

```
1 import pandas as pd
2 import json
3 import glob
4 import mysql.connector
5 from sqlalchemy import create_engine
6
7 #AGGREGATED TRANSACTION
8
9
10 #specify the folder path using '*' pattern
11 path_1_a =
12 'C:\\pulse\\data\\aggregated\\transaction\\country\\india\\state\\*\\*\\*.json'
13 files_1_a = glob.glob(path_1_a)
14 # declaring a empty list
15 dfs_1_a = []
16 #itering over the each file opened from the directory
17 for file in files_1_a:
18     parts = file.split("\\")
19     state_name = parts[8]
20     year = int(parts[9][:4])
21     quarter_number = int(parts[10][0])
22     quarter = "Q" + str(quarter_number)
23     # opening the json file one by one into python dictionary
24     with open(file) as f:
25         data = json.load(f)
26     # declaring a empty list to capture only the required data for analysing
27     transactions_1_a = []
28     for transaction in data['data']['transactionData']:
29         name = transaction['name']
30         payment_instrument = transaction['paymentInstruments'][0]
31         count = int(payment_instrument['count'])
32         amount = int(payment_instrument['amount'])
33         transactions_1_a.append({'country': 'India', 'state': state_name, 'year':
34 year, 'quarter': quarter, 'transaction_type': name, 'transaction_count': count,
35 'total_amount': amount})
36     # converting the list of dictionaries to dataframe
37     df_1_a = pd.DataFrame(transactions_1_a)
38     # adding converted data frame into new list
39     dfs_1_a.append(df_1_a)
40 #concatenate
41 df_agg_tran = pd.concat(dfs_1_a)
42 #index reset
43 df_agg_tran = df_agg_tran.reset_index(drop=True)
44 df_agg_tran.to_csv('C:\\Users\\Shruthy\\Downloads\\agg_trans.csv', index=False)
45 #print(df_agg_tran)
46
47
48 #AGGREGATED USER
49
50 #specify the folder path using '*' pattern
51 path_1_u = 'C:\\pulse\\data\\aggregated\\user\\country\\india\\state\\*\\*\\*.json'
52 files_1_u = glob.glob(path_1_u)
53 # declaring a empty list
54 dfs_1_u= []
55 #itering over the each file opened from the directory
56 for file in files_1_u:
57     parts = file.split("\\")
58     state_name = parts[8]
```

```

57     year = int(parts[9][:4])
58     quarter_number = int(parts[10][0])
59     quarter = "Q" + str(quarter_number)
60     # opening the json file one by one into python dictionary
61     with open(file) as f:
62         data = json.load(f)
63     # declaring a empty list to capture only the required data for analysing
64     transactions_1_u = []
65     # iterating through the first file and extracting values of name, type, count, and
amount of the payment instrument, and stores them in a new dictionary.
66     for user in data['data']['aggregated']:
67         registered = int(data['data']['aggregated']['registeredUsers'])
68         app = int(data['data']['aggregated']['appOpens'])
69         transactions_1_u.append({'country': 'india', 'state': state_name, 'year':
year, 'quarter': quarter, 'registered_users': registered, 'apps_opened': app})
70     # converting the list of dictionaries to dataframe
71     df_1_u = pd.DataFrame(transactions_1_u)
72     # adding converted data frame into new list
73     dfs_1_u.append(df_1_u)
74 # concatenate all the list having dataframe of each file in the directory
75 df_agg_user = pd.concat(dfs_1_u)
76 # resetting the index
77 df_agg_user = df_agg_user.reset_index(drop=True)
78 df_agg_user.to_csv('C:\\Users\\Shruthy\\Downloads\\agg_users.csv', index=False)
79 #print(df_agg_user)
80
81
82 #MAP TRANSACTIONS
83
84 #specify the folder path using '*' pattern
85 path_2_a =
'C:\\pulse\\data\\map\\transaction\\hover\\country\\india\\state\\*\\*\\*.json'
86 files_2_a = glob.glob(path_2_a)
87 dfs_2_a = []
88
89 for file in files_2_a:
90     parts = file.split("\\")
91     state_name = parts[9]
92     year = int(parts[10][:4])
93     quarter_number = int(parts[11][0])
94     quarter = "Q" + str(quarter_number)
95
96     with open(file) as f:
97         data = json.load(f)
98
99     hoverdatas_2_a = []
100    for hoverdata in data['data']['hoverDataList']:
101        name = hoverdata['name']
102        metrics = hoverdata['metric'][0]
103        count = int(metrics['count'])
104        amount = int(metrics['amount'])
105        #A list of dictionaries is created, where each dictionary corresponds to
one transaction.
106        hoverdatas_2_a.append({'country': 'india', 'state': state_name, 'year':
year, 'quarter': quarter, 'district_name': name, 'transaction_count': count,
'total_amount': amount})
107    # converting the list of dictionaries to dataframe
108    df_2_a = pd.DataFrame(hoverdatas_2_a)
109    dfs_2_a.append(df_2_a)
110

```

```

111 df_map_tran = pd.concat(dfs_2_a)
112 df_map_tran = df_map_tran.reset_index(drop=True)
113 df_map_tran.to_csv('C:\\Users\\Shruthy\\Downloads\\_map_trans.csv', index=False)
114 #print(df_map_tran)
115
116
117 #MAP USER
118
119
120 #specify the folder path using '*' pattern
121 path_2_u = 'C:\\pulse\\data\\map\\user\\hover\\country\\india\\state\\*\\*\\*.json'
122 files_2_u = glob.glob(path_2_u)
123 # declaring a empty list
124 dfs_2_u= []
125 #itering over the each file opened from the directory
126 for file in files_2_u:
127     parts = file.split("\\")
128     state_name = parts[9]
129     year = int(parts[10][:4])
130     quarter_number = int(parts[11][0])
131     quarter = "Q" + str(quarter_number)
132     # opening the json file one by one into python dictionary
133     with open(file) as f:
134         data = json.load(f)
135         # declaring a empty list to capture only the required data for analysing
136         transactions_2_u = []
137         # iterating through the first file and extracting values of name, type, count, and
138         # amount of the payment instrument, and stores them in a new dictionary.
139         for states, users in data['data']['hoverData'].items():
140             state = states
141             registered = int(users['registeredUsers'])
142             app = int(users['appOpens'])
143             transactions_2_u.append({'country': 'india', 'state': state_name, 'year':
144 year, 'quarter': quarter, 'states': state, 'registered_users': registered, 'apps_opened':
145 app})
146
147 # converting the list of dictionaries to dataframe
148 df_2_u = pd.DataFrame(transactions_2_u)
149 # adding converted data frame into new list
150 dfs_2_u.append(df_2_u)
151 # concatenate all the list having dataframe of each file in the directory
152 df_map_user = pd.concat(dfs_2_u)
153 # resetting the index
154 df_map_user = df_map_user.reset_index(drop=True)
155 df_map_user.to_csv('C:\\Users\\Shruthy\\Downloads\\map_users.csv', index=False)
156 #print(df_map_user)
157
158
159 #TOP TRANSACTIONS
160
161 #specify the folder path using '*' pattern
162 path_3_a = 'C:\\pulse\\data\\top\\transaction\\country\\india\\state\\*\\*\\*.json'
163 files_3_a = glob.glob(path_3_a)
164 dfs_3_a=[]
165 for file in files_3_a:
166     parts = file.split("\\")
167     state_name = parts[8]
168     year = int(parts[9][:4])
169     quarter_number = int(parts[10][0])
170     quarter = "Q" + str(quarter_number)
171
172     with open(file) as f:

```

```

168         data = json.load(f)
169
170     entity_list_3_a= []
171
172     for dist in data['data']['districts']:
173         name = dist['entityName']
174         metrics = dist['metric']
175         count = int(metrics['count'])
176         amount = int(metrics['amount'])
177         #A list of dictionaries is created, where each dictionary
corresponds to one transaction.
178         entity_list_3_a .append({'country': 'india', 'state':
state_name, 'year':
year, 'entity_type': 'district', 'quarter': quarter, 'district&pincode': name,
'transaction_count': count, 'total_amount': amount})
179     for pin in data['data']['pincodes']:
180         name = pin['entityName']
181         metrics = pin['metric']
182         count = int(metrics['count'])
183         amount = int(metrics['amount'])
184         #A list of dictionaries is created, where each dictionary
corresponds to one transaction.
185         entity_list_3_a .append({'country': 'india', 'state':
state_name, 'year':
year, 'entity_type': 'pincode', 'quarter': quarter, 'district&pincode': name,
'transaction_count': count, 'total_amount': amount,})
186     # converting the list of dictionaries to dataframe
187     #
188     df_3_a = pd.DataFrame(entity_list_3_a)
189     dfs_3_a.append(df_3_a)
190
191 df_top_agg = pd.concat(dfs_3_a)
192 df_top_agg = df_top_agg .reset_index(drop=True)
193 df_top_agg.to_csv('C:\\Users\\Shruthy\\Downloads\\top_trans.csv', index=False)
194 #print(df_top_agg)
195
196 #TOP USER
197
198 #specify the folder path using '*' pattern
199 path_3_u = 'C:\\pulse\\data\\top\\user\\country\\india\\state\\*\\*\\*.json'
200 files_3_u = glob.glob(path_3_u)
201 # declaring a empty list
202 dfs_3_u = []
203 #itering over the each file opened from the directory
204 for file in files_3_u:
205     parts = file.split("\\")
206     state_name = parts[8]
207     year = int(parts[9][:4])
208     quarter_number = int(parts[10][0])
209     quarter = "Q" + str(quarter_number)
210     # opening the json file one by one into python dictionary
211     with open(file) as f:
212         data = json.load(f)
213         # declaring a empty list to capture only the required data for analysing
214         transactions_3_u = []
215         # iterating through the first file and extracting values of name, type, count, and
amount of the payment instrument, and stores them in a new dictionary.
216         # for states in data['data']['states']:
217         #     state = states['name']
218         #     registered = int(states['registeredUsers'])

```

```
219     # transactions.append({'state_district_pin':state,'registered_users':
registered,'entity_type':'state'})
220     for dist in data['data']['districts']:
221         district = dist['name']
222         registered = int(dist['registeredUsers'])
223         transactions_3_u.append({'country': 'india','state': state_name,'year':
year,'quarter':quarter,'entity_type':'district','district&pin':district,'registered_
users': registered})
224     for pin in data['data']['pincodes']:
225         pincode = pin['name']
226         registered = int(pin['registeredUsers'])
227         transactions_3_u.append({'country': 'india','state': state_name,'year':
year,'quarter':quarter,'entity_type':'pincode','district&pin':pincode,'registered_us
ers': registered,})
228     # converting the list of dictionaries to dataframe
229     df_3_u = pd.DataFrame(transactions_3_u)
230     # adding converted data frame into new list
231     dfs_3_u.append(df_3_u)
232 # concatenate all the list having dataframe of each file in the directory
233 df_top_user = pd.concat(dfs_3_u)
234 # resetting the index
235 df_top_user = df_top_user.reset_index(drop=True)
236 df_top_user.to_csv('C:\\Users\\Shruthy\\Downloads\\top_users.csv',index=False)
237 #print(df_top_user)
238
239
240
241 # using create_engine module opening the MySQL with correct credentials
242 engine =
create_engine('mysql+mysqlconnector://root:Shruthy#123@127.0.0.1:3306/pulse')
243
244 config = {
245     'user': 'root',
246     'password': 'Shruthy#123',
247     'host': '127.0.0.1',
248     'database': 'pulse',
249     'raise_on_warnings': True
250 }
251
252 # Connect to the database
253 cnx = mysql.connector.connect(**config)
254
255 # Check if the connection is successful
256 if cnx.is_connected():
257     print("Connection to MySQL database established.")
258 else:
259     print("Connection to MySQL database failed.")
260
261
262 # create a table name and store the dataframe-1
263 df_agg_tran.to_sql(name='aggregate_transaction', con=engine, if_exists='replace',
index=False)
264 # create a table name and store the dataframe-2
265 df_agg_user.to_sql(name='aggregate_users', con=engine, if_exists='replace',
index=False)
266 # create a table name and store the dataframe-3
267 df_map_tran.to_sql(name='map_transcation', con=engine,
if_exists='replace',index=False)
268 # create a table name and store the dataframe-4
269 df_map_user.to_sql(name='map_users', con=engine, if_exists='replace', index=False)
```

```
270 # create a table name and store the dataframe-5
271 df_top_agg.to_sql(name='top_transaction', con=engine, if_exists='replace',
index=False)
272 #create a table name and store the dataframe-6
273 df_top_user.to_sql(name='top_users', con=engine, if_exists='replace', index=False)
274
275
276
277 # Create a cursor to execute SQL queries
278 cursor = cnx.cursor()
279
280 # Define the SQL query to retrieve data from the "table
281 query = "SELECT * FROM aggregate_transaction"
282
283 # Execute the SQL query and store the result in a Pandas dataframe
284 aggregate_transaction = pd.read_sql(query, cnx)
285
286 # Print the first 5 rows of the dataframe
287 print(aggregate_transaction.head())
288
289 aggregate_transaction['state'] = aggregate_transaction['state'].replace({'andaman-&-
nicobar-islands': 'Andaman & Nicobar Island', 'andhra-pradesh': 'Andhra Pradesh',
'arunachal-pradesh': 'Arunachal Pradesh',
290     'assam': 'Assam', 'bihar': 'Bihar', 'chandigarh': 'Chandigarh',
'chhattisgarh': 'Chhattisgarh',
291     'dadra-&-nagar-haveli-&-daman-&-diu': 'Dadra and Nagar Haveli and Daman and
Diu', 'delhi': 'Delhi', 'goa': 'Goa', 'gujarat': 'Gujarat',
292     'haryana': 'Haryana', 'himachal-pradesh': 'Himachal Pradesh', 'jammu-&-
kashmir': 'Jammu & Kashmir', 'jharkhand': 'Jharkhand',
293     'karnataka': 'Karnataka', 'kerala': 'Kerala', 'ladakh': 'Ladakh',
'lakshadweep': 'Lakshadweep', 'madhya-pradesh': 'Madhya Pradesh',
294     'maharashtra': 'Maharashtra', 'manipur': 'Manipur', 'meghalaya': 'Meghalaya',
'mizoram': 'Mizoram', 'nagaland': 'Nagaland',
295     'odisha': 'Odisha', 'puducherry': 'Puducherry', 'punjab': 'Punjab',
'rajasthan': 'Rajasthan', 'sikkim': 'Sikkim',
296     'tamil-nadu': 'Tamil Nadu', 'telangana': 'Telangana', 'tripura': 'Tripura',
'uttar-pradesh': 'Uttar Pradesh',
297     'uttarakhand': 'Uttarakhand', 'west-bengal': 'West Bengal'})
298
299 # SQL query to retrieve data from the "table
300 query = "SELECT * FROM aggregate_users"
301
302 # executing the SQL query and store the result in a Pandas dataframe
303 aggregate_users = pd.read_sql(query, cnx)
304
305 # Print the first 5 rows of the dataframe
306 print(aggregate_users.head())
307
308 aggregate_users['state'] = aggregate_users['state'].replace({'andaman-&-nicobar-
islands': 'Andaman & Nicobar Island', 'andhra-pradesh': 'Andhra Pradesh', 'arunachal-
pradesh': 'Arunachal Pradesh',
309     'assam': 'Assam', 'bihar': 'Bihar', 'chandigarh': 'Chandigarh',
'chhattisgarh': 'Chhattisgarh',
310     'dadra-&-nagar-haveli-&-daman-&-diu': 'Dadra and Nagar Haveli and Daman and
Diu', 'delhi': 'Delhi', 'goa': 'Goa', 'gujarat': 'Gujarat',
311     'haryana': 'Haryana', 'himachal-pradesh': 'Himachal Pradesh', 'jammu-&-
kashmir': 'Jammu & Kashmir', 'jharkhand': 'Jharkhand',
312     'karnataka': 'Karnataka', 'kerala': 'Kerala', 'ladakh': 'Ladakh',
'lakshadweep': 'Lakshadweep', 'madhya-pradesh': 'Madhya Pradesh',
```



```
313     'maharashtra':'Maharashtra', 'manipur':'Manipur', 'meghalaya':'Meghalaya',
    'mizoram':'Mizoram', 'nagaland':'Nagaland',
314     'odisha':'Odisha', 'puducherry':'Puducherry', 'punjab':'Punjab',
    'rajasthan':'Rajasthan', 'sikkim':'Sikkim',
315     'tamil-nadu': 'Tamil Nadu', 'telangana':'Telangana', 'tripura':'Tripura',
    'uttar-pradesh':'Uttar Pradesh',
316     'uttarakhand':'Uttarakhand', 'west-bengal':'West Bengal'})
317
318 # SQL query to retrieve data from the "table"
319 query = "SELECT * FROM map_transaction"
320
321 # executing the SQL query and store the result in a Pandas dataframe
322 map_transaction = pd.read_sql(query, cnx)
323
324 # Print the first 5 rows of the dataframe
325 print(map_transaction.head())
326
327 map_transaction['state'] = map_transaction['state'].replace({'andaman-&-nicobar-
    islands': 'Andaman & Nicobar Island', 'andhra-pradesh':'Andhra Pradesh', 'arunachal-
    pradesh':'Arunachal Pradesh',
328     'assam':'Assam', 'bihar':'Bihar', 'chandigarh':'Chandigarh',
    'chhattisgarh':'Chhattisgarh',
329     'dadra-&-nagar-haveli-&-daman-&-diu':'Dadra and Nagar Haveli and Daman and
    Diu', 'delhi': 'Delhi', 'goa':'Goa', 'gujarat': 'Gujarat',
330     'haryana':'Haryana', 'himachal-pradesh':'Himachal Pradesh', 'jammu-&-
    kashmir':'Jammu & Kashmir', 'jharkhand':'Jharkhand',
331     'karnataka':'Karnataka', 'kerala':'Kerala', 'ladakh':'Ladakh',
    'lakshadweep':'Lakshadweep', 'madhya-pradesh':'Madhya Pradesh',
332     'maharashtra':'Maharashtra', 'manipur':'Manipur', 'meghalaya':'Meghalaya',
    'mizoram':'Mizoram', 'nagaland':'Nagaland',
333     'odisha':'Odisha', 'puducherry':'Puducherry', 'punjab':'Punjab',
    'rajasthan':'Rajasthan', 'sikkim':'Sikkim',
334     'tamil-nadu': 'Tamil Nadu', 'telangana':'Telangana', 'tripura':'Tripura',
    'uttar-pradesh':'Uttar Pradesh',
335     'uttarakhand':'Uttarakhand', 'west-bengal':'West Bengal'})
336
337 # SQL query to retrieve data from the "table"
338 query = "SELECT * FROM map_users"
339
340 # executing the SQL query and store the result in a Pandas dataframe
341 map_users = pd.read_sql(query, cnx)
342
343 # Print the first 5 rows of the dataframe
344 print(map_users.head())
345
346 map_users['state'] = map_users['state'].replace({'andaman-&-nicobar-islands':
    'Andaman & Nicobar Island', 'andhra-pradesh':'Andhra Pradesh', 'arunachal-
    pradesh':'Arunachal Pradesh',
347     'assam':'Assam', 'bihar':'Bihar', 'chandigarh':'Chandigarh',
    'chhattisgarh':'Chhattisgarh',
348     'dadra-&-nagar-haveli-&-daman-&-diu':'Dadra and Nagar Haveli and Daman and
    Diu', 'delhi': 'Delhi', 'goa':'Goa', 'gujarat': 'Gujarat',
349     'haryana':'Haryana', 'himachal-pradesh':'Himachal Pradesh', 'jammu-&-
    kashmir':'Jammu & Kashmir', 'jharkhand':'Jharkhand',
350     'karnataka':'Karnataka', 'kerala':'Kerala', 'ladakh':'Ladakh',
    'lakshadweep':'Lakshadweep', 'madhya-pradesh':'Madhya Pradesh',
351     'maharashtra':'Maharashtra', 'manipur':'Manipur', 'meghalaya':'Meghalaya',
    'mizoram':'Mizoram', 'nagaland':'Nagaland',
352     'odisha':'Odisha', 'puducherry':'Puducherry', 'punjab':'Punjab',
    'rajasthan':'Rajasthan', 'sikkim':'Sikkim',
```

```
353         'tamil-nadu': 'Tamil Nadu', 'telangana': 'Telangana', 'tripura': 'Tripura',
354         'uttar-pradesh': 'Uttar Pradesh',
355         'uttarakhand': 'Uttarakhand', 'west-bengal': 'West Bengal'})
356 # SQL query to retrieve data from the "table"
357 query = "SELECT * FROM top_transaction"
358
359 # executing the SQL query and store the result in a Pandas dataframe
360 top_transaction = pd.read_sql(query, cnx)
361
362 # Print the first 5 rows of the dataframe
363 print(top_transaction.head())
364
365 top_transaction['state'] = top_transaction['state'].replace({'andaman-&-nicobar-
islands': 'Andaman & Nicobar Island', 'andhra-pradesh': 'Andhra Pradesh', 'arunachal-
pradesh': 'Arunachal Pradesh',
366         'assam': 'Assam', 'bihar': 'Bihar', 'chandigarh': 'Chandigarh',
367         'chhattisgarh': 'Chhattisgarh',
368         'dadra-&-nagar-haveli-&-daman-&-diu': 'Dadra and Nagar Haveli and Daman and
Diu', 'delhi': 'Delhi', 'goa': 'Goa', 'gujarat': 'Gujarat',
369         'haryana': 'Haryana', 'himachal-pradesh': 'Himachal Pradesh', 'jammu-&-
kashmir': 'Jammu & Kashmir', 'jharkhand': 'Jharkhand',
370         'karnataka': 'Karnataka', 'kerala': 'Kerala', 'ladakh': 'Ladakh',
371         'lakshadweep': 'Lakshadweep', 'madhya-pradesh': 'Madhya Pradesh',
372         'maharashtra': 'Maharashtra', 'manipur': 'Manipur', 'meghalaya': 'Meghalaya',
373         'mizoram': 'Mizoram', 'nagaland': 'Nagaland',
374         'odisha': 'Odisha', 'puducherry': 'Puducherry', 'punjab': 'Punjab',
375         'rajasthan': 'Rajasthan', 'sikkim': 'Sikkim',
376         'tamil-nadu': 'Tamil Nadu', 'telangana': 'Telangana', 'tripura': 'Tripura',
377         'uttar-pradesh': 'Uttar Pradesh',
378         'uttarakhand': 'Uttarakhand', 'west-bengal': 'West Bengal'})
379
380 # SQL query to retrieve data from the "table"
381 query = "SELECT * FROM top_users"
382
383 # executing the SQL query and store the result in a Pandas dataframe
384 top_users = pd.read_sql(query, cnx)
385
386 # Print the first 5 rows of the dataframe
387 print(top_users.head())
388
389 top_users['state'] = top_users['state'].replace({'andaman-&-nicobar-islands':
'Andaman & Nicobar Island', 'andhra-pradesh': 'Andhra Pradesh', 'arunachal-
pradesh': 'Arunachal Pradesh',
390         'assam': 'Assam', 'bihar': 'Bihar', 'chandigarh': 'Chandigarh',
391         'chhattisgarh': 'Chhattisgarh',
392         'dadra-&-nagar-haveli-&-daman-&-diu': 'Dadra and Nagar Haveli and Daman and
Diu', 'delhi': 'Delhi', 'goa': 'Goa', 'gujarat': 'Gujarat',
393         'haryana': 'Haryana', 'himachal-pradesh': 'Himachal Pradesh', 'jammu-&-
kashmir': 'Jammu & Kashmir', 'jharkhand': 'Jharkhand',
394         'karnataka': 'Karnataka', 'kerala': 'Kerala', 'ladakh': 'Ladakh',
395         'lakshadweep': 'Lakshadweep', 'madhya-pradesh': 'Madhya Pradesh',
396         'maharashtra': 'Maharashtra', 'manipur': 'Manipur', 'meghalaya': 'Meghalaya',
397         'mizoram': 'Mizoram', 'nagaland': 'Nagaland',
398         'odisha': 'Odisha', 'puducherry': 'Puducherry', 'punjab': 'Punjab',
399         'rajasthan': 'Rajasthan', 'sikkim': 'Sikkim',
400         'tamil-nadu': 'Tamil Nadu', 'telangana': 'Telangana', 'tripura': 'Tripura',
401         'uttar-pradesh': 'Uttar Pradesh',
402         'uttarakhand': 'Uttarakhand', 'west-bengal': 'West Bengal'})
```



```
394
395
396
397
398
399 # fig1 = px.bar(data_df,x='Year_Quarter',y=option, hover_data=
    ['Transaction_Amounts','Transaction_counts'], color=option,
        labels={'Transactions amounts and counts-Yearwise'})
400 #     st.plotly_chart(fig1)
401 #     fig = px.line(data_df,x='Year_Quarter', y=option)#hover_data=
    ['Transaction_Amounts','Transaction_counts'],
    #color='Transaction_counts')
402 #     st.plotly_chart(fig)
```