## DESCRIPTION

Table: Transactions

+---------------+---------+

| Column Name | Type |

+---------------+---------+

| id | int |

| country | varchar |

| state | enum |

| amount | int |

| trans_date | date |

+---------------+---------+

id is the primary key of this table.

The table has information about incoming transactions.

The state column is an enum of type ["approved", "declined"].

Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.

Return the result table in **any order**.

The query result format is in the following example.

**Example 1:**

**Input:**

Transactions table:

+------+---------+----------+--------+------------+

| id | country | state | amount | trans_date |

+------+---------+----------+--------+------------+

| 121 | US | approved | 1000 | 2018-12-18 |

| 122 | US | declined | 2000 | 2018-12-19 |

| 123 | US | approved | 2000 | 2019-01-01 |

| 124 | DE | approved | 2000 | 2019-01-07 |

+------+---------+----------+--------+------------+

**Output:**

+----------+---------+-------------+---------------+-------------------+----------------------+

| month | country | trans_count | approved_count | trans_total_amount | approved_total_amount |

+----------+---------+-------------+---------------+-------------------+----------------------+

| 2018-12 | US | 2 | 1 | 3000 | 1000 |

| 2019-01 | US | 1 | 1 | 2000 | 2000 |

| 2019-01 | DE | 1 | 1 | 2000 | 2000 |

+----------+---------+-------------+---------------+-------------------+----------------------+

## SOLUTION

Option 1:

- Add 'quality' column ('rating' divided by 'position')
- Add 'poor_query_percentage' ('rating' multiplied by 100 if 'rating' < 3)
- Groupby the result table with 'query_name' column
- Compute average of 'quality' and 'poor_query_percentage' columns using mean()
- Round columns using round(x + 1e-9, 2)) instead of of round(2) for two decimal places
- round(2) doesn't pass test case 13

```
import pandas as pd

def queries_stats(queries: pd.DataFrame) -> pd.DataFrame:
    queries['quality'] = queries['rating']/queries['position']
    queries['poor_query_percentage'] = (queries['rating']<3)*100
    result = queries.groupby('query_name')[['quality','poor_query_percentage']].mean().apply(lambda x:
round(x + 1e-9, 2)).reset_index()
    return result
```

- Snapshot of the same code above for readability purposes

```
import pandas as pd

def queries_stats(queries: pd.DataFrame) -> pd.DataFrame:
    queries['quality'] = queries['rating']/queries['position']
    queries['poor_query_percentage'] = (queries['rating']<3)*100
    result = queries.groupby('query_name')[['quality','poor_query_percentage']].mean().apply
(lambda x: round(x + 1e-9, 2)).reset_index()
    return result
```

Option 2:

- Create df1 and df2 using groupby('query_name') and apply()
- Within each apply(), compute average of 'quality' and 'poor_query_percentage' columns using same equations and mean()
- Round columns using round(x + 1e-9, 2)) instead of of round(2) for two decimal places
- Join df1 and df1 for the result table

```
import pandas as pd

def queries_stats(queries: pd.DataFrame) -> pd.DataFrame:
    df1 = queries.groupby('query_name').apply(lambda x: round((x['rating']/x['position']).mean()+1e-9,
2)).reset_index(name='quality')
    df2 = queries.groupby('query_name').apply(lambda x: round(((x['rating']<3)*100).mean()+1e-9,
2)).reset_index(name='poor_query_percentage')
    result = df1.merge(df2, how='inner')
    return result
```

```
import pandas as pd
import numpy as np

def monthly_transactions(transactions: pd.DataFrame) -> pd.DataFrame:
    transactions['month'] = transactions['trans_date'].dt.strftime('%Y-%m')
    transactions['approved'] = (transactions['state']=='approved')
    transactions['approved_amount'] = transactions['approved']*transactions['amount']
    trans_count = transactions.groupby(['month', 'country'],
dropna=False)['id'].count().reset_index(name='trans_count')
    approved_count = transactions.groupby(['month', 'country'],
dropna=False)['approved'].sum().reset_index(name='approved_count')
    trans_total_amount = transactions.groupby(['month', 'country'],
dropna=False)['amount'].sum().reset_index(name='trans_total_amount')
    approved_total_amount = transactions.groupby(['month', 'country'],
dropna=False)['approved_amount'].sum().reset_index(name='approved_total_amount')
    merged1 = pd.merge(trans_count, approved_count, how='inner', on=['month',
'country'])
    merged2 = pd.merge(trans_total_amount, approved_total_amount, how='inner',
on=['month', 'country'])
    result = pd.merge(merged1, merged2, how='inner', on=['month', 'country'])
    return result
```

- Snapshot of the same code above for readability purposes

```python
import pandas as pd
import numpy as np

def monthly_transactions(transactions: pd.DataFrame) -> pd.DataFrame:
    transactions['month'] = transactions['trans_date'].dt.strftime('%Y-%m')
    transactions['approved'] = (transactions['state']=='approved')
    transactions['approved_amount'] = transactions['approved']*transactions['amount']
    trans_count = transactions.groupby(['month', 'country'], dropna=False)['id'].count().reset_index
(name='trans_count')
    approved_count = transactions.groupby(['month', 'country'], dropna=False)['approved'].sum().
reset_index(name='approved_count')
    trans_total_amount = transactions.groupby(['month', 'country'], dropna=False)['amount'].sum().
reset_index(name='trans_total_amount')
    approved_total_amount = transactions.groupby(['month', 'country'], dropna=False)
['approved_amount'].sum().reset_index(name='approved_total_amount')
    merged1 = pd.merge(trans_count, approved_count, how='inner', on=['month', 'country'])
    merged2 = pd.merge(trans_total_amount, approved_total_amount, how='inner', on=['month',
'country'])
    result = pd.merge(merged1, merged2, how='inner', on=['month', 'country'])
    return result
```