Advanced Database Feature

Project Title: GlobalScholar: International Student Financial Navigator

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

1. Transactions

Recommend Universities
Description - Calculate diversity score and match score based on state preference and return top 10 universities with highest match score
Used in python (app.py) - @app.route('/recommend_universities', methods=['POST'])

User Budget
Description - This transaction fetches users whose tuition budgets are below the average in-state tuition fees, ensuring data consistency with the REPEATABLE READ isolation level to prevent non-repeatable reads during execution.
Used in python (app.py) - @app.route('/transaction-result', methods=['GET'])

2. Stored Procedures

With Advanced Query - GetTopDiverseUniversities
Description - This procedure calculates and ranks the top 10 universities by enrollment percentage for a specified race category, providing valuable insights into university diversity.
Used in python (app.py) - @app.route('/top-diverse-universities', methods=['GET'])
Code -

```
DELIMITER //

CREATE PROCEDURE GetTopDiverseUniversities(IN p_RaceCategory
VARCHAR(100), IN p_TopN INT)
BEGIN
    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,
        ROW_NUMBER() OVER (PARTITION BY D.RaceCategory ORDER BY
(SUM(D.RaceWiseEnrollment) / SUM(D.TotalEnrollment) * 100) DESC) AS rn
        FROM
        Diversity D
        WHERE
        D.RaceWiseEnrollment > 0
        AND (p_RaceCategory IS NULL OR D.RaceCategory = p_RaceCategory)
        GROUP BY
        D.UniversityName, D.RaceCategory
        ) AS RankedDiversity
        WHERE
        rn <= p_TopN
        ORDER BY
        RaceCategory, EnrollmentPercentage DESC;
END //

DELIMITER ;

-- Get top 10 diverse universities for all race categories
CALL GetTopDiverseUniversities(NULL, 10);
-- Get top 5 diverse universities for a specific race category
CALL GetTopDiverseUniversities('Asian', 5);
```

With Advanced Query - GetUserLivingCosts
Description - This procedure calculates the average living costs for universities shortlisted by a specific user and checks if they fall within their budget.

Used in python (app.py) - @app.route('/getUserLivingCosts/<int:user_id>', methods=['GET'])

Code -

```
DELIMITER $$
CREATE PROCEDURE GetUserLivingCosts(IN userId INT)
BEGIN
   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 = userId
      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;
END$$
DELIMITER ;
```

GetMatchingUniversities
Description - This stored procedure lists all universities matching a user's tuition and accommodation budget.
Used in python (app.py) - @app.route('/matching-universities/<int:user_id>', methods=['GET'])
Code -

```
DELIMITER $$

CREATE PROCEDURE GetMatchingUniversities(IN userId INT)
BEGIN
        DECLARE userTuitionBudget INT;
        DECLARE userAccommodationBudget INT;

        -- Fetch user budget
        SELECT TuitionFeeBudget, AccommodationBudget
        INTO userTuitionBudget, userAccommodationBudget
        FROM User
        WHERE Id = userId;

        -- Fetch universities matching criteria
        SELECT U.UniversityName, T.InStateTuitionFees, A.RoomAndBoardCost
        FROM University U
        JOIN TuitionFees T ON U.UniversityName = T.UniversityName
        JOIN Accommodation A ON U.UniversityName = A.UniversityName
        WHERE T.InStateTuitionFees <= userTuitionBudget
        AND A.RoomAndBoardCost <= userAccommodationBudget;
END$$

DELIMITER ;
```

GetUniversityDetails
Description - This procedure retrieves the university name, room and board cost, and out-of-state tuition fees for universities shortlisted by a specific user.

Used in python (app.py) - @app.route('/getUniversityDetails/<int:Id>', methods=['GET'])

Code - Create Procedure

```
(
   IN UserId INT
)
BEGIN
   SET @sql_query = CONCAT(
      'SELECT a.UniversityName,
            a.RoomAndBoardCost,
            t.OutOfStateTuitionFees
       FROM Accommodation a
       JOIN TuitionFees t
       ON a.UniversityName = t.UniversityName
       WHERE a.UniversityName IN (SELECT UniversityName FROM
User_UnivShortlist WHERE UserId = ', UserId, ')'
   );
   PREPARE stmt FROM @sql_query;
   EXECUTE stmt;
   DEALLOCATE PREPARE stmt;
END
```

GetAllUniversityDetails

Description - This procedure retrieves the university name, room and board cost, and out-of-state tuition fees.
Used in python (app.py) - @app.route('/getUniversityDetails/<int:Id>',methods=['GET'])
Code -

```
CREATE PROCEDURE `GetAllUniversityDetails`()
BEGIN
   SELECT
      a.UniversityName,
      a.RoomAndBoardCost,
      t.OutOfStateTuitionFees
   FROM
      Accommodation a
   JOIN
      TuitionFees t
   ON
      a.UniversityName = t.UniversityName;
END
```

3. Triggers

Logs trigger

Description - This trigger automatically logs every INSERT operation on the User table into the UserAudit table, recording the user ID, action type, and timestamp for auditing purposes.
Used in python (app.py) - @app.route('/user-logs', methods=['GET'])
Code -

```
DELIMITER $$

CREATE TRIGGER AfterUserInsert
AFTER INSERT ON User
FOR EACH ROW
BEGIN
   INSERT INTO UserAudit (UserId, Action)
   VALUES (NEW.Id, 'INSERT');
END$$

DELIMITER ;
```

4. Constraints
    a. mysql> Show create table Accommodation;

```
 PRIMARY KEY (`UniversityName`),
 CONSTRAINT `Accommodation_ibfk_1` FOREIGN KEY
(`UniversityName`) REFERENCES `University` (`UniversityName`) ON
DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `Accommodation_chk_1` CHECK
((`RoomAndBoardCost` between 200 and 22000))
```

    b. mysql> Show create table TuitionFees;

```
 PRIMARY KEY (`UniversityName`),
 CONSTRAINT `TuitionFees_ibfk_1` FOREIGN KEY (`UniversityName`)
 REFERENCES `University` (`UniversityName`) ON DELETE CASCADE
 ON UPDATE CASCADE,
 CONSTRAINT `TuitionFees_chk_1` CHECK ((`InStateTuitionFees` <=
60000)),
 CONSTRAINT `TuitionFees_chk_2` CHECK ((`OutOfStateTuitionFees`
<= 60000))
```

    c. mysql> Show create table User

```
PRIMARY KEY (`Id`)
```

    d. mysql> Show create table User_UnivShortlist;

```
 PRIMARY KEY (`UserId`,`UniversityName`),
 KEY `idx_user_univshortlist_uni_name` (`UniversityName`),
CONSTRAINT `fk_university_name` FOREIGN KEY (`UniversityName`)
REFERENCES `University` (`UniversityName`) ON DELETE CASCADE
ON UPDATE CASCADE,
 CONSTRAINT `fk_user_id` FOREIGN KEY (`UserId`) REFERENCES
`User` (`Id`)
```

e. mysql> Show create table University;
PRIMARY KEY (`UniversityName`)

f. mysql> Show create table Diversity;
PRIMARY KEY (`UniversityName`,`RaceCategory`),
CONSTRAINT `Diversity_ibfk_1` FOREIGN KEY (`UniversityName`)
REFERENCES `University` (`UniversityName`) ON DELETE CASCADE
ON UPDATE CASCADE)