Are there any data quality issues present?

Yes, multiple data quality issues were identified across the datasets:

1. Null Values Analysis

Missing values were identified in the following columns:

Users Table

Column	Missing Count	% Missing
BIRTH_DATE	3,675	3.7%
STATE	4,812	4.8%
LANGUAGE	30,508	30.5%
GENDER	5,892	5.9%

Transactions Table

Column	Missing Count	% Missing
BARCODE	5,762	11.5%

Products Table

Column	Missing Count	% Missing
CATEGORY_1	111	0.01%
CATEGORY_2	1,424	0.2%
CATEGORY_3	60,566	7.2%
CATEGORY_4	778,093	92.0%
MANUFACTURER	226,474	26.8%
BRAND	226,472	26.8%
BARCODE	4,025	0.5%

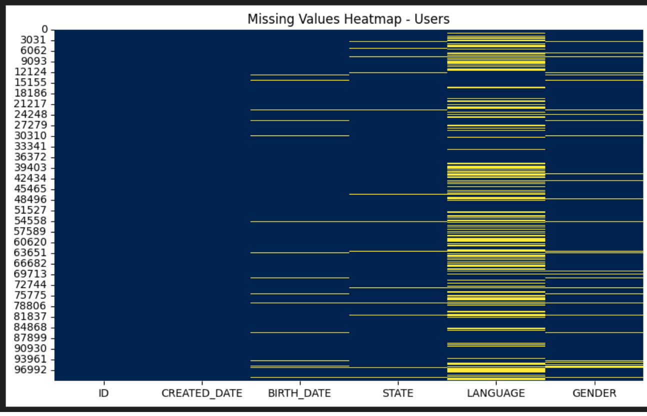
*Scripts and visualizations below show analysis via python pandas, seaborn and pyplot (script available in the attached notebook)

```
#check for missing values per column
print('-----USERS-----')
print(df_users.isnull().sum())
print('-----TRANSACTIONS-----')
print(df_transactions.isnull().sum())
print('-----PRODUCTS-----')
print(df_products.isnull().sum())
```
[28]

```
-----USERS-----
ID                 0
CREATED_DATE       0
BIRTH_DATE      3675
STATE           4812
LANGUAGE       30508
GENDER          5892
dtype: int64
-----TRANSACTIONS-----
RECEIPT_ID         0
PURCHASE_DATE      0
SCAN_DATE          0
STORE_NAME         0
USER_ID            0
BARCODE         5762
FINAL_QUANTITY     0
FINAL_SALE         0
dtype: int64
-----PRODUCTS-----
CATEGORY_1       111
CATEGORY_2      1424
CATEGORY_3     60566
CATEGORY_4    778093
MANUFACTURER  226474
BRAND         226472
BARCODE         4025
dtype: int64
```

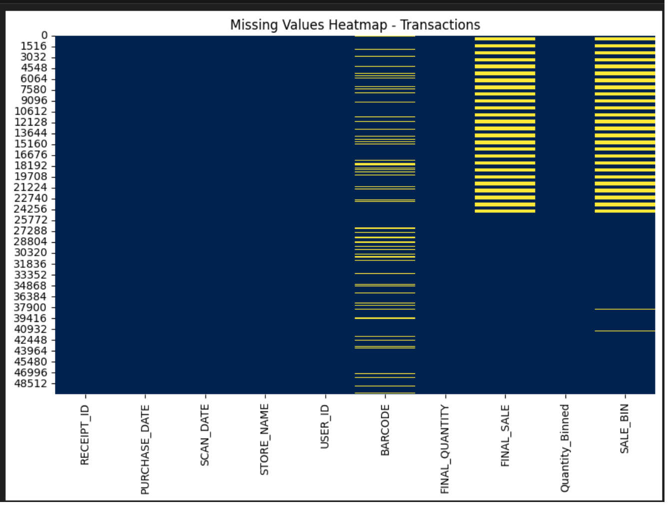
```python
#Heatmap to show how much missing values we have compared to those that have actual values - USERS
plt.figure(figsize=(10, 6))
sns.heatmap(df_users.isnull(), cmap="cividis", cbar=False)
plt.title("Missing Values Heatmap - Users")
plt.show()
```

[42] ✓ 1.1s



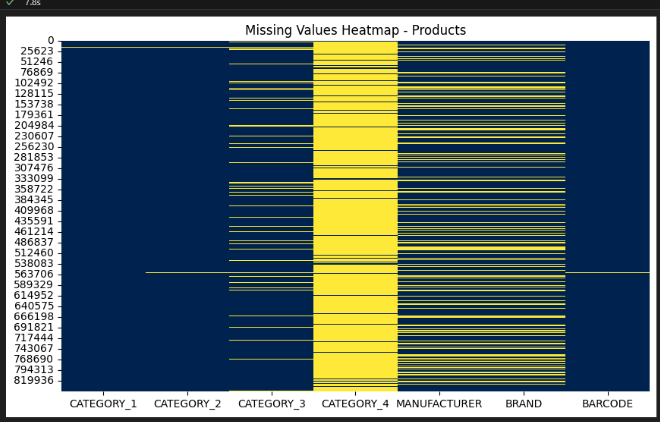

```python
#Heatmap to show how much missing values we have compared to those that have actual values - TRANSACTIONS
plt.figure(figsize=(10, 6))
sns.heatmap(df_transactions.isnull(), cmap="cividis", cbar=False)
plt.title("Missing Values Heatmap - Transactions")
plt.show()
```

[40] ✓ 1.2s




```python
#Heatmap to show how much missing values we have compared to those that have actual values - PRODUCTS
plt.figure(figsize=(10, 6))
sns.heatmap(df_products.isnull(), cmap="cividis", cbar=False)
plt.title("Missing Values Heatmap - Products")
plt.show()
```

[41] ✓ 7.8s



*Query below shows analysis via SQL Server (script available in the attached sql file)

```sql
--- check for the null value...e
SELECT 'USERS' AS TableName, 'BIRTH_DATE' AS ColumnName, COUNT(*) AS MissingCount
FROM USER_TAKEHOME WHERE BIRTH_DATE IS NULL
UNION ALL
SELECT 'USERS', 'STATE', COUNT(*) FROM USER_TAKEHOME WHERE STATE IS NULL
UNION ALL
SELECT 'USERS', 'LANGUAGE', COUNT(*) FROM USER_TAKEHOME WHERE LANGUAGE IS NULL
UNION ALL
SELECT 'USERS', 'GENDER', COUNT(*) FROM USER_TAKEHOME WHERE GENDER IS NULL;

-- TRANSACTIONS Table
SELECT 'TRANSACTIONS' AS TableName, 'BARCODE' AS ColumnName, COUNT(*) AS MissingCount
FROM TRANSACTION_TAKEHOME WHERE BARCODE IS NULL;

-- PRODUCTS Table
SELECT 'PRODUCTS' AS TableName, 'CATEGORY_1' AS ColumnName, COUNT(*) AS MissingCount
FROM PRODUCTS_TAKEHOME WHERE CATEGORY_1 IS NULL
UNION ALL
SELECT 'PRODUCTS', 'CATEGORY_2', COUNT(*) FROM PRODUCTS_TAKEHOME WHERE CATEGORY_2 IS NULL
UNION ALL
SELECT 'PRODUCTS', 'CATEGORY_3', COUNT(*) FROM PRODUCTS_TAKEHOME WHERE CATEGORY_3 IS NULL
UNION ALL
SELECT 'PRODUCTS', 'CATEGORY_4', COUNT(*) FROM PRODUCTS_TAKEHOME WHERE CATEGORY_4 IS NULL
UNION ALL
SELECT 'PRODUCTS', 'MANUFACTURER', COUNT(*) FROM PRODUCTS_TAKEHOME WHERE MANUFACTURER IS NULL
UNION ALL
SELECT 'PRODUCTS', 'BRAND', COUNT(*) FROM PRODUCTS_TAKEHOME WHERE BRAND IS NULL
UNION ALL
SELECT 'PRODUCTS', 'BARCODE', COUNT(*) FROM PRODUCTS_TAKEHOME WHERE BARCODE IS NULL;
```

100 %

Results | Messages

	TableName	ColumnName	MissingCount
1	USERS	BIRTH_DATE	3675
2	USERS	STATE	4812
3	USERS	LANGUAGE	30508
4	USERS	GENDER	5892

	TableName	ColumnName	MissingCount
1	TRANSACTIONS	BARCODE	5762

	TableName	ColumnName	MissingCount
1	PRODUCTS	CATEGORY_1	111
2	PRODUCTS	CATEGORY_2	1424
3	PRODUCTS	CATEGORY_3	60566
4	PRODUCTS	CATEGORY_4	778093
5	PRODUCTS	MANUFACT...	226474
6	PRODUCTS	BRAND	226472
7	PRODUCTS	BARCODE	4025

Query executed successfully.

2. Empty FINAL_SALE Values in Transactions

- **12,500 transactions** have empty (one space) FINAL_SALE values.

*Script below shows analysis via python pandas (script available in the attached notebook)

```python
#Check for any non-NaN string but empty or blank
# Convert BARCODE to string
df_transactions['BARCODE'] = df_transactions['BARCODE'].astype(str)
df_products['BARCODE'] = df_products['BARCODE'].astype(str)

# Check for blank spaces or special characters in the datasets
def check_blank_spaces(df, columns):
    for col in columns:
        # Check for blank spaces or empty strings (strip and check for '')
        blank_check = df[col].str.strip() == ''
        print(f"Blank space check for {col}:\n", blank_check.sum(), "blank spaces found.\n")


# Apply function to each dataset
user_columns = ['BIRTH_DATE', 'STATE', 'LANGUAGE', 'GENDER']
transaction_columns = ['STORE_NAME', 'USER_ID', 'BARCODE', 'FINAL_QUANTITY', 'FINAL_SALE']
product_columns = ['CATEGORY_1', 'CATEGORY_2', 'CATEGORY_3', 'CATEGORY_4', 'MANUFACTURER', 'BRAND', 'BARCODE']

check_blank_spaces(df_users, user_columns)
check_blank_spaces(df_transactions, transaction_columns)
check_blank_spaces(df_products, product_columns)
```

✓ 3.1s

```
Blank space check for BARCODE:
 0 blank spaces found.

Blank space check for FINAL_QUANTITY:
 0 blank spaces found.

Blank space check for FINAL_SALE:
 12500 blank spaces found.

Blank space check for CATEGORY_1:
 0 blank spaces found.
```

3. Non-ASCII Characters in Text Columns

Non-ASCII characters were found in the following fields:

Column	Count of Non-ASCII Characters	Example Values
STORE_NAME (Transactions)	18	FRESCO Y MÁS
CATEGORY_2 (Products)	5	À La Carte Item
CATEGORY_3 (Products)	44	Rosé
CATEGORY_4 (Products)	33	Rosé & Blends
MANUFACTURER (Products)	13,981	MONDELĒZ INTERNATIONAL
BRAND (Products)	9,256	NATURE MADE®

*Script below shows analysis via python pandas (script available in the attached notebook)

```
# Check for non-ASCII characters in the datasets
def check_non_ascii(df, columns):
    # Pattern to detect non-ASCII characters (anything outside the ASCII range)
    pattern = '[^\x00-\x7F]'  # Match characters that are outside the ASCII range (0x00 to 0x7F)

    for col in columns:
        if df[col].dtype == 'object':  # Only apply to string columns
            non_ascii_check = df[col].str.contains(pattern, na=False)  # Ignore NaNs
            print(f"Non-ASCII character check for {col}:\n", non_ascii_check.sum(), "non-ASCII characters found.\n")

# Apply function to each dataset
check_non_ascii(df_users, user_columns)
check_non_ascii(df_transactions, transaction_columns)
check_non_ascii(df_products, product_columns)
```

[94] ✓ 5.5s

```
Non-ASCII character check for STATE:
 0 non-ASCII characters found.

Non-ASCII character check for LANGUAGE:
 0 non-ASCII characters found.

Non-ASCII character check for GENDER:
 0 non-ASCII characters found.

Non-ASCII character check for STORE_NAME:
 18 non-ASCII characters found.

Non-ASCII character check for CATEGORY_1:
 0 non-ASCII characters found.

Non-ASCII character check for CATEGORY_2:
 5 non-ASCII characters found.

Non-ASCII character check for CATEGORY_3:
 44 non-ASCII characters found.

Non-ASCII character check for CATEGORY_4:
 33 non-ASCII characters found.
```

4. Duplicate Records

Duplicate records were found based on assumed primary keys, excluding null values:

Table	Column Checked	Duplicate Count
Transactions	RECEIPT_ID	25,560
Products	BARCODE	185

**For duplicate RECEIPT_ID, transactions differ in FINAL_QUANTITY or FINAL_SALE. Script below shows analysis via python pandas (script available in the attached notebook)*

```python
# TRANSACTION_TAKEHOME - Checking duplicates on 'RECEIPT_ID', ignoring NaNs
df_transactions.dropna(subset=['RECEIPT_ID']).loc[df_transactions.duplicated(subset=['RECEIPT_ID'], keep=False)].sort_values(by=['RECEIPT_ID'])
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
0	0000d256-4041-4a3e-adc4-5623fb6e0c99	2024-08-21	2024-08-21 14:19:06.539000+00:00	WALMART	63b73a7f3d310dceeabd4758	1.530001e+10	1.00	
41567	0000d256-4041-4a3e-adc4-5623fb6e0c99	2024-08-21	2024-08-21 14:19:06.539000+00:00	WALMART	63b73a7f3d310dceeabd4758	1.530001e+10	1.00	1.54
1	0001455d-7a92-4a7b-a1d2-c747af1c8fd3	2024-07-20	2024-07-20 09:50:24.206000+00:00	ALDI	62c08877baa38d1a1f6c211a	NaN	zero	1.49
39291	0001455d-7a92-4a7b-a1d2-c747af1c8fd3	2024-07-20	2024-07-20 09:50:24.206000+00:00	ALDI	62c08877baa38d1a1f6c211a	NaN	1.00	1.49
2	00017e0a-7851-42fb-bfab-0baa96e23586	2024-08-18	2024-08-19 15:38:56.813000+00:00	WALMART	60842f207ac8b7729e472020	7.874223e+10	1.00	
...	...	...	...	...	...	...	...	...
28152	fffbb112-3cc5-47c2-b014-08db2f87e0c7	2024-07-30	2024-08-04 11:43:31.474000+00:00	WALMART	5eb59d6be7012d13941af5e2	8.180000e+11	1.00	4.88
24998	fffbfb2a-7c1f-41c9-a5da-628fa7fcc746	2024-07-28	2024-07-28 11:47:34.180000+00:00	WALMART	62a0c8f7d966665570351bb8	1.300001e+10	1.00	
31602	fffbfb2a-7c1f-41c9-a5da-628fa7fcc746	2024-07-28	2024-07-28 11:47:34.180000+00:00	WALMART	62a0c8f7d966665570351bb8	1.300001e+10	1.00	3.48
25233	fffe8012-7dcf-4d84-b6c6-feaacab5074a	2024-09-07	2024-09-08 08:21:25.648000+00:00	WALGREENS	5f53c62bd683c715b9991b20	7.432310e+10	2.00	2.98
24999	fffe8012-7dcf-4d84-b6c6-feaacab5074a	2024-09-07	2024-09-08 08:21:25.648000+00:00	WALGREENS	5f53c62bd683c715b9991b20	7.432310e+10	zero	2.98

50000 rows × 8 columns

**For duplicate BARCODE, 48 records used the same barcode twice but have different category, manufacturer or brand.* Query below shows analysis via SQL Server (script available in the attached sql file)

```sql
SELECT PT.* FROM PRODUCTS_TAKEHOME PT
INNER JOIN PRODUCTS_TAKEHOME PTH ON PT.BARCODE=PTH.BARCODE
AND (
    PT.BRAND <> PTH.BRAND
    OR PT.MANUFACTURER <> PTH.MANUFACTURER
    OR PT.CATEGORY_1 <> PTH.CATEGORY_1
    OR PT.CATEGORY_2 <> PTH.CATEGORY_2
    OR PT.CATEGORY_3 <> PTH.CATEGORY_3
    OR PT.CATEGORY_4 <> PTH.CATEGORY_4
    )
ORDER BY BARCODE
```

100 %

Results | Messages

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE
1	Snacks	Candy	Chocolate Candy	NULL	MARS WRIGLEY	M&M'S	404310
2	Snacks	Candy	Chocolate Candy	NULL	PLACEHOLDER MANUFACTURER	BRAND NOT KNOWN	404310
3	Snacks	Chips	Crisps	NULL	TRADER JOE'S	TRADER JOE'S	701983
4	Snacks	Nuts & Seeds	Snack Seeds	NULL	SUNRIDGE FARMS	SUNRIDGE FARMS	701983
5	Snacks	Nuts & Seeds	Almonds	NULL	TRADER JOE'S	TRADER JOE'S	969307
6	Snacks	Nuts & Seeds	Covered Nuts	NULL	TRADER JOE'S	TRADER JOE'S	969307
7	Health & Wellness	Skin Care	Lip Balms & Treatments	Medicated Lip Treatments	E.T. BROWNE DRUG CO., INC.	PALMER'S SKIN & HAIR CARE	1018158
8	Health & Wellness	Skin Care	Facial Lotion & Moisturizer	NULL	R.M. PALMER COMPANY, LLC	PALMER	1018158
9	Snacks	Candy	Chocolate Candy	NULL	THE HERSHEY COMPANY	HERSHEY'S	3422007
10	Snacks	Candy	Candy Variety Pack	NULL	THE HERSHEY COMPANY	HERSHEY'S	3422007
11	Snacks	Candy	Chocolate Candy	NULL	THE HERSHEY COMPANY	REESE'S	3431207
12	Snacks	Candy	Confection Candy	NULL	THE HERSHEY COMPANY	REESE'S	3431207

Query executed successfully. | LAPTOP-A5L9BI8Q\SQLEXPRESS ... | LAPTOP-A5L9BI8Q\Jerome... | Fetch_TakeHome | 00:02:41 | 48 rows

5. Date Format Inconsistencies

- Date fields are in **inconsistent formats**.
- Affected columns: **PURCHASE_DATE, SCAN_DATE**.

*Script below shows analysis via python pandas (script available in the attached notebook)

```python
#display top 5 transactions
df_transactions.head()
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
0	0000d256-4041-4a3e-adc4-5623fb6e0c99	2024-08-21	2024-08-21 14:19:06.539 Z	WALMART	63b73a7f3d310dceeabd4758	1.530001e+10	1.00	
1	0001455d-7a92-4a7b-a1d2-c747af1c8fd3	2024-07-20	2024-07-20 09:50:24.206 Z	ALDI	62c08877baa38d1a1f6c211a	NaN	zero	1.49
2	00017e0a-7851-42fb-bfab-0baa96e23586	2024-08-18	2024-08-19 15:38:56.813 Z	WALMART	60842f207ac8b7729e472020	7.874223e+10	1.00	
3	000239aa-3478-453d-801e-66a82e39c8af	2024-06-18	2024-06-19 11:03:37.468 Z	FOOD LION	63fcd7cea4f8442c3386b589	7.833997e+11	zero	3.49
4	00026b4c-dfe8-49dd-b026-4c2f0fd5c6a1	2024-07-04	2024-07-05 15:56:43.549 Z	RANDALLS	6193231ae9b3d75037b0f928	4.790050e+10	1.00	

6. Logical Inconsistencies in Date Values

- **224 Transactions where SCAN_DATE occurs before PURCHASE_DATE** were found.

*Script below shows analysis via python pandas (script available in the attached notebook)

```python
# Convert to datetime and check for errors
df_transactions['PURCHASE_DATE'] = pd.to_datetime(df_transactions['PURCHASE_DATE']).dt.tz_localize(None)
df_transactions['SCAN_DATE'] = pd.to_datetime(df_transactions['SCAN_DATE']).dt.tz_localize(None)

# Check for invalid dates
print("Invalid PURCHASE_DATE:", df_transactions['PURCHASE_DATE'].isna().sum())
print("Invalid SCAN_DATE:", df_transactions['SCAN_DATE'].isna().sum())

# Check for transactions where SCAN_DATE is before PURCHASE_DATE
invalid_dates = df_transactions[df_transactions['SCAN_DATE'] < df_transactions['PURCHASE_DATE']]
print("Transactions with SCAN_DATE before PURCHASE_DATE:\n", invalid_dates)
```

[7] ✓ 0.2s

```
Invalid PURCHASE_DATE: 0
Invalid SCAN_DATE: 0
Transactions with SCAN_DATE before PURCHASE_DATE:
                                 RECEIPT_ID PURCHASE_DATE  \
51     008c1dcc-0f96-4b04-98c8-2a2bb63ef89d    2024-07-21
455    04a320ed-2903-45e5-8fd7-6eaf08daef32    2024-06-29
494    05023b3d-5f83-47a7-a17c-8e8521d0bc94    2024-09-08
675    06ce3da3-a588-4c37-93b4-0b6d11e42704    2024-06-22
870    08d0e78f-3e63-40a3-8eb0-73fdf76da52c    2024-06-22
...                                     ...           ...
46034  08d0e78f-3e63-40a3-8eb0-73fdf76da52c    2024-06-22
46539  718aa730-b62f-4e18-8dba-1d7105dac341    2024-09-05
46941  af2b818f-4a92-4e98-958c-65f2ce0b271d    2024-06-15
47653  72bb7b71-d958-4a46-ae62-43abdeb0e693    2024-06-15
47837  99c2e8dc-9dc7-4267-9342-0b19c3fb35a0    2024-06-15

                    SCAN_DATE            STORE_NAME                   USER_ID  \
51    2024-07-20 19:54:23.133               WALMART  5dc24cdb682fcf1229d04bd6
455   2024-06-28 11:03:31.783  DOLLAR GENERAL STORE  62855f67708670299a658035
494   2024-09-07 22:22:29.903             SHOP RITE  666a43c77c0469953bfd9ae0
675   2024-06-21 12:34:15.665              BIG LOTS  646f6ffb7a342372c858487e
870   2024-06-21 20:50:01.298  DOLLAR GENERAL STORE  664cafb6e04f743a096a837e
...                       ...                   ...                       ...
46034 2024-06-21 20:50:01.298  DOLLAR GENERAL STORE  664cafb6e04f743a096a837e
46539 2024-09-04 20:14:00.374               WALMART  5e0f561efa890112094202ad
46941 2024-06-14 10:57:23.892  DOLLAR GENERAL STORE  64de6465516348066e7c5690
47653 2024-06-14 19:55:56.672               WALMART  649726ea127ddb5d7f0004dc
47837 2024-06-14 22:07:18.702               WALMART  5e48ddd01a900e141874e241
```

*Query below shows analysis via SQL Server (script available in the attached sql file)

```sql
SELECT COUNT(*) FROM TRANSACTION_TAKEHOME WHERE
SCAN_DATE < PURCHASE_DATE
```

100 %

Results | Messages

	(No column name)
1	224

7. Foreign Key Integrity Checks

User IDs in Transactions Not Found in Users Table

- **49,738 transactions** have USER_IDs that **do not exist in USERS table**.

*Script below shows analysis via python pandas (script available in the attached notebook)

```python
# Check if all USER_IDs in transactions exist in users
missing_users = df_transactions[~df_transactions['USER_ID'].isin(df_users['ID'])]
missing_users
```

11] ✓ 0.3s

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
0	0000d256-4041-4a3e-adc4-5623fb6e0c99	2024-08-21	2024-08-21 14:19:06.539	WALMART	63b73a7f3d310dceeabd4758	1.530001e+10	1.00	
1	0001455d-7a92-4a7b-a1d2-c747af1c8fd3	2024-07-20	2024-07-20 09:50:24.206	ALDI	62c08877baa38d1a1f6c211a	NaN	zero	1.49
2	00017e0a-7851-42fb-bfab-0baa96e23586	2024-08-18	2024-08-19 15:38:56.813	WALMART	60842f207ac8b7729e472020	7.874223e+10	1.00	
3	000239aa-3478-453d-801e-66a82e39c8af	2024-06-18	2024-06-19 11:03:37.468	FOOD LION	63fcd7cea4f8442c3386b589	7.833997e+11	zero	3.49
4	00026b4c-dfe8-49dd-b026-4c2f0fd5c6a1	2024-07-04	2024-07-05 15:56:43.549	RANDALLS	6193231ae9b3d75037b0f928	4.790050e+10	1.00	
...	...	...	...	...	...	...	...	...
49995	b5cd61a9-8033-4913-a5c4-fb3f65e3a321	2024-08-21	2024-08-31 14:13:08.634	TARGET	6154bcf098f885648de2f299	8.523911e+10	2.00	1.18
49996	e1b2f634-c9ad-4152-b662-4b22efc25862	2024-08-11	2024-08-11 18:15:56.736	STOP & SHOP	60aa809f188b926b2244c974	4.610040e+10	1.00	2.00
49997	b07ef8dd-e444-40a2-819b-f74a3e5f1ae7	2024-07-11	2024-07-11 08:03:25.816	WALMART	60bd26e83dc3b13a15c5f4e7	6.466300e+11	1.00	20.96
49998	42475141-bef4-4df2-aa37-72577e2512bb	2024-06-18	2024-06-18 19:57:32.211	MARKET BASKET	6169912fac47744405af62b7	4.180050e+10	1.00	3.00
49999	3a179c4e-46f2-4126-b3d2-3514afc23a3e	2024-08-07	2024-08-07 15:30:07.911	WALMART	64e94d64ca929250373ef6e1	3.076607e+11	1.00	5.48

49738 rows × 8 columns

*Query below shows analysis via SQL Server (script available in the attached sql file)

```sql
SELECT COUNT(*) FROM TRANSACTION_TAKEHOME TT WHERE
NOT EXISTS (select 1 FROM USER_TAKEHOME TTH WHERE TTH.ID=TT.USER_ID)
```

100 %

Results | Messages

	(No column name)
1	49738

Product Barcodes in Transactions Not Found in Products Table

- **19,408 transactions** reference BARCODEs **not found in PRODUCTS table**.

*Script below shows analysis via python pandas (script available in the attached notebook)

```python
# Exclude null values from BARCODE in both dataframes before checking for BARCODES present in transactions that are not in products
missing_products = df_transactions[
    df_transactions['BARCODE'].notna() & ~df_transactions['BARCODE'].isin(df_products['BARCODE'].dropna())
]
missing_products
```

[96] ✓ 9.0s

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
2	00017e0a-7851-42fb-bfab-0baa96e23586	2024-08-18	2024-08-19 15:38:56.813 Z	WALMART	60842f207ac8b7729e472020	78742229751.0	1.00	
3	000239aa-3478-453d-801e-66a82e39c8af	2024-06-18	2024-06-19 11:03:37.468 Z	FOOD LION	63fcd7cea4f8442c3386b589	783399746536.0	zero	3.49
4	00026b4c-dfe8-49dd-b026-4c2f0fd5c6a1	2024-07-04	2024-07-05 15:56:43.549 Z	RANDALLS	6193231ae9b3d75037b0f928	47900501183.0	1.00	
6	000550b2-1480-4c07-950f-ff601f242152	2024-07-06	2024-07-06 19:27:48.586 Z	WALMART	5f850bc9cf9431165f3ac175	49200905548.0	1.00	
8	000e1d35-15e5-46c6-b6b3-33653ed3d27e	2024-08-13	2024-08-13 18:21:07.931 Z	WALMART	61a6d926f998e47aad33db66	52000011227.0	1.00	
...	...	...	...	...	...	...	...	...
49990	441b9ecd-38ed-4960-9780-eb44a464284a	2024-06-26	2024-07-02 09:37:07.656 Z	FRY'S FOOD STORE	6251c788e3d6762c55855c1d	72250021081.0	1.00	2.49
49991	840c30ae-bc0a-40a4-a47d-052ed0af2da2	2024-08-18	2024-08-18 14:44:02.530 Z	COSTCO	65b322787050d0a6206b3479	14074349.0	1.00	11.99
49992	68f74fb3-ccf2-41f3-896a-799eb9a80680	2024-08-13	2024-08-19 11:06:59.023 Z	PEPPERIDGE FARM	64f4aee2b84ba41db3fb246a	14100071198.0	1.00	2.89
49995	b5cd61a9-8033-4913-a5c4-fb3f65e3a321	2024-08-21	2024-08-31 14:13:08.634 Z	TARGET	6154bcf098f885648de2f299	85239110669.0	2.00	1.18
49998	42475141-bef4-4df2-aa37-72577e2512bb	2024-06-18	2024-06-18 19:57:32.211 Z	MARKET BASKET	6169912fac47744405af62b7	41800501519.0	1.00	3.00

19408 rows × 8 columns

*Query below shows analysis via SQL Server (script available in the attached sql file)

```sql
SELECT COUNT(*) FROM TRANSACTION_TAKEHOME TT WHERE
NOT EXISTS (SELECT 1 FROM PRODUCTS_TAKEHOME PTH WHERE PTH.BARCODE=TT.BARCODE)
AND TT.BARCODE IS NOT NULL
```

100 %

Results | Messages

	(No column name)
1	19408

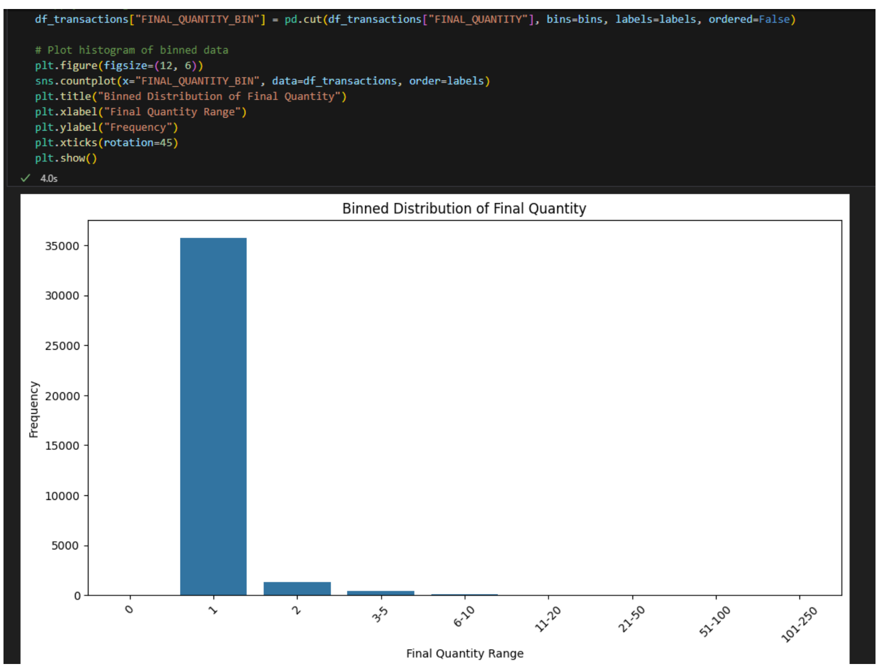
8. Outliers in Numerical Fields

FINAL_QUANTITY

- Outliers detected, including an **extreme value >250**.

**Boxplot below shows the outlier (also present in the committed notebook):*

**Histogram below shows distribution of Final Quantity (also present in the committed notebook):*



Query below shows analysis via SQL Server (script available in the attached sql file)

```sql
-- Check for extreme values in FINAL_QUANTITY (outlier based from boxplot)
SELECT * FROM TRANSACTION_TAKEHOME
WHERE CAST(REPLACE(FINAL_QUANTITY,'zero',0) AS float) > 250;
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
1	fe0780d1-2d02-4822-8f12-7056b1814f17	2024-08-09 00:00:00.0000000	2024-08-11 10:52:18.5230000	MAIN STREET MARKET	5d197f9dd08976510c49d0e6	48001353664	276.00	NULL
2	fe0780d1-2d02-4822-8f12-7056b1814f17	2024-08-09 00:00:00.0000000	2024-08-11 10:52:18.5230000	MAIN STREET MARKET	5d197f9dd08976510c49d0e6	48001353664	276.00	5.89

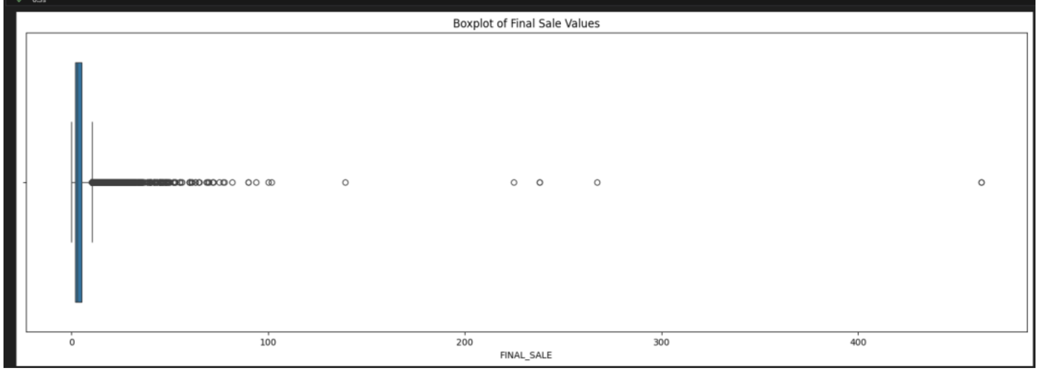
FINAL_SALE

- **Three outliers between 200-300**.
- **One extreme outlier >400**.

**Boxplot below shows the outlier (also present in the committed notebook):*

```python
# Boxplot for FINAL_SALE
plt.figure(figsize=(20, 6))
sns.boxplot(x=df_transactions["FINAL_SALE"])
plt.title("Boxplot of Final Sale Values")
plt.show()
```

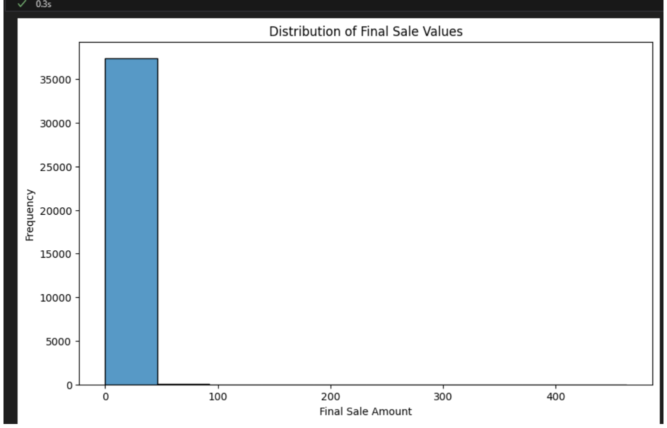
✓ 0.3s



**Histogram below shows distribution of Final Sale (also present in the committed notebook):*

```python
#the histogram below will show the distribution of data better for final sale
plt.figure(figsize=(10, 6))
sns.histplot(df_transactions["FINAL_SALE"], bins=10)
plt.title("Distribution of Final Sale Values")
plt.xlabel("Final Sale Amount")
plt.ylabel("Frequency")
plt.show()
```

✓ 0.3s



**Query below shows analysis via SQL Server (script available in the attached sql file)*

```sql
-- Check for extreme values in FINAL_SALE  (outlier based from boxplot)
SELECT DISTINCT * FROM TRANSACTION_TAKEHOME
WHERE CAST(FINAL_SALE AS float) > 400 OR CAST(FINAL_SALE AS float) BETWEEN 200 AND 300;
```

	RECEIPT_ID	PURCHASE_DATE	SCAN_DATE	STORE_NAME	USER_ID	BARCODE	FINAL_QUANTITY	FINAL_SALE
1	7f894472-dc11-4f4f-b994-8da1145a18b9	2024-07-17 00:00:00.0000000	2024-07-17 11:52:35.7630000	CVS	645add3bffe0d7e043ef1b63	NULL	1.00	224.99
2	8b4a1054-09e6-4701-a7b5-49e3db6b8f61	2024-08-23 00:00:00.0000000	2024-08-26 11:59:20.5650000	RITE AID	650874eafe41d365c2ee11d2	NULL	1.00	267.29
3	8bbb09f6-aae6-47ae-88af-7cd46cc8079d	2024-07-06 00:00:00.0000000	2024-07-06 08:01:20.3050000	CVS	630789e1101ae272a4852287	NULL	1.00	462.82
4	8bbb09f6-aae6-47ae-88af-7cd46cc8079d	2024-07-06 00:00:00.0000000	2024-07-06 08:01:20.3050000	CVS	630789e1101ae272a4852287	NULL	zero	462.82
5	bf4af9c2-ee08-48a1-ac97-1835f3f072f4	2024-08-29 00:00:00.0000000	2024-08-29 06:26:15.4660000	KROGER	63af23db9f3fc9c7546fdbec	NULL	1.00	238.17
6	bf4af9c2-ee08-48a1-ac97-1835f3f072f4	2024-08-29 00:00:00.0000000	2024-08-29 06:26:15.4660000	KROGER	63af23db9f3fc9c7546fdbec	NULL	zero	238.17

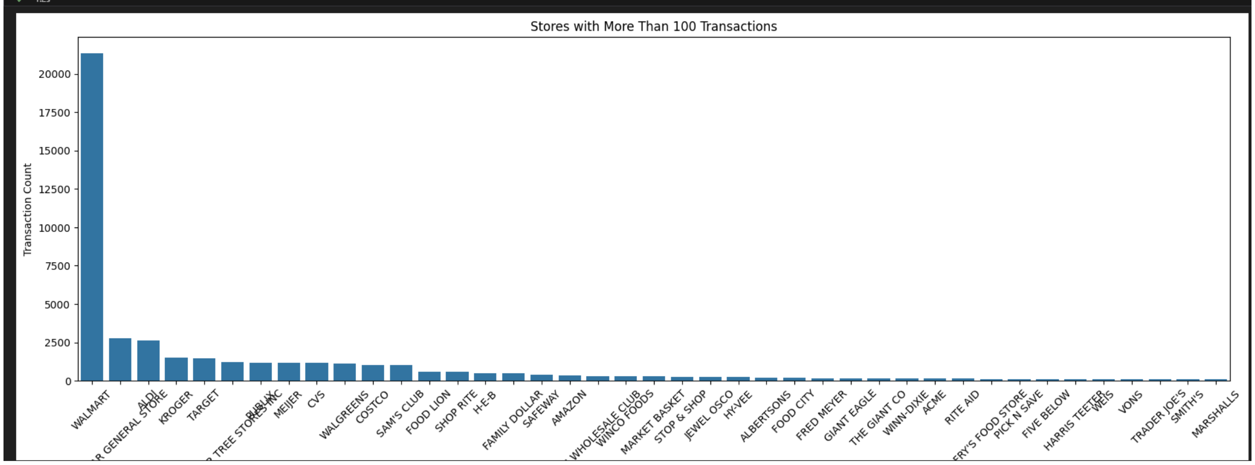
9. Walmart Transaction Gap & Sales Analysis

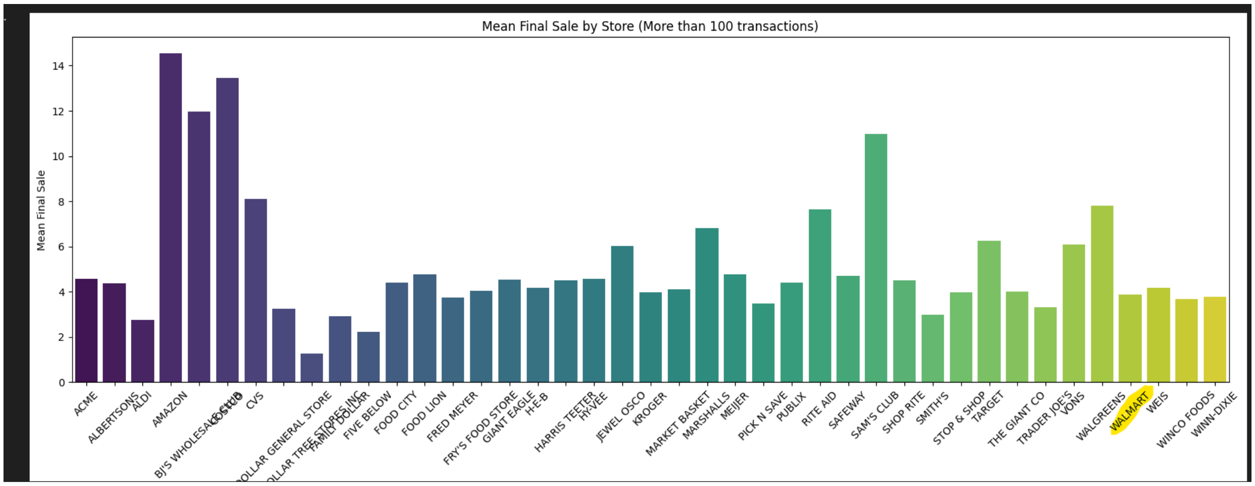
- Walmart has a significantly higher number of transactions than other stores, but its average FINAL_SALE is not particularly high.

*Histograms below shows the transaction count and mean final sales for stores with more than 100 transactions (script available in the attached notebook):

```python
#Barplot to show the distribution of transactions per stores (only for those who have more than 100 transactions)
plt.figure(figsize=(20, 6))
sns.barplot(x=frequent_stores.index, y=frequent_stores.values)
plt.xticks(rotation=45)
plt.title(f"Stores with More Than {min_transactions} Transactions")
plt.xlabel("Store Name")
plt.ylabel("Transaction Count")
plt.show()
```

✓ 1.2s





*One potential cause is that 25% of FINAL_SALE values for WALMART are NULL or empty.

*Query below shows analysis via SQL Server (script available in the attached sql file)

```sql
WITH COUNT_OF_NULL_SALES AS (
SELECT COUNT(*) [NULL_SALES_COUNT], STORE_NAME STORE_NAME
FROM TRANSACTION_TAKEHOME
WHERE FINAL_SALE IS NULL
GROUP BY STORE_NAME
),
TOTAL_COUNT_PER_STORE AS (
SELECT COUNT(*) [TOTAL_SALES_COUNT], STORE_NAME STORE_NAME
FROM TRANSACTION_TAKEHOME
GROUP BY STORE_NAME
)

SELECT CONS.STORE_NAME,(CAST(CONS.NULL_SALES_COUNT AS FLOAT) / TCPS.TOTAL_SALES_COUNT) *100 [PERCENTAGE_OF_NULL_SALES]
FROM COUNT_OF_NULL_SALES CONS
INNER JOIN TOTAL_COUNT_PER_STORE TCPS ON TCPS.STORE_NAME=CONS.STORE_NAME
WHERE CONS.STORE_NAME='WALMART'
ORDER BY 2 DESC
```

100 %

Results | Messages

	STORE_NAME	PERCENTAGE_OF_NULL_SALES
1	WALMART	25.1008159054675

Are there any fields that are challenging to understand?

Yes, the following fields require further clarification:

1. **CATEGORY_1 to CATEGORY_4 (Products Table):**
    - The hierarchical relationship between these categories is unclear. Are they nested? Independent?
2. **FINAL_SALE (Transactions Table):**
    - **12,500 missing values** need explanation. Are they due to refunds, processing issues, or specific stores/products?
3. **SCAN_DATE vs. PURCHASE_DATE:**
    - The distinction between these two fields needs clarification. Is SCAN_DATE the time of checkout, shipment, or something else?
4. **USER_ID in Transactions:**
    - Missing user records suggest either deleted accounts or missing ingestion data. What's the expected behavior?
5. **BARCODE in Products & Transactions:**
    - Some barcodes appear in transactions but not in the products dataset. Are new products not properly registered, or is there a delay in updating the master list?