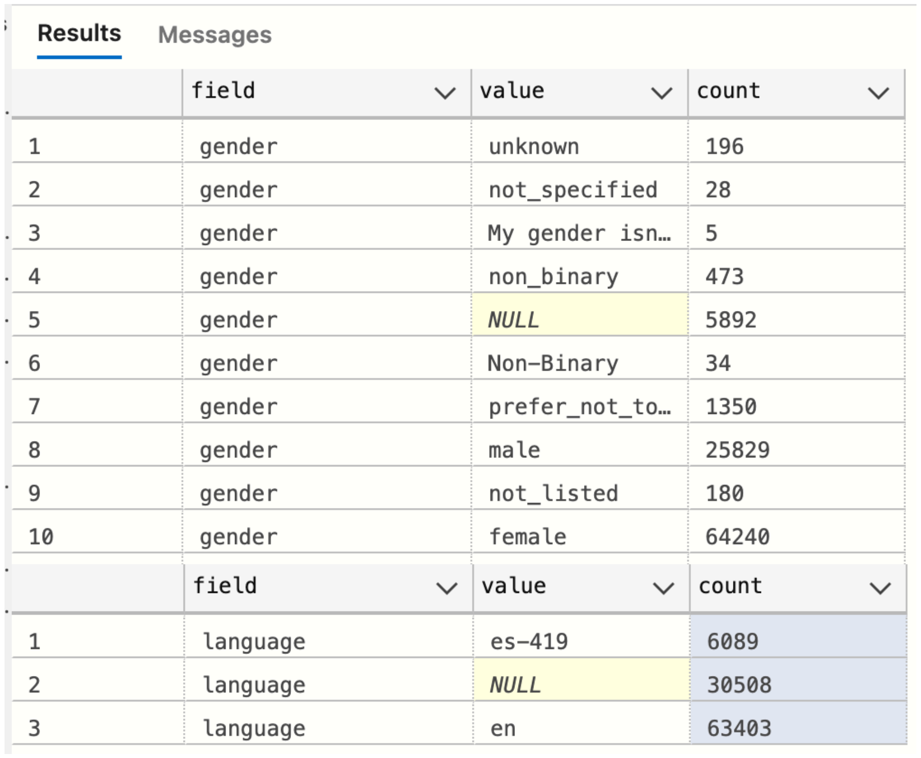
```sql
/*

-- First: explore the data

-- Are there any data quality issues present?
-- Are there any fields that are challenging to understand?


To explore the data, I will first dump all three entire CSV files into the staging table.

Note: I have created two tables for each CSV file, one is staging where raw data is
imported; and the second table is the main table where I am exploring and dumping
according to the requirement and ERM.

Let me start with the staging table

*/

select Count (*) from [dbo].[user_staging]; -- 100000
select Count (*) from [dbo].[transaction_staging]; -- 50000
select Count (*) from [dbo].[products_staging]; -- 845552

-- printing top 10 records to have a quick look

select top 10 * from [dbo].[user_staging]; -- first look says not specific values for
language and gender
select top 10 * from [dbo].[transaction_staging]; -- according to the first look -
final_quntity is having some text value
select top 10 * from [dbo].[products_staging]; -- first look ok though there are some
brads value NULL and need to be taken care


-- I will start by looking into the USER table - in detail
```

```sql
---------------------------------- USER Table ------------------------------------------

delete from [dbo].[user]; -- deleting the main table
-- select * from [dbo].[user];

-- checking unique user ID of staging table
select distinct ID from [dbo].[user_staging]; -- 100000 exact matching with the count of
records

-- Inserting into main table for analysis and then setting up Primary key
insert into [dbo].[user] (id, created_date, birth_date, [state], [language], [gender])
select * -- Inserting all records as both tables having same data types
from [dbo].[user_staging]
where id is not null; -- ensure primary key constraint -- (100000 rows affected)

-- Add primary key to user table (only if it doesn't exist)
if not exists (
    select 1 from information_schema.table_constraints
    where constraint_type = 'primary key' and table_name = '[dbo].[user]'
)
begin
    alter table [dbo].[user] add constraint pk_user primary key ([id]);
end

select distinct ID from [dbo].[user]; -- 100000 unique records inserted

-- Same number of records inserted
-- For User table, the only thing I would look into is the user_id should be the primary
key and not Null
```

```sql
-- query to check distinct values in gender and language
select 'gender' as field, gender as value, count(*) as count
from [dbo].[user]
group by gender; -- not adding any meaningful value to the analysis right now so, not
updating with specific ones

select 'language' as field, language as value, count(*) as count
from [dbo].[user]
group by language; -- not adding any meaningful value to the analysis right now so, not
updating with specific ones
```



Results Messages

	field	value	count
1	gender	unknown	196
2	gender	not_specified	28
3	gender	My gender isn…	5
4	gender	non_binary	473
5	gender	*NULL*	5892
6	gender	Non-Binary	34
7	gender	prefer_not_to…	1350
8	gender	male	25829
9	gender	not_listed	180
10	gender	female	64240

	field	value	count
1	language	es-419	6089
2	language	*NULL*	30508
3	language	en	63403

```sql
-- Now explore the products

---------------------------------- PRODUCT Table  --------------------------------------

delete from [dbo].[products];
-- select * from [dbo].[products];

-- Delete rows from the products table where the barcode is null, which should be the
primary key as per the ERM.
-- Removing it from the staging directly for a clean insert

delete
-- select *
from [dbo].[products_staging]
where barcode is null; -- (4025 rows affected)

-- Now again duplicates removal of a barcode as it's going to be the primary key



with cte as (
    select barcode,
           row_number() over (partition by barcode order by (select null)) as rownum
    from [dbo].[products_staging]
)
delete
-- select *
from cte where rownum > 1; -- 185
```

```sql
-- Clean and insert into the final products table
insert into [dbo].[products] (category_1, category_2, category_3, category_4,
manufacturer, brand, barcode)
select
    nullif(category_1, ''),
    nullif(category_2, ''),
    nullif(category_3, ''),
    nullif(category_4, ''),
    nullif(manufacturer, ''),
    nullif(brand, ''),
    try_cast(barcode as bigint) -- ensure the barcode is numeric and not null will be
inserted
from [dbo].[products_staging]
where barcode is not null; --- (841342 rows affected)


-- Add primary key to products table (only if it doesn’t exist)
if not exists (
    select 1 from information_schema.table_constraints
    where constraint_type = 'primary key' and table_name = 'products'
)
begin
    alter table [dbo].[products] add constraint pk_products primary key ([barcode]);
end

select count(*) from [dbo].[products_staging]; -- 845552
select count(*) from [dbo].[products];  -- 841342

/* Just checking whether the correct number of records was inserted or not
--- 4025 where a barcode is null
--- 185 where barcodes were duplicate
select (845552 - 841342) -- 4210
select (4025 + 185) -- 4210 */
```

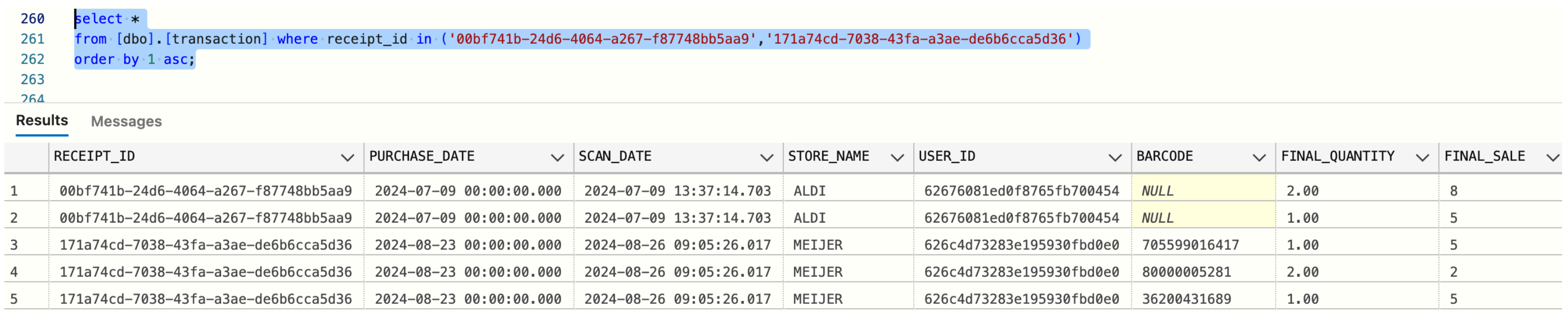
```
-- The next one is the transaction

---------------------------------- TRANSACTION Table ------------------------------------

-- delete from [dbo].[transaction];
select count(*) from [dbo].[transaction]; -- 0

-- there were some records where quantity was in the text - updated as '0' to insert as
numeric
-- updated directly in staging for fresh insert



update a
set a.[final_quantity] = 0
--select *
from [dbo].[transaction_staging] as a
where a.[final_quantity] = 'zero'; -- 12500


-- Check how many records of User_id from Transaction table present in the main User
table
select count(distinct t.user_id) as user_count
from [dbo].[transaction_staging] t
join [dbo].[user] u
on t.user_id = u.id; -- only 91

-- Similiar way check how many records of Barcode present in the main Products table
select count(distinct t.barcode) as barcode_count
from [dbo].[transaction_staging] t
join [dbo].[products] p
on t.barcode = p.barcode; -- only 6562 out of 50000
```

```sql
-- Check unique receipt_id
select count(distinct(receipt_id))
from [dbo].[transaction_staging]; -- 244440

/*

Thought process: In an ideal scenario the User_Id and Barcode should be created as a
foreign key and if the records do not exist in the Transaction table then it should be
either Null or set as any default value after inserting default values in the User and
Products table. data should be normalized.


But - I am assuming and it can be possible in this case that,
            - For the analysis exercise, I am working with CSV files and mainly focusing
on exploring and querying the data.
            - Setting up foreign keys can impact the numerical results of future queries.

But again - In a production-relational database setting it would be advisable to set up
foreign key constraints and it
                                would help maintain data integrity and ensure that every
transaction references valid-user
                                and product records.

*/

-- Insert into the final transactions table without setting up foreign keys as per the
above thought process

insert into [dbo].[transaction] (receipt_id, purchase_date, scan_date, store_name,
user_id, barcode, final_quantity, final_sale)
select *
from [dbo].[transaction_staging]
--(50000 rows affected)
```

```sql
/*

Some records in the Transaction table have missing or zero quantities, and some have null
final sales.
These records don’t add much value to the analysis.

After looking into Fetch Rewards’ business model, it seems like the focus should be on
receipt_id rather than the total dollar amount of transactions.

Also, after analyzing the data, there appear to be quite a few duplicate records among
them.

*/

delete
-- select *
from [dbo].[transaction]
where
    [final_quantity] is null
    or [final_sale] is null
    or ((ltrim(rtrim([final_quantity])) = '0' and  (ltrim(rtrim([final_sale])) = '0')));
-- (12791 rows affected)

delete
-- select  *
from [dbo].[transaction]
where FINAL_QUANTITY = '0' -- (12209 rows affected)

-- Validating unique receipt_id counts again
select count(distinct(receipt_id))
from [dbo].[transaction]; -- 24440 -- same as staging table
```

```
-- Additionally, some duplicates of entire rows present in the transaction data should be
cleaned up as well

with cte as (
    select *,
           row_number() over (
               partition by
                   ltrim(rtrim(receipt_id)),
                   purchase_date,
                   scan_date,
                   ltrim(rtrim(lower(store_name))),
                   ltrim(rtrim(lower(user_id))),
                   ltrim(rtrim(barcode)),
                   final_quantity,
                   final_sale
               order by (select null)
           ) as rownum
    from [dbo].[transaction]
)
-- select * from cte where rownum > 1
delete from cte where rownum > 1; -- (161 rows affected)


-- Validating unique receipt_id counts again
select count(distinct(receipt_id))
from [dbo].[transaction]; -- 24440 -- same as staging table

select count(*)
from [dbo].[transaction]; -- 24839
```

```sql
select (24839 - 24440)
-- 399 is the difference between the total number of records from the transaction table
and the unique receipt_Id after clean up

/*

Those records are where one receipt_id can have multiple barcodes, including null ones,
these records might still be relevant
and shouldn't be removed outright.

*/

select *
from [dbo].[transaction] where receipt_id in ('00bf741b-24d6-4064-a267-
f87748bb5aa9','171a74cd-7038-43fa-a3ae-de6b6cca5d36')
order by 1 asc;
```



```sql
select *
from [dbo].[transaction] where receipt_id in ('00bf741b-24d6-4064-a267-f87748bb5aa9','171a74cd-7038-43fa-a3ae-de6b6cca5d36')
order by 1 asc;
```

Results Messages

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
1	00bf741b-24d6-4064-a267-f87748bb5aa9	2024-07-09 00:00:00.000	2024-07-09 13:37:14.703	ALDI	62676081ed0f8765fb700454	*NULL*	2.00	8
2	00bf741b-24d6-4064-a267-f87748bb5aa9	2024-07-09 00:00:00.000	2024-07-09 13:37:14.703	ALDI	62676081ed0f8765fb700454	*NULL*	1.00	5
3	171a74cd-7038-43fa-a3ae-de6b6cca5d36	2024-08-23 00:00:00.000	2024-08-26 09:05:26.017	MEIJER	626c4d73283e195930fbd0e0	705599016417	1.00	5
4	171a74cd-7038-43fa-a3ae-de6b6cca5d36	2024-08-23 00:00:00.000	2024-08-26 09:05:26.017	MEIJER	626c4d73283e195930fbd0e0	80000005281	2.00	2
5	171a74cd-7038-43fa-a3ae-de6b6cca5d36	2024-08-23 00:00:00.000	2024-08-26 09:05:26.017	MEIJER	626c4d73283e195930fbd0e0	36200431689	1.00	5

```
/*

Here are some key points of confusion:

        * Out of 50,000 records, only 6,562 transactions have barcodes that match the
Products table,
            and just 91 users exist in the Users table. If a foreign key constraint is
applied following best practices,
            another 18,186 records would be removed.
        * There are many duplicate records in the Transaction table.
        * While exploring the Users table, I noticed that the 'gender' and 'language'
fields don’t have specific or standardized values.
            However, this doesn’t impact our analysis.

*/
```