Receipts Table

```
-- Checking for duplicates on receipt_id's. Knowing each receipt_id should be unique,
there should not be any duplicates.
select
  receipt_id,
  count(*)
from
  Fetch_Data.receipts
group by
  1
having count(*) > 1;

-- Checking for duplicate rows. There are no duplicate rows.
select
  *,
  count(*)
from
  Fetch_Data.receipts
group by
  ALL
having count(*) > 1;

-- How many null values are there in the receipts table?

select
  sum(case when receipt_id is null then 1 else 0 end) as receipt_id_null_count,
  sum(case when createDate is null then 1 else 0 end) as createDate_null_count,
  sum(case when dateScanned is null then 1 else 0 end) as dateScanned_null_count,
  sum(case when finishedDate is null then 1 else 0 end) as finishedDate_null_count,
  sum(case when modifyDate is null then 1 else 0 end) as modifyDate_null_count,
  sum(case when purchaseDate is null then 1 else 0 end) as purchaseDate_null_count,
  sum(case when bonusPointsEarned is null then 1 else 0 end) as
bonusPointsEarned_null_count,
  sum(case when bonusPointsEarnedReason is null then 1 else 0 end) as
bonusPointsEarnedReason_null_count,
  sum(case when pointsEarned is null then 1 else 0 end) as pointsEarned_null_count,
  sum(case when purchasedItemCount is null then 1 else 0 end) as
purchasedItemCount_null_count,
  sum(case when rewardsReceiptStatus is null then 1 else 0 end) as
rewardsReceiptStatus_null_count,
  sum(case when totalSpent is null then 1 else 0 end) as totalSpent_null_count,
  sum(case when user_id is null then 1 else 0 end) as user_id_null_count
from
  Fetch_Data.receipts;

-- Checking the status counts for the finishedDate null values.

select
  rewardsReceiptStatus,
  count(*)
from
```

```sql
  Fetch_Data.receipts
where
  finishedDate is null
group by
  1;

-- Are all the statuses expected present?

select
  rewardsReceiptStatus,
  count(*)
from
  Fetch_Data.receipts
group by
  1;

-- Here we can see there is no "ACCEPTED" status. Could the "ACCEPTED" status be the
"FINISHED" status? This would require a conversation with the data lead to understand
better.


-- There are a considerable number of null values across all columns. Null values in
the following columns makes sense based off the following conditions/assumptions:
  -- bonusPointsEarned and bonusPointsEarnedReason: If no bonus points were earned,
these should be null.
  -- finishedDate: If a receipt is currently in the "SUBMITTED" status, the
finishedDate should be null. This is assuming the finishedDate show when the receipt
finished processing in the Fetch system.
-- Columns where null values should be cause for concern:
  -- purchaseDate: If a receipt is scanned, the purchase date should populate.
  -- finishedDate: For the records not in "SUBMITTED" status,this date should not be
null.
  -- pointsEarned: This column should populate with the points that were earned or
would have been earned if the receipt was not rejected. This is important in tracking
points earned by users. For instance, if a user needs a certain number of points for
specific rewards, Fetch will need to accurately track their points. This poses a major
problem as the customer would become unhappy not seeing their points accurately
reflecting.
  -- purchasedItemCount: The total number of items should populate from the receipt.
If there is ever a promotion through Fetch for purchasing a specific number of items
for a reward, this would need to accurately track them.
  -- totalSpent: Once again, this should populate from a receipt. Like above, if Fetch
has a spending threshold for rewards/promotions, this will need to accurately track
the total spent.
```

Users Table

```sql
-- Checking for duplicate user_id's. Knowing each user_id should be unique, there
should not be any duplicates.
```

```sql
select
  user_id,
  count(*)
from
  Fetch_Data.users
group by
  1
having count(*) > 1;

-- Comparing the unique count of user_id to the overall count of user_id's.

select
  count(distinct user_id) as unique_count,
  count(user_id) as total_users_present
from
  Fetch_Data.users;

-- Let's compare the total unique user_id's in the receipts table to the users table.
We can see there are more unique user_id's in the receipts table than the users table.
If the users table is meant to be the main source table for all users, then every
user_id in the receipts table should be present in the users' table. This is not the
case; therefore, the quality of the users table is further hindered.

select
  count(distinct user_id)
from
  Fetch_Data.receipts;

-- After running these queries, it is evident there is a data quality issue with the
user's table. Most of the records in the users table are duplicate records. There are
only 212 unique records compared to the 495 total user records present in the data
table. This shows that 57% of all records are duplicates.
```

Brands Table

```sql
-- For this table, I am assuming the brand_id and barcode together would make the
primary key. For the first step, I am checking if there are any duplicates.

select
  brand_id,
  barcode,
  count(*)
from
  Fetch_Data.brand
group by
  1,2
having count(*) > 1;

-- Checking null values.
```

```sql
select
  sum(case when brand_id is null then 1 else 0 end) as brand_id_null_count,
  sum(case when barcode is null then 1 else 0 end) as barcode_null_count,
  sum(case when category is null then 1 else 0 end) as category_null_count,
  sum(case when categoryCode is null then 1 else 0 end) as categoryCode_null_count,
  sum(case when name is null then 1 else 0 end) as name_null_count,
  sum(case when topBrand is null then 1 else 0 end) as topBrand_null_count,
  sum(case when brandCode is null then 1 else 0 end) as brandCode_null_count
from
  Fetch_Data.brand

  -- Here we can see there are quite a few null values within the category,
categoryCode, topBrand, and brandCode columns. The quality issue this poses is not
being able to understand what exactly Fetch users are purchasing. Without having a
full understanding of this, Fetch is limited to what it can promote to its user base.
  -- Is this a problem because of the source system coding? Do we not have sufficient
data to fill out this information?
```