

EXPERIMENT NO 1

A.TASK ON SALARY DATA

1.FINDING DEPARTMENT WISE AVERAGE SALARY.

In [1]:

```
# importing pandas library
import pandas as pd

# importing numpy library
import numpy as np

# from matplotlib module importing pyplot
import matplotlib.pyplot as plt
```

In [2]:

```
# reading the dataset
df = pd.read_csv(
    'DOCUMENTS\\COLLEGE\\CLASSES\\EXPERIMENT_NO_1\\Salaries.csv\\Salaries.csv',
    low_memory=False)

# first few elements
df.head()
```

Out[2]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOUTAN TRANSPORT AUTHORITY	167411.18	0.0	400184.25	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011	NaN	San Francisco	NaN
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.6	NaN	335279.91	335279.91	2011	NaN	San Francisco	NaN
3	4	CHRISTOPHER CHONG	WIRED ROPE CABLE MAINTENANCE MECHANIC	77916.0	56120.71	198306.9	NaN	332343.61	332343.61	2011	NaN	San Francisco	NaN
4	5	PATRICK GARDNER	DEPUTY CHIEF OF FIRE DEPARTMENT)	134401.6	9737.0	182234.59	NaN	326373.19	326373.19	2011	NaN	San Francisco	NaN

In [3]:

```
# showing the data information
df.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
Column Non-Null Count Dtype

0 Id 148654 non-null int64
1 EmployeeName 148654 non-null object
2 JobTitle 148654 non-null object
3 BasePay 148043 non-null object
4 OvertimePay 148654 non-null object
5 OtherPay 148654 non-null object
6 Benefits 112495 non-null object
7 TotalPay 148654 non-null float64
8 TotalPayBenefits 148654 non-null float64
9 Year 148654 non-null int64
10 Notes 0 non-null float64
11 Agency 148654 non-null object
12 Status 38119 non-null object
dtypes: float64(3), int64(2), object(8)
memory usage: 14.7+ MB

In [4]:

```
# showing the data statistics
df.describe(include="all")
```

Out[4]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
count	148654.000000	148654	148654	148049	148654	148654	112495	148654.000000	148654.000000	148654.000000
unique	NaN	NaN	110811	2159	109724	66162	84243	98647	NaN	NaN
top	NaN	Kevin Lee	Transit Operator	0.0	0.0	0.0	0.0	NaN	NaN	NaN
freq	NaN	13	7036	872	57868	30068	1024	NaN	NaN	NaN
mean	74327.500000	NaN	NaN	NaN	NaN	NaN	NaN	74768.321972	93692.554811	2012.522643
std	42912.857795	NaN	NaN	NaN	NaN	NaN	NaN	50517.005274	62793.533483	1.117538
min	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	-618.130000	-618.130000	2011.000000
25%	37164.250000	NaN	NaN	NaN	NaN	NaN	NaN	36168.995000	44065.655000	2012.000000
50%	74327.500000	NaN	NaN	NaN	NaN	NaN	NaN	71426.610000	92404.090000	2013.000000
75%	111490.750000	NaN	NaN	NaN	NaN	NaN	NaN	105839.135000	132876.450000	2014.000000
max	148654.000000	NaN	NaN	NaN	NaN	NaN	NaN	567595.430000	567595.430000	2014.000000

In [5]:

```
# showing salary department wise mean
dept_wise_mean = df.groupby('JobTitle')['TotalPay'].mean()

dept_wise_mean
```

dept_wise_mean_df = pd.DataFrame(dept_wise_mean).reset_index()

dept_wise_mean_df.head()

Out[5]:

	JobTitle	TotalPay
0	ACCOUNT CLERK	44035.664337
1	ACCOUNTANT	47429.268000
2	ACCOUNTANT INTERN	29031.742917
3	ACPO/JuvP, Juv Prob (SFRS)	62290.780000
4	ACUPUNCTURIST	67594.400000

2.GRAPH OF DEPARTMENT WISE AVERAGE SALARY

In [6]:

```
# sort the obtained values
dept_wise_mean_df.sort_values('TotalPay', ascending=False, inplace=True)

dept_wise_mean_df['JobTitle'] = dept_wise_mean_df.index

dept_wise_mean_df.head()
```

Out[6]:

	JobTitle	TotalPay
831	831	399211.275000
466	466	339653.700000
474	474	329183.646667
475	475	325971.683333
586	586	307899.460000

In [7]:

```
# plot of top 15 department with their corresponding average salary
dept_wise_mean_df[['JobTitle', 'TotalPay']][1:16].plot(
    x='JobTitle',
    kind='bar',
    stacked=False,
    title='Plot Between Department & Their Corresponding Average Salary',
    color='orange')
```

Out[7]:

<AxesSubplot:title='center':'Plot Between Department & Their Corresponding Average Salary', xlabel='JobTitle'>

B.TWITTER DATA COLLECTION

In [8]:

```
# installing the tweepy

!pip install tweepy
```

Requirement already satisfied: tweepy in c:\users\hp\anaconda3\lib\site-packages (4.4.0)
Requirement already satisfied: requests<3,>=2.11.1 in c:\users\hp\anaconda3\lib\site-packages (from tweepy) (2.26.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\anaconda3\lib\site-packages (from requests<3,>=2.11.1->tweepy) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\hp\anaconda3\lib\site-packages (from requests<3,>=2.11.1->tweepy) (1.26.7)
Requirement already satisfied: charset-normalizer<=2.0.0 in c:\users\hp\anaconda3\lib\site-packages (from requests<3,>=2.11.1->tweepy) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\anaconda3\lib\site-packages (from requests<3,>=2.11.1->tweepy) (3.2)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\hp\anaconda3\lib\site-packages (from requests<3,>=2.11.1->tweepy) (3.1.1)

In [9]:

```
# Importing Libraries

import tweepy as tw
import pandas as pd
import numpy as np
```

In [10]:

```
# gathering the consumer key and secret

api_key = '188R8976d8B5C515R1krFpcPYC'
api_key_secret = '3VnG6gR8Rp4P4FF2B5ZaEUXFbL1VX2W21J4HYb4JwMAAKS1AMNlt'
access_token = '79789482566369280~B8R8BmQuU228B2gTAFUQc0J3wec0U3Kp'
access_token_secret = '1a3x6W65G7mUdW6W8G7bUa1JF5c701P6b0UydyzPw1B'
```

In [11]:

```
# searching keyword name

search_words = "#Election2022"

# searching number of tweets

number_posts = 2000
```

In [12]:

```
# using the OAuth for accessing tweets

auth = tw.OAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)
api = tw.API(auth, wait_on_rate_limit=True)
```

In [13]:

```
# downloading the tweets

tweets = tw.Cursor(api.search_tweets, q=search_words,
                    lang='en').items(number_posts)

# list to store downloaded data

tweets_text = []

# downloading different attributes

for tweet in tweets:
    tweets_text.append(
        [tweet.user.screen_name, tweet.user.location,
        tweet.user.followers_count, tweet.user.description, tweet.text
        ])
```

In [14]:

```
# storing tweet data

tweet_data = pd.DataFrame(tweets_text,
                           columns=[
                               "User_Name", "Location", "No of Followers",
                               "Description (Status)", "Tweets Posted"
                           ])

# first few tweet data

tweet_data.head()
```

Out[14]:

	User_Name	Location	No of Followers	Description (Status)	Tweets Posted
0	UKSINGH9	India	77		RT @Dalt_Swag: Elections in Uttar Pradesh are...
1	vikrantdesai	California, USA	101	विक्रान्त vs us "The important thing is to never...	RT @shweta_kamat: 26% of the candidates contes...
2	VickiMo8056436		1039	A daughter. A sister. A mother. A granny. A gr...	RT @StMarysOC: Please remember that MARYLAND ...
3	magipoe01	Norwood South Oz	595	Poet writer editor - atheist, pooodle tragic e...	RT @ JamesMeldrum: The words of opposition to ...
4	Bmzre_Elections	Baltimore City, MD	1779	The official feed of the #BaltimoreCity Board...	Election judges are needed in Baltimore City!...

In [15]:

```
# collected data information

tweet_data.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2005 entries, 0 to 1004
Data columns (total 5 columns):
Column Non-Null Count Dtype

0 User_Name 1005 non-null object
1 Location 1005 non-null object
2 No of Followers 1005 non-null int64
3 Description (Status) 1005 non-null object
4 Tweets Posted 1005 non-null object
dtypes: int64(1), object(4)
memory usage: 39.4+ KB

In [16]:

```
# saving the data as csv file

tweet_data.to_csv('DOCUMENTS\\COLLEGE\\CLASSES\\EXPERIMENT_NO_1\\tweet_data.csv')
```

C.STOCK MARKET DATA COLLECTION

In [17]:

```
# importing the libraries

import pandas as pd
import requests

from bs4 import BeautifulSoup
```

FETCHING THE WEB DATA

In [18]:

```
# creating an object of BeautifulSoup to get HTML code of the requested url

url = 'https://finance.yahoo.com/quote/MSFT/'

# sending url request to get unparsed html data

res = requests.get(url)

# using the python html.parser to parse the collected html code in formatted manner

soup = BeautifulSoup(res.text, "html.parser")

print(soup)
```


[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

<script defer="" src="https://s.yimg.com/aq/yw/r?url=rapid-16.0.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=bind-1.3.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=fetch-118.0.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=form-130.5.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=image-13.0.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=menu-1.5.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=tab-12.0.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=toggle-13.3.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=tooltip-1.3.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=beacon-13.3.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=casa-1.4.29.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=darla-1.2.0.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=loader-1.8.45.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=sticky-1.0.30.js"></script>
<script defer="" src="https://s.yimg.com/aq/yw/r?url=template-1.4.2.js"></script>
<script src="https://s.yimg.com/aq/yw/viewer/desktop.110.132.js"></script>
<script src="https://s.yimg.com/aq/yw/membership/omp-static/omp-widget/v2.0.0/switch-widget-prod.js"></script>
<script defer="" src="https://s.yimg.com/uc/finance/dn-wid/widget/main.208b59c55967d5e3a67f.min.js"></script>
<script async="" src="https://s.yimg.com/aq/yw/perf/vitals.2.1.1.js"></script>
</script>

(function () {
    var w = window.wafer || {}
    typeof w.ready == "function" && w.ready(function () {
        typeof w.on == "function" && w.on("tabselected", function (el) {
            try {
                if (o && e.meta && e.meta.targetElem.id === "header-notification-menu") {
                    window.setTimeout(function hidePage() {
                        w.base.state = {financeNotification:hideba

```

Day's Range	299.95 - 304.82
52 Week Range	224.26 - 349.67
Volume	9,269,800
Avg. Volume	34,279,433
Market Cap	2.278T
Beta (5Y Monthly)	0.89
PE Ratio (TTM)	32.37
EPS (TTM)	9.39
Earnings Date	Apr 25, 2022 - Apr 29, 2022
Forward Dividend & Yield	2.48 (0.82%)
Ex-Dividend Date	Feb 16, 2022
1y Target Est	370.97

X

X