Question-3

Now on running the hiveql, we set Hive properties for dynamic bucketing:

1. SET hive.enforce.bucketing=true;
2. SET hive.enforce.sorting=true;

```
0: jdbc:hive2://localhost:10000/> SET hive.enforce.bucketing=true;
No rows affected (0.018 seconds)
0: jdbc:hive2://localhost:10000/> SET hive.enforce.sorting=true;
No rows affected (0.009 seconds)
0: jdbc:hive2://localhost:10000/>
```

Now, the schema for the optimised dimensional and fact tables using data modelling concepts, and the effective concept of how the partitioning and bucketing will increase query performance. The schema for the tables are as follows:-

```
CREATE TABLE IF NOT EXISTS dim_grade_roster_optimised (
    academy_location STRING,
    student_id STRING,
    student_status STRING,
    admission_id STRING,
    admission_status STRING,
    program_name STRING,
    batch STRING,
    period STRING,
    faculty_name STRING,
    course_credit INT,
    obtained_marks_grade STRING,
    out_of_marks_grade STRING,
    exam_result STRING,
    subject_code_name STRING
)
PARTITIONED BY (section STRING)
CLUSTERED BY (student_id) INTO 8 BUCKETS
STORED AS ORC;



CREATE TABLE IF NOT EXISTS dim_attendance_data_optimised (
    instructor STRING,
    name STRING,
    member_id STRING,
    number_of_classes_attended INT,
    number_of_classes_absent INT,
    average_attendance_percent FLOAT
)
PARTITIONED BY (course STRING)
CLUSTERED BY (member_id) INTO 8 BUCKETS
STORED AS ORC;
```

```
CREATE TABLE IF NOT EXISTS fact_table_optimised (
    member_id STRING,
    course STRING,
    number_of_classes_attended INT,
    number_of_classes_absent INT,
    course_credit INT,
    average_attendance_percent FLOAT
)
CLUSTERED BY (member_id) INTO 8 BUCKETS
STORED AS ORC;


CREATE TABLE IF NOT EXISTS dim_enrollment_data_optimised (
  serial_no INT,
  course_type STRING,
  student_id STRING,
  program STRING,
  batch STRING,
  period STRING,
  enrollment_date STRING,
  primary_faculty STRING,
  section STRING
)
PARTITIONED BY (subject_code_name STRING)
CLUSTERED BY (student_id) INTO 8 BUCKETS
STORED AS ORC;
```

You can see that,we have partitioned and bucketed it into optimised tables.
The justification for that is as follows:-

1. `dim_grade_roster_optimised`

- **Partitioned by `section`:**
  - Improves query performance for section-specific queries.
  - Ideal for filtering when analyzing grades per class or section.
- **Clustered by `student_id`:**
  - Boosts performance for student-wise joins (e.g., with attendance or fact table).
  - Enables efficient aggregation operations like CGPA computation.

2. `dim_attendance_data_optimised`

- **Partitioned by `course`:**
  - Optimizes queries analyzing attendance by course.
  - Reduces scan overhead for course-level reports.
- **Clustered by `member_id`:**
  - Enhances performance for per-student attendance analysis.
  - Useful in joins and aggregations involving individual students.

3. **`fact_table_optimised`**

- **No Partitioning:**
  - Acts as a central fact table joined with all dimensions.
  - Uniform query access across multiple attributes, so partitioning might not help.
- **Clustered by `member_id`:**
  - Speeds up joins with `dim_grade_roster_optimised` and `dim_attendance_data_optimised`.
  - Supports student-wise performance analysis (e.g., attendance + credit aggregation).

4. **`dim_enrollment_data_optimised`**

- **Partitioned by `subject_code_name`:**
  - Reduces data scanned for subject-specific queries or filters.
  - Common in analytics related to specific courses.
- **Clustered by `student_id`:**
  - Improves query performance for student-based tracking.
  - Useful for joins with grade and fact tables on `student_id`.

Query-1

Objective:
To compute the CGPA (Cumulative Grade Point Average) for each student based on the grade obtained and course credits.

Approach:

- Join `dim_grade_roster_optimised` and `fact_table_optimised` on `student_id` and `subject_code_name`.
- Use a weighted sum of grade points (based on institutional grading system) multiplied by `course_credit`.
- Divide total weighted grade points by total credits to derive CGPA.
- Order results by CGPA and then by total credits in descending order.
- Since, we have done effective clustering and partitioning, we get the results in much lesser time, greater query performance.

Use Case:
This query is essential for academic performance analysis, ranking students, and eligibility for honors or scholarships.

```
| 2acad14ae711bdd8249e4f49c85fe872cb75313e0dc39c20c760d420a0072243 | 22.0                    | 2.3909090909090907  |
| 6451f1fdbfb51e85c217fc58523ac177fe9c1e59b0dd11cf65d47035fdb720b1 | 28.0                    | 2.3785714285714286  |
| b0026ddcb2635476e72f335b7ded6341ede34eb5f8f3e2ed34aa60062d3934fd | 32.0                    | 2.375               |
| 01021eb63ad8ca36d35a6fd4ead1a931e4dc4b74999a5cf98c7900d8540c97ae | 8.0                     | 2.35                |
| 5db0ea3e96305d5b2434e0a9e3657a76c67509c8327c7e47fd54d1b4b9063f31 | 26.0                    | 2.3307692307692305  |
| 4caf9268ea5658127bf8512445be6922eec2357c8c52b5c5a2c631cae6af0c5d | 22.0                    | 2.290909090909091   |
| 86b2b4629113bb3a78373aa25f95a8dcc6e5676327b2c3e36109872ac9bacc2c | 12.0                    | 2.266666666666667   |
| 447f6ae3c7fd293dabbad856074c77f5ac90b133b9b114cc8080e78770d60882 | 22.0                    | 2.2545454545454544  |
| ad841dec8c5b2f1952dc51dd68adf513a672c4ddeaf228935127ce06c875c20a | 22.0                    | 2.2363636363636363  |
| c4ff6797d1fb4433553653d82b0289c32a3c7837c7012da48ecab2ac9c01ee3a | 32.0                    | 2.1875              |
| 8777036516faed4eabf100af059a4c3e157ceaea6bcdfa28191d593415f64204 | 28.0                    | 2.157142857142857   |
| 46afcb932a74d834d21d0547dee5b8bfefa616f4049b3b77dd47c045c2101cc9 | 32.0                    | 2.1                 |
| cae5018bebe2ccff689e80a84c44d7d3a6acad9949e72e364e03c5e6116b9bed | 32.0                    | 2.1                 |
| 19c5c74b92596d69c552ea81c9aab4370c91a8753e1b64020a7cf16367ea3ea1 | 26.0                    | 2.0615384615384613  |
+-------------------------------------------------------------------+-------------------------+---------------------+
|                          g.student_id                             | total_credits_completed |        cgpa         |
+-------------------------------------------------------------------+-------------------------+---------------------+
| b7f97c4154f2b34129c7f5192e1c40e1436ec1f742924fddb0c19252dc5a15bb | 32.0                    | 2.025               |
| 9886071454e243de3f7fcd672bc754d453b29a5f08e6c6e0df0c5cf8b47f4362 | 4.0                     | 2.0                 |
| 1d26c5cdc02e2b8d9ea7983ca28faf91d58e2755852f9ff5d2321a18e21d3b49 | 4.0                     | 2.0                 |
| 1b5b60677927228f94b20a68dadf069e43e687a6ebeaebb81cff935eeeab4f67 | 28.0                    | 1.807142857142857   |
| 8177fca161b90d83cee14b5e9162f828670d8035a199b48bb5110432b623e9a7 | 22.0                    | 1.736363636363636   |
| d24d0df17e3bbea38e89f2eda4c03d9b30c756f4fa6aeba9ff7e3f8cb7e78bc4 | 26.0                    | 1.6846153846153844  |
| 08aa713e1d2c465191d99525020cf07f773e107a506a44229e7ffb500efd98dd | 26.0                    | 1.6230769230769229  |
| 2b6a85597239bfe683c37419733fb3a9db6d8c4abfd93fa6db9db6f3bca9d493 | 28.0                    | 1.5999999999999999  |
| 99f22d27d8c07ad9de06b443582e2346c5fdcfb9b3b55ab03e562a5c0c4c158a | 28.0                    | 1.4928571428571427  |
| fd9709ae2b08802a0cfc32aa1971dd29c0de7c8b4be3cc07a1cb968fe2405ed5 | 28.0                    | 1.4785714285714284  |
| 882faaed944cce28b59a882b46075b76dad00b0580d3012800a19a25ae9b3221 | 28.0                    | 1.4000000000000001  |
| 3a2e34b3ebca5885323eb7b26d74eab2436cab04d22897c099635d550c9e9201 | 22.0                    | 1.309090909090909   |
| df8239ca5372ac918ccc321ea281f93ca096766e88f3783471b81c7edc720b63 | 28.0                    | 1.307142857142857   |
| 41a9ffb21135b6c0a3007931bdfe1ec944ab4434f1435924a008aa9e3a3ec15f | 26.0                    | 1.2769230769230768  |
| 9bb1f24866e6e0ab55847f85f87829cd4f7be72d1d341c9e514f14602e4e30d4 | 28.0                    | 1.2642857142857142  |
| 075f4288380a972f084731c23f3ae382165107e4c5a2a2cd85363d3a96046fed | 24.0                    | 1.1333333333333333  |
| e03396384ee196698bfc8bc0e21b2af1f2d72950c2d505a6be57261f1a2b6634 | 28.0                    | 1.0357142857142858  |
| 84c39a5f9e43f9068fb7d4de358f0bbf67c90947463da8fe264c40c6895502e6 | 22.0                    | 0.890909090909091   |
| 5e66c5f5200f0426105d3639378ede436e1b0611b183df366fd42b5b4b3e7bac | 28.0                    | 0.5785714285714285  |
| 9582fbc05cf3288085a5d745452cebc2255674776c23c0495cdbd6e852418a02 | 28.0                    | 0.5285714285714286  |
| 7fd5a24c7f7cae6655cc5747682409abc28f5872ffd868bb033310ab07b1fa0c | 24.0                    | 0.0                 |
| 14c1e6f6db35fc08eb0fe6b496924fdb2280c15bb2ab9279e4ca9d4c8d73a4e2 | 20.0                    | 0.0                 |
| 56a2e07bec0c4250925b2bb8579ac06a309404e9d03d911627b986a2f8ad57a7 | 8.0                     | 0.0                 |
| 746fbb665bfd41bf0470020cf596bea17d648b383f88f991408c55c191059b59 | 8.0                     | 0.0                 |
+-------------------------------------------------------------------+-------------------------+---------------------+
524 rows selected (1.043 seconds)
0: jdbc:hive2://localhost:10000/>
```

Time Elapsed:1.643 seconds

Query-2

Objective:
To determine the number of students taught, average attendance, and maximum course credit for each faculty.

Approach:

- Join `dim_grade_roster_optimised` and `fact_table_optimised` on student and course.
- Filter for only those students who have passed (`exam_result = 'Pass'`).
- Aggregate data to:
  - Count distinct students per faculty.
  - Calculate average attendance using `average_attendance_percent`.
  - Determine the highest credit course taught by each faculty.
- Since, we have done effective clustering and partitioning, we get the results in much lesser time, greater query performance.

Use Case:
This helps analyze faculty engagement, workload distribution, and effectiveness in teaching based on student attendance and course difficulty.

Nanditha Rao	42	66.87857142857142	4.0
Pillalamarri Sridhar	160	80.71	4.0
Preeti Mudliar	33	80.2	4.0
Priyanka Das	6	77.18333333333335	4.0
Priyanka Sharma	280	66.44857142857144	2.0
Prof. Amrita Mishra	120	79.95333333333339	4.0
S Malapaka	166	80.00903614457827	4.0
Sachit Rao	150	74.71743119266057	4.0
Sakshi Arora	30	73.76666666666667	4.0
Srinath Srinivasa	3	88.90000000000002	4.0
Srinivas Vivek	198	77.55353535353527	4.0
Sujit Kumar Chakrabarti	160	86.43624999999997	2.0
Sushree Behera	4	81.825	4.0
Thangaraju B	149	92.32364864864864	4.0
Tulika Saha	120	73.93666666666667	2.0
Uttam Kumar	2	28.0	4.0
V Sridhar	313	83.2861271676299	4.0
Vinod Reddy	5	67.03999999999999	4.0
Vinu E V	59	87.05762711864405	4.0
Viswanath G	145	85.38620689655166	4.0

30 rows selected (0.793 seconds)

Time Elapsed:0.793 seconds

Query-3

Objective:
To identify students who have an attendance percentage below 75% in any course.

Approach:

- Join `dim_grade_roster_optimised` and `fact_table_optimised` on `student_id` and `subject_code_name`.
- Calculate overall attendance percentage as (classes_attended / (attended + absent)) * 100:
- Filter (`HAVING`) to return only those records with less than 75% attendance.
- Since, we have done effective clustering and partitioning, we get the results in much lesser time, greater query performance.

Use Case:
Used for academic warnings, eligibility checks for exams, and enforcing minimum attendance policies.

fd9709ae2b08802a0cfc32aa1971dd29c0de7c8b4be3cc07a1cb968fe2405ed5	EGC 112/Programming 1B (Python Programming)	7.0	4.0	63.63636363636363
fdb1bf0b3ff8d8048103388f108794de4164bbe8bdbf7d898a6036965cc2f292	AMS 101/Probability & Statistics	68.0	32.0	68.0
fdb1bf0b3ff8d8048103388f108794de4164bbe8bdbf7d898a6036965cc2f292	AMS 103/Calculus	92.0	40.0	69.6969696969697
fdb1bf0b3ff8d8048103388f108794de4164bbe8bdbf7d898a6036965cc2f292	EGC 102/Digital Design	25.0	9.0	73.52941176470588
fdb1bf0b3ff8d8048103388f108794de4164bbe8bdbf7d898a6036965cc2f292	EGC 112/Programming 1B (Python Programming)	7.0	4.0	63.63636363636363
fdb1bf0b3ff8d8048103388f108794de4164bbe8bdbf7d898a6036965cc2f292	GNL 101/English	7.0	4.0	63.63636363636363
fe6cacdcebbf5892a3583e6ec13530f2e6ea7c6c75a90fcced9a2645e7200033	AMS 101/Probability & Statistics	40.0	28.0	58.8235294117647
fe6cacdcebbf5892a3583e6ec13530f2e6ea7c6c75a90fcced9a2645e7200033	AMS 103/Calculus	48.0	56.0	46.15384615384615
fe6cacdcebbf5892a3583e6ec13530f2e6ea7c6c75a90fcced9a2645e7200033	EGC 102/Digital Design	17.0	10.0	62.96296296296296
fe6cacdcebbf5892a3583e6ec13530f2e6ea7c6c75a90fcced9a2645e7200033	GNL 101/English	0.0	1.0	0.0
fe6cacdcebbf5892a3583e6ec13530f2e6ea7c6c75a90fcced9a2645e7200033	HSS 111/Economics-1	2.0	10.0	16.666666666666668
fedafcd150b9a17932760554a0ec9208266957a49da49214f4f9c7e1776f340d	GNL 101/English	7.0	3.0	70.0
ff6358e8fa8dce631d81990d463738796e3eb5cb545a29edad662cd92864cbfb	VLS 505/System design with FPGA	6.0	3.0	66.66666666666667
ffba274d8a68b64e86980a5d807a0057faa389d2c7a5857424d47dc960e8c434	AIM 511/Machine Learning	0.0	4.0	0.0
ffe3d002fbf6b6c4020303b73c54bcef8c8e9c4b5db7108ac2c8f9b206f0f177	EGC 111/Programming 1A (C Programming)	32.0	28.0	53.333333333333336
ffe3d002fbf6b6c4020303b73c54bcef8c8e9c4b5db7108ac2c8f9b206f0f177	GNL 101/English	7.0	3.0	70.0

850 rows selected (1.018 seconds)
0: jdbc:hive2://localhost:10000/>

Time Elapsed:1.018 seconds

Comparsion With and Without Bucketing

: Comparison of HiveQL Query Execution Time With and Without Bucketing and Partitioning

Query	Without (seconds)	With (seconds)
Query 1	6.54	1.643
Query 2	0.912	0.793
Query 3	1.23	1.018

Here, we can clearly see an increase in the query performance where the elapsed time has decreased by almost 4 times in some cases. But, the create and loading data into the optimised tables takes ample time, due to their internal pre-processing.