Monthly-Percentage-Difference - SOLVED

August 6, 2022

0.0.1 PROBLEM : [Monthly Percentage Difference]

0.0.2 Given a table of Purchases by Date , calculate the Month-Over-Month Percentage change in revenue.The output should include the Year-Month-Date(YYYY -MM) and percentage Change, rounded to the 2nd Decimal point, and Sorted from the beginning of the year to the end of the Year.

The percentage change column will be populated from the second month Forward and can be calculated as ((this month's revenue - last month's revenue) / last month's revenue) *100

Dataframe Name : sf_transactions

[612]:
```python
import pandas as pd
import numpy as np
import datetime
import calendar
```

[613]:
```python
#Just for practice , this is how you can get the current Date and Time
x = datetime.datetime.now()
x
```

[613]:
```
datetime.datetime(2022, 8, 6, 15, 52, 42, 310831)
```

[614]:
```python
x.month #Extracting Month from Datetime Object
```

[614]:
```
8
```

[615]:
```python
#Extracting the Year frm the DateTime Object

print(x.year)
```

```
2022
```

[616]:
```python
x.strftime('%Y-%m')
```

[616]:
```
'2022-08'
```

0.0.3 START HERE FOR THE SOLUTION :

[617]:
```
#Reading the DataFrame
sf_transactions = pd.read_excel('sf_transactions.xlsx')
```

[ ]:
```
'''Note: This is a dummy Dataframe. It has 4 columns . "Created_at" is the Date␣
 ↪Field. We can have Dates for Jan, Feb, March, April.
Since we are focussing on pulling out details for March. Make sure March Dates␣
 ↪are present in the Dataset.No need to create a very big
dataset. You can have 10 Rows for each month.'''
```

[618]:
```
sf_transactions.shape
```

[618]:
```
(52, 4)
```

[619]:
```
sf_transactions.head(10)
```

[619]:
```
   id created_at   value  purchase_id
0   1 2019-01-01   20786           43
1   2 2019-01-05   30786           32
2   3 2019-01-09   30009           66
3   4 2019-03-09   45000           67
4   5 2019-03-21   55000           48
5   6 2019-03-25   78000           31
6   7 2019-03-20   79000           43
7   8 2019-03-30   30000            4
8   9 2019-03-25   39000           34
9  10 2019-03-20  110000           45
```

[620]:
```
#you may notice the "created_at" column is a DateTime Field
sf_transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52 entries, 0 to 51
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           52 non-null     int64
 1   created_at   52 non-null     datetime64[ns]
 2   value        52 non-null     int64
 3   purchase_id  52 non-null     int64
dtypes: datetime64[ns](1), int64(3)
memory usage: 1.8 KB
```

[621]:
```
#Extracting Month and Year from sf_transactions['created_at'] into two separate␣
 ↪columns
#Creating the "YYYY-MM" column :date_formatted
```

```
sf_transactions['month'] = sf_transactions['created_at'].dt.month
sf_transactions['year'] = sf_transactions['created_at'].dt.year
sf_transactions['date_formatted'] = sf_transactions['created_at'].dt.
 ↪strftime('%Y-%m')
```

[622]:
```
sf_transactions.head(3)
```

[622]:
```
   id created_at  value  purchase_id  month  year date_formatted
0   1 2019-01-01  20786           43      1  2019        2019-01
1   2 2019-01-05  30786           32      1  2019        2019-01
2   3 2019-01-09  30009           66      1  2019        2019-01
```

Extracting the Month Name from "month" column, to a new column : Month_Name

[623]:
```
sf_transactions['Month_Name'] = sf_transactions['month'].apply(lambda x:␣
 ↪calendar.month_abbr[x])
```

[95]:
```
#March_df = sf_transactions.loc[sf_transactions['month'] == 3]
```

[624]:
```
sf_transactions.head(2)
```

[624]:
```
   id created_at  value  purchase_id  month  year date_formatted Month_Name
0   1 2019-01-01  20786           43      1  2019        2019-01        Jan
1   2 2019-01-05  30786           32      1  2019        2019-01        Jan
```

Re-ordering the Dataset (Optional)

[625]:
```
sf_transactions.columns
```

[625]:
```
Index(['id', 'created_at', 'value', 'purchase_id', 'month', 'year',
       'date_formatted', 'Month_Name'],
      dtype='object')
```

[626]:
```
#Here , we may skip the columns we do not want to include
sf_transactions = sf_transactions[['id', 'created_at', 'value',␣
 ↪'month','Month_Name', 'year',
       'date_formatted']]
```

[627]:
```
sf_transactions.head(2)
```

[627]:
```
   id created_at  value  month Month_Name  year date_formatted
0   1 2019-01-01  20786      1        Jan  2019        2019-01
1   2 2019-01-05  30786      1        Jan  2019        2019-01
```

Value Counts

[628]:
```
#Unique Months in the dataset:

sf_transactions['month'].value_counts()
```

```
[628]: 3    18
       7     8
       1     7
       8     6
       4     5
       6     4
       5     4
       Name: month, dtype: int64
```

Creating a separate Dataframe from the Output of value counts (Note : This is not a part of the Problem Solution)

- Just for Practice

```
[629]: Month_value_counts = sf_transactions.month.value_counts().rename_axis('MONTH').
        ↪reset_index(name='COUNTS')
       print (Month_value_counts)
```

```
   MONTH  COUNTS
0      3      18
1      7       8
2      1       7
3      8       6
4      4       5
5      6       4
6      5       4
```

Observation - For the Month of March , there are 18 Records or 18 Observations in total

0.0.4 Grouping the Data by Month and looking at the Totals

```
[630]: sf_transactions.groupby(['month']).sum()
```

```
[630]:         id     value   year
       month
       1        68    837484  14133
       3       257   5249659  36342
       4       140  44444440  10095
       5       130  35955520   8076
       6       146  39555552   8076
       7       340  79200000  16152
       8       297  59806662  12114
```

Observation - When we group by "Month" and Aggregate is SUM() , the only column useful is "Value" - Because , summing up Dates or YEAR makes no sense - The dataframe is already Sorted on Months

```
[136]: #We group by month and use the SUM() as an Aggregate for "Value"
```

```
[631]: monthly_revenue=sf_transactions.groupby(['month']).sum().
        ↪reset_index()[['value']]
```

```
[632]: monthly_revenue
```

```
[632]:       value
       0    837484
       1   5249659
       2  44444440
       3  35955520
       4  39555552
       5  79200000
       6  59806662
```

Observation - We only see the Value column - Its better we have the Month Column as well in the view

```
[633]: #We group by month and use SUM() as an Aggregate for "Value" and we want the␣
        ↪"Month" column also in the view:
```

```
[634]: monthly_revenue=sf_transactions.groupby(['month',␣
        ↪'Month_Name','date_formatted']).sum().
        ↪reset_index()[['month','Month_Name','date_formatted','value']]
```

```
[635]: monthly_revenue
```

```
[635]:    month Month_Name date_formatted     value
       0      1        Jan        2019-01    837484
       1      3        Mar        2019-03   5249659
       2      4        Apr        2019-04  44444440
       3      5        May        2019-05  35955520
       4      6        Jun        2019-06  39555552
       5      7        Jul        2019-07  79200000
       6      8        Aug        2019-08  59806662
```

0.0.5 To Calculate this Month's Revenue - Last Month's Revenue

- We have a function called diff()
- Make sure your dataset is sorted in the Ascending order of Months (groupby automatically does that)
- diff() picks the value in the 1st cell and subtracts it with the 2nd and gives the output in the 2nd cell itself
- So , this way , the first cell in the Result column will have NAN / blank

1 METHOD 1:

```
[636]: monthly_revenue['value_difference'] = monthly_revenue['value'].diff()
```

```
[637]: monthly_revenue
```

```
[637]:    month Month_Name date_formatted     value  value_difference
       0      1        Jan        2019-01    837484               NaN
       1      3        Mar        2019-03   5249659         4412175.0
       2      4        Apr        2019-04  44444440        39194781.0
       3      5        May        2019-05  35955520        -8488920.0
       4      6        Jun        2019-06  39555552         3600032.0
       5      7        Jul        2019-07  79200000        39644448.0
       6      8        Aug        2019-08  59806662       -19393338.0
```

The above difference can be taken by implementing other logics as well. - We create a duplicate column of Values and shift 1 step down and take resulting diff in a new column

1.0.1 Alternate way of getting the difference :Month's Revenue - Last Month's Revenue

2 METHOD 2:

```
[638]: monthly_revenue.value
```

```
[638]: 0      837484
       1     5249659
       2    44444440
       3    35955520
       4    39555552
       5    79200000
       6    59806662
       Name: value, dtype: int64
```

```
[640]: len(monthly_revenue.value)  #How many members
```

```
[640]: 7
```

```
[654]: #We are creating 2 lists mylist1 and mylist2
       #mylist1 will be a copy of the "value" column
       #mylist2 will also be a copy of the "value" column
       #But we are removing the first item from mylist2
       #And appending 0 at the end of mylist2 in order to ajdust the length of the list
       #This way both the lists will have equal number of elements
       #Now we can Subtract List 2  from List 1
```

```
[641]: mylist1 = monthly_revenue['value'].to_list()
```

```
[642]: mylist1

[642]: [837484, 5249659, 44444440, 35955520, 39555552, 79200000, 59806662]

[643]: mylist2 = monthly_revenue['value'].to_list()

[645]: mylist2

[645]: [837484, 5249659, 44444440, 35955520, 39555552, 79200000, 59806662]

[646]: mylist2.pop(0)

[646]: 837484

[647]: mylist2

[647]: [5249659, 44444440, 35955520, 39555552, 79200000, 59806662]

[648]: mylist2.append(0)

[651]: mylist2

[651]: [5249659, 44444440, 35955520, 39555552, 79200000, 59806662, 0]

[652]: len(mylist1)

[652]: 7

[653]: len(mylist2)

[653]: 7

[655]: Subtracted_list  = [a - b for a, b in zip(mylist2, mylist1)]

[656]: Subtracted_list

[656]: [4412175, 39194781, -8488920, 3600032, 39644448, -19393338, -59806662]

[657]: #This subtracted List will be the new column : "value_difference2"

[658]: Subtracted_list = [0] + Subtracted_list #Appending 0 as first element

[659]: Subtracted_list

[659]: [0, 4412175, 39194781, -8488920, 3600032, 39644448, -19393338, -59806662]

[582]: Subtracted_list[-1]

[582]: -59806662
```

```
[660]: #Popping out the last element
```

```
[664]: Subtracted_list.pop(-1)
```

```
[664]: -59806662
```

```
[427]: #Ignore the Below code
       #Its a While loop created to do the same job as above
```

```
[485]:     mylist1=[]
           mylist2=[]
           i=0
           j=0
           while i< len(monthly_revenue.value):
               x =monthly_revenue['value'].iloc[i]
               mylist1.append(x)
               i += 1
               while j< len(monthly_revenue.value):
                   y =monthly_revenue['value'].iloc[j]
                   mylist2.append(y)
                   j += 1
               #break
           #mylist1 = [0] + mylist1
```

We append this Subtracted_list as a New column to the Dataframe :monthly_revenue

```
[662]: monthly_revenue.columns
```

```
[662]: Index(['month', 'Month_Name', 'date_formatted', 'value', 'value_difference'],
       dtype='object')
```

```
[665]: monthly_revenue['value_difference2'] = Subtracted_list
```

```
[666]: monthly_revenue.head(10)
```

```
[666]:    month Month_Name date_formatted     value  value_difference  \
       0      1        Jan        2019-01    837484               NaN
       1      3        Mar        2019-03   5249659         4412175.0
       2      4        Apr        2019-04  44444440        39194781.0
       3      5        May        2019-05  35955520        -8488920.0
       4      6        Jun        2019-06  39555552         3600032.0
       5      7        Jul        2019-07  79200000        39644448.0
       6      8        Aug        2019-08  59806662       -19393338.0

          value_difference2
       0                  0
       1            4412175
       2           39194781
```

```
3         -8488920
4          3600032
5         39644448
6        -19393338
```

Observation - You may notice we followed 2 different techniques to arrive at the "Month's Revenue - Last Month's Revenue" - Also notice that the Columns "value_difference" and "value_difference2" values are the same - We can use any of the two columns for further calculations

[325]:
```
#Note: We can use any of the two columns for further caculations␣
↪"value_difference" or "value_difference2"
```

2.0.1 Finally We want to get the Percent Change:

[326]:
```
#For this we create another column called "Last_month_revenue"
```

[587]:
```
monthly_revenue['Last_month_revenue'] = monthly_revenue['value'] -␣
↪monthly_revenue['value_difference']
```

[588]:
```
monthly_revenue.head(5)
```

[588]:
```
   month Month_Name date_formatted     value  value_difference  \
0      1        Jan        2019-01    837484               NaN
1      3        Mar        2019-03   5249659         4412175.0
2      4        Apr        2019-04  44444440        39194781.0
3      5        May        2019-05  35955520        -8488920.0
4      6        Jun        2019-06  39555552         3600032.0

   value_difference2  Last_month_revenue
0                  0                 NaN
1            4412175            837484.0
2           39194781           5249659.0
3           -8488920          44444440.0
4            3600032          35955520.0
```

[589]:
```
monthly_revenue['Pcnt_Change']= (monthly_revenue['value_difference']/␣
↪monthly_revenue['Last_month_revenue'])*100
```

[590]:
```
monthly_revenue.head(5)
```

[590]:
```
   month Month_Name date_formatted     value  value_difference  \
0      1        Jan        2019-01    837484               NaN
1      3        Mar        2019-03   5249659         4412175.0
2      4        Apr        2019-04  44444440        39194781.0
3      5        May        2019-05  35955520        -8488920.0
4      6        Jun        2019-06  39555552         3600032.0

   value_difference2  Last_month_revenue  Pcnt_Change
```

```
0              0              NaN          NaN
1        4412175         837484.0   526.836931
2       39194781        5249659.0   746.615752
3       -8488920       44444440.0   -19.100072
4        3600032       35955520.0    10.012460
```

[591]:
```
#Rounding the "Pcnt_Change" to 2 Decimal places

monthly_revenue['Pcnt_Change'] = monthly_revenue['Pcnt_Change'].round(2)
```

[592]:
```
monthly_revenue.head(5)
```

[592]:
```
   month Month_Name date_formatted     value  value_difference  \
0      1        Jan        2019-01    837484               NaN
1      3        Mar        2019-03   5249659         4412175.0
2      4        Apr        2019-04  44444440        39194781.0
3      5        May        2019-05  35955520        -8488920.0
4      6        Jun        2019-06  39555552         3600032.0

   value_difference2  Last_month_revenue  Pcnt_Change
0                  0                 NaN          NaN
1            4412175            837484.0       526.84
2           39194781           5249659.0       746.62
3           -8488920          44444440.0       -19.10
4            3600032          35955520.0        10.01
```

[593]:
```
monthly_revenue[['date_formatted' , 'Pcnt_Change']]
```

[593]:
```
  date_formatted  Pcnt_Change
0        2019-01          NaN
1        2019-03       526.84
2        2019-04       746.62
3        2019-05       -19.10
4        2019-06        10.01
5        2019-07       100.22
6        2019-08       -24.49
```

Conclusion : As required , the Output includes the year - Month Date (YYYY - MM) and Percentage Change rounded to 2nd Decimal point and Sorted from the beginning of the year to the end of the year.

Happy Learning

Contributed by Aisha Khalid

2.0.2 DATASET: Create your own

Below is the Data set : Just copy and paste this to an Excel - Format date “created_at” (YYYY-MM-DD) - Save as “sf_transactions”

[ ]:
```
id created_at        value        purchase_id
1        2019-01-01        20786        43
2        2019-01-05        30786        32
3        2019-01-09        30009        66
4        2019-03-09        45000        67
5        2019-03-21        55000        48
6        2019-03-25        78000        31
7        2019-03-20        79000        43
8        2019-03-30        30000        4
9        2019-03-25        39000        34
10        2019-03-20        110000        45
11        2019-03-30        110000        77
12        2019-01-01        20786        1
13        2019-01-05        30786        7
14        2019-01-09        30009        8
15        2019-03-09        555555        9
16        2019-03-21        674322        10
17        2019-03-25        590011        13
18        2019-03-20        172692        15
19        2019-03-30        183425        17
20        2019-03-25        543211        19
21        2019-03-20        540000        21
22        2019-03-30        555555        23
23        2019-01-09        674322        25
24        2019-03-09        444444        27
25        2019-03-21        444444        29
26        2019-04-09        8888888        11
27        2019-04-09        8888888        90
28        2019-04-09        8888888        91
29        2019-04-09        8888888        92
30        2019-04-09        8888888        93
31        2019-05-09        8988880        94
32        2019-05-09        8988880        95
33        2019-05-09        8988880        96
34        2019-05-09        8988880        97
35        2019-06-09        9888888        98
36        2019-06-09        9888888        99
37        2019-06-09        9888888        100
38        2019-06-09        9888888        101
39        2019-07-09        9900000        102
40        2019-07-09        9900000        103
41        2019-07-09        9900000        104
42        2019-07-09        9900000        105
```

```
43        2019-07-09        9900000        106
44        2019-07-09        9900000        107
45        2019-07-09        9900000        108
46        2019-07-09        9900000        109
47        2019-08-09        9967777        110
48        2019-08-09        9967777        111
49        2019-08-09        9967777        112
50        2019-08-09        9967777        113
51        2019-08-09        9967777        114
52        2019-08-09        9967777        115
```

Thank you!