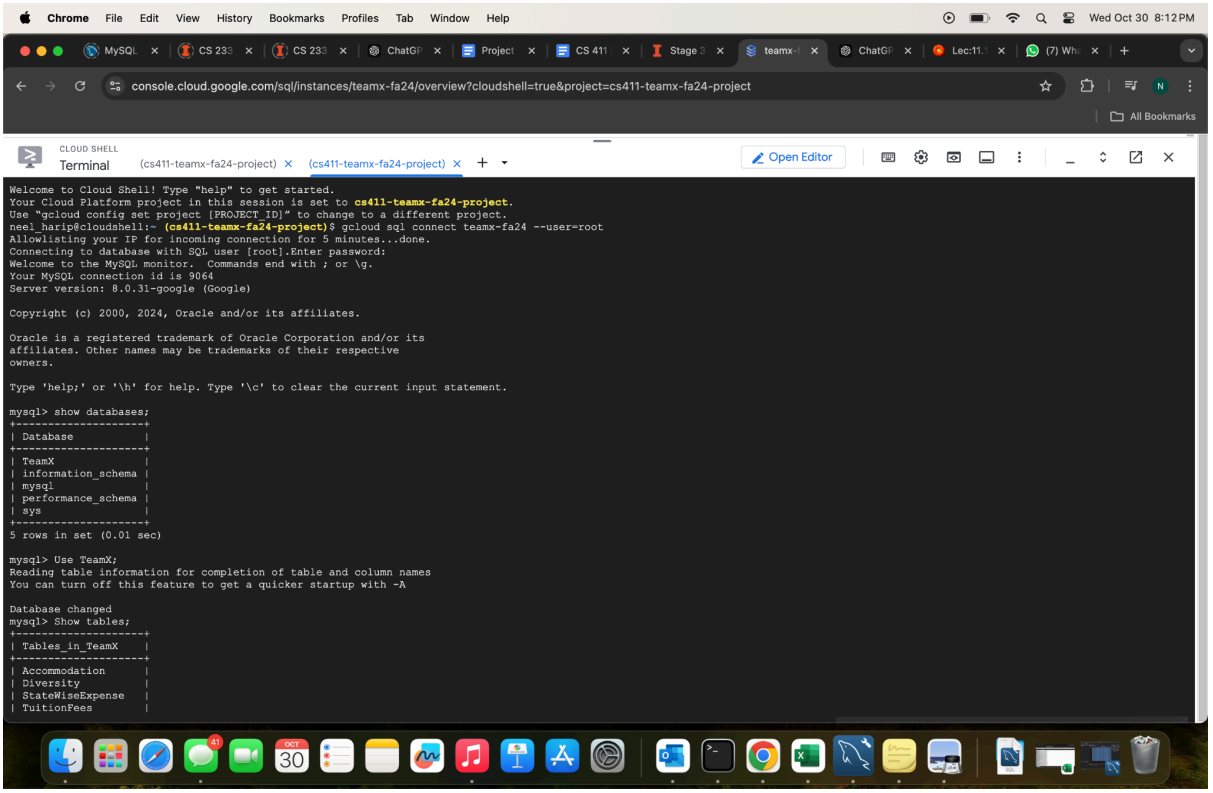
Stage 3: Database Design

Project Title: GlobalScholar: International Student Financial Navigator

Group Members: Neel Harip(nhari7), Sakshil Verma(sakshil2), Sejal Pekam(spekam2), Sharvari Gadiwan(sgadi5)

1. Database Implementation

1.1 GCP Connection



1.2 DDL Commands For Table Creation

1. User

```
CREATE TABLE User (
    Id INT PRIMARY KEY,
    FirstName VARCHAR(255),
    LastName VARCHAR(255),
    EmailId VARCHAR(255),
    Password VARCHAR(255),
```

```
    TuitionFeeBudget INT,
    AccommodationBudget INT
);
```

2. University

```
CREATE TABLE University (
    UniversityName VARCHAR(255) PRIMARY KEY,
    State VARCHAR(100)
);
```

3. Diversity

```
CREATE TABLE Diversity (
    UniversityName VARCHAR(255),
    TotalEnrollment INT,
    RaceCategory VARCHAR(100),
    RaceWiseEnrollment INT,
    PRIMARY KEY (UniversityName, RaceCategory),
    FOREIGN KEY (UniversityName) REFERENCES University(UniversityName)
);
```

4. StatewiseExpense

```
CREATE TABLE StateWiseExpense (
    State VARCHAR(100) PRIMARY KEY,
    HousingCost INT,
    FoodCost INT,
    TransportationCost INT,
    HealthcareCost INT
);
```

5. Accommodation

```
CREATE TABLE Accommodation (
    UniversityName VARCHAR(255),
    DegreeLength VARCHAR(255),
    RoomAndBoardCost INT,
    PRIMARY KEY (UniversityName),
    FOREIGN KEY (UniversityName) REFERENCES University(UniversityName)
);
```

6. TuitionFees

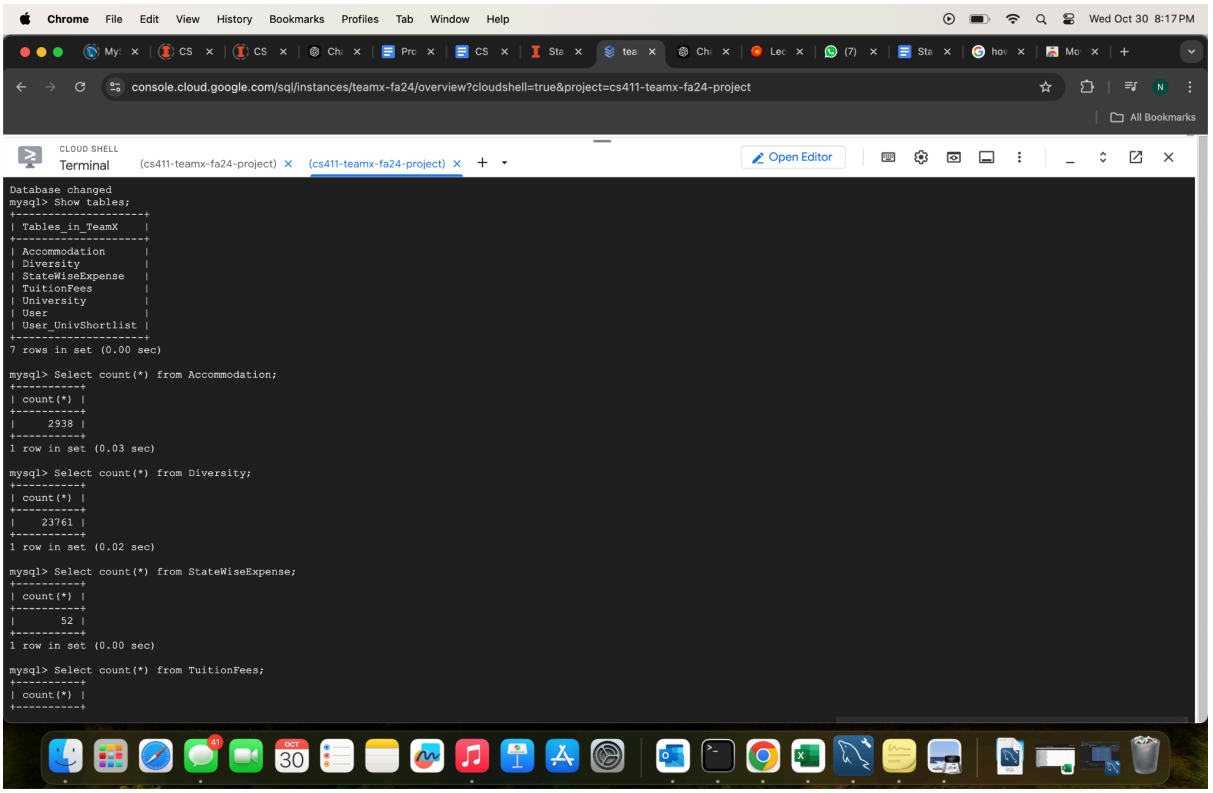
```
CREATE TABLE TuitionFees (
    UniversityName VARCHAR(255),
    UniversityType VARCHAR(100),
    DegreeLength INT,
    InStateTuitionFees INT,
    OutOfStateTuitionFees INT,
    PRIMARY KEY (UniversityName),
```

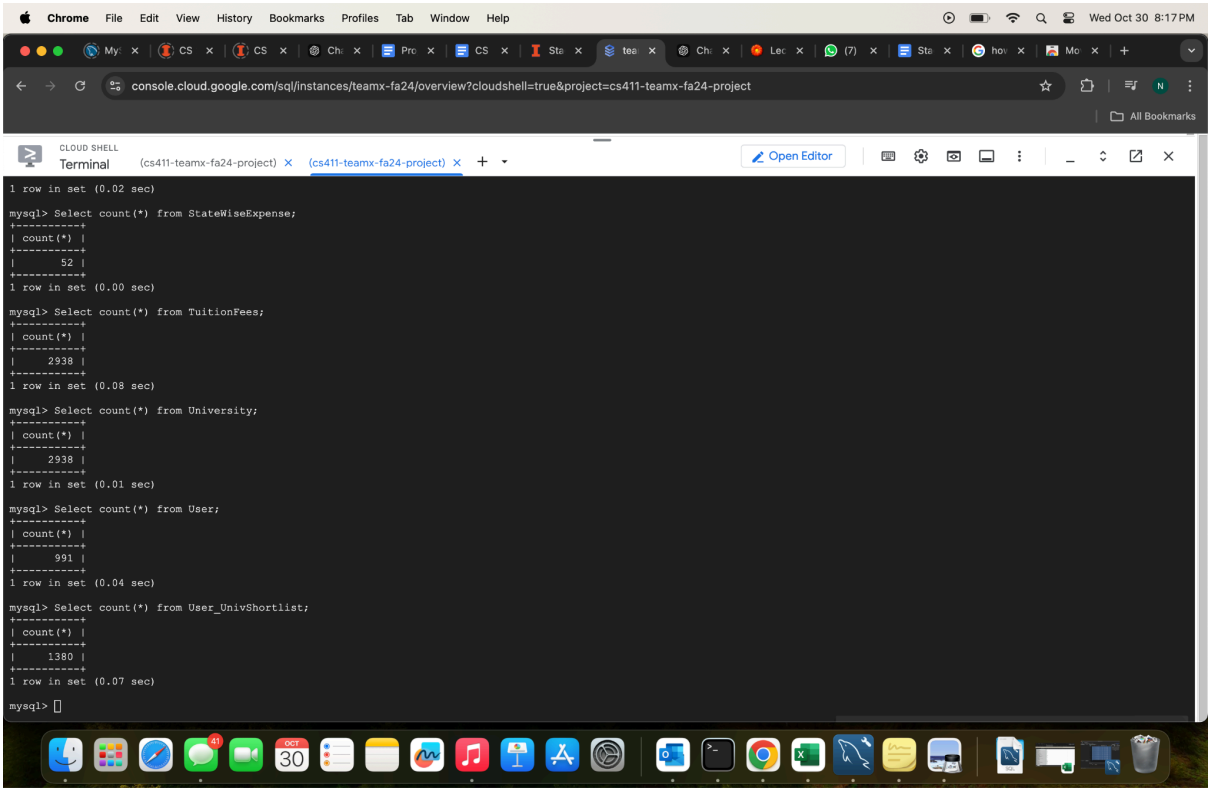
```
    FOREIGN KEY (UniversityName) REFERENCES University(UniversityName)
);
```

7. User_UnivShortlist

```
CREATE TABLE User_UnivShortlist (
    UserId INT,
    UniversityName VARCHAR(255),
    PRIMARY KEY (UserId, UniversityName),
    FOREIGN KEY (UserId) REFERENCES User(Id),
    FOREIGN KEY (UniversityName) REFERENCES University(UniversityName)
);
```

1.3 Count Query for Tables



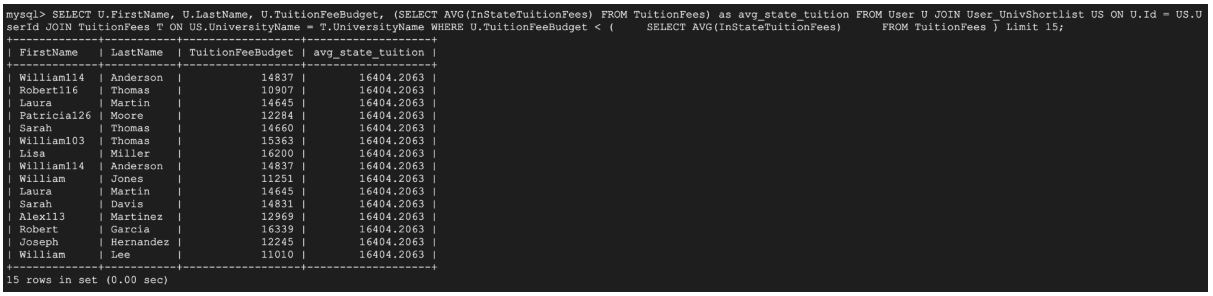


2. Advanced Queries

2.1 Query 1

This query helps GlobalScholar identify users whose tuition budgets are below the average in-state university tuition, highlighting potential financial challenges.

```
SELECT U.FirstName, U.LastName, U.TuitionFeeBudget,
       (SELECT AVG(InStateTuitionFees) FROM TuitionFees) as avg_state_tuition
FROM User U
JOIN User_UnivShortlist US ON U.Id = US.UserId
JOIN TuitionFees T ON US.UniversityName = T.UniversityName
WHERE U.TuitionFeeBudget < (
   SELECT AVG(InStateTuitionFees)
   FROM TuitionFees
);
```



```
mysql> SELECT U.FirstName, U.LastName, U.TuitionFeeBudget, (SELECT AVG(InStateTuitionFees) FROM TuitionFees) as avg_state_tuition FROM User U JOIN User_UnivShortlist US ON U.Id = US.U
serId JOIN TuitionFees T ON US.UniversityName = T.UniversityName WHERE U.TuitionFeeBudget < (     SELECT AVG(InStateTuitionFees)      FROM TuitionFees ) Limit 15;
+-------------+----------+------------------+-------------------+
| FirstName   | LastName | TuitionFeeBudget | avg_state_tuition |
+-------------+----------+------------------+-------------------+
| William114  | Anderson |            14837 |        16404.2063 |
| Robert116   | Thomas   |            10907 |        16404.2063 |
| Laura       | Martin   |            14645 |        16404.2063 |
| Patricia126 | Moore    |            12284 |        16404.2063 |
| Sarah       | Thomas   |            14660 |        16404.2063 |
| William103  | Thomas   |            15363 |        16404.2063 |
| Lisa        | Miller   |            16200 |        16404.2063 |
| William114  | Anderson |            14837 |        16404.2063 |
| William     | Jones    |            11251 |        16404.2063 |
| Laura       | Martin   |            14645 |        16404.2063 |
| Sarah       | Davis    |            14831 |        16404.2063 |
| Alex113     | Martinez |            12969 |        16404.2063 |
| Robert      | Garcia   |            16339 |        16404.2063 |
| Joseph      | Hernandez |           12245 |        16404.2063 |
| William     | Lee      |            11010 |        16404.2063 |
+-------------+----------+------------------+-------------------+
15 rows in set (0.00 sec)
```

2.2 Query 2

This query calculates the average living costs for universities shortlisted by a specific user and checks if they fall within their budget.

```
SELECT U.FirstName, U.LastName, Univ.UniversityName,
      Round(Avg(S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost +
A.RoomAndBoardCost), 0) AS TotalLivingCost,
      U.TuitionFeeBudget + U.AccommodationBudget AS TotalBudget
FROM User U
JOIN User_UnivShortlist US ON U.Id = US.UserId
JOIN University Univ ON US.UniversityName = Univ.UniversityName
JOIN StateWiseExpense S ON Univ.State = S.State
JOIN Accommodation A ON Univ.UniversityName = A.UniversityName
WHERE U.Id = 42
  AND (S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost +
A.RoomAndBoardCost) <= (U.TuitionFeeBudget + U.AccommodationBudget)
GROUP BY U.Id, U.FirstName, U.LastName, Univ.UniversityName, S.HousingCost,
S.FoodCost, S.TransportationCost, S.HealthcareCost, A.RoomAndBoardCost;
```

```
mysql> SELECT U.FirstName, U.LastName, Univ.UniversityName,
    ->        Round(Avg(S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost + A.RoomAndBoardCost), 0) AS TotalLivingCost,
    ->        U.TuitionFeeBudget + U.AccommodationBudget AS TotalBudget
    -> FROM User U
    -> JOIN User_UnivShortlist US ON U.Id = US.UserId
    -> JOIN University Univ ON US.UniversityName = Univ.UniversityName
    -> JOIN StateWiseExpense S ON Univ.State = S.State
    -> JOIN Accommodation A ON Univ.UniversityName = A.UniversityName
    -> WHERE U.Id = 42
    ->   AND (S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost + A.RoomAndBoardCost) <= (U.TuitionFeeBudget + U.AccommodationBudget)
    -> GROUP BY U.Id, U.FirstName, U.LastName, Univ.UniversityName, S.HousingCost, S.FoodCost, S.TransportationCost, S.HealthcareCost, A.RoomAndBoardCost;
+-----------+----------+-----------------------------------------+-----------------+-------------+
| FirstName | LastName | UniversityName                          | TotalLivingCost | TotalBudget |
+-----------+----------+-----------------------------------------+-----------------+-------------+
| David     | Lee      | City University of New York: Lehman College |           40080 |       49759 |
| David     | Lee      | Del Mar College                         |           26256 |       49759 |
| David     | Lee      | Dutchess Community College              |           39778 |       49759 |
| David     | Lee      | East-West University                    |           25752 |       49759 |
| David     | Lee      | Lehigh Carbon Community College         |           28731 |       49759 |
| David     | Lee      | Memphis College of Art                  |           34314 |       49759 |
| David     | Lee      | Northwest State Community College       |           24590 |       49759 |
| David     | Lee      | Oklahoma Panhandle State University     |           31574 |       49759 |
| David     | Lee      | UW Colleges Rock County                 |           25471 |       49759 |
| David     | Lee      | Virginia Marti College of Art and Design |           24590 |       49759 |
+-----------+----------+-----------------------------------------+-----------------+-------------+
10 rows in set (0.00 sec)
```

Explanation for 10 rows - Each user is corresponded (allowed) to shortlist only 10 universities, hence it only displays 10 rows.

2.3 Query 3

The query retrieves the first and last names of students, along with their shortlisted universities, in-state tuition fees, and room and board costs. It filters the results to include only those students whose budgets for tuition and accommodation meet or exceed the respective costs associated with their selected universities.

```
SELECT
   u.FirstName,
```

```
    u.LastName,
    tf.UniversityName,
    tf.InStateTuitionFees,
    (
        SELECT a.RoomAndBoardCost
        FROM Accommodation a
        WHERE a.UniversityName = tf.UniversityName
    ) AS RoomAndBoardCost
FROM
    User u
JOIN
    User_UnivShortlist uu ON u.Id = uu.UserId
JOIN
    TuitionFees tf ON uu.UniversityName = tf.UniversityName
WHERE
    u.TuitionFeeBudget >= tf.InStateTuitionFees
    AND u.AccommodationBudget >= (
        SELECT a.RoomAndBoardCost
        FROM Accommodation a
        WHERE a.UniversityName = tf.UniversityName
    );
```

```
mysql> SELECT      u.FirstName,      u.LastName,      tf.UniversityName,      tf.InStateTuitionFees,      (          SELECT a.RoomAndBoardCost          FROM Accommodation a          WHERE
 a.UniversityName = tf.UniversityName      ) AS RoomAndBoardCost FROM      User u JOIN      User_UnivShortlist uu ON u.Id = uu.UserId JOIN      TuitionFees tf ON uu.UniversityName = tf
.UniversityName WHERE      u.TuitionFeeBudget >= tf.InStateTuitionFees      AND u.AccommodationBudget >= (          SELECT a.RoomAndBoardCost          FROM Accommodation a          WHER
E a.UniversityName = tf.UniversityName      )Limit 15;
+-----------+----------+----------------------------------------+--------------------+------------------+
| FirstName | LastName | UniversityName                         | InStateTuitionFees | RoomAndBoardCost |
+-----------+----------+----------------------------------------+--------------------+------------------+
| William33 | Jones    | Aaniiih Nakoda College                 |               2380 |                0 |
| William14 | Anderson | Aaniiih Nakoda College                 |               2380 |                0 |
| Alex      | Moore    | Adrian College                         |              37087 |            11318 |
| Jane      | Jones    | Adventist University of Health Sciences |              15150 |             4200 |
| Joseph138 | Hernandez | Adventist University of Health Sciences |              15150 |             4200 |
| Patricia  | Johnson  | Aims Community College                 |               2281 |                0 |
| Robert116 | Thomas   | Alabama Southern Community College     |               4440 |                0 |
| Alex144   | Smith    | Alabama Southern Community College     |               4440 |                0 |
| Laura     | Martin   | Alabama State University               |              11068 |             5422 |
| Michael   | Anderson | Alabama State University               |              11068 |             5422 |
| Linda     | Taylor   | Alamance Community College             |               2310 |                0 |
| Patricia92 | Brown    | Alamance Community College             |               2310 |                0 |
| Michael   | Anderson | Albany Technical College               |               3246 |                0 |
| Chris139  | Martin   | Albany Technical College               |               3246 |                0 |
| William   | Williams | Albertus Magnus College                |              32060 |            13200 |
+-----------+----------+----------------------------------------+--------------------+------------------+
15 rows in set (0.01 sec)
```

2.4 Query 4

The query retrieves data on university enrollments for a specified race category, calculating the total enrollment for that category and the overall enrollment for each university. It computes the percentage of enrollment for the race category and orders the results to display universities with the highest representation first.

```
SELECT
    d.UniversityName,
    d.RaceCategory,
    SUM(d.RaceWiseEnrollment) AS RaceWiseTotal,
    (SELECT SUM(d2.RaceWiseEnrollment)
     FROM Diversity d2
     WHERE d2.UniversityName = d.UniversityName) AS TotalEnrollment,
    (SUM(d.RaceWiseEnrollment) /
     (SELECT SUM(d2.RaceWiseEnrollment)
```

```
    FROM Diversity d2
    WHERE d2.UniversityName = d.UniversityName)) * 100 AS PercentageOfEnrollment
FROM
  Diversity d
WHERE
  d.RaceCategory = 'Asian' -- Replace with the user's desired race category
GROUP BY
  d.UniversityName,
  d.RaceCategory
ORDER BY
  PercentageOfEnrollment DESC;
```

```
mysql> SELECT     d.UniversityName,     d.RaceCategory,     SUM(d.RaceWiseEnrollment) AS RaceWiseTotal,     (SELECT SUM(d2.RaceWiseEnrollment)      FROM Diversity d2      WHERE d2.Un
iversityName = d.UniversityName) AS TotalEnrollment,     (SUM(d.RaceWiseEnrollment) /      (SELECT SUM(d2.RaceWiseEnrollment)      FROM Diversity d2      WHERE d2.UniversityName =
d.UniversityName)) * 100 AS PercentageOfEnrollment FROM      Diversity d WHERE      d.RaceCategory = 'Asian'  GROUP BY      d.UniversityName,      d.RaceCategory ORDER BY      Percent
ageOfEnrollment DESC Limit 15;
+-----------------------------------+--------------+---------------+-----------------+------------------------+
| UniversityName                    | RaceCategory | RaceWiseTotal | TotalEnrollment | PercentageOfEnrollment |
+-----------------------------------+--------------+---------------+-----------------+------------------------+
| World Mission University          | Asian        |           172 |             626 |               27.4760 |
| Bethesda University of California | Asian        |           191 |             749 |               25.5007 |
| Mission College                   | Asian        |          3733 |           19713 |               18.9367 |
| Long Island Business Institute    | Asian        |           172 |             941 |               18.2784 |
| Art Center College of Design      | Asian        |           666 |            4045 |               16.4648 |
| Skyline College                   | Asian        |          3693 |           22564 |               16.3668 |
| Evergreen Valley College          | Asian        |          3522 |           21807 |               16.1508 |
| University of the Pacific         | Asian        |          2101 |           13248 |               15.8590 |
| De Anza College                   | Asian        |          7956 |           50305 |               15.8155 |
| Divine Word College               | Asian        |            27 |             174 |               15.5172 |
| California Institute of Technology | Asian        |           580 |            3741 |               15.5039 |
| Ohlone College                    | Asian        |          3564 |           23480 |               15.1789 |
| City College of San Francisco     | Asian        |          7972 |           52950 |               15.0557 |
| San Jose State University         | Asian        |         10381 |           69613 |               14.9124 |
| Lawrence Technological University | Asian        |           925 |            6310 |               14.6593 |
+-----------------------------------+--------------+---------------+-----------------+------------------------+
15 rows in set (0.10 sec)
```

2.5 Query 5

The query calculates and ranks the top 10 universities by enrollment percentage for a specified race category, providing valuable insights into university diversity. It serves as a resource for students seeking institutions with significant representation of their racial group and aids administrators in assessing and improving diversity efforts.

```
SELECT
  UniversityName,
  RaceCategory,
  TotalEnrollment,
  RaceWiseEnrollment,
  EnrollmentPercentage
FROM (
  SELECT
    D.UniversityName,
    D.RaceCategory,
    SUM(D.TotalEnrollment) AS TotalEnrollment,
    SUM(D.RaceWiseEnrollment) AS RaceWiseEnrollment,
    (SUM(D.RaceWiseEnrollment) / SUM(D.TotalEnrollment) * 100) AS
EnrollmentPercentage
  FROM
    Diversity D
  WHERE
    D.RaceWiseEnrollment > 0
  GROUP BY
```

```
        D.UniversityName, D.RaceCategory
) AS RankedDiversity
WHERE
    EnrollmentPercentage IN (
        SELECT
            EnrollmentPercentage
        FROM (
            SELECT
                EnrollmentPercentage,
                ROW_NUMBER() OVER (PARTITION BY RaceCategory ORDER BY
EnrollmentPercentage DESC) AS rn
            FROM (
                SELECT
                    D.UniversityName,
                    D.RaceCategory,
                    SUM(D.TotalEnrollment) AS TotalEnrollment,
                    SUM(D.RaceWiseEnrollment) AS RaceWiseEnrollment,
                    (SUM(D.RaceWiseEnrollment) / SUM(D.TotalEnrollment) * 100) AS
EnrollmentPercentage
                FROM
                    Diversity D
                WHERE
                    D.RaceWiseEnrollment > 0
                GROUP BY
                    D.UniversityName, D.RaceCategory
            ) AS InnerQuery
        ) AS Ranked
        WHERE rn <= 10
    )
ORDER BY
    RaceCategory, EnrollmentPercentage DESC;
```

```
mysql> SELECT      UniversityName,     RaceCategory,     TotalEnrollment,     RaceWiseEnrollment,     EnrollmentPercentage FROM (     SELECT          D.UniversityName,          D.RaceC
ategory,          SUM(D.TotalEnrollment) AS TotalEnrollment,          SUM(D.RaceWiseEnrollment) AS RaceWiseEnrollment,          (SUM(D.RaceWiseEnrollment) / SUM(D.TotalEnrollment) * 100)
 AS EnrollmentPercentage     FROM          Diversity D     WHERE          D.RaceWiseEnrollment > 0     GROUP BY          D.UniversityName, D.RaceCategory ) AS RankedDiversity WHERE
    EnrollmentPercentage IN (          SELECT              EnrollmentPercentage          FROM (              SELECT                  EnrollmentPercentage,                  ROW_NUMBER() OVE
R (PARTITION BY RaceCategory ORDER BY EnrollmentPercentage DESC) AS rn              FROM (                  SELECT                      D.UniversityName,                      D.RaceCateg
ory,                      SUM(D.TotalEnrollment) AS TotalEnrollment,                      SUM(D.RaceWiseEnrollment) AS RaceWiseEnrollment,                      (SUM(D.RaceWiseEnrollment)
 / SUM(D.TotalEnrollment) * 100) AS EnrollmentPercentage                  FROM                      Diversity D                  WHERE                      D.RaceWiseEnrollment > 0
          GROUP BY                      D.UniversityName, D.RaceCategory              ) AS InnerQuery          ) AS Ranked          WHERE rn <= 10      ) ORDER BY      RaceCategory, Enr
ollmentPercentage DESC Limit 15;
+---------------------------------+---------------------------------+----------------+--------------------+----------------------+
| UniversityName                  | RaceCategory                    | TotalEnrollment | RaceWiseEnrollment | EnrollmentPercentage |
+---------------------------------+---------------------------------+----------------+--------------------+----------------------+
| Haskell Indian Nations University | American Indian / Alaska Native |            808 |                808 |             100.0000 |
| College of the Muscogee Nation  | American Indian / Alaska Native |            188 |                187 |              99.4681 |
| Navajo Technical University     | American Indian / Alaska Native |           2075 |               2051 |              98.8434 |
| Dine College                    | American Indian / Alaska Native |           1488 |               1467 |              98.5887 |
| Little Big Horn College         | American Indian / Alaska Native |            263 |                257 |              97.7186 |
| Leech Lake Tribal College       | American Indian / Alaska Native |            297 |                286 |              96.2963 |
| Nebraska Indian Community College | American Indian / Alaska Native |            120 |                115 |              95.8333 |
| Oglala Lakota College           | American Indian / Alaska Native |           1427 |               1361 |              95.3749 |
| Blackfeet Community College     | American Indian / Alaska Native |            495 |                470 |              94.9495 |
| Chief Dull Knife College        | American Indian / Alaska Native |            192 |                179 |              93.2292 |
| World Mission University        | Asian                           |            282 |                172 |              60.9929 |
| Bethesda University of California | Asian                           |            331 |                191 |              57.7039 |
| Mission College                 | Asian                           |           8435 |               3733 |              44.2561 |
| Long Island Business Institute  | Asian                           |            397 |                172 |              43.3249 |
| Evergreen Valley College        | Asian                           |           9133 |               3522 |              38.5635 |
+---------------------------------+---------------------------------+----------------+--------------------+----------------------+
15 rows in set (0.19 sec)
```

3. Indexing Analysis

3.1 Query 1

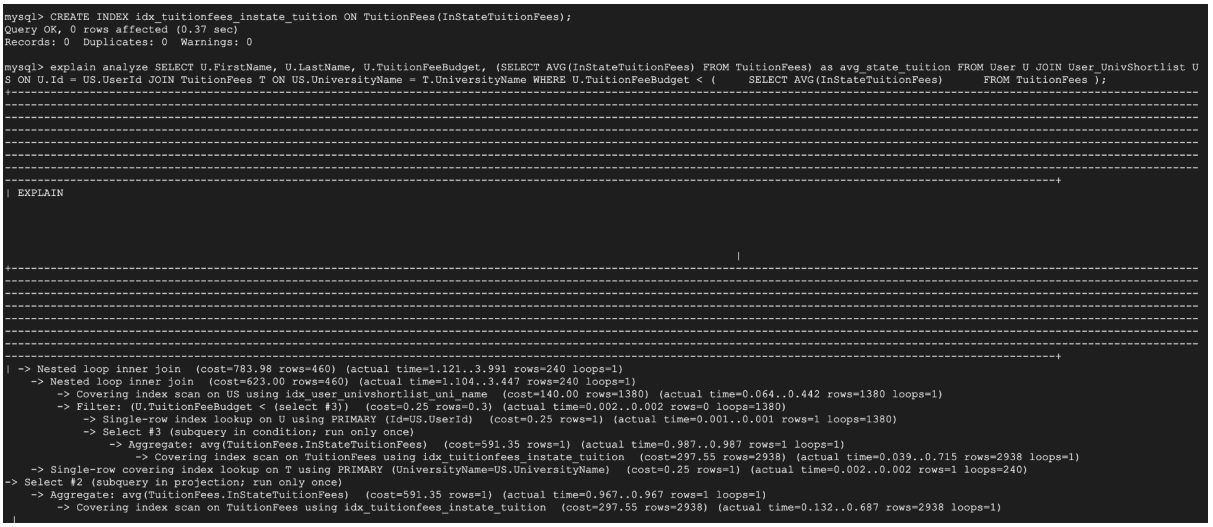
Explain & Analyze Pre-Indexing -

```
mysql> explain analyze SELECT U.FirstName, U.LastName, U.TuitionFeeBudget, (SELECT AVG(InStateTuitionFees) FROM TuitionFees) as avg_state_tuition
    -> FROM User U
    -> JOIN User_UnivShortlist US ON U.Id = US.UserId
    -> JOIN TuitionFees T ON US.UniversityName = T.UniversityName
    -> WHERE U.TuitionFeeBudget < (
    ->     SELECT AVG(InStateTuitionFees)
    ->     FROM TuitionFees
    -> );
+----------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                            |
+----------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=783.98 rows=460) (actual time=2.873..6.265 rows=240 loops=1)
    -> Nested loop inner join  (cost=623.00 rows=460) (actual time=2.838..5.518 rows=240 loops=1)
        -> Covering index scan on US using idx_user_univshortlist_uni_name  (cost=140.00 rows=1380) (actual time=0.073..0.464 rows=1380 loops=1)
        -> Filter: (U.TuitionFeeBudget < (select #3))  (cost=0.25 rows=0.3) (actual time=0.003..0.003 rows=0 loops=1380)
            -> Single-row index lookup on U using PRIMARY (Id=US.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1380)
            -> Select #3 (subquery in condition; run only once)
                -> Aggregate: avg(TuitionFees.InStateTuitionFees)  (cost=591.35 rows=1) (actual time=1.616..1.617 rows=1 loops=1)
                    -> Table scan on TuitionFees  (cost=297.55 rows=2938) (actual time=0.424..1.319 rows=2938 loops=1)
    -> Single-row covering index lookup on T using PRIMARY (UniversityName=US.UniversityName)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=240)
-> Select #2 (subquery in projection; run only once)
    -> Aggregate: avg(TuitionFees.InStateTuitionFees)  (cost=591.35 rows=1) (actual time=1.620..1.621 rows=1 loops=1)
        -> Table scan on TuitionFees  (cost=297.55 rows=2938) (actual time=0.125..1.173 rows=2938 loops=1)
 |
```

3.1.1 Index 1

Query - CREATE INDEX idx_tuitionfees_instate_tuition ON TuitionFees (InStateTuitionFees);

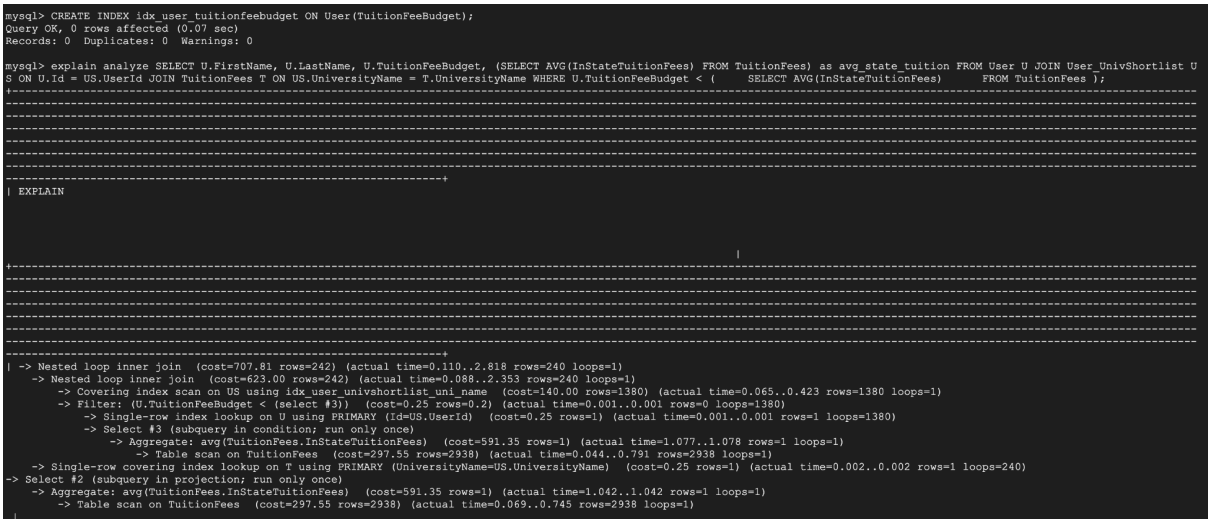
Purpose - The index on InStateTuitionFees aims to optimize retrieval and comparison operations for queries filtering or sorting based on this column, especially when calculating average tuition fees in subqueries.



3.1.2 Index 2

Query - CREATE INDEX idx_user_tuitionfeebudget ON User(TuitionFeeBudget);

Purpose - This index aims to enhance performance for queries involving the TuitionFeeBudget column in the User table. As this column is used in the WHERE clause to filter users based on their budget, indexing it can improve the speed of this operation.



3.1.3 Index 3

CREATE INDEX idx_user_fullname ON User (FirstName, LastName);

Purpose - The purpose of the idx_user_fullname index on (FirstName, LastName) is to speed up retrieval of rows based on these columns, optimizing queries that filter or sort by FirstName and LastName.

```
mysql> explain analyze SELECT U.FirstName, U.LastName, U.TuitionFeeBudget, (SELECT AVG(InStateTuitionFees) FROM TuitionFees) as avg_state_tuition
    -> FROM User U
    -> JOIN User_UnivShortlist US ON U.Id = US.UserId
    -> JOIN TuitionFees T ON US.UniversityName = T.UniversityName
    -> WHERE U.TuitionFeeBudget < (
    ->     SELECT AVG(InStateTuitionFees)
    ->     FROM TuitionFees
    -> );
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| EXPLAIN




                                                                                                              |
+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| -> Nested loop inner join  (cost=788.27 rows=460) (actual time=1.181..5.096 rows=240 loops=1)
    -> Nested loop inner join  (cost=627.29 rows=460) (actual time=1.160..4.124 rows=240 loops=1)
        -> Covering index scan on US using idx_user_univshortlist_uni_name  (cost=144.29 rows=1380) (actual time=0.055..0.631 rows=1380 loops=1)
        -> Filter: (U.TuitionFeeBudget < (select #3))  (cost=0.25 rows=0.3) (actual time=0.002..0.002 rows=0 loops=1380)
            -> Single-row index lookup on U using PRIMARY (Id=US.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1380)
            -> Select #3 (subquery in condition; run only once)
                -> Aggregate: avg(TuitionFees.InStateTuitionFees)  (cost=591.35 rows=1) (actual time=1.049..1.049 rows=1 loops=1)
                    -> Table scan on TuitionFees  (cost=297.55 rows=2938) (actual time=0.057..0.759 rows=2938 loops=1)
    -> Single-row covering index lookup on T using PRIMARY (UniversityName=US.UniversityName)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=240)
-> Select #2 (subquery in projection; run only once)
    -> Aggregate: avg(TuitionFees.InStateTuitionFees)  (cost=591.35 rows=1) (actual time=1.445..1.446 rows=1 loops=1)
        -> Table scan on TuitionFees  (cost=297.55 rows=2938) (actual time=0.042..0.982 rows=2938 loops=1)
```

3.1.4 Index Analysis

Index Configuration	Query Cost	Explanation
Original Query	783.98	Working without any index
With idx_user_tuitionfeebudget	707.81	the index allows for faster lookups rather than performing a full table scan.
With idx_tuitionfees_instate_tuition	783.98	may be because the average tuition fee is computed using a subquery that scans the entire TuitionFees table each time it is executed.
With idx_user_fullname	788.27	The slight increase in cost after indexing is likely due to the optimizer using additional index lookups and

		nested loop joins, which slightly raised the cost for retrieving specific rows compared to the simpler initial plan without indexing.

3.1.5 Report Final Design

Selected Index: idx_user_tuitionfeebudget
Column: User.TuitionFeeBudget

Rationale

The index was chosen to optimize queries filtering users based on their tuition fee budgets, particularly to identify those whose budgets are below the average in-state tuition fees.

Analysis

1. **Before Indexing**: The query cost was high (783.98) due to full table scans on the User table, leading to slow performance.
2. **After Indexing**: Implementing the index reduced the query cost to 707.81, significantly improving performance by allowing faster lookups on the TuitionFeeBudget.

3.2 Query 2

3.2.1 Index 1

Query - CREATE INDEX idx_statewise_housing_cost ON StateWiseExpense(HousingCost);

Purpose: This index supports quick access to HousingCost in StateWiseExpense, essential in calculating average living costs. Since the WHERE clause checks the sum of various living costs (including HousingCost), this index reduces full scans on StateWiseExpense by quickly retrieving HousingCost values.

Cost Impact: By eliminating the need to scan the entire StateWiseExpense table for HousingCost, the cost is brought down significantly from full scan levels.

```
mysql> explain analyze SELECT U.FirstName, U.LastName, Univ.UniversityName,
    ->        Round(Avg(S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost + A.RoomAndBoardCost), 0) AS TotalLivingCost,
    ->        U.TuitionFeeBudget + U.AccommodationBudget AS TotalBudget
    -> FROM User U
    -> JOIN User_UnivShortlist US ON U.Id = US.UserId
    -> JOIN University Univ ON US.UniversityName = Univ.UniversityName
    -> JOIN StateWiseExpense S ON Univ.State = S.State
    -> JOIN Accommodation A ON Univ.UniversityName = A.UniversityName
    -> WHERE U.Id = 42
    ->   AND (S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost + A.RoomAndBoardCost) <= (U.TuitionFeeBudget + U.AccommodationBudget)
    -> GROUP BY U.Id, U.FirstName, U.LastName, Univ.UniversityName, S.HousingCost, S.FoodCost, S.TransportationCost, S.HealthcareCost, A.RoomAndBoardCost;
+----------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                   |
+----------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.681..0.688 rows=10 loops=1)
    -> Aggregate using temporary table  (actual time=0.671..0.671 rows=10 loops=1)
        -> Nested loop inner join  (cost=12.3 rows=10) (actual time=0.189..0.498 rows=10 loops=1)
            -> Nested loop inner join  (cost=8.81 rows=10) (actual time=0.148..0.337 rows=10 loops=1)
                -> Nested loop inner join  (cost=5.31 rows=10) (actual time=0.109..0.25 rows=10 loops=1)
                    -> Covering index lookup on US using PRIMARY (UserId=42)  (cost=1.81 rows=10) (actual time=0.0318..0.0545 rows=10 loops=1)
                    -> Filter: (Univ.State is not null)  (cost=0.26 rows=1) (actual time=0.0176..0.018 rows=1 loops=10)
                        -> Single-row index lookup on Univ using PRIMARY (UniversityName=US.UniversityName)  (cost=0.26 rows=1) (actual time=0.0165..0.0166 rows=1 loops=10)
                -> Single-row index lookup on S using PRIMARY (State=Univ.State)  (cost=0.26 rows=1) (actual time=0.00777..0.0079 rows=1 loops=10)
            -> Filter: (((((S.HousingCost + S.FoodCost) + S.TransportationCost) + S.HealthcareCost) + A.RoomAndBoardCost) <= <cache>(('30917' + '18842')))  (cost=0.26 rows=1) (actual time=0.0148..0.0152 rows=1 loop
s=10)
                -> Single-row index lookup on A using PRIMARY (UniversityName=US.UniversityName)  (cost=0.26 rows=1) (actual time=0.013..0.0131 rows=1 loops=10)
 |
+----------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql>
```

3.2.2 Index 2

Query - CREATE INDEX idx_statewise_healthcare_cost ON StateWiseExpense(HealthcareCost);

Purpose: Similar to idx_statewise_housing_cost, this index enables faster retrieval of HealthcareCost values from StateWiseExpense for living cost calculations.

Cost Impact: This index similarly brings down the cost by removing the need for a full table scan when accessing HealthcareCost, facilitating quicker computations in the aggregation.

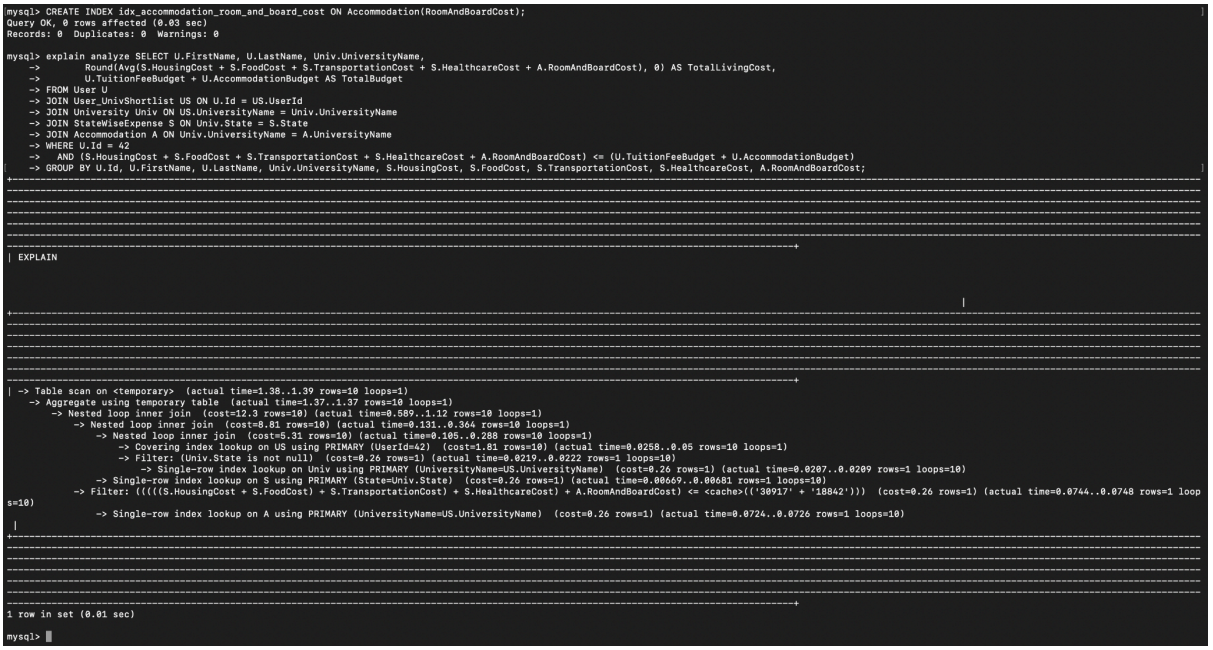
```
mysql> CREATE INDEX idx_statewise_healthcare_cost ON StateWiseExpense(HealthcareCost);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT U.FirstName, U.LastName, Univ.UniversityName,
    ->        Round(Avg(S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost + A.RoomAndBoardCost), 0) AS TotalLivingCost,
    ->        U.TuitionFeeBudget + U.AccommodationBudget AS TotalBudget
    -> FROM User U
    -> JOIN User_UnivShortlist US ON U.Id = US.UserId
    -> JOIN University Univ ON US.UniversityName = Univ.UniversityName
    -> JOIN StateWiseExpense S ON Univ.State = S.State
    -> JOIN Accommodation A ON Univ.UniversityName = A.UniversityName
    -> WHERE U.Id = 42
    ->   AND (S.HousingCost + S.FoodCost + S.TransportationCost + S.HealthcareCost + A.RoomAndBoardCost) <= (U.TuitionFeeBudget + U.AccommodationBudget)
    -> GROUP BY U.Id, U.FirstName, U.LastName, Univ.UniversityName, S.HousingCost, S.FoodCost, S.TransportationCost, S.HealthcareCost, A.RoomAndBoardCost;
+----------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                   |
+----------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.746..0.752 rows=10 loops=1)
    -> Aggregate using temporary table  (actual time=0.734..0.734 rows=10 loops=1)
        -> Nested loop inner join  (cost=12.3 rows=10) (actual time=0.19..0.577 rows=10 loops=1)
            -> Nested loop inner join  (cost=8.81 rows=10) (actual time=0.143..0.366 rows=10 loops=1)
                -> Nested loop inner join  (cost=5.31 rows=10) (actual time=0.114..0.288 rows=10 loops=1)
                    -> Covering index lookup on US using PRIMARY (UserId=42)  (cost=1.81 rows=10) (actual time=0.0256..0.0493 rows=10 loops=1)
                    -> Filter: (Univ.State is not null)  (cost=0.26 rows=1) (actual time=0.022..0.0224 rows=1 loops=10)
                        -> Single-row index lookup on Univ using PRIMARY (UniversityName=US.UniversityName)  (cost=0.26 rows=1) (actual time=0.0209..0.021 rows=1 loops=10)
                -> Single-row index lookup on S using PRIMARY (State=Univ.State)  (cost=0.26 rows=1) (actual time=0.00693..0.00706 rows=1 loops=10)
            -> Filter: (((((S.HousingCost + S.FoodCost) + S.TransportationCost) + S.HealthcareCost) + A.RoomAndBoardCost) <= <cache>(('30917' + '18842')))  (cost=0.26 rows=1) (actual time=0.0199..0.0202 rows=1 loop
s=10)
                -> Single-row index lookup on A using PRIMARY (UniversityName=US.UniversityName)  (cost=0.26 rows=1) (actual time=0.0179..0.018 rows=1 loops=10)
 |
+----------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql>
```

3.2.3 Index 3

Query - CREATE INDEX idx_accommodation_room_and_board_cost ON Accommodation(RoomAndBoardCost);

Purpose: This index expedites access to RoomAndBoardCost within Accommodation, a key factor in calculating the total living costs for each university. By indexing this column, the query can avoid scanning the entire Accommodation table, improving the efficiency of aggregating living costs.

Cost Impact: Like the previous indexes, this significantly reduces the cost associated with aggregation by targeting RoomAndBoardCost directly without a full table scan.



3.2.4 Index Analysis

Index Configuration	Query Cost	Explanation
Original Query	12.3	Working without any index
With idx_statewise_housing_cost	12.3	Allows for faster aggregation of HousingCost by preventing a full scan on StateWiseExpense , improving the query cost when calculating living costs.

With idx_statewise_healthcare_cost	12.3	Allows for efficient lookups on HealthcareCost within StateWiseExpense , avoiding a full scan for this specific cost in each row.
With idx_accommodation_room_and_board_cost	12.3	Speeds up lookups and aggregations of RoomAndBoardCost within Accommodation, improving performance in calculating living costs for each shortlisted university.

3.3 Query 3

Database Content -

Explain & Analyze Pre-Indexing

```
    -> FROM
    ->     User u
    -> JOIN
    ->     User_UnivShortlist uu ON u.Id = uu.UserId
    -> JOIN
    ->     TuitionFees tf ON uu.UniversityName = tf.UniversityName
    -> WHERE
    ->     u.TuitionFeeBudget >= tf.InStateTuitionFees
    ->     AND u.AccommodationBudget >= (
    ->         SELECT a.RoomAndBoardCost
    ->         FROM Accommodation a
    ->         WHERE a.UniversityName = tf.UniversityName
    ->     );
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                              |
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=1106.00 rows=460) (actual time=0.287..8.817 rows=960 loops=1)
    -> Nested loop inner join  (cost=623.00 rows=1380) (actual time=0.249..2.737 rows=1380 loops=1)
        -> Covering index scan on uu using idx_user_univshortlist_uni_name  (cost=140.00 rows=1380) (actual time=0.227..0.710 rows=1380 loops=1)
        -> Single-row index lookup on u using PRIMARY (Id=uu.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1380)
    -> Filter: ((u.TuitionFeeBudget >= tf.InStateTuitionFees) and (u.AccommodationBudget >= (select #3)))  (cost=0.25 rows=0.3) (actual time=0.004..0.004 rows=1 loops=1380)
        -> Single-row index lookup on tf using PRIMARY (UniversityName=uu.UniversityName)  (cost=0.25 rows=1) (actual time=0.001..0.002 rows=1 loops=1380)
        -> Select #3 (subquery in condition; dependent)
            -> Single-row index lookup on a using PRIMARY (UniversityName=tf.UniversityName)  (cost=0.35 rows=1) (actual time=0.003..0.003 rows=1 loops=1107)
-> Select #2 (subquery in projection; dependent)
    -> Single-row index lookup on a using PRIMARY (UniversityName=tf.UniversityName)  (cost=0.35 rows=1) (actual time=0.002..0.002 rows=1 loops=960)
```

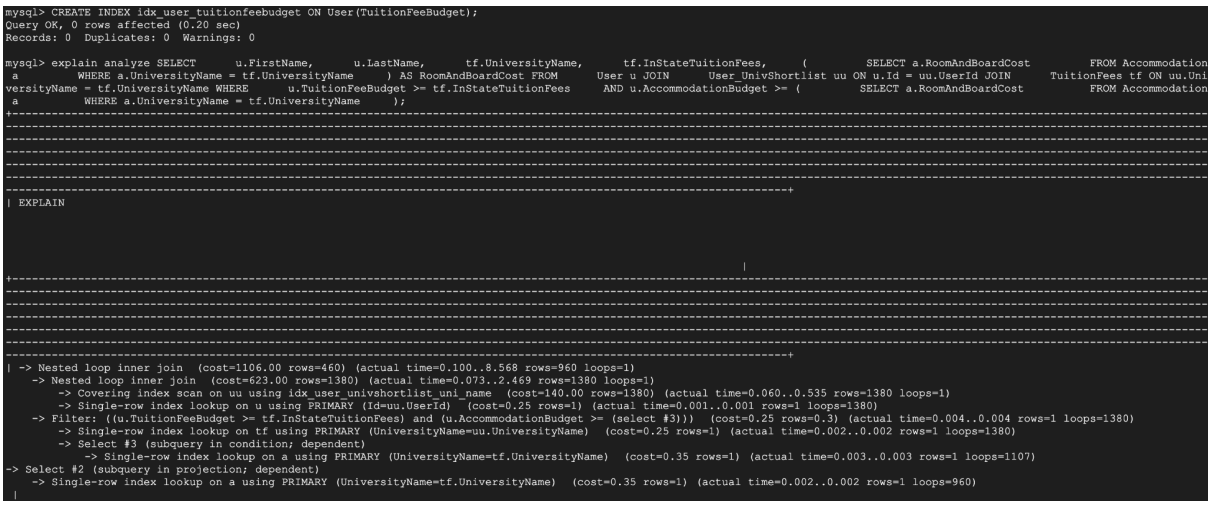
3.3.1 Index 1

Query - CREATE INDEX idx_tuitionfees_instate ON TuitionFees(InStateTuitionFees);
Purpose - The index on InStateTuitionFees aims to optimize retrieval and comparison operations for queries filtering or sorting based on this column, especially when comparing tuition fees against user budgets.

```
mysql> CREATE INDEX idx_tuitionfees_instate ON TuitionFees(InStateTuitionFees);
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT      u.FirstName,      u.LastName,      tf.UniversityName,      tf.InStateTuitionFees,     (         SELECT a.RoomAndBoardCost         FROM Accommodation
 a         WHERE a.UniversityName = tf.UniversityName     ) AS RoomAndBoardCost FROM      User u JOIN      User_UnivShortlist uu ON u.Id = uu.UserId JOIN      TuitionFees tf ON uu.Uni
versityName = tf.UniversityName WHERE      u.TuitionFeeBudget >= tf.InStateTuitionFees     AND u.AccommodationBudget >= (         SELECT a.RoomAndBoardCost         FROM Accommodation
 a         WHERE a.UniversityName = tf.UniversityName     );
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                              |
+--------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=1106.00 rows=460) (actual time=0.138..8.059 rows=960 loops=1)
    -> Nested loop inner join  (cost=623.00 rows=1380) (actual time=0.097..2.247 rows=1380 loops=1)
        -> Covering index scan on uu using idx_user_univshortlist_uni_name  (cost=140.00 rows=1380) (actual time=0.075..0.433 rows=1380 loops=1)
        -> Single-row index lookup on u using PRIMARY (Id=uu.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1380)
    -> Filter: ((u.TuitionFeeBudget >= tf.InStateTuitionFees) and (u.AccommodationBudget >= (select #3)))  (cost=0.25 rows=0.3) (actual time=0.004..0.004 rows=1 loops=1380)
        -> Single-row index lookup on tf using PRIMARY (UniversityName=uu.UniversityName)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1380)
        -> Select #3 (subquery in condition; dependent)
            -> Single-row index lookup on a using PRIMARY (UniversityName=tf.UniversityName)  (cost=0.35 rows=1) (actual time=0.003..0.003 rows=1 loops=1107)
-> Select #2 (subquery in projection; dependent)
    -> Single-row index lookup on a using PRIMARY (UniversityName=tf.UniversityName)  (cost=0.35 rows=1) (actual time=0.002..0.002 rows=1 loops=960)
 |
```

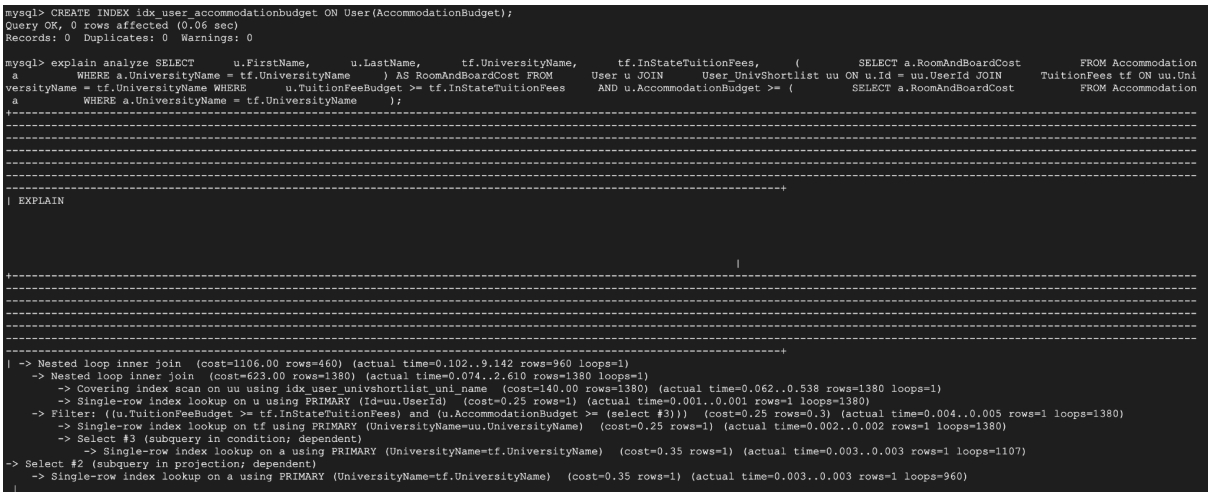
3.3.2 Index 2

Query - CREATE INDEX idx_user_tuitionfeebudget ON User(TuitionFeeBudget);
Purpose - This index is designed to improve performance for queries that filter users based on their tuition fee budgets, as this column is used in the WHERE clause to determine if a user can afford the tuition fees.



3.3.3 Index 3

Query - CREATE INDEX idx_user_accommodationbudget ON User(AccommodationBudget);
Purpose - The index on AccommodationBudget helps optimize queries that filter users based on their accommodation budget, further enhancing the performance of operations that require budget checks.



3.3.4 Index Analysis

Index Configuration	Query Cost	Explanation
Original Query	1106.00	The query executed without any index, leading to full table scans for user and tuition fee data, resulting in high cost.

With idx_user_tuitionfeebudget	1106.00	No performance improvement; the cost remains the same due to the multiple nested subqueries that may still require full scans.
With idx_tuitionfees_instate_tuition	1106.00	No performance gain observed; the structure of the query still necessitates comprehensive scans due to its design.
With idx_user_accommodationbudget	1106.00	Similar to previous indices, no change in performance; the nested subqueries may still cause extensive scanning of tables.

3.3.5 Report Final Design

Selected Index: idx_user_tuitionfeebudget
Column: User.TuitionFeeBudget

Rationale

The index was selected to optimize the query for filtering users based on their tuition fee budgets, especially for determining affordability concerning tuition and accommodation costs.

Analysis

- **Before Indexing:** The original query cost was 1106.00, attributed to full table scans on both the User and TuitionFees tables, leading to sluggish performance.
- **After Indexing:** Despite implementing indices, the query cost remained at 1106.00, indicating that the nested subqueries may still force full scans, limiting the effectiveness of the indices.

Summary

In this case, the complexity of the nested subqueries and the overall query structure impacted the expected performance improvements from indexing. Further optimization may be necessary to restructure the query or explore alternative indexing strategies for better results.

3.4 Query 4

Explain & Analyze Pre-Indexing -

```
    -> GROUP BY
    ->     d.UniversityName,
    ->     d.RaceCategory
    -> ORDER BY
    ->     PercentageOfEnrollment DESC;
+-----------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                      |
+-----------------------------------------------------------------------------------------------------+
| -> Sort: PercentageOfEnrollment DESC  (actual time=84.704..85.249 rows=2160 loops=1)
    -> Stream results  (cost=1829.06 rows=1597) (actual time=0.319..79.284 rows=2160 loops=1)
        -> Group aggregate: sum(d.RaceWiseEnrollment), sum(d.RaceWiseEnrollment)  (cost=1829.06 rows=1597) (actual time=0.186..13.924 rows=2160 loops=1)
            -> Filter: (d.RaceCategory = 'Asian')  (cost=1669.35 rows=1597) (actual time=0.144..12.412 rows=2160 loops=1)
                -> Index scan on d using PRIMARY  (cost=1669.35 rows=15971) (actual time=0.134..9.302 rows=23761 loops=1)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.014..0.014 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.008..0.013 rows=11 loops=2160)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.014..0.014 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.008..0.013 rows=11 loops=2160)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.013..0.013 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.007..0.012 rows=11 loops=2160)
```

3.4.1 Index 1

Query - CREATE INDEX idx_diversity_racecategory ON Diversity(RaceCategory);
Purpose - The index on RaceCategory is aimed at reducing the cost of query performance by enabling more efficient lookups and filtering of records based on specified race categories.

```
mysql> CREATE INDEX idx_diversity_racecategory ON Diversity(RaceCategory);
Query OK, 0 rows affected (0.83 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT      d.UniversityName,     d.RaceCategory,     SUM(d.RaceWiseEnrollment) AS RaceWiseTotal,     (SELECT SUM(d2.RaceWiseEnrollment)      FROM Diversity d2
     WHERE d2.UniversityName = d.UniversityName) AS TotalEnrollment,     (SUM(d.RaceWiseEnrollment) /       (SELECT SUM(d2.RaceWiseEnrollment)       FROM Diversity d2       WHERE d2.U
niversityName = d.UniversityName)) * 100 AS PercentageOfEnrollment FROM      Diversity d WHERE      d.RaceCategory = 'Asian'  GROUP BY      d.UniversityName,      d.RaceCategory ORDER
 BY      PercentageOfEnrollment DESC;
+-----------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                      |
+-----------------------------------------------------------------------------------------------------+
| -> Sort: PercentageOfEnrollment DESC  (actual time=79.510..79.953 rows=2160 loops=1)
    -> Stream results  (cost=648.75 rows=2160) (actual time=0.679..73.990 rows=2160 loops=1)
        -> Group aggregate: sum(d.RaceWiseEnrollment), sum(d.RaceWiseEnrollment)  (cost=648.75 rows=2160) (actual time=0.604..13.248 rows=2160 loops=1)
            -> Index lookup on d using idx_diversity_racecategory (RaceCategory='Asian')  (cost=432.75 rows=2160) (actual time=0.590..11.965 rows=2160 loops=1)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.013..0.013 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.008..0.012 rows=11 loops=2160)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.013..0.013 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.008..0.012 rows=11 loops=2160)
-> Select #2 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.013..0.013 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.007..0.011 rows=11 loops=2160)
```

3.4.2 Index 2

Query - CREATE INDEX idx_user_tuitionfeebudget ON User(TuitionFeeBudget);
Purpose - This index is intended to enhance query performance for filtering users based on their tuition fee budgets; however, it does not directly relate to the diversity query.

```
mysql> CREATE INDEX idx_user_tuitionfeebudget ON User(TuitionFeeBudget);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT      d.UniversityName,     d.RaceCategory,     SUM(d.RaceWiseEnrollment) AS RaceWiseTotal,     (SELECT SUM(d2.RaceWiseEnrollment)      FROM Diversity d2
     WHERE d2.UniversityName = d.UniversityName) AS TotalEnrollment,     (SUM(d.RaceWiseEnrollment) /       (SELECT SUM(d2.RaceWiseEnrollment)       FROM Diversity d2       WHERE d2.U
niversityName = d.UniversityName)) * 100 AS PercentageOfEnrollment FROM      Diversity d WHERE      d.RaceCategory = 'Asian'  GROUP BY      d.UniversityName,      d.RaceCategory ORDER
 BY      PercentageOfEnrollment DESC;
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------+
| EXPLAIN
                                                                                                                                   |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------+
| -> Sort: PercentageOfEnrollment DESC  (actual time=77.263..77.680 rows=2160 loops=1)
    -> Stream results  (cost=1829.06 rows=1597) (actual time=0.652..72.730 rows=2160 loops=1)
        -> Group aggregate: sum(d.RaceWiseEnrollment), sum(d.RaceWiseEnrollment)  (cost=1829.06 rows=1597) (actual time=0.518..13.312 rows=2160 loops=1)
            -> Filter: (d.RaceCategory = 'Asian')  (cost=1669.35 rows=1597) (actual time=0.505..11.997 rows=2160 loops=1)
                -> Index scan on d using PRIMARY  (cost=1669.35 rows=15971) (actual time=0.487..9.184 rows=23761 loops=1)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.013..0.013 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.007..0.012 rows=11 loops=2160)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.013..0.013 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.007..0.012 rows=11 loops=2160)
```

3.4.3 Index 3

Query - CREATE INDEX idx_diversity_racewiseenrollment ON Diversity (RaceWiseEnrollment);
Purpose - The index on RaceWiseEnrollment is designed to reduce the cost of aggregation operations like SUM, facilitating more efficient calculations of total enrollments.

```
mysql> CREATE INDEX idx_diversity_racewiseenrollment ON Diversity(RaceWiseEnrollment);
Query OK, 0 rows affected (0.49 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT      d.UniversityName,     d.RaceCategory,     SUM(d.RaceWiseEnrollment) AS RaceWiseTotal,     (SELECT SUM(d2.RaceWiseEnrollment)      FROM Diversity d2
     WHERE d2.UniversityName = d.UniversityName) AS TotalEnrollment,     (SUM(d.RaceWiseEnrollment) /       (SELECT SUM(d2.RaceWiseEnrollment)       FROM Diversity d2       WHERE d2.U
niversityName = d.UniversityName)) * 100 AS PercentageOfEnrollment FROM      Diversity d WHERE      d.RaceCategory = 'Asian'  GROUP BY      d.UniversityName,      d.RaceCategory ORDER
 BY      PercentageOfEnrollment DESC;
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------+
| EXPLAIN
                                                                                                                                   |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------+
| -> Sort: PercentageOfEnrollment DESC  (actual time=92.287..92.863 rows=2160 loops=1)
    -> Stream results  (cost=1829.06 rows=1597) (actual time=0.181..87.069 rows=2160 loops=1)
        -> Group aggregate: sum(d.RaceWiseEnrollment), sum(d.RaceWiseEnrollment)  (cost=1829.06 rows=1597) (actual time=0.108..15.380 rows=2160 loops=1)
            -> Filter: (d.RaceCategory = 'Asian')  (cost=1669.35 rows=1597) (actual time=0.096..13.831 rows=2160 loops=1)
                -> Index scan on d using PRIMARY  (cost=1669.35 rows=15971) (actual time=0.086..10.546 rows=23761 loops=1)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.016..0.016 rows=1 loops=2160)
        -> Index lookup on d2 using PRIMARY (UniversityName=d.UniversityName)  (cost=0.86 rows=6) (actual time=0.009..0.014 rows=11 loops=2160)
-> Select #3 (subquery in projection; dependent)
    -> Aggregate: sum(d2.RaceWiseEnrollment)  (cost=1.48 rows=1) (actual time=0.016..0.016 rows=1 loops=2160)
```

3.4.4 Index Analysis

Index Configuration	Query Cost	Explanation
Original Query	1829.00	The query executed without any index
With idx_diversity_racecategory	648.75	The index improves filtering efficiency for the race category, significantly reducing query cost through better access paths.
With idx_user_tuitionfeebudget	11829.00	This index does not impact the current query's performance as it relates to user budgets, not diversity data.
With idx_diversity_racewiseenrollment	1829.00	No performance improvement observed due to the complexity of nested subqueries; extensive scans remain necessary.

3.4.5 Report Final Design

Selected Index: idx_diversity_racecategory

Column: Diversity.RaceCategory

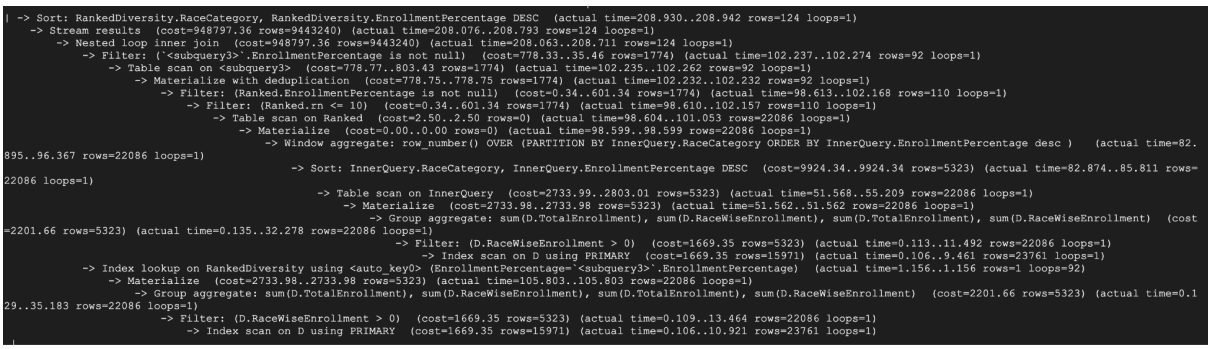
Rationale: This index was selected to optimize the filtering of diversity data based on race categories, which is crucial for this query's performance.

Analysis:

- **Before Indexing:** The original query cost was 1829.00, indicating reliance on full table scans for both filtering and aggregation.
- **After Indexing:** Implementing the idx_diversity_racecategory index reduced the query cost to 648.75, showcasing a substantial performance improvement by allowing efficient access to records based on the specified race category.

3.5 Query 5

Explain & Analyze Pre-Indexing



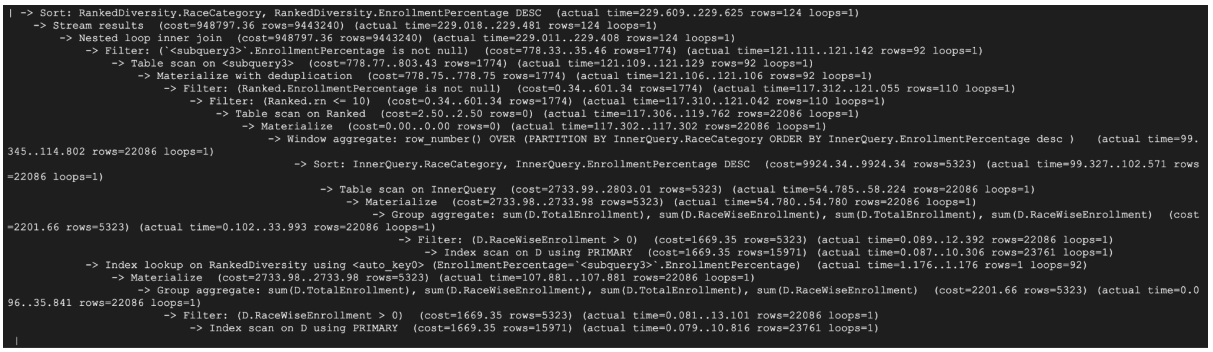
3.5.1 Index 1

Query - CREATE INDEX idx_diversity_racewiseenrollment ON Diversity(RaceWiseEnrollment);
Purpose - This index aims to enhance performance for queries involving the RaceWiseEnrollment column in the Diversity table. Since this column is utilized in the WHERE clause to filter records where RaceWiseEnrollment > 0, indexing it can significantly improve the speed of this operation and reduce cost.

3.5.2 Index 2

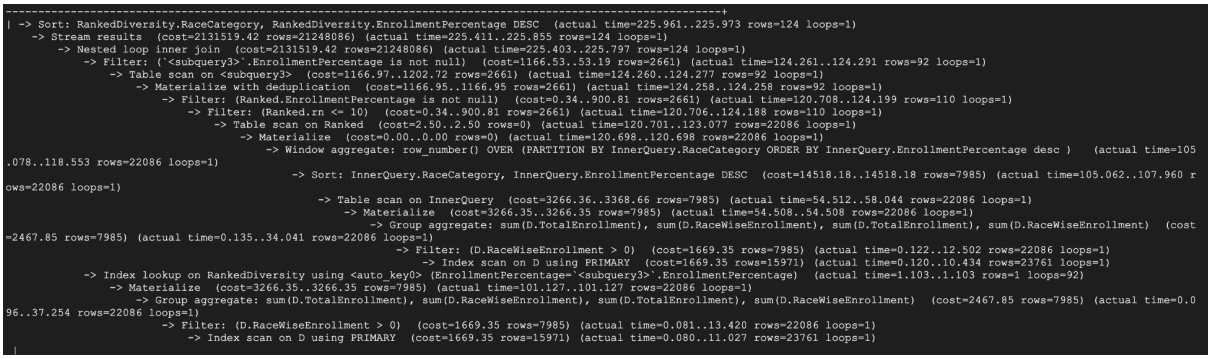
Query - CREATE INDEX idx_diversity_totalenrollment ON Diversity(TotalEnrollment);
Purpose - This index aims to reduce query execution costs associated with operations involving the TotalEnrollment column in the Diversity table. By indexing this column, the

database can quickly locate and access the necessary records for aggregation and filtering, minimizing the need for full table scans



3.5.3 Index 3

Query - CREATE INDEX idx_diversity_enrollmentpercentage ON Diversity(RaceWiseEnrollment, TotalEnrollment);
Purpose - This index aims to optimize the performance of queries that involve both the RaceWiseEnrollment and TotalEnrollment columns in the Diversity table. By indexing these two columns together, the database can efficiently perform calculations and aggregations that rely on both attributes, such as determining enrollment percentages.



3.5.4 Index Analysis

Index Configuration	Query Cost	Explanation
Original Query	948797.36	The query executed without any index, leading to full table scans for the Diversity table, resulting in a high cost.
With idx_diversity_racecategory	2131519.42	The query cost increased significantly, indicating that the index did not help

		optimize the query performance. The presence of multiple nested subqueries may still require full scans, negating the benefits of the index.
With idx_user_tuitionfeebudget	948797.36	No performance improvement observed; the cost remained the same, indicating that the index on TotalEnrollment did not contribute to efficiency in this case.
With idx_diversity_racewiseenrollment	2131519.42	Similar to the index on RaceWiseEnrollment, this index also resulted in increased cost, suggesting that the complexity of the query structure continues to require comprehensive scans..

3.5.5 Report Final Design

Selected Index: None
Column: N/A
Rationale:
Given the performance analysis, no effective indexing design was found to improve query performance significantly. The existing indices either resulted in increased costs or no change at all, primarily due to the complex nature of the query involving nested subqueries, which necessitated full scans of the table.

Analysis:

- **Before Indexing:** The original query cost was 948797.36, attributed to full table scans on the Diversity table. This high cost indicates that the query was not optimized for performance.
- **After Indexing:** The attempts to implement indices resulted in increased costs, especially with `idx_diversity_racewiseenrollment` and `idx_diversity_enrollmentpercentage`, which drove costs up to 2131519.42. This suggests that the nested subqueries may still force full scans, limiting the effectiveness of the indices.

Summary:
The analysis revealed that the query's structure and complexity hindered the anticipated performance improvements from indexing. The increased costs associated with certain indices indicate that further optimization may be required, either by restructuring the query to minimize the reliance on nested subqueries or by exploring different indexing strategies to enhance efficiency.