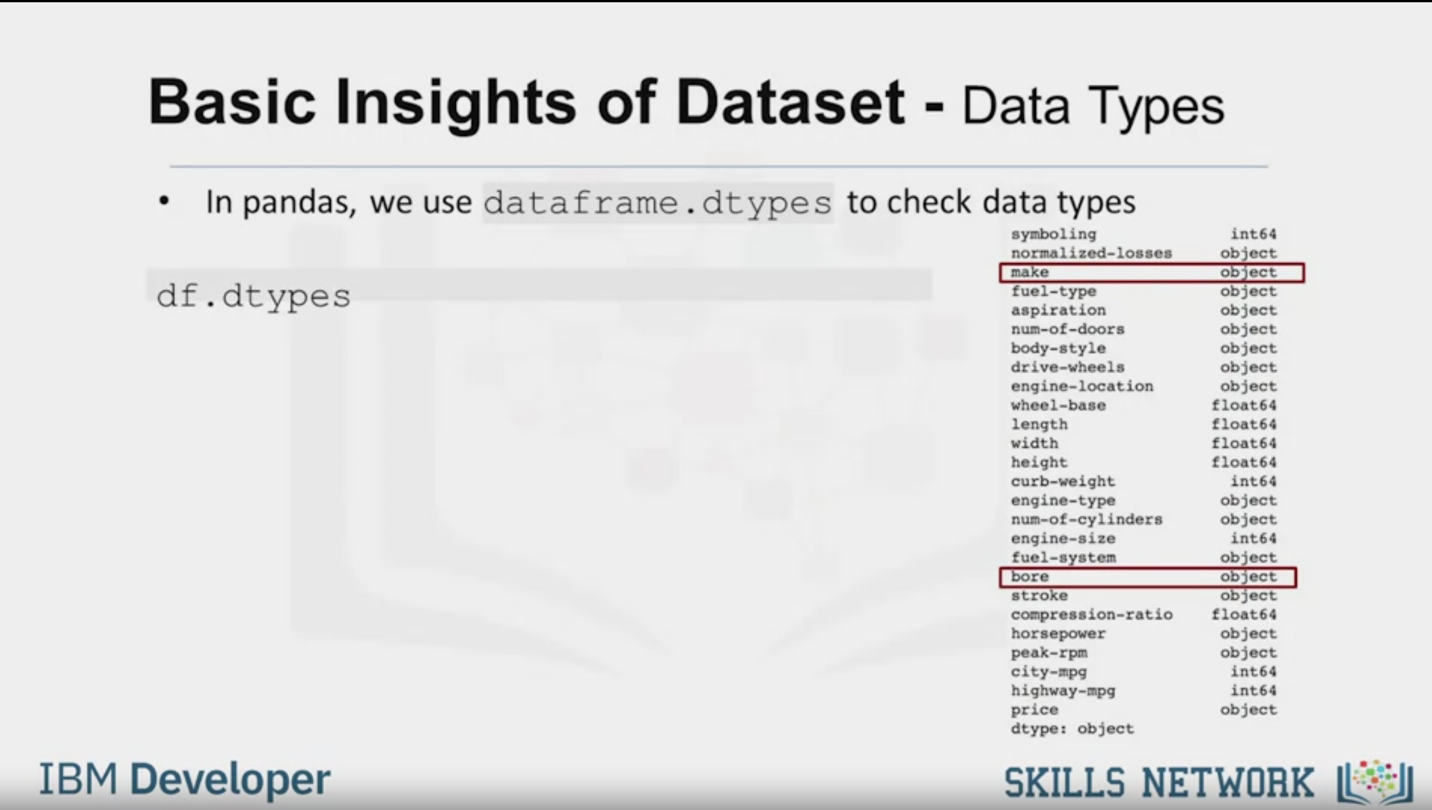
Why check data types?

potential info and type mismatch

compatibility with python methods

IBM Developer

SKILLS NETWORK



Basic Insights of Dataset - Data Types

In pandas, we use dataframe.dtypes to check data types

df.dtypes

symboling int64

normalized-losses object

make object

fuel-type object

aspiration object

num-of-doors object

body-style object

drive-wheels object

engine-location object

wheel-base float64

length float64

width float64

height float64

curb-weight int64

engine-type object

num-of-cylinders object

engine-size int64

fuel-system object

bore object

stroke object

compression-ratio float64

horsepower object

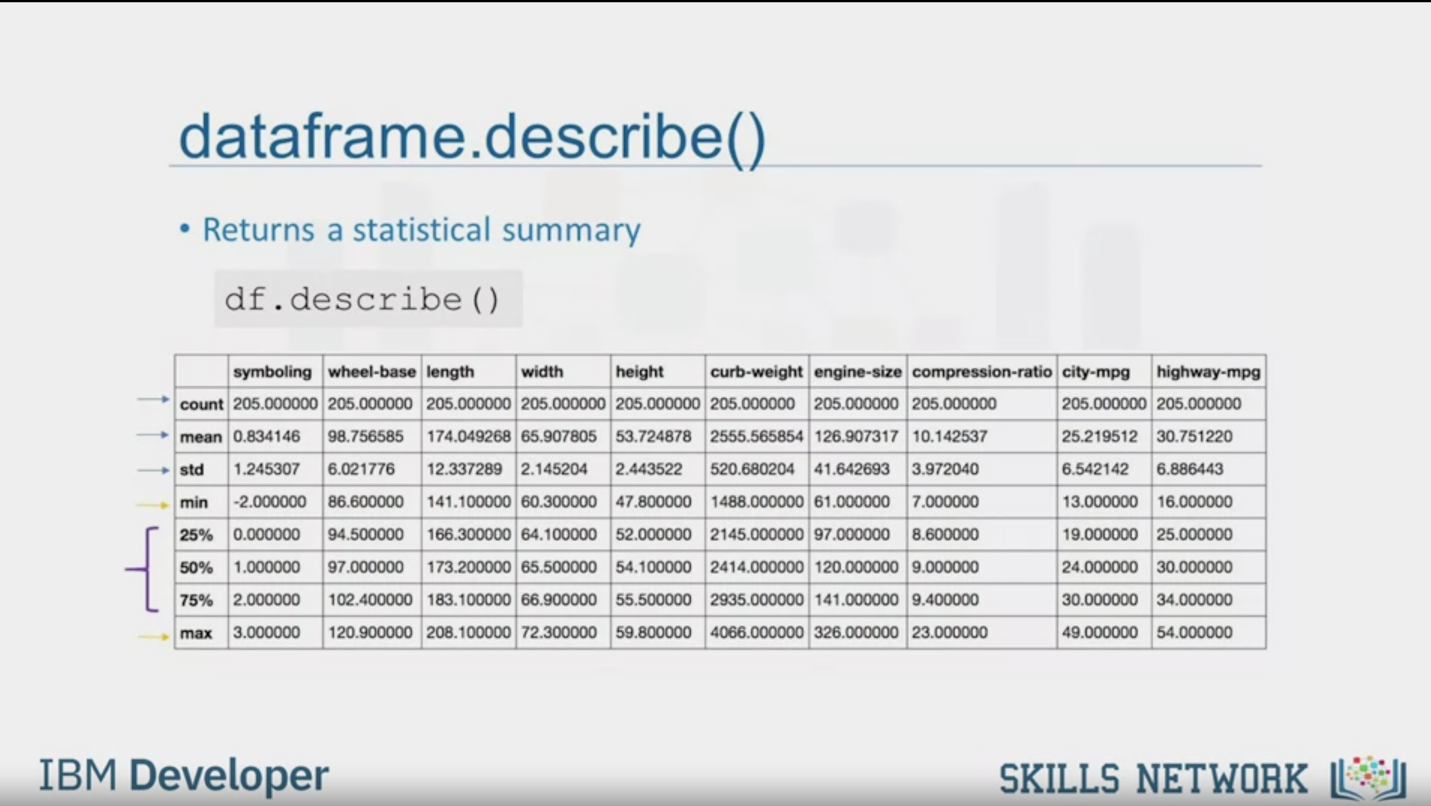
peak-rpm object

city-mpg int64

highway-mpg int64

price object

dtype: object



dataframe.describe()

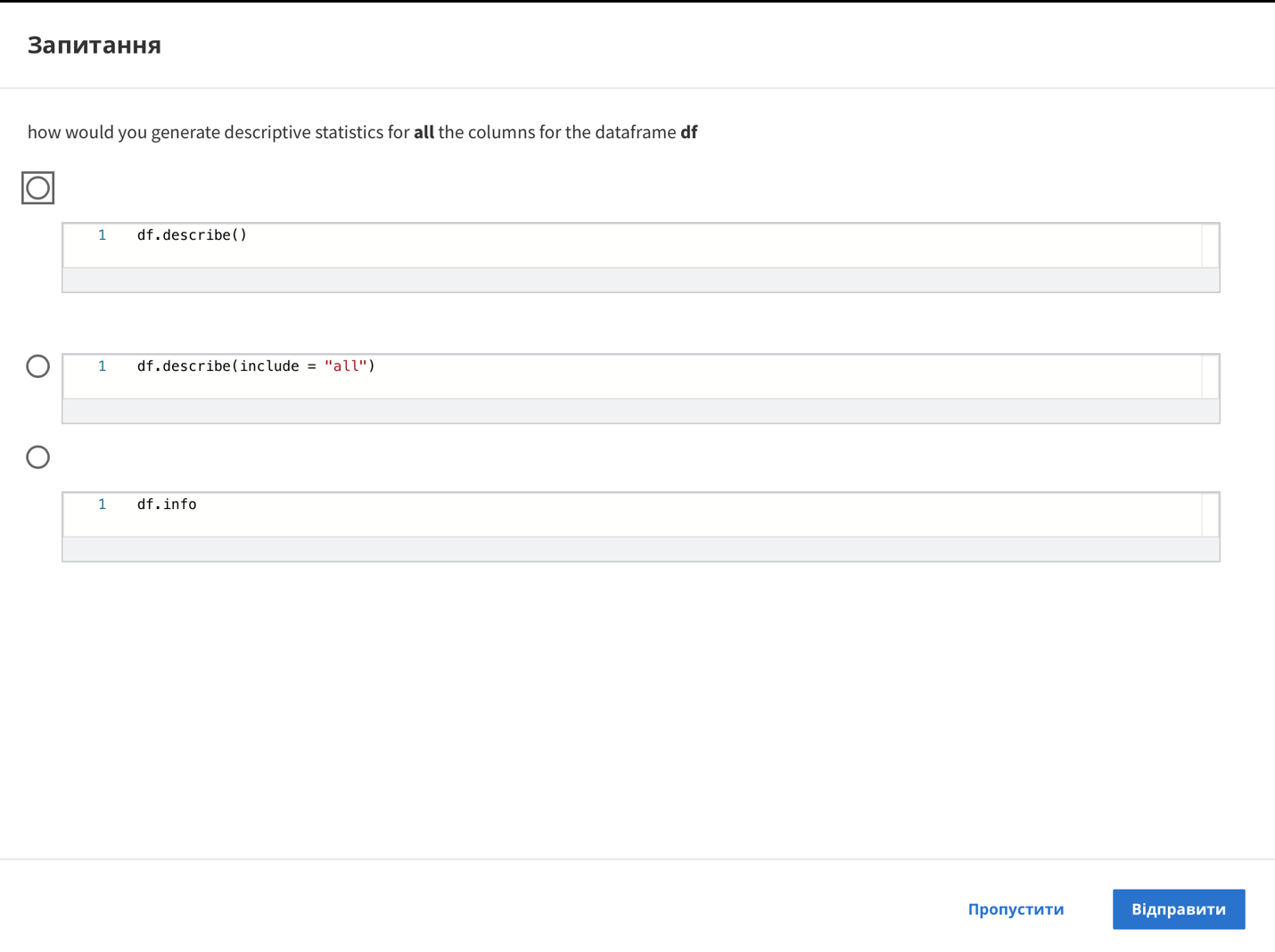
• Returns a statistical summary

df.describe ()

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | symboling | wheel-base | length | width | height | curb-weight | engine-size | compression-ratio | city-mpg | highway-mpg |
| count | 205.000000 | 205.000000 | 205.000000 | 205.000000 | 205.000000 | 205.000000 | 205.000000 | 205.000000 | 205.000000 | 205.000000 |
| mean | 0.834146 | 198.756585 | 174.049268 | 65.907805 | 53.724878 | 2555.565854 | 126.907317 | 10.142537 | 25.219512 | 30.751220 |
| std | 1.245307 | 6.021776 | 12.337289 | 2.145204 | 2.443522 | 520.680204 | 41.642693 | 3.972040 | 6.542142 | 6.886443 |
| min | -2.000000 | 86.600000 | 141.100000 | 60.300000 | 47.800000 | 1488.000000 | 61.000000 | 7.000000 | 13.000000| | 16.000000 |
| 25% | 0.000000 | 94.500000 | 166.300000 | 64.100000 | 52.000000 | 2145.000000 | 97.000000 | 8.600000 | 19.000000 | 25.000000 |
| 50% | 1.000000 | 97.000000 | 173.200000 | 65.500000 | 54.100000 | 2414.000000 | 120.000000 | 9.000000 | 24.000000 | 30.000000 |
| 75% | 2.000000 | 102.400000 | 183.100000 | 66.900000 | 55.500000 | 2935.000000 | 141.000000 | 9.400000 | 30.000000 | 34.000000 |
| max | 3.000000 | 120.900000 | 208.100000 | 72.300000 | 59.800000 | 4066.000000 | 326 000000 | 23.000000 | 49.000000 | 54.000000 |

IBM Developer

SKILLS NETWORK



Question

how would you generate descriptive statistics for all the columns for the dataframe df

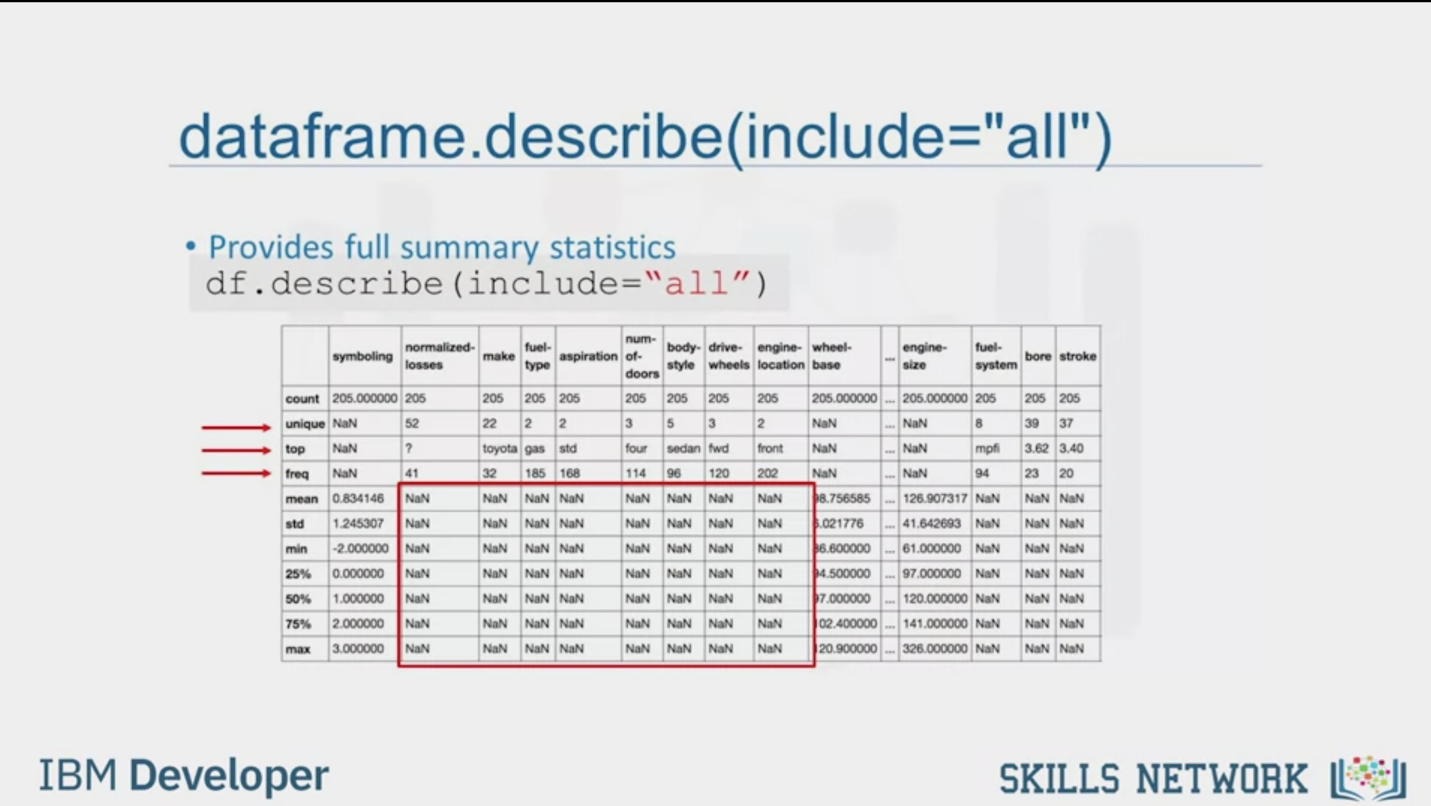
1 df.describe( )

1 df.describe(include = "all")

1 df.info

Skip

Send



dataframe.describe(include="all")

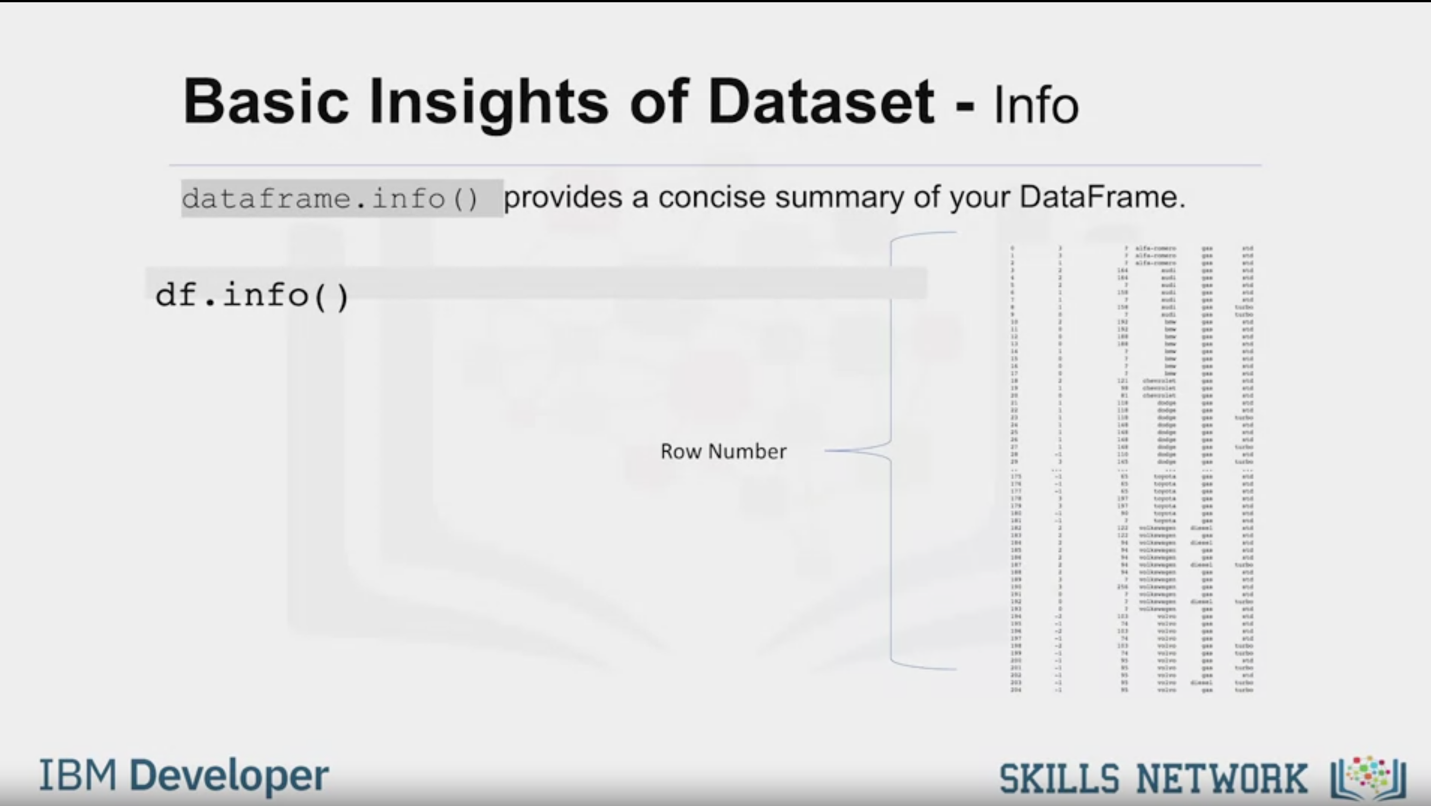
Provides full summary statistics

df.describe (include="all")

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | … | engine-size | fuel-system | bore | stroke |
| count | 205.000000 | 205 | 205 | 205 | 205 | 205 | 205 | 205 | 205 | 205.000000 | … | 205.000000 | 205 | 205 | 205 |
| unique | NaN | 52 | 22 | 2 | 2 | 3 | 5 | 3 | 2 | NaN | … | NaN | 8 | 39 | 37 |
| top | NaN | ? | toyota | gas | std | four | sedan | fwd | front | NaN | … | NaN | mpfi | 3.62 | 3.40 |
| freq | NaN | 41 | 32 | 185 | 168 | 114 | 96 | 120 | 202 | NaN | … | NaN | 94 | 23 | 20 |
| mean | 0.834146 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 98.756585 | … | 126.907317 | NaN | NaN | NaN |
| std | 1.245307 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 5.021776 | … | 41.642693 | NaN | NaN | NaN |
| min | -2.000000 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 36.600000 | … | 61.000000 | NaN | NaN | NaN |
| 25% | 0.000000 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 94.500000 | … | 97.000000 | NaN | NaN | NaN |
| 50% | 1.000000 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 97.000000 | … | 120.000000 | NaN | NaN | NaN |
| 75% | 2.000000 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 102.400000 | … | 141.000000 | NaN | NaN | NaN |
| max | 3.000000 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 120.900000 | … | 326.000000 | NaN | NaN | NaN |

IBM Developer

SKILLS NETWORK



Basic Insights of Dataset - Info

dataframe.info () provides a concise summary of our DataFrame.

df.info( )

Row Number

IBM Developer

SKILLS NETWORK



SKILLS NETWORK

IBM Developer

In this video, we introduce some simple Pandas methods that

all data scientists and analysts should know

when using Python, Pandas and data.

At this point, we assume that the data has been loaded.

It's time for us to explore the dataset.

Pandas has several built-in methods that can be used to understand

the datatype or features or to

look at the distribution of data within the dataset.

Using these methods, gives an overview of the dataset and also point out

potential issues such as the wrong data type of

features which may need to be resolved later on.

Data has a variety of types.

The main types stored in Pandas' objects are object,

float, Int, and datetime.

The data type names are somewhat different from those in native Python.

This table shows the differences and similarities between them.

Some are very similar such as the numeric data types, int and float.

The object pandas type function's similar to string in Python,

save for the change in name.

While the datetime Pandas type,

is a very useful type for handling time series data.

There are two reasons to check data types in a dataset.

Pandas automatically assigns types based

on the encoding it detects from the original data table.

For a number of reasons,

this assignment may be incorrect.

For example, it should be awkward if

the car price column which we should

expect to contain continuous numeric numbers,

is assigned the data type of object.

It would be more natural for it to have the float type.

Jerry may need to manually change the data type to float.

The second reason, is that allows an experienced data scientists

to see which Python functions can be applied to a specific column.

For example, some math functions can only be applied to numerical data.

If these functions are applied to non-numerical data an error may result.

When the dtype method is applied to the data set,

the data type of each column is returned in a series.

A good data scientists intuition

tells us that most of the data types make sense.

They make of cars for example are names.

So, this information should be of type object.

The last one on the list could be an issue.

As bore is a dimension of an engine,

we should expect a numerical data type to be used.

Instead, the object type is used.

In later sections, Jerry will have to correct these type mismatches.

Now, we would like to check the statistical summary of

each column to learn about the distribution of data in each column.

The statistical metrics can tell the data scientist if there are

mathematical issues that may exist such

as extreme outliers and large deviations.

The data scientists may have to address these issues later.

To get the quick statistics,

we use the describe method.

It returns the number of terms in the column as count,

average column value as mean,

column standard deviation as std,

the maximum minimum values,

as well as the boundary of each of the quartiles.

By default, the dataframe.describe functions

skips rows and columns that do not contain numbers.

It is possible to make the describe method

worked for object type columns as well.

To enable a summary of all the columns,

we could add an argument.

Include equals all inside the describe function bracket.

Now, the outcome shows the summary of all the 26 columns,

including object typed attributes.

We see that for the object type columns,

a different set of statistics is evaluated,

like unique, top, and frequency.

Unique is the number of distinct objects in the column.

Top is most frequently occurring object,

and freq is the number of times the top object appears in the column.

Some values in the table are shown here as

NaN which stands for not a number.

This is because that particular statistical metric

cannot be calculated for that specific column data type.

Another method you can use to check your dataset,

is the dataframe.info function.

This function shows the top 30 rows and bottom 30 rows of the data frame.