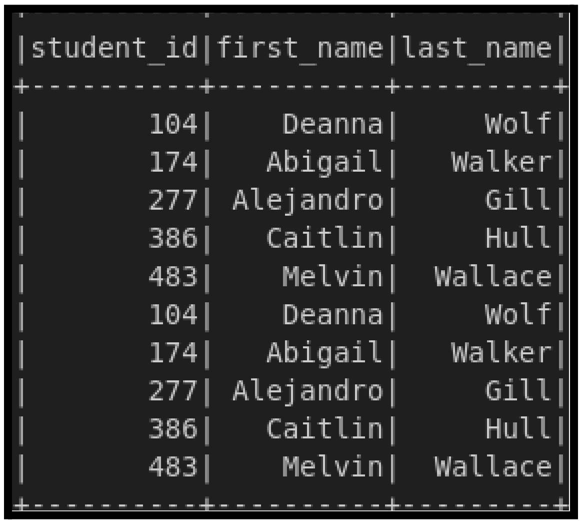
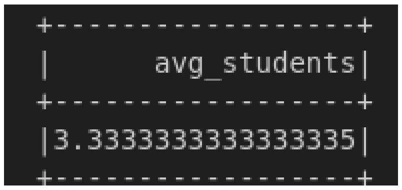
Part 3 and Part 4 Report: Query Implementation and performance optimization

1) Fetching all students enrolled in a specific course.



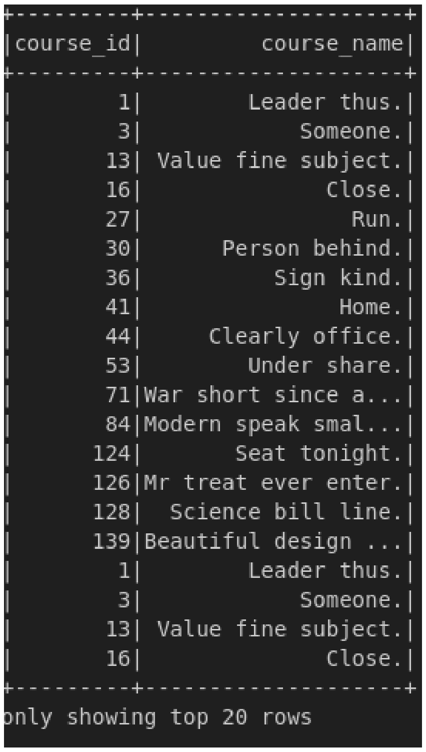
```
|student_id|first_name|last_name|
+----------+----------+---------+
|       104|    Deanna|     Wolf|
|       174|   Abigail|   Walker|
|       277| Alejandro|     Gill|
|       386|   Caitlin|     Hull|
|       483|    Melvin|  Wallace|
|       104|    Deanna|     Wolf|
|       174|   Abigail|   Walker|
|       277| Alejandro|     Gill|
|       386|   Caitlin|     Hull|
|       483|    Melvin|  Wallace|
+----------+----------+---------+
```

2) Calculating the average number of students enrolled in courses offered by a particular instructor at the university.



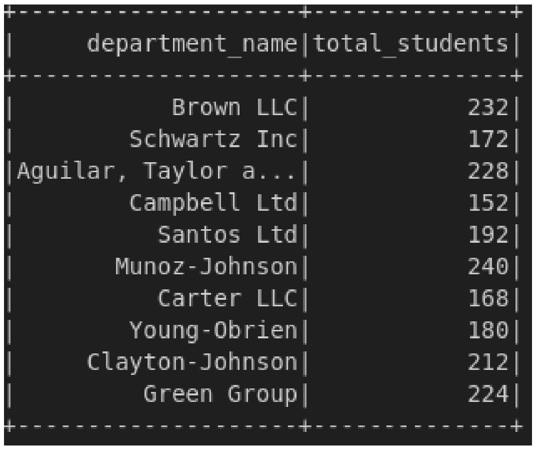
```
+------------------+
|      avg_students|
+------------------+
|3.3333333333333335|
+------------------+
```

3) Listing all courses offered by a specific department.



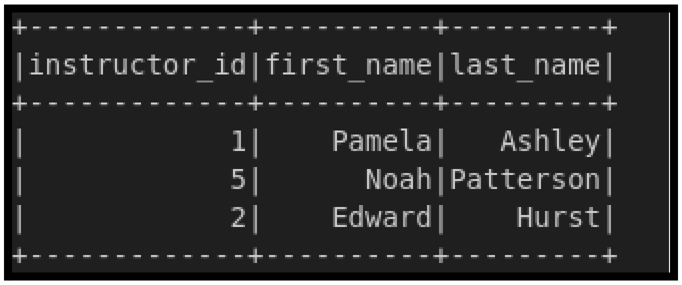
```
+---------+--------------------+
|course_id|         course_name|
+---------+--------------------+
|        1|        Leader thus.|
|        3|            Someone.|
|       13| Value fine subject.|
|       16|              Close.|
|       27|                Run.|
|       30|      Person behind.|
|       36|          Sign kind.|
|       41|               Home.|
|       44|     Clearly office.|
|       53|        Under share.|
|       71|War short since a...|
|       84|Modern speak smal...|
|      124|       Seat tonight.|
|      126|Mr treat ever enter.|
|      128|  Science bill line.|
|      139|Beautiful design ...|
|        1|        Leader thus.|
|        3|            Someone.|
|       13| Value fine subject.|
|       16|              Close.|
+---------+--------------------+
only showing top 20 rows
```

4) Finding the total number of students per department.



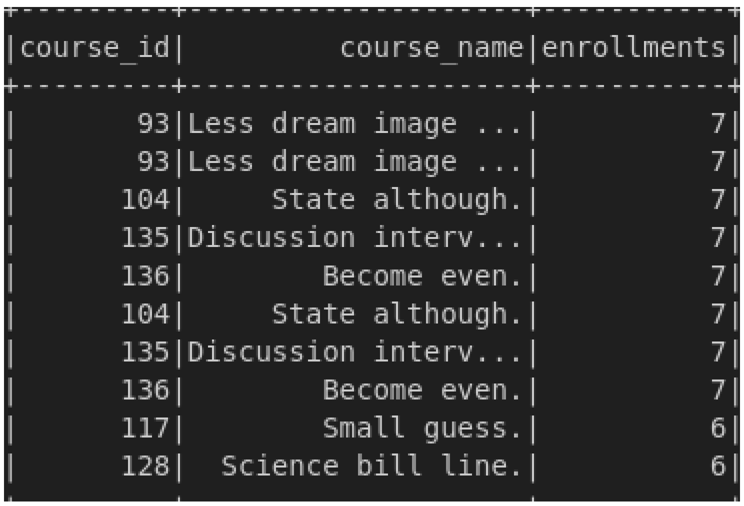
```
+--------------------+--------------+
|     department_name|total_students|
+--------------------+--------------+
|           Brown LLC|           232|
|        Schwartz Inc|           172|
|Aguilar, Taylor a...|           228|
|        Campbell Ltd|           152|
|          Santos Ltd|           192|
|       Munoz-Johnson|           240|
|          Carter LLC|           168|
|        Young-Obrien|           180|
|     Clayton-Johnson|           212|
|         Green Group|           224|
+--------------------+--------------+
```

5) Finding instructors who have taught all the BTech CSE core courses sometime during their tenure at the university.



```
+-------------+----------+---------+
|instructor_id|first_name|last_name|
+-------------+----------+---------+
|            1|    Pamela|   Ashley|
|            5|      Noah|Patterson|
|            2|    Edward|    Hurst|
+-------------+----------+---------+
```

6) Finding top-10 courses with the highest enrollments.



```
+---------+--------------------+-----------+
|course_id|         course_name|enrollments|
+---------+--------------------+-----------+
|       93|Less dream image ...|          7|
|       93|Less dream image ...|          7|
|      104|     State although.|          7|
|      135|Discussion interv...|          7|
|      136|        Become even.|          7|
|      104|     State although.|          7|
|      135|Discussion interv...|          7|
|      136|        Become even.|          7|
|      117|        Small guess.|          6|
|      128|  Science bill line.|          6|
```

1) Indexing Optimisation

- db.students.createIndex({ "student_id": 1 })
- db.courses.createIndex({ "course_id": 1 })
- db.departments.createIndex({ "department_id": 1 })
- db.instructors.createIndex({ "instructor_id": 1 })
- db.courses.createIndex({ "course_id": 1, "instructor_id": 1 })

- Indexes allow MongoDB to quickly locate and access relevant documents without scanning the entire collection. By indexing fields like **student_id, course_id, and instructor_id**, the database can reduce search times and lower input/output operations, leading to faster data retrieval.
- Compound indexes (e.g., on course_id and instructor_id) improve performance for queries involving multiple fields, reducing the need for post-retrieval filtering. This is crucial for maintaining query efficiency as the database grows, ensuring consistent performance in read-heavy applications.

Query Number	**Execution_Time_Without_Optimization**	**Execution_Time_Indexing**
1	0.0745 seconds	0.0672 seconds
2	0.0380 seconds	0.1107 seconds
3	0.0561 seconds	0.0611 seconds
4	0.0493 seconds	0.1130 seconds
5	0.1789 seconds	0.1513 seconds
6	0.0915 seconds	0.0587 seconds

2) Partitioning Optimization

By adjusting the number of partitions, we can ensure that our workload is evenly distributed across the various runners. This helps prevent some runners from getting over burdened than others.

i) On basis of dataframe

- students_df = students_df.repartition(n_paritions)
- departments_df = departments_df.repartition(n_paritions)
- courses_df = courses_df.repartition(n_paritions)
- instructors_df = instructors_df.repartition(n_paritions)

ii) On basis of Column

- students_df = students_df.repartition(n_partitions, "student_id")
- departments_df = departments_df.repartition(n_partitions, "department_id")
- courses_df = courses_df.repartition(n_partitions, "course_id")
- instructors_df = instructors_df.repartition(n_partitions, "instructor_id")

Query Number	**Execution_Time_Without_Optimization**	**Execution_Time_Partitioning_df**	**Execution_Time_Partitioning_column**
1	0.0745 seconds	0.0366 seconds	0.1371 seconds
2	0.0380 seconds	0.0853 seconds	0.1107 seconds
3	0.0561 seconds	0.0219 seconds	0.0526 seconds
4	0.0493 seconds	0.1459 seconds	0.0907 seconds
5	0.1789 seconds	0.1905 seconds	0.1044 seconds
6	0.0915 seconds	0.0398 seconds	0.0772 seconds

3) Partitioning Optimization and Indexing.

Query Number	**Execution_Time_Without_Optimization**	**Execution_Time_Partitioning and_Indexing**
1	0.0745 seconds	0.3370 seconds
2	0.0380 seconds	0.2067 seconds

3	0.0561 seconds	0.0714 seconds
4	0.0493 seconds	0.1489 seconds
5	0.1789 seconds	0.0533 seconds
6	0.0915 seconds	0.0587 seconds

- These optimizations will show a much more significant reduction in execution time when applied to larger datasets or in a distributed computing environment.
- For smaller datasets, the overhead of creating indexes or repartitioning can sometimes result in increased query execution times.