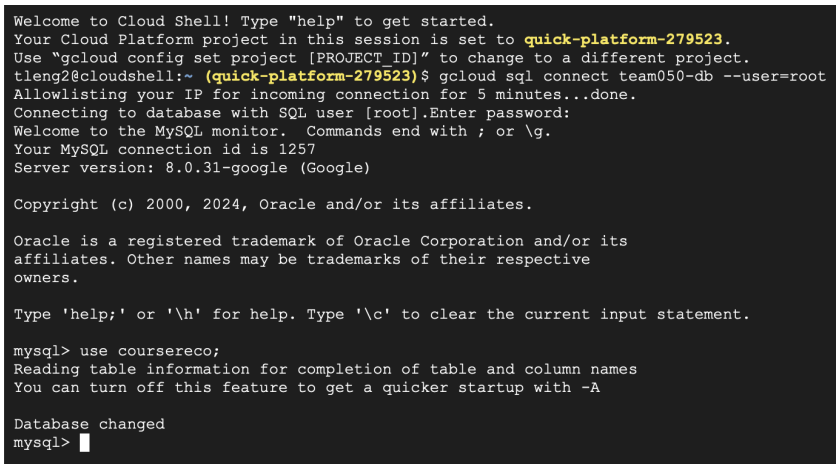
Database Implementation:

Screenshot of GCP Connection:



DDL Commands for Each Table:

Courses:

```
CREATE TABLE Courses (
   Subject VARCHAR(10) NOT NULL,
   CourseNumber INT NOT NULL,
   CourseName VARCHAR(255),
   Credits INT,
   AvgGPA DOUBLE,
   PRIMARY KEY (Subject, CourseNumber)
);
```

```
mysql> SELECT 'Courses' AS TableName, COUNT(*) AS RowCount FROM Courses;
+-----------+----------+
| TableName | RowCount |
+-----------+----------+
| Courses   |      648 |
+-----------+----------+
1 row in set (0.00 sec)
```

Enrollments:

```
CREATE TABLE Enrollments (
   NetID VARCHAR(10),
   Subject VARCHAR(10),
   CourseNumber INT,
   FOREIGN KEY (NetID) REFERENCES User(NetID),
   FOREIGN KEY (Subject, CourseNumber) REFERENCES Courses(Subject, CourseNumber)
);
```

```
mysql> SELECT 'Enrollments', COUNT(*) FROM Enrollments;
+-------------+----------+
| Enrollments | COUNT(*) |
+-------------+----------+
| Enrollments |       18 |
+-------------+----------+
1 row in set (0.01 sec)
```

Major:

```
CREATE TABLE Major (
   MajorName VARCHAR(255) NOT NULL,
   PRIMARY KEY (MajorName)
);
```

```
mysql> SELECT 'Major', COUNT(*) FROM Major;
+-------+----------+
| Major | COUNT(*) |
+-------+----------+
| Major |       19 |
+-------+----------+
1 row in set (0.01 sec)
```

Preference:

```
CREATE TABLE Preference (
   PreferenceID INT NOT NULL,
   GPAWeight INT,
   ProfessorWeight INT,
   CreditsWeight INT,
   OnlineWeight INT,
   NetID VARCHAR(10),
   PRIMARY KEY (PreferenceID),
   FOREIGN KEY (NetID) REFERENCES User(NetID)
);
```

```
mysql> SELECT 'Preference', COUNT(*) FROM Preference;
+------------+----------+
| Preference | COUNT(*) |
+------------+----------+
| Preference |        5 |
+------------+----------+
1 row in set (0.01 sec)
```

Professor:

```
CREATE TABLE Professor (
   ProfName VARCHAR(255) NOT NULL,
   Rating DOUBLE,
   PRIMARY KEY (ProfName)
);
```

```
mysql> SELECT 'Professor', COUNT(*) FROM Professor;
+-----------+----------+
| Professor | COUNT(*) |
+-----------+----------+
| Professor |     1222 |
+-----------+----------+
1 row in set (0.01 sec)
```

Section:

```
CREATE TABLE Section (
   CRN VARCHAR(10) NOT NULL,
   StartTime VARCHAR(50),
   EndTime VARCHAR(50),
   Days VARCHAR(10),
   Online_InPerson VARCHAR(50),
   Term VARCHAR(50),
   AvgGPA INT,
   ProfName VARCHAR(255),
   Subject VARCHAR(10),
   CourseNumber INT,
   PRIMARY KEY (CRN),
   FOREIGN KEY (ProfName) REFERENCES Professor(ProfName),
   FOREIGN KEY (Subject, CourseNumber) REFERENCES Courses(Subject, CourseNumber)
);
```

```
mysql> SELECT 'Section', COUNT(*) FROM Section;
+---------+----------+
| Section | COUNT(*) |
+---------+----------+
| Section |     1738 |
+---------+----------+
1 row in set (0.01 sec)
```

TiedTo:

```
CREATE TABLE TiedTo (
   MajorName VARCHAR(255),
   Subject VARCHAR(10),
```

CourseNumber INT,
FOREIGN KEY (MajorName) REFERENCES Major(MajorName),
FOREIGN KEY (Subject, CourseNumber) REFERENCES Courses(Subject, CourseNumber)
);

```
mysql> SELECT 'TiedTo', COUNT(*) FROM TiedTo;
+--------+----------+
| TiedTo | COUNT(*) |
+--------+----------+
| TiedTo |     4515 |
+--------+----------+
1 row in set (0.01 sec)
```

User:

CREATE TABLE User (
NetID VARCHAR(10) NOT NULL,
Name VARCHAR(255),
Year INT,
CreditHours INT,
PreferenceID INT,
MajorName VARCHAR(255),
PRIMARY KEY (NetID),
FOREIGN KEY (PreferenceID) REFERENCES Preference(PreferenceID),
FOREIGN KEY (MajorName) REFERENCES Major(MajorName)
);

```
mysql> SELECT 'User', COUNT(*) FROM User;
+------+----------+
| User | COUNT(*) |
+------+----------+
| User |        5 |
+------+----------+
1 row in set (0.00 sec)
```

Prerequisite:

CREATE TABLE Prerequisite (
Subject VARCHAR(10) NOT NULL,
CourseNumber INT NOT NULL,
PreSubject VARCHAR(10),
PreCourseNumber INT,
FOREIGN KEY (Subject, CourseNumber) REFERENCES Courses(Subject, CourseNumber),

FOREIGN KEY (PreSubject, PreCourseNumber) REFERENCES Courses(Subject, CourseNumber)
);

```
mysql> SELECT 'Prerequisites', COUNT(*) FROM Prerequisites;
+---------------+----------+
| Prerequisites | COUNT(*) |
+---------------+----------+
| Prerequisites |     1331 |
+---------------+----------+
1 row in set (0.00 sec)
```

Advanced Queries:

Query #1:

Jeremy Lee ranked his preferences as 3 for GPA, 1 for Professor Rating, and 2 for Credits. He is a Computer Engineering Major and has taken several courses.

```
SELECT s.Subject, s.CourseNumber, s.CRN, c.AvgGPA, c.Credits, s.ProfName, p.Rating
FROM Section s
JOIN Courses c ON c.CourseNumber = s.CourseNumber AND c.Subject = s.Subject
JOIN Professor p ON p.ProfName = s.ProfName
JOIN Prerequisites pre ON pre.CourseNumber = s.CourseNumber AND pre.Subject = s.Subject
JOIN TiedTo t ON t.Subject = c.Subject AND t.CourseNumber = s.CourseNumber

WHERE
  t.MajorName = (
    SELECT MajorName
    FROM User
    WHERE NetID = 'jeremy10'
  )
  AND NOT EXISTS (
    SELECT 1
    FROM Prerequisites pre2
    WHERE pre2.Subject = s.Subject
     AND pre2.CourseNumber = s.CourseNumber
     AND NOT EXISTS (
       SELECT 1
       FROM Enrollments e
       WHERE e.NetID = 'jeremy10'
         AND e.Subject = pre2.PreSubject
         AND e.CourseNumber = pre2.PreCourseNumber
```

```
        )
    )
ORDER BY c.AvgGPA DESC, c.Credits DESC, p.Rating DESC;
```

```
mysql> SELECT s.Subject, s.CourseNumber, s.CRN, c.AvgGPA, c.Credits, s.ProfName, p.Rating FROM Section s JOIN Courses c      ON c.CourseNumber = s.CourseNumber      AND c.Subject = s.Subje
ct JOIN Professor p      ON p.ProfName = s.ProfName JOIN Prerequisites pre      ON pre.CourseNumber = s.CourseNumber      AND pre.Subject = s.Subject JOIN TiedTo t      ON t.Subject = c.Su
bject      AND t.CourseNumber = s.CourseNumber WHERE      t.MajorName = (         SELECT MajorName          FROM User          WHERE NetID = 'jeremy10'      )      AND NOT EXISTS (         S
ELECT 1         FROM Prerequisites pre2         WHERE pre2.Subject = s.Subject           AND pre2.CourseNumber = s.CourseNumber           AND NOT EXISTS (             SELECT 1
     FROM Enrollments e             WHERE e.NetID = 'jeremy10'              AND e.Subject = pre2.PreSubject              AND e.CourseNumber = pre2.PreCourseNumber          )      )
 ORDER BY c.AvgGPA DESC, c.Credits DESC, p.Rating DESC limit 15;
+---------+--------------+------+--------+---------+--------------------+--------+
| Subject | CourseNumber | CRN  | AvgGPA | Credits | ProfName           | Rating |
+---------+--------------+------+--------+---------+--------------------+--------+
| CS      |          415 | 76067 |   3.82 |       3 | Eric Shaffer       |    3.6 |
| CS      |          415 | 76059 |   3.82 |       3 | Eric Shaffer       |    3.6 |
| CS      |          415 | 76058 |   3.82 |       3 | Eric Shaffer       |    3.6 |
| CS      |          415 | 74907 |   3.82 |       3 | Eric Shaffer       |    3.6 |
| CS      |          415 | 74906 |   3.82 |       3 | Eric Shaffer       |    3.6 |
| CS      |          467 | 56868 |   3.75 |       3 | Karrie Karahalios  |    3.4 |
| CS      |          467 | 56869 |   3.75 |       3 | Karrie Karahalios  |    3.4 |
| ECE     |          407 | 76034 |    3.7 |       3 | Yupeng Zhang       |      0 |
| CS      |          211 | 71616 |   3.68 |       3 | Ryan Cunningham    |    4.2 |
| CS      |          210 | 31205 |   3.67 |       2 | Ryan Cunningham    |    4.2 |
| CS      |          511 | 31602 |   3.66 |       4 | Daniel Kang        |      0 |
| CS      |          411 | 69361 |   3.45 |       3 | Abdussalam Alawini |    4.2 |
| CS      |          411 | 69362 |   3.45 |       3 | Abdussalam Alawini |    4.2 |
| CS      |          411 | 69453 |   3.45 |       3 | Abdussalam Alawini |    4.2 |
| CS      |          411 | 31355 |   3.45 |       3 | Abdussalam Alawini |    4.2 |
+---------+--------------+------+--------+---------+--------------------+--------+
15 rows in set (0.83 sec)
```

Query #2:

Kyle Smith ranked his preferences as 3 for GPA, 2 for Credits, and 1 for Professor Rating. He is a Computer Engineering major.

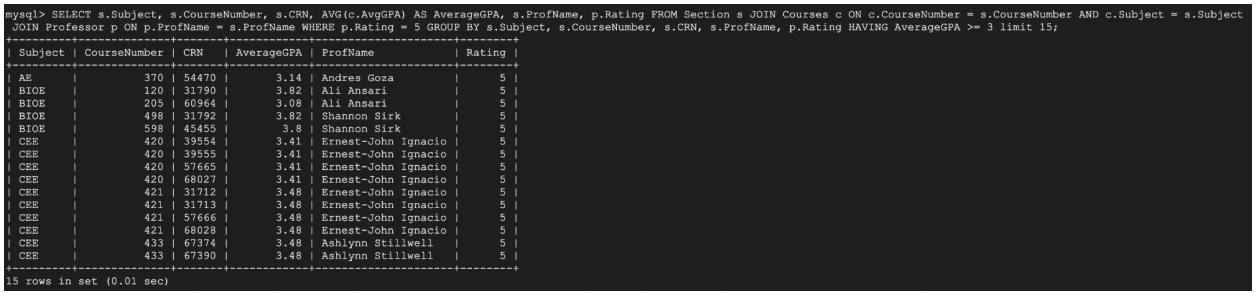
```
SELECT s.Subject, s.CourseNumber, s.CRN, c.AvgGPA, c.Credits, s.ProfName, p.Rating
FROM Section s
JOIN Courses c ON c.CourseNumber = s.CourseNumber AND c.Subject = s.Subject
JOIN Professor p ON p.ProfName = s.ProfName
JOIN Prerequisites pre ON pre.CourseNumber = s.CourseNumber AND pre.Subject = s.Subject
JOIN TiedTo t ON t.Subject = c.Subject AND t.CourseNumber = s.CourseNumber
WHERE t.MajorName = (SELECT MajorName FROM User WHERE NetID = 'kfsmith2')
Group BY s.CRN
ORDER BY c.Credits DESC,c.AvgGPA DESC, p.Rating DESC;
```

```
mysql> SELECT s.Subject, s.CourseNumber, s.CRN, c.AvgGPA, c.Credits, s.ProfName, p.Rating FROM Section s JOIN Courses c ON c.CourseNumber = s.CourseNumber AND c.Subject = s.Subject JOIN Pr
ofessor p ON p.ProfName = s.ProfName JOIN Prerequisites pre ON pre.CourseNumber = s.CourseNumber AND pre.Subject = s.Subject JOIN TiedTo t ON t.Subject = c.Subject AND t.CourseNumber = s.C
ourseNumber WHERE      t.MajorName = (SELECT MajorName FROM User WHERE NetID = 'kfsmith2') Group BY s.CRN ORDER BY c.Credits DESC,c.AvgGPA DESC, p.Rating DESC limit 15;
+---------+--------------+------+--------+---------+--------------------+--------+
| Subject | CourseNumber | CRN  | AvgGPA | Credits | ProfName           | Rating |
+---------+--------------+------+--------+---------+--------------------+--------+
| PHYS    |          102 | 45282 |   2.84 |       5 | Sanjiv Sinha       |    3.1 |
| PHYS    |          102 | 35793 |   2.84 |       5 | Sanjiv Sinha       |    3.1 |
| PHYS    |          102 | 35792 |   2.84 |       5 | Elaine Schulte     |    1.4 |
| PHYS    |          102 | 35775 |   2.84 |       5 | Pengjie Wang       |      0 |
| PHYS    |          101 | 41684 |    2.8 |       5 | Chih-Chung Liu     |    4.2 |
| PHYS    |          101 | 41683 |    2.8 |       5 | Aleksei Aksimentiev |      2 |
| PHYS    |          101 | 35571 |    2.8 |       5 | Katie Ansell       |      0 |
| PHYS    |          101 | 35570 |    2.8 |       5 | Katie Ansell       |      0 |
| MATH    |          220 | 48791 |   2.59 |       5 | Jenny Srikant      |      5 |
| CS      |          544 | 57440 |   3.96 |       4 | David Forsyth      |    4.6 |
| CS      |          568 | 76028 |   3.95 |       4 | Ranjitha Kumar     |    2.3 |
| CS      |          568 | 74383 |   3.95 |       4 | Ranjitha Kumar     |    2.3 |
| PHYS    |          403 | 46820 |   3.86 |       4 | Virginia Lorenz    |      0 |
| CS      |          521 | 76029 |   3.81 |       4 | Sasa Misailovic    |      5 |
| CS      |          521 | 72385 |   3.81 |       4 | Sasa Misailovic    |      5 |
+---------+--------------+------+--------+---------+--------------------+--------+
15 rows in set (0.02 sec)
```

Query #3:

List all the Sections where the average Course GPA is at least a 3.0 and the Professor rating is 5.

```
SELECT s.Subject, s.CourseNumber, s.CRN, AVG(c.AvgGPA) AS AverageGPA, s.ProfName,
p.Rating
FROM Section s
JOIN Courses c ON c.CourseNumber = s.CourseNumber AND c.Subject = s.Subject
JOIN Professor p ON p.ProfName = s.ProfName
WHERE p.Rating = 5
GROUP BY s.Subject, s.CourseNumber, s.CRN, s.ProfName, p.Rating
HAVING AverageGPA >= 3;
```



```
mysql> SELECT s.Subject, s.CourseNumber, s.CRN, AVG(c.AvgGPA) AS AverageGPA, s.ProfName, p.Rating FROM Section s JOIN Courses c ON c.CourseNumber = s.CourseNumber AND c.Subject = s.Subject
 JOIN Professor p ON p.ProfName = s.ProfName WHERE p.Rating = 5 GROUP BY s.Subject, s.CourseNumber, s.CRN, s.ProfName, p.Rating HAVING AverageGPA >= 3 limit 15;
+---------+--------------+-------+------------+--------------------+--------+
| Subject | CourseNumber | CRN   | AverageGPA | ProfName           | Rating |
+---------+--------------+-------+------------+--------------------+--------+
| AE      |          370 | 54470 |       3.14 | Andres Goza        |      5 |
| BIOE    |          120 | 31790 |       3.82 | Ali Ansari         |      5 |
| BIOE    |          205 | 60964 |       3.08 | Ali Ansari         |      5 |
| BIOE    |          498 | 31792 |       3.82 | Shannon Sirk       |      5 |
| BIOE    |          598 | 45455 |        3.8 | Shannon Sirk       |      5 |
| CEE     |          420 | 39554 |       3.41 | Ernest-John Ignacio |      5 |
| CEE     |          420 | 39555 |       3.41 | Ernest-John Ignacio |      5 |
| CEE     |          420 | 57665 |       3.41 | Ernest-John Ignacio |      5 |
| CEE     |          420 | 68027 |       3.41 | Ernest-John Ignacio |      5 |
| CEE     |          421 | 31712 |       3.48 | Ernest-John Ignacio |      5 |
| CEE     |          421 | 31713 |       3.48 | Ernest-John Ignacio |      5 |
| CEE     |          421 | 57666 |       3.48 | Ernest-John Ignacio |      5 |
| CEE     |          421 | 68028 |       3.48 | Ernest-John Ignacio |      5 |
| CEE     |          433 | 67374 |       3.48 | Ashlynn Stillwell   |      5 |
| CEE     |          433 | 67390 |       3.48 | Ashlynn Stillwell   |      5 |
+---------+--------------+-------+------------+--------------------+--------+
15 rows in set (0.01 sec)
```

Query #4:

List all sections for courses with an average GPA of at least 2.5, along with the professor's name, but only for courses that are tied to a major associated with jeremy10. Order by lowest.

```
SELECT s.Subject, s.CourseNumber, s.CRN, s.ProfName, c.CourseName, c.AvgGPA
FROM Section s
JOIN Courses c ON c.CourseNumber = s.CourseNumber AND c.Subject = s.Subject
JOIN TiedTo t ON t.Subject = c.Subject AND t.CourseNumber = c.CourseNumber
WHERE c.AvgGPA >= 2.5
   AND t.MajorName = (
      SELECT MajorName
      FROM User
      WHERE NetID = 'jeremy10'
   )
Order By c.AvgGPA;
```

```
mysql> SELECT s.Subject, s.CourseNumber, s.CRN, s.ProfName, c.CourseName, c.AvgGPA FROM Section s JOIN Courses c ON c.CourseNumber = s.CourseNumber AND c.Subject = s.Subject JOIN TiedTo t
ON t.Subject = c.Subject AND t.CourseNumber = c.CourseNumber WHERE c.AvgGPA >= 2.5     AND t.MajorName = (         SELECT MajorName          FROM User          WHERE NetID = 'jeremy10'
 ) Order By c.AvgGPA limit 15;
+---------+--------------+-------+-------------------+-------------------------------+--------+
| Subject | CourseNumber | CRN   | ProfName          | CourseName                    | AvgGPA |
+---------+--------------+-------+-------------------+-------------------------------+--------+
| MATH    |          112 | 69217 | Theresa  Dobbs    | Algebra                       |   2.57 |
| MATH    |          112 | 70125 | Yupeng Zhang      | Algebra                       |   2.57 |
| MATH    |          112 | 70124 | Yupeng Zhang      | Algebra                       |   2.57 |
| MATH    |          220 | 48791 | Jenny Srikant     | Calculus                      |   2.59 |
| ECE     |          340 | 56894 | Matthew Gilbert   | Semiconductor Electronics     |   2.62 |
| ECE     |          340 | 56895 | Simeon Bogdanov   | Semiconductor Electronics     |   2.62 |
| ECE     |          340 | 56893 | Wei He            | Semiconductor Electronics     |   2.62 |
| ECE     |          211 | 32581 | Juan Alvarez      | Analog Circuits &amp;amp; Systems |   2.65 |
| ECE     |          211 | 32620 | Olga Mironenko    | Analog Circuits &amp;amp; Systems |   2.65 |
| ECE     |          211 | 32660 | Jonathon Schuh    | Analog Circuits &amp;amp; Systems |   2.65 |
| ECE     |          210 | 32505 | Juan Alvarez      | Analog Signal Processing      |   2.69 |
| ECE     |          210 | 32533 | Jonathon Schuh    | Analog Signal Processing      |   2.69 |
| ECE     |          210 | 32532 | Olga Mironenko    | Analog Signal Processing      |   2.69 |
| ECE     |          313 | 52919 | Naresh Shanbhag   | Probability with Engrg Applic |   2.72 |
| ECE     |          313 | 48245 | Dimitrios Katselis | Probability with Engrg Applic |   2.72 |
+---------+--------------+-------+-------------------+-------------------------------+--------+
15 rows in set (0.01 sec)
```

Indexing:

Query #1:

Starting Cost: 241,173

```
| -> Sort: c.AvgGPA DESC, c.Credits DESC, p.Rating DESC  (actual time=1109.964..1109.967 rows=31 loops=1)
    -> Stream results  (cost=241173.71 rows=23785) (actual time=1109.483..1109.943 rows=31 loops=1)
```

Index 1:

CREATE INDEX idx_section_course_subject ON Section (CourseNumber, Subject);

```
| -> Sort: c.AvgGPA DESC, c.Credits DESC, p.Rating DESC  (actual time=919.295..919.298 rows=31 loops=1)
    -> Stream results  (cost=4701.87 rows=3886) (actual time=66.262..919.202 rows=31 loops=1)
```

Index 2:

CREATE INDEX idx_professor_name ON Section (ProfName);

```
| -> Sort: c.AvgGPA DESC, c.Credits DESC, p.Rating DESC  (actual time=910.685..910.687 rows=31 loops=1)
    -> Stream results  (cost=4701.87 rows=3886) (actual time=65.537..910.630 rows=31 loops=1)
```

Index 3:

CREATE INDEX idx_prereq_subj ON Prerequisites (Subject);

```
| -> Sort: c.AvgGPA DESC, c.Credits DESC, p.Rating DESC  (actual time=31.472..31.474 rows=31 loops=1)
    -> Stream results  (cost=1116.22 rows=1375) (actual time=2.053..31.448 rows=31 loops=1)
```

For our first advanced query, we identified key attributes involved in our JOIN, WHERE, and subquery sections and experimented with adding indexes on the attributes that are frequently accessed or used. We first recognized the relationships between Section with Courses and Sections with Prerequisties to be tied by CourseNumber and Subject. Thus, we first indexed Section on (CourseNumber, Subject) to see if it had any effect on optimizing the joins between the Section, Courses, and Prerequisites tables. This brought the cost down from about 241,000 to about 4,700. We then decided to create an index an index on Section using ProfName since we JOIN Professors with Sections. However, since no cost was minimized, we decided not to move forward with using this index. We created an index on Prerequisites using Subject. We thought this would help since Subject is used throughout our WHERE clauses. This minimized the cost

to about 1,116. Since the index of idx_prereq_subj has minimized the cost the most, we decided to keep this as our final index design.

Query #2:

Starting Cost: 5,090

```
| -> Sort: c.Credits DESC, c.AvgGPA DESC, p.Rating DESC  (actual time=10.553..10.585 rows=460 loops=1)
    -> Table scan on <temporary>  (cost=5090.52..5141.58 rows=3886) (actual time=10.363..10.418 rows=460 loops=1)
        -> Temporary table with deduplication  (cost=5090.51..5090.51 rows=3886) (actual time=10.361..10.361 rows=460 loops
```

Index 1:
CREATE INDEX idx_courses_course ON Courses(CourseNumber, Subject);

```
-> Sort: c.Credits DESC, c.AvgGPA DESC, p.Rating DESC  (actual time=9.738..9.770 rows=460 loops=1)
  -> Table scan on <temporary>  (cost=5090.52..5141.58 rows=3886) (actual time=9.547..9.602 rows=460 loops=1)
      -> Temporary table with deduplication  (cost=5090.51..5090.51 rows=3886) (actual time=9.546..9.546 rows=460 loops=1
```

Cost doesn't change, so we don't implement this.

Index 2:
CREATE INDEX idx_prerequisites_course ON Prerequisites(CourseNumber, Subject);

```
| -> Sort: c.Credits DESC, c.AvgGPA DESC, p.Rating DESC  (actual time=6.570..6.603 rows=460 loops=1)
    -> Table scan on <temporary>  (cost=1253.77..1273.44 rows=1375) (actual time=6.381..6.435 rows=460 loops=1)
        -> Temporary table with deduplication  (cost=1253.76..1253.76 rows=1375) (actual time=6.378..6.378 rows=460 loops=1
)
```

Cost greatly decreases, so this index is implemented.

Index 3:
CREATE INDEX idx_courses_credits_avgGPA ON Courses(Credits, AvgGPA);

```
| -> Sort: c.Credits DESC, c.AvgGPA DESC, p.Rating DESC  (actual time=6.551..6.583 rows=460 loops=1)
    -> Table scan on <temporary>  (cost=1253.77..1273.44 rows=1375) (actual time=6.359..6.414 rows=460 loops=1)
        -> Temporary table with deduplication  (cost=1253.76..1253.76 rows=1375) (actual time=6.356..6.356 rows=460 loops=1
)
```

Cost doesn't change, so we don't implement this.

Index 4:

CREATE INDEX idx_section_crn ON Section(CRN);

```
| -> Sort: c.Credits DESC, c.AvgGPA DESC, p.Rating DESC  (actual time=6.551..6.583 rows=460 loops=1)
    -> Table scan on <temporary>  (cost=1253.77..1273.44 rows=1375) (actual time=6.359..6.414 rows=460 loops=1)
        -> Temporary table with deduplication  (cost=1253.76..1253.76 rows=1375) (actual time=6.356..6.356 rows=460 loops=1
)
```

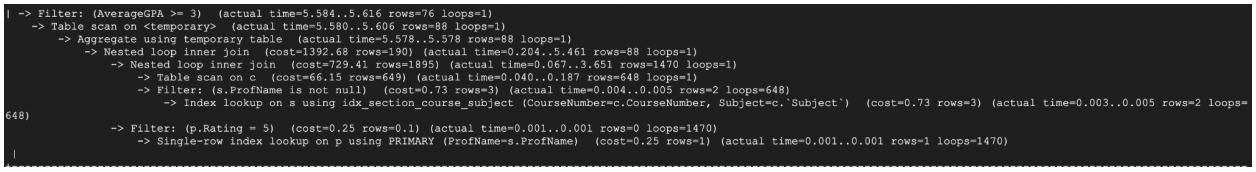
Again cost doesn't change, so we don't implement this

For our second advanced query, we again identified key attributes involved in our JOIN, WHERE, ORDER BY and GROUP BY sections, and added indexes on the attributes that are frequently accessed or used. First, we attempt to streamline our performance on joins in regards to matching CourseNumber and Subject, however it appears that this is already optimized. We then establish an index for our prerequisites, which dramatically decreases runtime cost due to

removing the need to search through all of the prerequisites. We then try to index course credits to improve sorting, which doesn't appear to decrease cost. We also try to index our section on CRN, which doesn't work since our CRN is unique in itself.

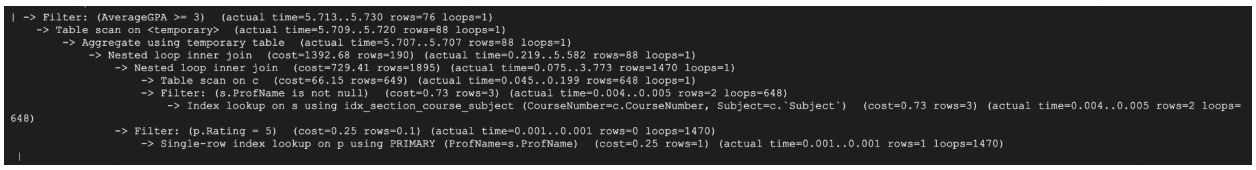
Query #3:

Starting Cost:



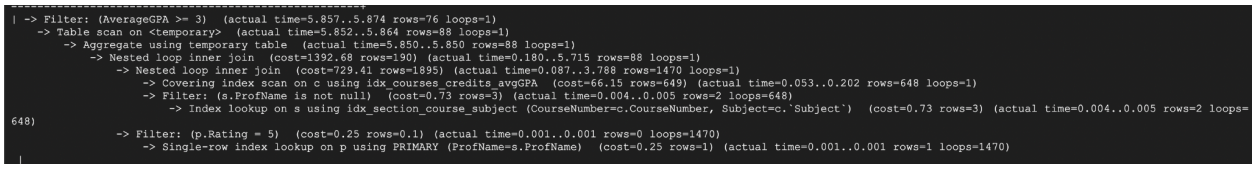
Index 1:

CREATE INDEX idx_courses_subject_course_number ON Courses (Subject, CourseNumber);



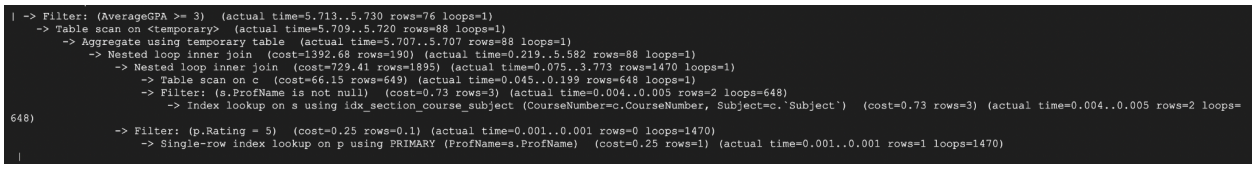
Index 2:

CREATE INDEX idx_section_subject_course_number_crn ON Section (Subject, CourseNumber, CRN);



Index 3:

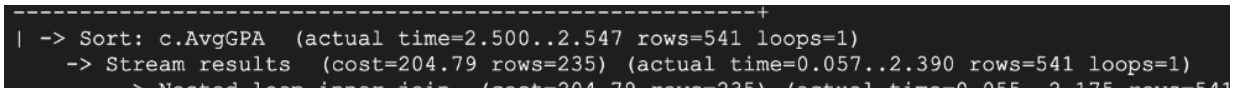
CREATE INDEX idx_professor_rating ON Professor (Rating);



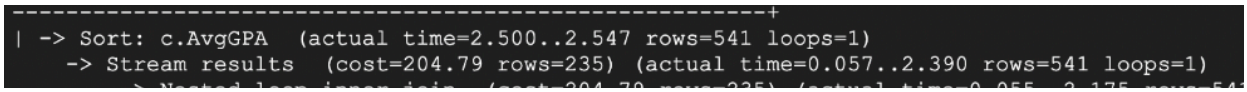
To optimize this query, we decided to try creating indexes that enhance performance for the JOIN conditions, WHERE clause, and GROUP BY clause. First, we tried an index on the Courses table for the JOIN on CourseNumber and Subject as these columns are used to connect

with the Section table. However, after analysis, we saw no reduction in cost. Next, an index on the Section table covering Subject, CourseNumber, and CRN was added to optimize both the JOIN operation and the GROUP BY grouping operations. Similarly, this index had no effect, only increasing the time elapsed. Lastly, because the query applies a filter on Rating in the Professor table, we believed an index on Rating would speed up the filtering process in the WHERE clause. Unfortunately, this gave the same result as the other two queries, failing to improve the cost while increasing the run time. Consequently, we decided that our initial implementation without the indexes provided the best results.

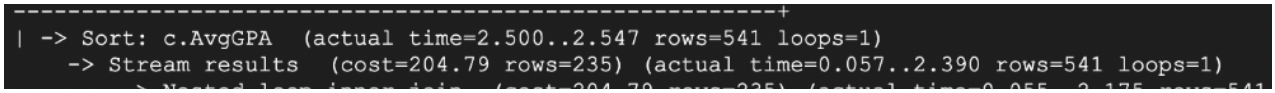
Query #4:
Starting Cost: 204



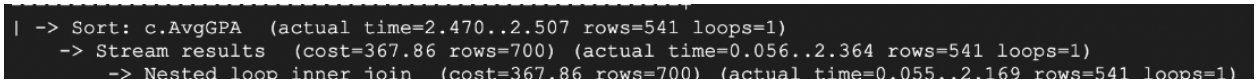
Index 1:
CREATE INDEX idx_courses_subject_course_number_avgGPA ON Courses (Subject, CourseNumber, AvgGPA);



Index 2:
CREATE INDEX idx_tiedto_subject_course_number_majorname ON TiedTo (Subject, CourseNumber, MajorName);



Index 3:
CREATE INDEX idx_courses_avgGPA ON Courses (AvgGPA);



For this query, indexes were created on key columns in the Courses, TiedTo, and AvgGPA fields to support the JOIN operations, filtering conditions, and sorting requirements. We tried indexing the Courses table since these columns are used both in the JOIN with Section and in filtering (WHERE c.AvgGPA >= 2.5), but this did not result in any improved cost. Next, an index on TiedTo for Subject, CourseNumber, and MajorName should theoretically optimize the join with Courses and supports filtering by MajorName based on the user's major. But, similar to the first index, this did not result in an improved cost. Lastly, a separate index on AvgGPA was

created to improve sorting efficiency for the ORDER BY clause. However, this actually resulted in our Nested inner loop join cost increasing from 204.79 to 367.86, so we ended up dropping this index. From all of our trials, we ended up deciding that the default indexing was the most efficient option.