

NoSQL Systems : Assignment 3

Krutik Patel
IMT2021024

Suyash Chavan
IMT2021048

Sai Madhavan G
IMT2021101

I. PROBLEM 1 - PIG

A. Part A : Top 10 frequently occurring predicates

1) **Relevant files:** The files from our submission relevant to this subpart are:

- parta/problem1_a.txt
- parta/output/1a/
- parta/src/1a.pig
- parta/logs/1a.log

2) **Program Logic:** The program logic to find Top 10 frequently occurring predicates is as follows:

- a) *LOAD Data*: Load the data from the TSV file `yago_full_clean.tsv` into a relation named `lines`, using the `PigStorage` function to specify the delimiter as a space (' ') and define the schema with three fields: `subject`, `predicate`, and `object`, all of type `chararray`.
- b) *GROUP Data*: Group the `lines` relation by the `predicate` field, creating groups of tuples with the same `predicate` value. This step aggregates the data based on the `predicate`, resulting in a relation named `predicate_groups`.
- c) *COUNT Predicate Groups*: For each group of lines with the same `predicate`, count the number of lines in the group using the `COUNT` function. Rename the `predicate` field as `predicate` and the count result as `count`. Store the results in a relation named `predicate_counts`.
- d) *ORDER Predicate Counts*: Order the `predicate_counts` relation by the `count` field in descending order (`DESC`), and if there are ties, then by the `predicate` field in ascending order (`ASC`). This step sorts the counts in descending order and within each count, sorts the predicates alphabetically.
- e) *LIMIT Top 10*: Select the top 10 records from the `ordered_predicate_counts` relation. This step limits the output to only the top 10 predicates with the highest counts.

[illegible]

Fig. 1. The beginning of execution for problem 1.a

- f) *STORE Results*: Store the top 10 records in a directory named 1a.

3) **Pseudocode:** The pseudocode to find Top 10 frequently occurring predicates is as follows:

Algorithm 1 Top 10 frequently occurring predicates

- 1. Load Data:** Load data from TSV file into relation *lines* using `PigStorage(' ')`
- 2. Group Data:** Group *lines* by *predicate* field, creating *predicate_groups*
- 3. Count Predicate Groups:** Count lines in each group, store results in *predicate_counts*
- 4. Order Predicate Counts:** Order *predicate_counts* by count DESC, predicate ASC, store in *ordered_predicate_counts*
- 5. Select Top 10:** Select top 10 records from *ordered_predicate_counts*, store in *top10*
- 6. Store Results:** Store *top10* records in directory `1a`

- 4) **Observations:** Refer figures 1 and 2

- a) Metrics:*

- **Number of Maps:** 4
- **Number of Reducers:** 1
- **Time taken:** 1 minute 33 seconds
- **Total Records written:** 10
- **Features Used:** GROUP BY, ORDER BY, LIMIT

Fig. 2. The end of execution for problem 1.a

Fig. 3. The beginning of execution for problem 1.b

B. Part B : Object values of the `hasGivenName` predicate

1) **Relevant files:** The files from our submission relevant to this subpart are:

- parta/problem1_b.txt
- parta/output/lb/
- parta/output/lb_d*istinct*/parta/src/lb.pig
- parta/src/lb_d*istinct.pig*parta/logs/lb.log
- parta/logs/lb_d*istinct.log*

2) **Program Logic:** The program logic to find Object values of the hasGivenName predicate is as follows:

- a) *LOAD Data:* Load the data from the TSV file `yago_full_clean.tsv` into a relation named `records`, specifying the delimiter as a space (' ') and defining the schema with three fields: `subject`, `predicate`, and `object`, all of type `chararray`.
- b) *FILTER by Predicate:* Filter the `records` relation to get all subjects with the `<livesIn>` predicate. Store the filtered records in a relation named `lives_in_records`.
- c) *GROUP by Subject:* Group the `lives_in_records` relation by the `subject` field, creating groups of tuples with the same `subject`. Store the result in a relation named `lives_in_groups`.
- d) *COUNT LivesIn Records:* For each group in `lives_in_groups`, count the number of records using the `COUNT` function. Rename the `subject` field as `li_subject` and the count result as `count`. Store the results in a relation named `subject_li_count`.
- e) *FILTER Valid Subjects:* Filter the `subject_li_count` relation to keep only those subjects with a count greater than 1. Store the filtered records in a relation named `valid_subjects`.
- f) *FILTER by hasGivenName Predicate:* Filter the `records` relation to get all records with the `<hasGivenName>` predicate. Store the filtered records in a relation named `given_name_records`.
- g) *JOIN Valid Subjects with Given Names:* Join the `valid_subjects` relation with the `given_name_records` relation based on the

subject field. Store the joined records in a relation named joined records.

- h) *GENERATE Objects:* For each record in `joined_records`, extract the object field. Store the extracted objects in a relation named `objects`.
- i) *STORE Results:* Store the `objects` relation into the directory `./output/1b`.

3) **Pseudocode:** The pseudocode to find Object values of the hasGivenName predicate is as follows:

Algorithm 2 Object values of the hasGivenName predicate

- 1. Load Data:** Load data from TSV file into relation *records* using `PigStorage(' ')`
- 2. Filter by Predicate:** Filter *records* to get subjects with the `<livesIn>` predicate, store in *lives_in_records*
- 3. Group by Subject:** Group *lives_in_records* by subject, store in *lives_in_groups*
- 4. Count LivesIn Records:** Count records for each subject in *lives_in_groups*, store in *subject_li_count*
- 5. Filter Valid Subjects:** Filter *subject_li_count* for subjects with count ≥ 1 , store in *valid_subjects*
- 6. Filter by hasGivenName Predicate:** Filter *records* to get records with the `<hasGivenName>` predicate, store in *given_name_records*
- 7. Join Valid Subjects with Given Names:** Join *valid_subjects* with *given_name_records* on subject, store in *joined_records*
- 8. Extract Objects:** Extract object field from *joined_records*, store in *objects*
- 9. Store Results:** Store *objects* in directory `./output/1b`

4) **Observations:** Refer figures 3, 4, 5 and 6

a) Metrics:

- **Number of Maps:** 2
- **Number of Reducers:** 1
- **Time taken:** 1 minute 28 seconds
- **Total Records written:** 10728
- **Features Used:** GROUP BY, HASH JOIN, FILTER

[illegible]

- 1. Load Data:** Load data from TSV file into relation *records* with schema (subject:chararray, predicate:chararray, object:chararray)
- 2. Register Jar:** Register custom Pig UDF contained in *SPCount.jar*
- 3. Group All Records:** Group all records together
- 4. Compute SP Count:** For each group, compute count of occurrences of subject-predicate pair '*< Alice_Roberts >*' and '*< isCitizenOf >*' using *SPCOUNT* UDF
- 5. Store Results:** Store computed SP count in directory */output/2*

```
2024-04-26 12:01:42,866 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.Task Task 'HDFSLocalJob36796_0001_0_000000_0' done
2024-04-26 12:01:42,893 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.Task Final Counters for attempt_1202034796_0001_0_000000_0: Counters: 18
File System Counters:
  FILE: Number of bytes read=435723
  FILE: Number of bytes written=2424089
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
MapReduce Framework:
  Map input records=833793
  Map output records=833793
  Map output bytes=915116
  Map output materialized bytes=0
  Map input splits=100
  Combine input records=833793
  Combine output records=0
  Spilled Records=0
  Failed Shuffles=0
  Retmap Map failures=0
  GC time elapsed (ms)=1243
  Total committed heap usage (bytes)=762720384
File Input Format Counters:
  Bytes Read=0
2024-04-26 12:01:42,902 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.LocalJobRunner Finalizing task: attempt_1202034796_0001_0_000000_0
2024-04-26 12:01:42,902 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.LocalJobRunner Starting task: attempt_1202034796_0001_0_000000_0
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter File output committer factory defined, defaulting to FileOutputCommitterFactory
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter File OutputCommitter Algorithm version is 2
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter FileOutputCommitter skip cleanup: temporary folders under output directory/fake, ignore ch
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.Task Using ResourceCalculatorTranslator: {}
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.Task Processing split: Number of splits: 3
Total Length = 3395452
Input split(s):
  Length = 1131850.33
  (Classname: org.apache.hadoop.mapreduce.lib.input.FileInput
  Location:
  )
```

Fig. 7. The beginning of execution for problem 2

```
JobName: job_1202034796_0001_0_000000_0
JobType: MapReduce
JobStatus: SUCCEEDED
JobProgress: 100%
JobCounters:
  FILE: Number of bytes read=435723
  FILE: Number of bytes written=2424089
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
MapReduce Framework:
  Map input records=833793
  Map output records=833793
  Map output bytes=915116
  Map output materialized bytes=0
  Map input splits=100
  Combine input records=833793
  Combine output records=0
  Spilled Records=0
  Failed Shuffles=0
  Retmap Map failures=0
  GC time elapsed (ms)=1243
  Total committed heap usage (bytes)=762720384
File Input Format Counters:
  Bytes Read=0
2024-04-26 12:01:42,902 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.LocalJobRunner Finalizing task: attempt_1202034796_0001_0_000000_0
2024-04-26 12:01:42,902 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.LocalJobRunner Starting task: attempt_1202034796_0001_0_000000_0
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter File output committer factory defined, defaulting to FileOutputCommitterFactory
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter File OutputCommitter Algorithm version is 2
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter FileOutputCommitter skip cleanup: temporary folders under output directory/fake, ignore ch
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.Task Using ResourceCalculatorTranslator: {}
2024-04-26 12:01:42,909 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapred.Task Processing split: Number of splits: 3
Total Length = 3395452
Input split(s):
  Length = 1131850.33
  (Classname: org.apache.hadoop.mapreduce.lib.input.FileInput
  Location:
  )
```

Fig. 8. The beginning of execution for problem 2

4) **Observations:** The following were the observed metrics when the input was 'Alice_Roberts' and 'isCitizenOf' (Fig. 7 and 8):

- **Number of Maps:** 18
- **Number of Reducers:** 1
- **Time taken:** 1 minute 6 seconds
- **Features Used:** GROUP_BY, COMBINER

III. PROBLEM 3 - HIVE

A. Part A : Top 3 frequently occurring predicates

1) **Relevant files:** The files from our submission relevant to this subpart are:

- partb/problem1/problem2_a.txt
- partb/problem1/output/2a/
- partb/problem1/src/2a.hiveql
- partb/problem1/logs/2a.log

2) **Program Logic:** The program logic to find Top 3 frequently occurring predicates is as follows:

- a) **Create Table:** Create a table named records with three columns: subject, predicate, and object, specifying the delimiter as a space and defining the location as './tables'.
- b) **Load Data:** Load data from the TSV file yago_full_clean.tsv into the records table.
- c) **Group by Predicate:** Create a view named predicate_groups by grouping the records table by the predicate column and counting the occurrences of each predicate.

```
Connected to: Apache Hive (version 3.13.0)
Driver: Hive JDBC (version 3.13.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
HiveLine version 3.13 by Apache Hive
0 jdbc:hive2://...
>>> ... Create table for records
>>> CREATE TABLE records (
  subject STRING,
  predicate STRING,
  object STRING
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
LOCATION: /tmp/hive_20240426_091344_0001_0_000000_0
24/04/26 09:13:44 [HiveServer2-Background-Pool: Thread-45]: WARN metastore.HiveMetaStore: Location: file:/home/youshay/Downloads/Assignments/Pig/Hive/HiveQL_Assignment_3/partb/problem1/tables specified for non-external table:records
OK
No rows affected (2.684 seconds)
>>> ... Load data into table records
>>> LOAD DATA LOCAL INPATH './data/yago_full_clean.tsv' INTO TABLE records;
24/04/26 09:13:44 [HiveServer2-Background-Pool: Thread-49]: WARN metastore.ObjectStore: datanucleus.autostartMechanism is set to unsupported value null
1 Setting it to value: ignored
Loading data to table default.records
24/04/26 09:13:44 [HiveServer2-Background-Pool: Thread-49]: WARN metastore.ObjectStore: datanucleus.autostartMechanism is set to unsupported value null
1 Setting it to value: ignored
OK
No rows affected (6.611 seconds)
>>> ... Group by predicate
>>> CREATE VIEW predicate_groups AS
SELECT predicate, count(*) as count
FROM records
```

Fig. 9. The beginning of execution for problem 3.a

```
ache.hadoop.mapreduce.TaskCounter Input
2024-04-26 09:13:44 Stage-1 map = 0%, reduce = 0%
2024-04-26 09:13:44 Stage-1 map = 100%, reduce = 0%
2024-04-26 09:13:44 [pool-15-thread-1]: WARN Impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/26 09:13:44 [pool-15-thread-1]: WARN Impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/26 09:13:44 [pool-15-thread-1]: WARN Impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-04-26 09:13:44 Stage-1 map = 100%, reduce = 100%
Ended Job = job_local147362729_0001
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=number
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=number
In order to set a constant number of reducers:
  set mapreduce.reduce.number=
24/04/26 09:13:44 [HiveServer2-Background-Pool: Thread-64]: WARN Impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/26 09:13:44 [HiveServer2-Background-Pool: Thread-64]: WARN Impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/26 09:13:44 [HiveServer2-Background-Pool: Thread-64]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement
the tool interface and execute your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
24/04/26 09:13:45 [pool-15-thread-1]: WARN Impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-04-26 09:13:45 Stage-2 map = 100%, reduce = 100%
Ended Job = job_local147362729_0001
Moving data to directory ./output/2a_test
Hadoop job finished
Stage-Stage-1: HDFS Read: 0 HDFS Write: 0 SUCCESS
Stage-Stage-2: HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 sec
OK
No rows affected (12.833 seconds)
>>>
0 jdbc:hive2://...
```

Fig. 10. The end of execution for problem 3.a

d) **Order by Count:** Create a view named ordered_predicate_counts by ordering the predicate_groups view by count in descending order.

e) **Select Top 3 Records:** Create a view named top3 by selecting the top 3 records from the ordered_predicate_counts view.

f) **Store Results:** Store the top 3 records from the top3 view into the directory './output/2a' with space delimiter.

3) **Pseudocode:** The pseudocode to find Top 3 frequently occurring predicates is as follows:

Algorithm 4 find Top 3 frequently occurring predicates

1. **Create Table:** Create a table named records with columns subject, predicate, and object, delimited by space, and located at './tables'.

2. **Load Data:** Load data from yago_full_clean.tsv into the records table.

3. **Group by Predicate:** Group the records table by predicate, counting occurrences for each predicate, and store in predicate_groups view.

4. **Order by Count:** Order predicate_groups by count in descending order, storing in ordered_predicate_counts view.

5. **Select Top 3 Records:** Select top 3 records from ordered_predicate_counts, storing in top3 view.

6. **Store Results:** Store the top 3 records from top3 into directory './output/2a' with space delimiter.

4) **Observations:** Refer figures 9 and 10

a) **Metrics:**

- **Number of Maps:** 1
- **Number of Reducers:** 1
- **Time taken:** 12.81 seconds (Faster than PIG)
- **Features Used:** GROUP_BY, VIEW, ORDER_BY, LIMIT

B. Part B : Object values of the hasGivenName predicate

1) **Relevant files:** The files from our submission relevant to this subpart are:

- partb/problem1/problem2_b.txt
- partb/problem1/output/2b/
- partb/problem1/src/2b.hiveql
- partb/problem1/logs/2b.log

2) **Program Logic:** The program logic to find Object values of the hasGivenName predicate is as follows:

- Load Data:** Load data from the TSV file into a Hive table named records2. Define the schema with columns subject, predicate, and object. Specify the delimiter as space and the location as './tables'.
- Load Data into Table:** Load data from the local path './data/yago_full_clean.tsv' into the records2 table.
- Get Subjects with Multiple 'livesIn' Predicates:**
 - Create a view lives_in_records to select all records from records2 where the predicate is '<livesIn>'.
 - Create a view lives_in_groups to count the occurrences of each subject in lives_in_records.
 - Create a view valid_subjects to filter lives_in_groups to keep only subjects with more than one '<livesIn>' predicate.
- Join with 'hasGivenName' Predicate:**
 - Create a view given_name_records to select all records from records2 where the predicate is 'hasGivenName'.
 - Create a view joined_records by joining valid_subjects with given_name_records on the subject field.
- Store Result:** Store the result from joined_records into the HDFS directory './output/2b'. Fields should be terminated by space.

3) **Pseudocode:** The pseudocode to find Object values of the hasGivenName predicate is as follows:

4) **Observations:** Refer figures 11 and 12

a) **Metrics:**

- **Number of Maps:** 3
- **Number of Reducers:** 1
- **Time taken:** 39.55 seconds (Faster than PIG)
- **Features Used:** GROUP_BY, JOIN, VIEW

Algorithm 5 find Object values of the hasGivenName predicate

1. **Load Data:** Load data from the TSV file into a Hive table named records2, with columns subject, predicate, and object, delimited by space, and located at './tables'.

2. **Load Data into Table:** Load data from the local path './data/yago_full_clean.tsv' into the records2 table.

3. **Get Subjects with Multiple 'livesIn' Predicates:**

Create a view lives_in_records to select all records from records2 where the predicate is '<livesIn>'.

Create a view lives_in_groups to count the occurrences of each subject in lives_in_records.

Create a view valid_subjects to filter lives_in_groups to keep only subjects with more than one '<livesIn>' predicate.

4. **Join with 'hasGivenName' Predicate:**

Create a view given_name_records to select all records from records2 where the predicate is '<hasGivenName>'.

Create a view joined_records by joining valid_subjects with given_name_records on the subject field.

5. **Store Result:** Store the result from joined_records into the HDFS directory './output/2b', with fields terminated by space.

```

$ jdbchive2//5 from 2b.hiveql
>>> -- Create table for records
>>> CREATE TABLE records (
  subject STRING,
  predicate STRING,
  object STRING
)
Row FORMAT DELIMITED FIELDS TERMINATED BY ' '
LOCATION '/tables/'
24/04/26 09:13:43 [hiveServer2-Background-Pool: Thread-49]: WARN metastore.HiveMetaStore: Location: file:/home/suyash/Downloads/Assignment3.Pig/Alive/NoSQL/assignment_3/partb/problem1/tables specified for non-external table:records
OK
No rows affected (2.004 seconds)
>>> -- Load data into table records
>>> LOAD DATA LOCAL INPATH './data/yago_full_clean.tsv' INTO TABLE records;
24/04/26 09:13:44 [hiveServer2-Background-Pool: Thread-49]: WARN metastore.ObjectStore: datanucleus.autoStartMechanismMode is set to unsupported value null
1 - Setting it to value: ignored
Loading data to table default.records
24/04/26 09:13:50 [hiveServer2-Background-Pool: Thread-49]: WARN metastore.ObjectStore: datanucleus.autoStartMechanismMode is set to unsupported value null
1 - Setting it to value: ignored
OK
No rows affected (6.911 seconds)
>>> -- Group by predicate
>>> CREATE VIEW predicate_group AS
SELECT predicate, COUNT(*) as count
FROM records
GROUP BY predicate;
24/04/26 09:13:58 [hiveServer2-Job-400-0004-a7297f0c700 main]: WARN metastore.ObjectStore: datanucleus.autoStartMechanismMode is set to unsupported value null
1 - Setting it to value: ignored
OK

```

Fig. 11. The beginning of execution for problem 3.b

```

2024-04-26 09:19:15,228 Stage-1 map = 100%, reduce = 100%
Ended Job = job_local42976925_0000
Stage-1 is filtered out by condition resolver.
Stage-1 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Execution completed successfully.
MapredLocal task succeeded
Launching Job 3 out of 4
Number of reduce tasks is set to 0 since there's no reduce operator
24/04/26 09:19:20 [hiveServer2-Background-Pool: Thread-144]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/26 09:19:20 [hiveServer2-Background-Pool: Thread-144]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/26 09:19:20 [hiveServer2-Background-Pool: Thread-144]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement it at the Tool interface and encode your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
24/04/26 09:19:22 [hiveServer2-Background-Pool: Thread-144]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task3Counter is deprecated. Use org.apache.hadoop.mapreduce.Task3Counter instead.
2024-04-26 09:19:22,192 Stage-5 map = 0%, reduce = 0%
2024-04-26 09:19:22,196 Stage-5 map = 100%, reduce = 0%
Ended Job = job_local434824239_0004
Writing data to directory - /output/2b/test
MapReduce Job Launched:
Stage-Stage-1: HDFS Read: 0 HDFS Write: 0 SUCCESS
Stage-Stage-5: HDFS Read: 0 HDFS Write: 0 SUCCESS
Total Mapreduce CPU Time Spent: 0 msec
OK
No rows affected (20.567 seconds)
$ jdbchive2//5

```

Fig. 12. The end of execution for problem 3.b

```

$ jdbchive2//5 source scripts/part_1/create_yago_part_buck.hiveql;
24/04/27 12:20:13 [hiveServer2-Background-Pool: Thread-61]: WARN metastore.HiveMetaStore: Location: file:/home/Arutik/Desktop/NoSQL/assignment_3/hive_tester/yago_part_specified for non-external table:yago_part_buck
OK
No rows affected (0.134 seconds)
$ jdbchive2//5

```

Fig. 13. Create query for partitioning and bucketing

```

to value: ignored
24/04/27 12:37:53 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported
to value: ignored
24/04/27 12:37:53 [HiveServer2-Background-Pool: Thread-285]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the
ing a different execution engine (i.e. spark, tez) or using Hive 1.x releases.
Query ID = krutik_20200427123753_f421414-6b33-4932-a192-af6d094577e
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks determined at compile time: 6
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reducenumber=
24/04/27 12:37:53 [HiveServer2-Background-Pool: Thread-285]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:37:53 [HiveServer2-Background-Pool: Thread-285]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:37:53 [HiveServer2-Background-Pool: Thread-285]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not perform
ed and execute your application with ToolRunner to remedy this.
Job running in-process (Local Hadoop)
WARN : Hive-on-MR is deprecated in hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. s
releases.
24/04/27 12:37:54 [HiveServer2-Background-Pool: Thread-285]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.TaskCounter is deprecated
duce.TaskCounter instead
2024-04-27 12:37:54,721 Stage-1 map = 0%, reduce = 0%
2024-04-27 12:37:59,720 Stage-1 map = 100%, reduce = 0%
24/04/27 12:37:59 [pool-65-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:38:00 [pool-65-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:38:00 [pool-65-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:38:00 [pool-65-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!

```

Fig. 14. Insert query for partitioning and bucketing

```

6: jdbc:hive2:// source script/part-3/create_yago_no_buck_hiveql
24/04/27 12:41:24 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:41:24 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:41:24 [HiveServer2-Background-Pool: Thread-2883]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:41:24 [HiveServer2-Background-Pool: Thread-2883]: WARN metastore.HiveMetastore: Location: file:/home/krutik/Desktop/NGSQL/assignment_3/hive_tester/yago_no_part
no_buck specified for non-external table:yago part no buck
OK
No rows affected (0.06 seconds)
6: jdbc:hive2://

```

Fig. 15. Create query for partitioning and no bucketing

IV. PROBLEM 4 - HIVE

A. Database Creation:

1) Partitioning and Bucketing : refer 13, 14

: Program logic :

- Create Table:** Create a Hive table named yago_part_buck with columns subject and object. Partition the table by the predicate column. Cluster the table by the subject column into 6 buckets.
- Specify Row Format:** Define the row format for the table. Fields are delimited by '\t' (tab) and lines are terminated by '\n' (newline).
- Specify Location:** Specify the location for storing the table data as 'yago_part'.

Pseudocode :

Algorithm 6 Creating Table yago_part_buck

1. Create Table yago_part_buck:

Specify columns: subject (string), object (string)
 Partition by: predicate (string)
 Cluster by: subject (string) into 4 buckets
 Row format: delimited by '\t', lines terminated by '\n'
 Location: 'yago_part'

```

24/04/27 12:42:23 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:42:23 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:42:23 [HiveServer2-Background-Pool: Thread-2210]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider u
sing a different execution engine (i.e. spark, tez) or using Hive 1.x releases.
Query ID = krutik_20200427124223_6b0534ff-f4de-4021-a22b-29d5b0f0c306
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reducenumber=
24/04/27 12:42:24 [HiveServer2-Background-Pool: Thread-2210]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:42:24 [HiveServer2-Background-Pool: Thread-2210]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:42:24 [HiveServer2-Background-Pool: Thread-2210]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool inter
face and execute your application with ToolRunner to remedy this.
Job running in-process (Local Hadoop)
WARN : Hive-on-MR is deprecated in hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.x
releases.
24/04/27 12:42:25 [HiveServer2-Background-Pool: Thread-2210]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.TaskCounter is deprecated. Use org.apache.hadoop.mapr
educe.TaskCounter instead
2024-04-27 12:42:25,344 Stage-1 map = 0%, reduce = 0%
24/04/27 12:42:25 [LocalJobRunner Map Task Executor #0]: WARN LazyStruct: Extra bytes detected at the end of the row ignoring similar problems.
24/04/27 12:42:30 [pool-583-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:42:30 [pool-583-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:42:30 [pool-583-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:42:30 [pool-583-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-04-27 12:42:30,334 Stage-1 map = 100%, reduce = 100%

```

Fig. 16. Insert query for partitioning and no bucketing

```

6: jdbc:hive2:// source script/part-3/create_yago_no_part_no_buck_hiveql
24/04/27 12:45:04 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:45:04 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:45:04 [HiveServer2-Background-Pool: Thread-3215]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:45:04 [HiveServer2-Background-Pool: Thread-3215]: WARN metastore.HiveMetastore: Location: file:/home/krutik/Desktop/NGSQL/assignment_3/hive_tester/yago_no_part
no_buck specified for non-external table:yago no part no buck
OK
No rows affected (0.852 seconds)
6: jdbc:hive2://

```

Fig. 17. Create query for no partitioning and no bucketing

```

24/04/27 12:43:43 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:43:43 [592f108f-57aa-48b7-99c8-abd4f58134f main]: WARN metastore.ObjectStore: datacatalog.autoStartMechanismMode is set to unsupported value null . Setting it
to value: ignored
24/04/27 12:43:43 [HiveServer2-Background-Pool: Thread-2711]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider u
sing a different execution engine (i.e. spark, tez) or using Hive 1.x releases.
Query ID = krutik_20200427124343_7d6c7b79-ec70-4742-ae0b-edcd272575ee
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reducenumber=
24/04/27 12:43:43 [HiveServer2-Background-Pool: Thread-2711]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:43:43 [HiveServer2-Background-Pool: Thread-2711]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:43:43 [HiveServer2-Background-Pool: Thread-2711]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool inter
face and execute your application with ToolRunner to remedy this.
Job running in-process (Local Hadoop)
WARN : Hive-on-MR is deprecated in hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.x
releases.
24/04/27 12:43:44 [HiveServer2-Background-Pool: Thread-2711]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.TaskCounter is deprecated. Use org.apache.hadoop.mapr
educe.TaskCounter instead
2024-04-27 12:43:44,729 Stage-1 map = 0%, reduce = 0%
24/04/27 12:43:44 [LocalJobRunner Map Task Executor #0]: WARN LazyStruct: Extra bytes detected at the end of the row ignoring similar problems.
24/04/27 12:43:48 [pool-476-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:43:48 [pool-476-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:43:48 [pool-476-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
24/04/27 12:43:48 [pool-476-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2024-04-27 12:43:48,333 Stage-1 map = 100%, reduce = 100%

```

Fig. 18. Insert query for no partitioning and no bucketing

2) Partitioning but not Bucketing : refer 15, 16

: Program logic :

- Create Table:** Define the table yago_part_no_buck with columns subject and object, both of type string.
- Partitioning:** Partition the table by the column predicate, which is also of type string.
- Row Format:** Specify the row format as delimited by '\t', indicating tab separation, and terminated by '\n', indicating newline.
- Location:** Set the location of the table data to 'yago_part_no_buck'.

Pseudocode :

Algorithm 7 Creating Table yago_part_no_buck

1. Create Table yago_part_no_buck:

Specify columns: subject (string), object (string)
 Partition by: predicate (string)
 Row format: delimited by '\t', lines terminated by '\n'
 Location: 'yago_part_no_buck'

3) Neither Partitioning nor Bucketing : refer 17, 18

: Program logic :

- Create Table:** Define the table yago_no_part_no_buck with columns subject, predicate, and object, all of type string.
- Row Format:** Specify the row format as delimited by space (' '), indicating space separation, and terminated by '\n', indicating newline.
- Location:** Set the location of the table data to 'assignment/'.

Pseudocode :

B. Query Logic :

1) Partitioning and Bucketing : refer 19

: Program logic :

- Insert Overwrite Local Directory 'out-put/yago_part_buck':

