

CSE 512 Assignment 2

Maximum Points Possible - 5

The required task is to build a simplified query processor that accesses data from the partitioned ratings table.

Input Data: - Same as in Assignment 1 i.e. ratings.dat file.

Required Task: - Below are the steps you need to follow to fulfill this assignment:

RangeQuery() –

- Implement a Python function *RangeQuery* that takes as input: (1) *RatingMinValue* (2) *RatingMaxValue* (3) *openconnection* (4) *outputPath*
- Please note that the *RangeQuery* would not use ratings table but it would use the range and round robin partitions of the ratings table.
- *RangeQuery()* then returns all tuples for which the *rating* value is larger than or equal to *RatingMinValue* and less than or equal to *RatingMaxValue*.
- The returned tuples should be stored in *outputPath*. Each line represents a tuple that has the following format such that *PartitionName* represents the full name of the partition i.e. *RangeRatingsPart1* or *RoundRobinRatingsPart4* etc. in which this tuple resides.

PartitionName, UserID, MovieID, Rating

Example:

RangeRatingsPart0,1,377,0.5
RoundRobinRatingsPart1,1,377,0.5

- Note: Please use ‘,’ (COMMA, no space character) as delimiter between *PartitionName*, *UserID*, *MovieID* and *Rating*.

PointQuery() –

- Implement a Python function *PointQuery* that takes as input: (1) *RatingValue*. (2) *openconnection* (3) *outputPath*
- Please note that the *PointQuery* would not use ratings table but it would use the range and round robin partitions of the ratings table.
- *PointQuery()* then returns all tuples for which the *rating* value is equal to *RatingValue*.
- The returned tuples should be stored in *outputPath*. Each line represents a tuple that has the following format such that *PartitionName* represents the full name of the partition i.e. *RangeRatingsPart1* or *RoundRobinRatingsPart4* etc. in which this tuple resides.

PartitionName, UserID, MovieID, Rating

Example

RangeRatingsPart3,23,459,3.5
RoundRobinRatingsPart4,31,221,0

- Note: Please use ‘,’ (COMMA, no space character) as delimiter between *PartitionName*, *UserID*, *MovieID* and *Rating*.

Please use the **function signature exactly same as mentioned in Assignment2_Interface.py**.

Naming Convention to be followed strictly:

Database name –*ddsassignment2*
 Name of Rating table –*ratings*
 Postgres User name –*postgres*
 Postgres password –*1234*

How to use tester.py:

Open the Assignment2.zip file and dump all the contents in a folder.
 Also move *ratings.dat* file to this folder.
 As per the naming conventions provided above, setup your postgresql.
 Once the setup is done, test the tester.py by simply running it.
 It should provide the following output:

```
Creating Database named as ddsassignment2
A database named ddsassignment2 already exists
Getting connection from the ddsassignment2 database
Creating and Loading the ratings table
Doing the Range Partitions
Doing the Round Robin Partitions
Performing Range Query
Performing Point Query
```

Now, you can implement your functions in Assignment2_Interface.py and once done, use the tester again to generate the output.

DONOT CHANGE tester.py and Assignment1.py. Changing anyone of these would cause problems and the system will stop working, and may lead to **deduction of marks**.

PLEASE KEEP IN MIND, this tester is just for your help. For grading purpose, an additional set of test cases would be used. It will try to break your code. So, please provide the functionalities accordingly, so that it handles all possible scenarios.

Instructions for Assignment: -

Please follow these instructions closely **else Marks will be deducted**.

1. Please follow the function signature as provided in the Assignment2_Interface.py.
2. Please use the same database name, table name, user name and password as provided in the assignment to keep it consistent.
3. Please make sure to run the provided tester and make sure there is no indentation error. In case of **any compilation error, 0 marks will be given**.
4. Do not modify Assignment1.pyc library and tester.py. In case any modification is needed, please post the same on discussion board.
5. For any case of doubt in the assignment, PLEASE USE Discussion Boards, Individual mails would not be entertained.
6. Also, it is an individual's responsibilities to clarify his/her doubts, so read and use Discussion Board extensively.
7. The output file should have the exactly same format with the sample. The grading is based on string comparison. One **mismatch** will cause 1 point loss, two mismatch will cause 2 points loss, etc.
8. Pay attention to the **outputPath**. If you output the result to a different path, the grading script will be able to locate your file. You will get 0 point.

Submission Instructions:

1. **Only** submit the .py file. Do **not** change the file name. Do **not** put it into a folder or upload a zip.
2. Multiple submissions are allowed. Only the **latest** submission will be graded. **No** late submission is accepted.

Note: -






Failure to follow the instructions provided in the document will result in the loss of the points.

1. *Ratings*

userid integer	movieid integer	rating real
1	122	0.5






2. *RangeRatingsPart0* to *RangeRatingsPartN* (where N = number of partition - 1)

- Same structure as *Ratings* table
- If value of N is 5 then here is the list of table created for Range Partitioning.

+		rangeratingspart0
+		rangeratingspart1
+		rangeratingspart2
+		rangeratingspart3
+		rangeratingspart4

3. *RoundRobinRatingsPart0* to *RoundRobinRatingsPartN* (where N = number of partition - 1)

- Same structure as *Ratings* table
- If value of N is 5 then here is the list of table created for Round Robin Partitioning

+		roundrobinratingspart0
+		roundrobinratingspart1
+		roundrobinratingspart2
+		roundrobinratingspart3
+		roundrobinratingspart4

4. *RangeRatingsMetadata*

partitionnum integer	minrating real	maxrating real
0	0	1
1	1	2
2	2	3
3	3	4
4	4	5

If *PartitionNum* is 0, then it means table *RangeRatingsPart0* is being referred and this table contains records for rating values \geq minrating and \leq maxrating.

If *PartitionNum* is N (N > 1), then it means table *RangeRatingsPartN* is being referred and this table contains records for rating values $>$ minrating and \leq maxrating.

Example: - If N is 1, the *RangeRatingsPart1* is being referred and this partition contains all the record for which the rating values are $1 < \text{value} \leq 2$.

5. *RoundRobinRatingsMetadata*

partitionnum integer	tablenextinsert integer
5	0

Partitionnum denotes total number of partition and *tablenextinsert* denotes that next partition the record should be inserted.