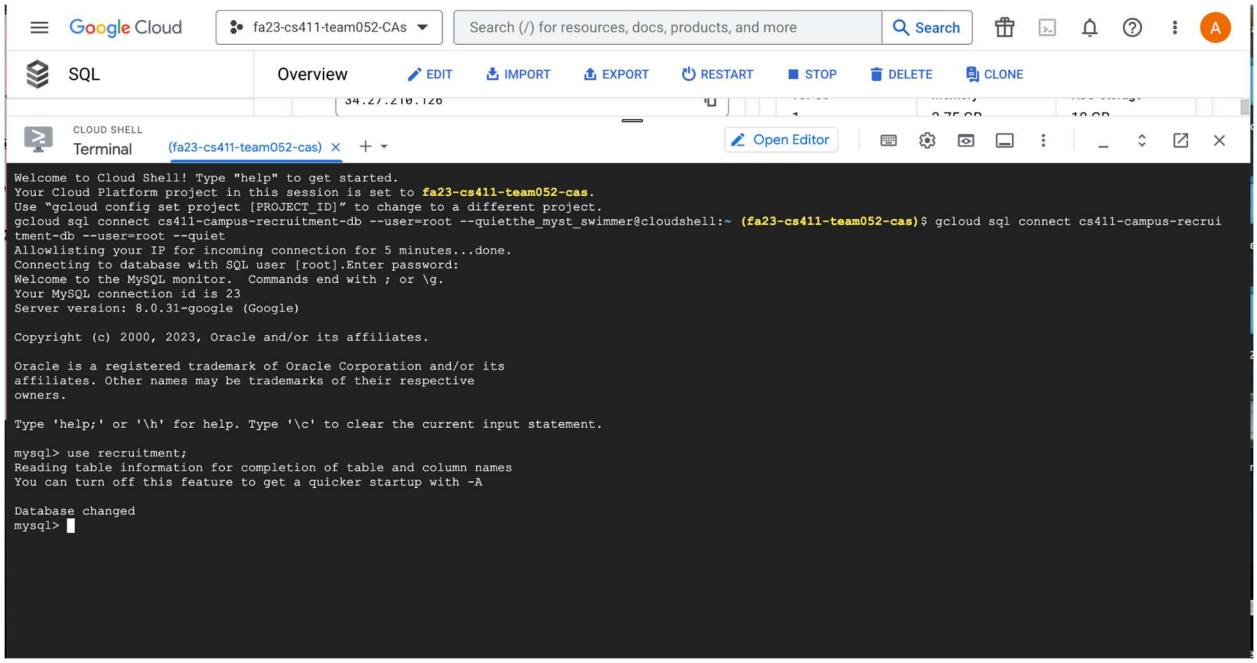
DDL COMMANDS:

```
CREATE TABLE Student_Profile (
	StudentId INT,
	HSC_Percent FLOAT,
	HSC_Subject VARCHAR(255),
	SSC_Percent FLOAT,
	Sector_Pref VARCHAR(255),
	Work_Exp VARCHAR(255),
	Undergrad_Degree VARCHAR(255),
	Gender VARCHAR(255),
	Degree_Percent FLOAT,
	PRIMARY KEY (StudentId)
);

CREATE TABLE User_Info (
	UserId INT,
	Password VARCHAR(255),
	Email VARCHAR(255),
	Name VARCHAR(255),
	PRIMARY KEY (UserId)
);

CREATE TABLE Grad_Student (
	StudentId INT,
	Emp_Percent FLOAT,
	Grad_Degree VARCHAR(255),
	FOREIGN KEY (StudentId) REFERENCES Student_Profile(StudentId)
);

CREATE TABLE Placement (
	StudentId INT,
	Grad_Percent FLOAT,
	Status VARCHAR(255),
	FOREIGN KEY (StudentId) REFERENCES Student_Profile(StudentId)
);

CREATE TABLE Placement_User_Data (
	StudentId INT,
	Status VARCHAR(255),
	Salary INT,
	Emp_Percent FLOAT,
	FOREIGN KEY (StudentId) REFERENCES Student_Profile(StudentId)
);
```

```
CREATE TABLE Company (
        CompanyId INT,
        Salary INT
);

CREATE TABLE Final_Output (
        StudentId INT,
        CompanyId INT,
        Salary INT,
        FOREIGN KEY (StudentId) REFERENCES Student_Profile(StudentId),
        FOREIGN KEY (CompanyId) REFERENCES Company(CompanyId)
);
```

COMMAND LINES IN GCP:



QUERIES:

```
mysql> SELECT SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree, AVG(GS.Grad_Percent) AS AvgGradPercent
    -> FROM Student_Profile AS SP
    -> LEFT JOIN Grad_Student AS GS ON SP.StudentId = GS.StudentId
    -> GROUP BY SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree;
+-----------+-------------+-------------+-------------+----------------------+
| StudentId | HSC_Percent | SSC_Percent | Grad_Degree | AvgGradPercent       |
+-----------+-------------+-------------+-------------+----------------------+
|         1 |          91 |          67 | Mkt&HR      |    58.79999923706055 |
|         2 |       78.33 |       79.33 | Mkt&Fin     |    66.27999877929688 |
|         3 |          68 |          65 | Mkt&Fin     |    57.79999923706055 |
|         4 |          52 |          56 | Mkt&HR      |    59.43000030517578 |
|         5 |        73.6 |        85.8 | Mkt&Fin     |                 55.5 |
|         6 |        49.8 |          55 | Mkt&Fin     |    51.58000183105469 |
|         7 |        49.2 |          46 | Mkt&Fin     |   53.290000915527344 |
|         8 |          64 |          82 | Mkt&Fin     |    62.13999938964844 |
|         9 |          79 |          73 | Mkt&Fin     |   61.290000915527344 |
|        10 |          70 |          58 | Mkt&Fin     |   52.209999084472656 |
|        11 |          61 |          58 | Mkt&HR      |   60.849998474121094 |
|        12 |        68.4 |        69.6 | Mkt&Fin     |    63.70000076293945 |
|        13 |          55 |          47 | Mkt&HR      |    65.04000091552734 |
|        14 |          87 |          77 | Mkt&Fin     |    68.62999725341797 |
|        15 |          47 |          62 | Mkt&HR      |   54.959999084472656 |
```

This query retrieves more than 15 rows but we have taken a screenshot of the first 15.

```
mysql> SELECT SP.Undergrad_Degree AS Degree, AVG(CO.Salary) AS AvgSalary, AVG(SP.Degree_Percent) AS AvgDegreePercent FROM Student_Profile SP NATURAL JOIN Placement PL NATURAL JOIN
Company CO GROUP BY SP.Undergrad_Degree;
+-----------+-------------+-------------------+
| Degree    | AvgSalary   | AvgDegreePercent  |
+-----------+-------------+-------------------+
| Comm&Mgmt | 239152.0951 | 54.61310346931072 |
| Others    | 239152.0951 | 52.23900754192403 |
| Sci&Tech  | 239152.0951 | 55.15546817653264 |
+-----------+-------------+-------------------+
3 rows in set (0.80 sec)
```

This query only retrieves 3 rows (rather than at least 15).
Note: might add the same query for Graduate degree later

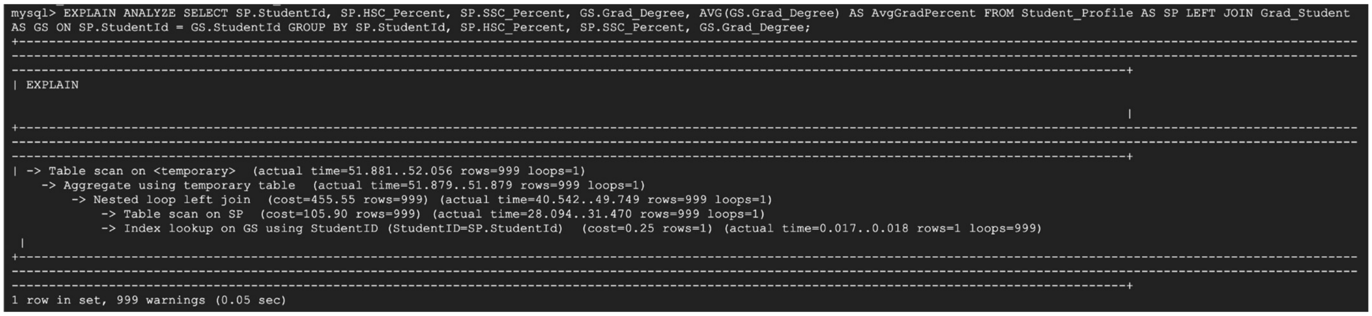
```
mysql> SELECT Gender, COUNT(*) AS NumberOfStudents FROM Student_Profile GROUP BY Gender;
+-------------+------------------+
| Gender      | NumberOfStudents |
+-------------+------------------+
| M           |              475 |
| F           |              459 |
| Bigender    |               15 |
| Genderqueer |                7 |
| Polygender  |               14 |
| Genderfluid |                9 |
| Agender     |               11 |
| Non-binary  |                9 |
+-------------+------------------+
8 rows in set (0.00 sec)
```

This query is just for future use in our application. Same with one below.

INDEXING:

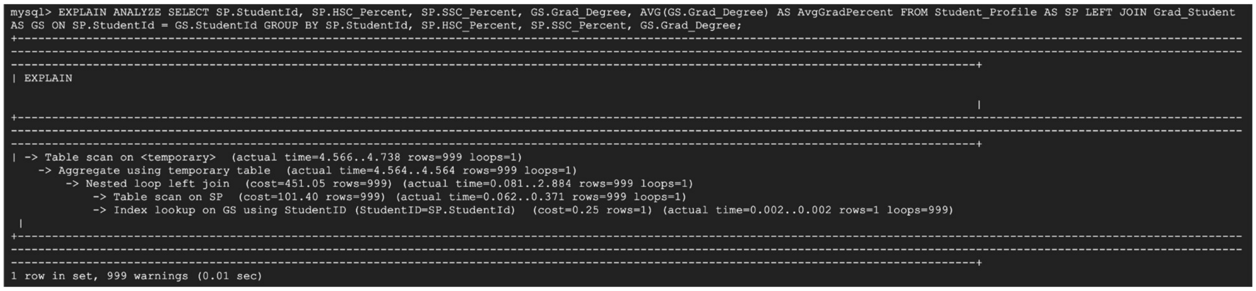
(1)

Original:



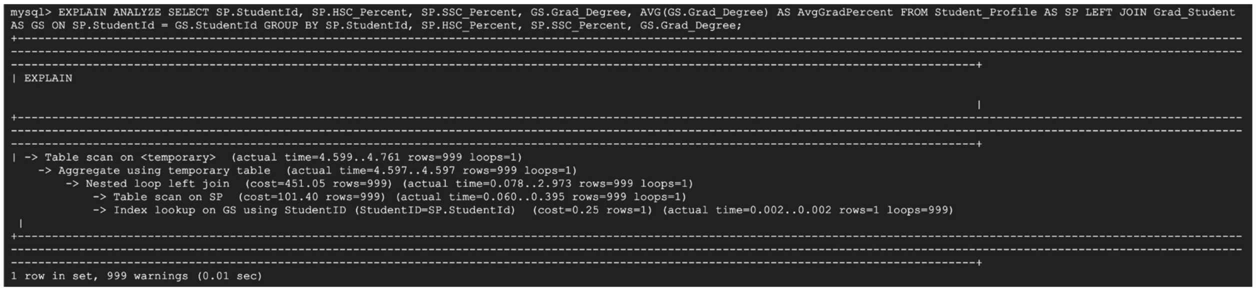
```
mysql> EXPLAIN ANALYZE SELECT SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree, AVG(GS.Grad_Degree) AS AvgGradPercent FROM Student_Profile AS SP LEFT JOIN Grad_Student
AS GS ON SP.StudentId = GS.StudentId GROUP BY SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=51.881..52.056 rows=999 loops=1)
    -> Aggregate using temporary table  (actual time=51.879..51.879 rows=999 loops=1)
        -> Nested loop left join  (cost=455.55 rows=999) (actual time=40.542..49.749 rows=999 loops=1)
            -> Table scan on SP  (cost=105.90 rows=999) (actual time=28.094..31.470 rows=999 loops=1)
            -> Index lookup on GS using StudentID (StudentID=SP.StudentId)  (cost=0.25 rows=1) (actual time=0.017..0.018 rows=1 loops=999)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set, 999 warnings (0.05 sec)
```

0.05 sec originally - **cost is 455.55 for left-join; Parsing data take cost = 105.90**



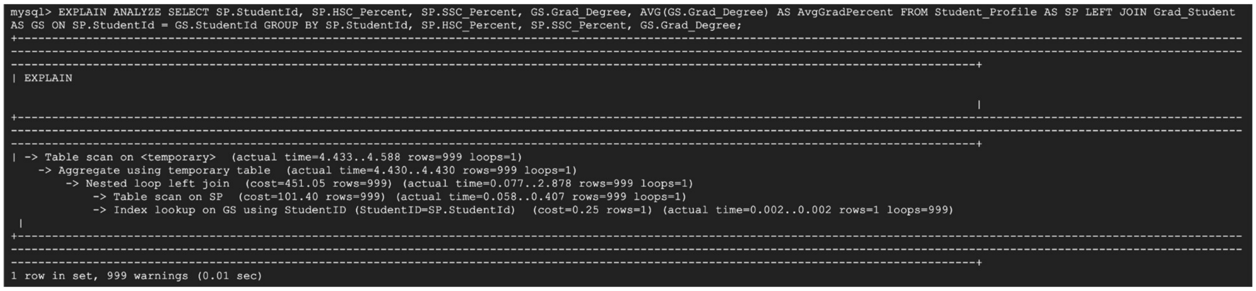
```
mysql> EXPLAIN ANALYZE SELECT SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree, AVG(GS.Grad_Degree) AS AvgGradPercent FROM Student_Profile AS SP LEFT JOIN Grad_Student
AS GS ON SP.StudentId = GS.StudentId GROUP BY SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=4.566..4.738 rows=999 loops=1)
    -> Aggregate using temporary table  (actual time=4.564..4.564 rows=999 loops=1)
        -> Nested loop left join  (cost=451.05 rows=999) (actual time=0.081..2.884 rows=999 loops=1)
            -> Table scan on SP  (cost=101.40 rows=999) (actual time=0.062..0.371 rows=999 loops=1)
            -> Index lookup on GS using StudentID (StudentID=SP.StudentId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=999)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set, 999 warnings (0.01 sec)
```

0.01 with idx_grad_degree from Grad_Student; **cost is 451.55 for left-join;Parsing data take cost = 101.40**



```
mysql> EXPLAIN ANALYZE SELECT SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree, AVG(GS.Grad_Degree) AS AvgGradPercent FROM Student_Profile AS SP LEFT JOIN Grad_Student
AS GS ON SP.StudentId = GS.StudentId GROUP BY SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=4.599..4.761 rows=999 loops=1)
    -> Aggregate using temporary table  (actual time=4.597..4.597 rows=999 loops=1)
        -> Nested loop left join  (cost=451.05 rows=999) (actual time=0.078..2.973 rows=999 loops=1)
            -> Table scan on SP  (cost=101.40 rows=999) (actual time=0.060..0.395 rows=999 loops=1)
            -> Index lookup on GS using StudentID (StudentID=SP.StudentId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=999)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set, 999 warnings (0.01 sec)
```

0.01 with just idx_hsc_percent from Student_Profile; cost remains same



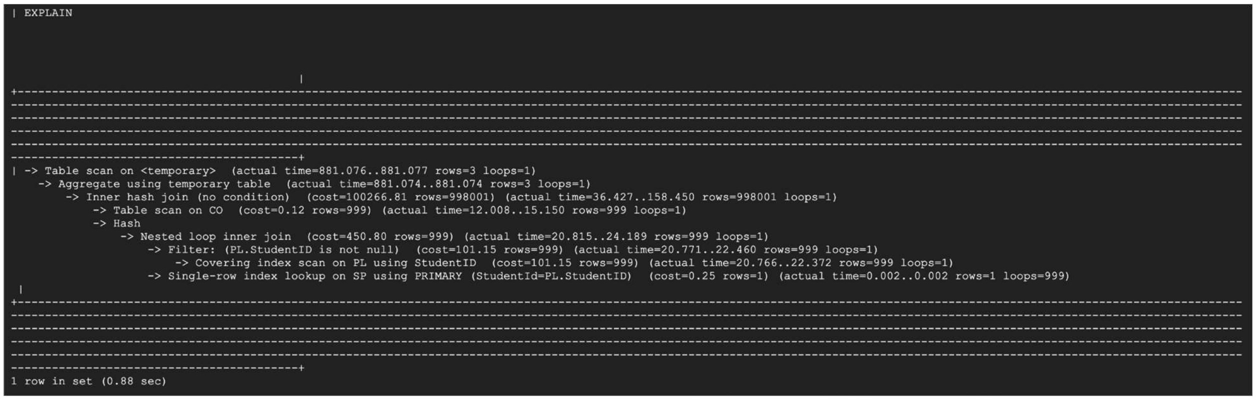
```
mysql> EXPLAIN ANALYZE SELECT SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree, AVG(GS.Grad_Degree) AS AvgGradPercent FROM Student_Profile AS SP LEFT JOIN Grad_Student
AS GS ON SP.StudentId = GS.StudentId GROUP BY SP.StudentId, SP.HSC_Percent, SP.SSC_Percent, GS.Grad_Degree;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=4.433..4.588 rows=999 loops=1)
    -> Aggregate using temporary table  (actual time=4.430..4.430 rows=999 loops=1)
        -> Nested loop left join  (cost=451.05 rows=999) (actual time=0.077..2.878 rows=999 loops=1)
            -> Table scan on SP  (cost=101.40 rows=999) (actual time=0.058..0.407 rows=999 loops=1)
            -> Index lookup on GS using StudentID (StudentID=SP.StudentId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=999)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set, 999 warnings (0.01 sec)
```

0.01 with both idx_grad_degree and idx_hsc_percent Grad_Student and Student_Profile

For our first index, we decided to index by grad_degree to begin with. We thought it was a good starting point to see how the time would be affected. Fortunately, the time dropped dramatically with our idx_grad_degree. It went from 0.05 to 0.01. Cost dropped from **455.55 to 451.05 for left-join and from 105.90 to 101.40 for parsing data.** For our second index design for this query, we index by HSC_Percent. This also dropped the time down significantly, but to an equal extent; it was the same 0.05 -> 0.01. Cost remained the same in this case.Our third index design was to try and implement both indices. As expected, the time decreased but by the same amount again. Cost remained the same. Our group decided to go with the 3rd index design because we believe it would cut down on cost/time when we run this with even more data/when this program is run at the same time as other tables.

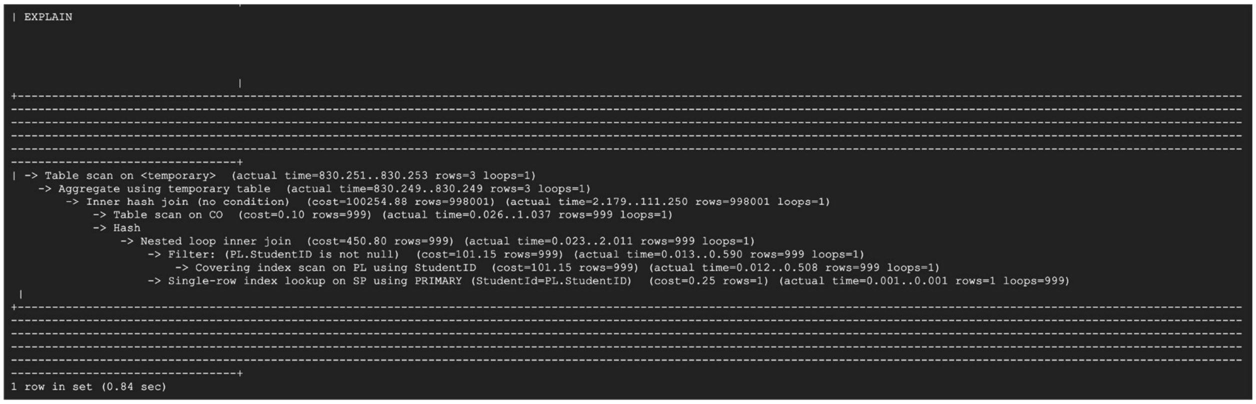
(2)

Original:



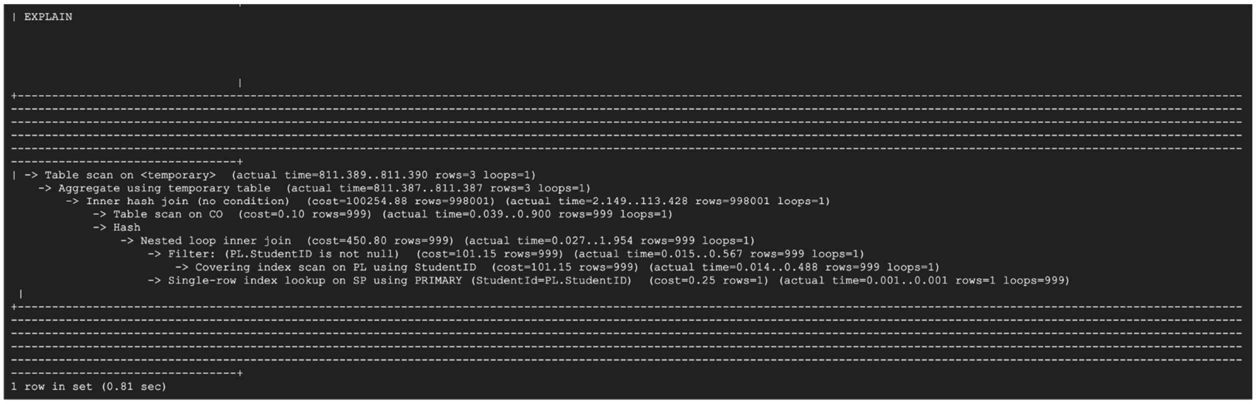
```
| EXPLAIN



                                           |
+--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------+
| -> Table scan on <temporary>  (actual time=881.076..881.077 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=881.074..881.074 rows=3 loops=1)
        -> Inner hash join (no condition)  (cost=100266.81 rows=998001) (actual time=36.427..158.450 rows=998001 loops=1)
            -> Table scan on CO  (cost=0.12 rows=999) (actual time=12.008..15.150 rows=999 loops=1)
            -> Hash
                -> Nested loop inner join  (cost=450.80 rows=999) (actual time=20.815..24.189 rows=999 loops=1)
                    -> Filter: (PL.StudentID is not null)  (cost=101.15 rows=999) (actual time=20.771..22.460 rows=999 loops=1)
                        -> Covering index scan on PL using StudentID  (cost=101.15 rows=999) (actual time=20.766..22.372 rows=999 loops=1)
                    -> Single-row index lookup on SP using PRIMARY (StudentId=PL.StudentID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=999)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------+
1 row in set (0.88 sec)
```

0.88 originally - cost for inner hash join is 100266.81



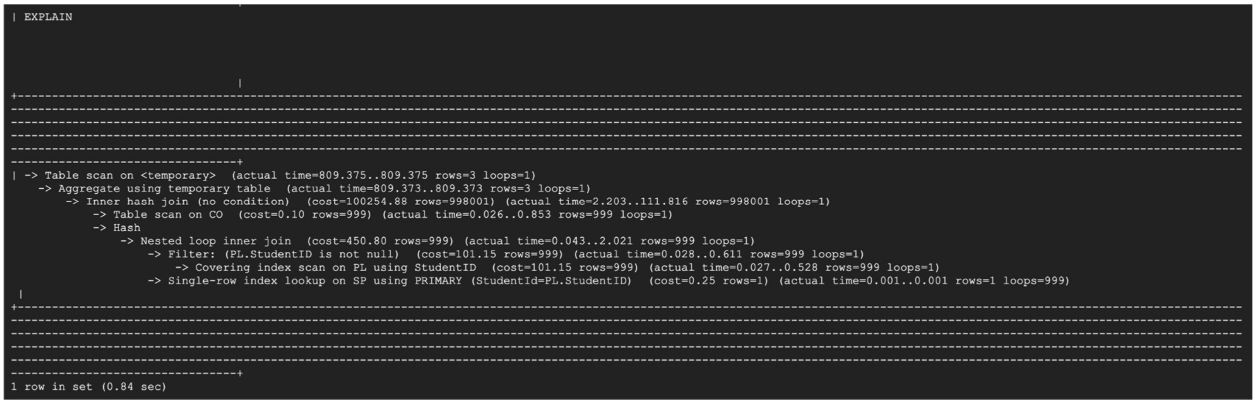
```
| EXPLAIN



                                  |
+--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------+
| -> Table scan on <temporary>  (actual time=830.251..830.253 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=830.249..830.249 rows=3 loops=1)
        -> Inner hash join (no condition)  (cost=100254.88 rows=998001) (actual time=2.179..111.250 rows=998001 loops=1)
            -> Table scan on CO  (cost=0.10 rows=999) (actual time=0.026..1.037 rows=999 loops=1)
            -> Hash
                -> Nested loop inner join  (cost=450.80 rows=999) (actual time=0.023..2.011 rows=999 loops=1)
                    -> Filter: (PL.StudentID is not null)  (cost=101.15 rows=999) (actual time=0.013..0.590 rows=999 loops=1)
                        -> Covering index scan on PL using StudentID  (cost=101.15 rows=999) (actual time=0.012..0.508 rows=999 loops=1)
                    -> Single-row index lookup on SP using PRIMARY (StudentId=PL.StudentID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=999)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------+
1 row in set (0.84 sec)
```

0.84 with idx_gender from Student_Profile; cost for inner hash join drops to 100254.88



```
| EXPLAIN



                                 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------+
| -> Table scan on <temporary>  (actual time=811.389..811.390 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=811.387..811.387 rows=3 loops=1)
        -> Inner hash join (no condition)  (cost=100254.88 rows=998001) (actual time=2.149..113.428 rows=998001 loops=1)
            -> Table scan on CO  (cost=0.10 rows=999) (actual time=0.039..0.900 rows=999 loops=1)
            -> Hash
                -> Nested loop inner join  (cost=450.80 rows=999) (actual time=0.027..1.954 rows=999 loops=1)
                    -> Filter: (PL.StudentID is not null)  (cost=101.15 rows=999) (actual time=0.015..0.567 rows=999 loops=1)
                        -> Covering index scan on PL using StudentID  (cost=101.15 rows=999) (actual time=0.014..0.488 rows=999 loops=1)
                    -> Single-row index lookup on SP using PRIMARY (StudentId=PL.StudentID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=999)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------+
1 row in set (0.81 sec)
```

0.81 with idx_degree_percent from Student_Profile



```
| EXPLAIN



                                 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------+
| -> Table scan on <temporary>  (actual time=809.375..809.375 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=809.373..809.373 rows=3 loops=1)
        -> Inner hash join (no condition)  (cost=100254.88 rows=998001) (actual time=2.203..111.816 rows=998001 loops=1)
            -> Table scan on CO  (cost=0.10 rows=999) (actual time=0.026..0.853 rows=999 loops=1)
            -> Hash
                -> Nested loop inner join  (cost=450.80 rows=999) (actual time=0.043..2.021 rows=999 loops=1)
                    -> Filter: (PL.StudentID is not null)  (cost=101.15 rows=999) (actual time=0.028..0.611 rows=999 loops=1)
                        -> Covering index scan on PL using StudentID  (cost=101.15 rows=999) (actual time=0.027..0.528 rows=999 loops=1)
                    -> Single-row index lookup on SP using PRIMARY (StudentId=PL.StudentID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=999)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------+
1 row in set (0.84 sec)
```

0.84 with idx_emp_percent from Placement; **Nested inner join cost decreased further to 450.88**

For our first index design, we went with idx_gender more so out of curiosity to see how the time would be impacted. It was a pretty small change. Our second index design was idx_degree_percent, which was a lot better than gender. We think this was because it had more correlation to our query. Instead of adding both indices, we wanted to make our third index design on emp_percent, which is from a different table. Again, the change was pretty small. We are going to go with idx_degree_percent because of its improved time and correlation to our overall project. The cost decreases the most in the third index design.

COUNT OF ROWS FOR EACH TABLE:

Our tables:

```
mysql> show tables;
+-----------------------+
| Tables_in_recruitment |
+-----------------------+
| Company               |
| Final_Output          |
| Grad_Student          |
| Placement             |
| Predicted_User_Data   |
| Student_Profile       |
| User_Info             |
| import                |
+-----------------------+
8 rows in set (0.01 sec)
```

Actual count of rows for each table:

```
mysql> SELECT COUNT(*) FROM Company;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM Final_Output;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.05 sec)
```

```
mysql> SELECT COUNT(*) FROM Grad_Student;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM Placement;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM Predicted_User_Data;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.05 sec)
```

```
mysql> SELECT COUNT(*) FROM Student_Profile;
+----------+
| COUNT(*) |
+----------+
|      999 |
+----------+
1 row in set (0.00 sec)
```

Note: “import” table is just the table we used to load the data into our actual tables. “User_Info” also currently only has 1 dummy row in it because the table will only have more data once Users actually create profiles on our website.