PERSONAL FINANCE TRACKER

Name: Pankuri Khare
Student Id: 921385202
GitHub Username: pankurik

Checkpoint #	**Date Submitted**
Checkpoint 1	Feb 20, 2024
Checkpoint 2	May 01, 2024
Checkpoint 3	June 14, 2024

Table of Contents

Project Description

Use Case

- **Actor**: Alex
- **Description**: Alex, a diligent budgeter, struggles with scheduling their financial activities, like paying bills, setting aside savings, and checking account balances. Occasionally, Alex sets reminders for two activities at the same time, which causes stress and can lead to missed payments or unbalanced accounts. They need a system that helps prevent scheduling conflicts and keeps their financial life in harmony.

Alex loves to keep their finances in check with meticulous budgeting and timely bill payments. To help Alex manage these tasks without a hitch, the Personal Finance Tracker is equipped with a user-friendly calendar that neatly displays all scheduled financial activities. But what if Alex accidentally sets two tasks, like depositing savings and paying off a credit card, for the same time? No worries! Our tracker is designed to catch this and instantly alerts Alex about the clash, ensuring they can address each financial task with the attention it deserves.
Life can be unpredictable, and sometimes Alex needs to shuffle things around. Our tracker makes rescheduling a breeze, and with its smart conflict detection, Alex can confidently move tasks to a new time slot without fear of overlapping. This way, the tracker not only helps Alex prevent double-booking but also brings peace of mind to their financial planning. It's the perfect tool for anyone who wants to keep their finances organized and on track, just like Alex.

Simplifying Personal Finance with Our Database System

The Heart of the Matter: Imagine a world where managing finances, from budgeting to investments and even splitting bills, is not just simpler but smarter. That's the world my Personal Finance Tracker database system aims to create.

What Our Database System Offers
This system is like the brain behind the operation, designed to work seamlessly with financial tools to make them better. It's about giving these tools a deeper understanding of what you need, when you need it, and how you can achieve your financial goals more effectively.

Key Features
- **Smart Budgeting Insights**: This database enhances budgeting tools by providing personalized budgeting insights, making sure you're always on track but flexible enough to enjoy life.
- **Enhanced Investment Tracking**: This database empower investment platforms with detailed analysis and insights, simplifying the complex world of investing for everyone.
- **Group Financial Management**: For tools like Splitwise, this adds a layer of collective financial planning, making it easier to manage shared goals and expenses with friends and family.

Elevating Existing Tools
- **Budgeting Tools (e.g., Simplifi by Quicken)**: With this database, these tools can offer more than just tracking; they can provide personalized advice and predictions, helping users stay ahead of their finances.
- **Investment Platforms (e.g., Personal Capital)**: This system can provide a deeper dive into users' investment patterns, offering actionable insights to make informed decisions.
- **Expense Splitting (e.g., Splitwise)**: Beyond splitting bills, this database system introduces a way to collaboratively manage and save for shared goals, turning tedious tasks into shared successes.

In Summary
My Personal Finance Tracker database system is the foundation for making personal finance management tools more intuitive and effective. It's about making the complex simple, the tedious enjoyable, and the personal financial goals achievable.

Functional Database Requirements

1. User
1.1. A user shall create only one account (One-to-One).
1.2. A user shall be able to record multiple transactions (One-to-Many).
1.3. A user shall set multiple savings goals (One-to-Many).
1.4. A user can participate in multiple shared savings groups (Many-to-Many).
1.5. A user can share transaction responsibilities with other users (Many-to-Many, Aggregation).
1.6. A user can have multiple roles (Admin, Standard User) (ISA).
1.7. A user shall be able to update their personal profile information.
1.8. A user shall be able to deactivate their account.
1.9. A user can follow multiple other users to share financial insights (Recursive, Many-to-Many).
1.10. A user can be part of multiple budgeting groups (Many-to-Many).
1.11. Users can customize the dashboard to display preferred financial metrics.
1.12. Users can set privacy settings to control the visibility of their financial data.
1.13. Users can invite other users to view or manage their financial data (Many-to-Many).

2. Account
2.1. An account shall be associated with one user (One-to-One).
2.2. An account can be linked to multiple financial institutions (One-to-Many).
2.3. An account shall have multiple transactions (One-to-Many).
2.4. An account can have multiple associated investment portfolios (One-to-Many).
2.5. An account can be set as either checking, savings, or investment (ISA).
2.6. Accounts can be classified into sub-accounts for detailed financial organization (Recursive).
2.7. Users can merge or split account data when consolidating finances.

3. Transaction
3.1. A transaction belongs to only one account (Many-to-One).
3.2. A transaction is categorized under one category (Many-to-One).
3.3. A transaction can be part of a split expense among users (Many-to-Many, Aggregation).
3.4. A transaction can recur on a scheduled basis (Recursive).
3.5. A transaction can be flagged as either expense or income (ISA).
3.6. Transactions can be tagged for easier search and organization (Aggregation).
3.7. Users can create custom rules for auto-categorizing transactions.

4. Category
4.1. A category can encompass multiple transactions (One-to-Many).
4.2. Categories can be nested under a parent category (Aggregation, Recursive).
4.3. A category shall be assignable to budget plans (One-to-Many).
4.4. Users can create custom spending categories beyond the default set.
4.5. Categories can have budget alerts set individually (Aggregation).

5. Goal

5.1. A goal is set by a user and linked to their account (Many-to-One).
5.2. A goal can have contributions from multiple users in shared savings groups (Many-to-Many).
5.3. A goal can be categorized into short-term, medium-term, or long-term (ISA).
5.4. A goal can include multiple sub-goals (Recursive).
5.5. Users can adjust goal timelines and amounts, with the system recalculating the saving pace.
5.6. Goals can be prioritized, with notifications adjusted accordingly (Aggregation).

6. Shared Savings Group
6.1. A shared savings group shall include multiple users (One-to-Many).
6.2. A group can have a group-specific budget plan (One-to-One).
6.3. Groups can set collective financial goals (One-to-Many).
6.4. Groups can vote on financial decisions or changes to the group's goals (Aggregation).
6.5. The system automatically distributes funds towards group goals based on preset rules.

7. Investment Portfolio
7.1. An investment portfolio belongs to one account (Many-to-One).
7.2. A portfolio can include multiple types of assets (stocks, bonds) (One-to-Many).
7.3. A portfolio can be part of a diversified investment strategy (Aggregation).
7.4. Portfolios can alert users based on performance thresholds (Aggregation).
7.5. Users can simulate portfolio performance based on historical data (Aggregation).

8. Report
8.1. A report is generated for a user account summarizing financial activity (One-to-One).
8.2. Reports can compare user spending across different categories (Many-to-One).
8.3. Reports can be customized by the user to include specific data points and time frames (Aggregation).
8.4. Reports can be scheduled for automatic generation and delivery via email.
8.5. Custom reports can be created by users, selecting specific data points and time frames.

9. Budget Plan
9.1. A budget plan is associated with one or more categories (One-to-Many).
9.2. A user can have multiple budget plans (One-to-Many).
9.3. Budget plans can be shared among users in a group (Many-to-Many).
9.4. Budgets can adjust automatically based on user spending habits (Aggregation).
9.5. Users receive suggestions for budget adjustments based on financial goals.

10. Notifications
10.1. Users receive notifications based on account activity (One-to-Many).
10.2. Notifications can be customized based on user preferences (Aggregation).
10.3. Notifications for shared expenses are sent to all involved parties.
10.4. Users can receive motivational messages as they progress towards their financial goals.

11. Financial Institution
11.1. A financial institution can be linked to multiple accounts, providing financial data and transaction capabilities (One-to-Many).
11.2. Financial institutions offer real-time updates on account balances and transactions (Aggregation).

12. Scheduled Payment
12.1. Users can schedule payments for bills or savings, ensuring timely transactions (One-to-Many).
12.2. Scheduled payments allow for recurring financial commitments, automating the process (Recursive).

13. Budget Alert
13.1. Budget alerts notify users when spending approaches or exceeds the budget limit in a specific category (One-to-Many).
13.2. Users can customize alert thresholds for each budget category (Aggregation).

14. Currency Converter
14.1. Users can convert amounts between different currencies within transactions (One-to-Many).
14.2. The system provides real-time currency exchange rates for accurate conversions (Aggregation).

15. Expense Tracker
15.1. Users can log and categorize individual expenses to track spending habits (One-to-Many).
15.2. The tracker offers insights into spending patterns and potential savings areas (Aggregation).

16. Income Tracker
16.1. Users can record different sources of income, including recurring earnings (One-to-Many).
16.2. The tracker helps users visualize income trends and optimize financial planning (Aggregation).

17. Debt Management
17.1. Users can manage and track their debts, including loans and credit card balances (One-to-Many).
17.2. The system offers strategies for debt repayment and reduction (Aggregation).

18. Subscription Management
18.1. Users can keep track of subscriptions and recurring payments, avoiding unwanted renewals (One-to-Many).
18.2. Alerts notify users of upcoming subscription renewals or expirations (Aggregation).

19. Financial News Feed
19.1 Users receive personalized financial news and tips based on their financial goals and interests (One-to-Many).
19.2. The system aggregates news from various trusted sources (Aggregation).

20. Digital Receipts Organizer
20.1Users can upload and organize digital receipts for transactions (One-to-Many).
20.2The system offers OCR (Optical Character Recognition) to automatically extract information from uploaded receipts (Aggregation).

21. Credit Score Goals
21.1 Users can set goals for their desired credit score within a specified timeframe (One-to-One).
21.2 The system tracks progress towards the credit score goal and offers guidance on achieving it (Aggregation).

22. Credit Score Insights
22.1 The system analyzes users' financial behavior and its impact on their credit score, offering personalized insights (One-to-One).
22.2 Users can access historical credit score data to see trends over time (Aggregation).

Non-functional Database Requirements

1. Performance
1.1. The system shall handle multiple users accessing it at the same time without slowing down.
1.2. The system shall process transactions instantly to ensure up-to-date financial data.
1.3. The system shall provide real-time updates and alerts to users about their financial activities.

2. Security
2.1. The system shall only store encrypted passwords to protect user accounts.
2.2. The system shall validate all data against its expected format to prevent errors and security vulnerabilities. 2.3. The system shall automatically back up all data daily to ensure data recovery options.

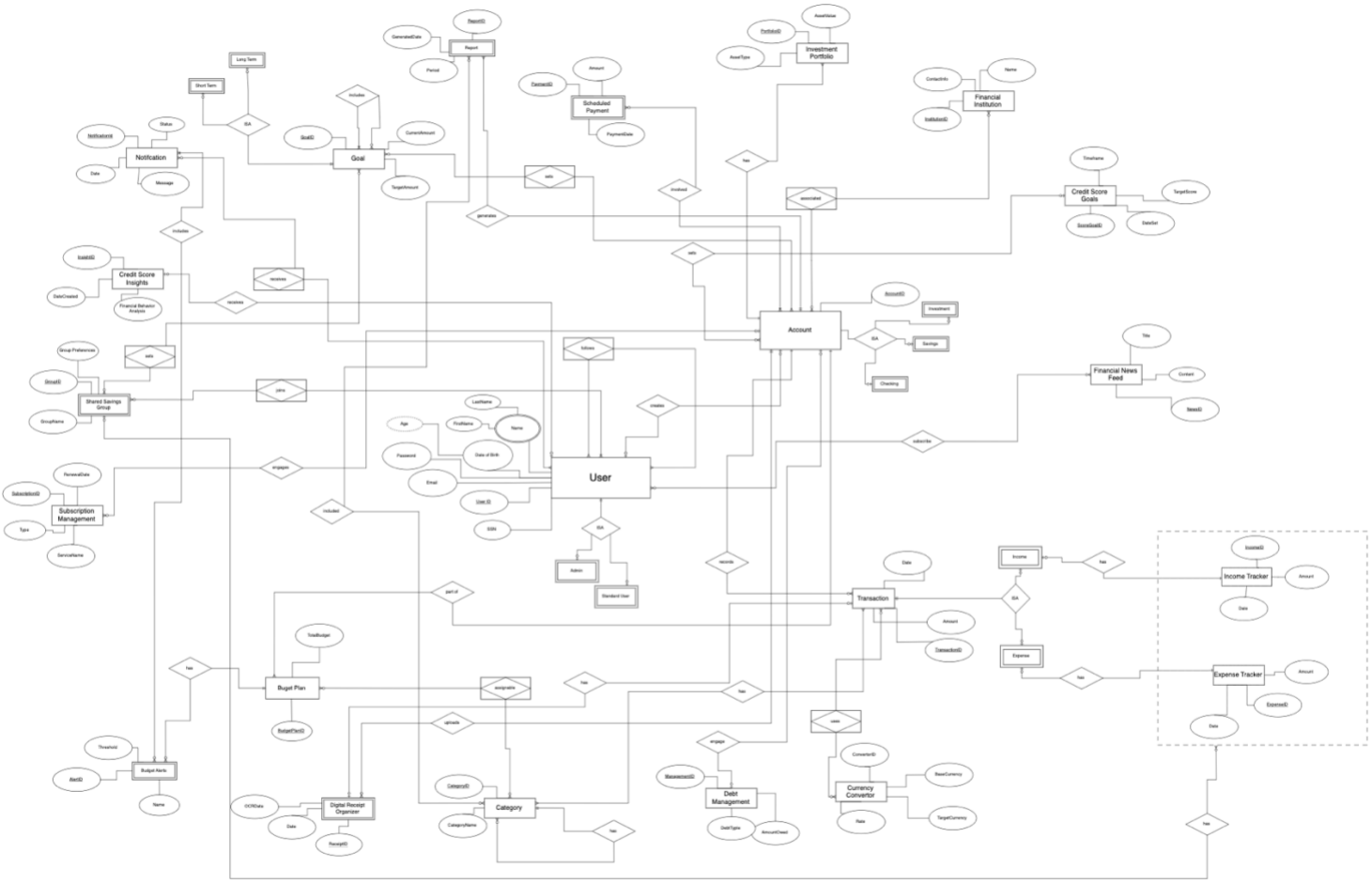
3. Storage
3.1. The system shall allocate at least 10 MB of memory for each table to efficiently manage and store data.
3.2. The system must support persistent storage, ensuring that data remains available and intact across system reboots and sessions.
3.3. The system shall automatically expand its storage capacity as needed to accommodate growing data volumes without manual intervention.
3.4. Sensitive data storage areas shall be isolated from general data storage to enhance security.

4. Capability
4.1. The system shall support transactions in multiple currencies with accurate conversion.
4.2. The system shall allow users to customize reports and dashboards according to their preferences.
4.3. The system shall integrate with external financial institutions for real-time data synchronization.

5. Media Storage
5.1. The system shall store large media files, like receipts, efficiently to ensure quick access.
5.2. The system shall compress images and documents to save storage space without losing quality.
5.3. The system shall secure media files with the same level of encryption as other sensitive data.

6. Usability
6.1. The system interface shall be intuitive and user-friendly, requiring minimal training for new users.
6.2. Response times for user interactions shall not exceed 2 seconds, ensuring a smooth user experience.
6.3. The system shall offer personalized user experiences based on individual preferences and usage patterns.
6.4. Accessibility features shall be incorporated to accommodate users with disabilities.
6.5. The system shall provide comprehensive help documentation and user support features.

Entity Relationship Diagram (ERD)



Entity Description

1. **User (Strong)**
   - UserID: key, numeric
   - Email: simple, alphanumeric
   - DateOfBirth: simple, date
   - Age: simple, numeric
   - Password: simple, alphanumeric
   - Name: simple, alphanumeric
   - FirstName: simple, alphanumeric
   - LastName: simple, alphanumeric
   - SSN: simple, alphanumeric

2. **Account (Strong)**
   - AccountID: key, numeric
   - UserID: foreign key, numeric
   - AccountType: simple, alphanumeric

3. **Transaction (Strong)**
   - TransactionID: key, numeric
   - Amount: simple, numeric
   - Date: simple, date
   - TransactionType: simple, alphanumeric

4. **Category (Strong)**
   - CategoryID: key, numeric
   - Name: simple, alphanumeric
   - Description: simple, alphanumeric

5. **Budget Plan (Strong)**
   - PlanID: key, numeric
   - UserID: foreign key, numeric
   - TotalBudget: simple, numeric

6. **Budget Alert (Weak)**
   - Name: simple, alphanumeric
   - AlertID: key, numeric
   - Threshold: simple, numeric

7. **Shared Savings Group (Weak)**
   - GroupID: key, numeric
   - GroupName: simple, alphanumeric
   - Description: simple, alphanumeric

8. **Investment Portfolio (Strong)**
   - GroupID: key, numeric

- GroupName: simple, alphanumeric
- GroupPreferences: simple, alphanumeric

9. **Financial News Feed (Strong)**
   - NewsID: key, numeric
   - Title: simple, alphanumeric
   - Content: simple, alphanumeric

10. **Digital Receipts Organizer (Weak)**
    - ReceiptID: key, numeric
    - OCRData: simple, alphanumeric
    - Date: simple, date

11. **Credit Score Goals (Strong)**
    - ScoreGoalID: key, numeric
    - DateSet: simple, date
    - TargetScore: simple, numeric
    - Timeframe: simple, alphanumeric

12. **Credit Score Insights (Strong)**
    - InsightID: key, numeric
    - DateCreated: simple, date
    - FinancialBehaviorAnalysis: simple, alphanumeric

13. **Expense Tracker (Strong)**
    - ExpenseID: key, numeric
    - Date: simple, date
    - Amount: simple, numeric

14. **Income Tracker (Strong)**
    - IncomeID: key, numeric
    - Date: simple, date
    - Amount: simple, numeric

15. **Debt Management (Strong)**
    - DebtID: key, numeric
    - DebtType: simple, alphanumeric
    - AmountOwed: simple, numeric

16. **Subscription Management (Strong)**
    - SubscriptionID: key, numeric
    - ServiceName: simple, alphanumeric
    - Type: simple, alphanumeric
    - RenewalDate: simple, date

17. **Scheduled Payment (Weak)**
    - PaymentID: key, numeric
    - PaymentDate: simple, date
    - Amount: simple, numeric

18. **Currency Converter (Strong)**
    - ConverterID: key, numeric
    - BaseCurrency: simple, alphanumeric
    - TargetCurrency: simple, alphanumeric
    - Rate: simple, numeric

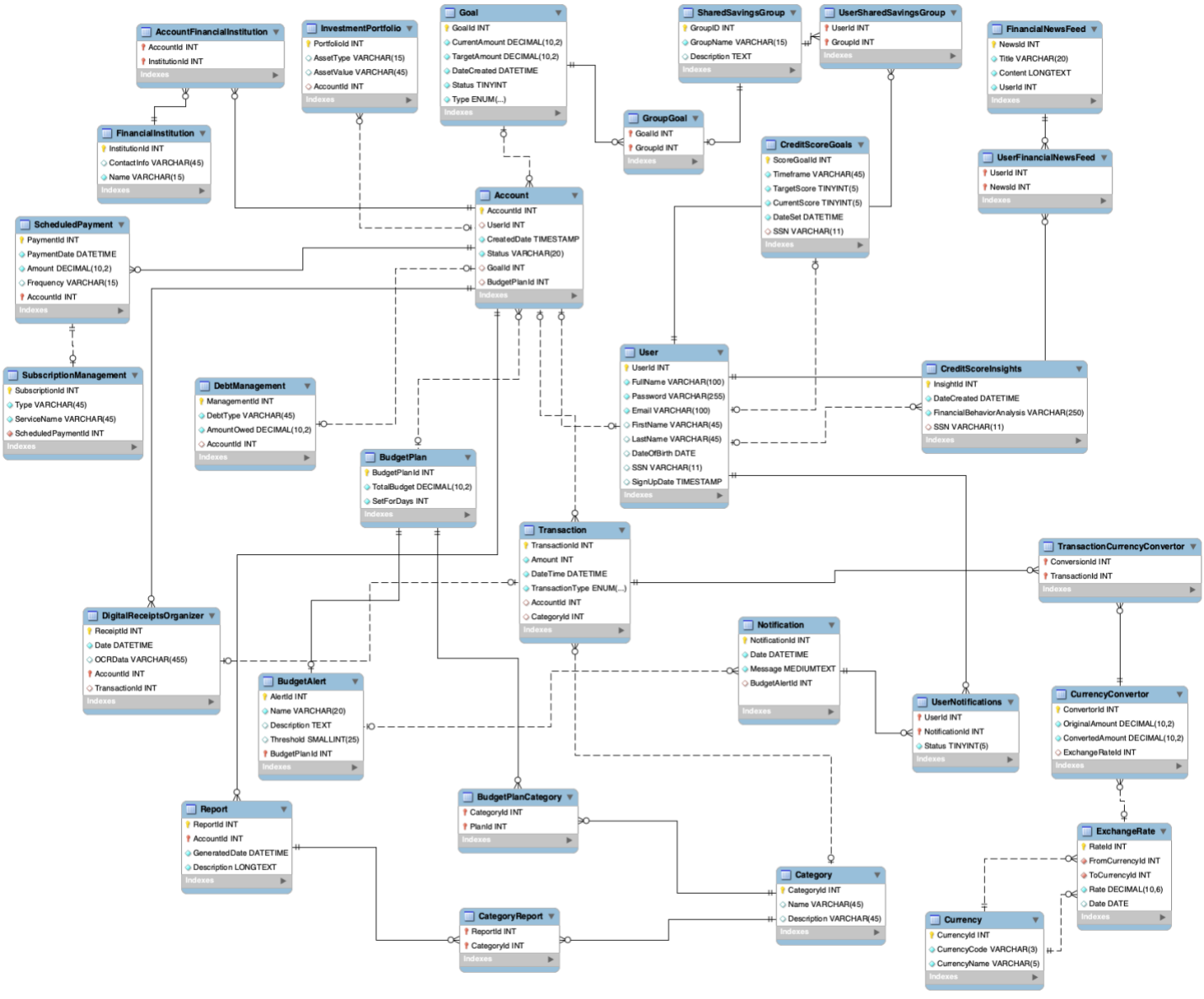
19. **Financial Institution (Strong)**
    - InstitutionID: key, numeric
    - Name: simple, alphanumeric
    - ContactInfo: simple, alphanumeric

20. **Report (Weak)**
    - ReportID: key, numeric
    - GeneratedDate: simple, date
    - Type: simple, alphanumeric

21. **Goal (Strong)**
    - GoalID: key, numeric
    - CurrentAmount: simple, numeric
    - TargetAmount: simple, numeric
    - Type: simple, alphanumeric

22. **Notification (Strong)**
    - NotificationID: key, numeric
    - Date: simple, date
    - Message: simple, alphanumeric
    - Status: simple, alphanumeric

Enhanced Entity Description (EER)



Constraints Description

Table	FK	On Delete	On Update	Comment
Account	UserId	Cascade	Cascade	The account referencing the deleted user must be deleted because the account had a one-to-one relationship with this user
Account	GoalId	Set Null	Cascade	If a goal is deleted, the GoalId in the Account table is set to NULL to dissociate the goal without deleting the account. Updates to GoalId are cascaded.
Account	BudgetPlanId	Set Null	Cascade	If a budget plan is deleted, the BudgetPlanId in the Account table is set to NULL to dissociate the budget plan without deleting the account. Updates to BudgetPlanId are cascaded.
AccountFinancialInsitution	AccountId	Cascade	Cascade	If an account is deleted, all associated financial institution records are also deleted to maintain integrity. Updates to AccountId are cascaded.
AccountFinancialInsitution	InstitutionId	Cascade	Cascade	If a financial institution is deleted, the associated records in AccountFinancialInstitution are also deleted. Updates to InstitutionId are cascaded.
BudgetAlert	BudgetPlanId	Cascade	Cascade	If a budget plan is deleted, the associated budget alerts are also deleted as they are dependent on the budget plan. Updates to BudgetPlanId are cascaded.
BudgetPlanCategory	CategoryId	Cascade	Cascade	If a category is deleted, the associated budget plan categories are also deleted as they are dependent on the category. Updates to CategoryId are cascaded.
BudgetPlanCategory	BudgetPlanId	Cascade	Cascade	If a budget plan is deleted, the associated categories are also deleted as they are dependent on the budget

				plan. Updates to BudgetPlanId are cascaded.
CategoryReport	ReportId	Cascade	Cascade	If a report is deleted, the associated category reports are also deleted as they are dependent on the report. Updates to ReportId are cascaded.
CategoryReport	CatergoryId	Cascade	Cascade	If a category is deleted, the associated category reports are also deleted as they are dependent on the category. Updates to CategoryId are cascaded.
CreditScoreGoals	SSN	Cascade	Cascade	If a social security number (SSN) is deleted, the associated credit score goals are also deleted as they are dependent on the SSN. Updates to SSN are cascaded.
CreditScoreInsights	SSN	Cascade	Cascade	If a social security number (SSN) is deleted, the associated credit score insights are also deleted as they are dependent on the SSN. Updates to SSN are cascaded.
CurrencyConvertor	ExchangeRateId	Cascade	Cascade	If an exchange rate is deleted, the associated currency conversions are also deleted to maintain integrity. Updates to ExchangeRateId are cascaded.
DebtManagement	AccountId	Cascade	Cascade	If an account is deleted, the associated debt management records are also deleted as they are dependent on the account. Updates to AccountId are cascaded.
DigitalReceiptsOrganizer	AccountId	Cascade	Cascade	If an account is deleted, the associated digital receipts are also deleted as they are dependent on the account. Updates to AccountId are cascaded.
DigitalReceiptsOrganizer	TransactionId	Cascade	Cascade	If a transaction is deleted, the associated digital receipts are also deleted as they are dependent on the transaction. Updates to TransactionId are cascaded.
ExchangeRate	CurrencyId	Cascade	Cascade	If a currency is deleted, the associated exchange rates are also deleted to maintain

				integrity. Updates to CurrencyId are cascaded.
GroupGoal	GoalId	Cascade	Cascade	If a goal is deleted, the associated group goals are also deleted as they are dependent on the goal. Updates to GoalId are cascaded.
GroupGoal	GroupId	Cascade	Cascade	If a group is deleted, the associated group goals are also deleted as they are dependent on the group. Updates to GroupId are cascaded.
InvestmentPortfolio	AccountId	Cascade	Cascade	If an account is deleted, the associated investment portfolio records are also deleted as they are dependent on the account. Updates to AccountId are cascaded.
Notification	BudgetAlertId	Cascade	Cascade	If a budget alert is deleted, the associated notifications are also deleted as they are dependent on the budget alert. Updates to BudgetAlertId are cascaded.
Report	AccountId	Cascade	Cascade	If an account is deleted, the associated reports are also deleted as they are dependent on the account. Updates to AccountId are cascaded.
ScheduledPayment	AccountId	Cascade	Cascade	If an account is deleted, the associated scheduled payments are also deleted as they are dependent on the account. Updates to AccountId are cascaded.
SubscriptionManagement	ScheduledPaymentId	Cascade	Cascade	If a scheduled payment is deleted, the associated subscriptions are also deleted as they are dependent on the scheduled payment. Updates to ScheduledPaymentId are cascaded.
Transaction	AccountId	Cascade	Cascade	If an account is deleted, the associated transactions are also deleted as they are dependent on the account. Updates to AccountId are cascaded.

Transaction	CategoryId	Set NULL	Cascade	If a category is deleted, the CategoryId in the Transaction table is set to NULL to dissociate the category without deleting the transaction. Updates to CategoryId are cascaded.
TransactionCurrencyConvertor	ConversionId	Cascade	Cascade	If a conversion is deleted, the associated currency conversions are also deleted to maintain integrity. Updates to ConversionId are cascaded.
TransactionCurrencyConvertor	TransactionId	Cascade	Cascade	If a transaction is deleted, the associated currency conversions are also deleted as they are dependent on the transaction. Updates to TransactionId are cascaded.
UserFinancialFeed	UserId	Cascade	Cascade	If a user is deleted, the associated financial feeds are also deleted as they are dependent on the user. Updates to UserId are cascaded.
UserFinancialFeed	NewsId	Cascade	Cascade	If a news item is deleted, the associated financial feeds are also deleted as they are dependent on the news item. Updates to NewsId are cascaded.
UserNotifications	UserId	Cascade	Cascade	If a user is deleted, the associated notifications are also deleted as they are dependent on the user. Updates to UserId are cascaded.
UserNotifications	NotificationId	Cascade	Cascade	If a notification is deleted, the associated user notifications are also deleted as they are dependent on the notification. Updates to NotificationId are cascaded.
UserSharedSavingsGroup	UserId	Set NULL	Cascade	If a user is deleted, the UserId in UserSharedSavingsGroup is set to NULL to preserve the group for other users who are part of it. Updates to UserId are cascaded.
UserSharedSavingsGroup	GroupId	Set NULL	Cascade	If a group is deleted, the GroupId in UserSharedSavingsGroup is set to NULL to maintain user

				records that may belong to multiple groups. Updates to GroupId are cascaded.