Practice - PANDASMoviesCities

October 7, 2024

1 PANDAS PRACTICE 2

1.1 Name : Ramya Chowdary Patchala

1.1.1 In this practice we will look at weather data from various cities and see how groupby can be used to run some analytics. Add code cells where applicable.

[1]:
```
import pandas as pd
```

Question 1

Let us explore the movie dataset

1. Load in the IMDB movies dataset
2. Display the top 5 and last 5 movies and columns
3. Display information about the columns. What are the datatypes?
4. Append the datframe to itself
5. Display the shape of the dataframe
6. Remove the duplicates
7. Confirm that the shape has been modified

[10]:
```
# Loading IMDB Movies dataset
movies_df = pd.read_csv("IMDB-Movie-Data.csv")

# Displaying first five columns of movies dataset
movies_df.head()
```

[10]:

	Rank	Title	Genre	\
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	
1	2	Prometheus	Adventure,Mystery,Sci-Fi	
2	3	Split	Horror,Thriller	
3	4	Sing	Animation,Comedy,Family	
4	5	Suicide Squad	Action,Adventure,Fantasy	

	Description	Director	\
0	A group of intergalactic criminals are forced …	James Gunn	
1	Following clues to the origin of mankind, a te…	Ridley Scott	
2	Three girls are kidnapped by a man with a diag…	M. Night Shyamalan	

```
3  In a city of humanoid animals, a hustling thea…  Christophe Lourdelet
4  A secret government agency recruits some of th…            David Ayer

                                              Actors  Year  Runtime (Minutes)  \
0  Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S…  2014                121
1  Noomi Rapace, Logan Marshall-Green, Michael Fa…  2012                124
2  James McAvoy, Anya Taylor-Joy, Haley Lu Richar…  2016                117
3  Matthew McConaughey,Reese Witherspoon, Seth Ma…  2016                108
4  Will Smith, Jared Leto, Margot Robbie, Viola D…  2016                123

   Rating   Votes  Revenue (Millions)  Metascore
0     8.1  757074              333.13       76.0
1     7.0  485820              126.46       65.0
2     7.3  157606              138.12       62.0
3     7.2   60545              270.32       59.0
4     6.2  393727              325.02       40.0
```

[11]:
```
# Displaying last five columns of dataset
movies_df.tail()
```

[11]:
```
     Rank                   Title                   Genre  \
995   996    Secret in Their Eyes    Crime,Drama,Mystery
996   997         Hostel: Part II                  Horror
997   998  Step Up 2: The Streets    Drama,Music,Romance
998   999            Search Party       Adventure,Comedy
999  1000              Nine Lives  Comedy,Family,Fantasy

                                          Description          Director  \
995  A tight-knit team of rising investigators, alo…        Billy Ray
996  Three American college students studying abroa…         Eli Roth
997  Romantic sparks occur between two dance studen…        Jon M. Chu
998  A pair of friends embark on a mission to reuni…    Scot Armstrong
999  A stuffy businessman finds himself trapped ins…  Barry Sonnenfeld

                                            Actors  Year  \
995  Chiwetel Ejiofor, Nicole Kidman, Julia Roberts…  2015
996  Lauren German, Heather Matarazzo, Bijou Philli…  2007
997  Robert Hoffman, Briana Evigan, Cassie Ventura,…  2008
998  Adam Pally, T.J. Miller, Thomas Middleditch,Sh…  2014
999  Kevin Spacey, Jennifer Garner, Robbie Amell,Ch…  2016

     Runtime (Minutes)  Rating  Votes  Revenue (Millions)  Metascore
995                111     6.2  27585                NaN       45.0
996                 94     5.5  73152              17.54       46.0
997                 98     6.2  70699              58.01       50.0
998                 93     5.6   4881                NaN       22.0
999                 87     5.3  12435              19.64       11.0
```

[14]:
```
# Displaying the columns and their datatypes
movies_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Rank                1000 non-null   int64
 1   Title               1000 non-null   object
 2   Genre               1000 non-null   object
 3   Description         1000 non-null   object
 4   Director            1000 non-null   object
 5   Actors              1000 non-null   object
 6   Year                1000 non-null   int64
 7   Runtime (Minutes)   1000 non-null   int64
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64
 10  Revenue (Millions)  872 non-null    float64
 11  Metascore           936 non-null    float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

[22]:
```
# Appending dataframe to itself
appended_df = pd.concat([movies_df, movies_df], ignore_index=True)
appended_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Rank                2000 non-null   int64
 1   Title               2000 non-null   object
 2   Genre               2000 non-null   object
 3   Description         2000 non-null   object
 4   Director            2000 non-null   object
 5   Actors              2000 non-null   object
 6   Year                2000 non-null   int64
 7   Runtime (Minutes)   2000 non-null   int64
 8   Rating              2000 non-null   float64
 9   Votes               2000 non-null   int64
 10  Revenue (Millions)  1744 non-null   float64
 11  Metascore           1872 non-null   float64
dtypes: float64(3), int64(4), object(5)
memory usage: 187.6+ KB
```

[23]:
```
# Shape of dataframe
movies_df.shape, appended_df.shape
```

[23]:
```
((1000, 12), (2000, 12))
```

[26]:
```
# Removing duplicates
appended_df = appended_df.drop_duplicates()
# Shape has changed back after removing duplicates
appended_df.shape
```

[26]:
```
(1000, 12)
```

Question 2

Let us explore another dataset. This time the weather dataset

1. Create the data frame from the given csv file
2. Display the first 10 rows
3. Display the last 5 rows
4. Display the datatypes
5. Display statistics for a numerical column

[29]:
```
# Loading the weather dataset
weather_df = pd.read_csv('weather_by_cities.csv')

# Displaying the first 10 rows
weather_df.head(10)
```

[29]:

	day	city	temperature	windspeed	event
0	1/1/2017	new york	32	6	Rain
1	1/2/2017	new york	36	7	Sunny
2	1/3/2017	new york	28	12	Snow
3	1/4/2017	new york	33	7	Sunny
4	1/1/2017	mumbai	90	5	Sunny
5	1/2/2017	mumbai	85	12	Fog
6	1/3/2017	mumbai	87	15	Fog
7	1/4/2017	mumbai	92	5	Rain
8	1/1/2017	paris	45	20	Sunny
9	1/2/2017	paris	50	13	Cloudy

[30]:
```
# Displaying the last five rows
weather_df.tail()
```

[30]:

	day	city	temperature	windspeed	event
7	1/4/2017	mumbai	92	5	Rain
8	1/1/2017	paris	45	20	Sunny
9	1/2/2017	paris	50	13	Cloudy

```
 10  1/3/2017   paris           54          8  Cloudy
 11  1/4/2017   paris           42         10  Cloudy
```

[40]:
```
# Displaying columns and datatypes
weather_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          12 non-null     object
 1   city         12 non-null     object
 2   temperature  12 non-null     int64
 3   windspeed    12 non-null     int64
 4   event        12 non-null     object
dtypes: int64(2), object(3)
memory usage: 612.0+ bytes
```

[32]:
```
# Displaying statistics for numerical columns
weather_df.describe()
```

[32]:

	temperature	windspeed
count	12.000000	12.000000
mean	56.166667	10.000000
std	25.044808	4.572646
min	28.000000	5.000000
25%	35.250000	6.750000
50%	47.500000	9.000000
75%	85.500000	12.250000
max	92.000000	20.000000

Question 3 For this dataset let us determine the following. We will explore splitting your dataset in smaller groups and then applying an operation (such as min or max) to get aggregate result is called Split-Apply-Combine approach.

[33]:
```
# Groupby city and print the data for all the groups
# Get the data group for Mumbai
# Get the max temp for all cities
# What is the avarage temperature and windspeed
# Display all the analytics for the data
# Let us do a rudimentary plot. See code below
'''
%matplotlib inline # load matplotlib
variable.plot()

'''
```

[33]:
```
'\n%matplotlib inline # load matplotlib\nvariable.plot()\n\n'
```

[43]:
```
# Groupby city and displaying groups
city_df = weather_df.groupby('city')

for city, group in city_df:
    print(f"\nCity: {city}")
    print(group)
```

```
City: mumbai
        day    city  temperature  windspeed  event
4  1/1/2017  mumbai           90          5  Sunny
5  1/2/2017  mumbai           85         12    Fog
6  1/3/2017  mumbai           87         15    Fog
7  1/4/2017  mumbai           92          5   Rain

City: new york
        day      city  temperature  windspeed  event
0  1/1/2017  new york           32          6   Rain
1  1/2/2017  new york           36          7  Sunny
2  1/3/2017  new york           28         12   Snow
3  1/4/2017  new york           33          7  Sunny

City: paris
         day   city  temperature  windspeed   event
8   1/1/2017  paris           45         20   Sunny
9   1/2/2017  paris           50         13  Cloudy
10  1/3/2017  paris           54          8  Cloudy
11  1/4/2017  paris           42         10  Cloudy
```

[44]:
```
# Data group for mumbai
mumbai_df = city_df.get_group('mumbai')
print(mumbai_df)
```

```
        day    city  temperature  windspeed  event
4  1/1/2017  mumbai           90          5  Sunny
5  1/2/2017  mumbai           85         12    Fog
6  1/3/2017  mumbai           87         15    Fog
7  1/4/2017  mumbai           92          5   Rain
```

[45]:
```
# Get max temp for all cities
max_temp = city_df['temperature'].max()
print(max_temp)
```

```
city
mumbai      92
new york    36
```

```
paris       54
Name: temperature, dtype: int64
```

[48]:
```
# average temperature and windspeed
avg_temp = city_df['temperature'].mean()
avg_windspeed = city_df['windspeed'].mean()

print("Average Temperature: " , avg_temp)
print("\n Average Windspeed: ",avg_windspeed)
```

```
Average Temperature:  city
mumbai      88.50
new york    32.25
paris       47.75
Name: temperature, dtype: float64

 Average Windspeed:  city
mumbai       9.25
new york     8.00
paris       12.75
Name: windspeed, dtype: float64
```

[50]:
```
# Display analytics for full weather data
weather_df.describe()
```

[50]:

	temperature	windspeed
count	12.000000	12.000000
mean	56.166667	10.000000
std	25.044808	4.572646
min	28.000000	5.000000
25%	35.250000	6.750000
50%	47.500000	9.000000
75%	85.500000	12.250000
max	92.000000	20.000000

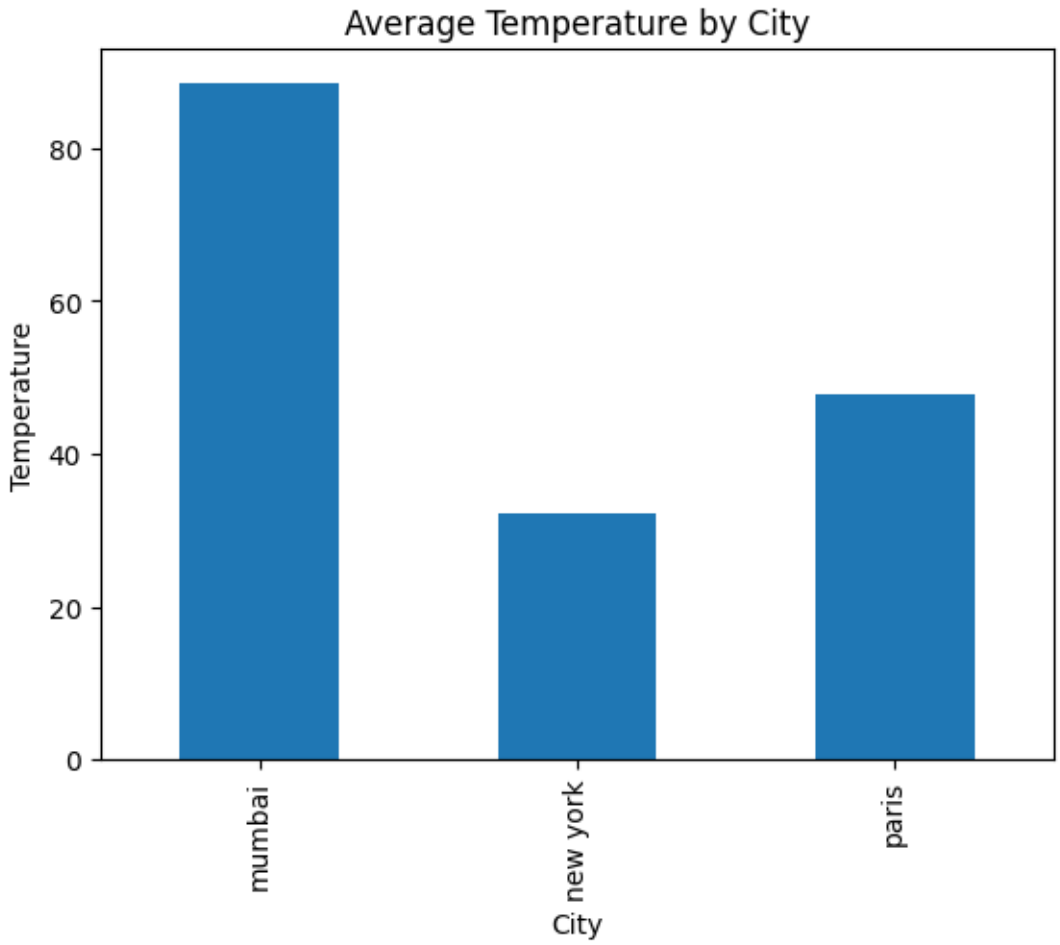
[51]:
```
# Display analytics for city wise weather data
city_df.describe()
```

[51]:

	temperature								windspeed							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
city																
mumbai	4.0	88.50	3.109126	85.0	86.50	88.5	90.50	92.0								
new york	4.0	32.25	3.304038	28.0	31.00	32.5	33.75	36.0								
paris	4.0	47.75	5.315073	42.0	44.25	47.5	51.00	54.0								

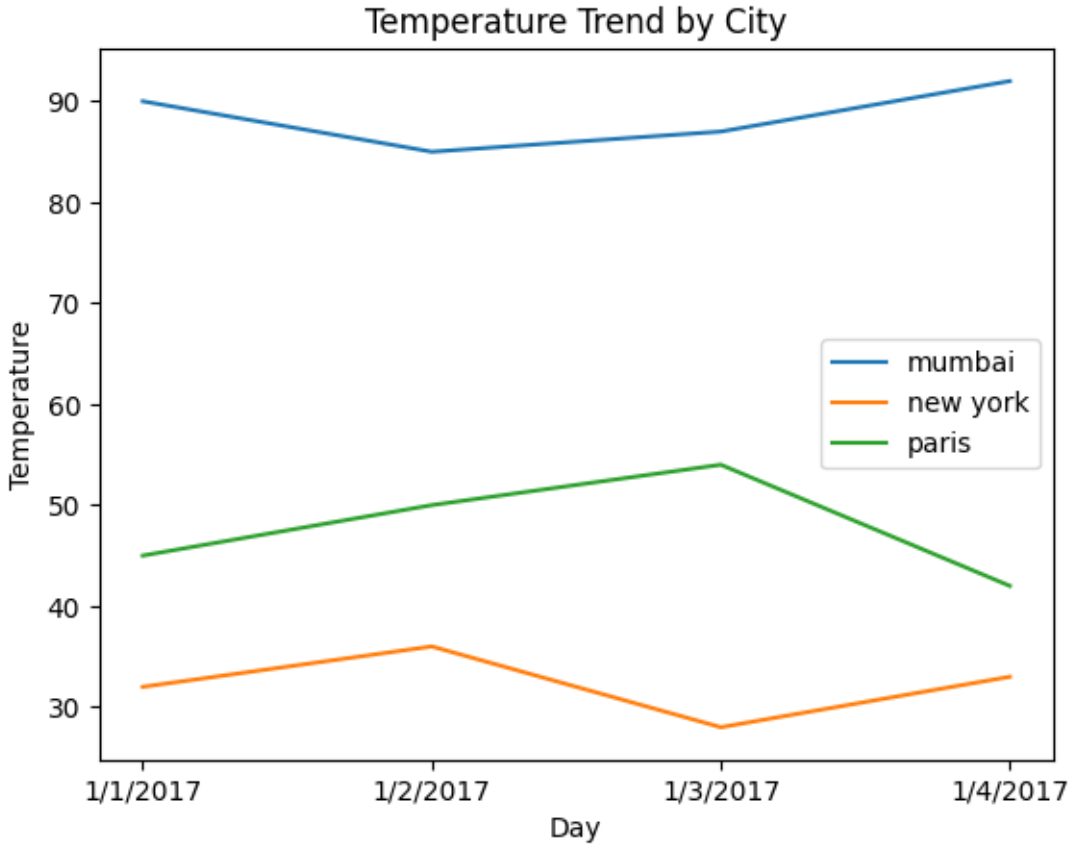
```
mumbai          4.0   9.25  5.057997  5.0  5.00   8.5  12.75  15.0
new york        4.0   8.00  2.708013  6.0  6.75   7.0   8.25  12.0
paris           4.0  12.75  5.251984  8.0  9.50  11.5  14.75  20.0
```

[52]:
```
import matplotlib.pyplot as plt

# Plot avg temperature for each city
city_df['temperature'].mean().plot(kind='bar')

plt.title('Average Temperature by City')
plt.xlabel('City')
plt.ylabel('Temperature')
plt.show()
```

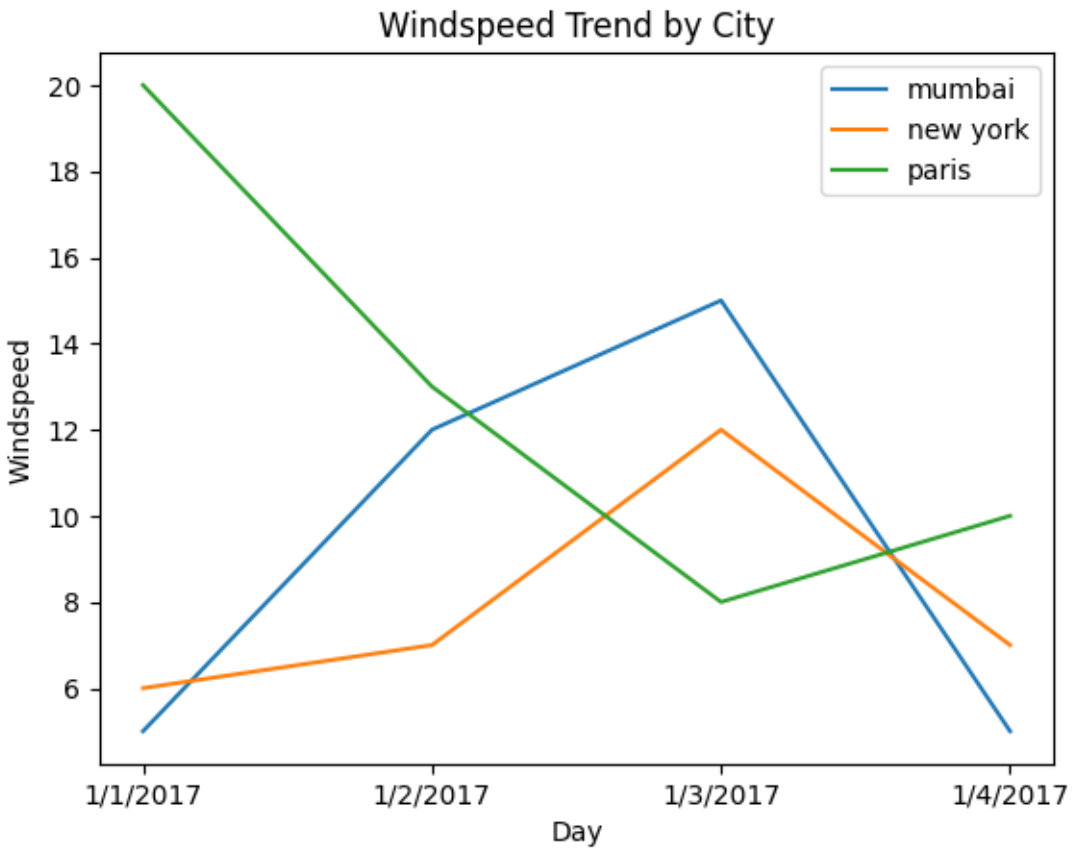


[53]:
```
# Plot temperature trends for each city
for city, group in city_df:
    plt.plot(group['day'], group['temperature'], label=city)
```

```
plt.xlabel('Day')
plt.ylabel('Temperature')
plt.title('Temperature Trend by City')
plt.legend()
plt.show()
```



[54]:
```
# Plot windspeed trends for each city
for city, group in city_df:
    plt.plot(group['day'], group['windspeed'], label=city)

plt.xlabel('Day')
plt.ylabel('Windspeed')
plt.title('Windspeed Trend by City')
plt.legend()
plt.show()
```


Windspeed Trend by City
mumbai
new york
paris
Windspeed
20
18
16
14
12
10
8
6
1/1/2017
1/2/2017
1/3/2017
1/4/2017
Day