

# Pandas DataFrame Operations - Pivot Tables and Functions

## WE'LL COVER THE FOLLOWING ^

- 11. Pivot Table
- 12. Applying Functions
- Jupyter Notebook
- Final Thoughts

## 11. Pivot Table #

We have seen how grouping lets us explore relationships within a dataset. A pivot table is a similar operation. You have probably encountered it in spreadsheets or some other programs that operate on tables. If you have ever worked with Excel, Pandas can be used to create Excel style pivot tables.

The pivot table takes simple column-wise data as input, and groups the entries into a two-dimensional table to give a multidimensional summary of the data.

***Hard to understand? Let's understand the concept with an example!***

Say we want to **compare the \$\$\$ earned by the various directors per year**. We can create a pivot table using `pivot_table`; we can set `index='Director'` (row of the pivot table) and get the yearly revenue information by setting `columns='Year'`:

```
# Let's calculate the mean revenue per director but by using a pivot table instead of groupby
movies_df_title_indexed.pivot_table('Revenue (Millions)', index='Director',
aggfunc='sum', columns='Year').head()
```

The ***aggfunc*** parameter controls what type of aggregation is applied (mean by default). As in *groupby*, this can be a string representing one of the many

common choices, like `'sum'`, `'mean'`, `'count'`, `'min'`, `'max'`.

```
#Let's calculate the mean revenue per director but by using a pivot table instead of groupby as seen previously
movies_df_title_indexed.pivot_table('Revenue (Millions)', index='Director',aggfunc='sum', columns='Year').head()
```

Year	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Director											
Aamir Khan	NaN	1.20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Abdellatif Kechiche	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.2	NaN	NaN	NaN
Adam Leon	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
Adam McKay	148.21	NaN	100.47	NaN	119.22	NaN	NaN	NaN	NaN	70.24	NaN
Adam Shankman	NaN	118.82	NaN	NaN	NaN	NaN	38.51	NaN	NaN	NaN	NaN

From our pivot table, we can observe that for Aamir Khan we have only revenue for 2007. This can imply two things: 2007 was the only year in this 10 year period when any of his movies got released or that we simply do not have complete data for this director. We can also see that Adam McKay has the most movies over this ten year period, with his highest revenue being in 2006 and lowest in 2015.

With a simple pivot table, we can see the **annual trend in revenue per Director**; pivot tables are indeed a powerful tool for data analysis!

## 12. Applying Functions #

Applying functions to a dataset, `apply()`, is very handy for playing with data and creating new variables. To *apply()* a function means returning some value after passing each row/column of a DataFrame through some function. The function can be a default one or user-defined.

You might be thinking, *“Can’t we do this by iterating over the DataFrame or Series like with lists?”* Yes, you are right, we can. The problem is that it would not be an efficient approach, especially when dealing with large datasets. Pandas utilizes which means vectorization (operations are applied to whole arrays instead of individual elements).

For example, we could use **a function to classify movies into four buckets (“great”, “good”, “average”, “bad”) based on their numerical ratings**. We can do this in two steps:

- First, define a function that when given a rating determines the right bucket for that movie.
- Then apply that function to the DataFrame.

This is how we can do it in code:

```
# 1. Let's define the function to put movies into buckets based on their rating
def rating_bucket(x):
    if x >= 8.0:
        return "great"
    elif x >= 7.0:
        return "good"
    elif x >= 6.0:
        return "average"
    else:
        return "bad"

# 2. Let's apply the function
movies_df_title_indexed["Rating_Category"] = movies_df_title_indexed["Rating"].apply(rating_b

# 3. Let's see some results
movies_df_title_indexed.head(10)[['Rating', 'Rating_Category']]
```

```
# 1. Let's define the function to put movies into buckets based on their rating
def rating_bucket(x):
    if x >= 8.0:
        return "great"
    elif x >= 7.0:
        return "good"
    elif x >= 6.0:
        return "average"
    else:
        return "bad"

# 2. Let's apply the function
movies_df_title_indexed["Rating_Category"] = movies_df_title_indexed["Rating"].apply(rating_bucket)

# 3. Let's see some results
movies_df_title_indexed.head(10)[['Rating', 'Rating_Category']]
```

	Rating	Rating_Category
Title		
Guardians of the Galaxy	8.1	great
Prometheus	7.0	good
Split	7.3	good
Sing	7.2	good
Suicide Squad	6.2	average
The Great Wall	6.1	average
La La Land	8.3	great
Mindhorn	6.4	average
The Lost City of Z	7.1	good
Passengers	7.0	good

According to our rating method, *we can see that “Guardians of the Galaxy” and “La La Land” are great movies, while “Suicide Squad” is just an average movie!*






La La Land (Image Source: <https://www.slashfilm.com/la-la-land-review/>)

## Jupyter Notebook #

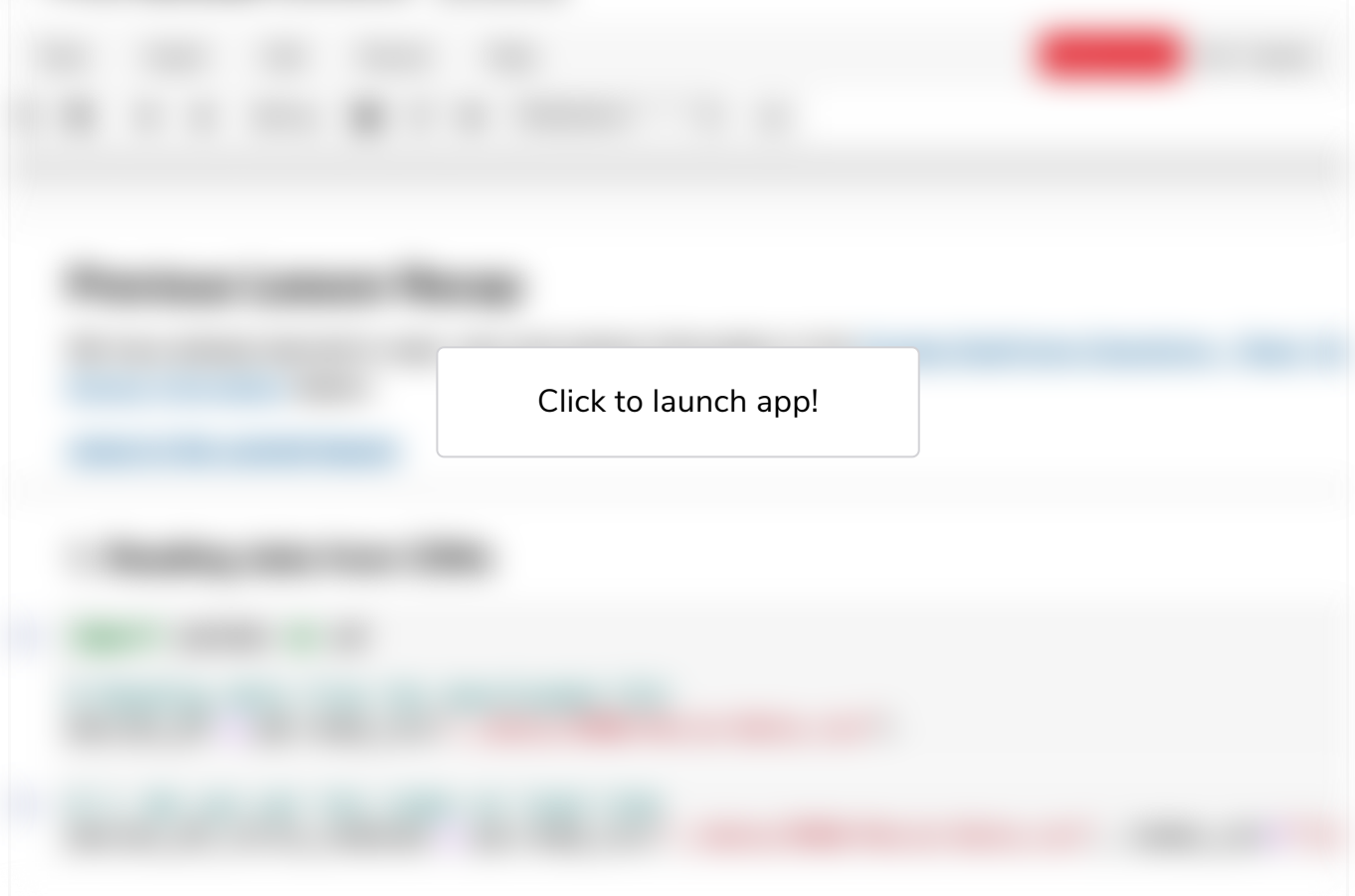
You can see the instructions running in the Jupyter Notebook below:

### 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 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/PivotTables%26Functions.ipynb>





## Final Thoughts #

**What a fun ride it has been!** From data exploration and data extraction to data transformation, we have learned so many Pandas magic tricks! And as a bonus, we have also gained many interesting movies-insights along the way!

Being well-versed in Pandas operation is one of the essential skills in data science. It's important to have a good grasp on these fundamentals. If you want to go further or learn more Pandas tricks, [here](#) is Pandas *extensive* official documentation.

*A handy cheat sheet and some exercises to embed these concepts in your long-term memory are awaiting you in the next lessons! Keep going — you are doing great! 🙌*