PROBLEM -PREMIUM VS FREEmium -SOLVED

August 6, 2022

0.0.1 Premium VS FREEmium

0.0.2 Find the total number of downloads for "paying" and "Non-paying" Users by date.

Include only Records where Non-paying Customers have more Downloads than the "Paying Customers". The Output should be sorted by earliest date first and should contain 3 Columns : "Date" , 'Non-Paying Downloads' and "Paying Downloads"

0.0.3 Dataframes : ms_user_dimension , ms_acc_dimension , ms_download_facts

DataFrame : **ms_user_dimension**

Coulmns :

- user_id
- acc_id

DataFrame : **ms_acc_dimension**

Coulmns :

- acc_id
- paying_customer

DataFrame : **ms_download_facts**

Coulmns :

- date
- user_id
- downloads

Logic :

- Its better to have all the information in a Single Dataframe
- We merge all the three Dataframes : ['ms_user_dimension' + 'ms_acc_dimension' + 'ms_download_facts']
- First we merge 'ms_user_dimension' + 'ms_acc_dimension' on **acc_id** = df_user
- Then we merge "df_user" with 'ms_download_facts' on **user_id**
- If we Filter out the Paying Customers ("yes") , the remaining would be Non-Paying ("No")

[14]:
```
import pandas as pd
import numpy as np
```

Merge DataFrames:

```
[4]: ms_user_dimension = pd.read_excel('ms_user_dimension.xlsx')
```

```
[5]: ms_acc_dimension = pd.read_excel('ms_acc_dimension.xlsx')
```

```
[17]: ms_download_facts = pd.read_excel('ms_download_facts.xlsx')
```

```
[79]: #To create this Dataset : simply copy the below output in a new excel
      #Paste special : text
      #Delete Column "A"
      #Save as :ms_user_dimension
```

```
[73]: ms_user_dimension.head(12)
```

```
[73]:     user_id  acc_id
      0         1     716
      1         2     749
      2         3     713
      3         4     744
      4         5     726
      5         6     713
      6         7     713
      7         8     744
      8         9     745
      9        10     788
      10       11     713
      11       12     744
```

```
[47]: ms_user_dimension.shape
```

```
[47]: (12, 2)
```

```
[80]: #To create this Dataset : simply copy the below output in a new excel
      #Paste special : text
      #Delete Column "A"
      #Save as :ms_acc_dimension
```

```
[75]: ms_acc_dimension.head(12)
```

```
[75]:     acc_id paying_customer
      0      716             yes
      1      749             yes
      2      713             yes
      3      744              no
      4      726             yes
      5      713              no
      6      713              no
      7      744              no
```

```
8      745             yes
9      788             yes
10     713              no
11     744              no
```

```
[48]: ms_acc_dimension.shape
```

```
[48]: (12, 2)
```

```
[82]: #To create this Dataset : simply copy the below output in a new excel
      #Paste special : text
      #Format Date column : YYYY-MM-DD
      #Delete Column "A"
      #Save as : ms_download_facts
```

```
[77]: ms_download_facts.head(12)
```

```
[77]:     user_id       date  downloads
      0         1 2020-08-24          6
      1         2 2020-08-20          5
      2         3 2020-08-21          6
      3         4 2020-08-25          2
      4         5 2020-08-26          4
      5         6 2020-08-28          1
      6         7 2020-08-16          2
      7         8 2020-08-15          3
      8         9 2020-08-11          4
      9        10 2020-08-31          4
      10       11 2020-08-12          5
      11       12 2020-08-29          2
```

```
[49]: ms_download_facts.shape
```

```
[49]: (12, 3)
```

```
[19]: #Merging ms_user_dimension and ms_acc_dimension on "acc_id"
```

```
[15]: df_user = ms_user_dimension.merge(ms_acc_dimension, on ='acc_id')
```

```
[16]: df_user.head(8)
```

```
[16]:    user_id  acc_id paying_customer
      0        1     716             yes
      1        2     749             yes
      2        3     713             yes
      3        3     713              no
      4        3     713              no
      5        3     713              no
```

```
6         6     713               yes
7         6     713                no
```

```
[50]: df_user.shape
```

```
[50]: (30, 3)
```

```
[51]: #Check for Duplicates

      df_user.duplicated().sum()
```

```
[51]: 14
```

```
[52]: df_user.drop_duplicates(inplace = True)
```

```
[20]: #Merging df_user and ms_download_facts on "user_id"
```

```
[53]: df_all = df_user.merge(ms_download_facts, on='user_id')
```

```
[54]: df_all.head(3)
```

```
[54]:    user_id  acc_id paying_customer       date  downloads
      0        1     716             yes 2020-08-24          6
      1        2     749             yes 2020-08-20          5
      2        3     713             yes 2020-08-21          6
```

```
[55]: df_all.shape
```

```
[55]: (16, 5)
```

```
[56]: #Check for Duplicates

      df_all.duplicated().sum()
```

```
[56]: 0
```

Observation - Now we have all three Data Frames merged . - We have all the Columns from all the dfs.

```
[57]: df_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 0 to 15
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   user_id          16 non-null     int64
 1   acc_id           16 non-null     int64
 2   paying_customer  16 non-null     object
 3   date              16 non-null     datetime64[ns]
 4   downloads         16 non-null     int64
dtypes: datetime64[ns](1), int64(3), object(1)
memory usage: 768.0+ bytes
```

[58]:
```
df_all.shape
```

[58]:
```
(16, 5)
```

[59]:
```
df_all.columns
```

[59]:
```
Index(['user_id', 'acc_id', 'paying_customer', 'date', 'downloads'],
dtype='object')
```

[60]:
```
df_all.describe()
```

[60]:
```
         user_id      acc_id  downloads
count  16.000000   16.000000   16.00000
mean    6.562500  728.750000    3.62500
std     3.424787   21.659486    1.78419
min     1.000000  713.000000    1.00000
25%     3.750000  713.000000    2.00000
50%     6.500000  714.500000    4.00000
75%     9.250000  744.000000    5.00000
max    12.000000  788.000000    6.00000
```

[35]:
```
#We can further filter out "downloads" as "Paying" and "Non-Paying" downloads
#If you look at the column "paying_customer" :it has got "yes" and "No"␣
↪categories
```

[61]:
```
df_all.head(30)
```

[61]:
```
    user_id  acc_id paying_customer       date  downloads
0         1     716             yes 2020-08-24          6
1         2     749             yes 2020-08-20          5
2         3     713             yes 2020-08-21          6
3         3     713              no 2020-08-21          6
4         6     713             yes 2020-08-28          1
5         6     713              no 2020-08-28          1
6         7     713             yes 2020-08-16          2
7         7     713              no 2020-08-16          2
8        11     713             yes 2020-08-12          5
9        11     713              no 2020-08-12          5
10        4     744              no 2020-08-25          2
11        8     744              no 2020-08-15          3
12       12     744              no 2020-08-29          2
13        5     726             yes 2020-08-26          4
14        9     745             yes 2020-08-11          4
```

```
15        10      788               yes  2020-08-31          4
```

LOGIC:

- Out of all the Non-Paying customers select only the ones whose Downloads are more than the Paying Customer Downloads.

[62]:
```
#Setting both of these columns to "downloads" column

df_all['paid'] = df_all['downloads']

df_all['unpaid'] = df_all['downloads']
```

[63]:
```
df_all.head(2)
```

[63]:
```
   user_id  acc_id paying_customer        date  downloads  paid  unpaid
0        1     716             yes 2020-08-24          6     6       6
1        2     749             yes 2020-08-20          5     5       5
```

[69]:
```
#check df_all.loc
#Extracting the Downloads of Payin_customer("yes") into "paid" column
##Extracting the Downloads of Payin_customer("no") into "unpaid" column
#If df_all["paying_customer"] is equal to "no" (means they are unpaid) then we␣
↪want to set the column "paid" equal to 0
#If df_all["paying_customer"] is equal to "yes" (means they are paid) then we␣
↪want to set the column "unpaid" equal to 0
```

[65]:
```
df_all.loc[df_all['paying_customer']=='no', 'paid'] = 0

df_all.loc[df_all['paying_customer']=='yes', 'unpaid'] = 0
```

[66]:
```
df_all.head(4)
```

[66]:
```
   user_id  acc_id paying_customer        date  downloads  paid  unpaid
0        1     716             yes 2020-08-24          6     6       0
1        2     749             yes 2020-08-20          5     5       0
2        3     713             yes 2020-08-21          6     6       0
3        3     713              no 2020-08-21          6     0       6
```

[67]:
```
daily_values = df_all.groupby('date').sum().reset_index()[['date' , 'paid' ,␣
↪'unpaid']]
```

[68]:
```
daily_values
```

[68]:
```
         date  paid  unpaid
0  2020-08-11     4       0
1  2020-08-12     5       5
2  2020-08-15     0       3
```

```
3   2020-08-16     2        2
4   2020-08-20     5        0
5   2020-08-21     6        6
6   2020-08-24     6        0
7   2020-08-25     0        2
8   2020-08-26     4        0
9   2020-08-28     1        1
10  2020-08-29     0        2
11  2020-08-31     4        0
```

[70]:
```
#Filter the Results where Paid is less than Unpaid

final_result = daily_values[daily_values['paid'] < daily_values['unpaid']]
```

FINAL SOLUTION :

[72]:
```
final_result
```

[72]:
```
          date  paid  unpaid
2   2020-08-15     0       3
7   2020-08-25     0       2
10  2020-08-29     0       2
```

Contributed by : Aisha Khalid