CS-839 NoSQL Systems
Dr. Vinu Venugopal, Chinmay Parekh & Bhulaxmi Yash Koushik

Assignment - III (Due by April 27th)

Please ensure that your solutions are consolidated into a single zip file, with your roll number serving as the filename, and submitted prior to the specified due date as mentioned. Only one member from each group is designated to make the submission. Add instructions for running the solutions to each problem in separate files named `problem_X.txt`. *Alongside the solutions, a comprehensive report must be included, containing details such as program logic, pseudocode explanations, runtime analysis, relevant screenshots, and any other relevant information. During the evaluation, if necessary, a group shall be asked to arrange a demonstration of their submitted solutions with one of the TAs. All members of the group are expected to participate in this demonstration. Furthermore, during the session, each individual member may be assessed independently.*

Part-A: PIG & PIG Latin

Problem 1. Processing YAGO dataset using PIG — 6 Points

a Load the YAGO dataset and find out the top ten frequently occurring predicates in the YAGO dataset using operators available in the Pig Latin. (3 marks)

b Identify all the given-names (i.e., the object values of the *hasGivenName* predicate) of persons who are associated with more than one *IivesIn* predicates from the YAGO dataset using the relational operators (join, grouping, etc.) in the Pig Latin. (3 marks)

Problem 2. User Defined Functions in PIG — 3 Points

a For the YAGO dataset, given a *subject-predicate* combination, determine the count of unique *objects* associated with it. Define a user defined function for this purpose. (3 marks)

Part-B: Hive & HiveQL

Problem 3. Processing YAGO dataset using Hive — 6 Points

a Load the YAGO dataset and find out the top three frequently occurring predicates in the YAGO dataset using operators available in the HiveQL. (3 marks)

b Identify all the given-names (i.e., the object values of the *hasGivenName* predicate) of persons who are associated with more than one *IivesIn* predicates from the YAGO dataset using the relational operators (join, grouping, etc.) in the HiveQL. (3 marks)

Problem 4. Partitioning and Bucketing — 6 Points

a Write a HiveQL query to find all the subjects (i.e., x) and objects (i.e., y and z) matching the pattern from the YAGO dataset:

```
?x <hasGivenName> ?y. ?x <livesIn> ?z.
```

Assignment - III (Due by April 27th)

Implement this problem by:

(i) by considering partitioning and bucketing;

(ii) by considering partitioning but not bucketing;

(iii) by considering neither partitioning nor bucketing.

For the first case alone perform a *Bucketized Merge-Join* by enabling the necessary parameters (see the note below). Compare the run time of the three cases by performing your experiments on your local system.

For case (i), follow the below steps:

1 Load the entire triples from the given *yago.tsv* files into a table named yago having three columns: subject, predicate and object.

2 Create a new table, named *yago_part_buck*, with a partition based on the predicate column and clustered based on the subject column.

3 Load data (statically) into the partitions for all the 29 predicates (listed below) in the dataset. This loading could be done by inserting data into the partitioned table from the yago table specifying the partition key – you may write all the insert statements into a single HiveQL script for loading data.

4 Write a HiveQL query to find the required pattern from the *yago_part_buck* table.

Note. You can set the following hive parameters to true (as given below), to enable the bucketized merge-join.

set hive.auto.convert.sortmerge.join = true;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;

29 predicates from YAGO:

```
<actedIn>, <hasAcademicAdvisor>, <hasChild>, <hasFamilyName>, <hasWebsite>,
<hasWonPrize>, <isInterestedIn>, <isKnownFor>, <directed>, <edited>,
<graduatedFrom>,  <hasGender>, <hasMusicalRole>, <isCitizenOf>, <isMarriedTo>,
<isPoliticianOf>, <playsFor>, <worksAt>, <wroteMusicFor>, <created>, <diedIn>,
<hasGivenName>, <influences>, <isAffiliatedTo>, <isLeaderOf>, <livesIn>, <owns>,
<participatedIn>, <wasBornIn>
```