Assignment 4

Details

1. Author : Varad Mashalkar
2. Roll Number : 33337
3. Batch : M11
4. Class : TE11

Problem Statement

Perform the following operations using Python on the Facebook metrics data sets

1. Create data subsets
2. Merge Data
3. Sort Data
4. Transposing Data
5. Shape and reshape Data

Implementation details

1. Dataset URL : [https://archive.ics.uci.edu/ml/datasets/Facebook+metrics (https://archive.ics.uci.edu/ml/datasets/Facebook+metrics)](https://archive.ics.uci.edu/ml/datasets/Facebook+metrics)
2. Python version : 3.7.4
3. Imports :
   A. pandas
   B. numpy
   C. matplotlib
   D. seaborn
4. conda environment : base

Dataset details

1. Given dataset is a representative of some of the Facebook metrics which are assosciated with the posts on social media.
2. These metrics are indicative of the engagement of the users with the corresponding post.
3. It includes various types of posts and their details

In [1]:

```
!python --version
```

```
Python 3.7.4
```

Importing required libraries

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline
```

Reading the dataset

In [3]:

```
# Reading the dataset
dataset = pd.read_csv("./dataset_Facebook.csv", sep=";")
dataset.head()
```

Out[3]:

	Page total likes	Type	Category	Post Month	Post Weekday	Post Hour	Paid	Lifetime Post Total Reach	Lifetime Post Total Impressions	Lifetime Engaged Users	C
0	139441	Photo	2	12	4	3	0.0	2752	5091	178	
1	139441	Status	2	12	3	10	0.0	10460	19057	1457	
2	139441	Photo	3	12	3	3	0.0	2413	4373	177	
3	139441	Photo	2	12	2	10	1.0	50128	87991	2211	
4	139441	Photo	2	12	2	3	0.0	7244	13594	671	

Dataset metadata

In [4]:

```
# Shape of the dataset
dataset.shape
```

Out[4]:

(500, 19)

In [5]:

```
dataset.describe(include="all")
```

Out[5]:

	Page total likes	Type	Category	Post Month	Post Weekday	Post Hour	Paid	Lif T
count	500.000000	500	500.000000	500.000000	500.000000	500.000000	499.000000	
unique	NaN	4	NaN	NaN	NaN	NaN	NaN	
top	NaN	Photo	NaN	NaN	NaN	NaN	NaN	
freq	NaN	426	NaN	NaN	NaN	NaN	NaN	
mean	123194.176000	NaN	1.880000	7.038000	4.150000	7.840000	0.278557	13
std	16272.813214	NaN	0.852675	3.307936	2.030701	4.368589	0.448739	22
min	81370.000000	NaN	1.000000	1.000000	1.000000	1.000000	0.000000	
25%	112676.000000	NaN	1.000000	4.000000	2.000000	3.000000	0.000000	3
50%	129600.000000	NaN	2.000000	7.000000	4.000000	9.000000	0.000000	5
75%	136393.000000	NaN	3.000000	10.000000	6.000000	11.000000	1.000000	13
max	139441.000000	NaN	3.000000	12.000000	7.000000	23.000000	1.000000	180

In [6]:

```
dataset.dtypes
```

Out[6]:

```
Page total likes
int64
Type
object
Category
int64
Post Month
int64
Post Weekday
int64
Post Hour
int64
Paid
float64
Lifetime Post Total Reach
int64
Lifetime Post Total Impressions
int64
Lifetime Engaged Users
int64
Lifetime Post Consumers
int64
Lifetime Post Consumptions
int64
Lifetime Post Impressions by people who have liked your Page
int64
Lifetime Post reach by people who like your Page
int64
Lifetime People who have liked your Page and engaged with your post
int64
comment
int64
like
float64
share
float64
Total Interactions
int64
dtype: object
```

Note :

1. There are 500 data points with 19 features.

Preprocessing the data

1. Dropping null values

In [7]:

```
dataset.isnull().sum()
```

Out[7]:

```
Page total likes
0
Type
0
Category
0
Post Month
0
Post Weekday
0
Post Hour
0
Paid
1
Lifetime Post Total Reach
0
Lifetime Post Total Impressions
0
Lifetime Engaged Users
0
Lifetime Post Consumers
0
Lifetime Post Consumptions
0
Lifetime Post Impressions by people who have liked your Page
0
Lifetime Post reach by people who like your Page
0
Lifetime People who have liked your Page and engaged with your post
0
comment
0
like
1
share
4
Total Interactions
0
dtype: int64
```

Note :

1. As seen above, there are null values in the dataset which can be either dropped or replaced

In [8]:

```
# Dropping rows with null values
dataset = dataset.dropna()
dataset.shape
```

Out[8]:

```
(495, 19)
```

In [9]:

```
# Testing data for null values
dataset.isnull().sum()
```

Out[9]:

```
Page total likes
0
Type
0
Category
0
Post Month
0
Post Weekday
0
Post Hour
0
Paid
0
Lifetime Post Total Reach
0
Lifetime Post Total Impressions
0
Lifetime Engaged Users
0
Lifetime Post Consumers
0
Lifetime Post Consumptions
0
Lifetime Post Impressions by people who have liked your Page
0
Lifetime Post reach by people who like your Page
0
Lifetime People who have liked your Page and engaged with your post
0
comment
0
like
0
share
0
Total Interactions
0
dtype: int64
```

All null value data points dropped

2. Generating subsets on the basis of type

Identifying unique values in the "Type" column

In [10]:

```
unique_type_entries = dataset["Type"].unique()
```

In [11]:

```
unique_type_entries
```

Out[11]:

```
array(['Photo', 'Status', 'Link', 'Video'], dtype=object)
```

Generating subsets

In [12]:

```
photo_subset = dataset[dataset["Type"] == "Photo"]
status_subset = dataset[dataset["Type"] == "Status"]
link_subset = dataset[dataset["Type"] == "Link"]
video_subset = dataset[dataset["Type"] == "Video"]
```

Shape of subsets

In [13]:

```
print("Photo Subset shape  : ", photo_subset.shape)
print("Status Subset shape : ", status_subset.shape)
print("Link Subset shape   : ", link_subset.shape)
print("Video Subset shape  : ", video_subset.shape)
```

```
Photo Subset shape  :  (421, 19)
Status Subset shape :  (45, 19)
Link Subset shape   :  (22, 19)
Video Subset shape  :  (7, 19)
```

Graphical representation of distribution of each subset

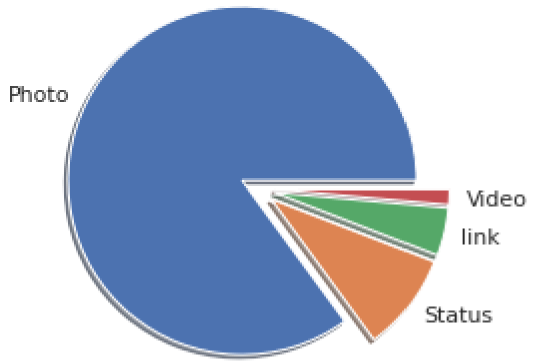
In [14]:

```
# Gathering distribution data
distribution_frequencies = [
    photo_subset.shape[0],
    status_subset.shape[0],
    link_subset.shape[0],
    video_subset.shape[0],
]

# Generating legend for pie chart
legend = [
    "Photo",
    "Status",
    "link",
    "Video"
]

# Defining explode values
explode = [0.1, 0.1, 0.1, 0.1]

# Generating and displaying piechart
plt.pie(
    x=distribution_frequencies,
    labels=legend,
    shadow=True,
    explode=explode
)
plt.show()
```



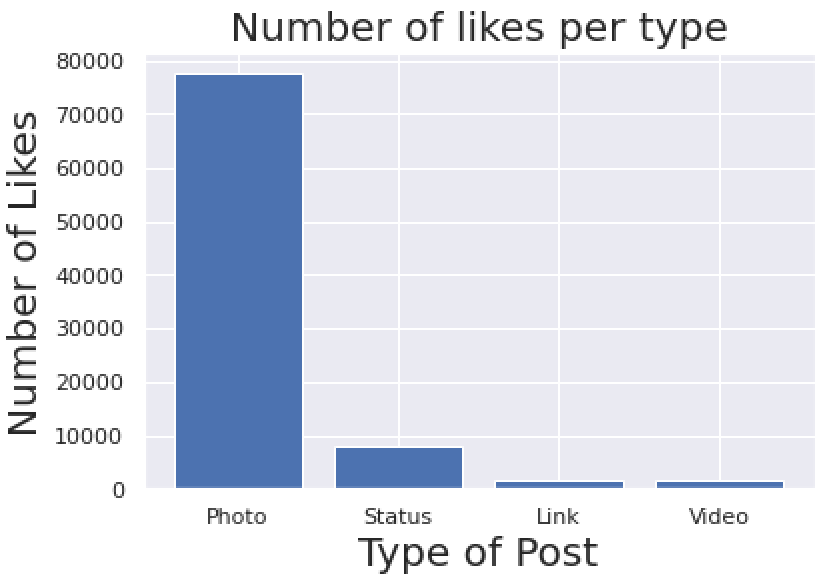
Comparing subsets

a) Likes per subset

In [15]:

```
# Calculating Likes per subset
likes_data = [
    int(photo_subset["like"].sum()),
    int(status_subset["like"].sum()),
    int(link_subset["like"].sum()),
    int(video_subset["like"].sum()),
]

# Generating and displaying bar chart
plt.bar(
    x=["Photo", "Status", "Link", "Video"],
    height=likes_data
)
plt.xlabel("Type of Post", fontsize=20)
plt.ylabel("Number of Likes", fontsize=20)
plt.title("Number of likes per type", fontsize=20)
plt.show()
```

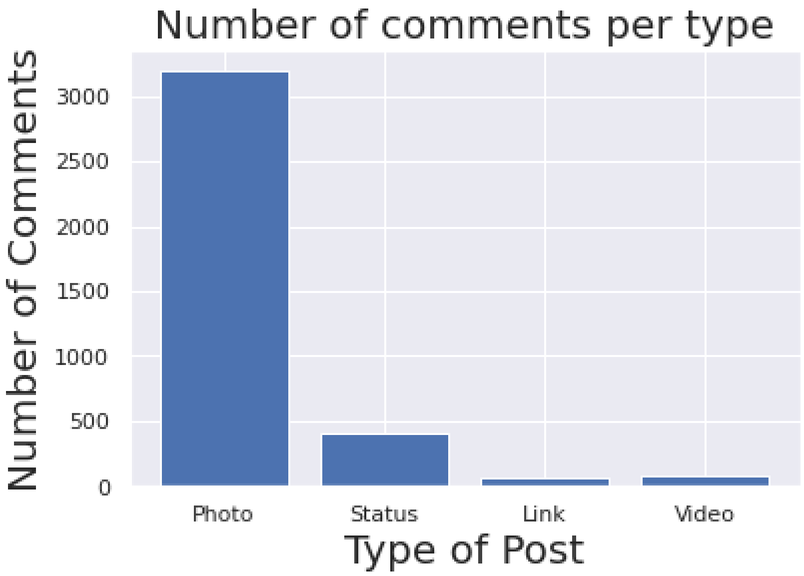


b) Comments per subset

In [16]:

```
# Calculating Likes per subset
commments_data = [
    int(photo_subset["comment"].sum()),
    int(status_subset["comment"].sum()),
    int(link_subset["comment"].sum()),
    int(video_subset["comment"].sum()),
]

# Generating and displaying bar chart
plt.bar(
    x=["Photo", "Status", "Link", "Video"],
    height=commments_data
)
plt.xlabel("Type of Post", fontsize=20)
plt.ylabel("Number of Comments", fontsize=20)
plt.title("Number of comments per type", fontsize=20)
plt.show()
```

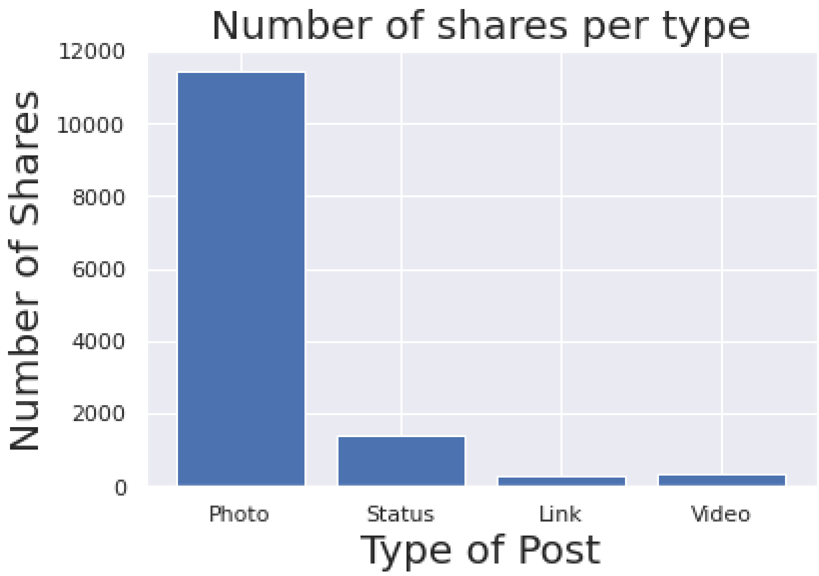


c) Shares per subset

In [17]:

```
# Calculating Likes per subset
shares_data = [
    int(photo_subset["share"].sum()),
    int(status_subset["share"].sum()),
    int(link_subset["share"].sum()),
    int(video_subset["share"].sum()),
]

# Generating and displaying bar chart
plt.bar(
    x=["Photo", "Status", "Link", "Video"],
    height=shares_data
)
plt.xlabel("Type of Post", fontsize=20)
plt.ylabel("Number of Shares", fontsize=20)
plt.title("Number of shares per type", fontsize=20)
plt.show()
```



Exploratory analysis for Photos subset

In [18]:

```
# Statistical description of numerical subset
photo_subset.describe(include="all")
```

Out[18]:

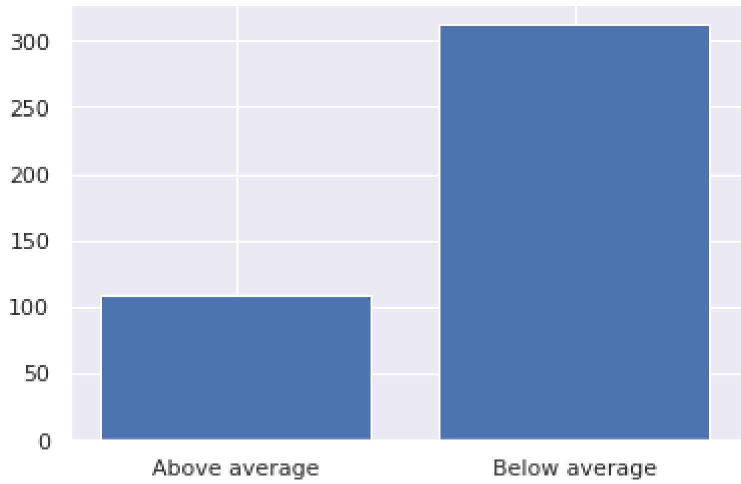
	Page total likes	Type	Category	Post Month	Post Weekday	Post Hour	Paid	Li
count	421.000000	421	421.000000	421.000000	421.000000	421.000000	421.000000	
unique	NaN	1	NaN	NaN	NaN	NaN	NaN	
top	NaN	Photo	NaN	NaN	NaN	NaN	NaN	
freq	NaN	421	NaN	NaN	NaN	NaN	NaN	
mean	122319.612827	NaN	1.926366	6.790974	4.087886	8.004751	0.282660	13
std	16242.669134	NaN	0.884681	3.228447	2.056203	4.432561	0.450828	22
min	81370.000000	NaN	1.000000	1.000000	1.000000	1.000000	0.000000	
25%	109670.000000	NaN	1.000000	4.000000	2.000000	3.000000	0.000000	3
50%	128032.000000	NaN	2.000000	7.000000	4.000000	9.000000	0.000000	4
75%	136013.000000	NaN	3.000000	10.000000	6.000000	11.000000	1.000000	10
max	139441.000000	NaN	3.000000	12.000000	7.000000	23.000000	1.000000	18

In [19]:

```
# Number of posts with more than and less than average likes
mean_photo_likes = photo_subset["like"].mean()
above_average_photo_likes = photo_subset[photo_subset["like"] >= mean_photo_likes]
below_average_photo_likes = photo_subset[photo_subset["like"] < mean_photo_likes]
print("Average likes             : ", mean_photo_likes)
print("Above average photo likes : ", above_average_photo_likes.shape[0])
print("Below average photo likes : ", below_average_photo_likes.shape[0])

# Graphical representation
plt.bar(
    x=["Above average", "Below average"],
    height=[
        above_average_photo_likes.shape[0],
        below_average_photo_likes.shape[0]
    ]
)
plt.show()
```

```
Average likes             :  184.0665083135392
Above average photo likes :  109
Below average photo likes :  312
```



In [20]:

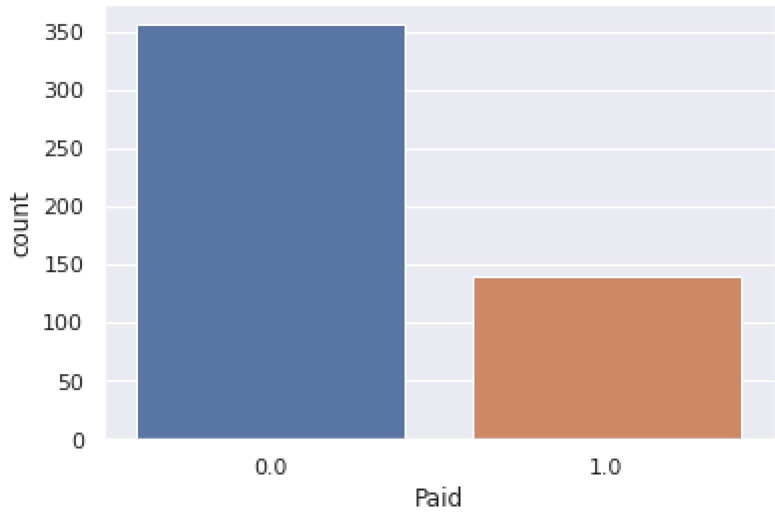
```
photo_subset["Paid"].unique()
```

Out[20]:

```
array([0., 1.])
```

In [22]:

```
# Counting number of paid and unpaid posts
sns.countplot(x=dataset["Paid"])
plt.show()
```



3. Transpose of data

Note :

1. The smallest subset is considered for transposing

In [23]:

```
# Shape of data before transposing
print("Shape of Video subset : ", video_subset.shape)
```

```
Shape of Video subset :  (7, 19)
```

In [24]:

```
# Transposing data
video_subset_transpose = video_subset.transpose()
```

In [25]:

```
# Shape of data after transposing
print("Shape of Video subset transpose: ", video_subset_transpose.shape)
```

```
Shape of Video subset transpose:  (19, 7)
```

In [26]:

```
video_subset_transpose
```

Out[26]:

	29	55	71	74	183	243	277
Page total likes	138895	138329	137893	137893	134879	130791	126424
Type	Video	Video	Video	Video	Video	Video	Video
Category	1	1	1	1	1	1	1
Post Month	12	11	11	11	9	7	6
Post Weekday	4	6	5	3	2	3	2
Post Hour	11	2	3	11	10	11	13
Paid	1.0	1.0	1.0	0.0	0.0	1.0	0.0
Lifetime Post Total Reach	36208	16416	100768	13544	30624	21872	139008
Lifetime Post Total Impressions	61262	31950	220447	30235	56950	40413	277100
Lifetime Engaged Users	1141	459	2101	517	2080	3872	1779
Lifetime Post Consumers	1068	411	1735	458	1956	3822	1643
Lifetime Post Consumptions	1728	539	2331	667	3253	7327	2356
Lifetime Post Impressions by people who have liked your Page	30131	21436	59658	26622	32033	24667	107502
Lifetime Post reach by people who like your Page	14112	9568	18880	11760	15744	12920	38720
Lifetime People who have liked your Page and engaged with your post	559	363	885	447	1376	2218	1008
comment	18	2	17	2	6	18	23
like	143.0	65.0	449.0	99.0	345.0	315.0	204.0
share	13.0	14.0	84.0	13.0	121.0	76.0	44.0
Total Interactions	174	81	550	114	472	409	271

4. Merging data

Note :

1. For performing merging operation, 2 subsets of the given dataset are considered (Photo and video subset)

In [27]:

```
print("Shape of photo subset : ", photo_subset.shape)
print("Shape of video subset : ", video_subset.shape)
```

```
Shape of photo subset :  (421, 19)
Shape of video subset :  (7, 19)
```

In [29]:

```
# Checking columns of both data subsets
print("Columns of photo subset : ", photo_subset.columns)
print("Columns of video subset : ", video_subset.columns)
```

```
Columns of photo subset :  Index(['Page total likes', 'Type', 'Categor
y', 'Post Month', 'Post Weekday',
       'Post Hour', 'Paid', 'Lifetime Post Total Reach',
       'Lifetime Post Total Impressions', 'Lifetime Engaged Users',
       'Lifetime Post Consumers', 'Lifetime Post Consumptions',
       'Lifetime Post Impressions by people who have liked your Page',
       'Lifetime Post reach by people who like your Page',
       'Lifetime People who have liked your Page and engaged with your
post',
       'comment', 'like', 'share', 'Total Interactions'],
      dtype='object')
Columns of video subset :  Index(['Page total likes', 'Type', 'Categor
y', 'Post Month', 'Post Weekday',
       'Post Hour', 'Paid', 'Lifetime Post Total Reach',
       'Lifetime Post Total Impressions', 'Lifetime Engaged Users',
       'Lifetime Post Consumers', 'Lifetime Post Consumptions',
       'Lifetime Post Impressions by people who have liked your Page',
       'Lifetime Post reach by people who like your Page',
       'Lifetime People who have liked your Page and engaged with your
post',
       'comment', 'like', 'share', 'Total Interactions'],
      dtype='object')
```

In [32]:

```
# Merging the 2 subsets (DataFrames)
photo_video_merged = pd.merge(
    left=photo_subset,
    right=video_subset,
    on="Paid"
)
```

In [33]:

```
photo_video_merged.head()
```

Out[33]:

	Page total likes_x	Type_x	Category_x	Post Month_x	Post Weekday_x	Post Hour_x	Paid	Lifetime Post Total Reach_x	Lifetime Post Total Impressions_x
0	139441	Photo	2	12	4	3	0.0	2752	5091
1	139441	Photo	2	12	4	3	0.0	2752	5091
2	139441	Photo	2	12	4	3	0.0	2752	5091
3	139441	Photo	3	12	3	3	0.0	2413	4373
4	139441	Photo	3	12	3	3	0.0	2413	4373

5 rows × 37 columns

In [35]:

```
photo_video_merged.shape
```

Out[35]:

```
(1382, 37)
```

5. Sorting data

Sorting the data on the basis of the number of likes

In [36]:

```
# Sorting the data on the basis of number of likes
likes_sorted_data = dataset.sort_values(by="Page total likes")
```

In [37]:

```
# Displaying the top 5 liked records
likes_sorted_data.head()
```

Out[37]:

	Page total likes	Type	Category	Post Month	Post Weekday	Post Hour	Paid	Lifetime Post Total Reach	Lifetime Post Total Impressions	Lifetime Engaged Users	C
498	81370	Photo	3	1	4	11	0.0	4156	7564	626	
497	81370	Photo	1	1	5	2	0.0	3778	7216	625	
496	81370	Photo	2	1	5	8	0.0	3480	6229	537	
493	85093	Photo	3	1	1	2	0.0	8412	13960	1179	
495	85093	Photo	3	1	7	2	0.0	4684	7536	733	

In [39]:

```
# Displaying the bottom 10 liked records
likes_sorted_data.tail(10)
```

Out[39]:

	Page total likes	Type	Category	Post Month	Post Weekday	Post Hour	Paid	Lifetime Post Total Reach	Lifetime Post Total Impressions	Lifetime Engaged Users
4	139441	Photo	2	12	2	3	0.0	7244	13594	671
6	139441	Photo	3	12	1	3	1.0	11692	19479	481
12	139441	Photo	2	12	5	10	0.0	2847	5133	193
8	139441	Status	2	12	7	3	0.0	11844	22538	1530
9	139441	Photo	3	12	6	10	0.0	4694	8668	280
10	139441	Status	2	12	5	10	0.0	21744	42334	4258
11	139441	Photo	2	12	5	10	0.0	3112	5590	208
13	139441	Photo	2	12	5	3	0.0	2549	4896	249
7	139441	Photo	3	12	7	9	1.0	13720	24137	537
0	139441	Photo	2	12	4	3	0.0	2752	5091	178

6. Reshaping the data

Note :

Here, the operations of melt and pivot are used to reshape the data in computer readable format

Melting

In [40]:

```
# Melting the data on the value variables as type and category
melting_result = pd.melt(
    frame=dataset,
    id_vars="Page total likes",
    value_vars=["Type", "Category"]
)
```

In [41]:

```
melting_result.head()
```

Out[41]:

	Page total likes	variable	value
0	139441	Type	Photo
1	139441	Type	Status
2	139441	Type	Photo
3	139441	Type	Photo
4	139441	Type	Photo

In [42]:

```
melting_result.tail()
```

Out[42]:

	Page total likes	variable	value
985	85093	Category	3
986	85093	Category	3
987	81370	Category	2
988	81370	Category	1
989	81370	Category	3

In [43]:

```
# Checking shape of melted data
melting_result.shape
```

Out[43]:

```
(990, 3)
```

End of Notebook