There have been multiple ways to implement ER diagrams. I am mentioning my assumptions and methods.

- If there is a FK relationship we are representing it using the Relationship not mapping directly.
- I have used google colab to analyse the data.

ER Diagram explanation:

- We have 4 entities
    - Users, receipts, receipts_item,brands
    - According to the data, barcodes in the brands are unique and non null so we are considering it as PK.

- We have 3 relationships:
    - UserReceiptRelationship, ReceiptItemRelationship,BrandItemRelationship

- The double line in between UserReceiptRelationship and Receipts is to represent full participation.
- In the remaining cases, I found scenarios where some rows are not available in the corresponding csv data.

The schema for this ER diagram:

1) Users:

```
CREATE TABLE USERS (
  CREATED_DATE DATE NOT NULL,
  BIRTH_DATE DATE NOT NULL,
  GENDER CHAR(1) NOT NULL,
  LAST_REWARDS_LOGIN DATE,
  STATE VARCHAR(255),
  SIGN_UP_PLATFORM VARCHAR(255),
  SIGN_UP_SOURCE VARCHAR(255),
  ID VARCHAR(255) NOT NULL,
  PRIMARY KEY (ID)
);
```

2)

```
CREATE TABLE Receipt (
  ID VARCHAR(255) NOT NULL,
  STORE_NAME VARCHAR(50),
  PURCHASE_DATE DATE,
  PURCHASE_TIME TIME,
  DATE_SCANNED DATE,
  TOTAL_SPENT DECIMAL(10, 2),
  REWARDS_RECEIPT_STATUS VARCHAR(255),
  USER_ID VARCHAR(255) NON NULL,
  USER_VIEWED BOOLEAN,
  PURCHASED_ITEM_COUNT INT,
```

```
  CREATE_DATE TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
  PENDING_DATE TIMESTAMP,
  MODIFY_DATE TIMESTAMP,
  FLAGGED_DATE TIMESTAMP,
  PROCESSED_DATE TIMESTAMP,
  FINISHED_DATE TIMESTAMP,
  REJECTED_DATE TIMESTAMP,
  NEEDS_FETCH_REVIEW BOOLEAN,
  DIGITAL_RECEIPT VARCHAR(255),
  DELETED BOOLEAN,
  NON_POINT_EARNING_RECEIPT BOOLEAN,
  PRIMARY KEY (ID),
  FOREIGN KEY (USER_ID) REFERENCES users(ID)
);
```

3)

```
CREATE TABLE Receipts_Item (
  REWARDS_RECEIPT_ID VARCHAR(255) ,
  ITEM_INDEX INT,
  REWARDS_RECEIPT_ITEM_ID VARCHAR(255) PRIMARY KEY,
  DESCRIPTION VARCHAR(255),
  BARCODE VARCHAR(255),
  BRAND_CODE VARCHAR(255),
  QUANTITY_PURCHASED INT,
  TOTAL_FINAL_PRICE DECIMAL(10, 2),
  POINTS_EARNED INT,
  REWARDS_GROUP VARCHAR(50),
  ORIGINAL_RECEIPT_ITEM_TEXT VARCHAR(255),
  MODIFY_DATE TIMESTAMP,
  FOREIGN KEY (REWARDS_RECEIPT_ID) REFERENCES Receipt(ID)
);
```

4)

```
CREATE TABLE Brands(
  ID VARCHAR(255),
  BARCODE VARCHAR(255) PRIMARY KEY,
  BRAND_CODE VARCHAR(255),
  CPG_ID VARCHAR(255),
  CATEGORY VARCHAR(255),
  CATEGORY_CODE VARCHAR(255),
  NAME VARCHAR(255),
  ROMANCE_TEXT VARCHAR(255),
  RELATED_BRAND_IDS VARCHAR(255)
);
```

1) What user bought the most expensive item?

Query : SELECT r.USER_ID, ri.TOTAL_FINAL_PRICE/ri.QUANTITY_PURCHASED as single_item_price
FROM Receipt r
INNER JOIN Receipts_Item ri ON r.ID = ri.REWARDS_RECEIPT_ID
ORDER BY single_item_price DESC
LIMIT 1;

Output:

```
+--------------------------+-------------------+
| USER_ID                  | single_item_price |
+--------------------------+-------------------+
| 617376b8a9619d488190e0b6 |       31005.990000 |
+--------------------------+-------------------+
1 row in set, 7774 warnings (1.04 sec)
```

2) Which user spent the most money in the month of August?

```
SELECT r.USER_ID, SUM(ri.TOTAL_FINAL_PRICE) AS total_spent
FROM Receipt r
INNER JOIN Receipts_Item ri ON r.ID = ri.REWARDS_RECEIPT_ID
WHERE MONTH(r.PURCHASE_DATE) = 8
GROUP BY r.USER_ID
ORDER BY total_spent DESC
LIMIT 1;
```

Output :

```
+--------------------------+-------------+
| USER_ID                  | total_spent |
+--------------------------+-------------+
| 609ab37f7a2e8f2f95ae968f |   157739.14 |
+--------------------------+-------------+
1 row in set (0.42 sec)
```