JOSÉ RIZAL UNIVERSITY
Find your Inner Hero

husay rizal

Unveiling the Data

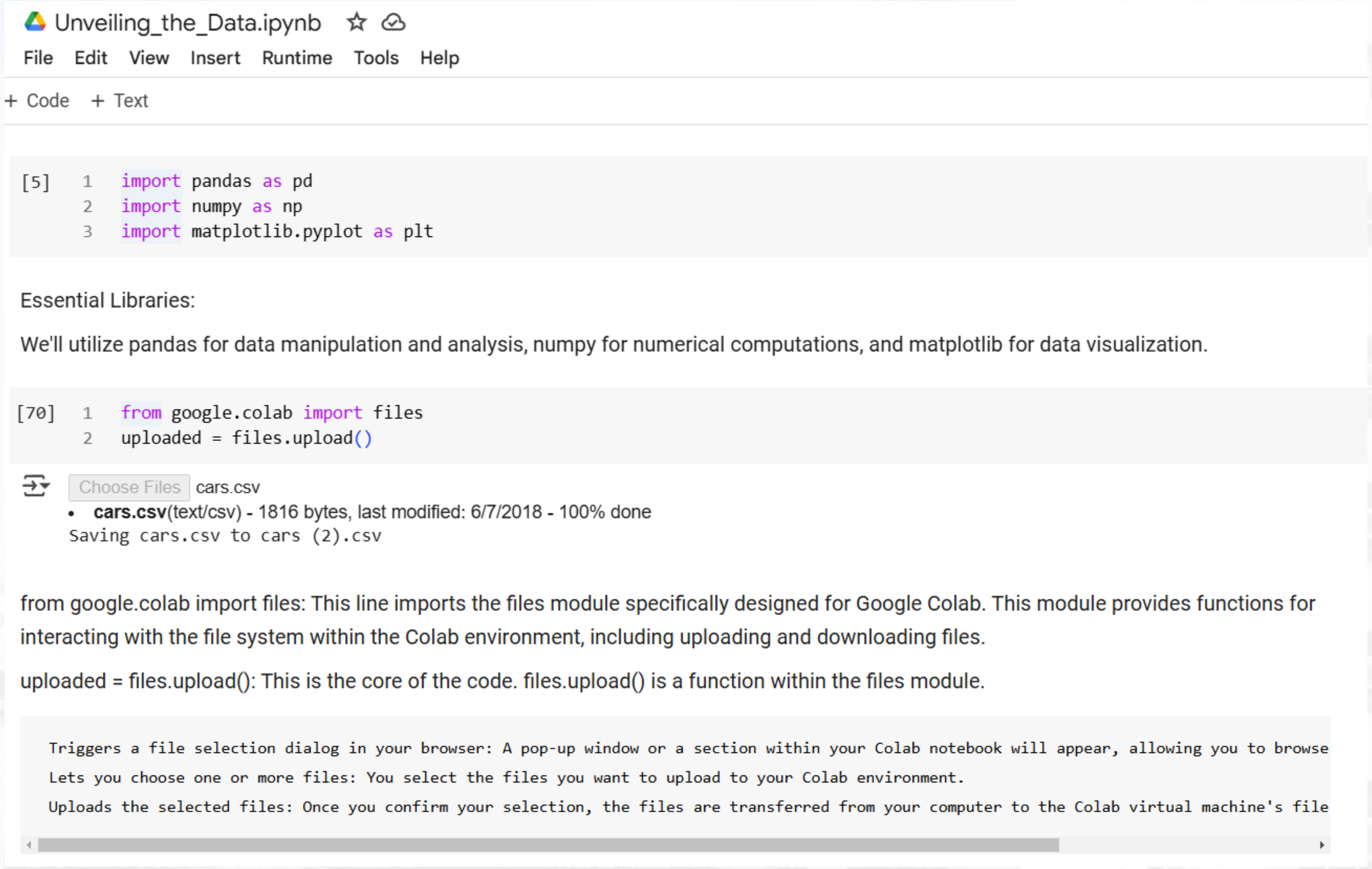
ISO 21001:2018
TÜVNORD

DATA PREPROCESSING

Data Preprocessing is an important step of data preparation since datasets often have missing values, incorrect datatypes, and irrelevant features. Through this, the dataset will be cleaned and ready for model training.

- This section consists of:
  - *Preliminary Steps* - In this part, the following are performed: importing essential tools, loading the dataset, constructing the DataFrame, and renaming columns.
  - *Understanding the Dataset* - This part involves performing operations that help us understand the dataset and determine what preprocessing steps must be taken.
  - *Data Cleaning* - Data cleaning was performed by removing duplicates, handling missing values, and ensuring data type consistency and accuracy.

- *Data Transformation* - It involves transforming data through standardization, handling outliers, and extracting components to create new features for improved analysis.
- *Feature Engineering* - Through this, categorical variables were encoded and additional interaction features from existing variables were generated.
- *Data Analysis* - Summary statistics was performed to provide valuable insights about the dataset.
- *Feature Selection and Splitting* - Through correlation, relevant features were selected based on the problem being solved.



```
[5]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

Essential Libraries:

We'll utilize pandas for data manipulation and analysis, numpy for numerical computations, and matplotlib for data visualization.

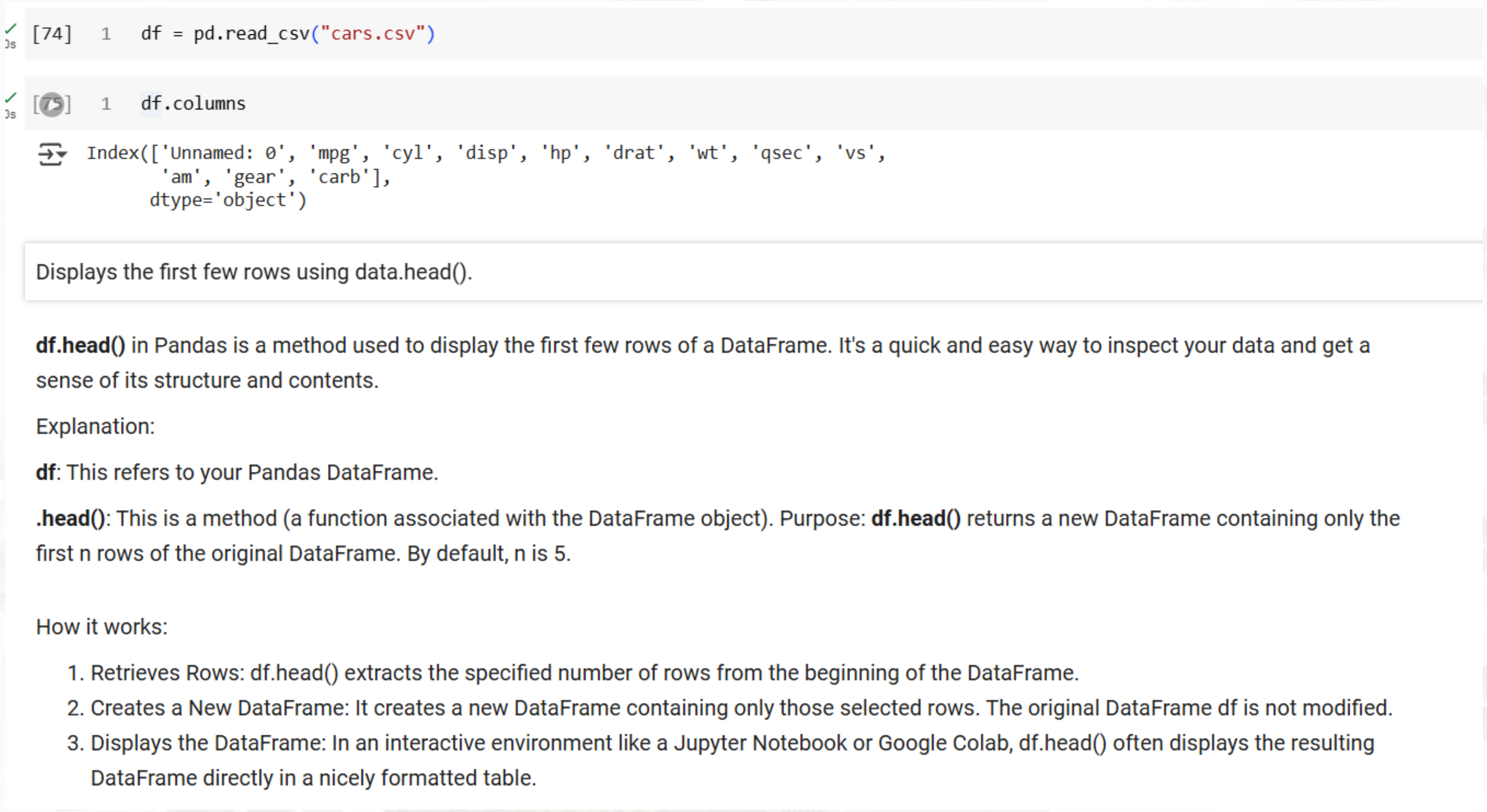
```
[70] from google.colab import files
     uploaded = files.upload()
```

```
Choose Files cars.csv
• cars.csv(text/csv) - 1816 bytes, last modified: 6/7/2018 - 100% done
Saving cars.csv to cars (2).csv
```

from google.colab import files: This line imports the files module specifically designed for Google Colab. This module provides functions for interacting with the file system within the Colab environment, including uploading and downloading files.

uploaded = files.upload(): This is the core of the code. files.upload() is a function within the files module.

```
Triggers a file selection dialog in your browser: A pop-up window or a section within your Colab notebook will appear, allowing you to browse
Lets you choose one or more files: You select the files you want to upload to your Colab environment.
Uploads the selected files: Once you confirm your selection, the files are transferred from your computer to the Colab virtual machine's file
```



```
[74]  df = pd.read_csv("cars.csv")
```

```
[75]  df.columns
```

```
Index(['Unnamed: 0', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs',
       'am', 'gear', 'carb'],
      dtype='object')
```

Displays the first few rows using data.head().

df.head() in Pandas is a method used to display the first few rows of a DataFrame. It's a quick and easy way to inspect your data and get a sense of its structure and contents.

Explanation:

df: This refers to your Pandas DataFrame.

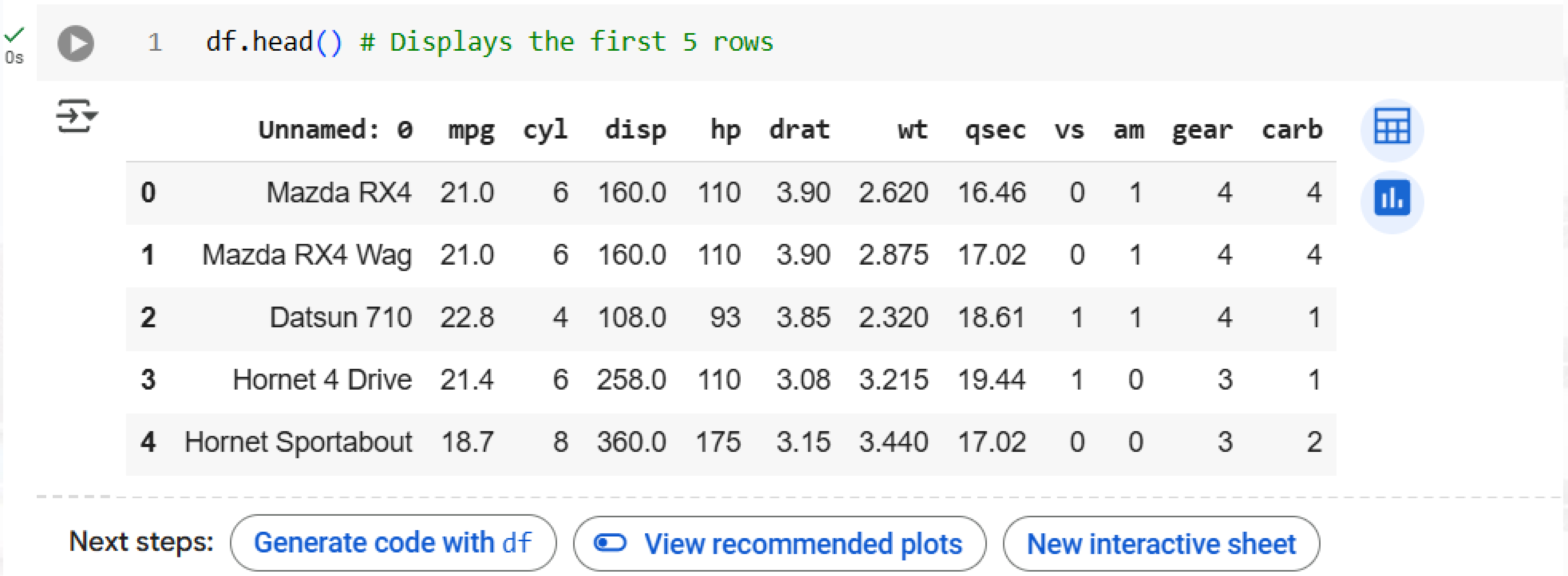
.head(): This is a method (a function associated with the DataFrame object). Purpose: **df.head()** returns a new DataFrame containing only the first n rows of the original DataFrame. By default, n is 5.

How it works:

1. Retrieves Rows: df.head() extracts the specified number of rows from the beginning of the DataFrame.
2. Creates a New DataFrame: It creates a new DataFrame containing only those selected rows. The original DataFrame df is not modified.
3. Displays the DataFrame: In an interactive environment like a Jupyter Notebook or Google Colab, df.head() often displays the resulting DataFrame directly in a nicely formatted table.

JOSÉ RIZAL UNIVERSITY
Find your Inner Hero

husay rizal



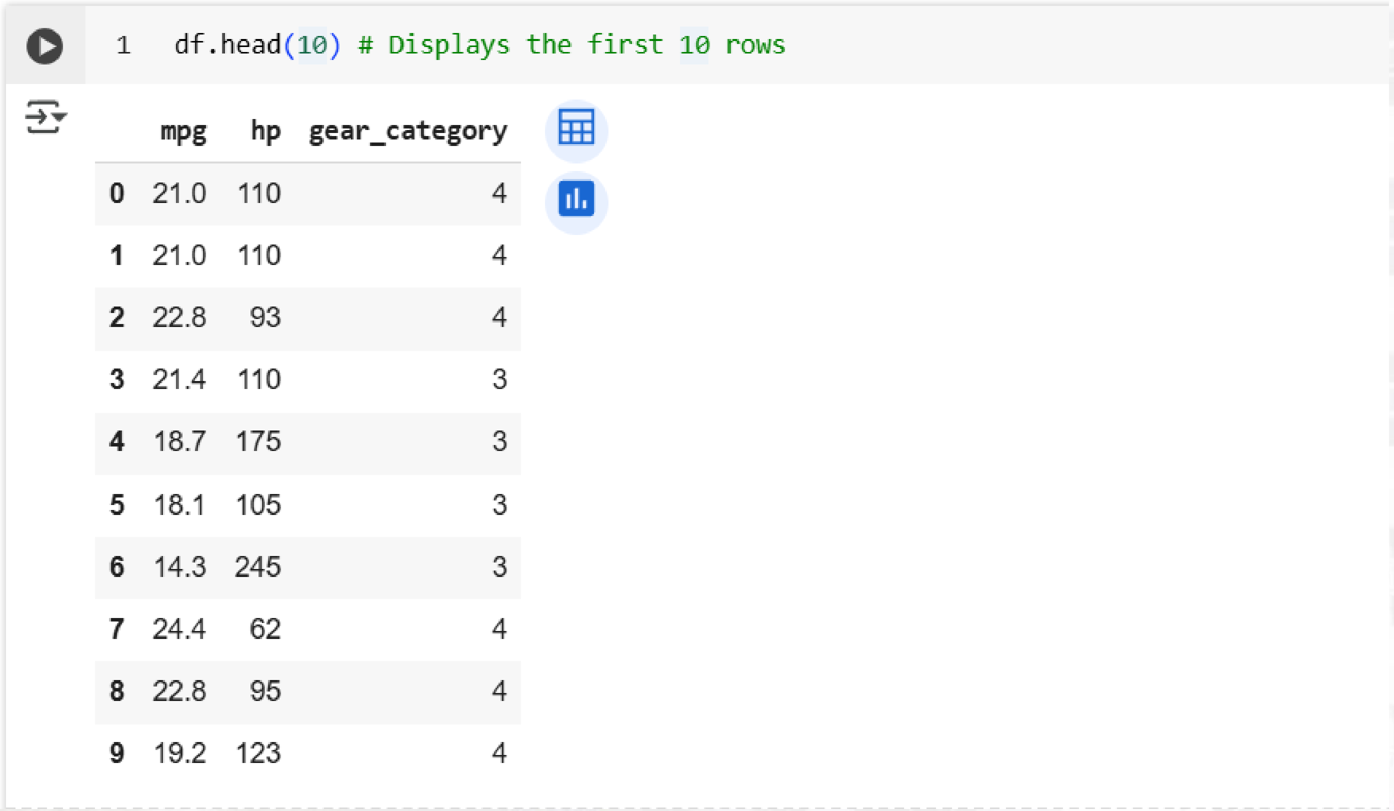
```
df.head() # Displays the first 5 rows
```

```
        Unnamed: 0   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  carb
0        Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4     4
1    Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4     4
2       Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4     1
3   Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3     1
4  Hornet Sportabout  18.7  8  360.0  175  3.15  3.440  17.02   0   0     3     2
```

JOSÉ RIZAL UNIVERSITY
Find your Inner Hero

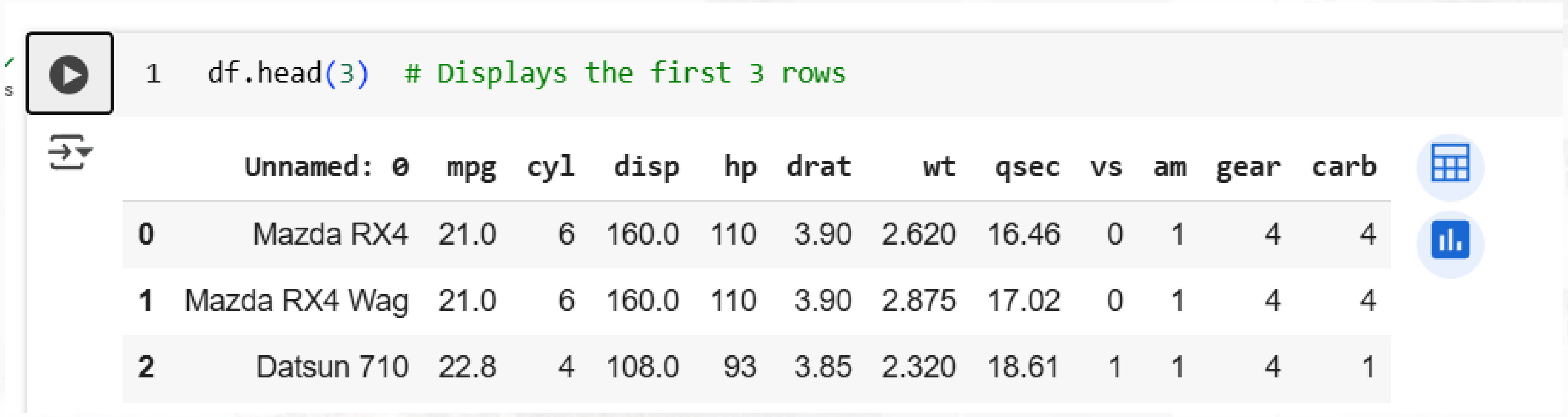
husay rizal

Specifying the number of rows:



```
df.head(10) # Displays the first 10 rows
```

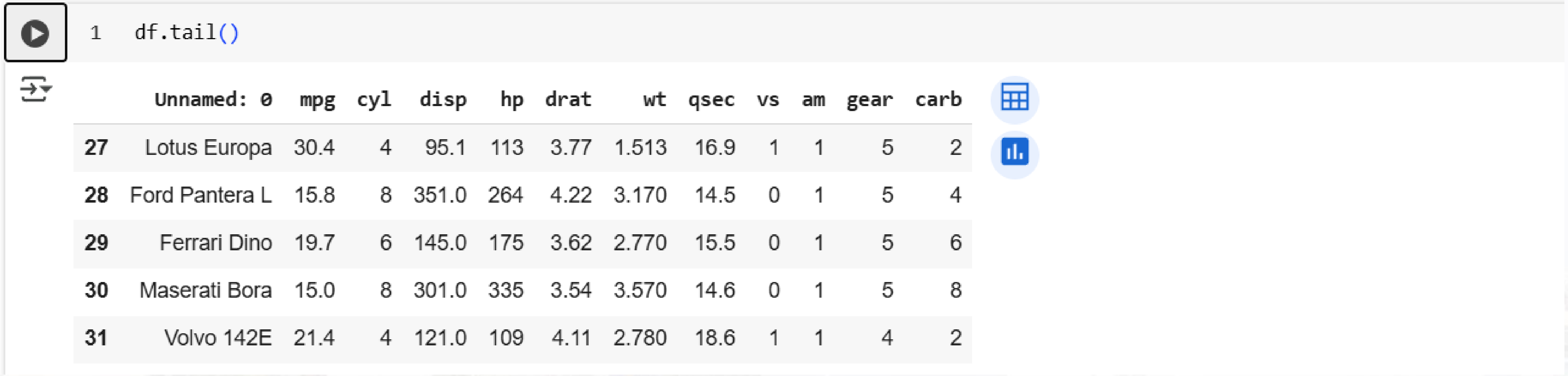
```
   mpg   hp  gear_category
0  21.0  110              4
1  21.0  110              4
2  22.8   93              4
3  21.4  110              3
4  18.7  175              3
5  18.1  105              3
6  14.3  245              3
7  24.4   62              4
8  22.8   95              4
9  19.2  123              4
```



```
df.head(3)  # Displays the first 3 rows
```

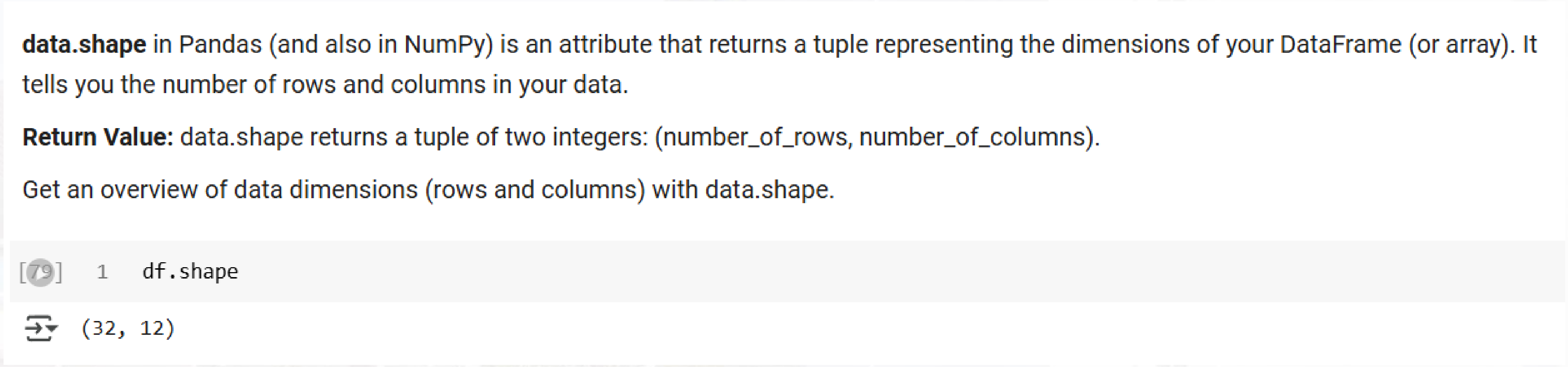
```
   Unnamed: 0      mpg  cyl  disp   hp   drat  wt     qsec   vs  am  gear  carb
0  Mazda RX4       21.0  6   160.0  110  3.90  2.620  16.46  0   1   4     4
1  Mazda RX4 Wag   21.0  6   160.0  110  3.90  2.875  17.02  0   1   4     4
2  Datsun 710      22.8  4   108.0   93  3.85  2.320  18.61  1   1   4     1
```

df.tail() is very useful for quickly previewing the end of your data. This is often helpful when you've just added new data to your DataFrame or when you want to check the last few entries. Combined with **df.head()**, you can get a good overview of your data's structure and content without having to display the entire DataFrame, especially when it's large.



```
df.tail()
```

	Unnamed: 0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
27	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.9	1	1	5	2
28	Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.5	0	1	5	4
29	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.5	0	1	5	6
30	Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.6	0	1	5	8
31	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.6	1	1	4	2



data.shape in Pandas (and also in NumPy) is an attribute that returns a tuple representing the dimensions of your DataFrame (or array). It tells you the number of rows and columns in your data.

Return Value: data.shape returns a tuple of two integers: (number_of_rows, number_of_columns).

Get an overview of data dimensions (rows and columns) with data.shape.

```
[79] df.shape
```

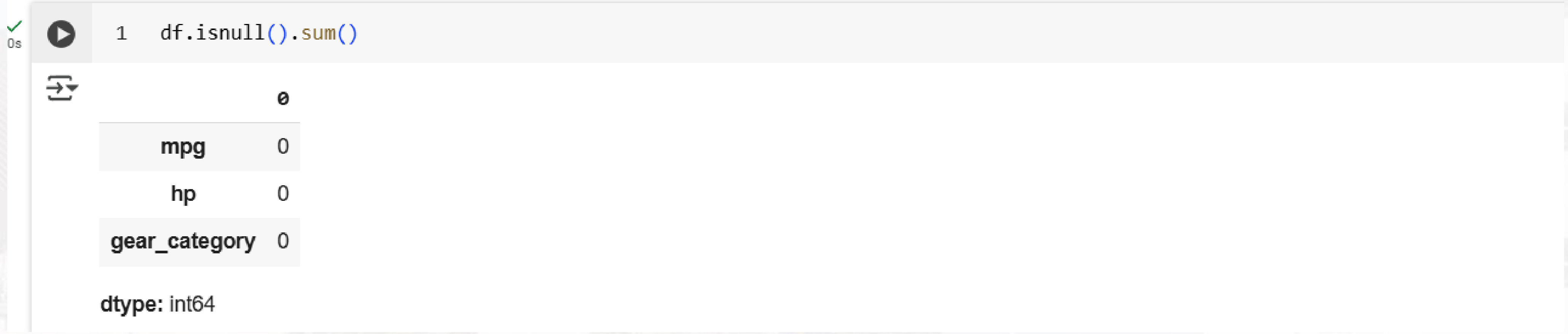
```
(32, 12)
```

df.isnull().sum() is a powerful combination of Pandas methods used to count the number of missing (or null) values in each column of a DataFrame.

isnull(): This method, when applied to a DataFrame, returns a new DataFrame of the same shape as the original. Each cell in the new DataFrame contains a boolean value:

True if the corresponding cell in the original DataFrame contains a missing value (NaN, None, etc.).

False if the cell contains a non-missing value.



```
df.isnull().sum()
```

```
                0
          mpg   0
           hp   0
gear_category   0

dtype: int64
```

df.info() in Pandas is a method that provides a concise summary of information about your DataFrame. It's a very useful way to get a quick overview of your data's structure and characteristics.

.info(): This is a method (a function associated with the DataFrame object). Purpose: df.info() prints information about the DataFrame to the console (or output area).

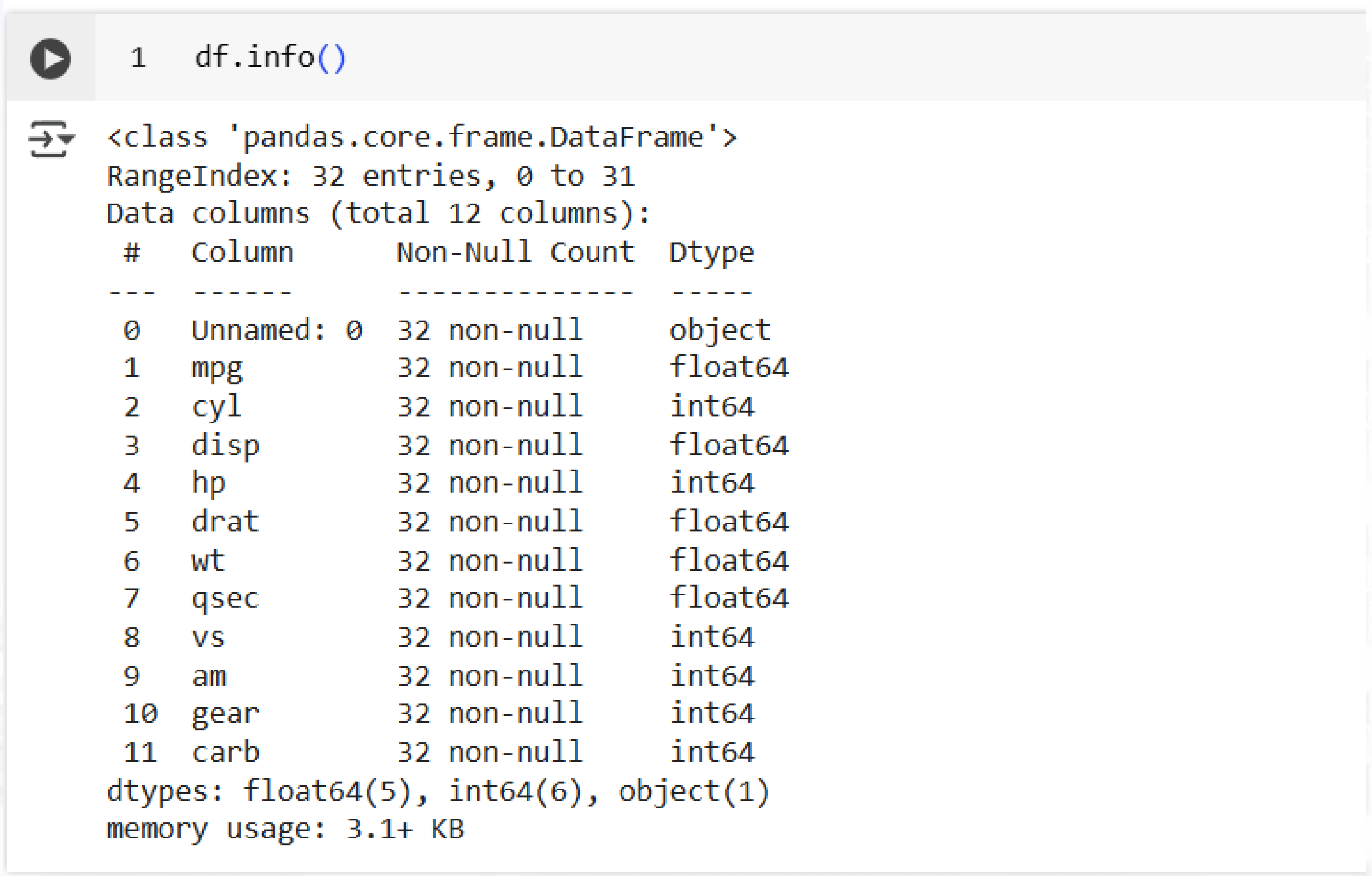
What information does df.info() provide?

DataFrame Class: It tells you that you're working with a Pandas DataFrame.

RangeIndex: It shows the index range (row labels). Often, this is a simple numerical range like RangeIndex: 8 entries, 0 to 7 if you have 8 rows.

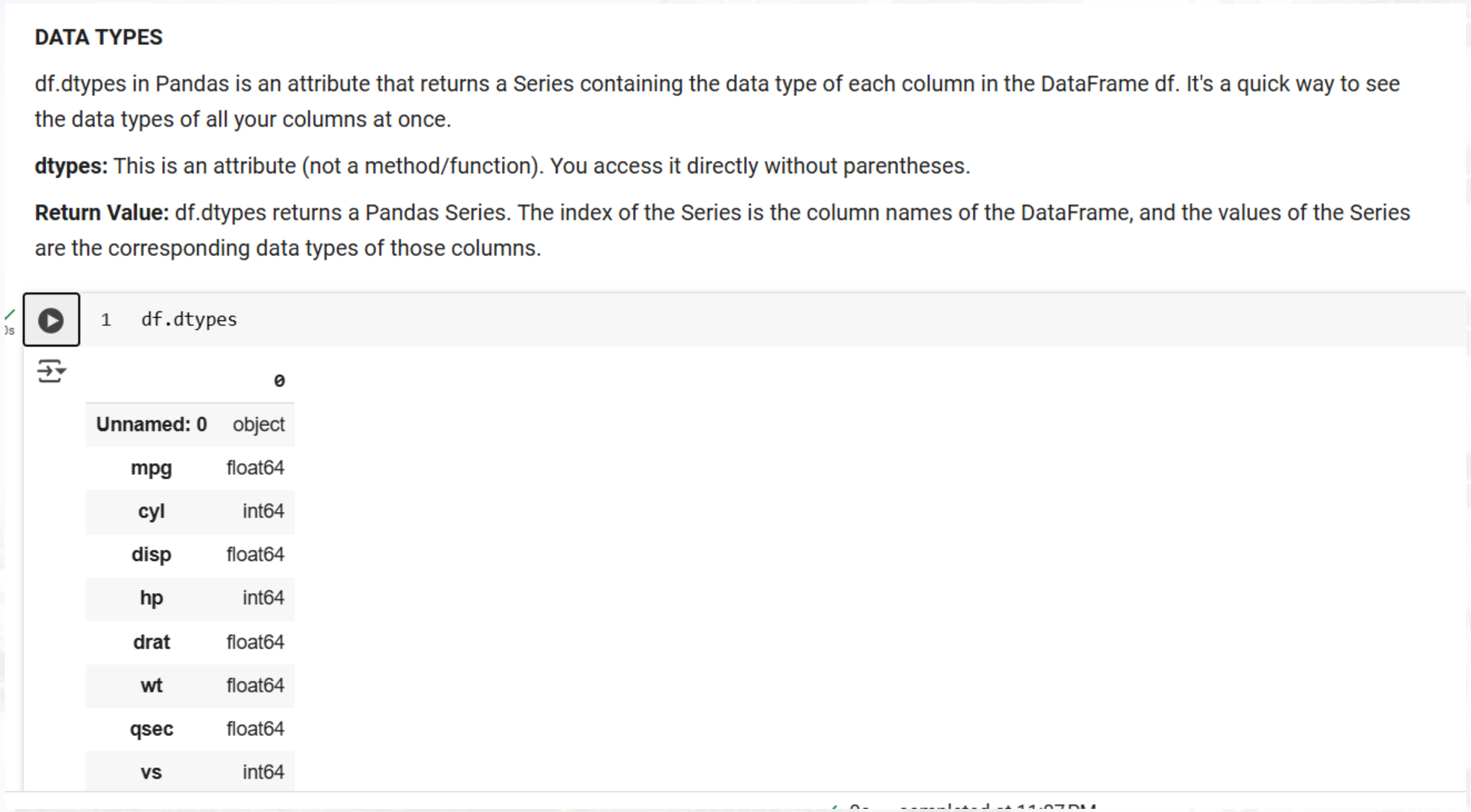
Data columns (total n columns): It lists all the columns in your DataFrame, along with:

```
    Column Name: The name of each column.
    Non-Null Count: The number of non-missing (non-null) values in each column. This is very helpful for identifying missing data.
    Dtype: The data type of each column (e.g., int64, float64, object, bool). object often indicates string data or mixed data types.
    Memory Usage: It shows the amount of memory the DataFrame is using.  This can be useful for understanding memory efficiency, especially w
```



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0  32 non-null     object
 1   mpg         32 non-null     float64
 2   cyl         32 non-null     int64
 3   disp        32 non-null     float64
 4   hp          32 non-null     int64
 5   drat        32 non-null     float64
 6   wt          32 non-null     float64
 7   qsec        32 non-null     float64
 8   vs          32 non-null     int64
 9   am          32 non-null     int64
 10  gear        32 non-null     int64
 11  carb        32 non-null     int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
```



DATA TYPES

df.dtypes in Pandas is an attribute that returns a Series containing the data type of each column in the DataFrame df. It's a quick way to see the data types of all your columns at once.

dtypes: This is an attribute (not a method/function). You access it directly without parentheses.

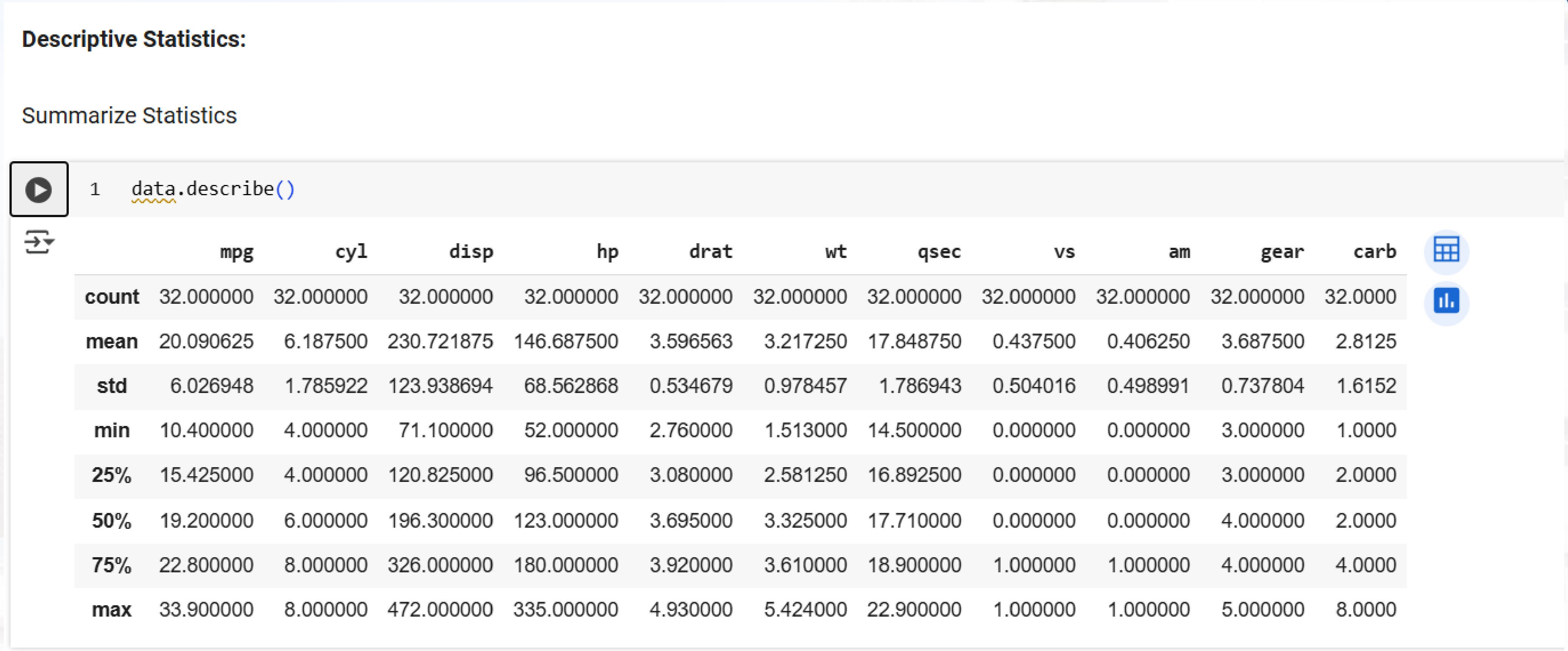
Return Value: df.dtypes returns a Pandas Series. The index of the Series is the column names of the DataFrame, and the values of the Series are the corresponding data types of those columns.

```
df.dtypes
```

```
                  0
Unnamed: 0   object
       mpg  float64
       cyl    int64
      disp  float64
        hp    int64
      drat  float64
        wt  float64
      qsec  float64
        vs    int64
```

JOSÉ RIZAL UNIVERSITY
Find your Inner Hero

husay rizal



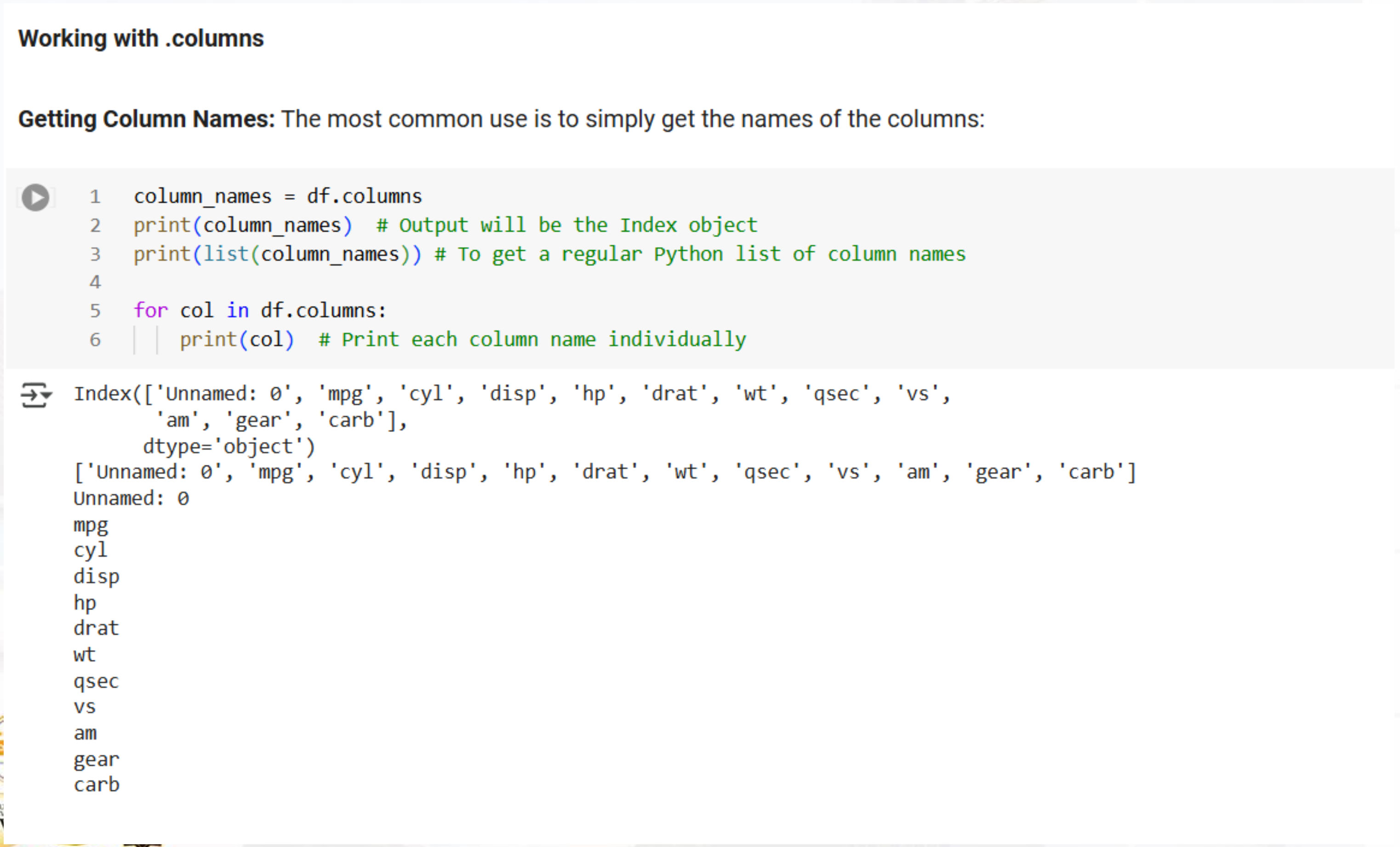
Descriptive Statistics:

Summarize Statistics

```
data.describe()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.0000
mean	20.090625	6.187500	230.721875	146.687500	3.596563	3.217250	17.848750	0.437500	0.406250	3.687500	2.8125
std	6.026948	1.785922	123.938694	68.562868	0.534679	0.978457	1.786943	0.504016	0.498991	0.737804	1.6152
min	10.400000	4.000000	71.100000	52.000000	2.760000	1.513000	14.500000	0.000000	0.000000	3.000000	1.0000
25%	15.425000	4.000000	120.825000	96.500000	3.080000	2.581250	16.892500	0.000000	0.000000	3.000000	2.0000
50%	19.200000	6.000000	196.300000	123.000000	3.695000	3.325000	17.710000	0.000000	0.000000	4.000000	2.0000
75%	22.800000	8.000000	326.000000	180.000000	3.920000	3.610000	18.900000	1.000000	1.000000	4.000000	4.0000
max	33.900000	8.000000	472.000000	335.000000	4.930000	5.424000	22.900000	1.000000	1.000000	5.000000	8.0000

Great Place To Work. Certified JUN 2024-JUN 2025 PH
PACUCOA LEVEL IV
CHED AUTONOMOUS STATUS
ISO 21001:2018
TÜVNORD



Working with .columns

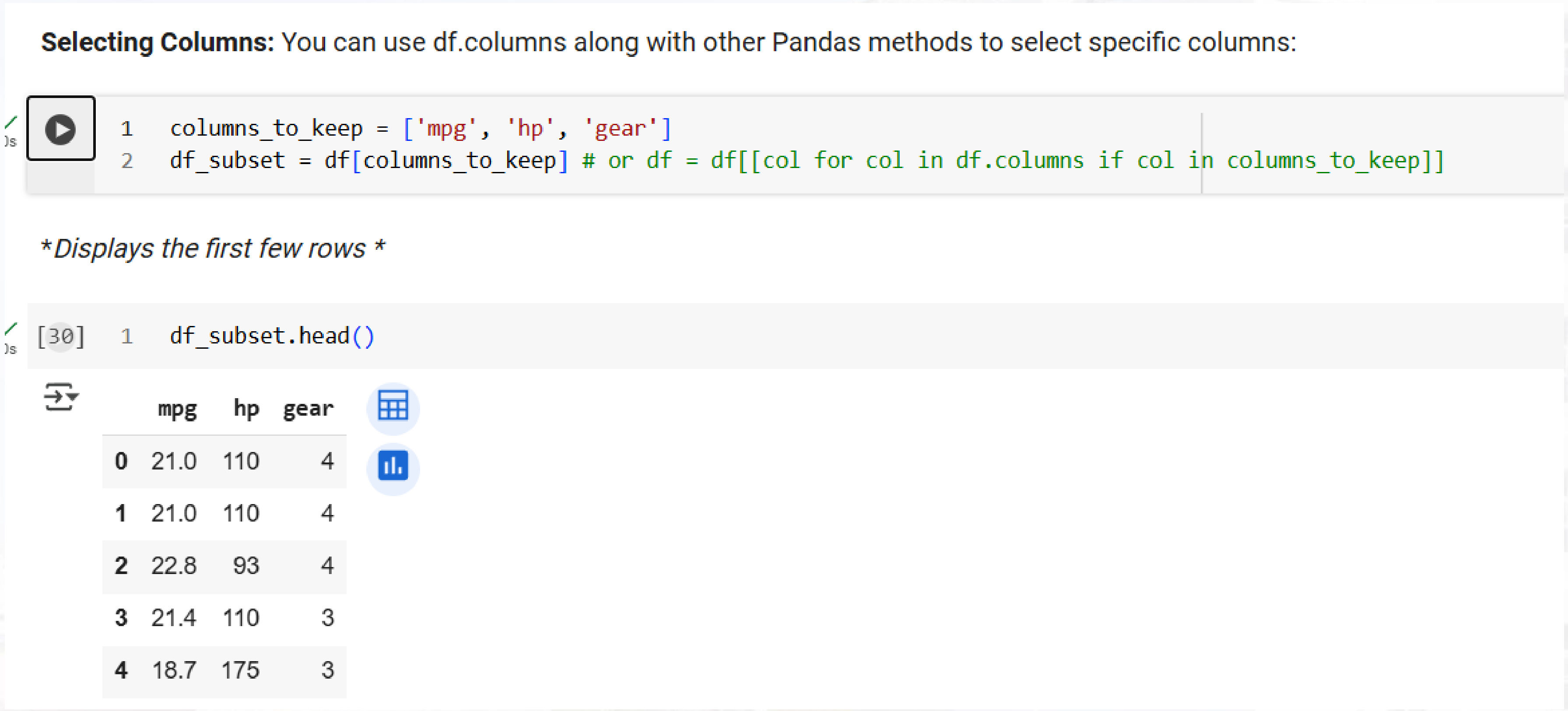
Getting Column Names: The most common use is to simply get the names of the columns:

```python
column_names = df.columns
print(column_names)  # Output will be the Index object
print(list(column_names)) # To get a regular Python list of column names

for col in df.columns:
    print(col)  # Print each column name individually
```

```
Index(['Unnamed: 0', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs',
       'am', 'gear', 'carb'],
      dtype='object')
['Unnamed: 0', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear', 'carb']
Unnamed: 0
mpg
cyl
disp
hp
drat
wt
qsec
vs
am
gear
carb
```

JOSÉ RIZAL UNIVERSITY
Find your Inner Hero



Selecting Columns: You can use df.columns along with other Pandas methods to select specific columns:

```
columns_to_keep = ['mpg', 'hp', 'gear']
df_subset = df[columns_to_keep] # or df = df[[col for col in df.columns if col in columns_to_keep]]
```

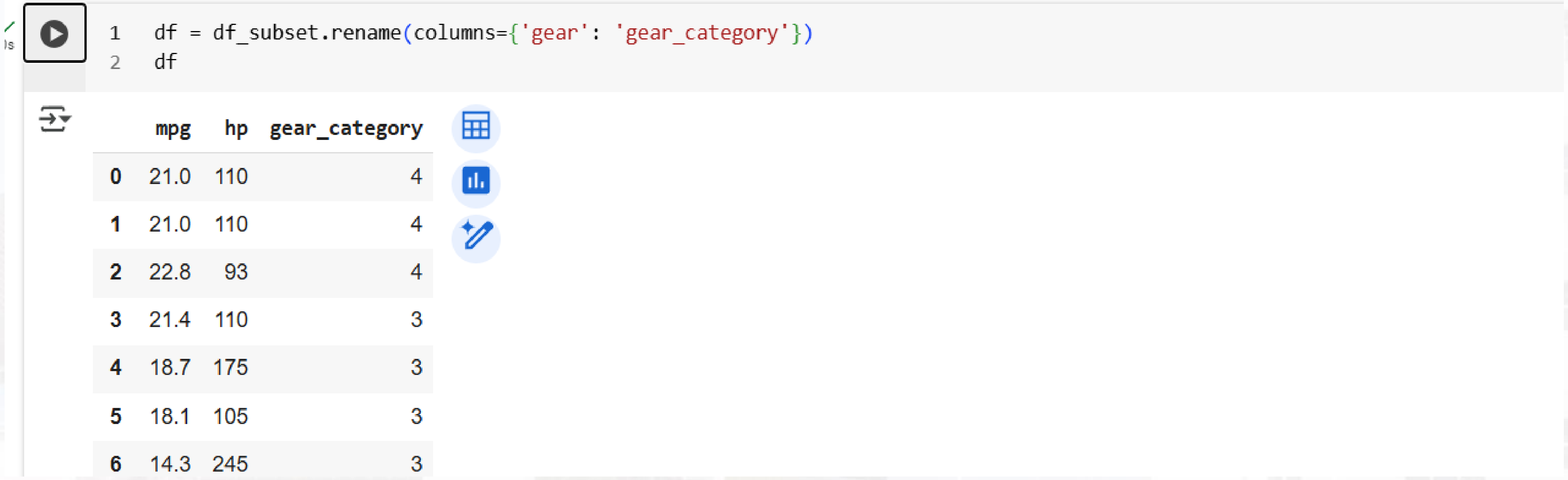
Displays the first few rows

```
[30] df_subset.head()
```

```
    mpg   hp  gear
0  21.0  110     4
1  21.0  110     4
2  22.8   93     4
3  21.4  110     3
4  18.7  175     3
```

ISO 21001:2018
TÜVNORD

Renaming Columns: While you don't directly modify df.columns, you can use it in conjunction with the df.rename() function to rename columns:



```
df = df_subset.rename(columns={'gear': 'gear_category'})
df
```

```
    mpg   hp  gear_category
0  21.0  110              4
1  21.0  110              4
2  22.8   93              4
3  21.4  110              3
4  18.7  175              3
5  18.1  105              3
6  14.3  245              3
```