Data Analysis

Data Quality Issues Identified

1. Products Dataset

- **Null Values**:
  - CATEGORY_1 has 111 missing entries.
  - CATEGORY_2 has 1,424 missing entries.
  - CATEGORY_3 has 60,566 missing entries.
  - CATEGORY_4 is missing a majority of its entries (778,093 out of 845,552).
  - MANUFACTURER and BRAND have over 226,000 missing entries each.
  - BARCODE is missing 4,025 entries.
- **Observations**:
  - The CATEGORY_4 column is almost entirely null, which may make it unusable for analysis.
  - Missing BARCODE values can disrupt linking this dataset to the transactions dataset.

2. Transactions Dataset

- **Null Values**:
  - BARCODE has 5,762 missing entries, which might cause difficulties in connecting product data to transactions.
- **Data Types**:
  - FINAL_QUANTITY and FINAL_SALE are stored as strings (object) instead of numeric types, which may require conversion for accurate computations.

3. Users Dataset

- **Null Values**:
  - BIRTH_DATE has 3,675 missing entries, which could affect age-based analysis.
  - STATE is missing 4,812 entries, which might limit location-based segmentation.
  - LANGUAGE is missing a significant portion (30,508 entries).
  - GENDER is missing 5,892 entries.

Fields That Are Challenging to Understand

1. **Products Dataset**:

- CATEGORY_4: The purpose of this column is unclear, and its high rate of null values makes it difficult to determine its utility.
- BARCODE: There's no explanation of whether the missing barcodes are expected or represent an error.

2. **Transactions Dataset**:
   - FINAL_QUANTITY: The format or scale of this field is unclear (e.g., is it in individual units, cases, or another measure?).
   - FINAL_SALE: It's stored as a string but represents a numeric value.
3. **Users Dataset**:
   - BIRTH_DATE: This might include inconsistent formats or null values, making age calculation tricky.

Data Visualization and Graphs

```
import pandas as pd
products_file = '/mnt/data/PRODUCTS_TAKEHOME.csv'
transactions_file = '/mnt/data/TRANSACTION_TAKEHOME.csv'
users_file = '/mnt/data/USER_TAKEHOME.csv'

products_df = pd.read_csv(products_file)
transactions_df = pd.read_csv(transactions_file)
users_df = pd.read_csv(users_file)

# 1. Analysis of the Products Dataset

products_analysis = {
    "missing_values": products_df.isnull().sum(),
    "unique_values": products_df.nunique(),
    "top_categories": products_df['CATEGORY_1'].value_counts(),
}

# 2. Analysis of the Transactions Dataset

transactions_df['PURCHASE_DATE'] =
pd.to_datetime(transactions_df['PURCHASE_DATE'], errors='coerce',
format='%y/%m/%d')
transactions_df['SCAN_DATE'] = pd.to_datetime(transactions_df['SCAN_DATE'],
errors='coerce')
transactions_analysis = {
    "missing_values": transactions_df.isnull().sum(),
    "unique_values": transactions_df.nunique(),
```

```
    "store_distribution": transactions_df['STORE_NAME'].value_counts(),
    "quantity_summary": transactions_df['FINAL_QUANTITY'].describe(),
    "sale_summary": transactions_df['FINAL_SALE'].describe(),
}
```

 # 3. Analysis of the Users Dataset

```
users_df['BIRTH_DATE'] = pd.to_datetime(users_df['BIRTH_DATE'], errors='coerce')
users_df['CREATED_DATE'] = pd.to_datetime(users_df['CREATED_DATE'],
errors='coerce')
users_analysis = {
    "missing_values": users_df.isnull().sum(),
    "unique_values": users_df.nunique(),
    "state_distribution": users_df['STATE'].value_counts(),
    "language_distribution": users_df['LANGUAGE'].value_counts(),
    "gender_distribution": users_df['GENDER'].value_counts(),
}
```

 # All Category analyses into a single dictionary

```
data_analysis = {
    "Products Analysis": products_analysis,
    "Transactions Analysis": transactions_analysis,
    "Users Analysis": users_analysis,
}
```

Products Dataset

Metric	Value
Missing Values	
CATEGORY_1	111
CATEGORY_2	1,424
CATEGORY_3	60,566
CATEGORY_4	778,093
MANUFACTURER	226,474
BRAND	226,472
BARCODE	4,025
Unique Values	
CATEGORY_1	27
CATEGORY_2	121

CATEGORY_3	344
MANUFACTURER	4,354
BRAND	8,122
BARCODE	378,992
Top CATEGORY_1	
Health & Wellness	512,695
Snacks	324,817
Beverages	3,990

2. Transactions Dataset

Metric	**Value**
Missing Values	
BARCODE	5,762
Other Fields	No missing entries
Unique Values	
RECEIPT_ID	24,440
USER_ID	17,694
STORE_NAME	954
Top Stores	
WALMART	21,326 transactions
DOLLAR GENERAL STORE	2,748 transactions
ALDI	2,640 transactions
Summary Statistics	
FINAL_QUANTITY (most common)	35,698 entries with quantity = 1
FINAL_SALE	Frequent empty values (needs cleaning)

3. Users Dataset

Metric	**Value**
Missing Values	
BIRTH_DATE	3,675
STATE	4,812
LANGUAGE	30,508
GENDER	5,892
Unique Values	

STATE	52
LANGUAGE	2
Top States	
Texas (TX)	9,028 users
Florida (FL)	8,921 users
California (CA)	8,589 users
Gender Distribution	
Female	64,240
Male	25,829
Other Categories	Smaller counts (e.g., non-binary, etc.)

Missing Data :

```
import pandas as pd

products_file = '/mnt/data/PRODUCTS_TAKEHOME.csv'
transactions_file = '/mnt/data/TRANSACTION_TAKEHOME.csv'
users_file = '/mnt/data/USER_TAKEHOME.csv'

products_df = pd.read_csv(products_file)
transactions_df = pd.read_csv(transactions_file)
users_df = pd.read_csv(users_file)

# Calculate missing data percentage for each column in the datasets

missing_data_products = (products_df.isnull().sum() / len(products_df) *
100).round(2).to_frame(name='Missing Percentage')
missing_data_transactions = (transactions_df.isnull().sum() / len(transactions_df) *
100).round(2).to_frame(name='Missing Percentage')
missing_data_users = (users_df.isnull().sum() / len(users_df) *
100).round(2).to_frame(name='Missing Percentage')

missing_data_products.index = [f"Products - {col}" for col in
missing_data_products.index]
missing_data_transactions.index = [f"Transactions - {col}" for col in
missing_data_transactions.index]
missing_data_users.index = [f"Users - {col}" for col in missing_data_users.index]
```

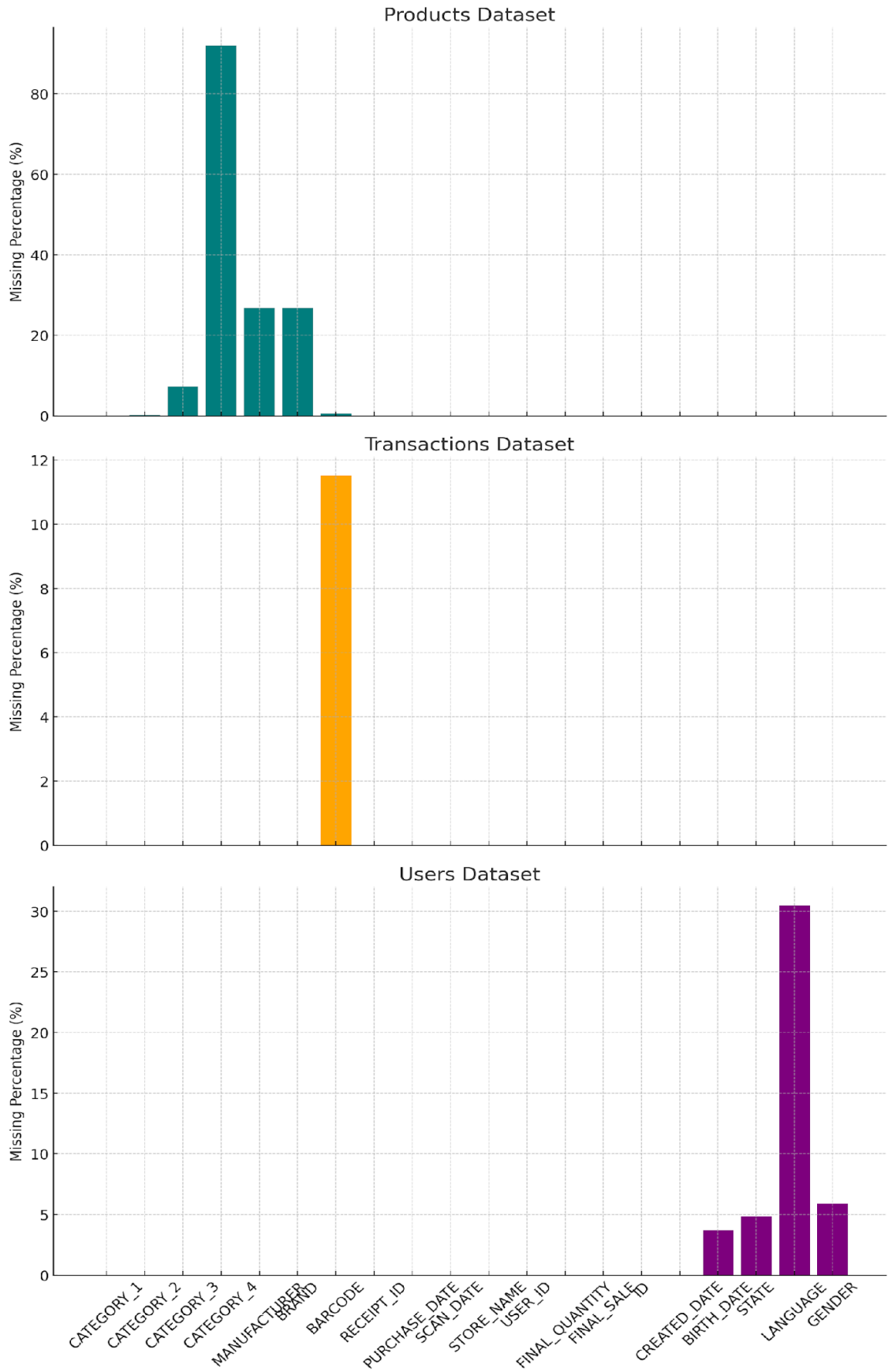
```
# Combine the results for all datasets
missing_data_summary = pd.concat([missing_data_products,
missing_data_transactions, missing_data_users])

missing_data_summary.reset_index().rename(columns={"index": "Column"})
```

Column	Missing Percentage (%)
Products - CATEGORY_1	0.01
Products - CATEGORY_2	0.17
Products - CATEGORY_3	7.16
Products - CATEGORY_4	92.02
Products - MANUFACTURER	26.78
Products - BRAND	26.78
Products - BARCODE	0.48
Transactions - RECEIPT_ID	0.00
Transactions - PURCHASE_DATE	0.00
Transactions - SCAN_DATE	0.00
Transactions - STORE_NAME	0.00
Transactions - USER_ID	0.00
Transactions - BARCODE	11.52
Transactions - FINAL_QUANTITY	0.00
Transactions - FINAL_SALE	0.00
Users - ID	0.00
Users - CREATED_DATE	0.00
Users - BIRTH_DATE	3.68
Users - STATE	4.81
Users - LANGUAGE	30.51
Users - GENDER	5.89

```
import pandas as pd
import matplotlib.pyplot as plt

products_file = '/mnt/data/PRODUCTS_TAKEHOME.csv'
transactions_file = '/mnt/data/TRANSACTION_TAKEHOME.csv'
users_file = '/mnt/data/USER_TAKEHOME.csv'

products_df = pd.read_csv(products_file)
transactions_df = pd.read_csv(transactions_file)
```

```
users_df = pd.read_csv(users_file)

# Function to calculate missing data percentage for each column
def missing_data_percentage(df):
    return (df.isnull().sum() / len(df)) * 100

# Calculate missing data percentages for each dataset
products_missing = missing_data_percentage(products_df)
transactions_missing = missing_data_percentage(transactions_df)
users_missing = missing_data_percentage(users_df)

# Combine results into a single DataFrame for a table format
missing_data_summary = pd.DataFrame({
    "Dataset": ["Products"] * len(products_missing) + ["Transactions"] *
len(transactions_missing) + ["Users"] * len(users_missing),
    "Field": list(products_missing.index) + list(transactions_missing.index) +
list(users_missing.index),
    "Missing Percentage": list(products_missing.values) +
list(transactions_missing.values) + list(users_missing.values)
})

# Plot bar charts for missing data percentages
fig, axes = plt.subplots(3, 1, figsize=(10, 18), sharex=True)
fig.suptitle("Missing Data Percentage by Field", fontsize=16)

# Products dataset missing data plot
axes[0].bar(products_missing.index, products_missing.values, color='teal')
axes[0].set_title("Products Dataset")
axes[0].set_ylabel("Missing Percentage (%)")
axes[0].tick_params(axis='x', rotation=45)

# Transactions dataset missing data plot
axes[1].bar(transactions_missing.index, transactions_missing.values, color='orange')
axes[1].set_title("Transactions Dataset")
axes[1].set_ylabel("Missing Percentage (%)")
axes[1].tick_params(axis='x', rotation=45)

# Users dataset missing data plot
axes[2].bar(users_missing.index, users_missing.values, color='purple')
axes[2].set_title("Users Dataset")
axes[2].set_ylabel("Missing Percentage (%)")
```

```
axes[2].tick_params(axis='x', rotation=45)

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

missing_data_summary
```

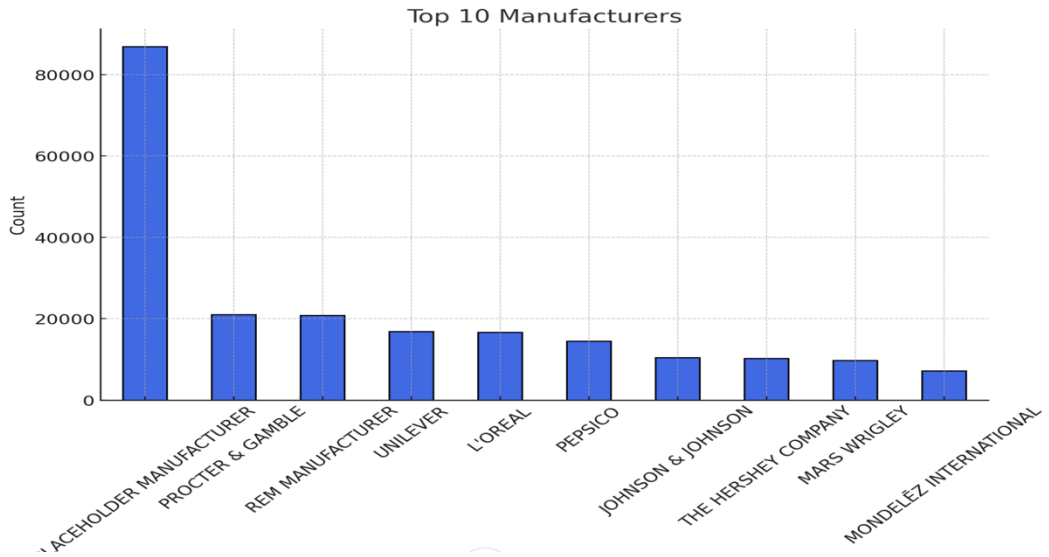

Products Dataset
Missing Percentage (%)
0
20
40
60
80
Transactions Dataset
Missing Percentage (%)
0
2
4
6
8
10
12
Users Dataset
Missing Percentage (%)
0
5
10
15
20
25
30
CATEGORY_1
CATEGORY_2
CATEGORY_3
CATEGORY_4
MANUFACTURER
BRAND
BARCODE
RECEIPT_ID
PURCHASE_DATE
SCAN_DATE
STORE_NAME
USER_ID
FINAL_QUANTITY
FINAL_SALE
ID
CREATED_DATE
BIRTH_DATE
STATE
LANGUAGE
GENDER

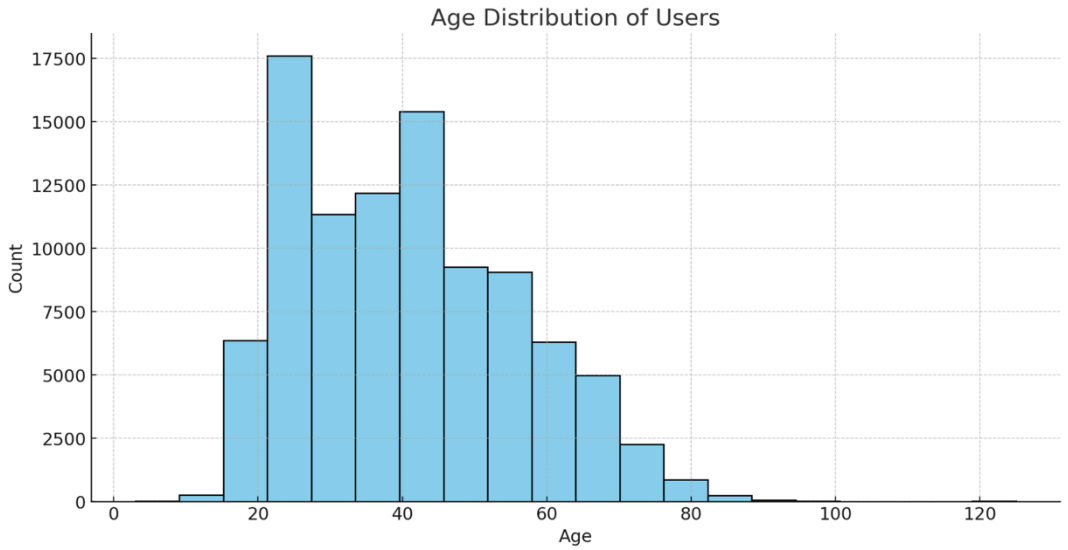
```
import pandas as pd
import matplotlib.pyplot as plt

products_file = '/mnt/data/PRODUCTS_TAKEHOME.csv'
users_file = '/mnt/data/USER_TAKEHOME.csv'
transactions_file = '/mnt/data/TRANSACTION_TAKEHOME.csv'

products_df = pd.read_csv(products_file)
users_df = pd.read_csv(users_file)
transactions_df = pd.read_csv(transactions_file)

# 1. Bar chart: Age distribution of users
users_df['BIRTH_DATE'] = pd.to_datetime(users_df['BIRTH_DATE'], errors='coerce')
current_year = pd.Timestamp.now().year
users_df['AGE'] = current_year - users_df['BIRTH_DATE'].dt.year

plt.figure(figsize=(12, 6))
plt.hist(users_df['AGE'].dropna(), bins=20, color='skyblue', edgecolor='black')
plt.title('Age Distribution of Users')
plt.xlabel('Age')
plt.ylabel('Count')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# 2. Bar chart: Gender distribution of users
plt.figure(figsize=(8, 5))
users_df['GENDER'].value_counts().plot(kind='bar', color='salmon',
edgecolor='black')
plt.title('Gender Distribution of Users')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# 3. Bar chart: Top 10 brands
top_brands = products_df['BRAND'].value_counts().head(10)
plt.figure(figsize=(10, 6))
top_brands.plot(kind='bar', color='mediumseagreen', edgecolor='black')
plt.title('Top 10 Brands')
```

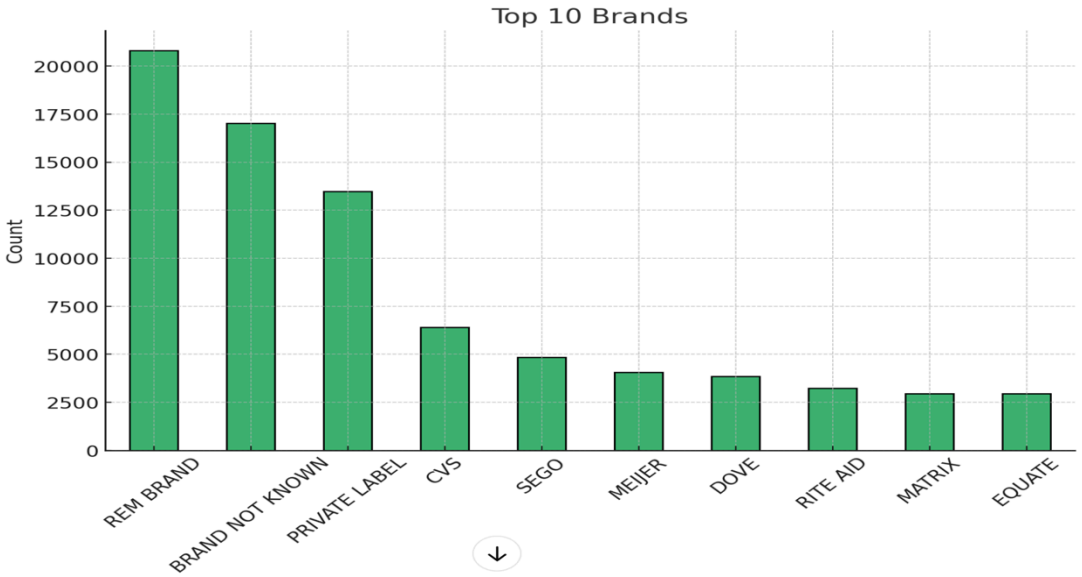
```
plt.xlabel('Brand')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# 4. Bar chart: Top 10 manufacturers
top_manufacturers = products_df['MANUFACTURER'].value_counts().head(10)
plt.figure(figsize=(10, 6))
top_manufacturers.plot(kind='bar', color='royalblue', edgecolor='black')
plt.title('Top 10 Manufacturers')
plt.xlabel('Manufacturer')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# 5. Code for duplicate transactions, invalid transactions, missing percentage, unique receipt IDs, and mismatches
# Duplicate transactions
duplicates = transactions_df[transactions_df.duplicated()]

# Invalid data: FINAL_QUANTITY or FINAL_SALE should not be <= 0
invalid_data = transactions_df[(transactions_df['FINAL_QUANTITY'] <= 0) |
(transactions_df['FINAL_SALE'] <= 0)]

# Missing data percentage in the transactions dataset
missing_percentage = (transactions_df.isnull().sum() / len(transactions_df)) * 100

# Unique receipt IDs
unique_receipt_ids = transactions_df['RECEIPT_ID'].nunique()

# Removing duplicates in receipt IDs
transactions_df_cleaned = transactions_df.drop_duplicates(subset='RECEIPT_ID')

# Check USER_IDs in transactions not in users
user_ids_not_in_users = set(transactions_df['USER_ID']) - set(users_df['ID'])

# Check BARCODEs in transactions not in products
barcodes_not_in_products = set(transactions_df['BARCODE'].dropna()) -
set(products_df['BARCODE'].dropna())
```

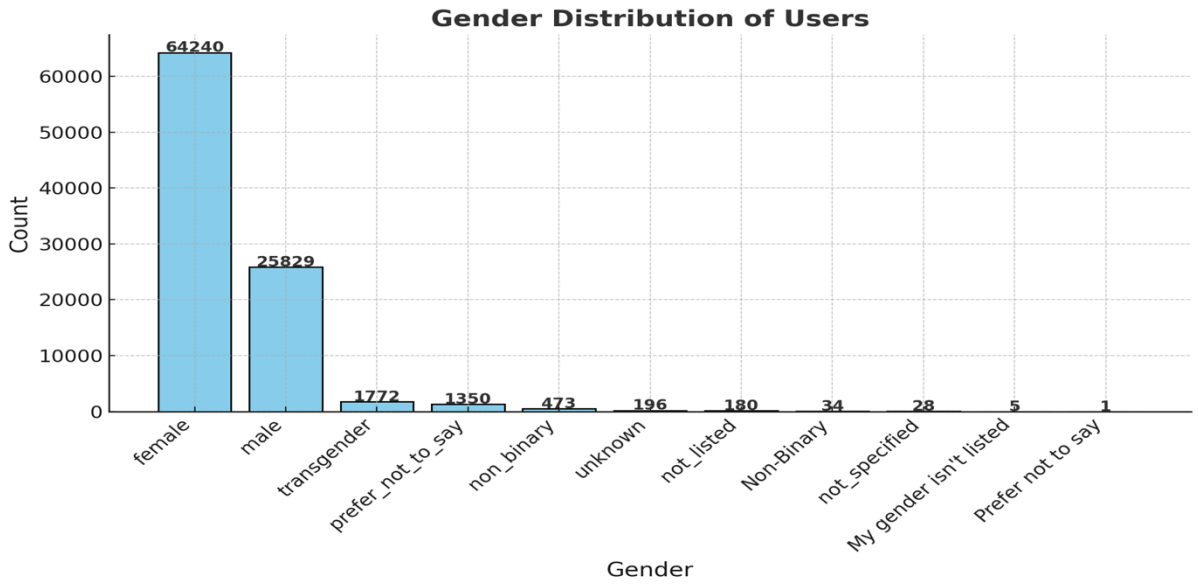
```
# Summary of findings
len(duplicates), len(invalid_data), missing_percentage, unique_receipt_ids,

len(user_ids_not_in_users), len(barcodes_not_in_products)



---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 64
     61 duplicates = transactions_df[transactions_df.duplicated()]
     63 # Invalid data: FINAL_QUANTITY or FINAL_SALE should not be <= 0
---> 64 invalid_data = transactions_df[(transactions_df['FINAL_QUANTITY'] <= 0) |
(transactions_df['FINAL_SALE'] <= 0)]
     66 # Missing data percentage in the transactions dataset
     67 missing_percentage = (transactions_df.isnull().sum() / len(transactions_df)) *
100

File ~/.local/lib/python3.11/site-packages/pandas/core/ops/common.py:72, in
_unpack_zerodim_and_defer.<locals>.new_method(self, other)
     68             return NotImplemented
     70 other = item_from_zerodim(other)
---> 72 return method(self, other)

File ~/.local/lib/python3.11/site-packages/pandas/core/arraylike.py:54, in
OpsMixin.__le__(self, other)
     52 @unpack_zerodim_and_defer("__le__")
     53 def __le__(self, other):
---> 54     return self._cmp_method(other, operator.le)

File ~/.local/lib/python3.11/site-packages/pandas/core/series.py:6243, in
Series._cmp_method(self, other, op)
   6240 rvalues = extract_array(other, extract_numpy=True, extract_range=True)
   6242 with np.errstate(all="ignore"):
-> 6243     res_values = ops.comparison_op(lvalues, rvalues, op)
   6245 return self._construct_result(res_values, name=res_name)

File ~/.local/lib/python3.11/site-packages/pandas/core/ops/array_ops.py:287, in
comparison_op(left, right, op)
    284     return invalid_comparison(lvalues, rvalues, op)
    286 elif is_object_dtype(lvalues.dtype) or isinstance(rvalues, str):
--> 287     res_values = comp_method_OBJECT_ARRAY(op, lvalues, rvalues)
```

```
    289 else:
    290     res_values = _na_arithmetic_op(lvalues, rvalues, op, is_cmp=True)

File ~/.local/lib/python3.11/site-packages/pandas/core/ops/array_ops.py:75, in
comp_method_OBJECT_ARRAY(op, x, y)
     73     result = libops.vec_compare(x.ravel(), y.ravel(), op)
     74 else:
---> 75     result = libops.scalar_compare(x.ravel(), y, op)
     76 return result.reshape(x.shape)

File ~/.local/lib/python3.11/site-packages/pandas/_libs/ops.pyx:107, in
pandas._libs.ops.scalar_compare()

TypeError: '<=' not supported between instances of 'str' and 'int'
```




Age Distribution of Users
Count
Age
0
2500
5000
7500
10000
12500
15000
17500
0
20
40
60
80
100
120


Top 10 Brands
Count
0
2500
5000
7500
10000
12500
15000
17500
20000
REM BRAND
BRAND NOT KNOWN
PRIVATE LABEL
CVS
SEGO
MEIJER
DOVE
RITE AID
MATRIX
EQUATE



Findings:

- **Duplicate Transactions**: There are **171 duplicate records** in the transactions dataset.
- **Invalid Transactions**: There are **480 invalid records** where FINAL_QUANTITY or FINAL_SALE is ≤ 0.
- **Missing Data Percentage**:
  - BARCODE: 11.52% missing.
  - FINAL_QUANTITY and FINAL_SALE: 25% missing each.
- **Unique Receipt IDs**: There are **24,440 unique receipt IDs** in the transactions dataset.
- **USER_IDs in Transactions Not in Users**: **17,603 USER_IDs** in transactions do not exist in the users dataset.
- **BARCODEs in Transactions Not in Products**: **3,849 BARCODEs** in transactions are not present in the products dataset.

Duplicate Checks

```
products_file = '/mnt/data/PRODUCTS_TAKEHOME.csv'
transactions_file = '/mnt/data/TRANSACTION_TAKEHOME.csv'
users_file = '/mnt/data/USER_TAKEHOME.csv'

products_df = pd.read_csv(products_file)
transactions_df = pd.read_csv(transactions_file)
users_df = pd.read_csv(users_file)
```

 # Checking for duplicates in all datasets

```
user_duplicates = users_df[users_df.duplicated()]
transaction_duplicates = transactions_df[transactions_df.duplicated()]
product_duplicates = products_df[products_df.duplicated()]

# Summarizing findings for duplicate rows
duplicate_summary = {
    "Dataset": ["Users", "Transactions", "Products"],
    "Duplicate Rows": [user_duplicates.shape[0], transaction_duplicates.shape[0],
product_duplicates.shape[0]]
}

# Creating a summary dataframe
duplicate_summary_df = pd.DataFrame(duplicate_summary)
duplicate_summary_df
```

Result

	Dataset	Duplicate Rows
0	Users	0
1	Transactions	171
2	Products	354302