MET CS 669 Database Design and Implementation for Business
Term Project Iteration 6

Name: Jacky Cheng

Table of Contents

Project Direction Overview

Inspired by discussions with a friend who had recently started a tutoring business, I recognized the importance of personalized tutoring in today's dynamic educational environment. 'AchieveMe Tutor' solves this problem by empowering tutors to develop tailored strategies based on comprehensive insights into student progress. It emphasizes students' achievements while also pinpointing areas that require intervention. By identifying students' strengths and areas of improvement, the app helps tutors establish realistic milestones and adjust their approach in real-time. In this way, each coaching session evolves into a purposeful engagement that specifically addresses individual struggles and builds on strengths. This not only enhances the effectiveness of each tutoring session but also optimizes the tutor's time and efforts, ensuring holistic academic development.

Example Scenario:

Aiden, an eighth-grader initially disheartened by his struggles in Geometry, was particularly challenged by "Circle Theorems". Mr.Chen, his tutor, used "AchieveMe Tutor" to get a clear picture of his struggles. The app's in-depth analytics highlighted that Aiden was often mixing up concepts within Circle Theorems. Based on this, Mr.Chen adjusted his lessons, introducing Aiden to interactive online tools suggested by the app. They focused on visualizing circle properties, making it easier for Aiden to grasp. In a few weeks, with the app's guidance and Mr.Chen's adapted teaching, Aiden began to tackle Geometry problems he once found difficult.

Key Features:

- **Adaptive Learning Analytics**: The app continually assesses student progress and identifies areas that need attention, enabling tutors to adjust their approach.
- **Milestone Planne**r: Tutors can set, modify, and track academic milestones based on student performance and progress.
- **Strengths and Weaknesses Dashboard:** Visually highlights areas where students excel and areas where they need additional support.
- **Course Customizer:** Enables tutors to craft courses using past performance data, ensuring individualized content for every student.
- **Progress Reports**: Generate weekly,monthly or yearly detailed reports to gain insights into student performance.
- **Feedback Integration**: Utilize student feedback to refine teaching methods or curriculum focus.
- **Resource Recommendations**: Suggests targeted resources or exercises based on identified areas of difficulty.

Use Cases and Fields

Use Case:Account Registration

1. A user (tutor/student/parent) visits the "AchieveMe Tutor" application or website.
2. The application presents an option to register or log in.
3. For new users, they click on 'Register' and are prompted to select their role (tutor/student/parent).
4. Depending on the chosen role, the user is presented with a registration form to fill in their details.
5. After entering the required information and setting up security measures (like a password), they submit the form. The system validates the information, creates an account in the database, and confirms registration to the user.

Field	What it Stores	Why it's Needed
UserID	Unique Identifier for each user.	To differentiate each user and facilitate personalized experiences.
Role	Role fo the User(tutor/student/parent).	To customize user experience and provide access to relevant features.
FirstName	First name of the user.	Displaying the person’s name on screens and addressing them when sending them message.
LastName	Last name of the user.	Displaying the person’s name on screens and addressing them when sending them message.
Email	Email address of the user.	For communication, notifications, and as a unique identifier for login purposes.
Password	Password for the user.	For security.

Use Case:Tutor Session Creation & Student Progress Tracking

1. A tutor logs into the "AchieveMe Tutor" application.
2. The tutor selects a class/student and initiates a new tutoring session.
3. During the session, the tutor inputs data regarding the student's performance, such as topics covered, areas of strength, and areas needing improvement.
4. At the end of the session, the tutor saves this information to the student's progress profile in the database.

Field	What it Stores	Why it’s Needed
SesionID	Unique Identifier for each Session.	To differentiate each session and trace back details if needed.
StudentName	Name of the Student.	To associate the session with the specific student
Date	Date of the tutoring session.	Helps in tracking student

		progress over time.
TopicsCovered	List of topics that were covered.	Provides a summary of what was taught during the session.
Strengths	Areas where the student performed well.	For positive reinforcements and to know what topics dont't require as much attention.
Improvements	Areas where the student needs help	To focus on these areas in the next sessions and tailor teaching methods accordingly.

Use Case: Setting Milestone & Tracking Achievements

1. The tutor selects a student and accesses their profile.
2. Tutor sets specific milestones for the student based on previous sessions and desired outcomes.
3. As sessions proceed, achievements toward these milestones are updated in real-time. From the database's perspective, this use case involves storing milestone data and achievement metrics.

Field	What it Stores	Why it's Needed
MilestoneID	Unique Identifier for each milestone.	To differentiate each milestone set for various students.
StudentName	Name of the Student.	To associate the milestone with a particular student.
Description	Brief about the milestone, for example "Master Circle Theorems".	To give clarity on what the expected achievement is.
ExpectedDate	The target date ti achieve the milestone.	Helps in setting at timeline and tracking progress accordingly.
AchievementStatus	Whether the milestone is achieved or not.	For real time updates on the student's progress and to know what's pending.

Use Case: Goal Setting and Mile Stone Achievements

1. Tutors and students collaboratively set academic goals within the app.
2. The system records the goal, along with a deadline or expected achievement date.
3. Periodic reminders are sent to students as the goal's deadline approaches.
4. Once achieved, students mark the goal as completed, and the tutor verifies.
5. Achievements are visually presented in the student's profile, and notifications are sent to both students and their parents.

Field	What is Stores	Why it's Needed
GoalID	Unique identifier for the goal.	To manage individual goals.

Description	Detailed information on the goal.	To inform students what they're working towards.
Deadline	Expected achievement date.	To keep students on track and send reminders.
Status	Current state of the goal.	To differentiate between ongoing and achieved goals.
VerifiedBy	Tutor who confirms the goal.	To ensure goals are genuinely achieved.

Use Case: Assignment Submission& Review

1. After a session, the tutor assigns homework or tasks to a student within the app.
2. The student receives a notification about the new assignment.
3. Upon completion, the student submits the assignment through the app.
4. The tutor gets notified of the submission, reviews it, and provides feedback or grades within the app.
5. Feedback is instantly available to the student for review.

Field	What it Stores	Why it's Needed
AssignmentID	Unique identifier for the assignment	To manage individual assignments.
TutorID	Identifier of the assigning tutor.	To link the assignment to the tutor.
StudentID	Identifier of the assigned student.	To link the assignment to the student.
Submission	Content or files of the submission.	To store the student's work for review.
Feedback	Tutor's remarks and comments.	To guide and inform the student's performance and learning.
Grade	Score or grade for the assignment.	To evaluate and assess the student's progress.

Use Case: Parent Monitoring and Reporting

1. Parents log into the "AchieveMe Tutor" app with their unique credentials.
2. They navigate to their child's profile and access a dashboard summarizing performance, milestones, and upcoming sessions.
3. Parents can generate or request detailed reports, which chart progress over a selected timeframe.
4. Reports are either available instantly or sent to the parent's registered email.
5. Parents can schedule meetings with tutors based on these insights.

Field	What it Stores	Why it's Needed
ParentID	Unique identifier for the parent.	To give parents access and link them to their child.
StudentID	Identifier of their child.	To access relevant data for the specific student.
ReportID	Identifier for generated reports.	To manage and store different reports.
ReportData	Data points, charts, graphs.	To visualize and provide detailed insights on progress.
MeetingDate	Scheduled meetings with tutors.	To facilitate and record parent-tutor interactions.

Use Case: Resource Library Access

1. Tutors or administrators can upload educational resources to a shared library within the app.
2. Students search the library using filters like subject, grade level, or topic.
3. They access resources, which can include video lectures, PDFs, quizzes, etc.
4. Students can bookmark or save essential resources for easier access.
5. Tutors can recommend specific resources to individual students or groups.

Field	What it Stores	Why it's Needed
ResourceID	Unique identifier for the resource.	To manage and categorize resources.
UploadDate	Date of resource upload.	To organize and potentially archive older resources.
ResourceType	Type (video, PDF, quiz, etc.)	To filter and search based on resource type.
RecommendedTo	List of students for recommendation.	To tailor resource suggestions to specific students.
Subject	Academic subject of the resource.	To categorize and facilitate easy searching.

Use Case: FeedBack and Ratings

1. After a tutoring session, both student and tutor are prompted by the application to leave feedback and a rating.
2. The student rates the session and provides comments on what they found helpful or areas they'd like to focus on next time.
3. The tutor can leave feedback on the student's performance, noting areas of improvement and strengths.
4. Ratings and feedback are stored and can be viewed in the session history.

Field	What it Stores	Why it’s Needed
FeedbackID	Unique identifier for the feedback.	To manage individual feedback entries.
SessionID	Identifier linking it to a session.	To relate feedback to specific sessions.
Rating	Numerical rating (e.g., out of 5).	To get a quick understanding of session effectiveness.
Comment	Detailed feedback or comments.	To provide context and actionable insights for improvement.
DateGiven	Date the feedback was provided.	To keep track of feedback over time.

Structural Database Rules

Use Case:Account Registration

1. A user (tutor/student/parent) visits the "AchieveMe Tutor" application or website.
2. The application presents an option to register or log in.
3. For new users, they click on 'Register' and are prompted to select their role (tutor/student/parent).
4. Depending on the chosen role, the user is presented with a registration form to fill in their details.
5. After entering the required information and setting up security measures (like a password), they submit the form.
6. The system validates the information, creates an account in the database, and confirms registration to the user.

From this use case the entities I identified are as follows:

- **User:**This can represent a tutor, student, or parent.

Use Case:Tutor Session Creation & Student Progress Tracking

1. A tutor logs into the "AchieveMe Tutor" application.
2. The tutor selects a class/student and initiates a new tutoring session.
3. During the session, the tutor inputs data regarding the student's performance, such as topics covered, areas of strength, and areas needing improvement.
4. At the end of the session, the tutor saves this information to the student's progress profile in the database.

From this use case the entities I identified are as follows:

- **Session**: This Represents the tutoring session.
- **Progress:**This captures the student’s performance data during the session

Use Case: Setting Milestone & Tracking Achievements

1. The tutor selects a student and accesses their profile.
2. Tutor sets specific milestones for the student based on previous sessions and desired outcomes.
3. As sessions proceed, achievements toward these milestones are updated in real-time. From the database's perspective, this use case involves storing milestone data and achievement metrics.

From this use case the entities I identified are as follows:

- **Milestone:** This represents academic goals set for students.
- **Achievement**: This captures the progress or attainment of these milestones

Use Case: Goal Setting and Mile Stone Achievements

1. Tutors and students collaboratively set academic goals within the app.
2. The system records the goal, along with a deadline or expected achievement date.
3. Periodic reminders are sent to students as the goal's deadline approaches.
4. Once achieved, students mark the goal as completed, and the tutor verifies.
5. Achievements are visually presented in the student's profile, and notifications are sent to both students and their parents.

From this use case the entities I identified are as follows:

- **Goal**:This represent academic goals with time frame.

Use Case: Assignment Submission& Review

1. After a session, the tutor assigns homework or tasks to a student within the app.
2. The student receives a notification about the new assignment.
3. Upon completion, the student submits the assignment through the app.
4. The tutor gets notified of the submission, reviews it, and provides feedback or grades within the app.
5. Feedback is instantly available to the student for review.

From this use case the entities I identified are as follows:

- **Assignment:**Represents tasks or homework set by the tutor
- **Feedback**:Represents tutor's comments, reviews,or grades on assignments.

Use Case: Parent Monitoring and Reporting

1. Parents log into the "AchieveMe Tutor" app with their unique credentials.
2. They navigate to their child's profile and access a dashboard summarizing performance, milestones, and upcoming sessions.
3. Parents can generate or request detailed reports, which chart progress over a selected timeframe.
4. Reports are either available instantly or sent to the parent's registered email.
5. Parents can schedule meetings with tutors based on these insights.

From this use case the entities I identified are as follows:

- **Report**:Represents a summary of a student's performance.
- **Meeting:**Represents a scheduled interaction between parents and tutors.
- **Dashboard**:Represents a consolidated view of a student's performance, milestones, and sessions.

Use Case: Resource Library Access

1. Tutors or administrators can upload educational resources to a shared library within the app.
2. Students search the library using filters like subject, grade level, or topic.
3. They access resources, which can include video lectures, PDFs, quizzes, etc.
4. Students can bookmark or save essential resources for easier access.
5. Tutors can recommend specific resources to individual students or groups.

From this use case the entities I identified are as follows:

- **Resource:**Represents educational materials in the library
- **Bookmark**:Represents the saved resources for easier future access.
- **Recommendation:** Represents the suggestion by tutors of specific resources to the students.

Use Case: FeedBack and Ratings

1. After a tutoring session, both student and tutor are prompted by the application to leave feedback and a rating.
2. The student rates the session and provides comments on what they found helpful or areas they'd like to focus on next time.
3. The tutor can leave feedback on the student's performance, noting areas of improvement and strengths.
4. Ratings and feedback are stored and can be viewed in the session history.

From this use case the entities I identified are as follows:

- **SessionFeedback:** Represents the reviews given by students and tutors post-session.

- **Rating**: Represents the numerical or star score given to the session.

Rules

1. **Each User has one Role, each Role can be associated with one or many Users.**

This rule implies that each user, whether they are a tutor, student, or parent, will be assigned a distinct role. This role defines the user's permissions and functions within the system. Since roles are crucial in determining user access rights, users must be linked to specific roles. It is possible that this rule could be modified in the future if an update allows users to switch roles or have multiple roles

2. **Each User has one Account; each Account is tied to a unique User.**

This signifies that each user has a distinct account within the system, which includes their personal and authentication details. This linkage is mandatory to ensure that every account corresponds to a user, granting them access to their account. If the system evolves to accommodate business or shared accounts in the future, this rule might be reconsidered

3. **Each Session is conducted by one Tutor and involves one Student.**

This rule stipulates that tutoring sessions inherently consist of one tutor and one student, resulting in two primary participants in each meeting. This one-on-one dynamic is essential for delivering personalized attention during coaching. If group sessions or multi-mentor sessions are introduced in the future, this relationship may require adjustment.

4. **Each Student can have multiple Progress entries, originating from multiple Sessions.**

This implies that as tutoring sessions progress, more student-related progress entries are recorded, with each entry directly tied to a specific session. This mandatory relationship ensures consistent documentation of student growth and challenges. However, this rule may undergo changes if future iterations introduce self-learning or non-session-related progression.

5. **Each Student has one or more Milestones; each Milestone belongs to a unique Student.**

This rule allows students to have multiple, individually tailored academic goals or milestones set for them during their educational journey. These milestones are highly relevant to each student's unique needs. Due to their central role in a student's academic plan, these milestones are of utmost importance. However, this rule may require adjustment if group or class milestones are introduced in the future

6. **Each Milestone can have multiple Achievements; every Achievement is associated with a single Milestone.**

This emphasizes that students, as they progress, may fully or partially achieve different milestones, making each achievement directly tied to a milestone. These achievements offer detailed tracking of a student's journey toward established milestones. However, this rule may require reconsideration if the system later incorporates achievements unrelated to milestones.

7. **Each Student can set one or more Goals; every Goal is linked to a unique Student.**

This means students can have multiple academic goals, with or without deadlines. The system tracks each goal for a specific student. Because goals provide direction and motivation, their connection to students is integral. This relationship may be modified if shared goals or group goals are introduced.

8. **Each Assignment is given to a Student by a Tutor; both Student and Tutor are tied to multiple Assignments.**

This highlights that tutors can assign various assignments to students over time, with each assignment linked to a specific student and the instructor who assigned it. Due to the dynamic nature of mentoring, this many-to-many relationship is crucial. If the system later incorporates bulk assignments or group tasks as a feature, we may need to reconsider this rule.

9. **Each Report pertains to a Student and can be viewed by their Parent; every Student can have multiple Reports.**

This rule states that students have performance reports that parents can access. This relationship ensures parents are kept informed about their child's academic progress. If group presentations or class summaries are incorporated, adjustments may be necessary.

10. **Each Resource in the library can have multiple Bookmarks; each Bookmark refers to a single Resource.**

This means students can mark multiple educational resources as important, creating a direct link between the resources and their bookmarks. This relationship ensures easy access for students. If the system allows a bookmark to contain multiple resources, such as a playlist or collection, this rule can be adjusted

11. **Each Session concludes with SessionFeedback from both Student and Tutor; every SessionFeedback relates to a unique Session.**

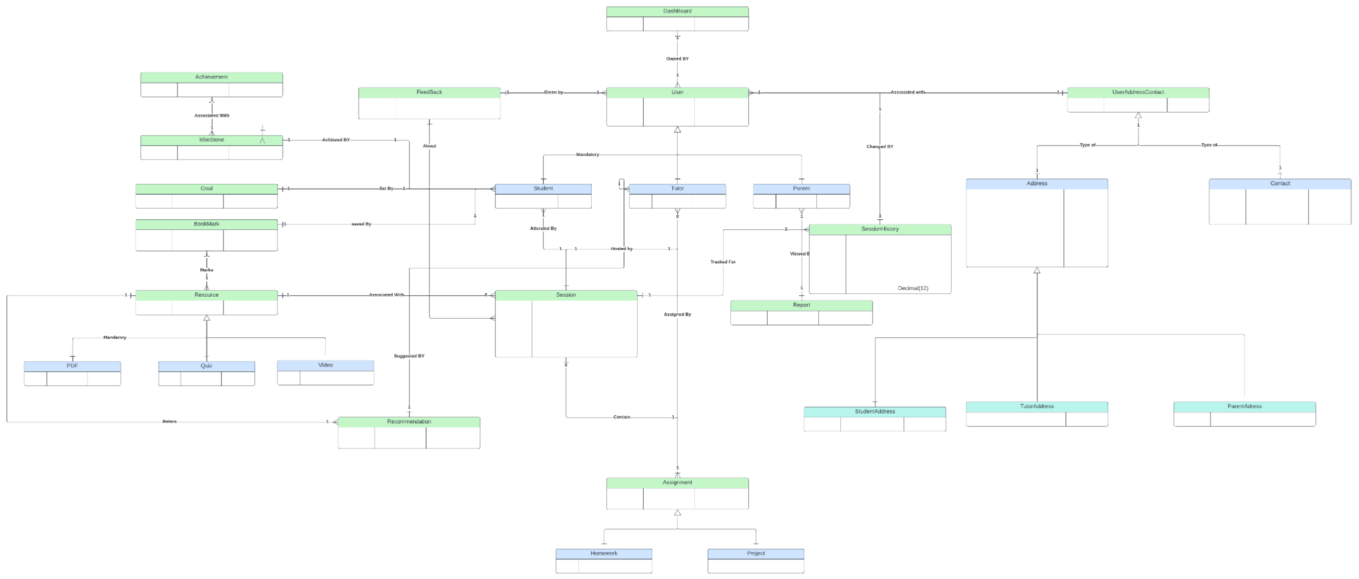
This rule emphasizes the significance of post-meeting feedback, where each session concludes with comments from both participants. As feedback is essential for improvement and understanding, its connection to meetings is crucial. This rule may require modification if group feedback or peer review is introduced

New Structural Database Rule:

12. **Any change to the session timings (sessionStartTime or sessionEndTime) in the 'Session' table should be recorded with a timestamp of the change, the user who made the change, and the previous and new values**.

Any time there's a change to the sessionStartTime or sessionEndTime in the 'Session' table, it's essential to keep a detailed record for auditing or review purposes. This rule ensures transparency and accountability for scheduling changes.

Conceptual Entity-Relationship Diagram



User (Tutor, Student, Parent):

- Each user can be associated with many assignments.
- Each user (as a tutor) can give many recommendations.
- Each user (as a student) can have many goals.
- Each user (as a student) can have many bookmarks.
- Each user (as a student) can have many milestones.
- Each user (as a tutor or student) can provide many feedbacks.
- Each user can have a single dashboard.
- Each user can be associated with many reports (if considered as a parent).
- Each user (as a student) can have long-term and short-term goals.
- Each user (as a student) can have many homework and project assignments.
- Each user can have one contact information.
- Each user can have one address.

Assignment:

- Each assignment is associated with one Session.
- Each assignment is associated with one tutor (user).
- Each assignment can be associated with many students (users).

Session:

- Each session can have many assignments.
- Each session can have many resources.

Resource:

- Each resource is associated with one Session.
- Each resource can be recommended many times.

Goal:

- Each goal is associated with one student (user).
- Goals can be categorized as long-term or short-term.

Bookmark:

- Each bookmark is associated with one student (user).
- Each bookmark is associated with one resource.

Feedback:

- Each feedback is associated with one giver (user).
- Each feedback is related to one session.

Dashboard:

- Each dashboard is associated with one user.

Recommendation:

- Each recommendation is associated with one tutor (user).
- Each recommendation is associated with one resource.

Milestone:

- Each milestone is associated with one student (user).

Achievement:

- Each achievement is associated with one milestone.

Report:

- Each report is associated with one parent (user).

Homework and Project:

- Both are types of assignments.
- Each can be associated with one student (user).
- Each is associated with one session.

Contact Information:

- Each set of contact information is associated with one user.

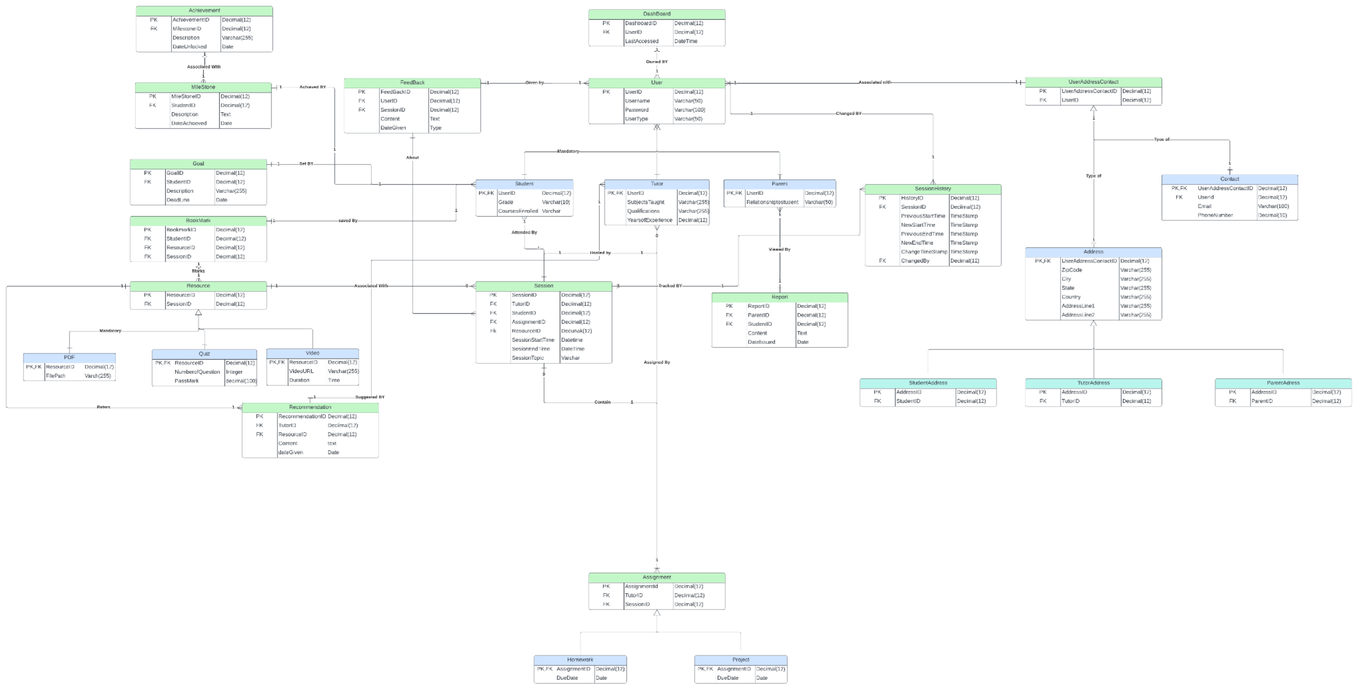
Address:

- Each address is associated with one user.

SessionHistory:

- Each historical record is associated with one session.
- Each historical record is associated with one user who made the change.

Full DBMS Physical ERD



Normalization:

1NF:

All the given tables for AchieveMe Tutor are already in 1NF because they have atomic column values and no repeating groups.

2NF:

For the AchieveMe Tutor tables to be in 2NF, no non-key attribute should be functionally dependent on a part of a composite primary key. For the given relationships, we do not see explicit composite primary keys that lead to partial dependencies.

3NF:

For a table to be in 3NF, it must remove all transitive dependencies so that attributes not part of the primary key are not functionally dependent upon other attributes that are not part of the primary key. Based on the provided information, there doesn't seem to be any transitive dependencies that need addressing.

BCNF:

No additional normalization appears to be required for BCNF since all determinants functionally determine all other attributes in the AchieveMe Tutor database.

Explanation:

User (Supertype):

Attributes:

- userID (Primary Key): Decimal(12). Serves as the unique identifier for each user. EX: 123456789012.
- username: varchar(100). The name chosen by the user for the platform.

- password: varchar(100). The secret passphrase for authentication.
- userType: varchar(50). Specifies the type: Student, Tutor, or Parent.
- Each user can be a student, tutor, or parent and has specific relations according to the type.

Student (Subtype):
Attributes:
- userID (Primary Key, Foreign Key): Decimal(12). Inherits from User. EX: 123456789012.
- grade: varchar(10). Represents the grade/class of the student. EX: 10th Grade, Junior, Sophomore.
- coursesEnrolled: varchar(255). A list or comma-separated values of courses the student is currently enrolled in. EX: Math, Science, History.

Tutor (Subtype):
Attributes:
- userID (Primary Key, Foreign Key): Decimal(12). Inherits from User. EX: 123456789012.
- subjectsTaught: varchar(255). Subjects that the tutor can teach. EX: Calculus, Physics, Chemistry.
- yearsOfExperience: int. Number of years the tutor has been teaching.

Parent (Subtype):
Attributes:
- userID (Primary Key, Foreign Key): Decimal(12). Inherits from User. EX: 123456789012.
- relationshipToStudent: varchar(50). Clarifies the relationship with the associated student. EX: Mother, Father, Guardian.

Assignment:
Attributes:
- assignmentID (Primary Key): Decimal(12). Differentiates one assignment from others. EX: 123456789012.
- tutorID (Foreign Key): Decimal(12). References data from the User table when the userType is 'Tutor'. EX: 123456789012.

Session:
Attributes:
- sessionID (Primary Key): Decimal(12). Distinguishes one session from another within the database. EX: 123456789012.
- tutorID (Foreign Key): Decimal(12). References data from the User table when the userType is 'Tutor'. EX: 234567890123.
- studentID (Foreign Key): Decimal(12). References data from the User table when the userType is 'Student'. EX: 345678901234.
- sessionStartTime: datetime. Marks the beginning time of the session. EX: 2023-10-10 10:00:00.
- sessionEndTime: datetime. Marks the ending time of the session. EX: 2023-10-10 11:00:00.
- sessionTopic: varchar(255). Provides a brief about what the session covers. EX: Calculus Basics.

Resource:
Attributes:
- resourceID (Primary Key): Decimal(12). Unique identifier for resources. EX: 123456789012.
- sessionID (Foreign Key): Decimal(12). Relates to the session. EX: 123456789012.

Reports:
Attributes:
- reportID (Primary Key): Decimal(12). EX: 123456789012.
- parentID (Foreign Key): Decimal(12). References data from the User table when the userType is 'Parent'. EX: 123456789012.

- studentID (Foreign Key): Decimal(12). References data from the User table when the userType is 'Student'. EX: 345678901234.
- content: text. Detailed content of the report.
- DateIssued: Date. The date when the report was issued.

Milestones:
Attributes:
- milestoneID (Primary Key): Decimal(12). EX: 123456789012.
- studentID (Foreign Key): Decimal(12). References data from the User table when the userType is 'Student'. EX: 345678901234.
- description: text. Details about the milestone achieved.
- DateAchieved: Date. The date when the milestone was achieved.

UserAddressContact (General):
Attributes:
- UAC_ID (Primary Key): Decimal(12)
- UserID (Foreign Key): Decimal(12) [References User]

Address (Specialization of UserAddressContact):
Attributes:
- AddressID (Primary Key): Decimal(12). Distinguishes one address from another. EX: 123456789012.
- UserAddressContactID (Foreign Key): Decimal(12) [References UserAddressContact].
- AddressLine1: varchar(255). Street name. EX: 5th Avenue.
- AddressLine2: varchar(255). Optionally, for apartment number or additional information. EX: Apt 12A.
- ZipCode: varchar(10). Specific area of the address. EX: 10001.
- City: varchar(255). EX: New York.
- State: varchar(255). EX: New York.
- Country: varchar(255). EX: USA.

StudentAddress (Subtype of Address):
Attributes:
- AddressID (Primary Key, Foreign Key): Decimal(12) [References Address].
- StudentID (Foreign Key, Mandatory): Decimal(12) [References Student].

TutorAddress (Subtype of Address):
Attributes:
- AddressID (Primary Key, Foreign Key): Decimal(12) [References Address].
- TutorID (Foreign Key, Mandatory): Decimal(12) [References Tutor].

ParentAddress (Subtype of Address):
Attributes:
- AddressID (Primary Key, Foreign Key): Decimal(12) [References Address].
- ParentID (Foreign Key, Mandatory): Decimal(12) [References Parent].

Contact (Specialization of UserAddressContact):
Attributes:
- UAC_ID (Primary Key, Foreign Key): Decimal(12) [References UserAddressContact].
- UserID (Foreign Key, Mandatory): Decimal(12) [References User]
- Email: varchar(255). Contact email of the user.
- PhoneNumber: Decimal(10). Optional.

Feedback:
Attributes:

- FeedbackID (Primary Key): Decimal(12). Unique identifier for feedback entries. EX: 123456789012.
- UserID (Foreign Key): Decimal(12) [References User].
- SessionID (Foreign Key): Decimal(12) [References Session].
- Content: Text. The feedback content.
- DateGiven: Date. The date when the feedback was provided.

Goal:
Attributes:
- GoalID (Primary Key): Decimal(12). Unique identifier for goals. EX: 123456789012.
- StudentID (Foreign Key): Decimal(12) [References Student].
- Description: varchar(255). Details of the goal. EX: Master Trigonometry by December.
- Deadline: Date. The date by which the student aims to achieve the goal.

Bookmark:
Attributes:
- BookmarkID (Primary Key): Decimal(12). Unique identifier for bookmarks. EX: 123456789012.
- StudentID (Foreign Key): Decimal(12) [References Student].
- ResourceID (Foreign Key, optional): Decimal(12) [References Resource].
- SessionID (Foreign Key, optional): Decimal(12) [References Session].

Dashboard:
Attributes:
- DashboardID (Primary Key): Decimal(12). Unique identifier for dashboards. EX: 123456789012.
- UserID (Foreign Key): Decimal(12) [References User].
- LastAccessed: Datetime. The last time the dashboard was viewed.

Homework (Subtype of Assignment):
Attributes:
- Inherits AssignmentID.
- DueDate: Date. Deadline for homework submission.

Project (Subtype of Assignment):
Attributes:
- Inherits AssignmentID.
- DueDate: Date. Deadline for project submission.

PDF (Subtype of Resource):
Attributes:
- Inherits ResourceID.
- FilePath: varchar(255). The location/path of the stored PDF.

Quiz (Subtype of Resource):
Attributes:
- Inherits ResourceID.
- NumberOfQuestions: Integer. The total number of questions in the quiz.
- PassMark: Decimal. The mark to achieve a pass status.

Video (Subtype of Resource):
Attributes:
- Inherits ResourceID.
- VideoURL: varchar(255). The URL or path of the stored video.
- Duration: Time. The length of the video.

Achievement:
Attributes:

- AchievementID (Primary Key): Decimal(12). Unique identifier for achievements. EX: 123456789012.
- StudentID (Foreign Key): Decimal(12) [References Student].
- Description: varchar(255). Description or title of the achievement.
- DateUnlocked: Date. The date when the achievement was unlocked

Recommendation:

Attributes:

- RecommendationID (Primary Key): Decimal(12). Unique identifier for recommendations. EX: 123456789012.
- TutorID (Foreign Key): Decimal(12) [References Tutor].
- StudentID (Foreign Key): Decimal(12) [References Student].
- ResourceID (Foreign Key, optional): Decimal(12) [References Resource]. The resource being recommended or based upon which the recommendation is made.
- Content: Text. The recommendation content.
- DateGiven: Date. The date when the recommendation was provided.

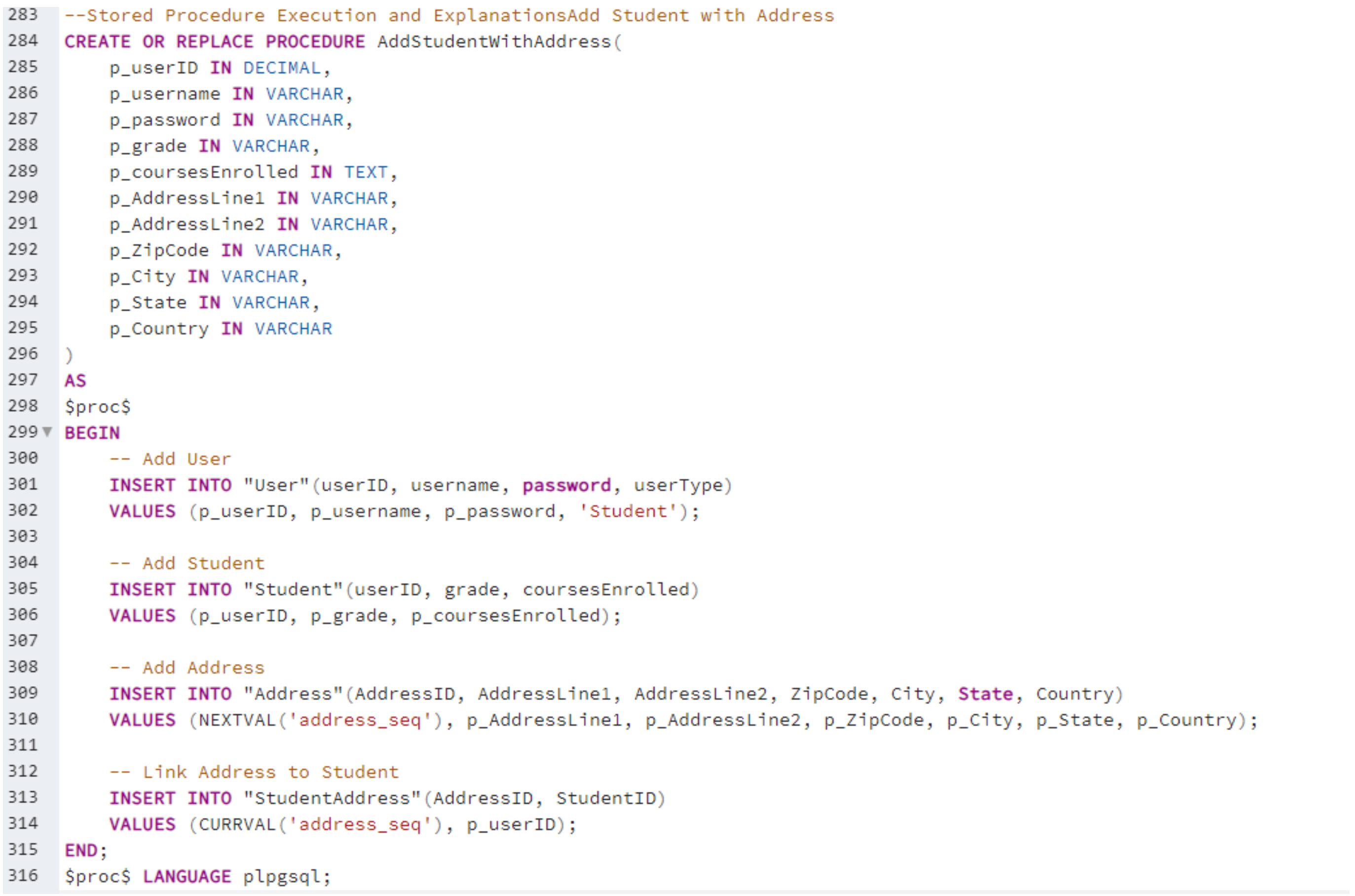
SessionHistory:

Attributes:

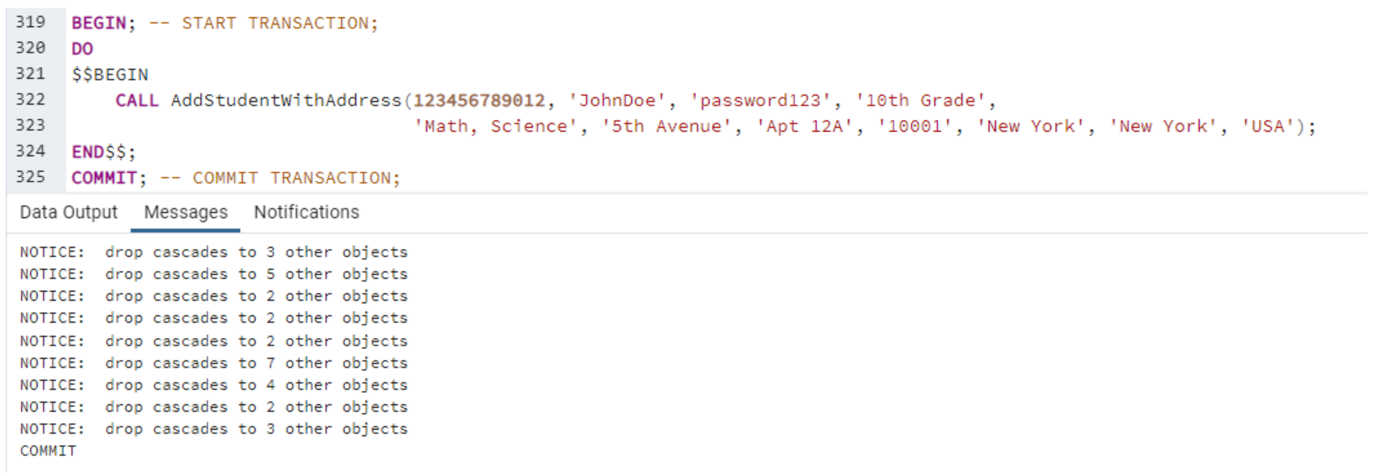
- historyID (Primary Key): Decimal(12)
- sessionID (Foreign Key): Decimal(12) [References Session]
- previousStartTime: TIMESTAMP
- newStartTime: TIMESTAMP
- previousEndTime: TIMESTAMP
- newEndTime: TIMESTAMP
- changeTimestamp: TIMESTAMP (The exact time the change was made)
- changedBy (Foreign Key): Decimal(12) [References User]

Stored Procedure Execution and Explanations

USe Case 1:Add Student With Addres



```
--Stored Procedure Execution and ExplanationsAdd Student with Address
CREATE OR REPLACE PROCEDURE AddStudentWithAddress(
    p_userID IN DECIMAL,
    p_username IN VARCHAR,
    p_password IN VARCHAR,
    p_grade IN VARCHAR,
    p_coursesEnrolled IN TEXT,
    p_AddressLine1 IN VARCHAR,
    p_AddressLine2 IN VARCHAR,
    p_ZipCode IN VARCHAR,
    p_City IN VARCHAR,
    p_State IN VARCHAR,
    p_Country IN VARCHAR
)
AS
$proc$
BEGIN
    -- Add User
    INSERT INTO "User"(userID, username, password, userType)
    VALUES (p_userID, p_username, p_password, 'Student');

    -- Add Student
    INSERT INTO "Student"(userID, grade, coursesEnrolled)
    VALUES (p_userID, p_grade, p_coursesEnrolled);

    -- Add Address
    INSERT INTO "Address"(AddressID, AddressLine1, AddressLine2, ZipCode, City, State, Country)
    VALUES (NEXTVAL('address_seq'), p_AddressLine1, p_AddressLine2, p_ZipCode, p_City, p_State, p_Country);

    -- Link Address to Student
    INSERT INTO "StudentAddress"(AddressID, StudentID)
    VALUES (CURRVAL('address_seq'), p_userID);
END;
$proc$ LANGUAGE plpgsql;
```



```
BEGIN; -- START TRANSACTION;
DO
$$BEGIN
    CALL AddStudentWithAddress(123456789012, 'JohnDoe', 'password123', '10th Grade',
                               'Math, Science', '5th Avenue', 'Apt 12A', '10001', 'New York', 'New York', 'USA');
END$$;
COMMIT; -- COMMIT TRANSACTION;
```

Data Output Messages Notifications

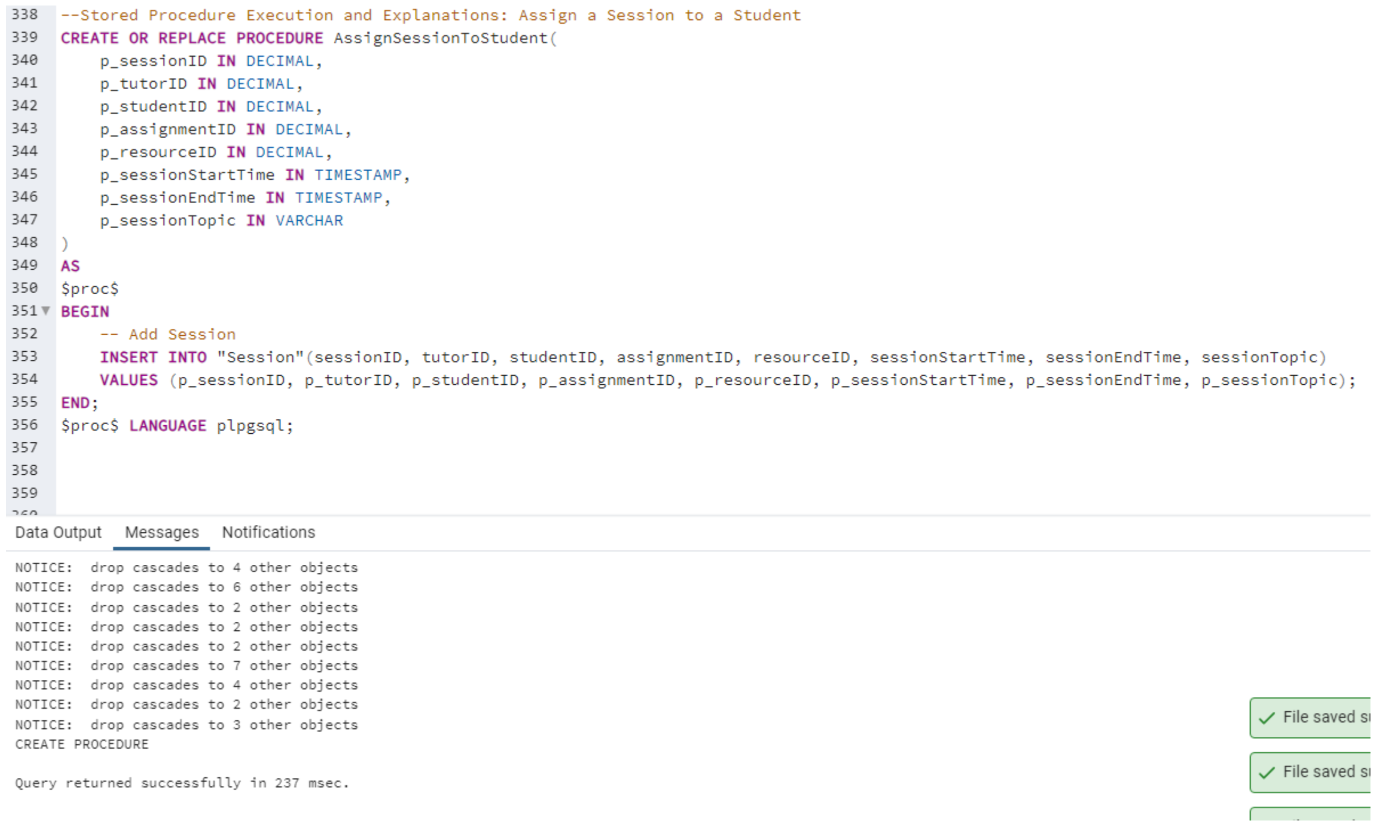
```
NOTICE:  drop cascades to 3 other objects
NOTICE:  drop cascades to 5 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 7 other objects
NOTICE:  drop cascades to 4 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 3 other objects
COMMIT
```

Explanations:

- Parameters: We define input parameters for this procedure which will be passed when the procedure is called. These parameters represent details for the user, student, and address.
- -- Add User: Here, we are inserting a record into the "user" table with the provided details and setting userType to 'Student'.
- -- Add Student: Inserts the student details into the Student table.

- -- Add Address: Adds a new address for the student. We use NEXTVAL('address_seq') to automatically generate the next value from the address sequence, ensuring that the AddressID is unique.
- -- Link Address to Student: Links the newly added address to the student using a relationship table (StudentAddress). used CURRVAL('address_seq') to get the most recently generated sequence value for the AddressID, which is the address just added.

Use Case 2: Assign a Session to a Student



```
--Stored Procedure Execution and Explanations: Assign a Session to a Student
CREATE OR REPLACE PROCEDURE AssignSessionToStudent(
    p_sessionID IN DECIMAL,
    p_tutorID IN DECIMAL,
    p_studentID IN DECIMAL,
    p_assignmentID IN DECIMAL,
    p_resourceID IN DECIMAL,
    p_sessionStartTime IN TIMESTAMP,
    p_sessionEndTime IN TIMESTAMP,
    p_sessionTopic IN VARCHAR
)
AS
$proc$
BEGIN
    -- Add Session
    INSERT INTO "Session"(sessionID, tutorID, studentID, assignmentID, resourceID, sessionStartTime, sessionEndTime, sessionTopic)
    VALUES (p_sessionID, p_tutorID, p_studentID, p_assignmentID, p_resourceID, p_sessionStartTime, p_sessionEndTime, p_sessionTopic);
END;
$proc$ LANGUAGE plpgsql;
```

```
NOTICE:  drop cascades to 4 other objects
NOTICE:  drop cascades to 6 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 7 other objects
NOTICE:  drop cascades to 4 other objects
NOTICE:  drop cascades to 2 other objects
NOTICE:  drop cascades to 3 other objects
CREATE PROCEDURE

Query returned successfully in 237 msec.
```

Explanations:

- Parameters: We define input parameters for this procedure which are used to detail the session details.
- -- Add Session: Here, we are inserting a record into the Session table with the provided details. The session details include the ID of the tutor, student, related assignment, resources, start and end times, and the topic.

Question Identification and Explanations

1. Which tutors teaching a specific subject have received feedback, and who are the students involved in those sessions?

Explanation: To ensure that tutoring sessions are effective and of high quality, it's important to monitor feedback given to tutors. This question helps in identifying tutors who have received feedback for sessions related to a particular subject. By knowing the students involved, the platform can gather more detailed feedback if necessary.

2. For a specific course, who are the enrolled students and their associated tutors' contact details?

Explanation: When managing a tutoring platform, it's essential to have quick access to the list of students enrolled in a particular course and their respective tutors. This aids in communication, planning, and monitoring the progress of the course.

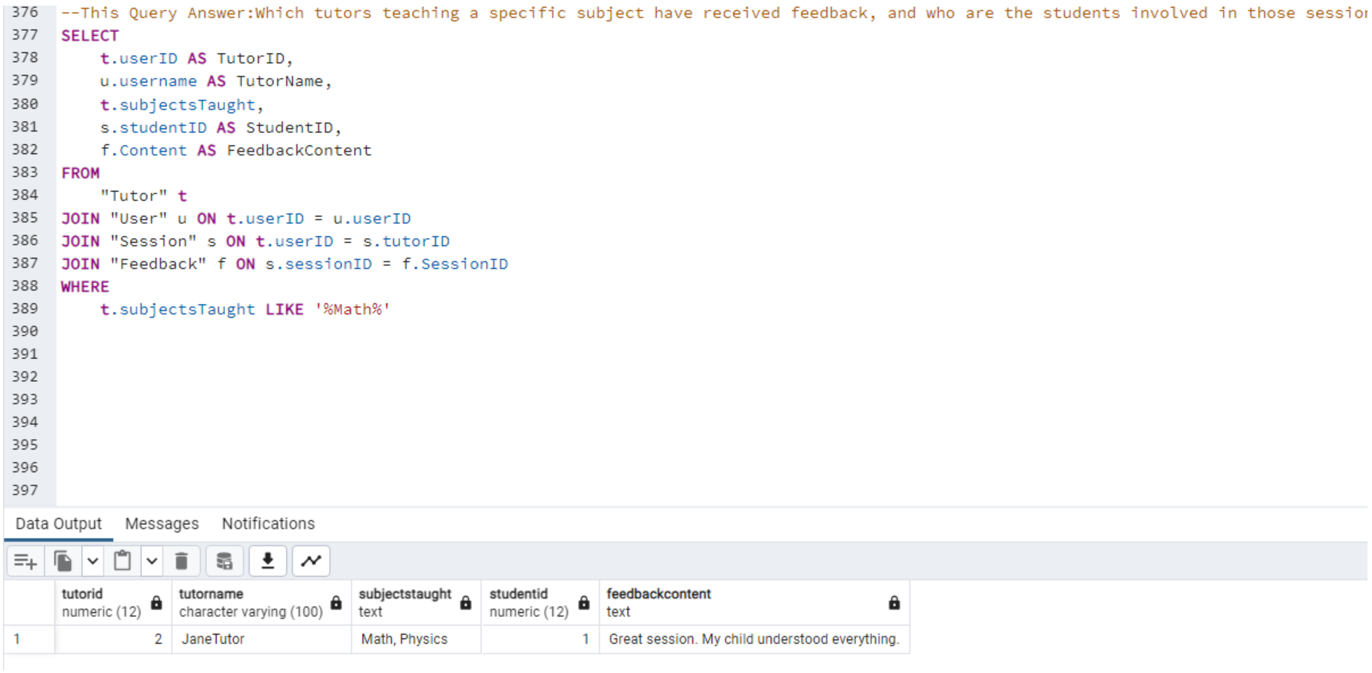
3. Which resources are frequently recommended by tutors?

Explanation: In a tutoring platform, resources play a pivotal role in enhancing the learning experience. Knowing which resources are frequently recommended can help the platform identify high-quality materials and promote them more.

Query Executions and Explanations

1. Which tutors teaching a specific subject have received feedback, and who are the students involved in those sessions?

Query:



```sql
--This Query Answer:Which tutors teaching a specific subject have received feedback, and who are the students involved in those sessio
SELECT
    t.userID AS TutorID,
    u.username AS TutorName,
    t.subjectsTaught,
    s.studentID AS StudentID,
    f.Content AS FeedbackContent
FROM
    "Tutor" t
JOIN "User" u ON t.userID = u.userID
JOIN "Session" s ON t.userID = s.tutorID
JOIN "Feedback" f ON s.sessionID = f.SessionID
WHERE
    t.subjectsTaught LIKE '%Math%'
```

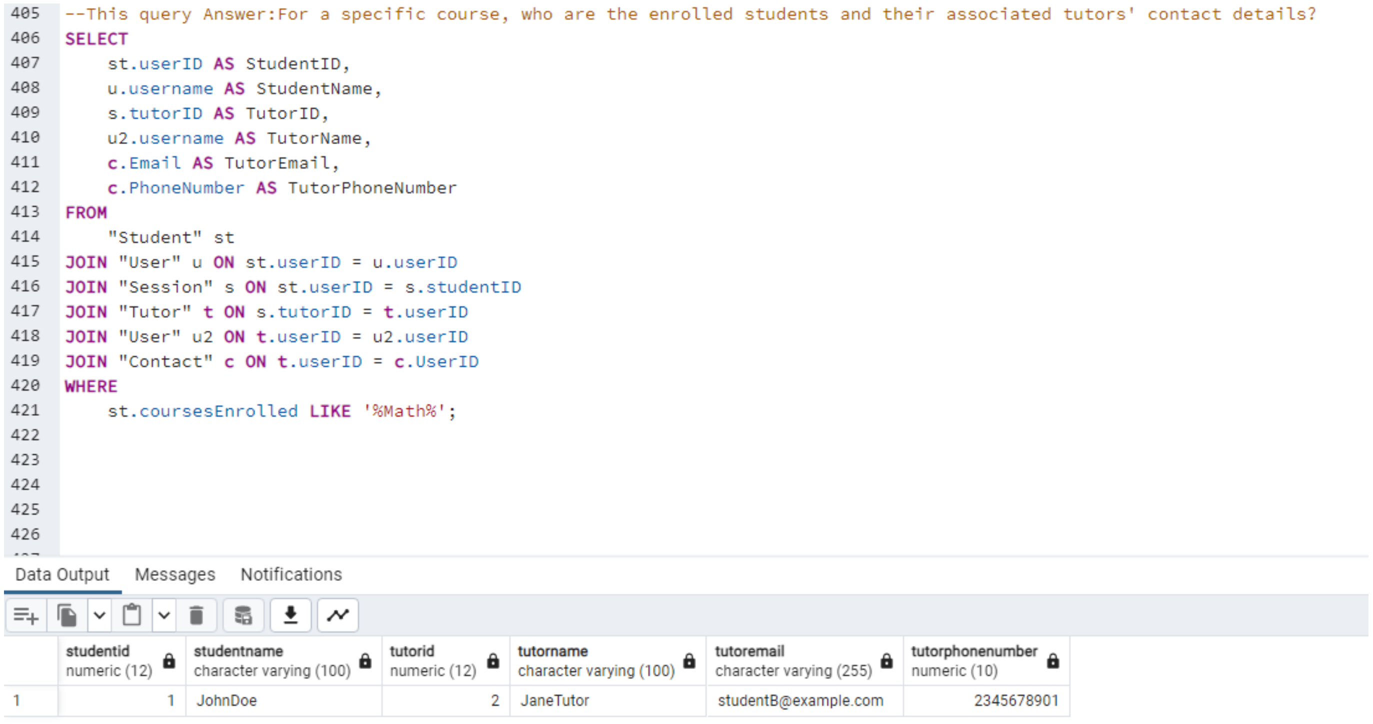
Data Output Messages Notifications

	tutorid numeric (12)	tutorname character varying (100)	subjectstaught text	studentid numeric (12)	feedbackcontent text
1	2	JaneTutor	Math, Physics	1	Great session. My child understood everything.

Explanation: The query retrieves the names of tutors who have taught a specific subject and have received feedback in related sessions. Additionally, the names of students who participated in these sessions are fetched. This is achieved by joining the User, Tutor, Session, and Feedback tables, ensuring that the tutor's subject matches the specified criteria.

2. For a specific course, who are the enrolled students and their associated tutors' contact details?

Query :



```sql
--This query Answer:For a specific course, who are the enrolled students and their associated tutors' contact details?
SELECT
    st.userID AS StudentID,
    u.username AS StudentName,
    s.tutorID AS TutorID,
    u2.username AS TutorName,
    c.Email AS TutorEmail,
    c.PhoneNumber AS TutorPhoneNumber
FROM
    "Student" st
JOIN "User" u ON st.userID = u.userID
JOIN "Session" s ON st.userID = s.studentID
JOIN "Tutor" t ON s.tutorID = t.userID
JOIN "User" u2 ON t.userID = u2.userID
JOIN "Contact" c ON t.userID = c.UserID
WHERE
    st.coursesEnrolled LIKE '%Math%';
```

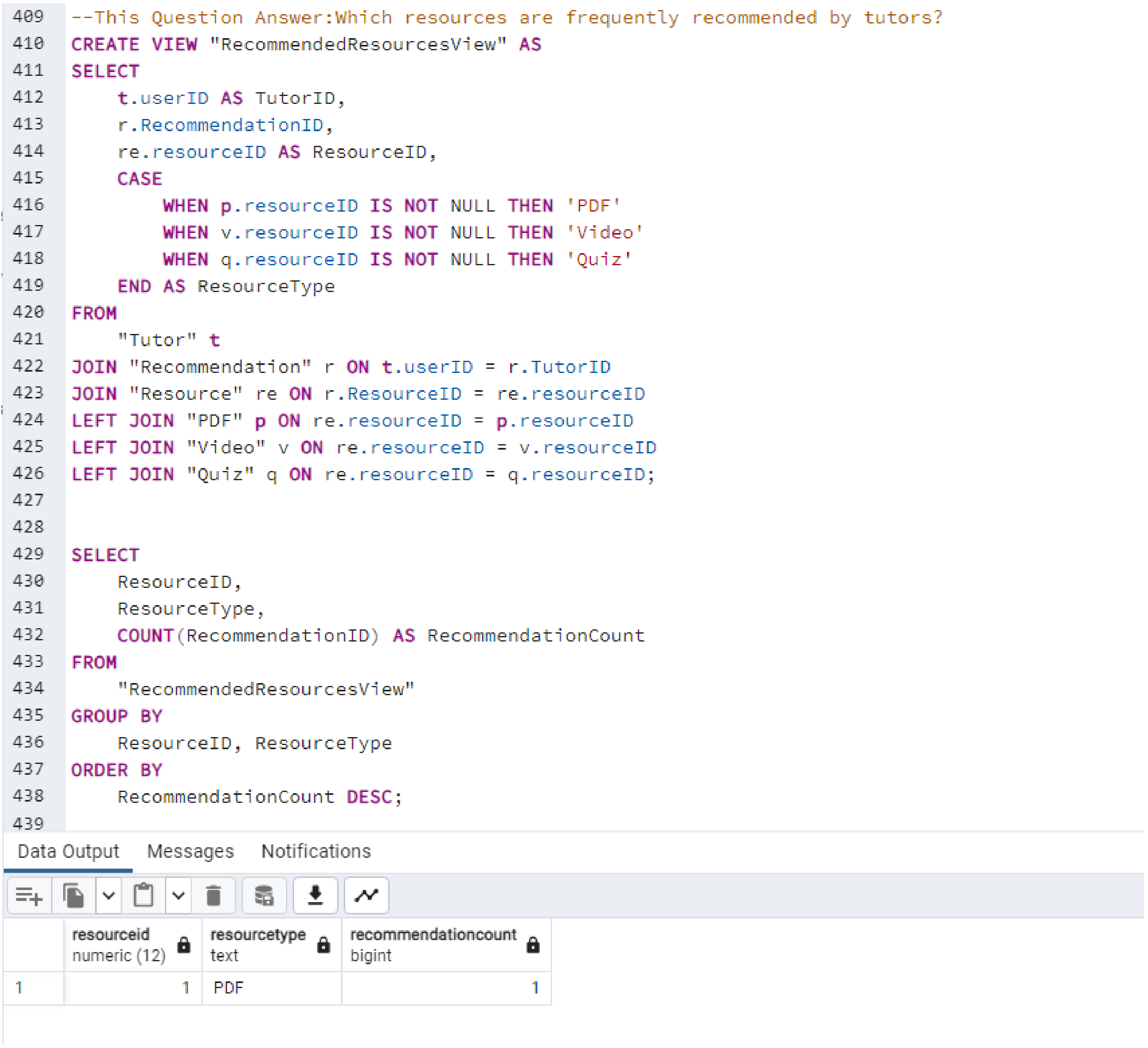
Data Output Messages Notifications

	studentid numeric (12)	studentname character varying (100)	tutorid numeric (12)	tutorname character varying (100)	tutoremail character varying (255)	tutorphonenumber numeric (10)
1	1	JohnDoe	2	JaneTutor	studentB@example.com	2345678901

Explanation: This query fetches the names and contact details of tutors associated with a particular course, as well as the names of students enrolled in that course. The User, Student, Tutor, Session, and Contact tables are joined to provide comprehensive details, making sure that the course corresponds to the specified criteria.

3. Which resources are frequently recommended by tutors?

Query:



```
--This Question Answer:Which resources are frequently recommended by tutors?
CREATE VIEW "RecommendedResourcesView" AS
SELECT
    t.userID AS TutorID,
    r.RecommendationID,
    re.resourceID AS ResourceID,
    CASE
        WHEN p.resourceID IS NOT NULL THEN 'PDF'
        WHEN v.resourceID IS NOT NULL THEN 'Video'
        WHEN q.resourceID IS NOT NULL THEN 'Quiz'
    END AS ResourceType
FROM
    "Tutor" t
JOIN "Recommendation" r ON t.userID = r.TutorID
JOIN "Resource" re ON r.ResourceID = re.resourceID
LEFT JOIN "PDF" p ON re.resourceID = p.resourceID
LEFT JOIN "Video" v ON re.resourceID = v.resourceID
LEFT JOIN "Quiz" q ON re.resourceID = q.resourceID;


SELECT
    ResourceID,
    ResourceType,
    COUNT(RecommendationID) AS RecommendationCount
FROM
    "RecommendedResourcesView"
GROUP BY
    ResourceID, ResourceType
ORDER BY
    RecommendationCount DESC;
```

Data Output | Messages | Notifications

	resourceid numeric (12)	resourcetype text	recommendationcount bigint
1	1	PDF	1

Explanation: The query identifies resources that are frequently recommended by tutors to students. It fetches the resource ID, type (e.g., PDF), and the number of times it has been recommended. This information is gathered by joining the Resource, Recommendation, and Tutor tables, and counting the number of recommendations for each resource.

Index Identification and Creations

As far as primary keys which are already indexed, here is the list:

1. User.userID
2. Student.userID
3. Tutor.userID
4. Parent.userID
5. Assignment.assignmentID
6. Session.sessionID
7. Resource.resourceID
8. UserAddressContact.UAC_ID
9. Address.AddressID
10. StudentAddress.AddressID
11. TutorAddress.AddressID
12. ParentAddress.AddressID
13. Contact.UAC_ID
14. PDF.resourceID
15. Video.resourceID
16. Quiz.resourceID
17. Homework.assignmentID
18. Project.assignmentID
19. Reports.reportID
20. Milestones.milestoneID
21. Feedback.FeedbackID
22. Goal.GoalID
23. Bookmark.BookmarkID
24. Dashboard.DashboardID
25. Achievement.AchievementID
26. Recommendation.RecommendationID

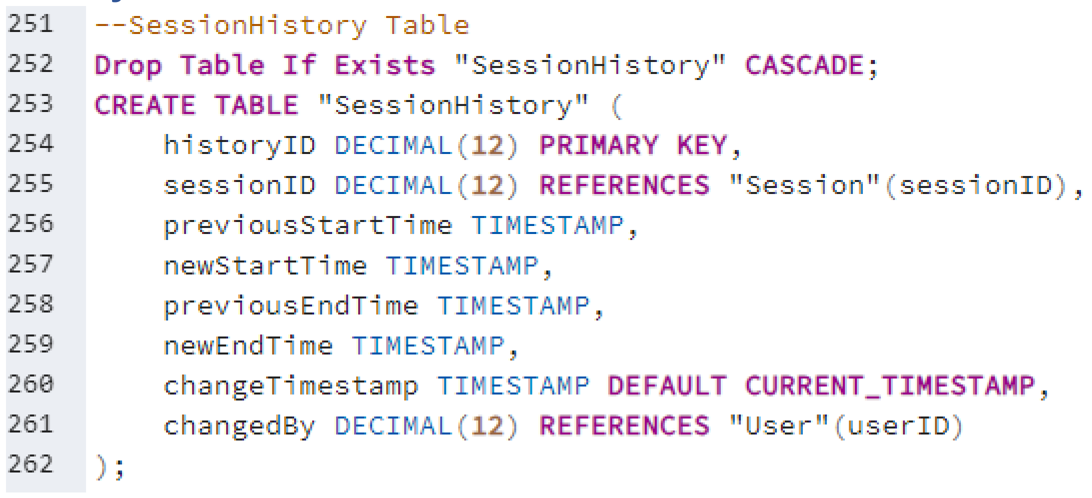
As far as foreign keys, I know all of them need an index. Below is a table identifying each foreign key column, whether or not the index should be unique or not, and why.

Foreign Key	Unique?	Description
Session.tutorID	Not unique	The foreign key in Session referencing Tutor can have many sessions for the same tutor.
Session.studentID	Not unique	The foreign key in Session referencing Student can have many sessions for the same student.
Assignment.tutorID	Not unique	The foreign key in Assignment referencing Tutor can have many assignments for the same tutor.
Resource.sessionID	Not unique	The foreign key in Resource referencing Session indicates which session the resource belongs to.
UserAddressContact.UserID	Not unique	The foreign key in UserAddressContact referencing User links contact details to a specific user.
Address.UserAddressContactID	Not unique	The foreign key in Address referencing UserAddressContact links an address to its contact details.
StudentAddress.StudentID	Not	The foreign key in StudentAddress referencing Student links an address to

	unique	a specific student.
TutorAddress.TutorID	Not unique	The foreign key in TutorAddress referencing Tutor links an address to a specific tutor.
ParentAddress.ParentID	Not unique	The foreign key in ParentAddress referencing Parent links an address to a specific parent.
Contact.UserID	Not unique	The foreign key in Contact referencing User links contact details to a specific user.
Reports.parentID	Not unique	The foreign key in Reports referencing Parent can have many reports for the same parent.
Reports.studentID	Not unique	The foreign key in Reports referencing Student can have many reports for the same student.
Milestones.studentID	Not unique	The foreign key in Milestones referencing Student can have many milestones for the same student.
Feedback.UserID	Not unique	The foreign key in Feedback referencing User can have feedback from or for the same user multiple times.
Feedback.SessionID	Not unique	The foreign key in Feedback referencing Session indicates which session the feedback is about.
Goal.StudentID	Not unique	The foreign key in Goal referencing Student can have many goals for the same student.
Bookmark.StudentID	Not unique	The foreign key in Bookmark referencing Student can have many bookmarks for the same student.
Bookmark.ResourceID	Not unique	The foreign key in Bookmark referencing Resource can have many bookmarks for the same resource.
Bookmark.SessionID	Not unique	The foreign key in Bookmark referencing Session indicates which session the bookmark is about.
Dashboard.UserID	Not unique	The foreign key in Dashboard referencing User can have multiple dashboards for the same user.
Recommendation.TutorID	Not unique	The foreign key in Recommendation referencing Tutor can have many recommendations for the same tutor.
Recommendation.StudentID	Not unique	The foreign key in Recommendation referencing Student can have many recommendations for the same student.
Recommendation.ResourceID	Not unique	The foreign key in Recommendation referencing Resource can have many recommendations for the same resource.

```sql
--Indexs
CREATE INDEX idx_session_tutor ON "Session" (tutorID);
CREATE INDEX idx_session_student ON "Session" (studentID);
CREATE INDEX idx_assignment_tutor ON "Assignment" (tutorID);
CREATE INDEX idx_resource_session ON "Resource" (sessionID);
CREATE INDEX idx_uac_user ON "UserAddressContact" (UserID);
CREATE INDEX idx_address_uac ON "Address" (UserAddressContactID);
CREATE INDEX idx_studaddress_student ON "StudentAddress" (StudentID);
CREATE INDEX idx_tutoraddress_tutor ON "TutorAddress" (TutorID);
CREATE INDEX idx_parentaddress_parent ON "ParentAddress" (ParentID);
CREATE INDEX idx_contact_user ON "Contact" (UserID);
CREATE INDEX idx_reports_parent ON "Reports" (parentID);
CREATE INDEX idx_reports_student ON "Reports" (studentID);
CREATE INDEX idx_milestones_student ON "Milestones" (studentID);
CREATE INDEX idx_feedback_user ON "Feedback" (UserID);
CREATE INDEX idx_feedback_session ON "Feedback" (SessionID);
CREATE INDEX idx_goal_student ON "Goal" (StudentID);
CREATE INDEX idx_bookmark_student ON "Bookmark" (StudentID);
CREATE INDEX idx_bookmark_resource ON "Bookmark" (ResourceID);
CREATE INDEX idx_bookmark_session ON "Bookmark" (SessionID);
CREATE INDEX idx_dashboard_user ON "Dashboard" (UserID);
CREATE INDEX idx_recommendation_tutor ON "Recommendation" (TutorID);
CREATE INDEX idx_recommendation_student ON "Recommendation" (StudentID);
CREATE INDEX idx_recommendation_resource ON "Recommendation" (ResourceID);

```

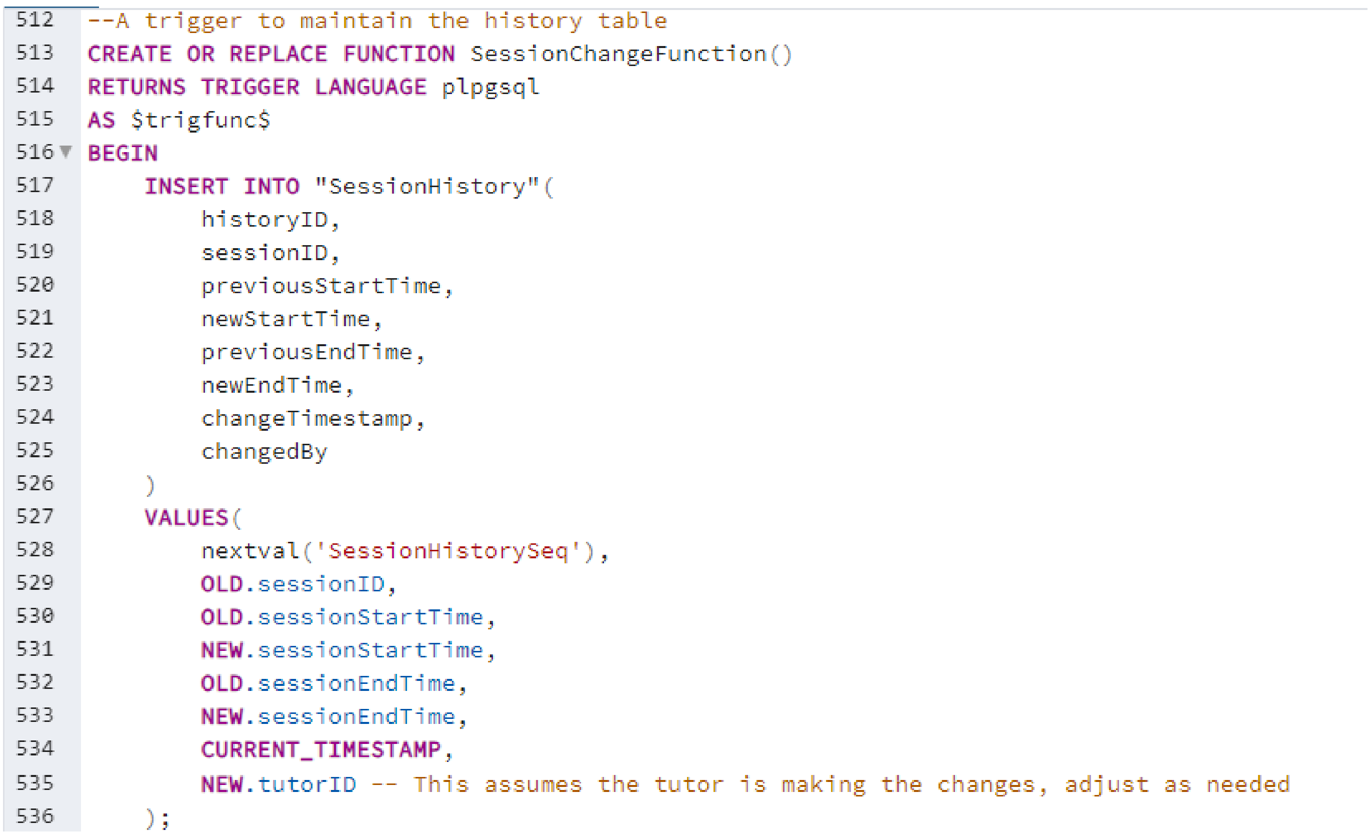
History Table Demonstration



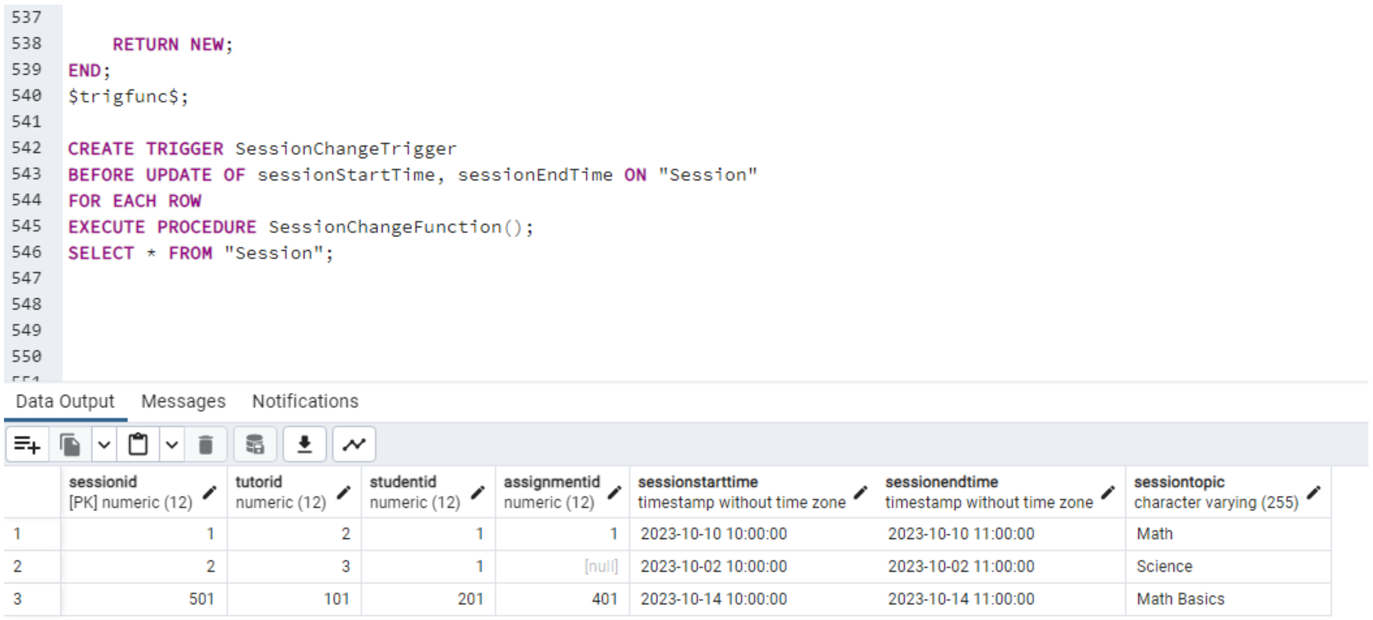
```sql
--SessionHistory Table
Drop Table If Exists "SessionHistory" CASCADE;
CREATE TABLE "SessionHistory" (
    historyID DECIMAL(12) PRIMARY KEY,
    sessionID DECIMAL(12) REFERENCES "Session"(sessionID),
    previousStartTime TIMESTAMP,
    newStartTime TIMESTAMP,
    previousEndTime TIMESTAMP,
    newEndTime TIMESTAMP,
    changeTimestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    changedBy DECIMAL(12) REFERENCES "User"(userID)
);
```

SessionHistory Table:
This table is designed to capture historical changes made to session timings in the "Session" table.
Columns in SessionHistory:

- historyID: A unique identifier for each historical record.
- sessionID: The ID of the session that had its timings changed. It references the sessionID in the "Session" table.
- previousStartTime: The original start time of the session before the change.
- newStartTime: The updated start time after the change.
- previousEndTime: The original end time of the session before the change.
- newEndTime: The updated end time after the change.
- changeTimestamp: The time at which the change was made. It defaults to the current timestamp.
- changedBy: The ID of the user who made the change. It references the userID in the "User" table.



```
--A trigger to maintain the history table
CREATE OR REPLACE FUNCTION SessionChangeFunction()
RETURNS TRIGGER LANGUAGE plpgsql
AS $trigfunc$
BEGIN
    INSERT INTO "SessionHistory"(
        historyID,
        sessionID,
        previousStartTime,
        newStartTime,
        previousEndTime,
        newEndTime,
        changeTimestamp,
        changedBy
    )
    VALUES(
        nextval('SessionHistorySeq'),
        OLD.sessionID,
        OLD.sessionStartTime,
        NEW.sessionStartTime,
        OLD.sessionEndTime,
        NEW.sessionEndTime,
        CURRENT_TIMESTAMP,
        NEW.tutorID -- This assumes the tutor is making the changes, adjust as needed
    );
```



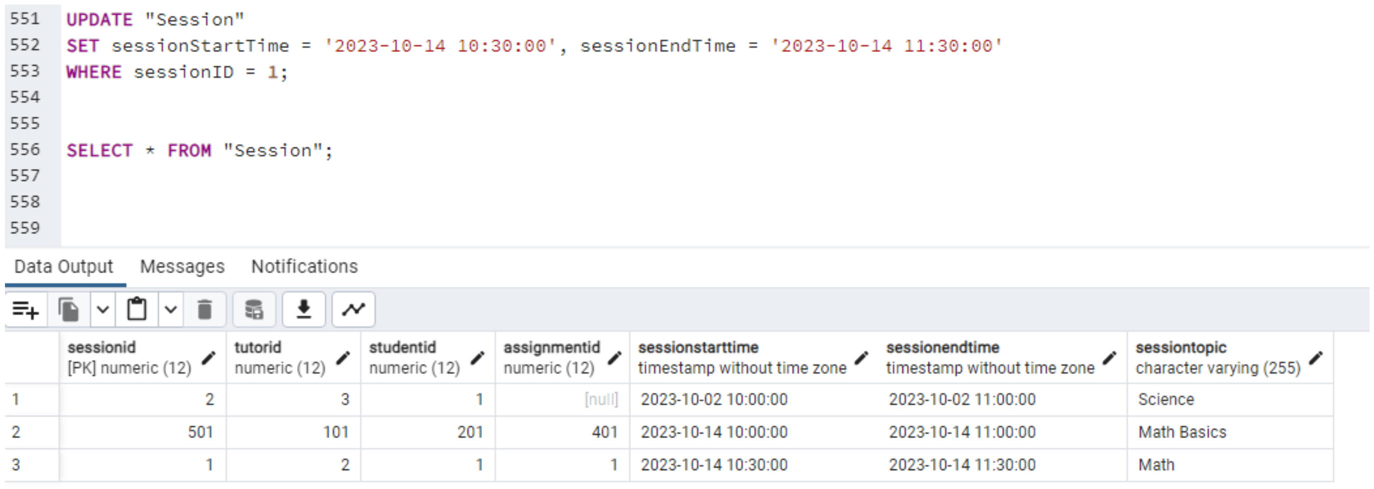
```

    RETURN NEW;
END;
$trigfunc$;

CREATE TRIGGER SessionChangeTrigger
BEFORE UPDATE OF sessionStartTime, sessionEndTime ON "Session"
FOR EACH ROW
EXECUTE PROCEDURE SessionChangeFunction();
SELECT * FROM "Session";
```

Data Output Messages Notifications

	sessionid [PK] numeric (12)	tutorid numeric (12)	studentid numeric (12)	assignmentid numeric (12)	sessionstarttime timestamp without time zone	sessionendtime timestamp without time zone	sessiontopic character varying (255)
1	1	2	1	1	2023-10-10 10:00:00	2023-10-10 11:00:00	Math
2	2	3	1	[null]	2023-10-02 10:00:00	2023-10-02 11:00:00	Science
3	501	101	201	401	2023-10-14 10:00:00	2023-10-14 11:00:00	Math Basics

CODE	DESCRIPTION
CREATE OR REPLACE FUNCTION SessionChangeFunction()	This starts the definition of the function and names it SessionChangeFunction. It indicates that if this function already exists, it should be replaced with this definition.
RETURNS TRIGGER LANGUAGE plpgsql	Specifies the return type of the function as TRIGGER and the language used is plpgsql.

AS $trigfunc$	The start of the function body. $trigfunc$ is a dollar-quoted string constant which is a way to define string literals in PostgreSQL.
BEGIN	Begins the main execution block of the function.
INSERT INTO "SessionHistory"...	This inserts a new record into the SessionHistory table, capturing the old and new session timings.
... VALUES(...	Provides the values that will be inserted into the SessionHistory table.
RETURN NEW;	Indicates that the new modified row should be returned to the Session table. This is necessary when using a BEFORE trigger.
END;	Ends the main execution block of the function.
$trigfunc$;	Marks the end of the function body.
CREATE TRIGGER SessionChangeTrigger	This starts the definition of the trigger and names it "SessionChangeTrigger".
BEFORE UPDATE OF sessionStartTime, sessionEndTime ON "Session"	Specifies that this trigger should be executed before any updates to the sessionStartTime or sessionEndTime columns of the Session table.
FOR EACH ROW	Specifies that the trigger should be fired once for every row affected by the update.
EXECUTE PROCEDURE SessionChangeFunction();	Links the trigger to the SessionChangeFunction function, indicating that this function should be executed whenever the trigger is activated.



```
UPDATE "Session"
SET sessionStartTime = '2023-10-14 10:30:00', sessionEndTime = '2023-10-14 11:30:00'
WHERE sessionID = 1;


SELECT * FROM "Session";
```

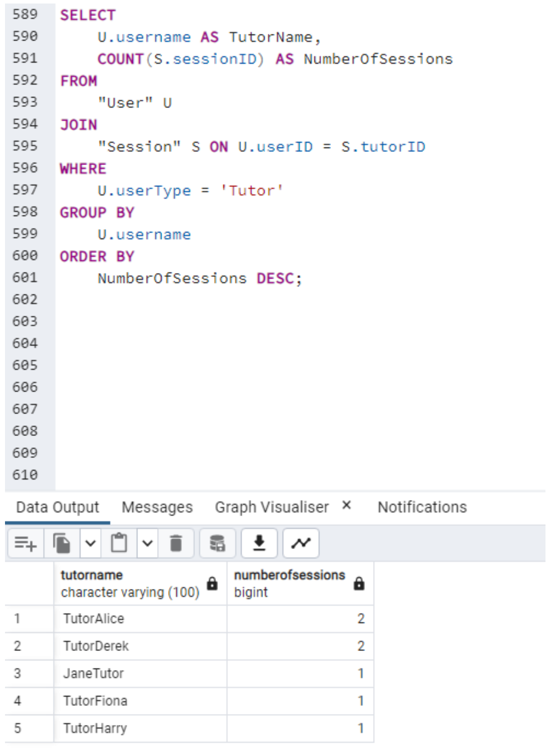
Data Output Messages Notifications

	sessionid [PK] numeric (12)	tutorid numeric (12)	studentid numeric (12)	assignmentid numeric (12)	sessionstarttime timestamp without time zone	sessionendtime timestamp without time zone	sessiontopic character varying (255)
1	2	3	1	[null]	2023-10-02 10:00:00	2023-10-02 11:00:00	Science
2	501	101	201	401	2023-10-14 10:00:00	2023-10-14 11:00:00	Math Basics
3	1	2	1	1	2023-10-14 10:30:00	2023-10-14 11:30:00	Math

- The first SELECT query displays the original state of the Session table.
- The UPDATE statement then modifies the timings for the session with sessionID = 1.
- The final SELECT query showcases the changed state of the Session table.
- As a result of the UPDATE operation, the SessionChangeTrigger activates and calls the SessionChangeFunction, resulting in the creation of a new entry in the SessionHistory table that logs this change.

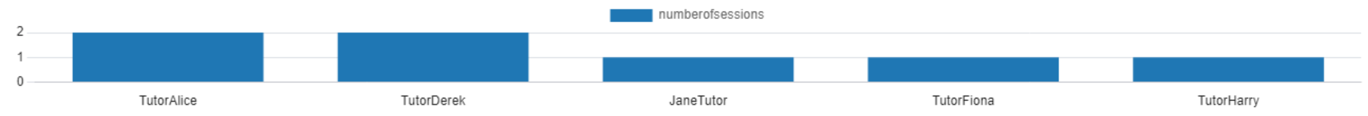
Data Visualizations

Visualization 1: Number of Sessions per Tutor



```
SELECT
    U.username AS TutorName,
    COUNT(S.sessionID) AS NumberOfSessions
FROM
    "User" U
JOIN
    "Session" S ON U.userID = S.tutorID
WHERE
    U.userType = 'Tutor'
GROUP BY
    U.username
ORDER BY
    NumberOfSessions DESC;
```

	tutorname character varying (100)	numberofsessions bigint
1	TutorAlice	2
2	TutorDerek	2
3	JaneTutor	1
4	TutorFiona	1
5	TutorHarry	1

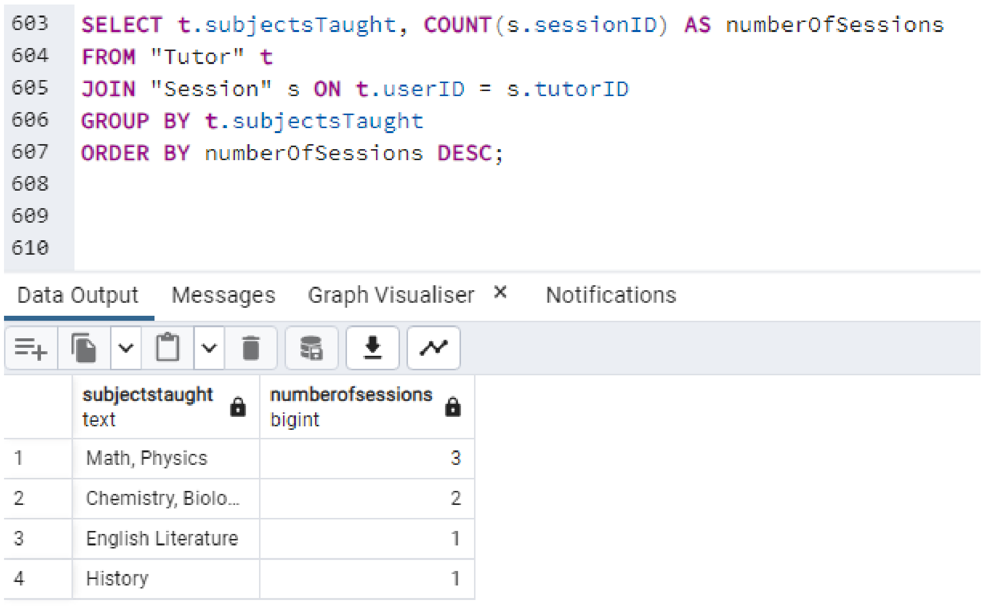


Data Story:
To ensure the efficient operation of AchieveMe Tutor, understanding the distribution of workload among tutors is crucial. It raises the question, "Which tutors have the highest session count, and are there tutors with very few sessions?"

Upon analyzing the bar chart showing the number of sessions conducted by each tutor, a few observations come to light. First, a small group of tutors appear to be the primary contributors, conducting a significant percentage of all sessions. This could mean that these tutors are highly sought after by students or are more active and available. On the flip side, there are tutors with considerably fewer sessions, possibly indicating either their recent induction into the platform or less activity.

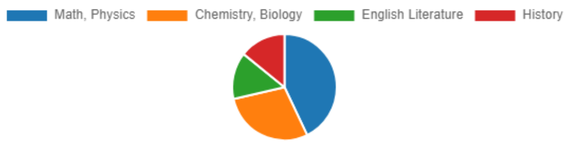
Using this information, AchieveMe Tutor can initiate several actions. They might want to gather feedback from students about the popular tutors to understand what makes them sought-after, which can be used as a training guide for others. Additionally, they could engage with less active tutors to understand and address any challenges they might face. This balanced distribution of sessions ensures that tutors do not get overwhelmed and students have a broad selection of available educators.

Visualization 2: Proportion of Subjects Taught Across All Tutors



```sql
SELECT t.subjectsTaught, COUNT(s.sessionID) AS numberOfSessions
FROM "Tutor" t
JOIN "Session" s ON t.userID = s.tutorID
GROUP BY t.subjectsTaught
ORDER BY numberOfSessions DESC;
```

	subjectstaught text	numberofsessions bigint
1	Math, Physics	3
2	Chemistry, Biolo...	2
3	English Literature	1
4	History	1



Data Story:
Diversity in subjects offered is a hallmark of a comprehensive tutoring platform. It prompts the query, "How varied are our tutors in terms of the subjects they teach?"

The pie chart reveals a snapshot of the subjects' distribution taught by our tutors. Certain subjects dominate, reflecting their popularity or the platform's strength in those areas. Conversely, subjects with smaller slices might indicate areas where there's potential for expansion or where student demand isn't as high.

These insights can be harnessed in several strategic ways. Prominent subjects can be highlighted in marketing campaigns to attract more students looking for those subjects. For the subjects with less representation, AchieveMe Tutor can either decide to invest in them (by hiring more specialized tutors or marketing them more) or understand why those subjects are less popular and if there's a genuine demand to meet.

.

Summary and Reflection

My database is designed for AchieveMe Tutor, an online tutoring platform that aims to bridge educational gaps by connecting students with qualified tutors. The platform's primary mission is to offer tailored learning experiences that cater to individual student needs. Unlike conventional tutoring services that restrict students to a particular institution or geographic location, AchieveMe Tutor uses the power of technology to bring learning to the fingertips, making quality education more accessible.

The structural database rules and the conceptual ERD for my database design encapsulate the essential entities of User, Session, Assignment, and their subtypes like Tutor, Student, and Parent. These entities and their relationships reflect the multifaceted interactions on the platform. The design illustrates a hierarchy, differentiating between Tutors and Students under the User entity, acknowledging the unique attributes and operations associated with each.

The physical ERD for the DBMS aligns with the conceptual representation, incorporates best practices like synthetic keys, and contains crucial attributes vital for the platform's smooth operation. An SQL script delineates table creations that strictly adhere to the DBMS physical ERD. Notable indexes have been introduced to expedite database access. Stored procedures transactionally populate the database with data, ensuring that the platform is not just theoretical but practically viable.

Several pertinent questions that AchieveMe Tutor might face have been addressed with SQL queries, and data visualizations have been presented, elucidating the stories behind the numbers. These insights are pivotal in shaping the platform's strategy, improving its offerings, and understanding user behavior.

Reflecting on the journey and the development of this database, it has indeed been a fulfilling endeavor. It's exhilarating to conceptualize a database that could seamlessly integrate with the AchieveMe Tutor platform. While visualizing the user interface and interactions, the robustness of the current database assures that the foundation is strong. While there's always room for enhancement, the current design is a solid stepping stone towards a revolution in online tutoring.