Section 2: Customer Activity Analysis

In this lesson, the customers activity data throughout the month will be analyzed.

We'll cover the following

Users activity

As mentioned in the previous lesson, the user can perform three actions that get recorded in the dataset.

- **view**: The user can view an item.
- **cart**: The user can add the item to the cart.
- **purchase**: The user can purchase the item.

Analyzing the *view* and *purchasing* actions of the user across the different timelines in a month can provide very important information as to at what time most of the users visit the site. When such times are known, resources can be allocated according to that information to optimize performance.

For example, if we know that a significant amount of users visit the site on *Sunday* just to view the products, resources from other components can be transferred to viewing components to enhance the user experience. Similarly, the same approach can be used on other components if we know at what times certain, user activity is preferred.

Let’s apply this approach to our data and review what analysis can be drawn from it.

Preprocessing

Before we move to extract information, some preprocessing needs to be done on our initial `DataFrame`. The time values are separated from the `event_time` column and are made into separate columns. The `day`, `week_day`, and `hour` are computed for each `event_time` value.

```
import pandas as pd

df = pd.read_csv("2019-Oct.csv") # Reading the data from file

#Convert the type of event_time column to datetime
df['event_time'] = pd.to_datetime(df.event_time)

# Calculate and add relevant columns to track users activity
df["week_day"] = df['event_time'].map(lambda x: x.dayofweek + 1)
df["day"] = df['event_time'].map(lambda x: x.day)
df["hour"] = df['event_time'].map(lambda x: x.hour)

print(df)
```

event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session	week_day	day	hour
2019-10-01 :00:00+00:00	view	44600062	2103807459595387724	NaN	shiseido	35.79	541312140	72d76fde-8bb3-4e00-8c23-a032dfed738c	2	1	0
2019-10-01 :00:00+00:00	view	3900821	2053013552326770905	appliances.environment.water_heater	aqua	33.20	554748717	9333dfbd-b87a-4708-9857-6336556b0fcc	2	1	0
2019-10-01 :00:01+00:00	view	17200506	2053013559792632471	furniture.living_room.sofa	NaN	543.10	519107250	566511c2-e2e3-422b-b695-cf8e6e792ca8	2	1	0
2019-10-01 :00:01+00:00	view	1307067	2053013558920217191	computers.notebook	lenovo	251.74	550050854	7c90fc70-0e80-4590-96f3-13c02c18c713	2	1	0
								c6bd7419-			

On **line 6**, the `event_time` column is converted to *DateTime* data type using the `to_datatime` function of `pandas`. The respective `event_time` column is provided as the parameter to this function. This allows us to extract the required information using the `DateTime` built-in functions.

On **lines 9-11**, three new columns are created and assigned relevant values using the *pandas* `map()` function and the time functions of the `DateTime` package.

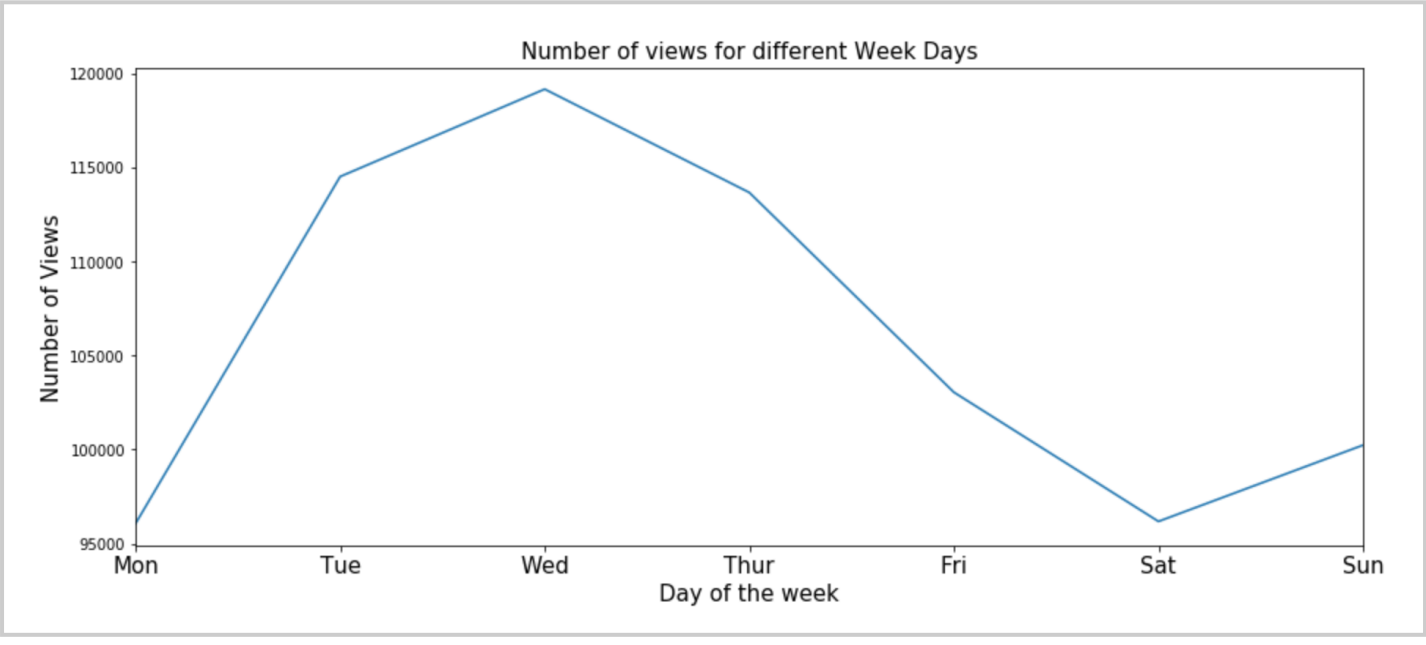
- The `day` column stores the daily values starting from 1 to 31, in case of *October*.
- The `week_day` column stores which day of the week it is, from *1 to 7*, as *Monday* to *Sunday*.

- The `hour` column stores every hour value of every day starting from 0-23, denoting what hour it is at that moment.

Weekly analysis

In this part, we will review a weekly analysis of the number of views. This will reveal the day of the week on which the most or least number of views occur for the website.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("2019-Oct.csv") # Reading the data from file

#Convert the type of event_time column to datetime
df['event_time'] = pd.to_datetime(df.event_time)
# Calculate and add relevant columns to track users activity
df["week_day"] = df['event_time'].map(lambda x: x.dayofweek + 1)
df["day"] = df['event_time'].map(lambda x: x.day)
df["hour"] = df['event_time'].map(lambda x: x.hour)

# Get all the view events of all users
viewed = df[df['event_type'] == 'view']

# Plot the number views against all week days in a line chart
view_plot = viewed.groupby('event_type')['week_day'].value_counts().sort_index().plot(kind = 'line

# Set properties of the plot
view_plot.set_xlabel('Day of the week',fontsize = 15)
view_plot.set_ylabel('Number of Views',fontsize = 15)
view_plot.set_title('Number of views for different Week Days',fontsize = 15)
view_plot.set_xticklabels(('Mon','Tue','Wed','Thur','Fri', 'Sat','Sun'), rotation = 'horizontal', 

#plot the graph
plt.show()
```



In the above graph, it can be observed that most items are viewed during the working days instead of on the weekends. This represents the aggregated number of website views for all the weekdays of *October 2019*.

On **line 13**, the products whose `event_type` was equal to `view` are fetched and are stored in another `DataFrame` for further analysis.

On **line 16**, a lot of functions are being called. First, the `DataFrame` is grouped on the `event_type` and then aggregated on the count of `week_days`. This produces a `Series` that assigns the number of views to each weekday. The `sort_index()` function sorts the number of days in ascending order. Finally, this information is plotted using the `plot()` function. The plot is a `line` plot, as mentioned in the `kind` parameter.

On **lines 19-22**, the different properties of the plot are being set.

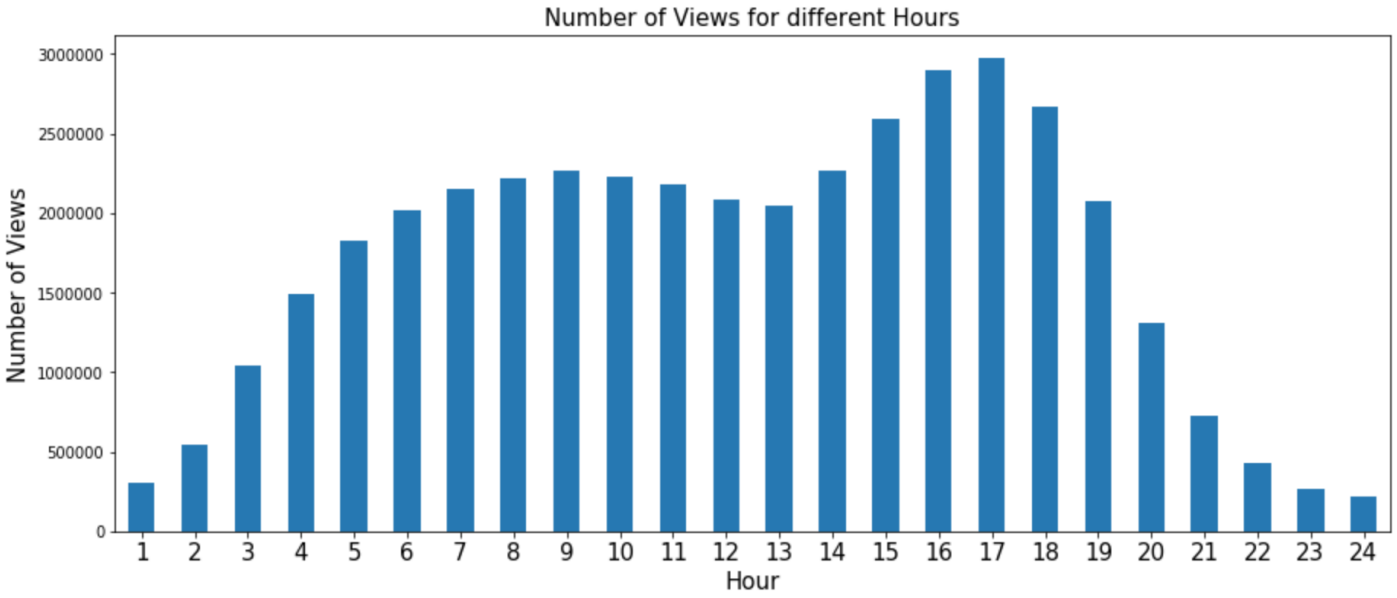
On **line 25**, the plot is displayed using the `plt.show()` function.

Hourly analysis

In this part, an hourly analysis of the number of views will be created. This will reveal at which hour of the day the most and least number of views occur for the website.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("2019-Oct.csv") # Reading the data from file

#Convert the type of event_time column to datetime
df['event_time'] = pd.to_datetime(df.event_time)

df["week_day"] = df['event_time'].map(lambda x: x.dayofweek + 1)
df["day"] = df['event_time'].map(lambda x: x.day)
df["hour"] = df['event_time'].map(lambda x: x.hour)

# Get all the view events of all users
viewed = df[df['event_type'] == 'view']

# Plot the number views against all 24 hours of the days in a bar chart
view_plot = viewed.groupby('event_type')['hour'].value_counts().sort_index().plot(kind = 'bar', fi

# Set properties of the plot
view_plot.set_xlabel('Hour',fontsize = 15)
view_plot.set_ylabel('Number of Views',fontsize = 15)
view_plot.set_title('Number of views for different Hours of Days',fontsize = 15)
view_plot.set_xticklabels(range(1,32), rotation='horizontal', fontsize=15)
```

```
#plot the graph
plt.show()
```



In the above graph, it can be observed that most items are viewed in the working hours instead of the free hours. The number of views starts increasing from the start of the day, reaching their peak between **3 and 5 P.M**. Then it starts to drop. This is the combined result for each day of *October* 2019.

The code is exactly the same for the weekly analysis. On **line 16**, just the `week_day` parameter is changed using the `hour` parameter, and some properties are renamed according to the new analysis.

The hypothesis

From the above weekly and hourly analysis, it can be observed that most of the users like to browse the items during working hours of working days. Other time slots are also important but at these time slots, most resources should be allocated to the viewing or browsing component of the website to optimize and enhance user experience which in turn brings profit.

Try doing the same weekly and hourly analysis for the number of products purchased to determine whether the `view` results hold for the `purchase` part or not.

In the next lesson, insights about brand and categories are discussed.