## DESCRIPTION

Table: Queries

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

| Column Name | Type    |

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

| query_name  | varchar |

| result      | varchar |

| position    | int     |

| rating      | int     |

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

This table may have duplicate rows.

This table contains information collected from some queries on a database.

The position column has a value from **1** to **500**.

The rating column has a value from **1** to **5**. Query with rating less than 3 is a poor query.

We define query quality as:

The average of the ratio between query rating and its position.

We also define poor query percentage as:

The percentage of all queries with rating less than 3.

Write a solution to find each query_name, the quality and poor_query_percentage.

Both quality and poor_query_percentage should be **rounded to 2 decimal places**.

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

The result format is in the following example.

**Example 1:**

**Input:**

Queries table:

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

| query_name | result            | position | rating |

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

| Dog | Golden Retriever | 1 | 5 |

| Dog | German Shepherd | 2 | 5 |

| Dog | Mule | 200 | 1 |

| Cat | Shirazi | 5 | 2 |

| Cat | Siamese | 3 | 3 |

| Cat | Sphynx | 7 | 4 |

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

**Output:**

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

| query_name | quality | poor_query_percentage |

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

| Dog | 2.50 | 33.33 |

| Cat | 0.66 | 33.33 |

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

**Explanation:**

Dog queries quality is ((5 / 1) + (5 / 2) + (1 / 200)) / 3 = 2.50

Dog queries poor_ query_percentage is (1 / 3) * 100 = 33.33

Cat queries quality equals ((2 / 5) + (3 / 3) + (4 / 7)) / 3 = 0.66

Cat queries poor_ query_percentage is (1 / 3) * 100 = 33.33

## 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
```

- Snapshot of the same code above for readability purposes

```python
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
```