Pandas Walkthrough

In this walkthrough, you'll get to practice some of the concepts and skills covered in Pandas and Scikit Learn. You are provided with the dataset to use. Make sure you run the first line in order to be able to work with the data.

As you go through this notebook, certain places are left with double quotation marks (i.e "") or underscore (i.e _) or both (i.e "__"). To complete this assignment, you must replace all the "" with appropriate values, expressions, or statements to ensure that the notebook runs appropriately. Some things to keep in mind:

1. Make sure to run all the code cells, otherwise, you may get errors like `NameError` for undefined variables.
2. Do not change variable names, delete cells, or disturb other existing code.
3. In some cases, you may need to add some code cells or new statements before or after the line of code containing the ""

Introduction to Pandas

To make data analysis quick and simple in Python, Pandas includes high-level data structures and manipulation capabilities. Because pandas are built on top of NumPy, it is simple to use in applications that focus on NumPy. Pandas have two data structures called Series and Dataframe. A series is a 1D array while A DataFrame is a 2-dimensional data structure similar to a spreadsheet or a SQL table. To get started with pandas, you can install them using the pip install pandas command

To demonstrate the analytics capabilities of Pandas, we will be using the Google play store dataset. It is a publicly available dataset containing information such as app name, category, ratings, reviews, etc. We will see how Pandas can be used to derive actionable insights to caption the Android market. The data is stored in a Comma Separated Value (CSV) format. Pandas provide the `read_csv()` function to read data stored as a CSV file into a pandas DataFrame. Before using Pandas, we need to import it using the `import` keyword.

```
# import pandas
import pandas as pd

# read the data
playstore_df = pd.read_csv("Google_Playstore_cleaned.csv")
```

We can check the first and last N row(s) of the data using `.head()` and `.tail()`

```
# let's check the first 4 rows
playstore_df.head(4)
```

	AppName	AppId	Category	Rating	RatingCount	Installs	MinimumInstalls	MaximumInstalls	Free	Price	...	DeveloperW
0	Gakondo	com.ishakwe.gakondo	Adventure	0.0	0.0	10	10.0	15	True	0.0	...	https://beniyizibyo
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4.4	64.0	5000	5000.0	7662	True	0.0	...	https://webserveis.netli
2	Vibook	com.doantiepvien.crm	Productivity	0.0	0.0	50	50.0	58	True	0.0	...	
3	Smart City Trichy Public Service Vehicles 17UC...	cst.stJoseph.ug17ucs548	Communication	5.0	5.0	10	10.0	19	True	0.0	...	http://www.climatesmarttec

4 rows × 24 columns

```
# let's check the last 6 rows
playstore_df.tail(6)
```

	AppName	AppId	Category	Rating	RatingCount	Installs	MinimumInstalls	MaximumInstalls	Free	Price	...	D
399942	Learn Bhojpuri. Speak Bhojpuri. Study Bhojpuri.	com.pronunciatorllc.bluebird.bhojpuri	Education	0.0	0.0	1000	1000.0	1370	True	0.0	...	https://bluebi
399943	Parcelist- Easy package pickup	com.parcelist.StoreFrontMobile	Business	0.0	0.0	10	10.0	19	True	0.0	...	htt
399944	Spektrum DXe Programmer	org.as3x.programmertk	Sports	2.0	279.0	10000	10000.0	29242	True	0.0	...	http://www.l
399945	Drawchemy, abstract drawing	draw.chemy	Video Players & Editors	4.2	133.0	10000	10000.0	39189	True	0.0	...	http://drawc
399946	Princess Christmas Shopping	air.fizizi.PrincessChristmasShopping	Role Playing	3.8	986.0	100000	100000.0	429715	True	0.0	...	http://www.fizi
399947	DeFi Loans	com.ledgerblocks	Finance	0.0	0.0	50	50.0	52	True	0.0		

Indexing, Sorting and Filtering

To select a subset of the data, we can use either indexing operator [], attribute operator ., and methods such as loc, iloc, at,iat etc. This can be used in combination with comparison operators to make more powerful selection and filtering.

```
# what is the total number of app category included in Playstore
total_cat = playstore_df["Category"].unique()
total_cat_no = len(total_cat)
print(f"The total number of app category in Google playstore is:
{total_cat_no}")
```

```
The total number of app category in Google playstore is: 48
```

```
# What category has the most app?
playstore_df["Category"].value_counts(ascending=False)
```

```
Education              41599
Music & Audio          27139
Tools                  24995
Business               24702
Entertainment          23774
Lifestyle              20524
Books & Reference      20151
Personalization        15503
Health & Fitness       14378
Productivity           13751
Shopping               13117
Food & Drink           12825
Travel & Local         11643
Finance                11273
Arcade                  9364
Puzzle                  8794
Casual                  8701
Communication           8273
Sports                  8199
Social                  7854
News & Magazines        7414
Photography             6216
Medical                 5452
```

```
# What are the number of apps with 4+ star rating

# we first slice the dataframe where the rating is greater four
four_star = playstore_df[playstore_df["Rating"] > 4.0]

# then we can get the names of the app by specifying the AppName column
four_star_app = four_star["AppName"]
four_star_app
```

```
1                                       Ampere Battery Info
3         Smart City Trichy Public Service Vehicles 17UC...
6                           unlimited 4G data prank free app
9                           Neon 3d Iron Tech Keyboard Theme
10                                       Dodge The Cars!
                                  ...
399938                                       Whatscan 2020
399939                                    Axar Gk In Gujarati
399940                                          Ant Runner
399941                              Valentine's Day Photo Frame.
399945                               Drawchemy, abstract drawing
Name: AppName, Length: 129817, dtype: object
```

```
# we can use the len() function to get the total number
print(f"There are {len(four_star_app)} apps with more than 4 star ratings")
```

```
There are 129817 apps with more tha 4 star ratings
```

```
#What are the top 4 choice apps for teenagers?
print('Top 4 choices of teens')

# we first get the apps rated "teen" using the loc method
teen_top = playstore_df.loc[(playstore_df.ContentRating == 'Teen')]

# then we can sort the rows based on maximum number of installations for
each app
teen_top.sort_values(by='MaximumInstalls',ascending=False).head(4)
```

```
Top 4 choices of teens
```

	AppName	AppId	Category	Rating	RatingCount	Installs	MinimumInstalls	MaximumInstalls	Free	Price	...	
167758	Google TV (previously Play Movies & TV)	com.google.android.videos	Video Players & Editors	4.0	1825673.0	5000000000	5.000000e+09	6156518915	True	0.0	...	http://suppo
304783	Instagram	com.instagram.android	Social	3.8	120206190.0	1000000000	1.000000e+09	3559871277	True	0.0	...	h
337821	Samsung Experience Service	com.samsung.android.mobileservice	Tools	4.2	184659.0	1000000000	1.000000e+09	1682763021	True	0.0	...	http:
65030	TikTok	com.zhiliaoapp.musically	Social	4.4	36446381.0	1000000000	1.000000e+09	1645811582	True	0.0	...	

4 rows × 24 columns

Q1: What are the top 5 apps prefered by teengers in the Adventure category?

```
print("These are the top 5 apps preferred by teens in Adventure category")

# get the dataframe with "adventure" category and "teen" content rating
top_adventure_cat = ___.loc[(playstore_df.Category == "") &
(playstore_df.ContentRating == "")]

# sort using maximum installation and display the result
___.sort_values(by="",ascending=___).head(5)
```

Data Aggregation with Pandas

We can gather and express the data in a summary form using a combination of aggregate functions in Pandas.

```
# What is the most expensive app in the playstore?

# get the row where the 'Price' column is maximum by using max() method
most_expensive = playstore_df[playstore_df["Price"] ==
playstore_df["Price"].max()]

# get the app name by using the "AppName" column and use iloc to get only
the first row
most_expensive_name = most_expensive["AppName"].iloc[0]

# get the price of the most expensive
most_expensive_price = playstore_df["Price"].max()

# display the info
print(f"The most expensive app in Google play store is
{most_expensive_name} with a price of {most_expensive_price} USD")
```

```
The most expensive app in Google play store is TEST EGY with a price of 399.99 USD
```

```
# What is the size of the market in terms of number of downloads and price?

# group the app using Category and get only the price Price and Installs
columns and get the sum
market_share = playstore_df.groupby("Category")[["Price",
"Installs"]].agg('sum')

# sort the resulting dataframe by using the Price and then Installs
market_share_order = market_share.sort_values(by=["Price", "Installs"],
ascending=[False,False])

#display the info
market_share_order
```

Category	Price	Installs
Education	6405.878595	1183919256
Books & Reference	3554.574362	712848162
Medical	3495.836324	121097798
Tools	3383.684770	7078722347
Business	1813.775821	490142382
Productivity	1723.779154	5214719720
Personalization	1654.922906	1804765145
Health & Fitness	1390.930736	1594484926
Sports	1304.344623	1245263759
Arcade	1118.051103	2426925869
Music & Audio	1099.541901	1845703178
Lifestyle	968.283831	1479555871
Travel & Local	779.798578	469163375
Entertainment	757.122674	4026468267
Adventure	699.797442	981825770
Finance	689.700000	1072475720
Educational	672.482138	895102254

What is the average rating per category?

```
# What is the average rating per category?

# group the app by category and get the mean for each of the category
avg_rating = playstore_df.groupby("Category").mean()

# sort the dataframe by using rating
avg_rating_sorted = avg_rating.sort_values(by="Rating", ascending=False)

# display the info
avg_rating_sorted
```

Category	Rating	RatingCount	Installs	MinimumInstalls	MaximumInstalls	Free	Price	AdSupported	InAppPurchases	EditorsChoice
Role Playing	3.327160	11299.995679	3.809769e+05	3.809769e+05	7.112104e+05	0.934242	0.288226	0.648697	0.458531	0.007701
Simulation	3.252073	10603.902348	5.539900e+05	5.539900e+05	9.479616e+05	0.970186	0.096695	0.868035	0.313783	0.001711
Casino	3.199664	6515.968680	1.856562e+05	1.856562e+05	3.251972e+05	0.986696	0.074847	0.732816	0.497783	0.000000
Weather	3.182825	4594.244783	3.335844e+05	3.335844e+05	6.318502e+05	0.964595	0.116872	0.632573	0.248623	0.000000
Card	3.125592	3896.514053	1.978952e+05	1.978952e+05	3.651855e+05	0.961286	0.166762	0.735573	0.327246	0.001461
Racing	2.982018	15929.188943	9.873820e+05	9.873820e+05	1.735831e+06	0.983658	0.044117	0.882446	0.299947	0.002109
Word	2.935051	4920.973559	1.955518e+05	1.955518e+05	3.329393e+05	0.977151	0.046243	0.872984	0.387769	0.002016
Video Players & Editors	2.903329	10055.832709	2.780672e+06	2.780672e+06	3.814640e+06	0.976053	0.094525	0.677539	0.092486	0.001239
Strategy	2.864918	20658.695960	5.262378e+05	5.262378e+05	1.143862e+06	0.934708	0.199438	0.644674	0.395876	0.008247
Adventure	2.845886	5439.322564	2.440531e+05	2.440531e+05	4.124490e+05	0.961472	0.173949	0.777529	0.204325	0.001491
Board	2.805753	5700.122740	3.285903e+05	3.285903e+05	5.486951e+05	0.961538	0.153012	0.708017	0.241062	0.000000
Personalization	2.764893	1418.502456	1.164139e+05	1.164139e+05	1.925693e+05	0.933948	0.106749	0.820293	0.045153	0.000000
Music	2.737989	6546.902235	4.135947e+05	4.135947e+05	9.763184e+05	0.978142	0.050328	0.812842	0.148907	0.001366
Comics	2.686465	1864.486869	1.544615e+05	1.544615e+05	2.581036e+05	0.984032	0.028735	0.736527	0.169661	0.000000
Puzzle	2.662500	5143.384722	2.381015e+05	2.381015e+05	4.257037e+05	0.976006	0.058941	0.826700	0.239254	0.001478

Q2: What is the least installed app?

```
# get the row where the 'Installs' column is minimum by using min() method
least_installed = playstore_df[___[""] == playstore_df[""].min()]

# get the app name by using the "AppName" column and use iloc to get only
the first row
least_installed_name = least_installed[""].iloc[]

# get the number of installation for the app
least_installed = playstore_df["Installs"].___

# display the information
print(f"The most expensive app in Google play store is
{least_installed_name} with {least_installed} number of installation")
```

Quick Analysis with Pivot table

With just a few lines of codes, we can use pivot tables to drill down into the granular details of our data. Inorder to demonstrate this, we'll be using a subset of our data.

```
# create a subset of the data
playstore_subset_df = playstore_df[['Category', 'Rating', 'ContentRating',
'Installs', 'Free', 'Price']]

# now let's create our pivot table
rating_table = playstore_subset_df.pivot_table(index = "ContentRating")

# display the info
rating_table
```

ContentRating	Free	Installs	Price	Rating
Adults	1.000000	88378.304348	0.000000	2.404545
Everyone	0.979424	146554.309885	0.098337	2.183311
Mature 17+	0.988712	252242.398988	0.059109	2.486494
Teen	0.987867	552974.112397	0.073918	2.310962

With this, we can see the average Installs, Price, Rating, and Free apps based on their rating

We can as well use multiple index to answer some questions

```
# What is the average installation, Price, and rating based on Content
rating and Type of the app?

# create a pivot table
rating_table_expanded = playstore_subset_df.pivot_table(index =
["ContentRating", "Free"])

# display the table
rating_table_expanded
```

ContentRating	Free	Installs	Price	Rating
Adults	True	88378.304348	0.000000	2.404545
Everyone	False	8083.384037	4.779198	2.301933
	True	149463.339563	0.000000	2.180802
Mature 17+	False	22822.000000	5.236237	2.776724
	True	254861.765748	0.000000	2.483150
Teen	False	20197.300242	6.092419	2.867073
	True	559517.569245	0.000000	2.304101

we can also customize the type of aggregation performed on different features

```
# we need some functions from numpy so we have to import it
import numpy as np

# What is the total number of installations and the average price per
category?

cat_pivot = playstore_subset_df.pivot_table(index = "Category", aggfunc =
{"Installs":np.sum,
                                            "Price":np.mean})

cat_pivot
```

Category	Installs	Price
Action	4372708361	0.084610
Adventure	981825770	0.173949
Arcade	2426925869	0.119399
Art & Design	147477778	0.114630
Auto & Vehicles	83641623	0.100793
Beauty	50857400	0.007162
Board	606577686	0.153012
Books & Reference	712848162	0.176397
Business	490142382	0.073426
Card	270918557	0.166762
Casino	167461849	0.074847
Casual	4204390113	0.038697
Comics	77385229	0.028735
Communication	7239640969	0.036976
Dating	87255461	0.055689
Education	1183919256	0.153991
Educational	895102254	0.187112
Entertainment	4026468267	0.031847
Events	17967298	0.002679

Q3: Using Pivot table, find the average rating and highest price per category

```
# create the table using np.mean on Rating and np.max on Price
price_rating_pivot = playstore_subset_df.___(index = "___", aggfunc =
{"Rating":___,
                                            "___":np.max})

# display the table
price_rating_pivot
```

Time series with Pandas

We can leverage the power of Pandas to derive insights from our data based on time orientation. In order to do this, we are going to take a subset of the original dataset.

```
# extract a subset of the data
playstore_time_df = playstore_df[["Released", "Rating", "Price", "Installs"]]

# let's convert the Released column to a datetime type
playstore_time_df["Released"] = pd.to_datetime(playstore_time_df['Released'],
                                infer_datetime_format=True, errors='coerce')

# let's set the date as the index of the dataframe
playstore_time_df = playstore_time_df.set_index("Released")

# display the first five rows
playstore_time_df.head(5)
```

Released	Rating	Price	Installs
2020-02-26	0.0	0.0	10
2020-05-21	4.4	0.0	5000
2019-08-09	0.0	0.0	50
2018-09-10	5.0	0.0	10
2020-02-21	0.0	0.0	100

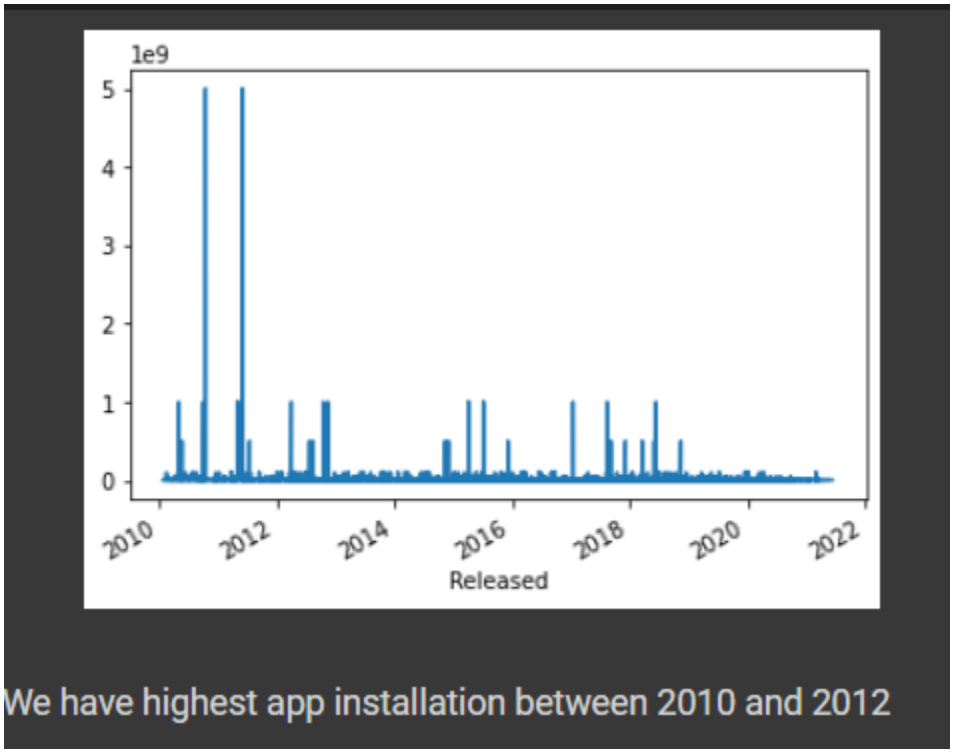
we can do a couple of other things such as extracting the year, month, and weekday name

```
playstore_time_df["Year"] = playstore_time_df.index.year
playstore_time_df["Month"] = playstore_time_df.index.month
playstore_time_df["MonthName"] = playstore_time_df.index.month_name()
playstore_time_df["WeekDayName"] = playstore_time_df.index.day_name()

# display the first five rows

playstore_time_df.head()
```

Released	Rating	Price	Installs	Year	Month	MonthName	WeekDayName
2020-02-26	0.0	0.0	10	2020.0	2.0	February	Wednesday
2020-05-21	4.4	0.0	5000	2020.0	5.0	May	Thursday
2019-08-09	0.0	0.0	50	2019.0	8.0	August	Friday
2018-09-10	5.0	0.0	10	2018.0	9.0	September	Monday
2020-02-21	0.0	0.0	100	2020.0	2.0	February	Friday

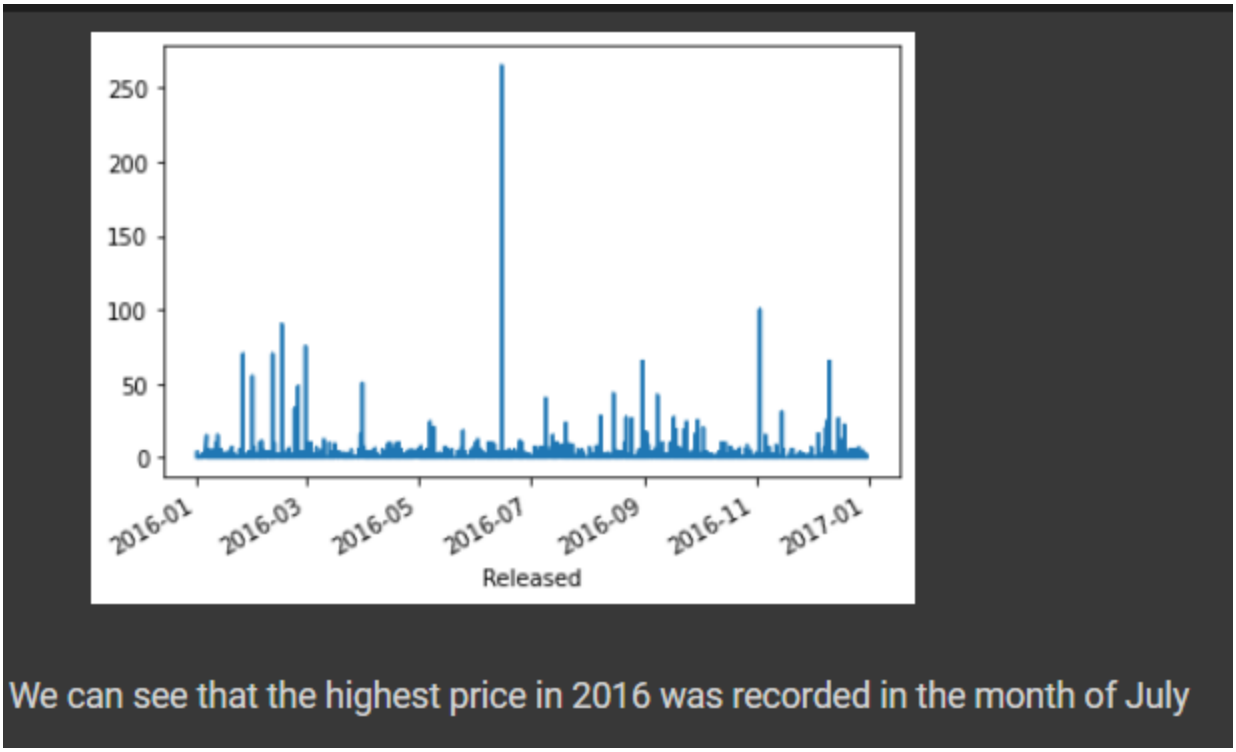
What is the trend of App installation over the years?
First, before we plot our charts, we'll need to import Matplotlib which is a Charting library in Python

```
# import matplotlib for plotting
import matplotlib.pyplot as plt

# What is the year with most app installation
playstore_time_df["Installs"].plot()

#dsiplay the chart
plt.show()
```



We can zoom in into a particular year, let's see the price of apps in the year 2016.

```
playstore_time_df.loc["2016", "Price"].plot()

#display the chart
plt.show()
```



Q4: What is the Installation trend in 2020? Dispaly using a line chart.

```
___.loc["__", "__"].plot()

#display the chart
plt.show()
```

Using Functions in Pandas

For the purpose of organization and reusability, we can write our own custom functions in Pandas

Let's write a function that determines the proportion of free to paid apps

```
def get_app_proportion(df, col_name):

    # get the number of free app
    free_app = len(df[df[col_name] == True])

    # get the number of paid app
    paid_app = len(df[df[col_name] == False])

    # get the total number of paid and free apps
    total_app = free_app + paid_app

    # calculate the proportion of free and paid app
    free_prop = round((free_app/total_app)*100)
    paid_prop = round((paid_app/total_app)*100)

    # display the info
    print(f"The proportion of free to paid app is {free_prop}:{paid_prop}")

# call the function
get_app_proportion(df = playstore_df, col_name = "Free")
```

We can write another function to determine the percentage market share of each app by category?

```
def market_share_pct(df):

    # first group by category and take the sum of Installs
    mkt_share = df.groupby("Category").agg('sum')["Installs"]

    # take the percentage of every row using transform
    mkt_share_pct = mkt_share.transform(lambda x:x/x.sum())*100

    # transfrom to frame and sort using Install
    mkt_share_pct_sorted =
mkt_share_pct.to_frame().sort_values(by="Installs",   ascending=False)

    return mkt_share_pct_sorted

# call the function
market_share_pct(playstore_df)
```

Q5: what is the percentage market share of each app by content rating?

```
# name the function as get_market_share
def ___(df):

    # first group by ContentRating and take the sum of Installs
    share_by_rating = __.groupby("___").agg('sum')["Installs"]

    # take the percentage of every row using transform
    share_by_rating_pct = share_by_rating.___(lambda x:x/x.sum())*100

    # transfrom to frame and sort using Install
    share_by_rating_pct_sorted = ___.___().sort_values(by="___",
ascending=False)

    return ___

# call the function
get_market_share(playstore_df)
```