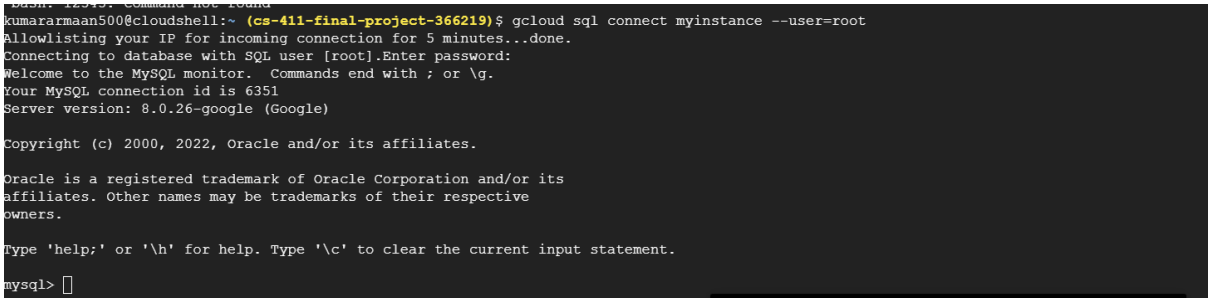
CS 411 Stage 3

GCP Connection :

1. **Python Connection :**

2. **Terminal Info :**



Creating Tables :

1. **Courses Table**

```
CREATE TABLE Courses (
    yearTerm VARCHAR(50),
    subject VARCHAR(50),
    courseNo INT,
    instructor VARCHAR(50),
    avgGPA FLOAT,
```

```
    courseId   VARCHAR(50)         NOT NULL,

    PRIMARY KEY(courseId)
);
```

```
mycursor.execute("SELECT COUNT(courseNo) FROM Courses;")
print_cursor(mycursor)
```
✓ 0.4s Python

```
(1300,)
```

2. **Users Table**

```
CREATE TABLE Users (
    userId         VARCHAR(50) NOT NULL,
    password       VARCHAR(50),
    username       VARCHAR(50),

    PRIMARY KEY(userId)
);
```

```
mycursor.execute("SELECT COUNT(userId) FROM Users;")
print_cursor(mycursor)
```
✓ 0.6s Python

```
(1247,)
```

3. **Friends Table**

```
CREATE TABLE Friends (
    friendId VARCHAR(50) NOT NULL,
    gpaDiff FLOAT,
    userId VARCHAR(50),

    PRIMARY KEY(friendId),
    FOREIGN KEY(userId) REFERENCES Users(userId) ON DELETE CASCADE
);
```

```
mycursor.execute("SELECT COUNT(friendId) FROM Friends;")
print_cursor(mycursor)
```
✓ 0.6s Pytho

```
(1001,)
```

4. **Reviews Table**

```
CREATE TABLE Reviews (
  reviewId VARCHAR(50)  NOT NULL,
  rating  FLOAT,
  body VARCHAR(140),
  courseId VARCHAR(50),

    PRIMARY KEY(reviewId),
  FOREIGN KEY(courseId) REFERENCES Courses(courseId) ON DELETE CASCADE
);
```

```
mycursor.execute("SELECT COUNT(reviewId) FROM Reviews;")
print_cursor(mycursor)
```
✓ 0.5s Pytho

```
(100,)
```

Advanced Subqueries :

1. **Query 1**

```
mycursor.execute("""SELECT DISTINCT Subject, courseNo, ROUND(AVG(avgGPA),3)
FROM Courses
WHERE YearTerm IN (SELECT YearTerm
                   FROM Courses
                   WHERE YearTerm LIKE '%2021%')
GROUP BY Subject, courseNo;""")
print_cursor(mycursor)
```
✓ 0.5s Python

```
Output exceeds the size limit. Open the full output data in a text editor
('AAS', 100, 3.448)
('AAS', 120, 3.43)
('ACCY', 517, 3.825)
('ECON', 102, 3.43)
('ACCY', 551, 3.47)
('ACCY', 556, 3.635)
('ECON', 103, 3.312)
('ACCY', 569, 3.33)
('ACCY', 570, 3.793)
('ECON', 202, 2.825)
('ECON', 203, 2.411)
('ECON', 302, 2.648)
('ECON', 303, 3.007)
('ECON', 418, 2.81)
('ECON', 420, 3.33)
('ECON', 425, 2.9)
('ECON', 426, 3.08)
('ACCY', 578, 3.9)
('ECON', 437, 3.83)
```

This query will be required while creating the visualization of GPA averages as it returns the average GPA for courses in the past 5 years (we are only querying 2021 as we had to reduce the size of our data base for testing). Grouped by Subjects and CourseNo to include the same course taught by different instructors

2. **Query2**

```
mycursor.execute("""SELECT DISTINCT c.Subject, c.courseNo, ROUND(MAX(avgGPA), 2) as HighestGPA
FROM
(SELECT * FROM Courses WHERE courseNo LIKE '1%' AND YearTerm LIKE '%2021%') c
GROUP BY c.Subject, c.courseNo
HAVING HighestGPA > 3.75;""")
print_cursor(mycursor)
```

✓ 0.7s Python

```
Output exceeds the size limit. Open the full output data in a text editor
('ENG', 101, 3.79)
('ENG', 177, 3.91)
('ENG', 198, 3.88)
('ENG', 199, 3.94)
('ENGL', 103, 3.78)
('ENGL', 104, 3.91)
('ENGL', 109, 3.9)
('ACE', 199, 3.85)
('FR', 156, 3.93)
('ACES', 101, 3.9)
('ADV', 150, 3.8)
('AE', 100, 3.77)
('AE', 199, 3.92)
('AHS', 125, 3.77)
('AIS', 101, 3.85)
('ALEC', 110, 3.88)
('ANSC', 103, 3.87)
('ANSC', 110, 3.78)
('ANTH', 103, 3.8)
('ANTH', 180, 3.84)
('BIOE', 100, 3.93)
('BIOE', 120, 3.97)
('BUS', 101, 3.97)
```

This query is targeted to improve functionality for incoming freshmen as it returns the maximum GPA of 100 level courses if they are over 3.75. This query is also GroupedBy subject and courseNo to include the same course taught by different professors.

Explain Analyze

1. **Query 1 :**
    a. Without Indexing :

```
+----------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                      |
+----------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.002..0.055 rows=686 loops=1)
    -> Aggregate using temporary table  (actual time=3.417..3.508 rows=686 loops=1)
        -> Nested loop inner join  (cost=19038.15 rows=187759) (actual time=1.179..2.458 rows=1300 loops=1)
            -> Filter: (Courses.yearTerm is not null)  (cost=132.25 rows=1300) (actual time=0.071..0.595 rows=1300 loops=1)
                -> Table scan on Courses  (cost=132.25 rows=1300) (actual time=0.069..0.504 rows=1300 loops=1)
            -> Single-row index lookup on <subquery2> using <auto_distinct_key> (YearTerm=Courses.yearTerm)  (actual time=0.000..0.000 r
ows=1 loops=1300)
                -> Materialize with deduplication  (cost=146.69..146.69 rows=144) (actual time=1.528..1.635 rows=1 loops=1)
                    -> Filter: (Courses.yearTerm is not null)  (cost=132.25 rows=144) (actual time=0.033..0.687 rows=1300 loops=1)
                        -> Filter: (Courses.yearTerm like '%2021%')  (cost=132.25 rows=144) (actual time=0.033..0.600 rows=1300 loops=1)
                            -> Table scan on Courses  (cost=132.25 rows=1300) (actual time=0.030..0.383 rows=1300 loops=1)
 |
+----------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

Time Taken : *0.002..055 units*

b. With Indexing (Instructor) :

```
+----------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                      |
+----------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.001..0.077 rows=686 loops=1)
    -> Aggregate using temporary table  (actual time=3.646..3.760 rows=686 loops=1)
        -> Nested loop inner join  (cost=19038.15 rows=187759) (actual time=1.239..2.598 rows=1300 loops=1)
            -> Filter: (Courses.yearTerm is not null)  (cost=132.25 rows=1300) (actual time=0.126..0.799 rows=1300 loops=1)
                -> Table scan on Courses  (cost=132.25 rows=1300) (actual time=0.124..0.703 rows=1300 loops=1)
            -> Single-row index lookup on <subquery2> using <auto_distinct_key> (YearTerm=Courses.yearTerm)  (actual time=0.000..0.000 r
ows=1 loops=1300)
                -> Materialize with deduplication  (cost=146.69..146.69 rows=144) (actual time=1.469..1.577 rows=1 loops=1)
                    -> Filter: (Courses.yearTerm is not null)  (cost=132.25 rows=144) (actual time=0.038..0.694 rows=1300 loops=1)
                        -> Filter: (Courses.yearTerm like '%2021%')  (cost=132.25 rows=144) (actual time=0.038..0.594 rows=1300 loops=1)
                            -> Table scan on Courses  (cost=132.25 rows=1300) (actual time=0.034..0.376 rows=1300 loops=1)
 |
+----------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Time Taken : *0.001..0.077 units*

c. With Indexing (Subjects) :

Time Taken : *0.001..0.065 units*

d. With Indexing (avgGPA) :

Time Taken : *0.002..0.090 units*

Conlusion :

Subjects was added as index to optimize query performance as it performed better than the other indexes we chose (runtimes listed above)

2. **Query 2 :**

   a. Without Indexing :

```
| EXPLAIN                                                                                              |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| -> Filter: (HighestGPA > 3.75)  (actual time=1.349..1.377 rows=39 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.015 rows=107 loops=1)
        -> Aggregate using temporary table  (actual time=1.338..1.358 rows=107 loops=1)
            -> Filter: ((Courses.courseNo like '1%') and (Courses.yearTerm like '%2021%'))  (cost=132.25 rows=16) (actual time=0.082..1.003 rows=297 loops=1)
                -> Table scan on Courses  (cost=132.25 rows=1300) (actual time=0.074..0.628 rows=1300 loops=1)
 |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
```

   Time Taken : *1.349..1.377 units*

   b. With Indexing (Instructor):

```
mysql> EXPLAIN ANALYZE SELECT DISTINCT c.Subject, c.courseNo, ROUND(MAX(avgGPA), 2) as HighestGPA
    -> FROM
    -> (SELECT * FROM Courses WHERE courseNo LIKE '1%' AND YearTerm LIKE '%2021%') c
    -> GROUP BY c.Subject, c.courseNo
    -> HAVING HighestGPA > 3.75;
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                      |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| -> Filter: (HighestGPA > 3.75)  (actual time=1.126..1.150 rows=39 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.010 rows=107 loops=1)
        -> Aggregate using temporary table  (actual time=1.117..1.132 rows=107 loops=1)
            -> Filter: ((Courses.courseNo like '1%') and (Courses.yearTerm like '%2021%'))  (cost=132.25 rows=16) (actual time=0.097..0.
853 rows=297 loops=1)
                -> Table scan on Courses  (cost=132.25 rows=1300) (actual time=0.086..0.559 rows=1300 loops=1)
 |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

   Time Taken : *1.126..1.150 units*

   c. With Indexing (Subject) :

```
mysql> EXPLAIN ANALYZE SELECT DISTINCT c.Subject, c.courseNo, ROUND(MAX(avgGPA), 2) as HighestGPA FROM  (SELECT * FROM Courses WHERE cou
rseNo LIKE '1%' AND YearTerm LIKE '%2021%') c GROUP BY c.Subject, c.courseNo HAVING HighestGPA > 3.75;
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                      |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
| -> Filter: (HighestGPA > 3.75)  (actual time=1.059..1.082 rows=39 loops=1)
    -> Table scan on <temporary>  (actual time=0.000..0.009 rows=107 loops=1)
        -> Aggregate using temporary table  (actual time=1.051..1.066 rows=107 loops=1)
            -> Filter: ((Courses.courseNo like '1%') and (Courses.yearTerm like '%2021%'))  (cost=132.25 rows=16) (actual time=0.083..0.
797 rows=297 loops=1)
                -> Table scan on Courses  (cost=132.25 rows=1300) (actual time=0.067..0.505 rows=1300 loops=1)
 |
+------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

   Time Taken : *1.059..1.082 units*

d. With Indexing (Subject, CourseNo) :

```
mysql> DROP INDEX subjectidx ON Courses;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX subjectcourseidx ON Courses (Subject, courseNo);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT DISTINCT c.Subject, c.courseNo, ROUND(MAX(avgGPA), 2) as HighestGPA FROM  (SELECT * FROM Courses WHERE cou
rseNo LIKE '1%' AND YearTerm LIKE '%2021%') c GROUP BY c.Subject, c.courseNo HAVING HighestGPA > 3.75;
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+
| EXPLAIN


                                                                                   |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+
| -> Filter: (HighestGPA > 3.75)  (cost=133.85 rows=16) (actual time=1.938..4.757 rows=39 loops=1)
    -> Group aggregate: max(Courses.avgGPA)  (cost=133.85 rows=16) (actual time=0.441..4.734 rows=107 loops=1)
        -> Filter: ((Courses.courseNo like '1%') and (Courses.yearTerm like '%2021%'))  (cost=132.25 rows=16) (actual time=0.374..4.589
rows=297 loops=1)
            -> Index scan on Courses using subjectcourseidx  (cost=132.25 rows=1300) (actual time=0.364..4.339 rows=1300 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

Time Taken : *1.938..4.4757 units*

Conclusion :

Subjects was added as index to optimize query performance as it performed better than the other indexes we chose (runtimes listed above)