**Report on Database Schema Creation for Budgetwise Application PART 1**

**Overview:**

I wrote SQL code to create a database schema for a budget management application named "Budgetwise." The schema includes several tables to store data related to users, security questions, budgets, categories, budget accounts, vendors, alerts, reports, transactions, and audit logs. Each table has specific columns and constraints to ensure data integrity and relationships between different entities.

**Details:**

1. **Database Connection:**
   * Established a connection to the SQLite database named 'Budgetwise.db'. If the database does not exist, it will be created.
   * Enabled foreign key constraints using PRAGMA foreign\_keys = 1.
2. **Tables Created:**
   * **users**: Stores user information.
     + Columns: userID, username, passwordhash, theme, notificationsEnabled, Language, default\_chart, createdAt, updatedAT.
   * **security\_questions**: Stores security questions for user authentication.
     + Columns: questionID, question\_text.
   * **user\_security\_questions**: Links users to their security questions and stores their answers.
     + Columns: the\_user, the\_question, answer.
   * **budget**: Stores budget details for users.
     + Columns: budgetID, the\_user, budget\_Name, allocatedMonitaryAmount, start\_Date, end\_Date.
   * **category**: Stores categories for budget transactions.
     + Columns: categoryID, the\_user, catagory\_name, description, catagory\_usage\_ranking.
   * **budget\_accounts**: Stores accounts within budgets, linked to users and categories.
     + Columns: budget\_accounts\_id, the\_user, the\_category, the\_budget, account\_name, account\_type, balance, savings\_goal, created\_at, updated\_at, notes, importance\_rating.
   * **vendor**: Stores vendor information for transactions.
     + Columns: vendor\_id, the\_user, vendor\_name, description, vendor\_usage\_ranking.
   * **alerts**: Stores alerts for budget accounts.
     + Columns: alert\_id, budget\_accounts\_id, the\_user, threshhold\_amount, alert\_type, frequency, status, created\_at, updated\_at, active\_from, actice\_until.
   * **report**: Stores reports generated for users.
     + Columns: report\_id, the\_user, report\_type, report\_date, report\_data, report\_frequency.
   * **transactions**: Stores transaction details.
     + Columns: transaction\_id, budget\_accounts\_id, the\_user, the\_category, vendor\_id, transaction\_type, amount, transaction\_date, description, recurring, importance\_rating.
   * **audit\_log**: Stores audit logs for transactions and actions performed by users.
     + Columns: log\_id, the\_user, transaction\_id, action\_type, related\_table, record\_id, timestamp.
3. **Constraints:**
   * Primary keys are defined for each table to ensure unique identification of records. From my attempts I only had could have 1 Primary key
   * Foreign keys are used to establish relationships between tables, ensuring referential integrity.
   * The user\_security\_questions table has a composite primary key combining the\_user and the\_question.
4. **Commit and Close:**
   * Changes are committed to the database using conn.commit().
   * **The connection to the database is closed using conn.close().**

**Research:**

Since it has been about 4 years since I took database, I spent a lot of my time looking up how to create a database in SQLite since I have not had to work in and on a database from the code side.

1. **Struggles**
   1. **Relationships**
      1. While working on the code I struggled to get the the foreign keys implanted into each table, getting this working took a lot of stack overflow, reddit, and even ai to figure out how to get it working. I ended up finding out in the main source that the Reference exists in the same line as the creation
         1. the\_user INTEGER PRIMARY KEY REFERENCES users(userID)
   2. **Multiple choices**
      1. In our program, we plan to have set choices for the user to use to set up personal setting. At first, I was setting those up as text and going to let the team sort it out but later decided that to make it easier on the team I would set them up as integers for easy manipulation on their part
         1. Originally Language TEXT, now Language INTEGER
   3. Decimals
      1. At first, I had no idea how the Decimal works, in both SQL and SQLite. After doing some research I ended up finding out it was a REAL object

**PART 1**

For now the code just creates the Database adds the info the closes it. I will be turning it into a class that does all these items. I do not know when I will do this so I will update here when I do make these changes, cause I wanted to make sure I was creating the table properly