DEDP PRESENTATION

A

PROJECT ON

Top_200_Comman_password _by_Different_Country_2022

BY
AKSHATA SHIRWALE
&
PAURAVI KHATPE

GUIDED BY:-
PROF. SUSHMITA BHAMBHURE

Shikshana Prasaraka Mandali's

Accredited with A+ Grade by NAAC

SIR PARASHURAMBHAU COLLEGE (AUTONOMOUS)

TILAK ROAD, PUNE – 411 030.

Department of Computer Science

Certificate

This is to certify that Akshata Dattatray Shirwale has presented a mini project titled Top 200 common passwords by different country in partial fulfillment of the requirements of M.Sc.

(Computer Science) Part I (semester I) course.

Date: 14 /01/2023

Teacher In-Charge

Head,

Department of Computer Science

ACKNOWLEDGEMENT

We would like to express our profound thanks to all those who helped in making this project a reality. Much needed moral support and encouragement was provided on numerous occasions by all teachers and fellow students.

We are thankful to our Head of the Department. We are also especially grateful to our teacher and project guide *Prof. Sushmita Bhambhure* for his time to time, much needed, valuable guidance. she has been available to help us whenever required. she has been our guiding star, guiding us through the right direction to make this project a success. Without his full support and cheerful encouragement we would have not been able to complete the project in time. We are sincerely thankful to him.

Last but not the least we would like to thank our fellow student for providing their support to us in different ways.

INDEX

1) Description of the data

- We take data from kagal.com website (Top 200 common passwords by country 2022).
- We have all country data from A-Z.
- In data we have column name such as- country_code, country/territory name, rank, password, user count, time to crack password, global rank, time to crack password (in seconds).

2) Data observation techniques

- Bar graph
- Histogram
- Heat map
- Word cloud
- Box plot
- Scatter plot
- Waterfall chart
- Pictogram chart

3) Techniques used for Preprocessing Data

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for processing. Data preprocessing is a required task for cleaning the data and making it suitable for processing which also increases the accuracy and efficiency of the model.

1) Importing Libraries :-

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

- **Numpy:-**

Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports adding large, multidimensional arrays and matrices.

- **Matplotlib:-**

The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code.

- **Pandas:-**

The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library.

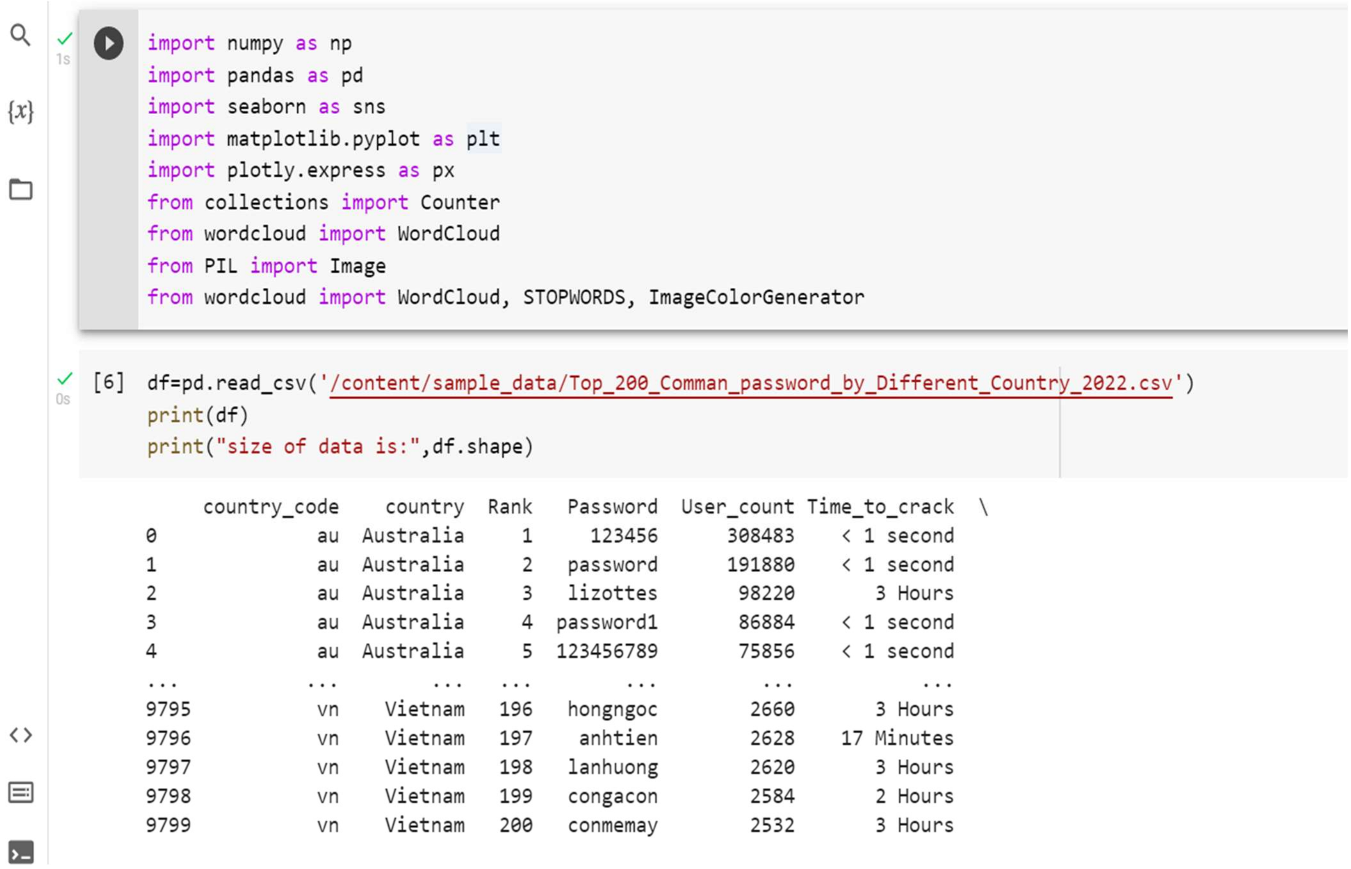
2) Handling Missing data :-

The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

Calculating the mean :-

In this way, we will calculate the mean of that column or row which contains any missing value and will put it in the place of missing value. This strategy is useful for the features which have numeric data.

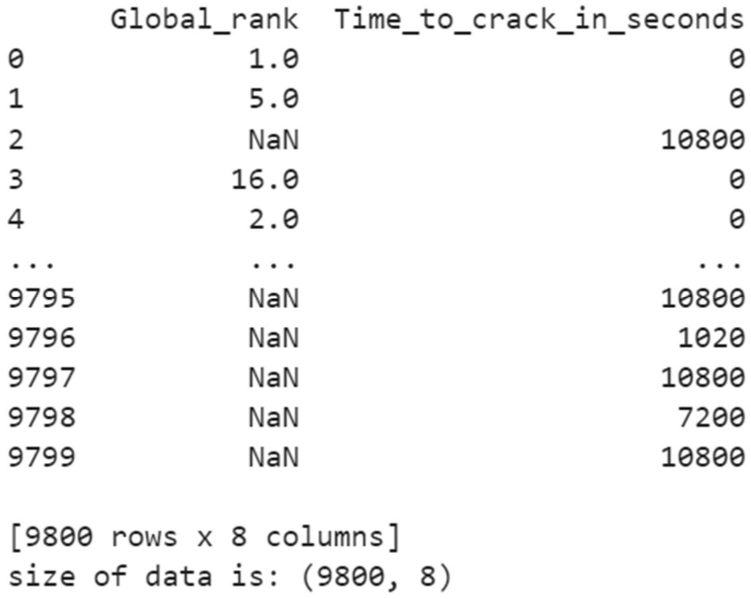
4) Python Code



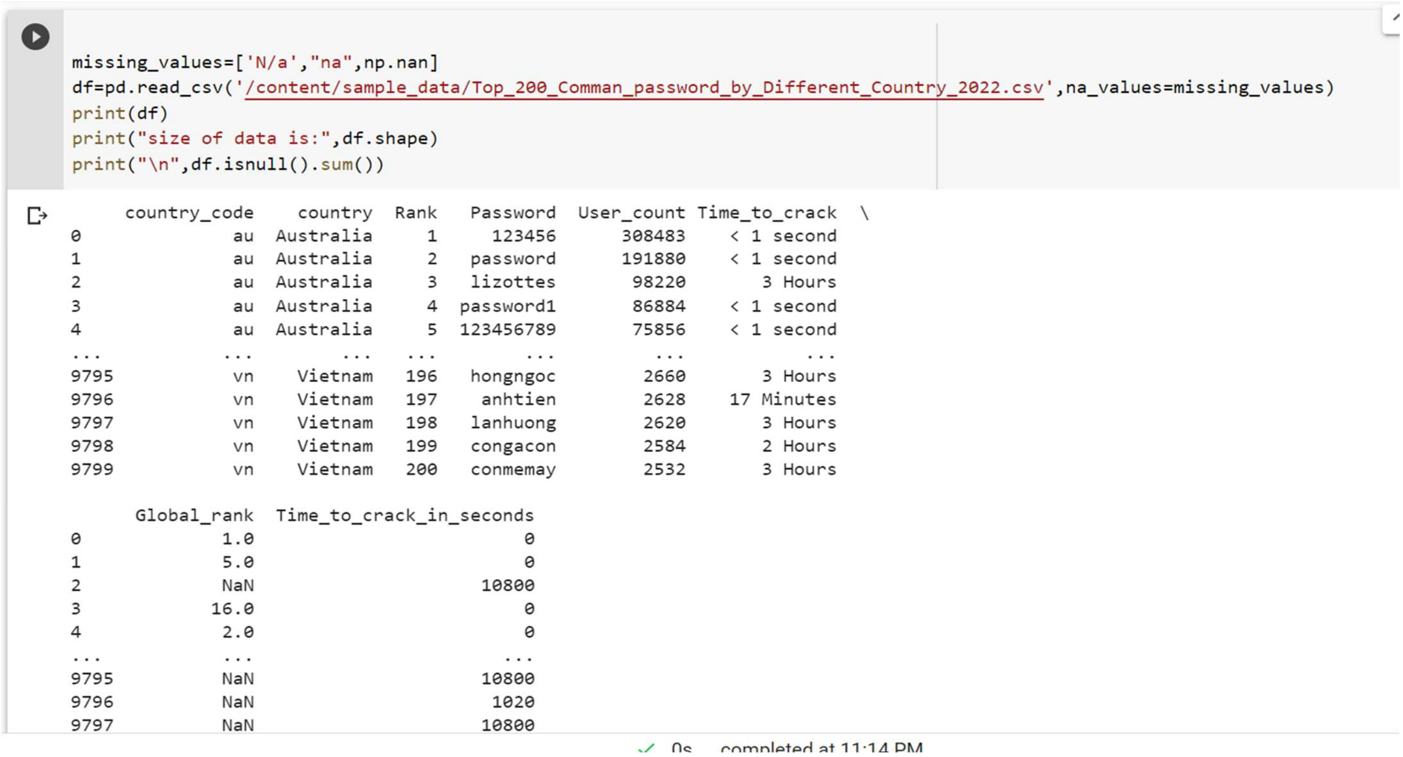
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from collections import Counter
from wordcloud import WordCloud
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

```
[6] df=pd.read_csv('/content/sample_data/Top_200_Comman_password_by_Different_Country_2022.csv')
    print(df)
    print("size of data is:",df.shape)
```

```
     country_code    country  Rank   Password  User_count Time_to_crack  \
0              au  Australia     1     123456      308483    < 1 second
1              au  Australia     2   password      191880    < 1 second
2              au  Australia     3   lizottes       98220       3 Hours
3              au  Australia     4  password1       86884    < 1 second
4              au  Australia     5  123456789       75856    < 1 second
...           ...        ...   ...        ...         ...           ...
9795           vn    Vietnam   196   hongngoc        2660       3 Hours
9796           vn    Vietnam   197    anhtien        2628    17 Minutes
9797           vn    Vietnam   198   lanhuong        2620       3 Hours
9798           vn    Vietnam   199   congacon        2584       2 Hours
9799           vn    Vietnam   200   conmemay        2532       3 Hours
```



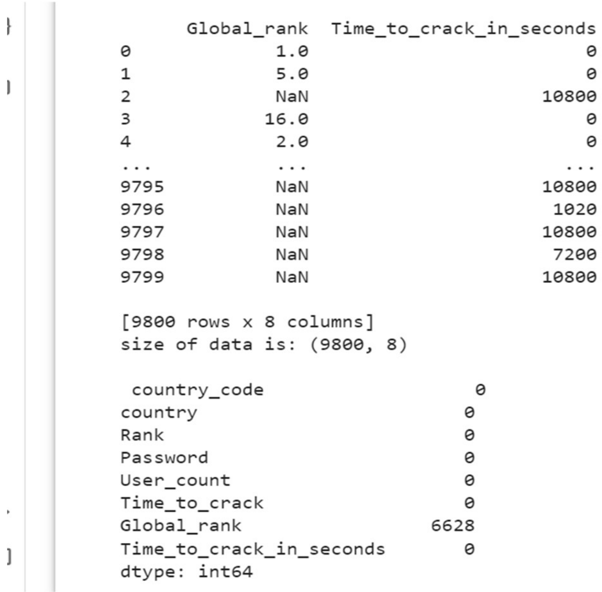
```
      Global_rank  Time_to_crack_in_seconds
0             1.0                         0
1             5.0                         0
2             NaN                     10800
3            16.0                         0
4             2.0                         0
...           ...                       ...
9795          NaN                     10800
9796          NaN                      1020
9797          NaN                     10800
9798          NaN                      7200
9799          NaN                     10800

[9800 rows x 8 columns]
size of data is: (9800, 8)
```



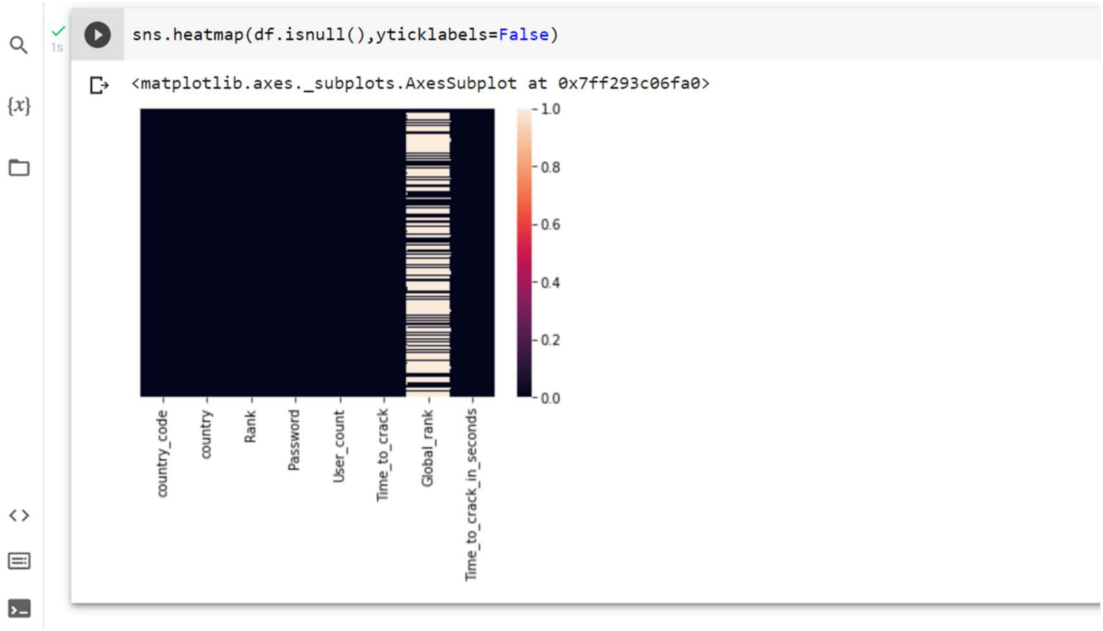
```
missing_values=['N/a',"na",np.nan]
df=pd.read_csv('/content/sample_data/Top_200_Comman_password_by_Different_Country_2022.csv',na_values=missing_values)
print(df)
print("size of data is:",df.shape)
print("\n",df.isnull().sum())
```

```
     country_code    country  Rank   Password  User_count Time_to_crack  \
0              au  Australia     1     123456      308483      < 1 second
1              au  Australia     2   password      191880      < 1 second
2              au  Australia     3   lizottes       98220        3 Hours
3              au  Australia     4  password1       86884      < 1 second
4              au  Australia     5  123456789       75856      < 1 second
...           ...        ...   ...        ...         ...           ...
9795           vn    Vietnam   196   hongngoc        2660        3 Hours
9796           vn    Vietnam   197    anhtien        2628     17 Minutes
9797           vn    Vietnam   198  lanhuong        2620        3 Hours
9798           vn    Vietnam   199   congacon        2584        2 Hours
9799           vn    Vietnam   200   conmemay        2532        3 Hours

      Global_rank  Time_to_crack_in_seconds
0             1.0                         0
1             5.0                         0
2             NaN                     10800
3            16.0                         0
4             2.0                         0
...           ...                       ...
9795          NaN                     10800
9796          NaN                      1020
9797          NaN                     10800
```

✓ 0s completed at 11:14 PM

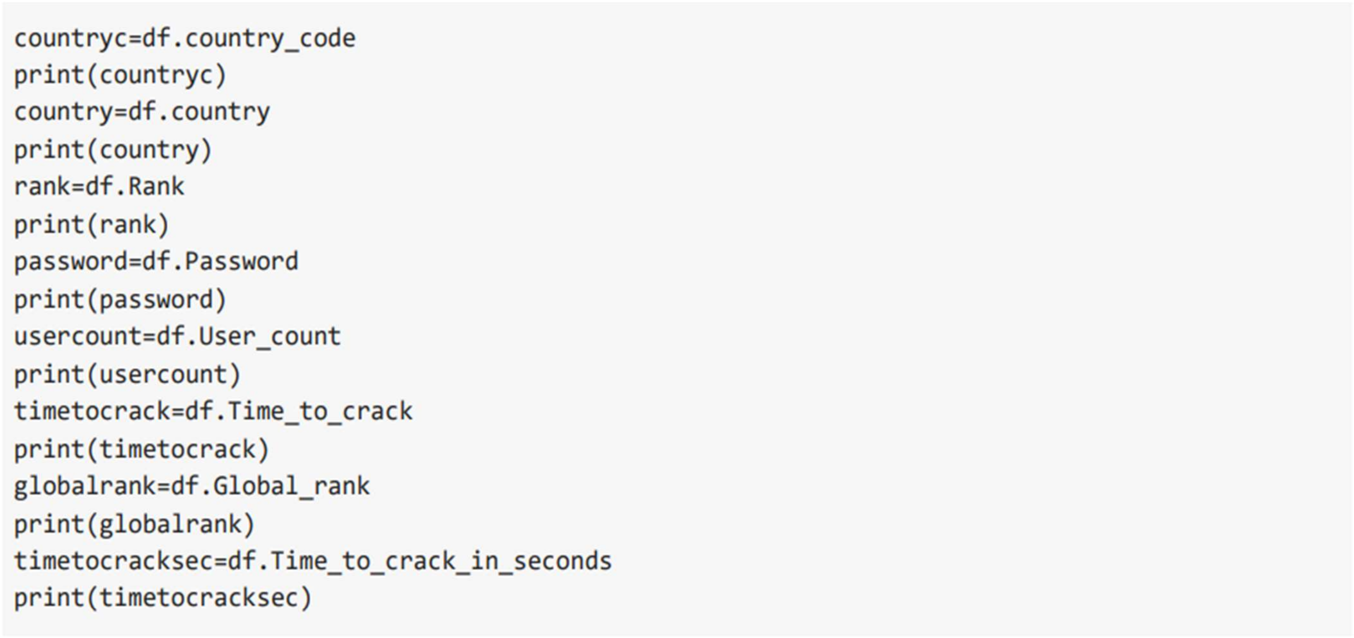


```
      Global_rank  Time_to_crack_in_seconds
0             1.0                         0
1             5.0                         0
2             NaN                     10800
3            16.0                         0
4             2.0                         0
...           ...                       ...
9795          NaN                     10800
9796          NaN                      1020
9797          NaN                     10800
9798          NaN                      7200
9799          NaN                     10800

[9800 rows x 8 columns]
size of data is: (9800, 8)

 country_code                   0
country                        0
Rank                           0
Password                       0
User_count                     0
Time_to_crack                  0
Global_rank                 6628
Time_to_crack_in_seconds       0
dtype: int64
```

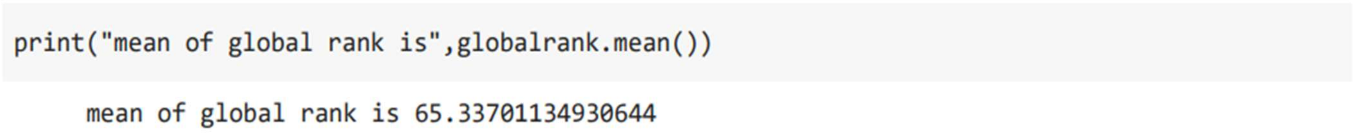


```
sns.heatmap(df.isnull(),yticklabels=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff293c06fa0>
```

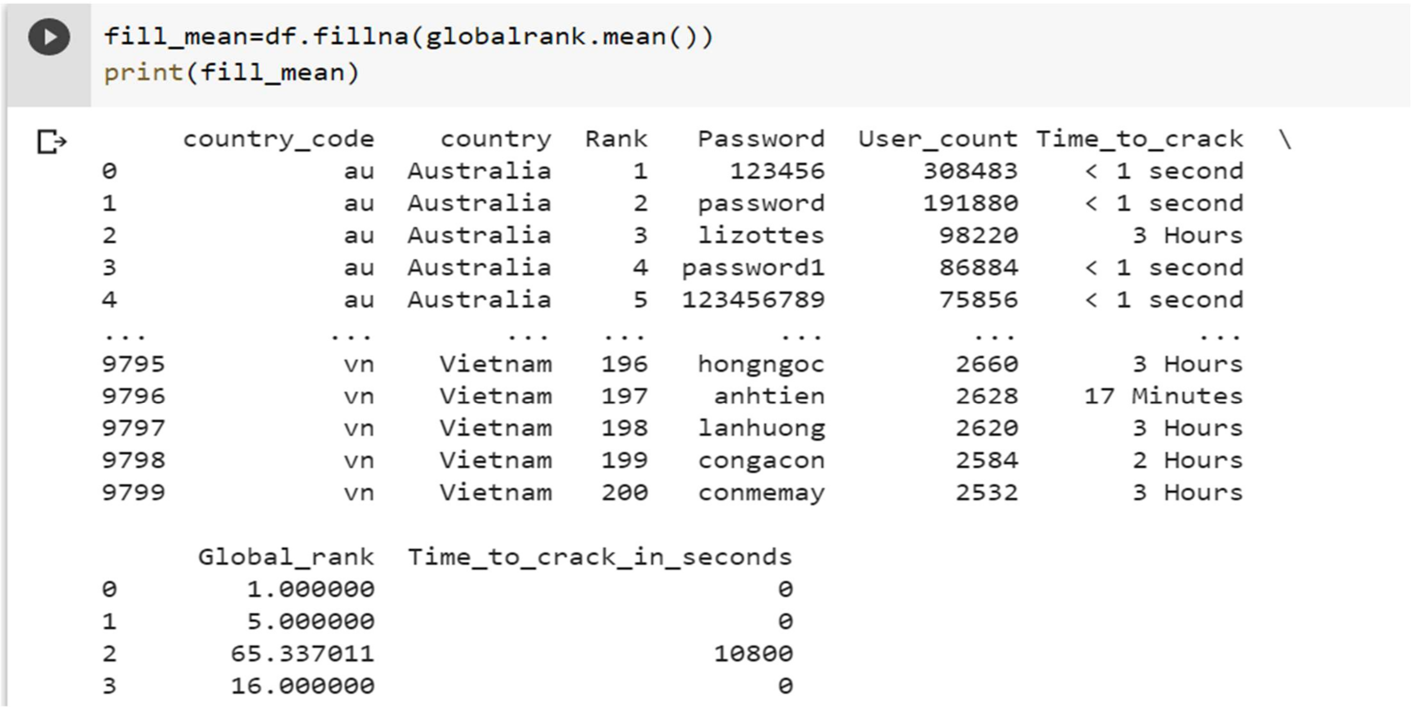


```
countryc=df.country_code
print(countryc)
country=df.country
print(country)
rank=df.Rank
print(rank)
password=df.Password
print(password)
usercount=df.User_count
print(usercount)
timetocrack=df.Time_to_crack
print(timetocrack)
globalrank=df.Global_rank
print(globalrank)
timetocracksec=df.Time_to_crack_in_seconds
print(timetocracksec)
```



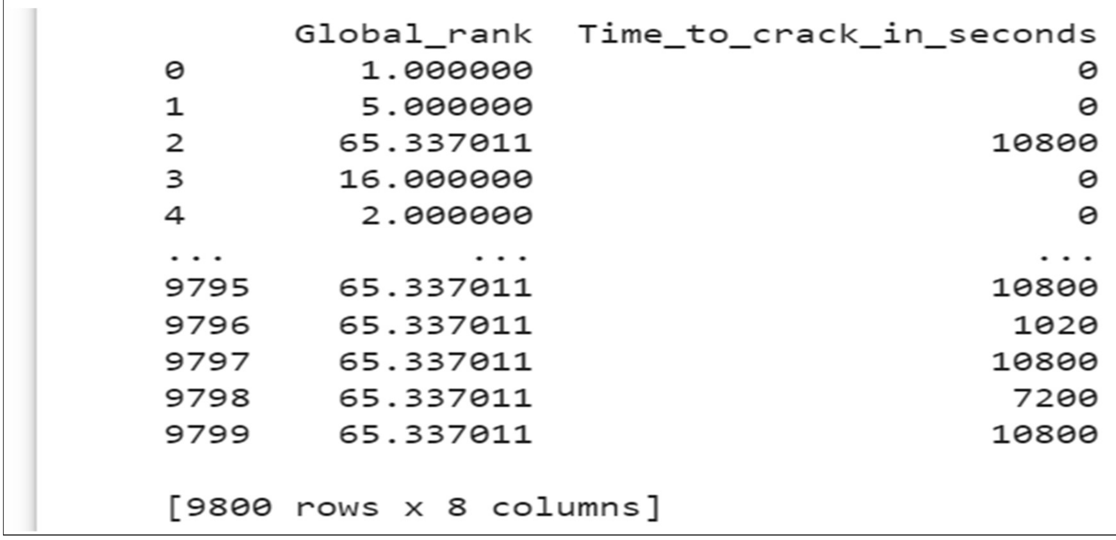
```
print("mean of global rank is",globalrank.mean())
```

```
mean of global rank is 65.33701134930644
```

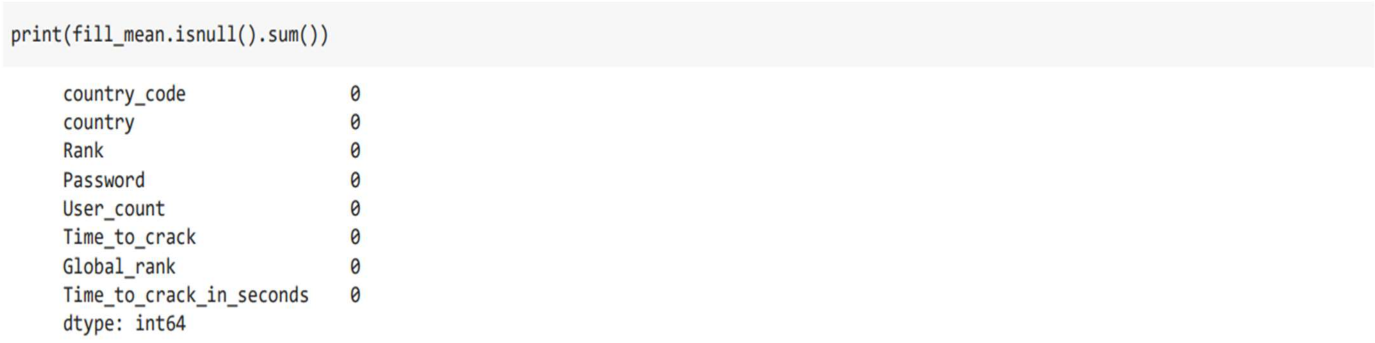


```
fill_mean=df.fillna(globalrank.mean())
print(fill_mean)
```

```
     country_code    country  Rank   Password  User_count Time_to_crack  \
0              au  Australia     1     123456      308483    < 1 second
1              au  Australia     2   password      191880    < 1 second
2              au  Australia     3   lizottes       98220       3 Hours
3              au  Australia     4  password1       86884    < 1 second
4              au  Australia     5  123456789       75856    < 1 second
...           ...        ...   ...        ...         ...           ...
9795           vn    Vietnam   196   hongngoc        2660       3 Hours
9796           vn    Vietnam   197    anhtien        2628    17 Minutes
9797           vn    Vietnam   198   lanhuong        2620       3 Hours
9798           vn    Vietnam   199    congacon        2584       2 Hours
9799           vn    Vietnam   200   conmemay        2532       3 Hours

      Global_rank  Time_to_crack_in_seconds
0        1.000000                         0
1        5.000000                         0
2       65.337011                     10800
3       16.000000                         0
```

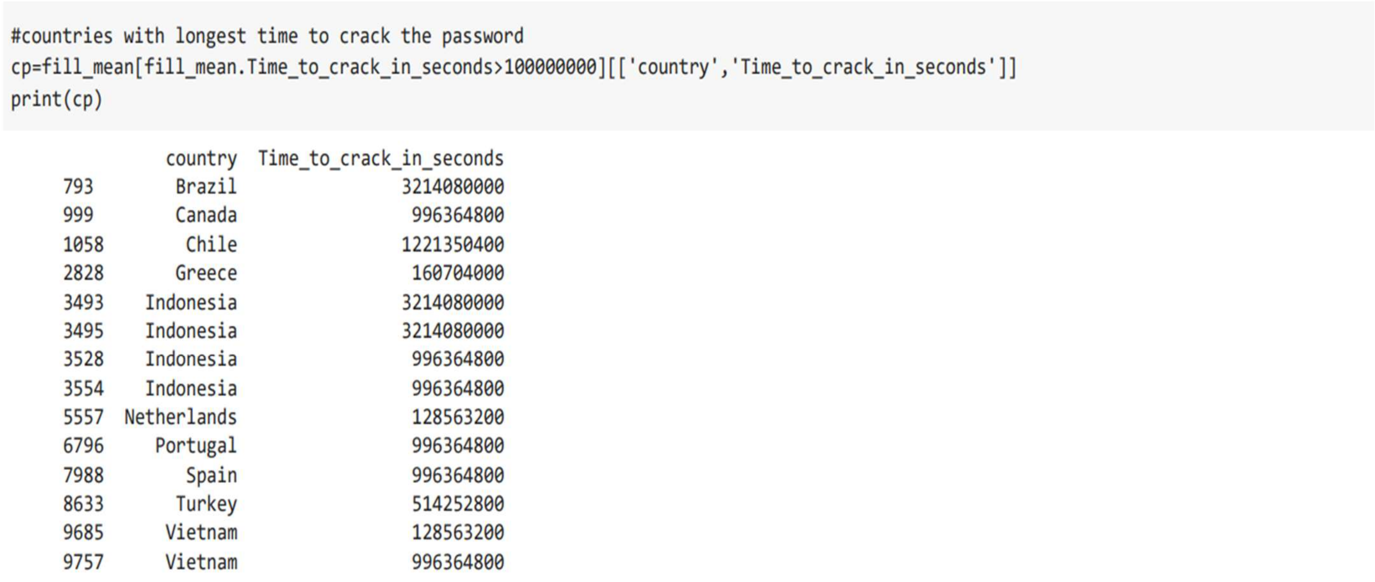


```
      Global_rank  Time_to_crack_in_seconds
0        1.000000                         0
1        5.000000                         0
2       65.337011                     10800
3       16.000000                         0
4        2.000000                         0
...           ...                       ...
9795    65.337011                     10800
9796    65.337011                      1020
9797    65.337011                     10800
9798    65.337011                      7200
9799    65.337011                     10800

[9800 rows x 8 columns]
```



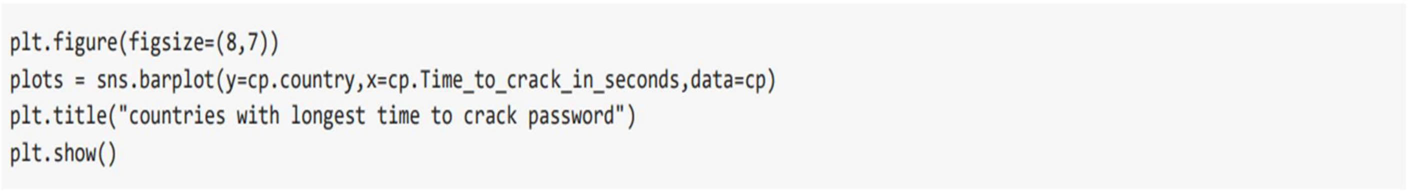
```
print(fill_mean.isnull().sum())
```

```
country_code               0
country                    0
Rank                       0
Password                   0
User_count                 0
Time_to_crack              0
Global_rank                0
Time_to_crack_in_seconds   0
dtype: int64
```

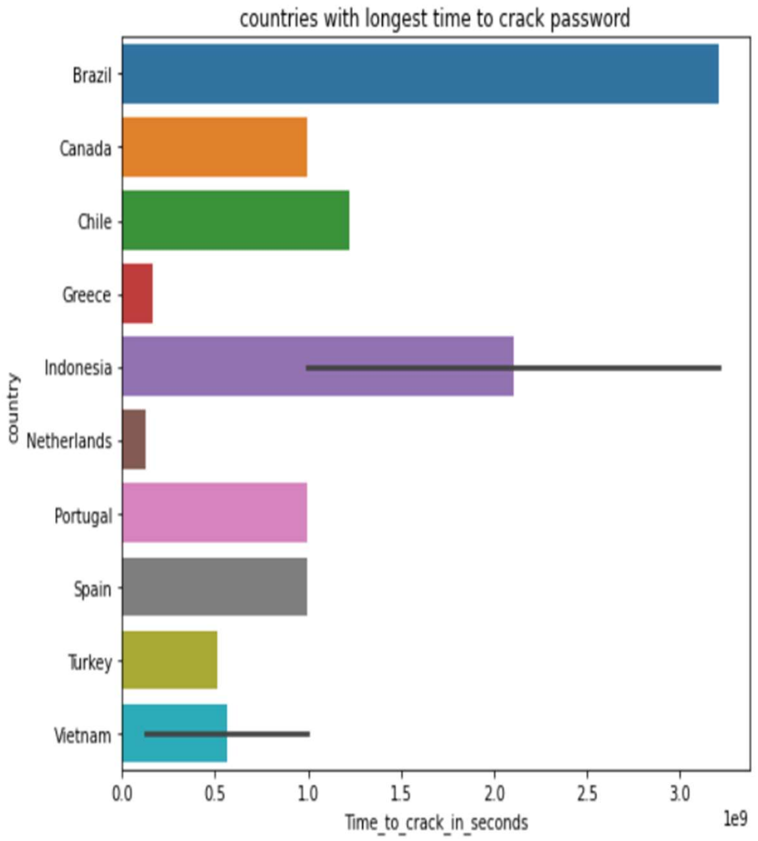


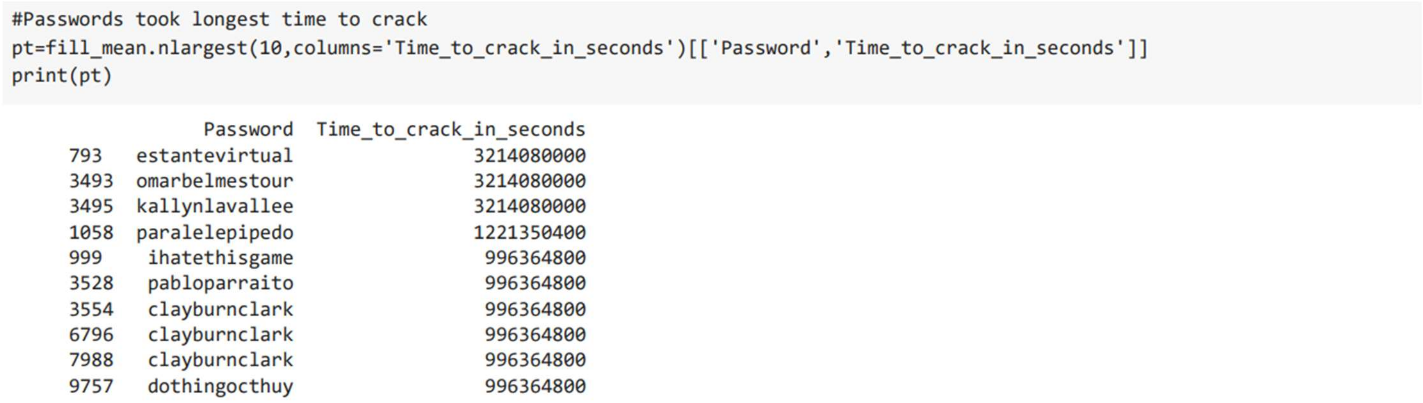
```
#countries with longest time to crack the password
cp=fill_mean[fill_mean.Time_to_crack_in_seconds>100000000][['country','Time_to_crack_in_seconds']]
print(cp)
```

```
          country  Time_to_crack_in_seconds
793        Brazil                3214080000
999        Canada                 996364800
1058        Chile                1221350400
2828       Greece                 160704000
3493    Indonesia                3214080000
3495    Indonesia                3214080000
3528    Indonesia                 996364800
3554    Indonesia                 996364800
5557  Netherlands                 128563200
6796     Portugal                 996364800
7988        Spain                 996364800
8633       Turkey                 514252800
9685      Vietnam                 128563200
9757      Vietnam                 996364800
```



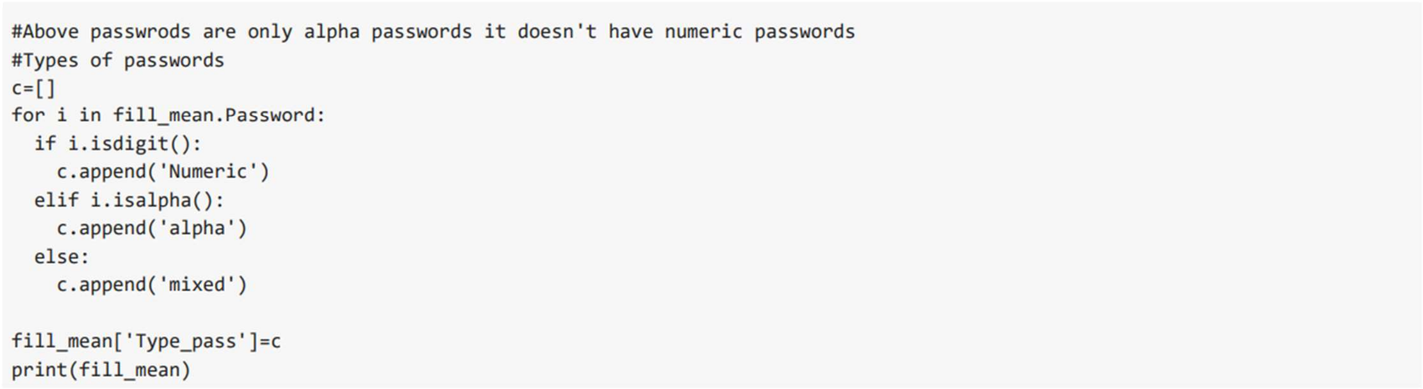
```
plt.figure(figsize=(8,7))
plots = sns.barplot(y=cp.country,x=cp.Time_to_crack_in_seconds,data=cp)
plt.title("countries with longest time to crack password")
plt.show()
```



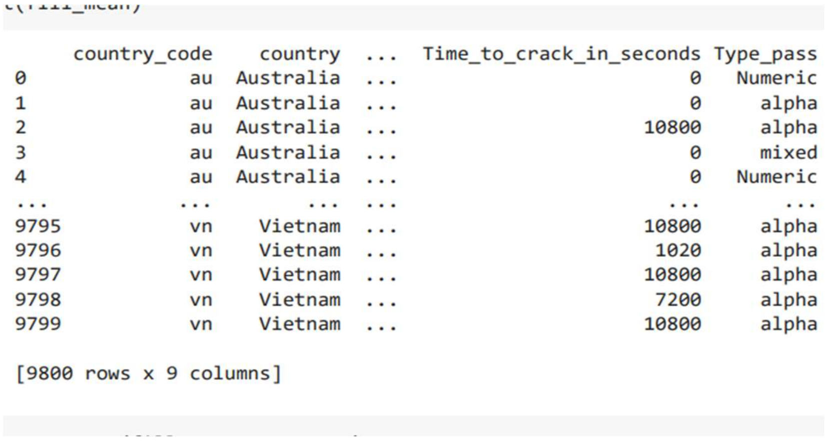


```
#Passwords took longest time to crack
pt=fill_mean.nlargest(10,columns='Time_to_crack_in_seconds')[['Password','Time_to_crack_in_seconds']]
print(pt)
```

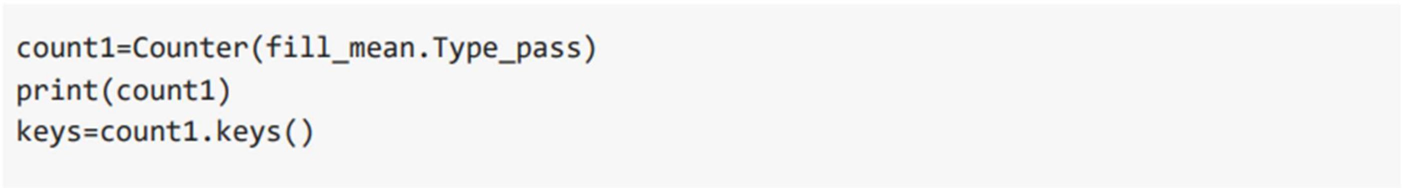
```
               Password  Time_to_crack_in_seconds
     793   estantevirtual                3214080000
     3493  omarbelmestour                3214080000
     3495  kallynlavallee                3214080000
     1058  paralelepipedo                1221350400
     999    ihatethisgame                 996364800
     3528   pabloparraito                 996364800
     3554   clayburnclark                 996364800
     6796   clayburnclark                 996364800
     7988   clayburnclark                 996364800
     9757   dothingocthuy                 996364800
```



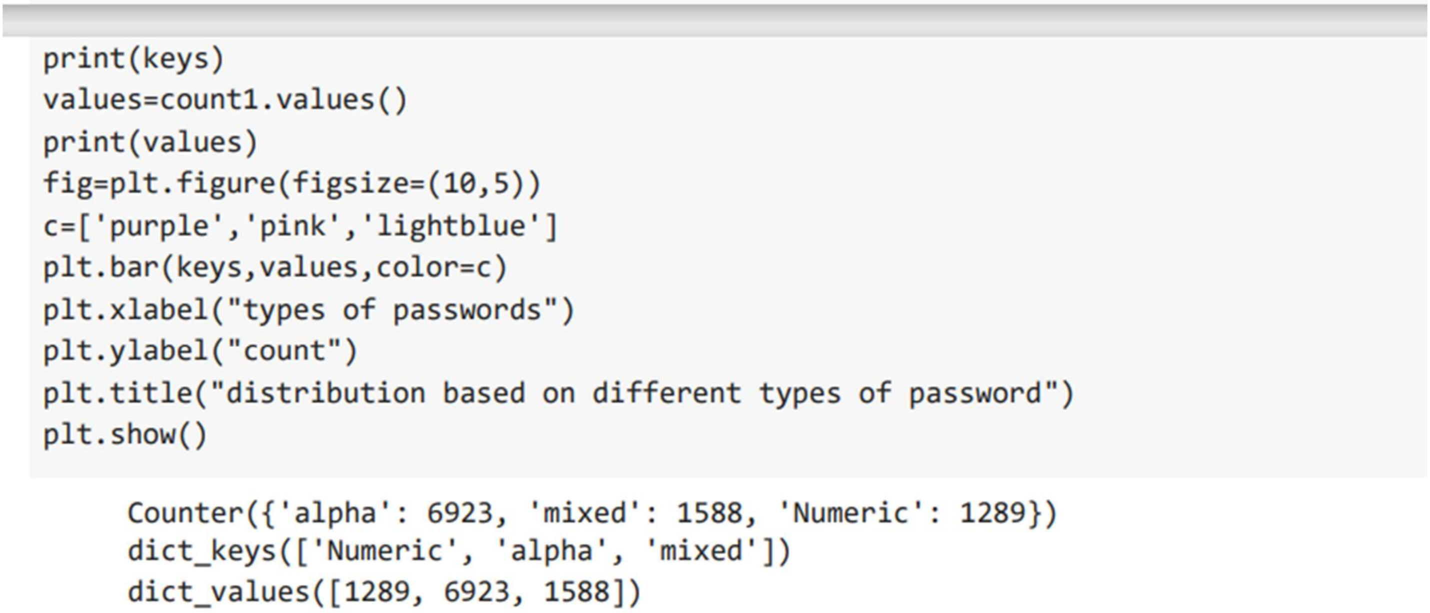
```
#Above passwrods are only alpha passwords it doesn't have numeric passwords
#Types of passwords
c=[]
for i in fill_mean.Password:
  if i.isdigit():
    c.append('Numeric')
  elif i.isalpha():
    c.append('alpha')
  else:
    c.append('mixed')

fill_mean['Type_pass']=c
print(fill_mean)
```



```
     country_code    country  ...  Time_to_crack_in_seconds Type_pass
0              au  Australia  ...                         0   Numeric
1              au  Australia  ...                         0     alpha
2              au  Australia  ...                     10800     alpha
3              au  Australia  ...                         0     mixed
4              au  Australia  ...                         0   Numeric
...           ...        ...  ...                       ...       ...
9795           vn    Vietnam  ...                     10800     alpha
9796           vn    Vietnam  ...                      1020     alpha
9797           vn    Vietnam  ...                     10800     alpha
9798           vn    Vietnam  ...                      7200     alpha
9799           vn    Vietnam  ...                     10800     alpha

[9800 rows x 9 columns]
```

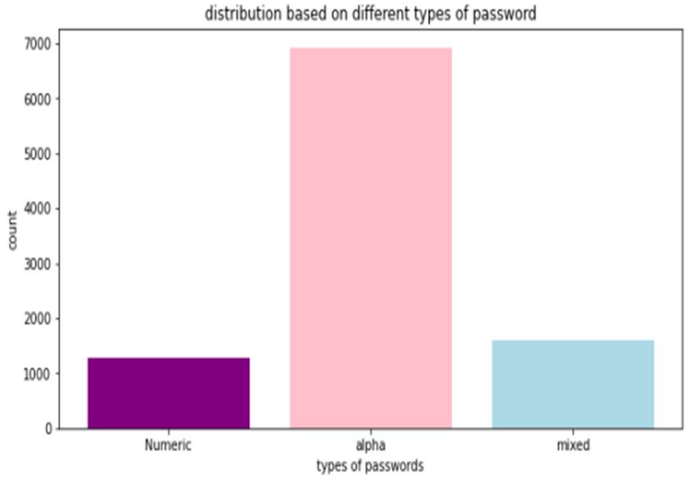


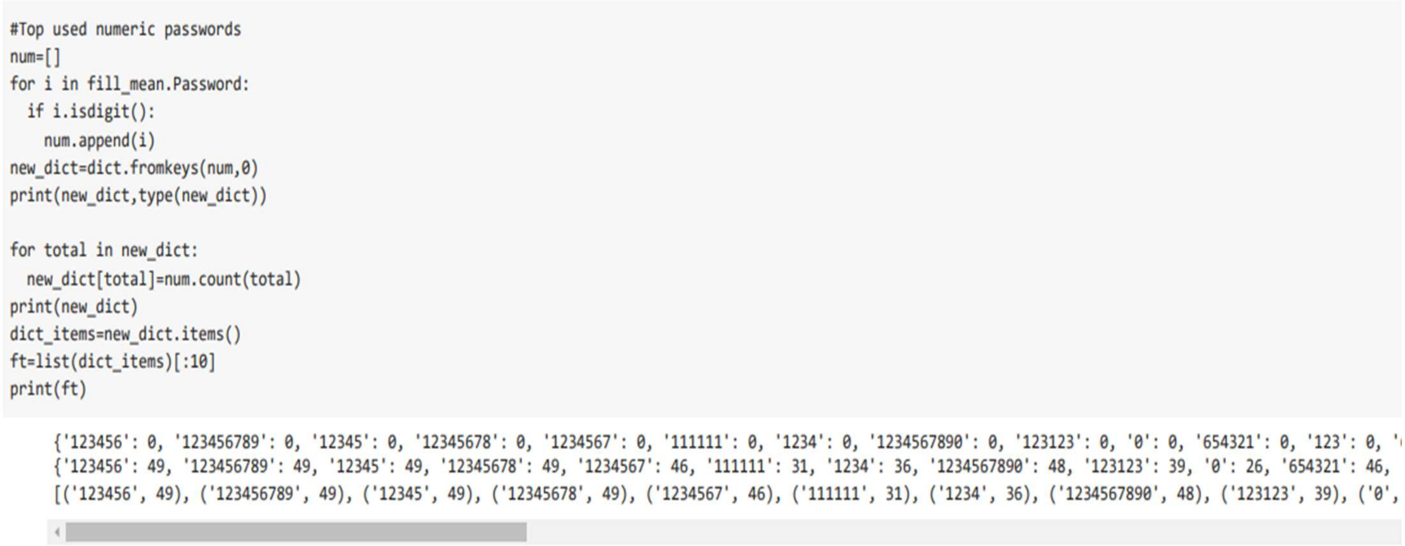
```
count1=Counter(fill_mean.Type_pass)
print(count1)
keys=count1.keys()
```



```
print(keys)
values=count1.values()
print(values)
fig=plt.figure(figsize=(10,5))
c=['purple','pink','lightblue']
plt.bar(keys,values,color=c)
plt.xlabel("types of passwords")
plt.ylabel("count")
plt.title("distribution based on different types of password")
plt.show()
```

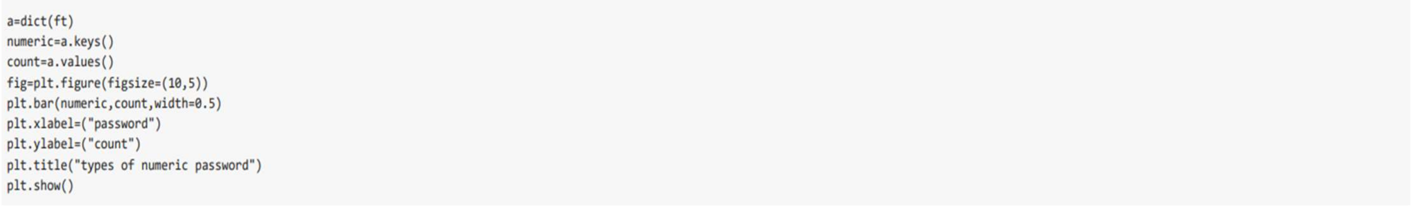
```
Counter({'alpha': 6923, 'mixed': 1588, 'Numeric': 1289})
dict_keys(['Numeric', 'alpha', 'mixed'])
dict_values([1289, 6923, 1588])
```



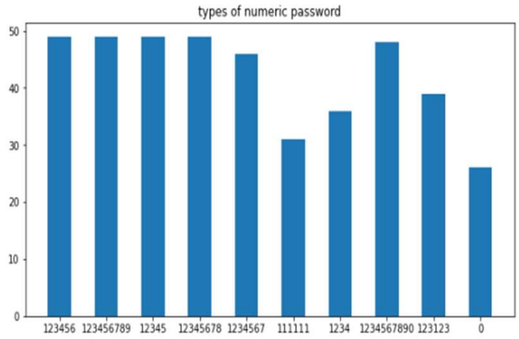


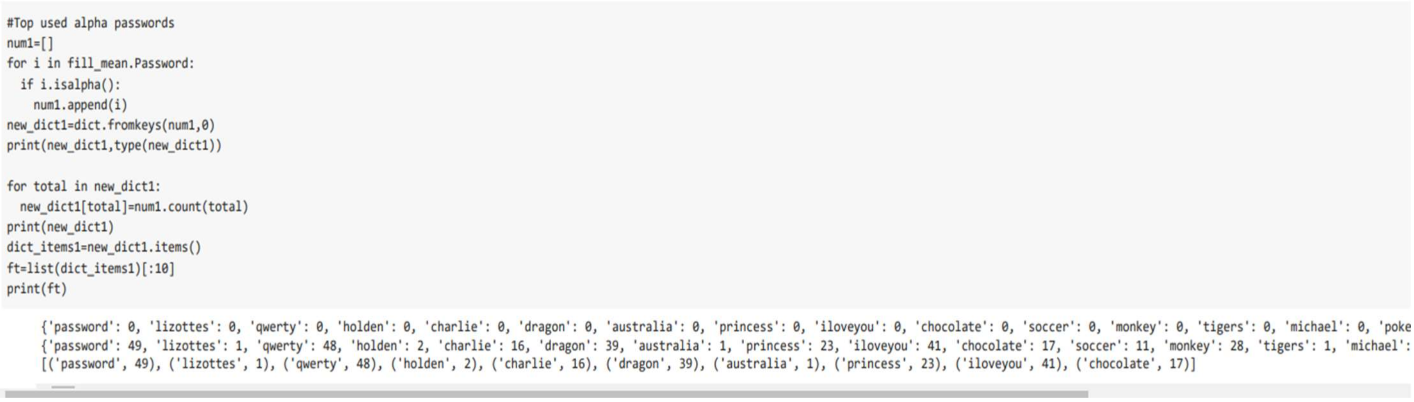
```
#Top used numeric passwords
num=[]
for i in fill_mean.Password:
  if i.isdigit():
    num.append(i)
new_dict=dict.fromkeys(num,0)
print(new_dict,type(new_dict))

for total in new_dict:
  new_dict[total]=num.count(total)
print(new_dict)
dict_items=new_dict.items()
ft=list(dict_items)[:10]
print(ft)
```

```
{'123456': 0, '123456789': 0, '12345': 0, '12345678': 0, '1234567': 0, '111111': 0, '1234': 0, '1234567890': 0, '123123': 0, '0': 0, '654321': 0, '123': 0, '
{'123456': 49, '123456789': 49, '12345': 49, '12345678': 49, '1234567': 46, '111111': 31, '1234': 36, '1234567890': 48, '123123': 39, '0': 26, '654321': 46,
[('123456', 49), ('123456789', 49), ('12345', 49), ('12345678', 49), ('1234567', 46), ('111111', 31), ('1234', 36), ('1234567890', 48), ('123123', 39), ('0',
```



```
a=dict(ft)
numeric=a.keys()
count=a.values()
fig=plt.figure(figsize=(10,5))
plt.bar(numeric,count,width=0.5)
plt.xlabel=("password")
plt.ylabel=("count")
plt.title("types of numeric password")
plt.show()
```

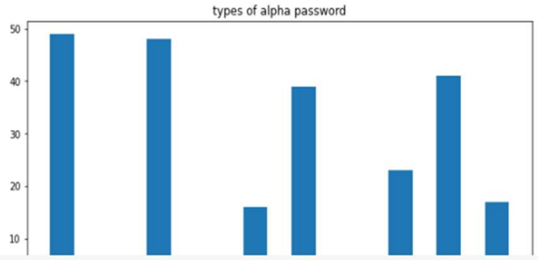




```
#Top used alpha passwords
num1=[]
for i in fill_mean.Password:
  if i.isalpha():
    num1.append(i)
new_dict1=dict.fromkeys(num1,0)
print(new_dict1,type(new_dict1))

for total in new_dict1:
  new_dict1[total]=num1.count(total)
print(new_dict1)
dict_items1=new_dict1.items()
ft=list(dict_items1)[:10]
print(ft)
```

```
{'password': 0, 'lizottes': 0, 'qwerty': 0, 'holden': 0, 'charlie': 0, 'dragon': 0, 'australia': 0, 'princess': 0, 'iloveyou': 0, 'chocolate': 0, 'soccer': 0, 'monkey': 0, 'tigers': 0, 'michael': 0, 'poke
{'password': 49, 'lizottes': 1, 'qwerty': 48, 'holden': 2, 'charlie': 16, 'dragon': 39, 'australia': 1, 'princess': 23, 'iloveyou': 41, 'chocolate': 17, 'soccer': 11, 'monkey': 28, 'tigers': 1, 'michael':
[('password', 49), ('lizottes', 1), ('qwerty', 48), ('holden', 2), ('charlie', 16), ('dragon', 39), ('australia', 1), ('princess', 23), ('iloveyou', 41), ('chocolate', 17)]
```

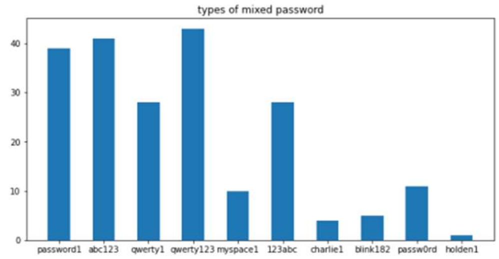
```
a=dict(ft)
alpha=a.keys()
count=a.values()
fig=plt.figure(figsize=(10,5))
plt.bar(alpha,count,width=0.5)
plt.xlabel=("password")
plt.ylabel=("count")
plt.title("types of alpha password")
plt.show()
```



```
#Top used mixed passwords
num2=[]
for i in fill_mean.Password:
  if (i.isalpha()==False and i.isdigit()==False) :
        num2.append(i)
new_dict2=dict.fromkeys(num2,0)
print(new_dict2,type(new_dict2))

for total in new_dict2:
  new_dict2[total]=num2.count(total)
print(new_dict2)
dict_items2=new_dict2.items()
ft=list(dict_items2)[:10]
print(ft)
```

```
{'password1': 0, 'abc123': 0, 'qwerty1': 0, 'qwerty123': 0, 'myspace1': 0, '123abc': 0, 'charlie1': 0, 'blink182': 0, 'passw0rd': 0, 'holden1': 0, 'trustno1': 0, 'hello123': 0, 'hello1': 0, 'australia1'
{'password1': 39, 'abc123': 41, 'qwerty1': 28, 'qwerty123': 43, 'myspace1': 10, '123abc': 28, 'charlie1': 4, 'blink182': 5, 'passw0rd': 11, 'holden1': 1, 'trustno1': 7, 'hello123': 6, 'hello1': 2, 'aust
[('password1', 39), ('abc123', 41), ('qwerty1', 28), ('qwerty123', 43), ('myspace1', 10), ('123abc', 28), ('charlie1', 4), ('blink182', 5), ('passw0rd', 11), ('holden1', 1)]
```

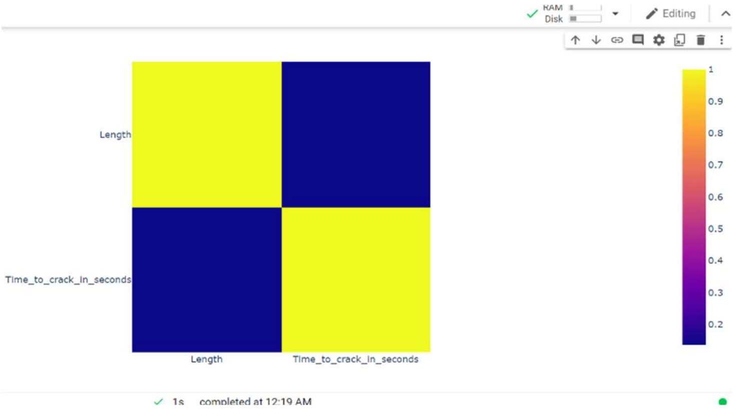
```
a=dict(ft)
mixed=a.keys()
count=a.values()
fig=plt.figure(figsize=(10,5))
plt.bar(mixed,count,width=0.5)
plt.xlabel=("password")
plt.ylabel=("count")
plt.title("types of mixed password")
plt.show()
```



```
fill_mean['Length']=fill_mean['Password'].str.len()
print(fill_mean)
```

```
      country_code    country  ...  Time_to_crack_in_seconds Length
0               au  Australia  ...                         0      6
1               au  Australia  ...                         0      8
2               au  Australia  ...                     10800      8
3               au  Australia  ...                         0      9
4               au  Australia  ...                         0      9
...            ...        ...  ...                       ...    ...
9795            vn    Vietnam  ...                     10800      8
9796            vn    Vietnam  ...                      1020      7
9797            vn    Vietnam  ...                     10800      8
9798            vn    Vietnam  ...                      7200      8
9799            vn    Vietnam  ...                     10800      8

[9800 rows x 9 columns]
```

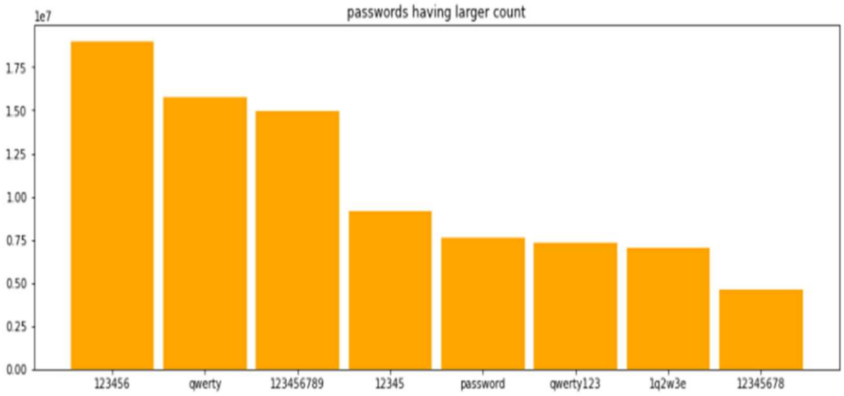
```
#correlation between password length and time to crack
fill_mean['Length']=fill_mean['Password'].str.len()
dp=fill_mean[['Length','Time_to_crack_in_seconds']].corr()
px.imshow(dp)
```



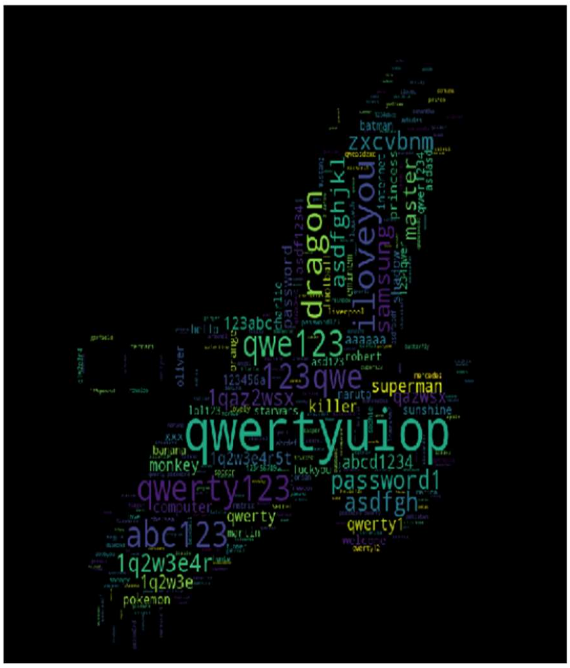
```
#bar plot between user count and password
a=fill_mean.nlargest(10,columns="User_count")[['Password','User_count']]
a
```

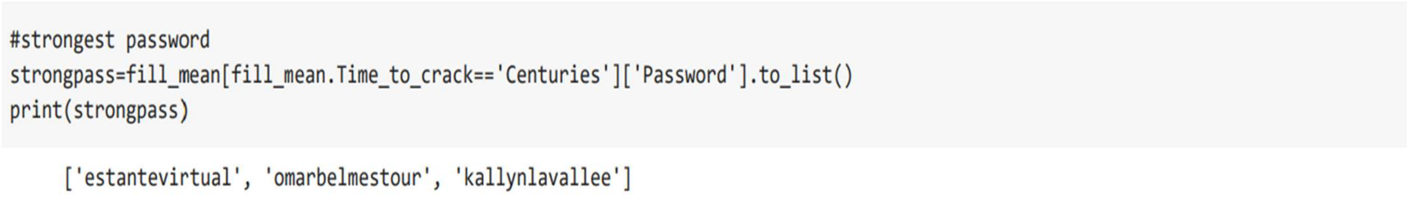
```
pass1=a['Password']
print(a)
user=a['User_count']
fig=plt.figure(figsize=(15,5))
plt.bar(pass1,user,color='orange',width=0.9)
plt.xlabel=("Password")
plt.ylabel=("User count")
plt.title("passwords having larger count")
plt.show()
```

```
       Password  User_count
7000     123456    19000630
7001     qwerty    15738011
7002  123456789    14975791
7003      12345     9139679
1200     123456     8159358
7004   password     7593503
7005  qwerty123     7343013
7006     1q2w3e     7051194
7007   12345678     4632962
9400     123456     3572081
```



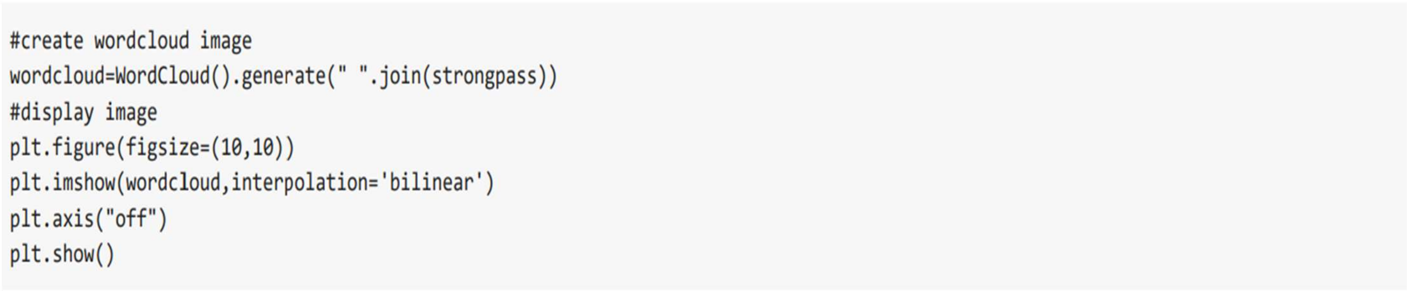
```
#weekest password
mask=np.array(Image.open("/content/birdimage.jpg"))
mask
weekpass=fill_mean[fill_mean.Time_to_crack_in_seconds==0]['Password'].to_list()
#print(weekpass)
wc=WordCloud(stopwords=STOPWORDS,mask=mask,background_color="black",
             max_words=5000,max_font_size=1000,random_state=99,
             width=mask.shape[1],height=mask.shape[0])
wc.generate(" ".join(weekpass))
plt.figure(figsize=(10,10),facecolor='k')
plt.imshow(wc,interpolation='None')
plt.axis("off")
plt.show()
```



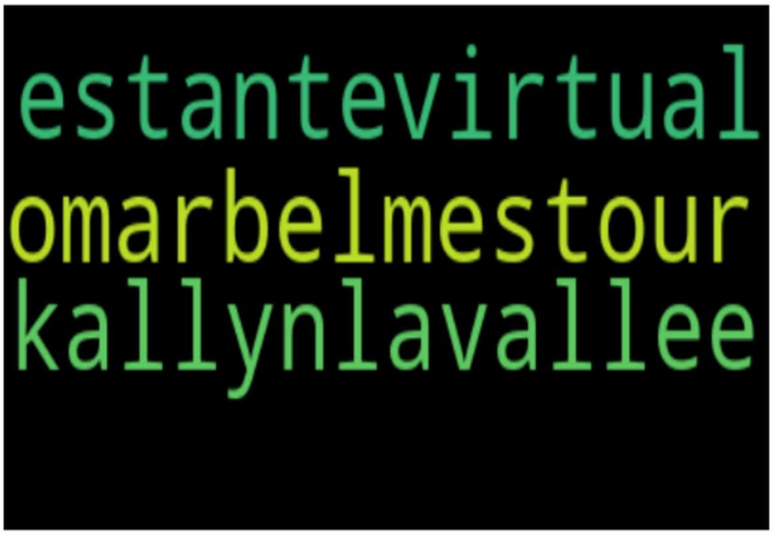


```
#strongest password
strongpass=fill_mean[fill_mean.Time_to_crack=='Centuries']['Password'].to_list()
print(strongpass)
```

```
    ['estantevirtual', 'omarbelmestour', 'kallynlavallee']
```



```
#create wordcloud image
wordcloud=WordCloud().generate(" ".join(strongpass))
#display image
plt.figure(figsize=(10,10))
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis("off")
plt.show()
```



5) <u>Data Visualization & Interpretation</u>

- **In countries we see Brazil country take longest time to crack the password.**
- **We can see the alpha passwords are mostly used in all the countries.**

- **In numeric password pattern (1234567890) , In alpha pattern (password/qwerty) , In mixed (qwerty123) paaword are used most rather than remaining patterns.**
- **Password (123456) has large user_count in all countries.**
- **In we have analyze the weakest passwords with the help of word cloud.**
- **By word cloud image qwertyuiop is bolder so this is the weakest password.**
- **In this we have analyse the strongest passwords with the help of word cloud.**
- **In this case kallynlavallee is bolder so this is the strongest password.**

6) Conclusion

- **In this project we have analyzed the which countries take longer time to crack the passwords, password which took longer time to crack, types of passwords, strongest and weakest passwords.**
- **finally we concluded that we have to use alpha_password or mixed password because it is difficult to crack that type of password.**

- **For data processing we used data cleaning method.**
- **For analysis we used bar chart, horizontal bar chart, heat map and word cloud.**

7) Bibliography

1) ***Kaggle, Top_200_comman_password_by_different_countrries[Online]. Available: http://www.kaggle.com/***

2) ***https://www.analyticsvidhya.com/machine-learning.***

3) ***Wikipedia. Logistic Regression [Online]. Available:- https://en.wikipedia.org/wiki/Logistic_regression.***

4) ***"The 200 Passwords of 2022 Are Here and Oh My God". gizmodo.com.***

5) ***Korosec, Kirsten (December 19, 2017). "The 25 Most Common Passwords of 2017 Include 'Star Wars'". FORTUNE.***