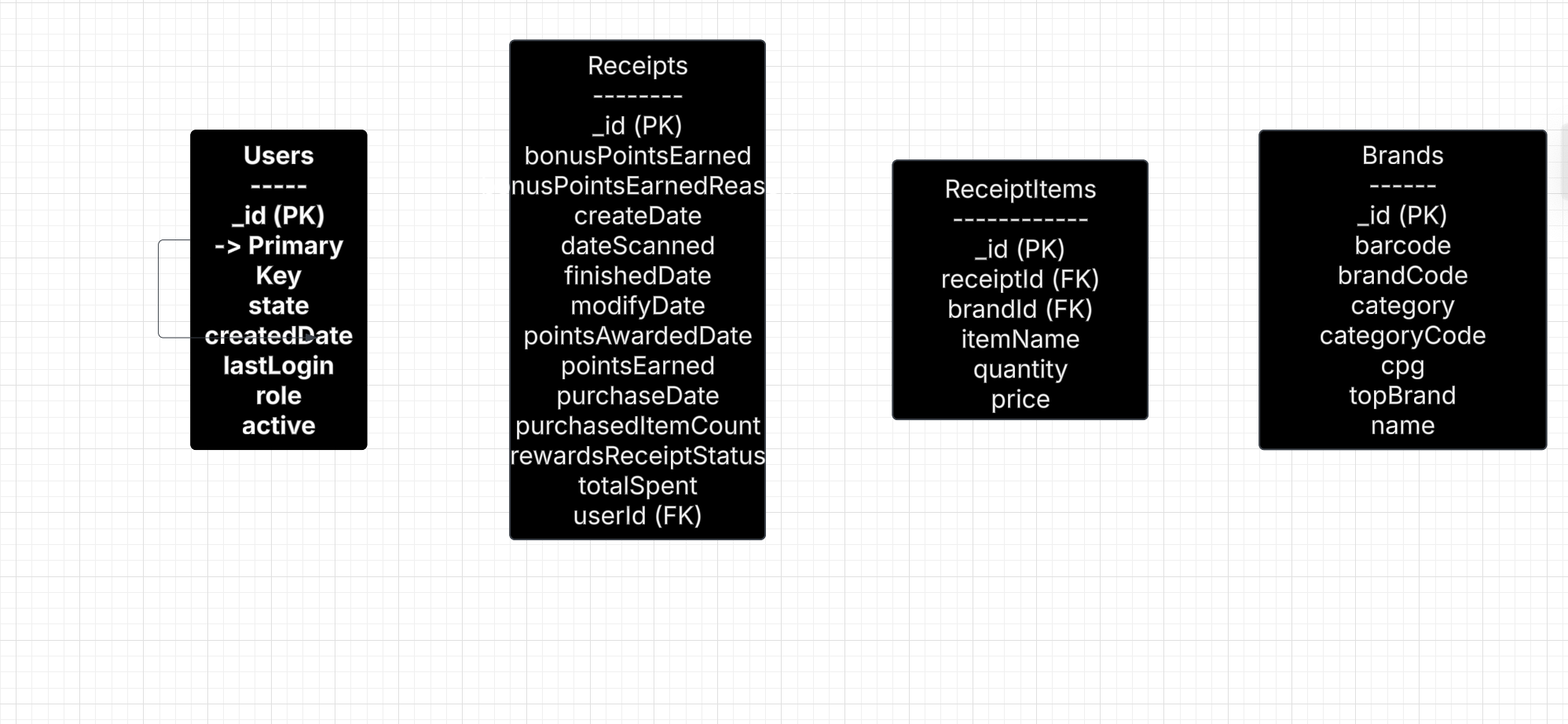
**Review unstructured JSON data and diagram a new structured relational data model**

By analyzing users, brands and receipts diagrams, I have designed a snowflake schema.

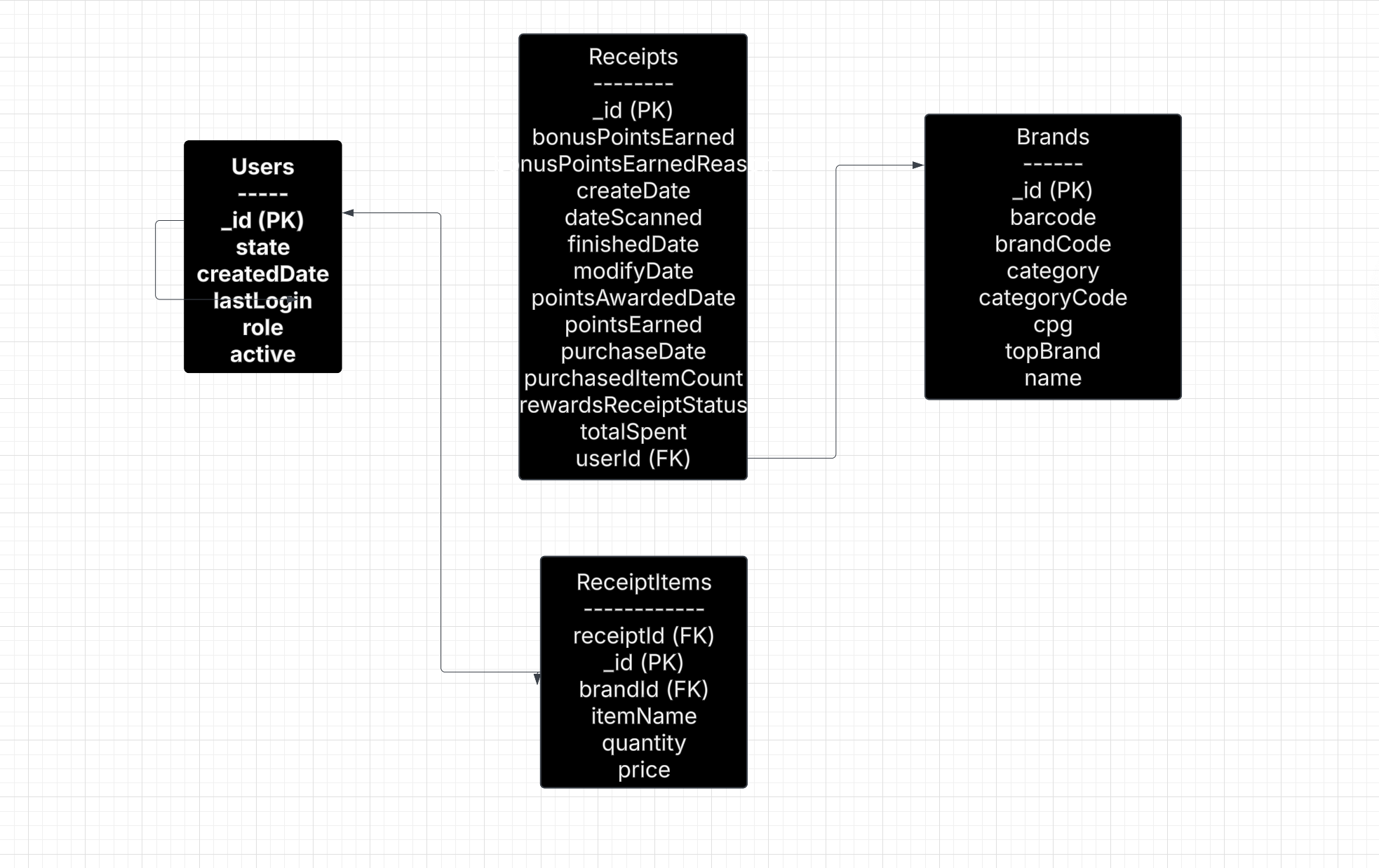


I created the **ReceiptItems** table as part of the structured relational data model. This table was not explicitly provided in the original JSON data, but it is necessary to normalize the data and properly represent the relationships between **Receipts**, **Brands**, and the items purchased.

### **Why I Created ReceiptItems**

1. **Normalization**:
   * The rewardsReceiptItemList field in the **Receipts** JSON contains a list of items purchased. Embedding this list directly in the **Receipts** table would violate the principles of normalization (specifically, the **First Normal Form (1NF)**, which requires atomic values).
   * By creating a separate **ReceiptItems** table, we can store each item as a distinct row, making the data easier to query and maintain.
2. **Relationship with Brands**:
   * Each item in the rewardsReceiptItemList likely corresponds to a specific brand. To capture this relationship, we need a table (**ReceiptItems**) that links items to their respective brands via a foreign key (brandId).
3. **Scalability and Flexibility**:
   * A separate **ReceiptItems** table allows for easier updates and queries. For example:
     + Adding new attributes to items (e.g., discounts, categories) can be done without modifying the **Receipts** table.
     + Querying for specific items or brands becomes more efficient.

**Entity relationship:**

****

I created this ER diagram to structure the unstructured JSON data into a relational model that is clean, efficient, and scalable. Here’s why I designed it this way:

#### **1. Users Table**

* **Purpose**: Stores information about users who scan receipts.
* **Fields**: \_id (primary key), state, createdDate, lastLogin, role, active.
* **Why**: This table is necessary to track user activity and link users to their receipts. The \_id field is used as a foreign key in the **Receipts** table to establish the relationship between users and their receipts.

#### **2. Receipts Table**

* **Purpose**: Stores metadata about each receipt scanned by users.
* **Fields**: \_id (primary key), bonusPointsEarned, bonusPointsEarnedReason, createDate, dateScanned, finishedDate, modifyDate, pointsAwardedDate, pointsEarned, purchaseDate, purchasedItemCount, rewardsReceiptStatus, totalSpent, userId (foreign key to Users).
* **Why**: This table centralizes all receipt-related data. The userId field links each receipt to the user who scanned it, enabling queries like "Which user scanned the most receipts?"

#### **3. Brands Table**

* **Purpose**: Stores information about brands associated with items purchased.
* **Fields**: \_id (primary key), barcode, brandCode, category, categoryCode, cpg, topBrand, name.
* **Why**: This table is essential for categorizing and analyzing brand-level data. The \_id field is used as a foreign key in the **ReceiptItems** table to link items to their respective brands.

#### **4. ReceiptItems Table**

* **Purpose**: Stores details about individual items purchased on each receipt.
* **Fields**: \_id (primary key), receiptId (foreign key to Receipts), brandId (foreign key to Brands), itemName, quantity, price.
* **Why**: I created this table to normalize the data and avoid embedding lists of items directly in the **Receipts** table. This design:
  + Ensures atomicity (each item is a separate row).
  + Enables efficient queries like "Which brand has the most items purchased?" or "What are the top 5 items by quantity?"
  + Links items to their respective brands via the brandId field.

### **Relationships**

1. **Users -> Receipts (1 -> Many)**
   * A single user can have multiple receipts, but each receipt belongs to only one user.
   * Relationship: Users.\_id -> Receipts.userId.
2. **Receipts -> ReceiptItems (1 -> Many)**
   * A single receipt can have multiple items, but each item belongs to only one receipt.
   * Relationship: Receipts.\_id -> ReceiptItems.receiptId.
3. **Brands -> ReceiptItems (1 -> Many)**
   * A single brand can appear in multiple receipt items, but each item is associated with only one brand.
   * Relationship: Brands.\_id -> ReceiptItems.brandId.

### **Why I Chose This Structure**

1. **Normalization**:
   * I normalized the data to eliminate redundancy and ensure data integrity. For example, instead of duplicating user or brand information in every receipt or item, I used foreign keys to reference the respective tables.
2. **Scalability**:
   * This structure is scalable. For example, if new attributes are added to brands or items, they can be added to their respective tables without affecting the receipts or users.
3. **Query Efficiency**:
   * By separating receipts and items, I made it easier to query specific data. For example, finding the top brands by spend or the most purchased items can be done efficiently.
4. **Business Questions**:
   * This structure directly supports the business questions provided. For example:
     + To find the top 5 brands by receipts scanned, I can join **Receipts**, **ReceiptItems**, and **Brands**.
     + To calculate average spend or total items purchased, I can aggregate data from **Receipts** and **ReceiptItems**.