NoSQL Assignment 3

Group members:
• Adithya Nangarath - IMT2022024
• Mallikarjun Chakoti - IMT2022116
• Ullas G - IMT2022125
• Vrajnandak Nangunoori - IMT2022527

Introduction

This assignment involves the integration and analysis of student academic records drawn from three separate datasets, each offering unique insights into different aspects of student engagement with courses. The datasets are:

Course_Attendance.csv
This file contains detailed records of student attendance for various courses. The key columns include:
Course, Instructor, Name, Email Id, Member Id, Number of Classes Attended, Number of Classes Absent, and Average Attendance.
Here, the Member Id corresponds to the unique Student ID for each individual. This dataset provides a comprehensive view of students' attendance patterns and participation in their respective courses.

Enrollment_Data.csv
This dataset records enrollment details of students in different courses. It provides insights into which students have enrolled in which courses, along with relevant enrollment metadata. The columns include:
Serial No., Course, Status, Course Type, Course Variant, Academia+LMS, Student ID, Student Name, Program, Batch, Period, Enrollment Date, and Primary Faculty.
This file is crucial for understanding course registration trends and the structure of academic programs.

Grade_Roster.csv
This file contains academic performance data, specifically the grades obtained by students in various courses. Its columns include:

Academy Location, Student ID, Student Status, Admission ID, Admission Status, Student Name, Program Code / Name, Batch, Period, Subject Code / Name, Course Type, Section, Faculty Name, Course Credit, Obtained Marks/Grade, Out of Marks/Grade, and Exam Result.
It captures the final outcomes of student performance in each enrolled course.

Steps to start using hive:

1)start hadoop services: command - start-all.sh
2)In another terminal start hive server , command - hive --service hiveserver2, this starts the hive server which allows users to connect.
3)In another terminal run - hive --service metastore &, to start the metastore
4)In a new terminal - run - beeline -u jdbc:hive2://, this opens up beeline(hive prompt) where we can write create tables ,write queries , etc.

Question 1: Table Creation and Data Loading

For this assignment, we created four tables in Hive:

course_attendance
enrollment_data
grade_roster
error_log

The first three tables store data from the CSV files, while the **error_log** table is used to store all incorrect or problematic records found in the other three tables.
The schema (column definitions) for all four tables is mentioned in the file 'Q1_Create_Tables.sql'

Step 1: Creating the Tables
We first created the tables in Hive using the queries given in the SQL file. This defines the structure/schema for each table.

Step 2: Loading Data into Hadoop
To load the CSV files into Hive, we first copied them into HDFS (Hadoop Distributed File System) using these commands:

hdfs dfs -put ~/Downloads/Enrollment_data.csv /user/ullas/Assignment3/
hdfs dfs -put ~/Downloads/Course_Attendance.csv /user/ullas/Assignment3/

hdfs dfs -put ~/Downloads/GradeRosterReport.csv /user/ullas/Assignment3/
This stored the files inside Hadoop so Hive could access them.

Step 3: Loading Data into Hive Tables
After the files were in Hadoop, we loaded them into the Hive tables using the following commands:

LOAD DATA INPATH '/user/ullas/Assignment3/Enrollment_data.csv' INTO TABLE enrollment_data;

LOAD DATA INPATH '/user/ullas/Assignment3/Course_Attendance.csv' INTO TABLE course_attendance;

LOAD DATA INPATH '/user/ullas/Assignment3/GradeRosterReport.csv' INTO TABLE grade_roster;
Hive reads the data from the CSV and stores it inside the respective tables. If a value is missing or blank in the CSV, Hive stores it as NULL.

Handling Invalid or Missing Data
To handle records with errors, we created an error_log table. This table helps us store:

source_file: Which CSV file the error came from.
column_name: Which column has the problem.
error_type: For example, "Missing Value" or "Invalid Format".
full_row: The full row as a string.
We used the **concat_ws(',', col1, col2, ..., colN)** function to combine all columns into a single string for the full_row.

The logic for inserting incorrect records from each table into the error_log is written in ‘Q1_Load_Data_Error_Table.sql’.

Problem Faced While Inserting Data
While inserting data into the tables, we faced an issue with columns like Instructor and Primary Faculty. In some records, these columns had multiple values separated by commas, like:

Chandrashekar Ramanathan,Thangaraju B.
rc@iiitb.ac.in, b.thangaraju@iiitb.ac.in
By default, Hive uses a comma (,) to separate columns in CSV. So when there is a comma within a column value, Hive treats it as a new column, which causes errors.

Solution: Using SerDe (Serializer/Deserializer)
To fix this, we used SerDe (Serializer/Deserializer). SerDe helps Hive understand how to read and write complex data formats.

We used an OpenCSV SerDe that correctly handles:

Fields enclosed in double quotes (like "name1,name2")
Commas inside fields
This made sure that multiple instructors or faculty names were treated as a single field, not split into different columns.

Question 2: Creation of a Unified Data Warehouse and Analytical Querying

In this, our idea was to create a single data warehouse table by consolidating the data of the three most crucial existing tables—enrollment_data, course_attendance, and grade_roster. The rationale for this step was to make the subsequent querying operations simpler by having the student-related information in one format.

Designing the Data Warehouse
We began by identifying the key fields that would be required for analysis. Instead of carrying all columns from the original tables, we selected only those that were important for our queries and insights.

The resulting data warehouse table, named student_data_warehouse, included the following fields:

row_id (auto-generated row number for uniqueness)
course, status, course_type, course_variant
student_id, student_name, program, batch, period
primary_faculty, instructors, email_id
classes_attended, classes_absent, avg_attendance
obtained_grade, out_of_grade, exam_result
The table was defined in Hive using a CSV-friendly format, and the schema and creation logic can be found in Q2_Data_Warehouse_Creation.sql.

Joining the Source Tables
To populate this warehouse, we needed to bring together information from the three tables mentioned above. This required joining them on appropriate key columns:

First, we joined enrollment_data and course_attendance on the columns **student_id = member_id** and **course = course**. This allowed us to combine registration and attendance details for each student-course pair.
Then, we joined the result with grade_roster on **student_id and course = subject_name** to include grade-related data.
This joining process gave us a comprehensive view of each student's enrollment, attendance, and academic performance for each course.

Initial Issue with Duplicates
At first, we performed the join and inserted the data into the warehouse without using any GROUP BY clause. This caused an unexpected issue—a large number of duplicate records.

Upon examination, we found that certain of the source tables (particularly the CSVs) had duplicate entries. For example, the same combination of student_id and course would occur several times because of duplicate rows in either enrollment_data, course_attendance, or grade_roster. When these duplicates were joined, they created multiple duplicates, where many similar but slightly different records matched and ended up creating many repeated combinations in the final warehouse.

This meant that querying over the warehouse could yield misleading results. For example, while checking the number of distinct (student_id, course) pairs, we noticed it was much smaller than the total number of rows—indicating that the same pair appeared multiple times with slightly varied attributes.

Using GROUP BY to Eliminate Duplicates
In order to fix this problem, we added a GROUP BY clause to the last insert query. Here is how it worked:

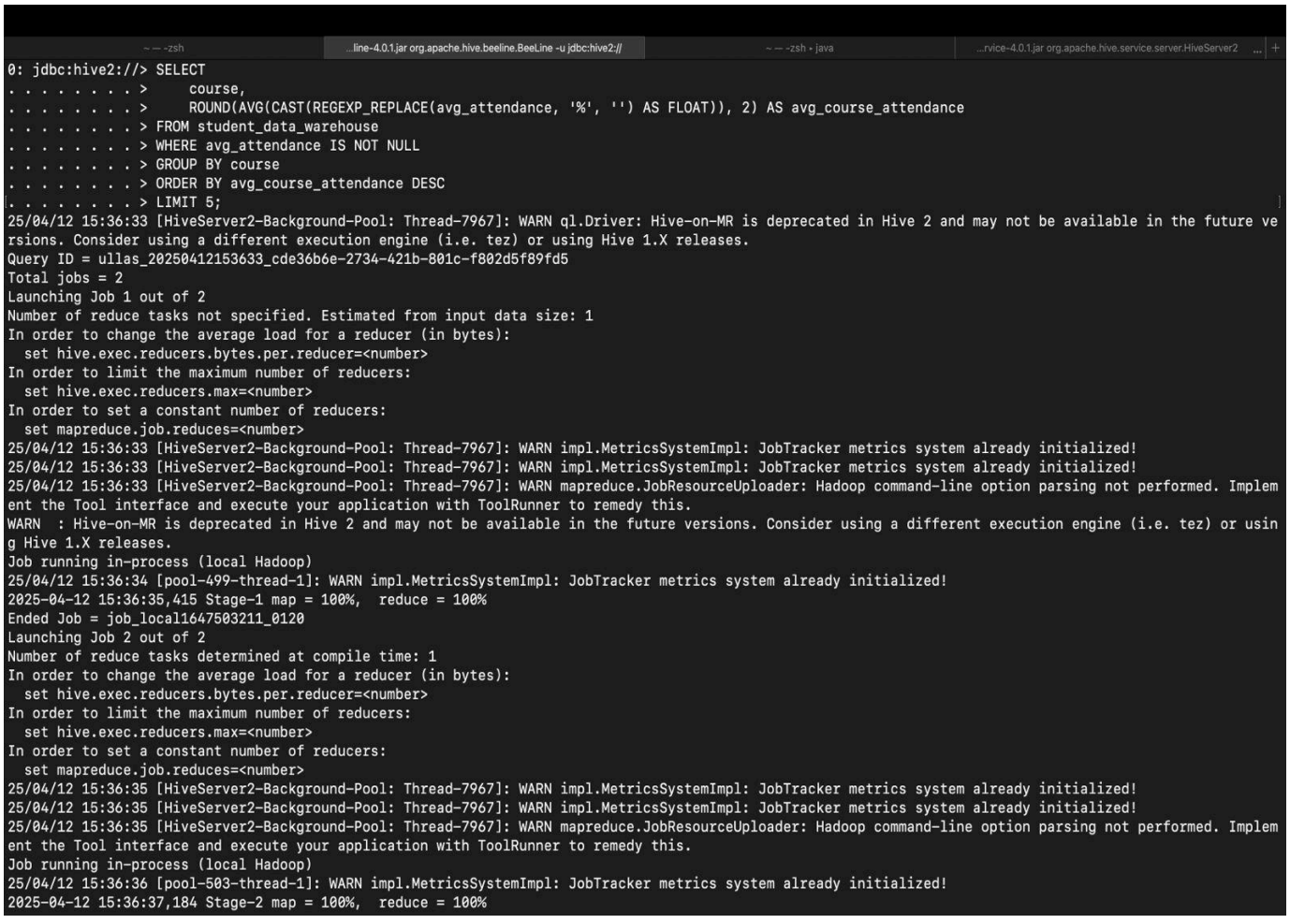
GROUP BY clause groups the joined data on all the columns which we needed in our end warehouse.
Where there are several duplicate or very close duplicate rows for the same (student_id, course), they are merged so that only one record per distinct group is inserted.
This efficiently reduces duplicate rows to a representative row by default, hence maintaining data integrity and uniqueness in the final warehouse.
By grouping on all the selected fields, we ensured that only clean and unique records were being inserted into the student_data_warehouse table. This helped ensure that all the subsequent queries were more accurate and the dataset much more reliable.

Queries on the Data Warehouse

With the clean warehouse in place, we then performed some analytical queries to extract insights. These queries were written and executed from the ‘Q2_Data_Warehouse_Creation.sql’ file.

1. Top 5 Courses with Highest Average Attendance

This query computes the average attendance percentage for each course and lists the top 5 with the highest attendance. It helps identify which courses students are most consistently attending, potentially indicating better engagement or teaching practices. While running the first query, we initially faced an issue where the AVG() function returned all NULL values. This happened because the avg_attendance column contained percentage symbols (e.g., "87%"), making it a string instead of a numeric value. As a result, Hive couldn't perform the aggregation. To fix this, we used the REGEXP_REPLACE() function to remove the '%' symbol and then cast the cleaned string to FLOAT. This allowed us to correctly calculate the average attendance per course and get the desired results.



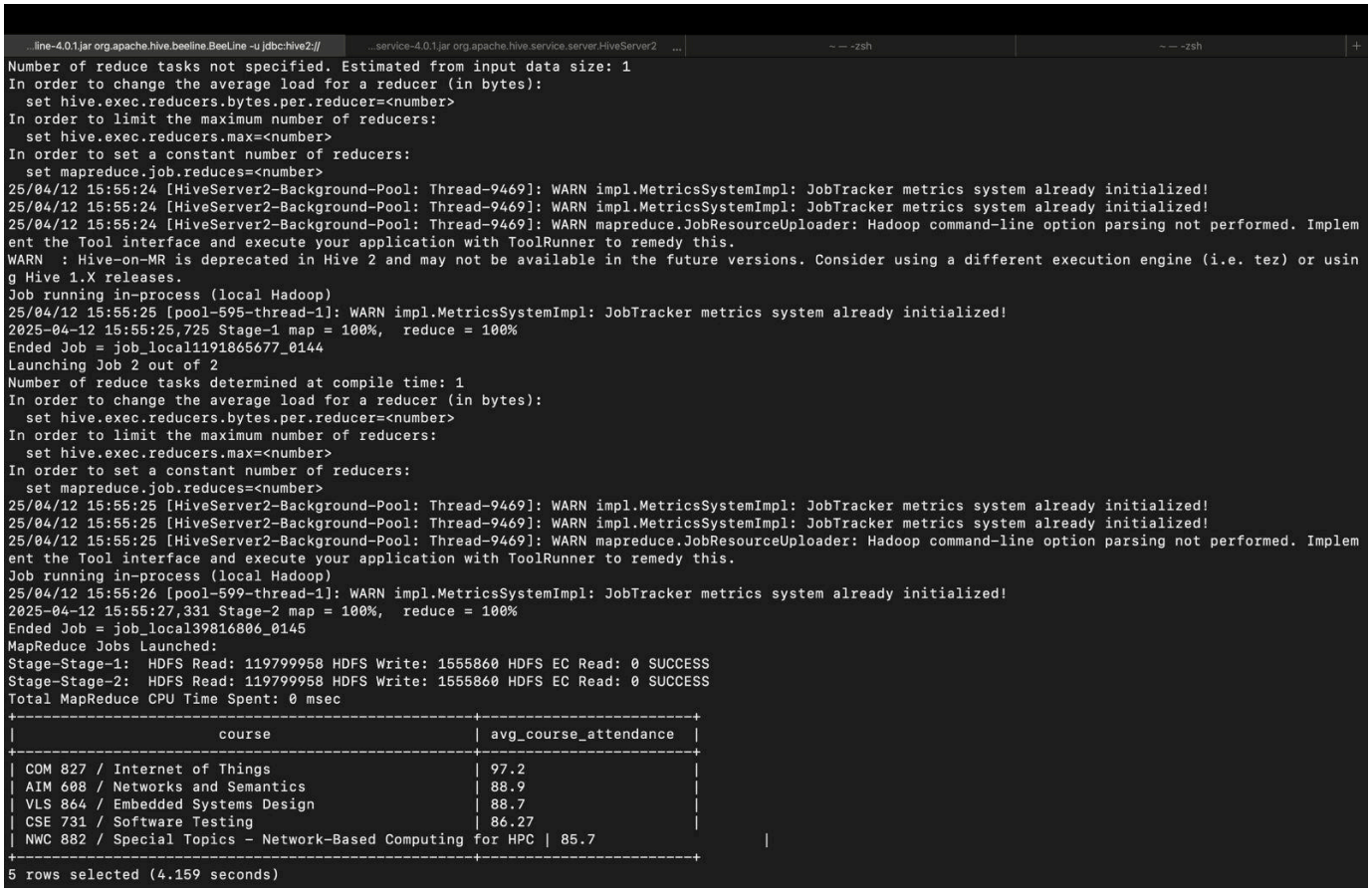


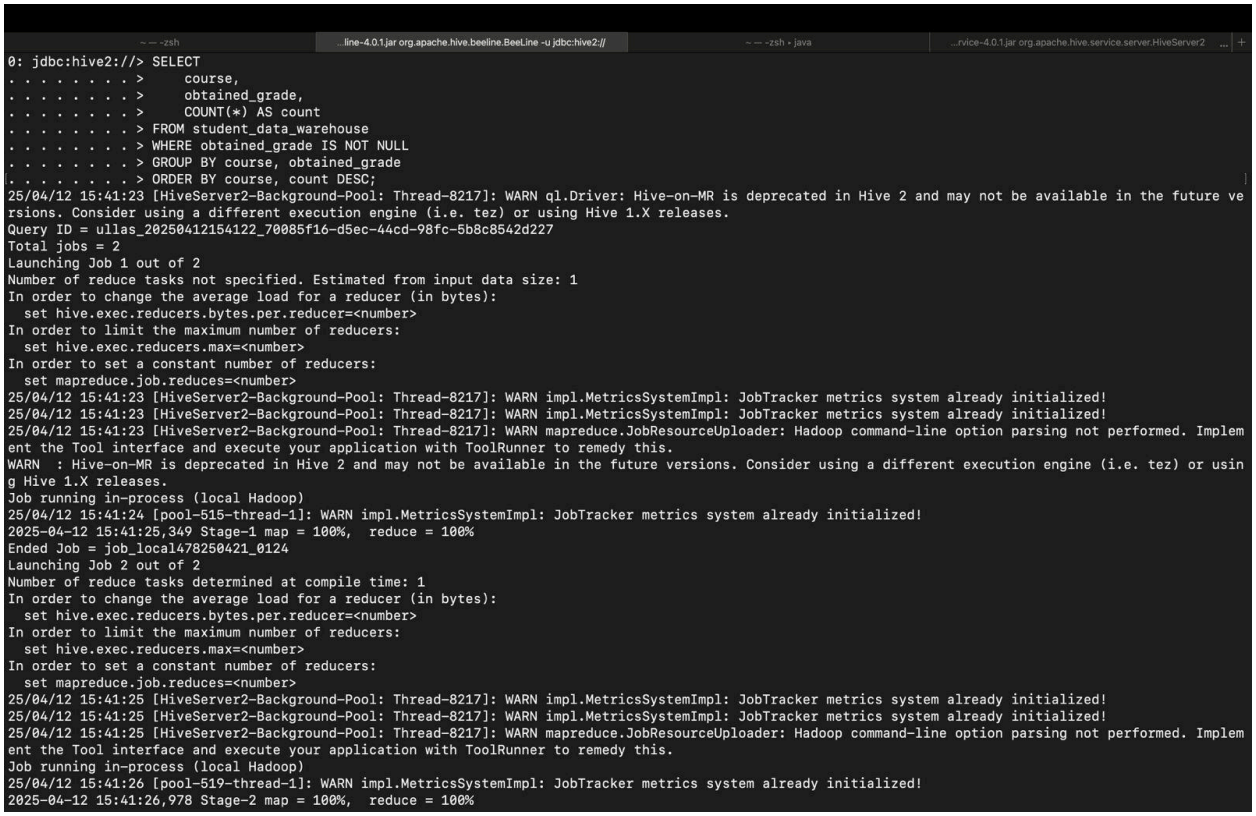
Figure 1 and Figure 2 are output of Query 1 for question 2

2.Grade Distribution per Course

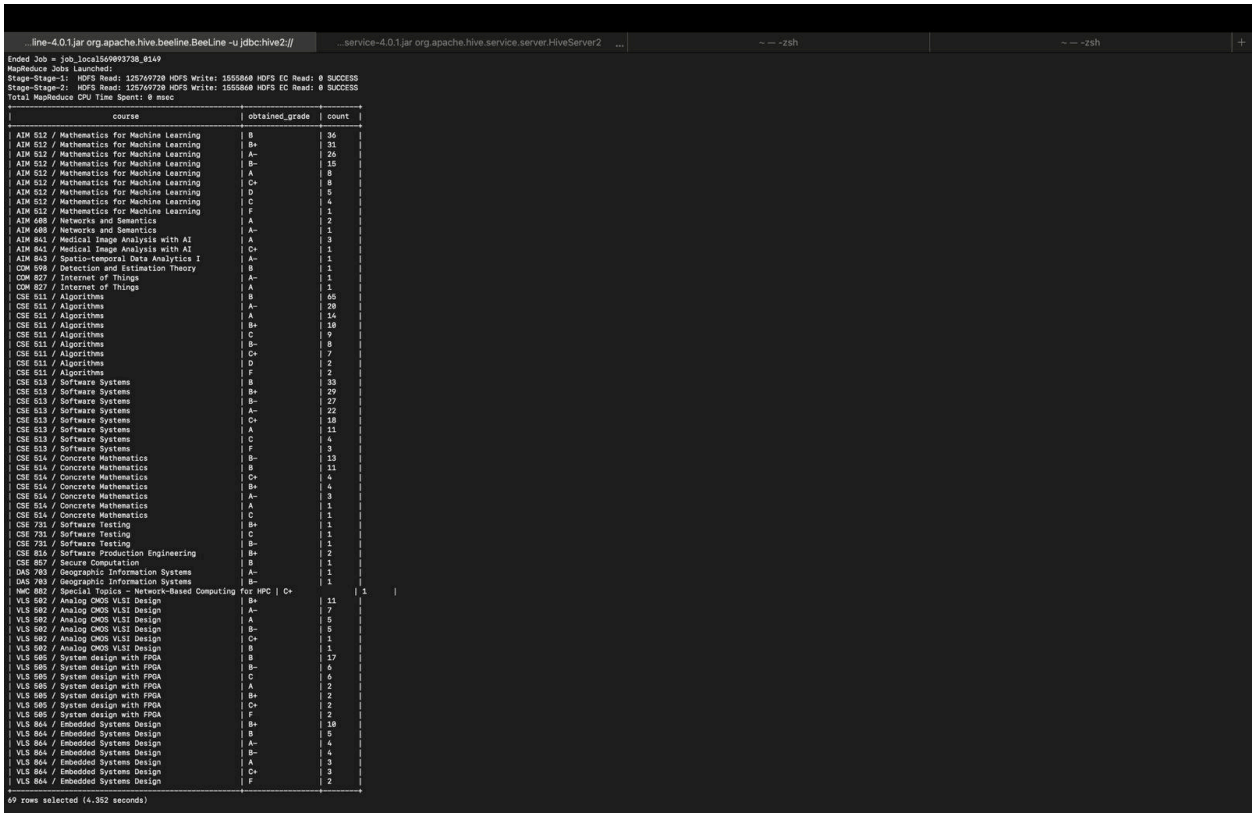
This query shows how grades are distributed across each course. It groups by course and obtained_grade to count how many students received each grade. This gives insights into student performance trends and helps assess course difficulty or grading fairess.

3.Most Common Courses per Program

This query highlights which courses are most frequently taken by students in each academic program. It counts enrollments grouped by program and course. This shows the most preferred or common courses in each academic program, which can be helpful for curriculum planning, resource allocation, or understanding trends in student interests.

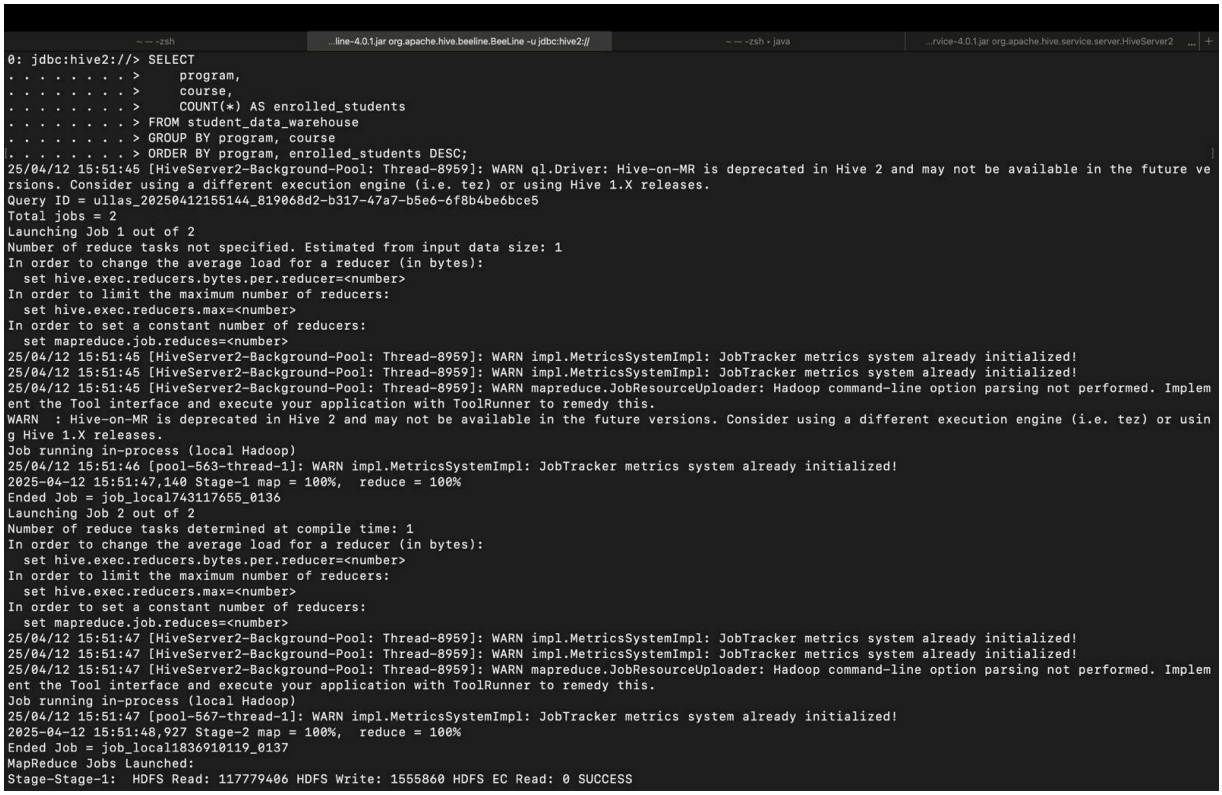


```
~ — -zsh   ..line-4.0.1.jar org.apache.hive.beeline.BeeLine -u jdbc:hive2://   ~ — -zsh ▸ java   ...rvice-4.0.1.jar org.apache.hive.service.server.HiveServer2

0: jdbc:hive2://> SELECT
. . . . . .>     course,
. . . . . .>     obtained_grade,
. . . . . .>     COUNT(*) AS count
. . . . . .> FROM student_data_warehouse
. . . . . .> WHERE obtained_grade IS NOT NULL
. . . . . .> GROUP BY course, obtained_grade
. . . . . .> ORDER BY course, count DESC;
25/04/12 15:41:23 [HiveServer2-Background-Pool: Thread-8217]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ve
rsions. Consider using a different execution engine (i.e. tez) or using Hive 1.X releases.
Query ID = ullas_20250412154122_70085f16-d5ec-44cd-98fc-5b8c8542d227
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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.job.reduces=<number>
25/04/12 15:41:23 [HiveServer2-Background-Pool: Thread-8217]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:41:23 [HiveServer2-Background-Pool: Thread-8217]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:41:23 [HiveServer2-Background-Pool: Thread-8217]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
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. tez) or usin
g Hive 1.X releases.
Job running in-process (local Hadoop)
25/04/12 15:41:24 [pool-515-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-04-12 15:41:25,349 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local478250421_0124
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.job.reduces=<number>
25/04/12 15:41:25 [HiveServer2-Background-Pool: Thread-8217]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:41:25 [HiveServer2-Background-Pool: Thread-8217]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:41:25 [HiveServer2-Background-Pool: Thread-8217]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
25/04/12 15:41:26 [pool-519-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-04-12 15:41:26,978 Stage-2 map = 100%,  reduce = 100%
```

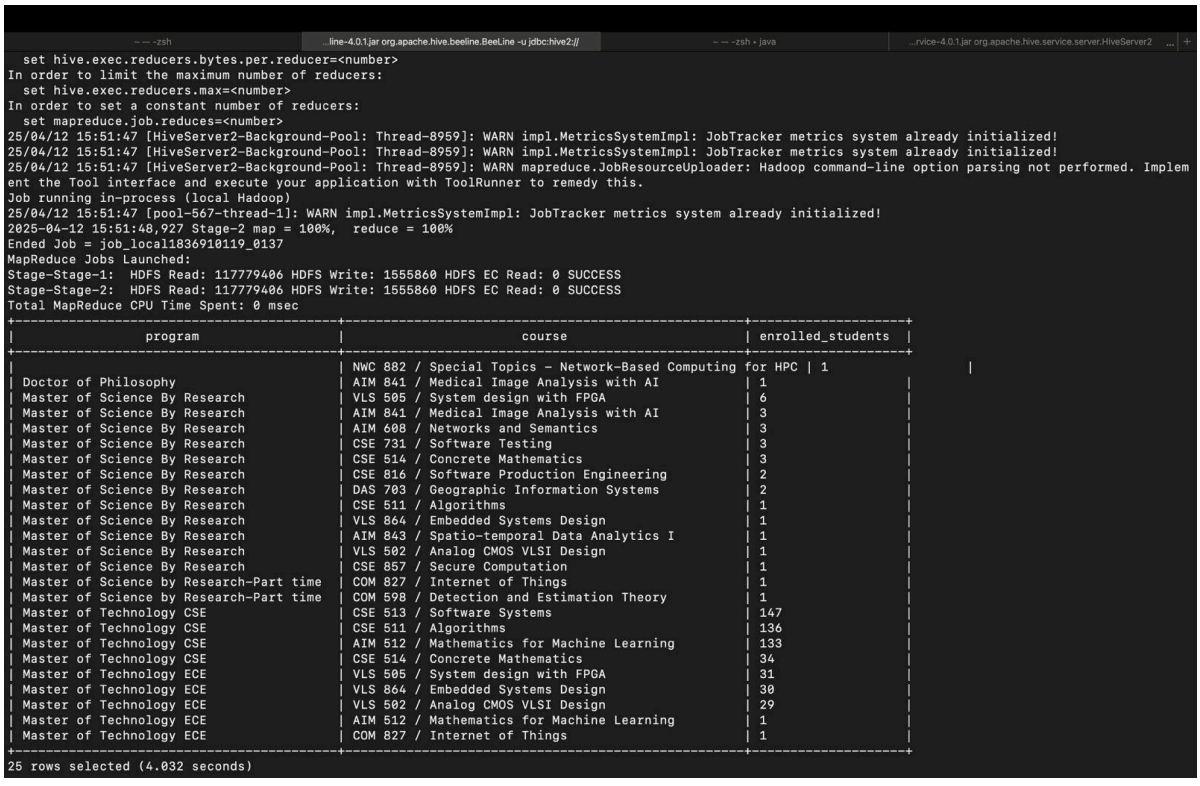


```
...line-4.0.1.jar org.apache.hive.beeline.BeeLine -u jdbc:hive2://   ...service-4.0.1.jar org.apache.hive.service.server.HiveServer2   ~ — -zsh   ~ — -zsh

Ended Job = job_local1569093738_0149
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 125769720 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
Stage-Stage-2:  HDFS Read: 125769720 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
+-----------------------------------------------------------------+-----------------+--------+
|                             course                              | obtained_grade  | count  |
+-----------------------------------------------------------------+-----------------+--------+
| AIM 512 / Mathematics for Machine Learning                      | B               | 36     |
| AIM 512 / Mathematics for Machine Learning                      | B+              | 31     |
| AIM 512 / Mathematics for Machine Learning                      | A-              | 26     |
| AIM 512 / Mathematics for Machine Learning                      | B-              | 15     |
| AIM 512 / Mathematics for Machine Learning                      | A               | 8      |
| AIM 512 / Mathematics for Machine Learning                      | C+              | 8      |
| AIM 512 / Mathematics for Machine Learning                      | D               | 5      |
| AIM 512 / Mathematics for Machine Learning                      | C               | 4      |
| AIM 512 / Mathematics for Machine Learning                      | F               | 1      |
| AIM 608 / Networks and Semantics                                | A               | 2      |
| AIM 608 / Networks and Semantics                                | A-              | 1      |
| AIM 841 / Medical Image Analysis with AI                        | A               | 3      |
| AIM 845 / Medical Image Analysis with AI                        | C+              | 1      |
| AIM 843 / Spatio-temporal Data Analytics I                      | A-              | 1      |
| COM 598 / Detection and Estimation Theory                       | B               | 1      |
| COM 827 / Internet of Things                                    | A-              | 1      |
| COM 827 / Internet of Things                                    | A               | 1      |
| CSE 511 / Algorithms                                            | B               | 66     |
| CSE 511 / Algorithms                                            | A-              | 20     |
| CSE 511 / Algorithms                                            | A               | 14     |
| CSE 511 / Algorithms                                            | B+              | 10     |
| CSE 511 / Algorithms                                            | C               | 9      |
| CSE 511 / Algorithms                                            | B-              | 8      |
| CSE 511 / Algorithms                                            | C+              | 7      |
| CSE 511 / Algorithms                                            | D               | 2      |
| CSE 511 / Algorithms                                            | F               | 2      |
| CSE 513 / Software Systems                                      | B               | 33     |
| CSE 513 / Software Systems                                      | B+              | 29     |
| CSE 513 / Software Systems                                      | B-              | 27     |
| CSE 513 / Software Systems                                      | A-              | 22     |
| CSE 513 / Software Systems                                      | C+              | 18     |
| CSE 513 / Software Systems                                      | A               | 11     |
| CSE 513 / Software Systems                                      | C               | 4      |
| CSE 513 / Software Systems                                      | F               | 3      |
| CSE 514 / Concrete Mathematics                                  | B-              | 13     |
| CSE 514 / Concrete Mathematics                                  | B               | 11     |
| CSE 514 / Concrete Mathematics                                  | C+              | 4      |
| CSE 514 / Concrete Mathematics                                  | B+              | 4      |
| CSE 514 / Concrete Mathematics                                  | A-              | 3      |
| CSE 514 / Concrete Mathematics                                  | A               | 1      |
| CSE 514 / Concrete Mathematics                                  | C               | 1      |
| CSE 731 / Software Testing                                      | B+              | 1      |
| CSE 731 / Software Testing                                      | C               | 1      |
| CSE 731 / Software Testing                                      | B-              | 1      |
| CSE 816 / Software Production Engineering                       | B+              | 2      |
| CSE 857 / Secure Computation                                    | B               | 1      |
| DAS 703 / Geographic Information Systems                        | A-              | 1      |
| DAS 703 / Geographic Information Systems                        | B-              | 1      |
| NWC 882 / Special Topics - Network-Based Computing for HPC | C+              | 1      |
| VLS 502 / Analog CMOS VLSI Design                               | B+              | 11     |
| VLS 502 / Analog CMOS VLSI Design                               | A-              | 7      |
| VLS 502 / Analog CMOS VLSI Design                               | A               | 5      |
| VLS 502 / Analog CMOS VLSI Design                               | B-              | 5      |
| VLS 502 / Analog CMOS VLSI Design                               | C+              | 1      |
| VLS 502 / Analog CMOS VLSI Design                               | B               | 1      |
| VLS 505 / System design with FPGA                               | B               | 17     |
| VLS 505 / System design with FPGA                               | B-              | 6      |
| VLS 505 / System design with FPGA                               | C               | 6      |
| VLS 505 / System design with FPGA                               | A               | 2      |
| VLS 505 / System design with FPGA                               | B+              | 2      |
| VLS 505 / System design with FPGA                               | C+              | 2      |
| VLS 505 / System design with FPGA                               | F               | 2      |
| VLS 864 / Embedded Systems Design                               | B+              | 10     |
| VLS 864 / Embedded Systems Design                               | B               | 5      |
| VLS 864 / Embedded Systems Design                               | A-              | 4      |
| VLS 864 / Embedded Systems Design                               | B-              | 4      |
| VLS 864 / Embedded Systems Design                               | A               | 3      |
| VLS 864 / Embedded Systems Design                               | C+              | 3      |
| VLS 864 / Embedded Systems Design                               | F               | 2      |
+-----------------------------------------------------------------+-----------------+--------+
69 rows selected (4.352 seconds)
```

Figure 3 and Figure 4 are output of Query 2 for question 2



```
0: jdbc:hive2://> SELECT
. . . . . .>     program,
. . . . . .>     course,
. . . . . .>     COUNT(*) AS enrolled_students
. . . . . .> FROM student_data_warehouse
. . . . . .> GROUP BY program, course
. . . . . .> ORDER BY program, enrolled_students DESC;
25/04/12 15:51:45 [HiveServer2-Background-Pool: Thread-8959]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ve
rsions. Consider using a different execution engine (i.e. tez) or using Hive 1.X releases.
Query ID = ullas_20250412155144_819068d2-b317-47a7-b5e6-6f8b4be6bce5
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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.job.reduces=<number>
25/04/12 15:51:45 [HiveServer2-Background-Pool: Thread-8959]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:51:45 [HiveServer2-Background-Pool: Thread-8959]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:51:45 [HiveServer2-Background-Pool: Thread-8959]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
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. tez) or usin
g Hive 1.X releases.
Job running in-process (local Hadoop)
25/04/12 15:51:46 [pool-563-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-04-12 15:51:47,140 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local743117655_0136
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.job.reduces=<number>
25/04/12 15:51:47 [HiveServer2-Background-Pool: Thread-8959]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:51:47 [HiveServer2-Background-Pool: Thread-8959]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:51:47 [HiveServer2-Background-Pool: Thread-8959]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
25/04/12 15:51:47 [pool-567-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-04-12 15:51:48,927 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1836910119_0137
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 117779406 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
```



```
  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.reduces=<number>
25/04/12 15:51:47 [HiveServer2-Background-Pool: Thread-8959]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:51:47 [HiveServer2-Background-Pool: Thread-8959]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:51:47 [HiveServer2-Background-Pool: Thread-8959]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
25/04/12 15:51:47 [pool-567-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-04-12 15:51:48,927 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1836910119_0137
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 117779406 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
Stage-Stage-2:  HDFS Read: 117779406 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
+-----------------------------------------+-----------------------------------------------------+--------------------+
|                 program                 |                       course                        | enrolled_students  |
+-----------------------------------------+-----------------------------------------------------+--------------------+
|                                         | NWC 882 / Special Topics - Network-Based Computing for HPC | 1                  |
| Doctor of Philosophy                    | AIM 841 / Medical Image Analysis with AI            | 1                  |
| Master of Science By Research           | VLS 505 / System design with FPGA                   | 6                  |
| Master of Science By Research           | AIM 841 / Medical Image Analysis with AI            | 3                  |
| Master of Science By Research           | AIM 608 / Networks and Semantics                    | 3                  |
| Master of Science By Research           | CSE 731 / Software Testing                          | 3                  |
| Master of Science By Research           | CSE 514 / Concrete Mathematics                      | 3                  |
| Master of Science By Research           | CSE 816 / Software Production Engineering           | 2                  |
| Master of Science By Research           | DAS 703 / Geographic Information Systems            | 2                  |
| Master of Science By Research           | CSE 511 / Algorithms                                | 1                  |
| Master of Science By Research           | VLS 864 / Embedded Systems Design                   | 1                  |
| Master of Science By Research           | AIM 843 / Spatio-temporal Data Analytics I          | 1                  |
| Master of Science By Research           | VLS 502 / Analog CMOS VLSI Design                   | 1                  |
| Master of Science By Research           | CSE 857 / Secure Computation                        | 1                  |
| Master of Science by Research-Part time | COM 827 / Internet of Things                        | 1                  |
| Master of Science by Research-Part time | COM 598 / Detection and Estimation Theory           | 1                  |
| Master of Technology CSE                | CSE 513 / Software Systems                          | 147                |
| Master of Technology CSE                | CSE 511 / Algorithms                                | 136                |
| Master of Technology CSE                | AIM 512 / Mathematics for Machine Learning          | 133                |
| Master of Technology CSE                | CSE 514 / Concrete Mathematics                      | 34                 |
| Master of Technology ECE                | VLS 505 / System design with FPGA                   | 31                 |
| Master of Technology ECE                | VLS 864 / Embedded Systems Design                   | 30                 |
| Master of Technology ECE                | VLS 502 / Analog CMOS VLSI Design                   | 29                 |
| Master of Technology ECE                | AIM 512 / Mathematics for Machine Learning          | 1                  |
| Master of Technology ECE                | COM 827 / Internet of Things                        | 1                  |
+-----------------------------------------+-----------------------------------------------------+--------------------+
25 rows selected (4.032 seconds)
```

Figure 5 and Figure 6 are output of Query 3 for question 2

Question 3: Creating an Optimized Data Warehouse Using Partitioning and Bucketing

In this part , we created a new Hive table—student_data_warehouse_optimized—to improve query performance using partitioning and bucketing techniques. These methods help Hive reduce the amount of data scanned during queries, leading to faster execution times.

Choosing Columns for Partitioning and Bucketing

Partitioned by: program

Reason:

Low cardinality: There are only 6 distinct programs (e.g., Doctor of Philosophy, M.Tech CSE, M.Tech ECE, etc.), which makes it ideal for partitioning.

Efficient pruning: Partitioning by program allows Hive to scan only the relevant partition directory during queries, avoiding a full table scan.

Bucketed by: course

Reason:

High cardinality: There are 17 distinct courses, making it a good candidate for bucketing.

All queries use GROUP BY or ORDER BY course, so bucketing improves performance by localizing similar data into the same buckets.

Why 8 buckets?

Though there are 17 courses, the number of records per course is highly imbalanced. Bucketing into 17 buckets would overload buckets corresponding to high-enrollment courses and underutilize others.

Using 8 (a power of 2) buckets allows even data distribution and aligns well with distributed file systems.

Loading Data and Running Queries

After creating the table, we loaded data into it from the existing student_data_warehouse (which had no partitions or buckets). We then ran the same three analytical queries as in Question 2 on this optimized version.

All code for this section is present in the file Q3_Bucketing_And_Partitioning.sql.

Runtime Comparison & Performance Observations

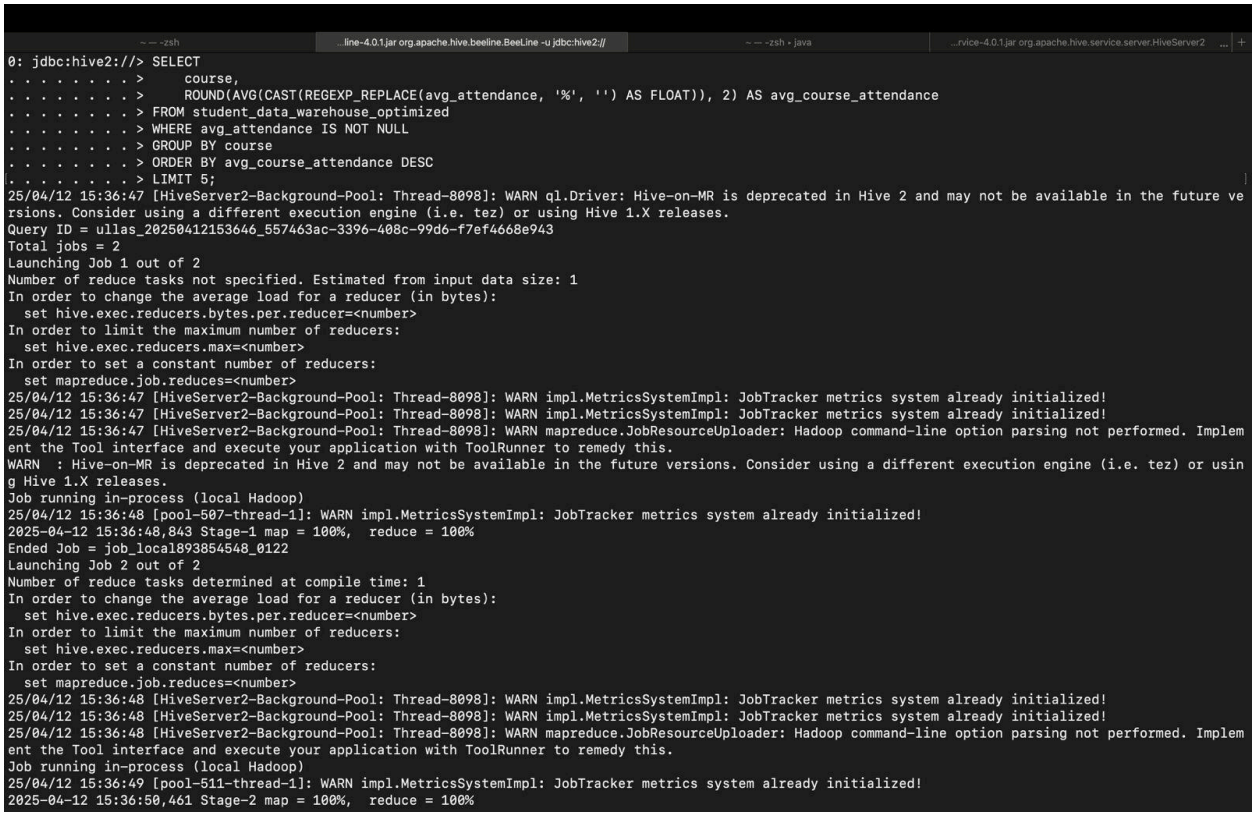
The optimized table contained 573 rows.

Why so few rows?

The low row count is due to inconsistencies in the dataset. For example:

In grade_roster.csv: CS 732 / Data Visualization
In enrollment_data.csv: DAS 732 / Data Visualization
Though both refer to the same course content, the prefix mismatch (CS vs. DAS) and formatting differences prevent Hive from recognizing them as the same value during the join operation.
As a result:
Many course entries failed to match across the two datasets.
Only courses with exactly matching names were successfully joined and included in the final table.
This significantly reduced the number of valid, matched rows—resulting in a final count of only 573 rows.

Effect of optimization:
Since the dataset is small, performance improvement from partitioning and bucketing is minimal.However, for larger datasets, these optimizations would significantly reduce query runtime by narrowing the amount of scanned data.



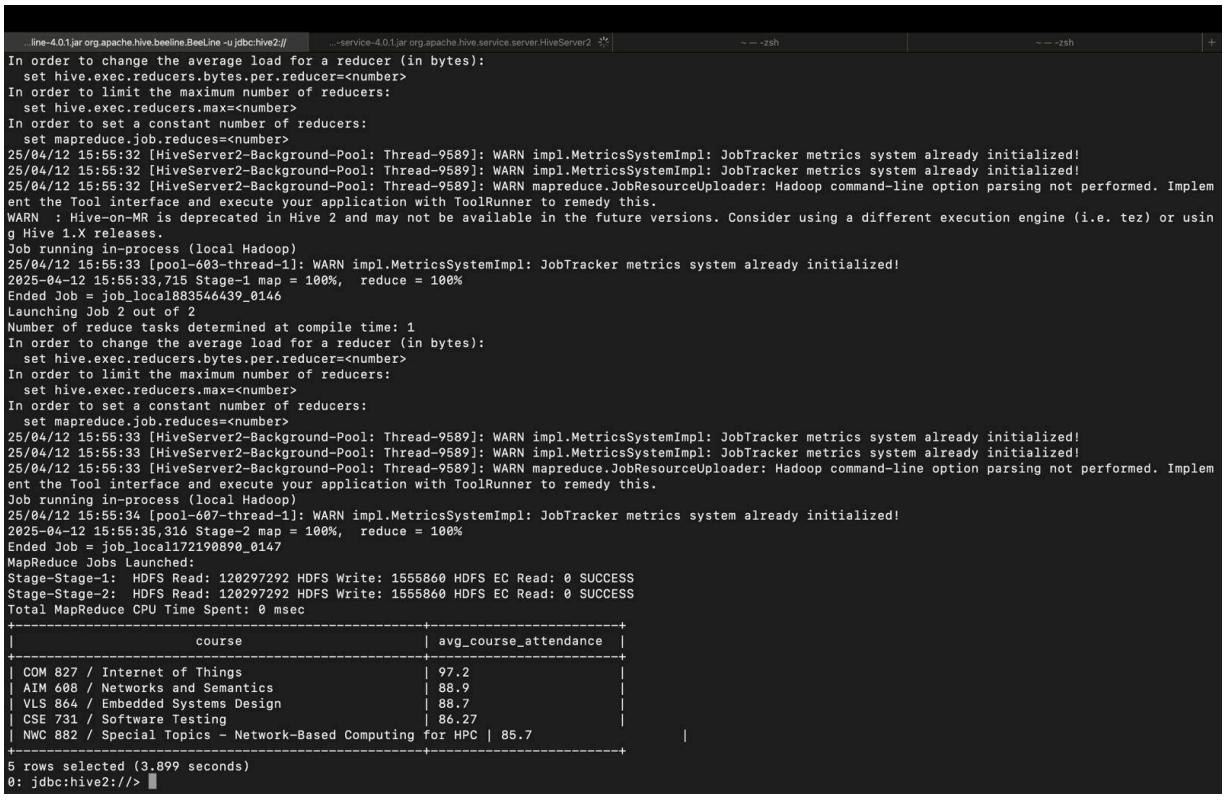
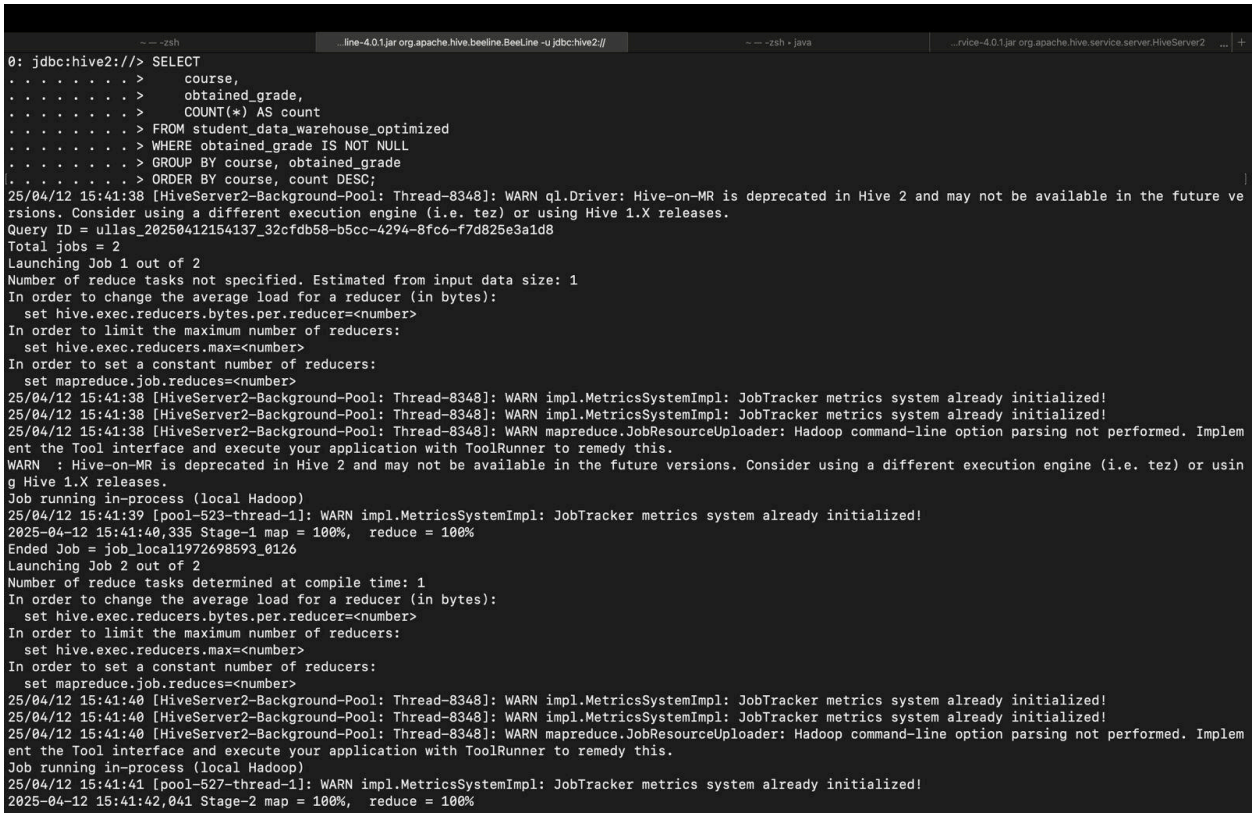


Figure 7 and Figure 8 are output of Query 1 for question 3



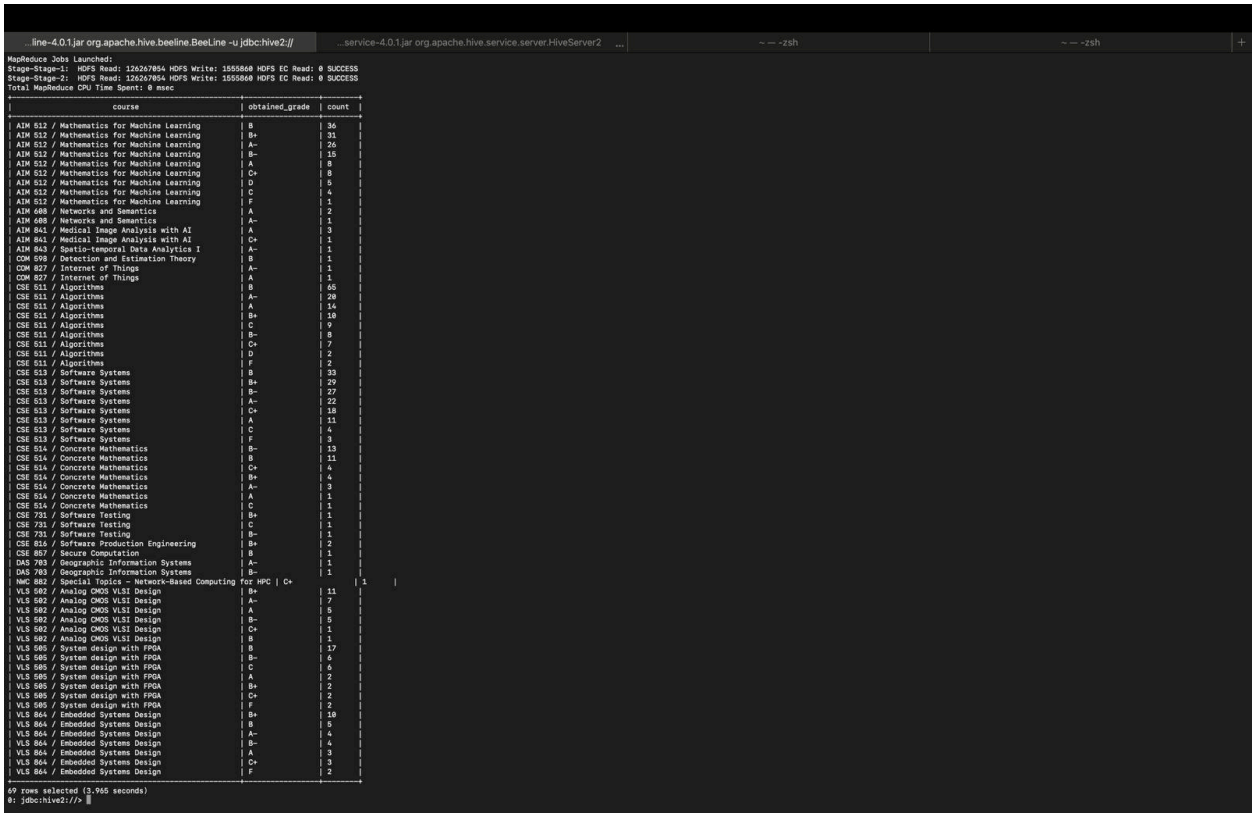


Figure 9 and Figure 10 are output of Query 2 for question 3

```
0: jdbc:hive2://> SELECT
. . . . . >     program,
. . . . . >     course,
. . . . . >     COUNT(*) AS enrolled_students
. . . . . > FROM student_data_warehouse_optimized
. . . . . > GROUP BY program, course
. . . . . > ORDER BY program, enrolled_students DESC;
25/04/12 15:52:30 [HiveServer2-Background-Pool: Thread-9215]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ve
rsions. Consider using a different execution engine (i.e. tez) or using Hive 1.X releases.
Query ID = ullas_20250412155230_4b4f7df8-3cd5-4d98-8c07-aed39703d2d2
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 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.job.reduces=<number>
25/04/12 15:52:30 [HiveServer2-Background-Pool: Thread-9215]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:52:30 [HiveServer2-Background-Pool: Thread-9215]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:52:30 [HiveServer2-Background-Pool: Thread-9215]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
25/04/12 15:52:31 [pool-579-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
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. tez) or usin
g Hive 1.X releases.
2025-04-12 15:52:32,235 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1845936104_0140
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.job.reduces=<number>
25/04/12 15:52:32 [HiveServer2-Background-Pool: Thread-9215]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:52:32 [HiveServer2-Background-Pool: Thread-9215]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:52:32 [HiveServer2-Background-Pool: Thread-9215]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
25/04/12 15:52:32 [pool-583-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-04-12 15:52:33,906 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local152465242_0141
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 118774074 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
```

```
  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.reduces=<number>
25/04/12 15:52:32 [HiveServer2-Background-Pool: Thread-9215]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:52:32 [HiveServer2-Background-Pool: Thread-9215]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
25/04/12 15:52:32 [HiveServer2-Background-Pool: Thread-9215]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implem
ent the Tool interface and execute your application with ToolRunner to remedy this.
Job running in-process (local Hadoop)
25/04/12 15:52:32 [pool-583-thread-1]: WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-04-12 15:52:33,906 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local152465242_0141
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 118774074 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
Stage-Stage-2:  HDFS Read: 118774074 HDFS Write: 1555860 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
+-----------------------------------------+-----------------------------------------------------+--------------------+
|                 program                 |                       course                        | enrolled_students  |
+-----------------------------------------+-----------------------------------------------------+--------------------+
| Doctor of Philosophy                    | AIM 841 / Medical Image Analysis with AI            | 1                  |
| Master of Science By Research           | VLS 505 / System design with FPGA                   | 6                  |
| Master of Science By Research           | AIM 608 / Networks and Semantics                    | 3                  |
| Master of Science By Research           | AIM 841 / Medical Image Analysis with AI            | 3                  |
| Master of Science By Research           | CSE 514 / Concrete Mathematics                      | 3                  |
| Master of Science By Research           | CSE 731 / Software Testing                          | 3                  |
| Master of Science By Research           | CSE 816 / Software Production Engineering           | 2                  |
| Master of Science By Research           | DAS 703 / Geographic Information Systems            | 2                  |
| Master of Science By Research           | VLS 864 / Embedded Systems Design                   | 1                  |
| Master of Science By Research           | VLS 502 / Analog CMOS VLSI Design                   | 1                  |
| Master of Science By Research           | CSE 857 / Secure Computation                        | 1                  |
| Master of Science By Research           | CSE 511 / Algorithms                                | 1                  |
| Master of Science By Research           | AIM 843 / Spatio-temporal Data Analytics I          | 1                  |
| Master of Science by Research-Part time | COM 827 / Internet of Things                        | 1                  |
| Master of Science by Research-Part time | COM 598 / Detection and Estimation Theory           | 1                  |
| Master of Technology CSE                | CSE 513 / Software Systems                          | 147                |
| Master of Technology CSE                | CSE 511 / Algorithms                                | 136                |
| Master of Technology CSE                | AIM 512 / Mathematics for Machine Learning          | 133                |
| Master of Technology CSE                | CSE 514 / Concrete Mathematics                      | 34                 |
| Master of Technology ECE                | VLS 505 / System design with FPGA                   | 31                 |
| Master of Technology ECE                | VLS 864 / Embedded Systems Design                   | 30                 |
| Master of Technology ECE                | VLS 502 / Analog CMOS VLSI Design                   | 29                 |
| Master of Technology ECE                | AIM 512 / Mathematics for Machine Learning          | 1                  |
| Master of Technology ECE                | COM 827 / Internet of Things                        | 1                  |
| __HIVE_DEFAULT_PARTITION__              | NWC 882 / Special Topics - Network-Based Computing for HPC | 1          |
+-----------------------------------------+-----------------------------------------------------+--------------------+
25 rows selected (3.757 seconds)
```

Figure 11 and 12 are output of Query 3 for question 3

Question 4:Exporting Hive Data and Performing Analysis using Pig Latin

Data Export from Hive

The first step involved exporting the data from the Hive data warehouse into a CSV format. This was done using Hive's capability to write query output to a local directory in a delimited format. The schema and data types from the Hive table were preserved in the CSV during this process to ensure compatibility when loading into Pig.

Loading Data into Pig

After exporting the data, the CSV files were loaded into Pig using `PigStorage`, specifying the correct delimiters and manually defining the schema to match the structure of the Hive table. This step ensured that the data format and type information remained consistent, which was essential for accurate analysis in Pig.

Redefining Analytical Queries in Pig

The three analytical queries previously executed using HiveQL were redefined and implemented in Pig Latin:

1. Identifying Top 5 Courses with the Highest Average Attendance:
The attendance values were cleaned, converted to numerical format, and grouped by course to calculate average attendance, followed by sorting to identify the top 5.

```
Successfully stored 5 records in: "file:/tmp/temp-857825041/tmp-630419935"

Counters:
Total records written : 5
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local374928789_0001 ->      job_local1629548799_0002,
job_local1629548799_0002        ->      job_local1300450254_0003,
job_local1300450254_0003        ->      job_local6907524_0004,
job_local6907524_0004


2025-04-12 22:11:00,069 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,070 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,070 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,073 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,074 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,074 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,076 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,077 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,077 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,079 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,079 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,080 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-04-12 22:11:00,081 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD_DISCARDED_TYPE_C
ONVERSION_FAILED 1 time(s).
2025-04-12 22:11:00,081 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2025-04-12 22:11:00,083 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2025-04-12 22:11:00,083 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2025-04-12 22:11:00,083 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(COM 827 / Internet of Things,97.2)
(AIM 608 / Networks and Semantics,88.9)
(VLS 864 / Embedded Systems Design,88.7)
(CSE 731 / Software Testing,86.27)
(NWC 882 / Special Topics - Network-Based Computing for HPC,85.7)
2025-04-12 22:11:00,107 [main] INFO  org.apache.pig.Main - Pig script completed in 2 seconds and 745 milliseconds (2745 ms)
(base) mallikarjun@mallikarjun-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~/Desktop/IIITB course/nosql/assignment3$
```

Figure 13 is output for Query 1 in PIG

```
(CSE 513 / Software Systems,9,9)
(CSE 513 / Software Systems,1,7)
(CSE 513 / Software Systems,13,5)
(CSE 513 / Software Systems,11,4)
(CSE 513 / Software Systems,16,4)
(CSE 513 / Software Systems,12,4)
(CSE 513 / Software Systems,0,3)
(CSE 513 / Software Systems,28,2)
(CSE 513 / Software Systems,15,2)
(CSE 513 / Software Systems,18,2)
(CSE 513 / Software Systems,21,2)
(CSE 513 / Software Systems,19,1)
(CSE 513 / Software Systems,20,1)
(CSE 513 / Software Systems,25,1)
(CSE 513 / Software Systems,27,1)
(CSE 514 / Concrete Mathematics,B-,13)
(CSE 514 / Concrete Mathematics,B,11)
(CSE 514 / Concrete Mathematics,C+,4)
(CSE 514 / Concrete Mathematics,B+,4)
(CSE 514 / Concrete Mathematics,A-,3)
(CSE 514 / Concrete Mathematics,A,1)
(CSE 514 / Concrete Mathematics,C,1)
(CSE 731 / Software Testing,B+,1)
(CSE 731 / Software Testing,B-,1)
(CSE 731 / Software Testing,C,1)
(CSE 816 / Software Production Engineering,B+,2)
(CSE 857 / Secure Computation,B,1)
(DAS 703 / Geographic Information Systems,B-,1)
(DAS 703 / Geographic Information Systems,A-,1)
(NWC 882 / Special Topics - Network-Based Computing for HPC,C+,1)
(VLS 502 / Analog CMOS VLSI Design,B+,11)
(VLS 502 / Analog CMOS VLSI Design,A-,7)
(VLS 502 / Analog CMOS VLSI Design,A,5)
(VLS 502 / Analog CMOS VLSI Design,B-,5)
(VLS 502 / Analog CMOS VLSI Design,C+,1)
(VLS 502 / Analog CMOS VLSI Design,B,1)
(VLS 505 / System design with FPGA,B,17)
(VLS 505 / System design with FPGA,B-,6)
(VLS 505 / System design with FPGA,C,6)
(VLS 505 / System design with FPGA,B+,2)
(VLS 505 / System design with FPGA,A,2)
(VLS 505 / System design with FPGA,F,2)
(VLS 505 / System design with FPGA,C+,2)
(VLS 864 / Embedded Systems Design,B+,10)
(VLS 864 / Embedded Systems Design,B,5)
(VLS 864 / Embedded Systems Design,A-,4)
(VLS 864 / Embedded Systems Design,B-,4)
(VLS 864 / Embedded Systems Design,A,3)
(VLS 864 / Embedded Systems Design,C+,3)
(VLS 864 / Embedded Systems Design,F,2)
(course,obtained_grade,0)
2025-04-13 11:06:58,736 [main] INFO  org.apache.pig.Main - Pig script completed in 2 seconds and 263 milliseconds (2263 ms)
(base) mallikarjun@mallikarjun-HP-Pavilion-x360-2-in-1-Laptop-14-ek0xxx:~/Desktop/IIITB course/nosql/assignment3$
```

Figure 14 is output for Query 2 in PIG

2.Grade Distribution per Course

In Pig, this query was executed by grouping the data based on both `course` and `obtained_grade`, followed by a count to determine how many students received each grade in each course.

3.Most Common Courses per Program

To identify the most common courses in each academic program, Similar to what was done in hive , we grouped the data by `program` and `course`, then counted the number of enrollments in each group.

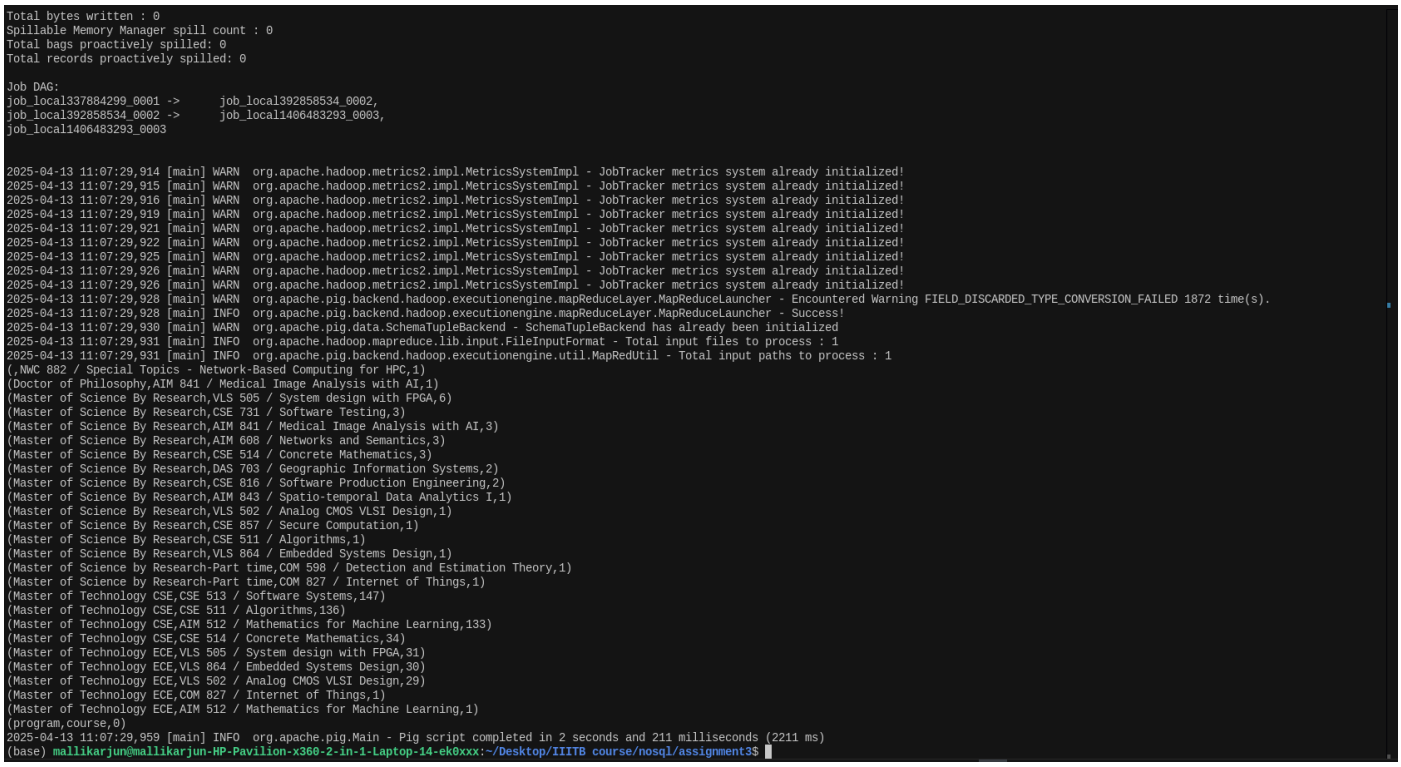


Figure 15 is output for Query 3 in PIG

Runtime Analysis:

	Query 1	Query 2	Query 3
Hive(without Partition)	4.159 seconds	4.352 seconds	4.032 seconds
Hive(with Partition)	3.899 seconds	3.965 seconds	3.757 seconds
Pig	2.745 seconds	2.263 seconds	2.211 seconds

Table 1 - Runtimes of queries

Conclusion:

Based on runtimes alone, Pig is faster than Hive for all 3 queries.Pig queries themselves took about 1–1.5 seconds less than their equivalent HiveQL counterparts even after using Hive optimization techniques like partitioning. What this means is that Pig's data flow-oriented execution model is more suited to simple script-based batch processing workloads.

Pig is particularly well-suited for procedural data manipulation and provides flexibility in expressing complex work flows and thus is well-suited for ETL scenarios. It is less expressive and less natural than SQL, though, and this can be a limitation for those accustomed to traditional query languages.

Hive, on the other hand, allows for a more natural SQL-like syntax that is optimized for data warehousing operations, structured queries, and integration with business intelligence tools. It also supports partitioning, bucketing, and indexing, which simplify scalable querying of large data but at the expense of increased query planning and execution overhead.