Pandas DataFrame Operations - Read, View and Extract Information

WE'LL COVER THE FOLLOWING

Learning Pandas with IMDB-Movies Dataset

Time for some real fun! We have learned how to create DataFrames already. Now we are going to explore the many operations that can be performed on them.

To make this step more engaging and fun, we are going to work with the ***IMDB Movies Dataset***. The IMDB dataset is a publicly available dataset that contains information about 14,762 movies. Each row consists of a movie and for each movie we have information like title, year of release, director, number of awards, rating, duration etc. Sounds fun to explore, right? Let's put our data scientist's hat on, and dive into the world of the movies! 🔎🤖

IMDB Data from 2006 to 2016 (Image Source: Kaggle)

Important DataFrame Operations

We are going to go through the most important DataFrame operations for a data scientist to know, one by one:

1. Reading Data From CSVs

The dataset for these lessons is here on Kaggle. Once we have downloaded the data, we can load it using the DataFrame creation method we mentioned in the previous lesson.

> 📌 **Note: Once you have gone through these "IMDB-lessons", I highly recommend you download this dataset and play with it. It is really important to get your hands dirty; don't just read through these lessons!**

> 📌 **You can also find the Juptyter Notebook with all the code for these "IMDB-lessons" on my Git profile, here.**

> 📌 **You can find the live execution of the Jupyter Notebook at the end of this lesson.**

```
import pandas as pd

# Reading data from the downloaded CSV:
movies_df = pd.read_csv("IMDB-Movie-Data.csv")
```

We have created a DataFrame, *movies_df*, with a default index. What if we want to be able to access each row by the title of the movie, and not by some integer index?

One way to **set titles as our index** is by passing the column name as an additional parameter, *index_col*, to the `read_csv` method at file load time. The second way is to do this at a later stage by explicitly calling the `set_index()` method on the created DataFrame. We can see this in the code snippet below.

Having an index allows for easy and efficient searches. Looking up rows based on index values is like looking up dictionary values based on a key. For example, when the title is used as an index, we can quickly fetch the row for a particular movie by simply using its title for lookup instead of trying to find out its row number first.

Note that we are creating both title-indexed and default DataFrames (we don't need both), so that we can understand the indexing concept better by comparing the two as, in steps 2 and 4.

```
# 1. We can set the index at load time
movies_df_title_indexed = pd.read_csv("IMDB-Movie-Data.csv", index_col='Title')

# 2. We can set the index after the DataFrame has been created
movies_df_title_indexed = movies_df.set_index('Title')
```

2. Viewing the Data

Once we have created the DataFrame, a helpful first step is to take a sneak peek at the data. This helps us create a mental picture of the data and become more familiar with it.

We can use the `head()` method to visualize the first few rows of our dataset. This method outputs the first 5 rows of the DataFrame by default, but we can pass the number of rows we want as input parameter.

Let's first view the non-explicitly indexed (default) DataFrame:

```
movies_df.head()

# To ouput the top ten rows
movies_df.head(10)
```

```
movies_df.head()
```

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.13	76.0
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.3	157606	138.12	62.0
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016	108	7.2	60545	270.32	59.0
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	2016	123	6.2	393727	325.02	40.0

Now if we print the rows of the DataFrame with the explicit index, we can see that the name of the indexed column, *"Title"*, gets printed slightly lower than the rest of the columns, and it is displayed in place of the column which was showing row numbers in the previous case (with default index):

```
movies_df_title_indexed.head()
```

```
movies_df_title_indexed.head()
```

Title	Rank	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.13	76.0
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
Split	3	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.3	157606	138.12	62.0
Sing	4	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016	108	7.2	60545	270.32	59.0
Suicide Squad	5	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	2016	123	6.2	393727	325.02	40.0

After this simple visualization, we are already more familiar with our dataset. Now we know which columns make up our data and what the values in each

Now we know which columns make up our data and what the values in each column look like. We can now see that each row in our dataset consists of a

movie and for each movie we have information like *"Rating"*, *"Revenue"*, *"Actors"* and *"Genre"*. Each column is also called a feature, attribute, or a variable.

We can also see that each movie has an associated rank as well and that rows are ordered by the *"Rank"* feature.

Similarly, it can be useful to observe the last rows of the dataset. We can do this by using the `tail()` method. Like the *head()* method, *tail()* also accepts the number of rows we want to view as input parameter.

Let's look at the last three rows of our dataset (worst movies in terms of rank):

```
movies_df_title_indexed.tail(3)
```

movies_df_title_indexed.tail(3)

Title	Rank	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
Step Up 2: The Streets	998	Drama,Music,Romance	Romantic sparks occur between two dance studen...	Jon M. Chu	Robert Hoffman, Briana Evigan, Cassie Ventura,...	2008	98	6.2	70699	58.01	50.0
Search Party	999	Adventure,Comedy	A pair of friends embark on a mission to reuni...	Scot Armstrong	Adam Pally, T.J. Miller, Thomas Middleditch,Sh...	2014	93	5.6	4881	NaN	22.0
Nine Lives	1000	Comedy,Family,Fantasy	A stuffy businessman finds himself trapped ins...	Barry Sonnenfeld	Kevin Spacey, Jennifer Garner, Robbie Amell,Ch...	2016	87	5.3	12435	19.64	11.0

3. Getting Information About the Data

As a first step, it is recommended to ***get the 10000-foot view of the data***. We are going to look at two different methods for getting this high-level view: `info()` and `describe()`.

a. *info():* This method allows us to get some essential details about our dataset, like the number of rows and columns, the number of index entries within that index range, the type of data in each column, the number of non-null values, and the memory used by the DataFrame:

```
# This should be one of the very first commands you run after loading your data:
movies_df_title_indexed.info()
```

```
movies_df_title_indexed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, Guardians of the Galaxy to Nine Lives
Data columns (total 11 columns):
Rank                    1000 non-null int64
Genre                   1000 non-null object
Description             1000 non-null object
Director                1000 non-null object
Actors                  1000 non-null object
Year                    1000 non-null int64
Runtime (Minutes)       1000 non-null int64
Rating                  1000 non-null float64
Votes                   1000 non-null int64
Revenue (Millions)      872 non-null float64
Metascore               936 non-null float64
dtypes: float64(3), int64(4), object(4)
memory usage: 93.8+ KB
```

As we can see from the snippet above, our dataset consists of 1000 rows and 11 columns, and it is using about 93KB of memory. An important thing to notice is that we have two columns with missing values: *"Revenue"* and *"Metascore"*. Knowing which columns have missing values is important for the next steps. Handling missing data is an important data preparation step in any data science project; more often than not, we need to use machine learning algorithms and methods for data analysis that are not able to handle missing data themselves.

The output of the *info()* method also allows shows us if we have any columns that we expected to be integers but are actually strings instead. For example, if the revenue had been recorded as a string type, before doing any numerical analysis on that feature, we would have needed to convert the values for revenue from string to float.

`.shape`: This is a fast and useful attribute which outputs a tuple, *<rows, columns>*, representing the number of rows and columns in the DataFrame. This attribute comes in very handy when cleaning and transforming data. Say we had filtered the rows based on some criteria. We can use *shape* to quickly check how many rows we are left with in the filtered DataFrame:

```
movies_df_title_indexed.shape
# Output: (1000, 11)

## Note: .shape has no parentheses and is a simple tuple of format (rows, columns)
```

```
## Note: .shape has no parentheses and is a simple tuple of format (rows, columns).
# From the output we can see that we have 1000 rows and 11 columns in our movies DataFrame.
```

b. *describe():* This is a great method for doing a quick analysis of the dataset. It computes summary statistics of integer/double variables and gives us some basic statistical details like percentiles, mean, and standard deviation:

```
# We can do a quick analysis of any data set using:
movies_df_title_indexed.describe()
```

```
movies_df_title_indexed.describe()
```

	Rank	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	872.000000	936.000000
mean	500.500000	2012.783000	113.172000	6.723200	1.698083e+05	82.956376	58.985043
std	288.819436	3.205962	18.810908	0.945429	1.887626e+05	103.253540	17.194757
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.000000
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	13.270000	47.000000
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	47.985000	59.500000
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	113.715000	72.000000
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.000000

We can see that we have a lot of useful high-level insights about our data now. For example, we can tell that our dataset consists only of movies from 2006 (min Year) to 2016 (max Year). The maximum revenue generated by any movie during that period was 936.63M USD while the mean revenue was 82.9M USD. We can analyze all the other features as well and ***extract important information like a breeze!***

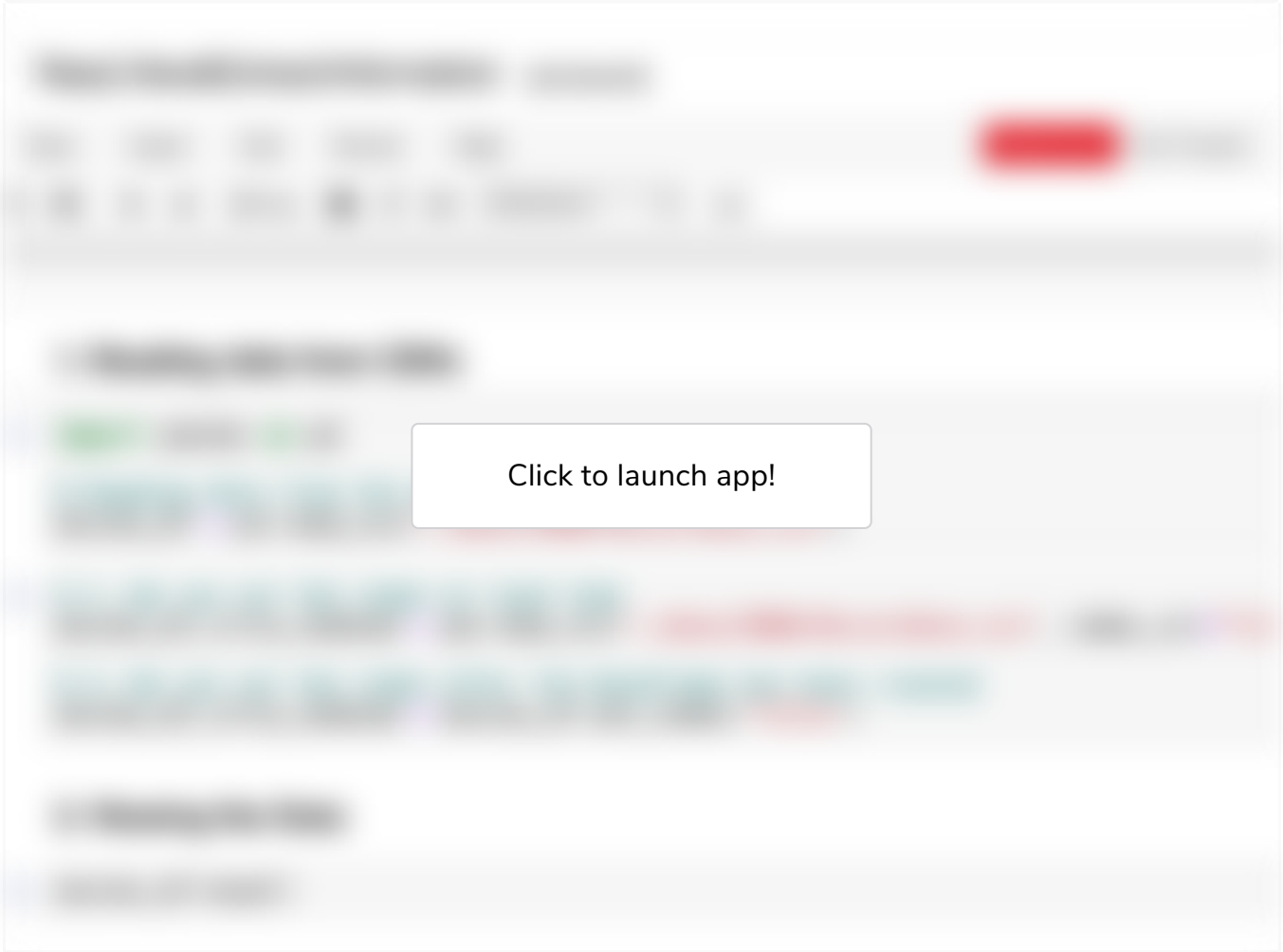
Jupyter Notebook

You can see the instructions running in the Jupyter Notebook below:

> 📝 **How to Use a Jupyter NoteBook?**
>
> - Click on **"Click to Launch"** 🚀 button to work and see the code running live in the notebook.
> - You can click  to open the **Jupyter Notebook in a new tab**.
> - Go to files and click *Download as* and then choose the format of the

- Go to files and click *Download as* and then choose the format of the file to **download** . You can choose Notebook(.ipynb) to download the file and work locally or on your personal Jupyter Notebook.
- ⚠ The notebook **session expires after 30 minutes of inactivity**. It will reset if there is no interaction with the notebook for 30 consecutive minutes.

Your app can be found at: https://1dgnrwwoynk5m-live-app.educative.run/notebooks/Read%2CView%26ExtractInformation.ipynb



Note: We will talk about these statistics concepts in very detail later, so don't worry if any of these stats sounds alien to you.