TOP 200 COMMON PASSWORDS BY DIFFERENT COUNTRY 2022....

By

AKSHATA SHIRWALE

&

PAURAVI KHATPE

UNDER THE GUIDEENCE OF :-

PROF . SUSHMITA BHAMBHURE

Data Visualization

- Data visualization is the secret art of turning data into visual graphics that people can understand (graphs, charts, info graphics, etc.).

- Here are a few additional statistics highlighting the importance of data visualization over text when presenting information:
  - 90% of the information transmitted to the brain is visual
  - Humans process images 60,000 times faster than text
  - 70% of our sensory receptors are in our eyes
  - 65% of people are visual learners

- By using visual elements like **charts**, **graphs**, and **maps**, data visualization techniques provide an accessible way to see and **understand trends, outliers, and patterns in data**.

Benefits of Good Data Visualization

- Whenever we visualize a chart, we quickly identify the trends and outliers present in the dataset.

- The basic uses of the Data Visualization technique are as follows :-

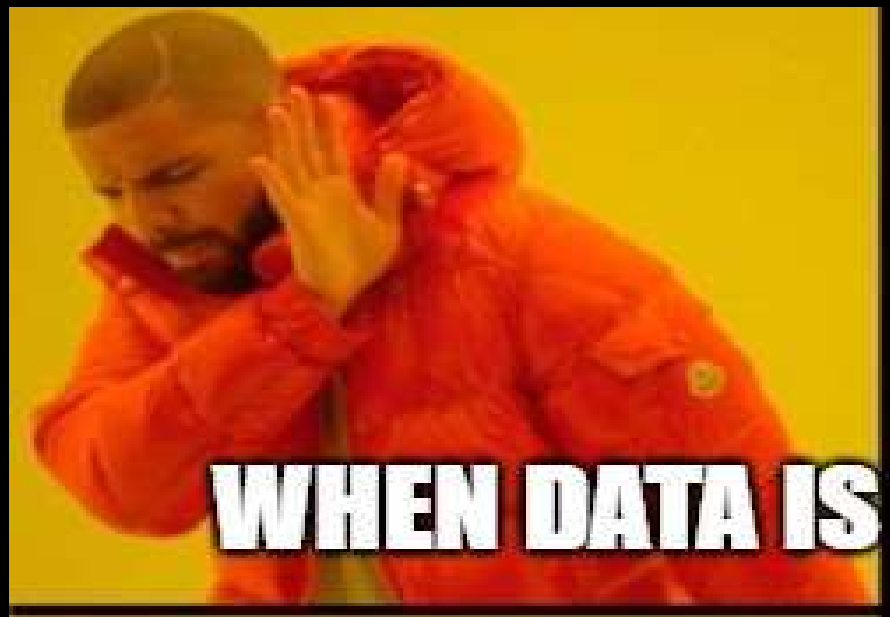
  - It is a powerful technique to explore the data with **presentable** and **interpretable** results.

  - In the **data mining process**, it acts as a primary step in the pre-processing portion.

  - It supports the **data cleaning process** by finding incorrect data and corrupted or missing values.

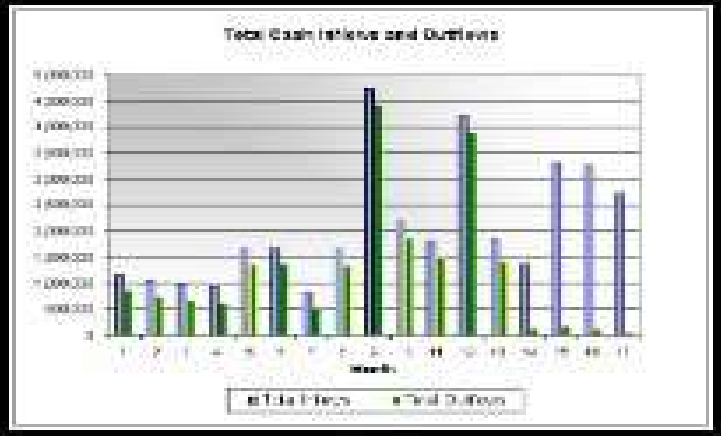
  - It also helps to **construct and select variables**, which means we have to determine which variable to include and discard in the analysis.

  - In the process of **Data Reduction**, it also plays a crucial role while combining the categories.



ID	NAME	CLASS	MARK	SEX
1	John Doe	Four	75	female
2	Max Ruin	Three	85	male
3	Arnold	Three	90	male
4	Priya Sher	Four	80	female
5	John Mike	Four	60	female
6	Alex John	Four	60	male
7	My John Rob	Fifth	79	male
8	Arnold	Five	85	male
9	Tim Cay	Six	70	male
10	Big John	Four	55	female

WHEN DATA IS IN TABLE FORM



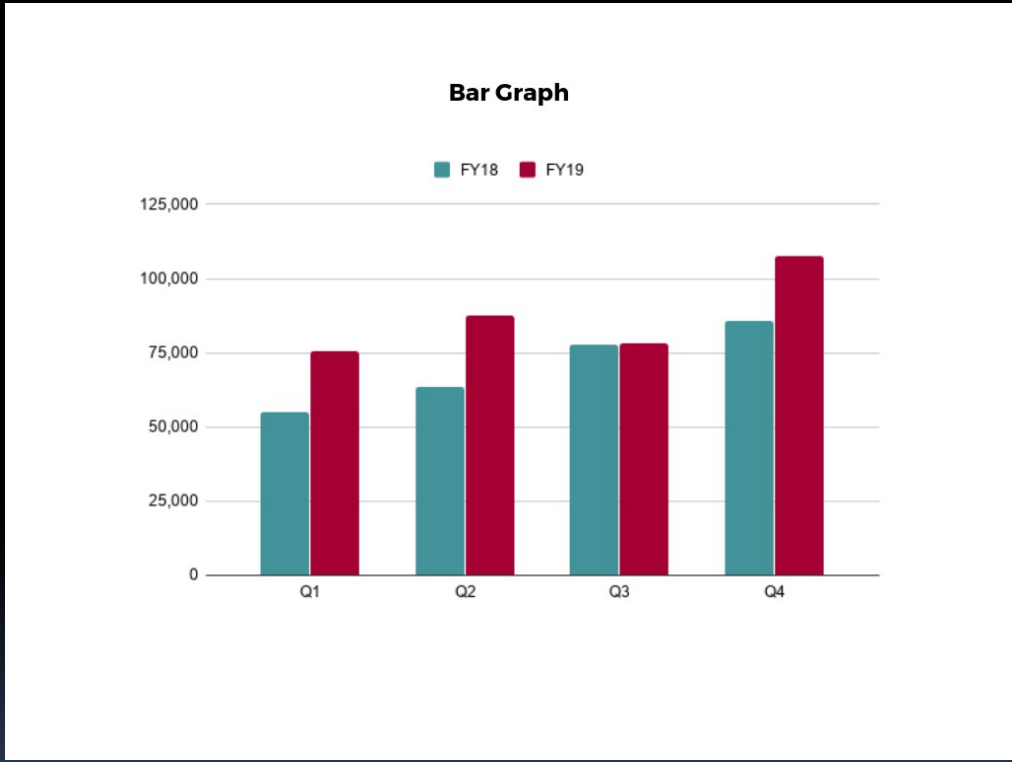
WHEN DATA IS IN PLOT

DATA VISUALIZATION TECHNIQUES

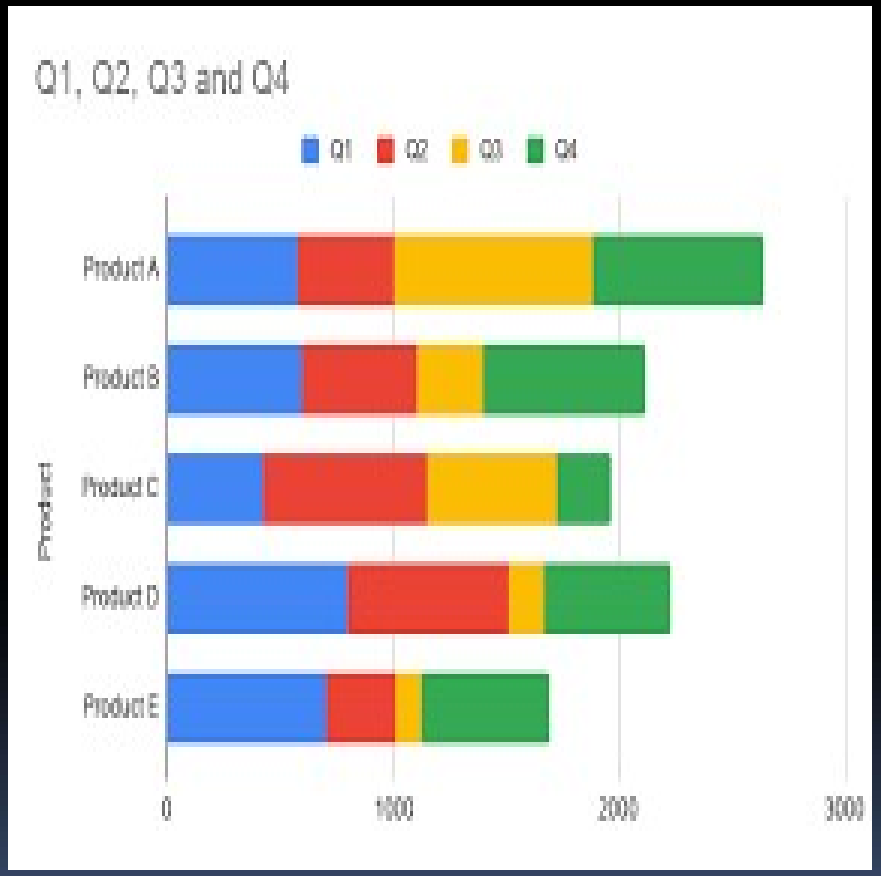
- Bar Chart
- Histogram
- Heat Map
- Box and Whisker Plot
- Waterfall Chart
- Area Chart
- Scatter Plot
- Pictogram Chart
- Timeline
- Highlight Table
- Bullet Graph
- Choropleth Map
- Word Cloud
- Network Diagram
- Correlation Matrices

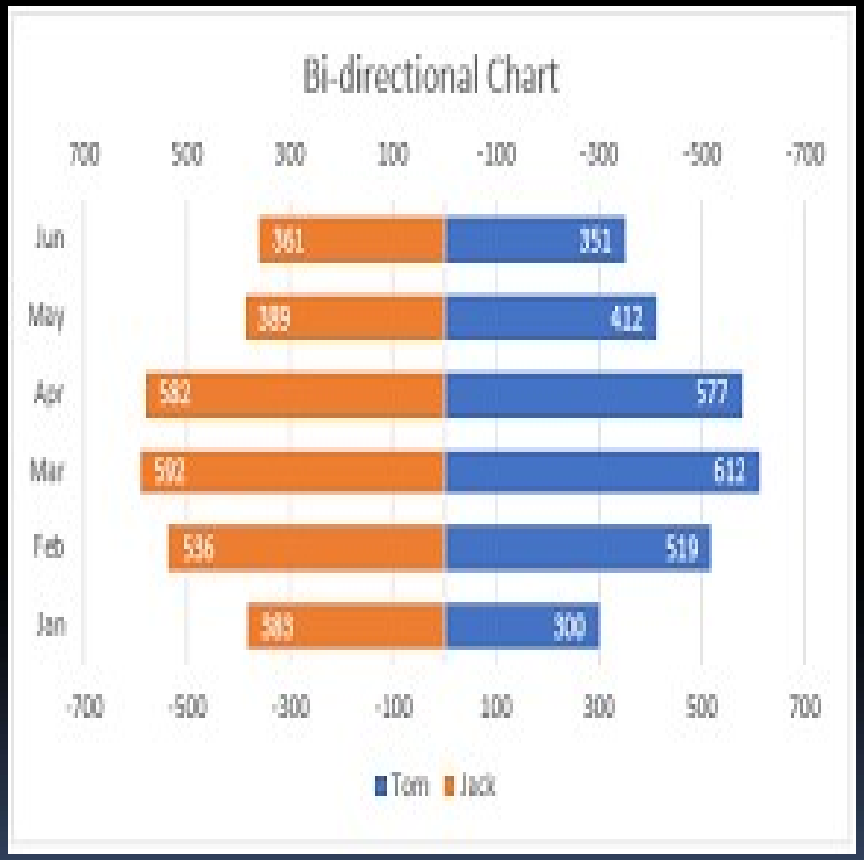
1. Bar Chart

In this type of visualization, one axis of the chart shows the categories being compared, and the other, a measured value. The length of the bar indicates how each group measures according to the value.



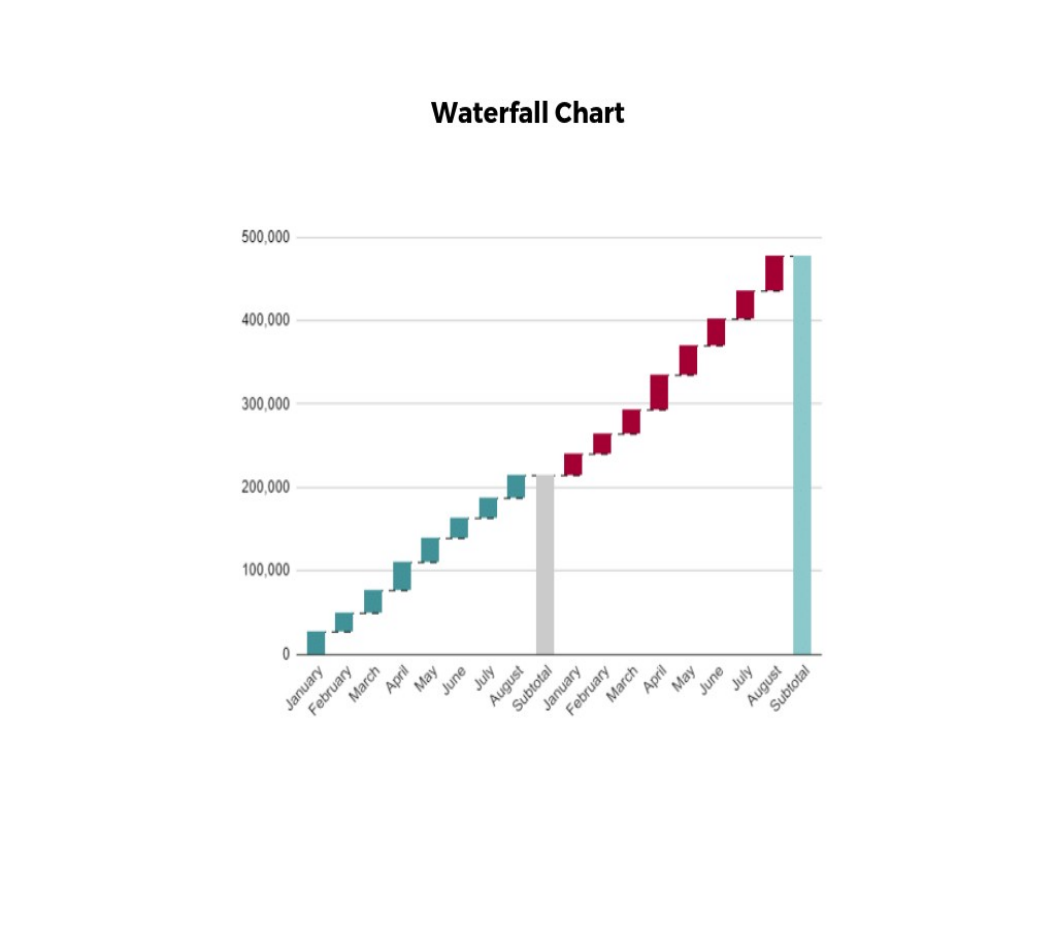
2. Horizontal and Bi-directional





3. Waterfall Chart

The main goal of this chart is to show the viewer how a value has grown or declined over a defined period. For example, waterfall charts are popular for showing spending or earnings over time.

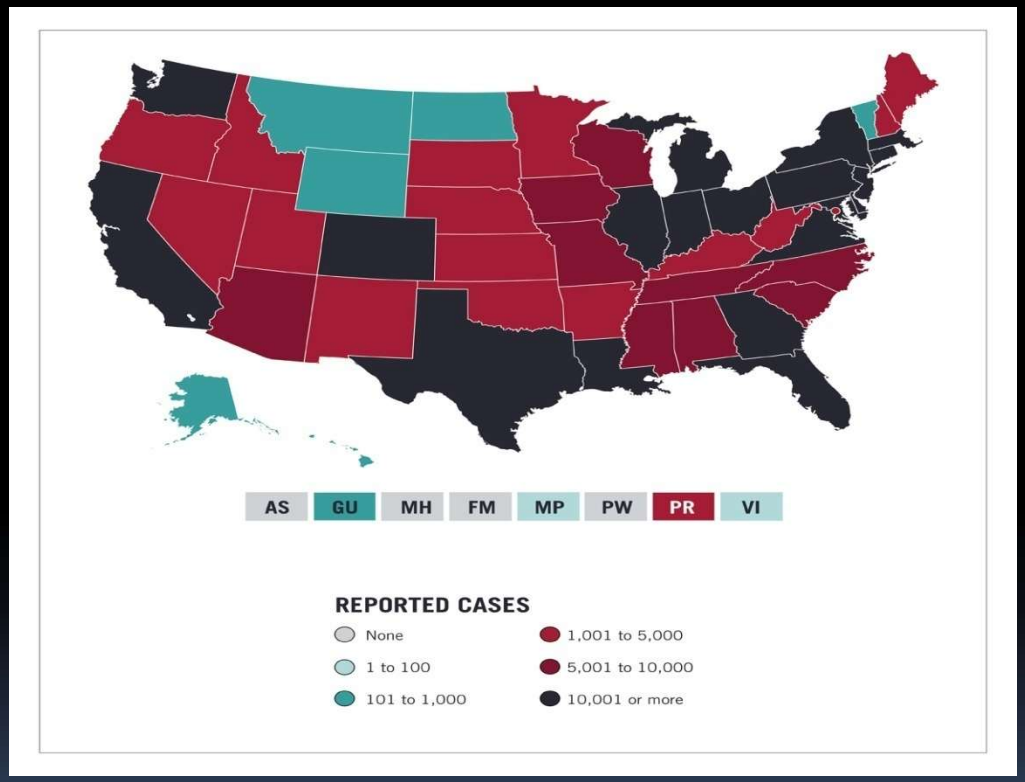


4.Infographic Example

An infographic is a collection of imagery, data visualizations like pie charts and bar graphs, and minimal text that gives an easy-to-understand overview of a topic.

5.Choro pleth Maps

A choropleth map uses color, shading, and other patterns to visualize numerical values across geographic regions. These visualizations use a progression of color (or shading) on a spectrum to distinguish high values from low.



Top 200 Common Passwords By Different Country 2022

- 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)

We have used data processing techniques on the data

- There are techniques we can use for data processing.
- The one we used for our data is data cleaning.
- To begin with, check the null values in the data, isnull() is used to check null values.
- If there are any null values replace it with mean or median values , or remove null values, or use forward or backword fill.
- We replaced it with mean values in place of null values

First we have imported the libraries and then displayed the data

- Libraries like seaborn and matplotlib, wordcloud are used for data visualization.
- For displaying data we use pd.read_csv.
- df.shape displays the size of the data.
- is.null().sum() checks the null values in data and sum gives the total of null values in each column.

+ Code + Text

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
from wordcloud import WordCloud
```

+ Code + Text

```
missing_values=['N/a',"na",np.nan]
df=pd.read_csv('/content/top_200_password_2020_by_country (1).csv',na_values=missing_values)
print(df)
print("size of data is:",df.shape)
print("\n",df.isnull().sum())
```

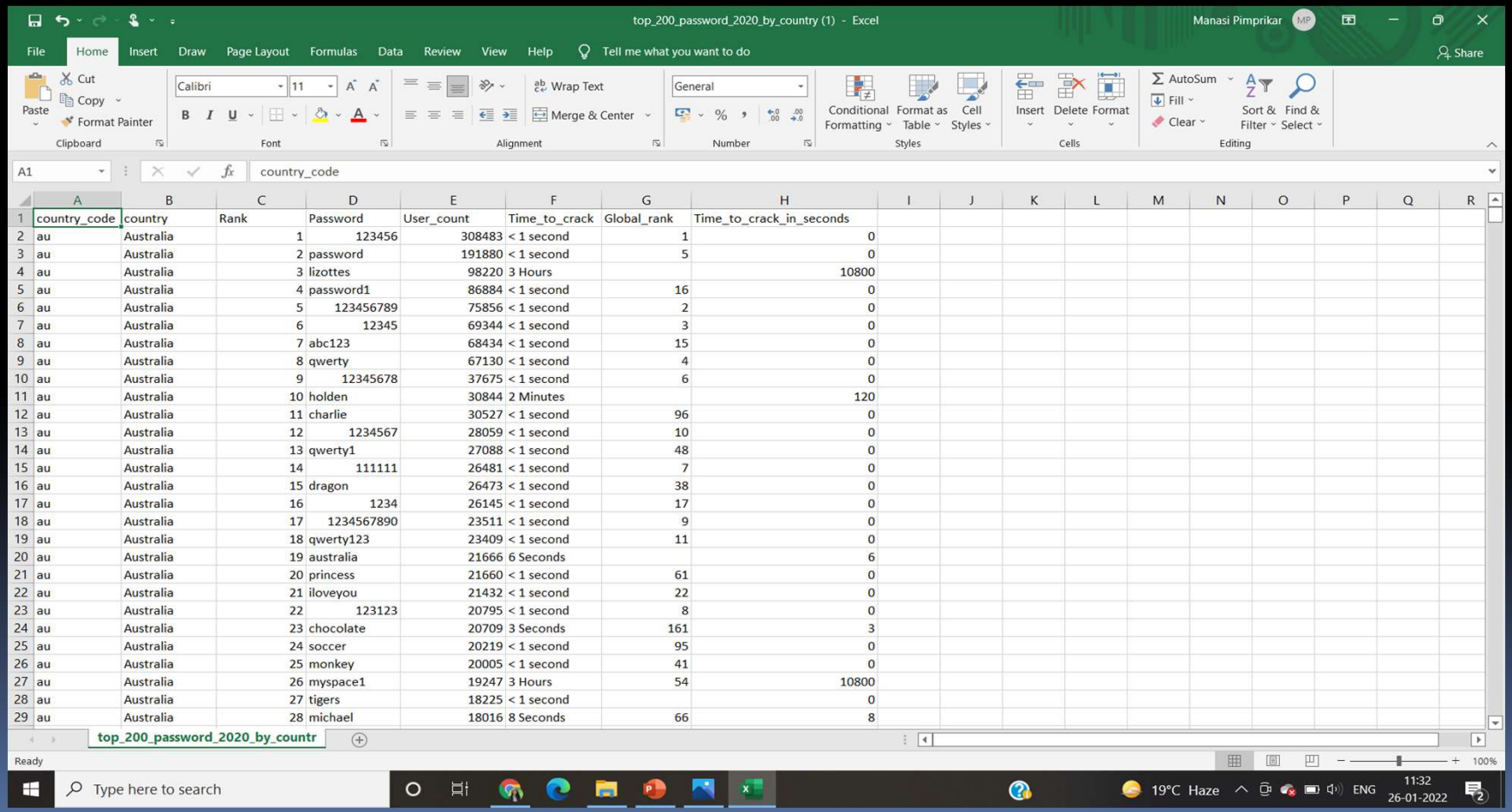
```
     country_code    country  ...  Global_rank Time_to_crack_in_seconds
0              au  Australia  ...          1.0                       0
1              au  Australia  ...          5.0                       0
2              au  Australia  ...          NaN                   10800
3              au  Australia  ...         16.0                       0
4              au  Australia  ...          2.0                       0
...           ...        ...  ...          ...                     ...
9795           vn    Vietnam  ...          NaN                   10800
9796           vn    Vietnam  ...          NaN                    1020
9797           vn    Vietnam  ...          NaN                   10800
9798           vn    Vietnam  ...          NaN                    7200
9799           vn    Vietnam  ...          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
```

✓ 0s completed at 11:20

This is our data

This is the link of data from kaggle

kaggle datasets download -d prasertk/top-200-passwords-by-country-2022

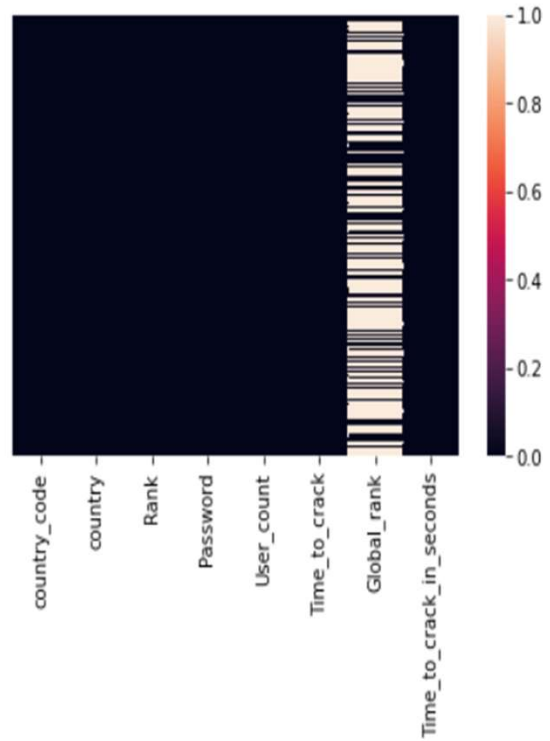


country_code	country	Rank	Password	User_count	Time_to_crack	Global_rank	Time_to_crack_in_seconds
au	Australia	1	123456	308483	< 1 second	1	0
au	Australia	2	password	191880	< 1 second	5	0
au	Australia	3	lizottes	98220	3 Hours		10800
au	Australia	4	password1	86884	< 1 second	16	0
au	Australia	5	123456789	75856	< 1 second	2	0
au	Australia	6	12345	69344	< 1 second	3	0
au	Australia	7	abc123	68434	< 1 second	15	0
au	Australia	8	qwerty	67130	< 1 second	4	0
au	Australia	9	12345678	37675	< 1 second	6	0
au	Australia	10	holden	30844	2 Minutes		120
au	Australia	11	charlie	30527	< 1 second	96	0
au	Australia	12	1234567	28059	< 1 second	10	0
au	Australia	13	qwerty1	27088	< 1 second	48	0
au	Australia	14	111111	26481	< 1 second	7	0
au	Australia	15	dragon	26473	< 1 second	38	0
au	Australia	16	1234	26145	< 1 second	17	0
au	Australia	17	1234567890	23511	< 1 second	9	0
au	Australia	18	qwerty123	23409	< 1 second	11	0
au	Australia	19	australia	21666	6 Seconds		6
au	Australia	20	princess	21660	< 1 second	61	0
au	Australia	21	iloveyou	21432	< 1 second	22	0
au	Australia	22	123123	20795	< 1 second	8	0
au	Australia	23	chocolate	20709	3 Seconds	161	3
au	Australia	24	soccer	20219	< 1 second	95	0
au	Australia	25	monkey	20005	< 1 second	41	0
au	Australia	26	myspace1	19247	3 Hours	54	10800
au	Australia	27	tigers	18225	< 1 second		0
au	Australia	28	michael	18016	8 Seconds	66	8

- This is heat map showing the null values in Global_rank column.
- White area represents the null values and black is not null values.
- It checks the null values in data and represents it in heap map

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa6724b6a90>
```



- Now we know that we have null values in only one column, so first we have to create a variable of each column.
- Then we have to replace the null values with mean values, globalrank.mean() gives the mean of global rank column.

```
+ Code  + Text

[ ] 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
```

- In second picture we have displayed the data with mean values.
- Now if we check there are no null values.

```
+ Code  + Text

[ ] fill_mean=df.fillna(globalrank.mean())
    print(fill_mean)

      country_code    country  ...  Global_rank Time_to_crack_in_seconds
0               au  Australia  ...     1.000000                       0
1               au  Australia  ...     5.000000                       0
2               au  Australia  ...    65.337011                   10800
3               au  Australia  ...    16.000000                       0
4               au  Australia  ...     2.000000                       0
...            ...        ...  ...          ...                     ...
9795            vn    Vietnam  ...    65.337011                   10800
9796            vn    Vietnam  ...    65.337011                    1020
9797            vn    Vietnam  ...    65.337011                   10800
9798            vn    Vietnam  ...    65.337011                    7200
9799            vn    Vietnam  ...    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

✓ 0s  completed at 11:20
```

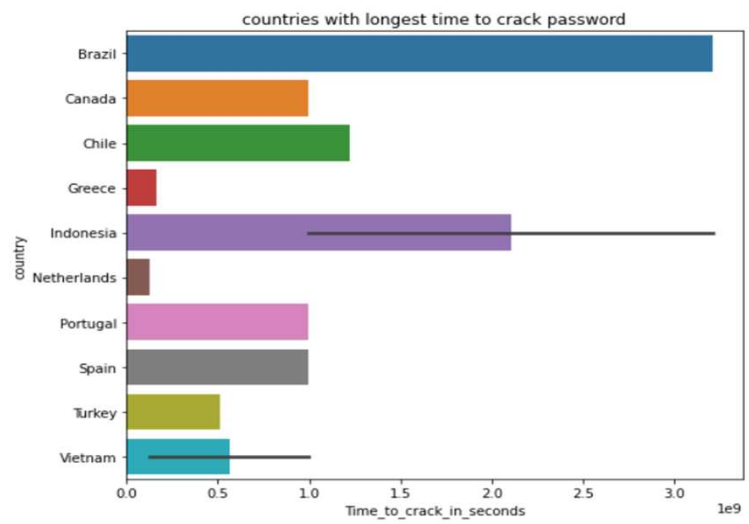
- Now we have to see the countries which take longest time to crack the password.
- For that we have used horizontal bar chart.
- The time should be >100000000.Output we get is two columns country and time to crack in seconds.

```
[15] #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

- The code in the picture represents the horizontal bar chart.
- The graph is plot with country against time to crack.

```
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()
```



- Now we will check for passwords who took longest time to crack.
- The 10 passwords which took longest time are printed.
- If we look the passwords are only alpha values, so for printing numeric, alpha and mixed passwords .
- So for that we have to make another column named type_pass which print password types.

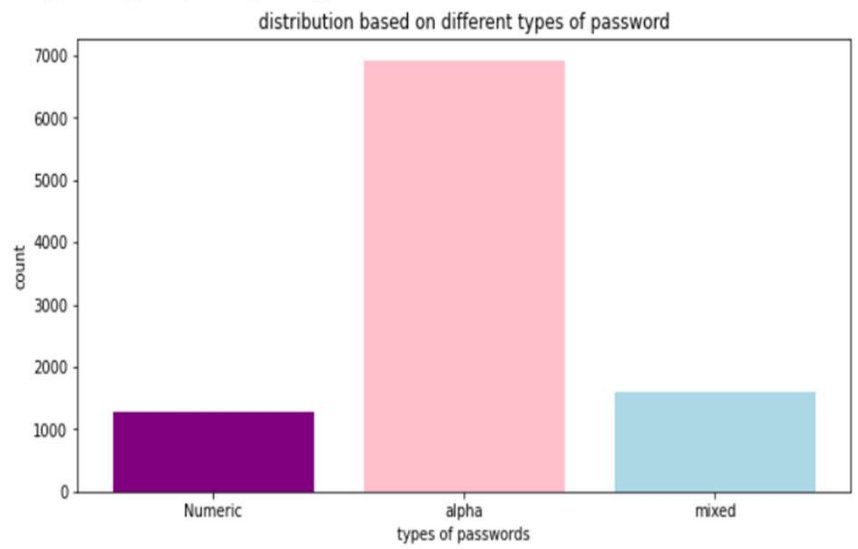
```
+ Code   + Text

[ ] #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]
```

- We have displayed here the bar chart with types of password against count.
- Counter is used to store elements as dictionary. Type_pass is used and it print elements as keys and count as values
- We can see that alpha passwords are used more.

```
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])
```

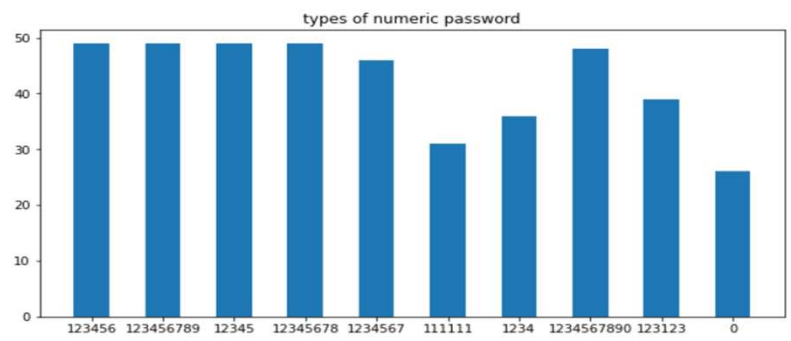


- Top 10 numeric passwords are displayed.
- The graph is plot numeric passwords against count
- And from this graph we can tell that 1-4 passwords have same count.

```
#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()
```



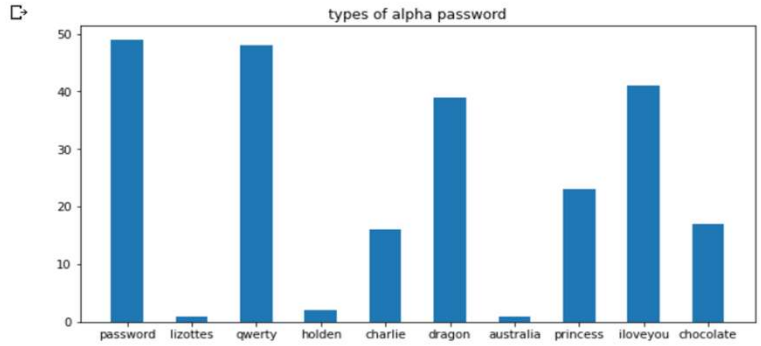
- Top 10 alpha passwords are displayed.
- The graph is plot alpha passwords against count
- And from this graph we can tell **password** and **qwerty** are most used passwords.

```
[ ] #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': (
{'password': 49, 'lizottes': 1, 'qwerty': 48, 'holden': 2, 'charlie': 16, 'dragon': 39, 'australia': 1, 'princess': 23, 'iloveyou': 41, 'chocolate': 17, 'so
[('password', 49), ('lizottes', 1), ('qwerty', 48), ('holden', 2), ('charlie', 16), ('dragon', 39), ('australia', 1), ('princess', 23), ('iloveyou', 41), ('
```

+ Code + Text

```
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()
```



0s completed at 13:07

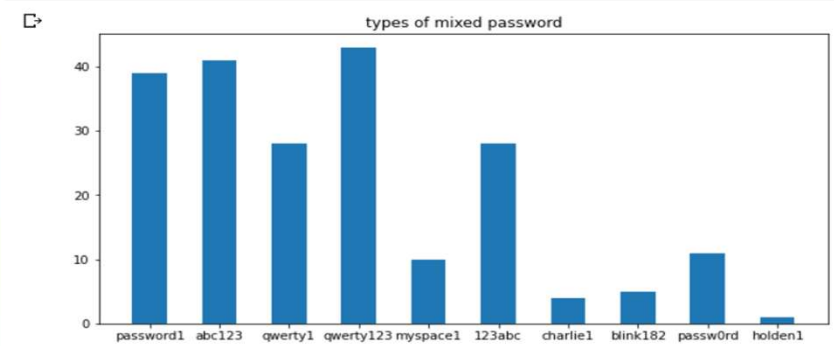
- Top 10 mixed passwords are displayed.
- The graph is plot mixed passwords against count
- And from this graph we can tell that password **qwerty123** is most used.

```
#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
{'password1': 39, 'abc123': 41, 'qwerty1': 28, 'qwerty123': 43, 'myspace1': 10, '123abc': 28, 'charlie1': 4, 'blink182': 5, 'passw0rd': 11, 'holden1': 1, 't
[('password1', 39), ('abc123', 41), ('qwerty1', 28), ('qwerty123', 43), ('myspace1', 10), ('123abc', 28), ('charlie1', 4), ('blink182', 5), ('passw0rd', 11)
```

```
[('password1', 39), ('abc123', 41), ('qwerty1', 28), ('qwerty123', 43), ('myspace1', 10), ('123a
```

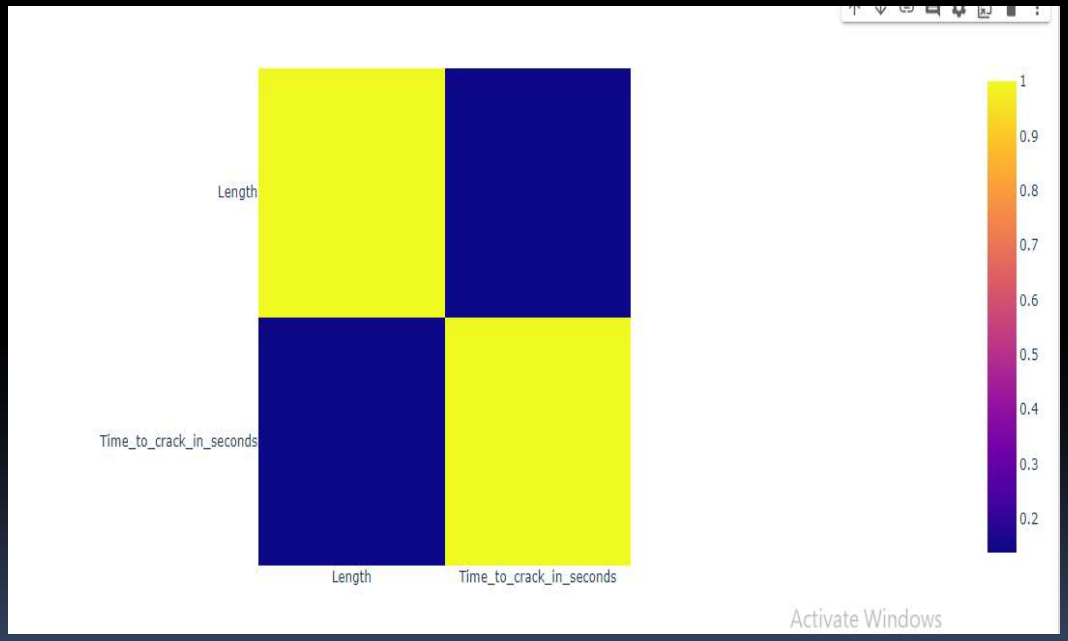
```
a=dict(ft)
mixed=a.keys()
count=a.values()
fig=plt.figure(figsize=(10,5))
plt.bar(mixed,count,width=0.5)
plt.xlabel=("password")
plt.ylabel=("count")
plt.title("types of mixed password")
plt.show()
```



0s completed at 13:07

- In this code the length of the passwords are displayed in data.
- Then we have find the correlation between length and time to crack and we have drawn a heat map.
- Based on map we can tell that between length and time to crack we have very small correlation

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

dp=df[['Length', 'Time_to_crack_in_seconds']].corr()
px.imshow(dp)
```



In this code we take user_count column.

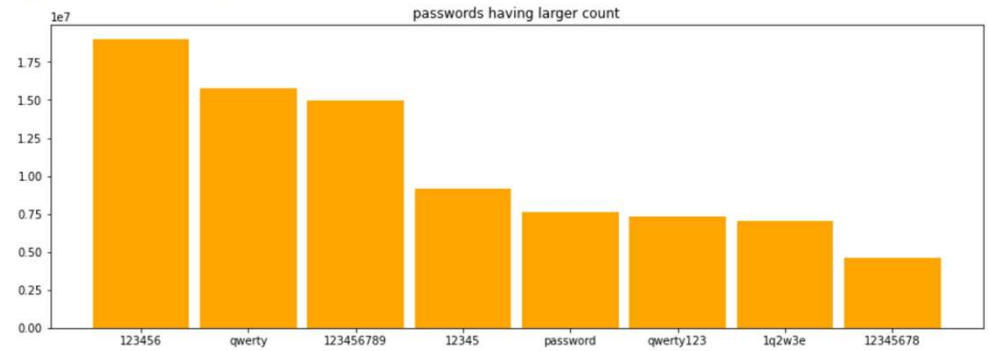
And we found out top 10 largest user count and their password.

```
#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()
```

+ Code + Text

RAM Disk

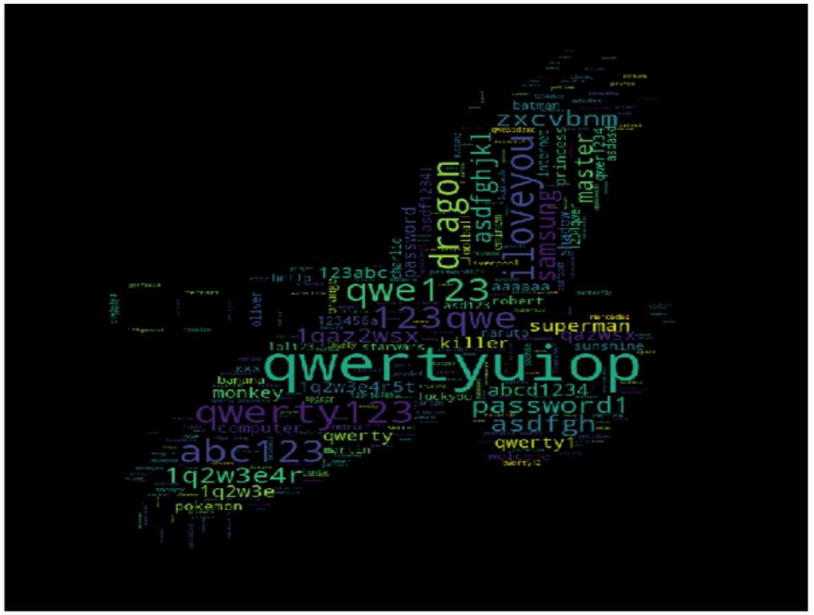
```
      Password  User_count
7000    123456    19000630
7001    qwerty    15738011
7002 123456789    14975791
7003     12345     9139679
1200    123456     8159358
7004  password     7593503
7005 qwerty123     7343013
7006    1q2w3e     7051194
7007  12345678     4632962
9400    123456     3572081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



- In this we have analyze the weakest passwords with the help of word cloud.
- Word cloud means it is a collection of words in different sizes. The bolder the word appears it is most important.
- In our case **qwertyuiop** is bolder so this is the weakest password.

```
#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()
```

+ Code + Text



4s completed at 16:11

- 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.

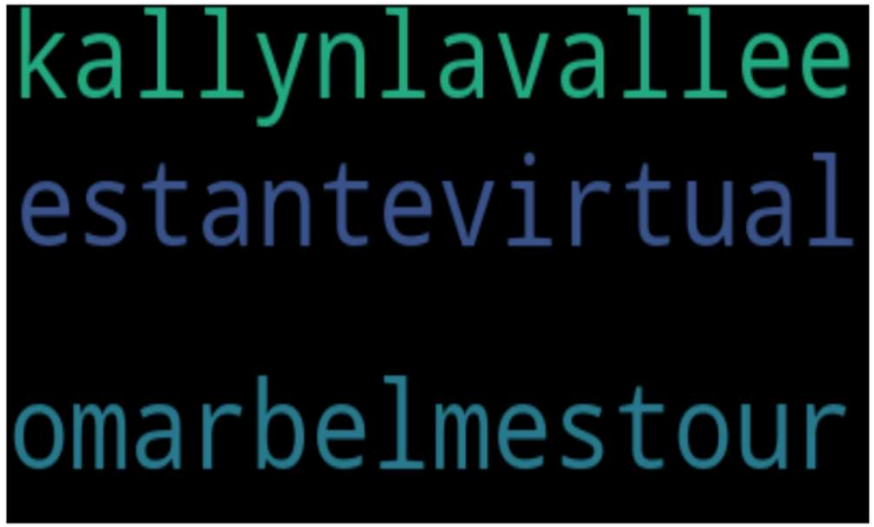
Code + Text

```
#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=(15,10))
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis("off")
plt.show()
```

+ Code + Text



Data Visualization & Interpretation

- 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) these paaword are mostly used rather than remaining password 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 kallynlavallee is the strongest password.

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.
- And finally we concluded that we have to use **alpha password or mixed password** because it is difficult to crack that type of password and its safe.
- For data processing we used data cleaning method.
- For analysis we used bar chart, horizontal bar chart, heat map and word cloud.