**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")

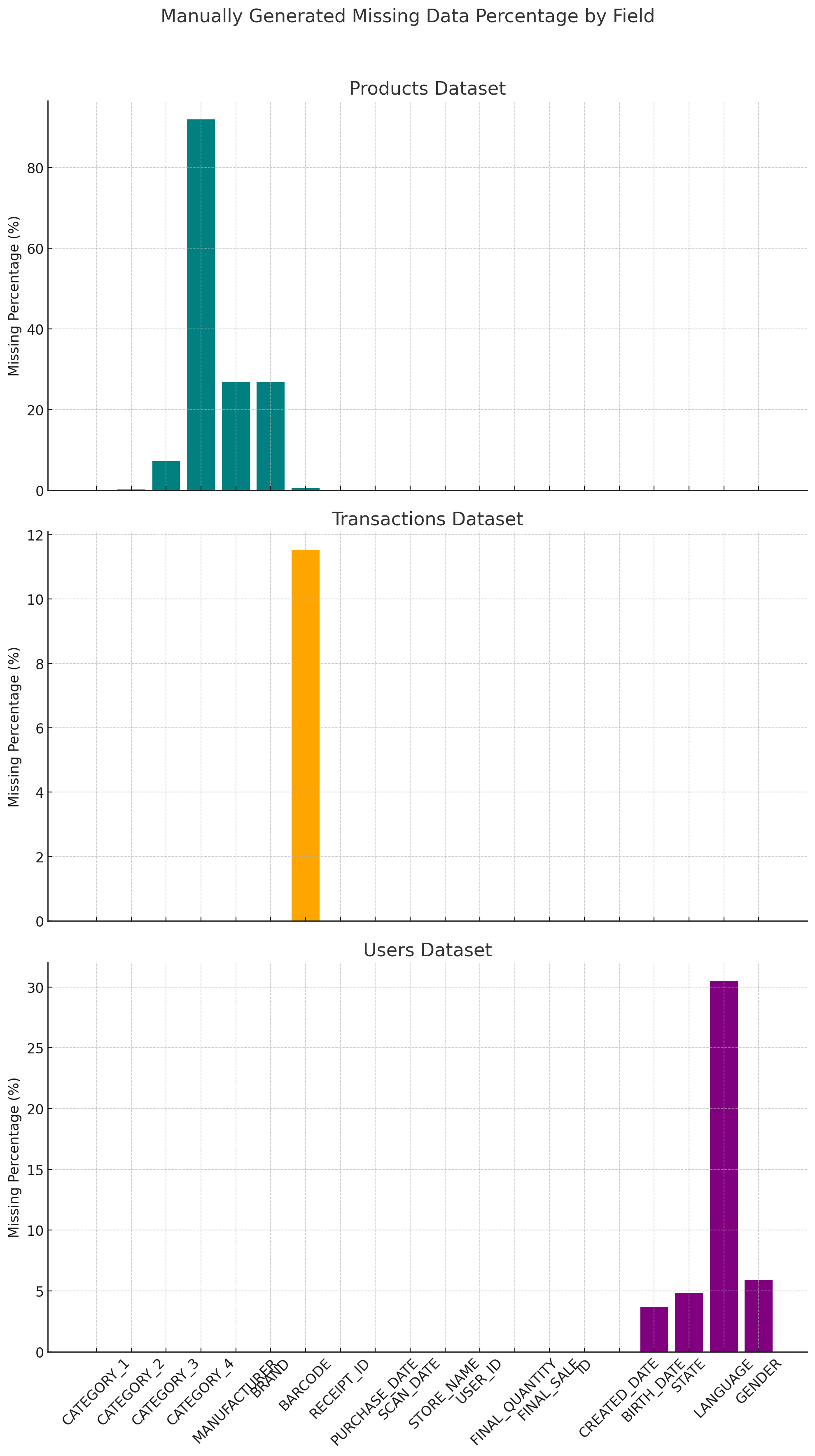
axes[2].set\_ylabel("Missing Percentage (%)")

axes[2].tick\_params(axis='x', rotation=45)

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.show()

missing\_data\_summary



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'  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chart

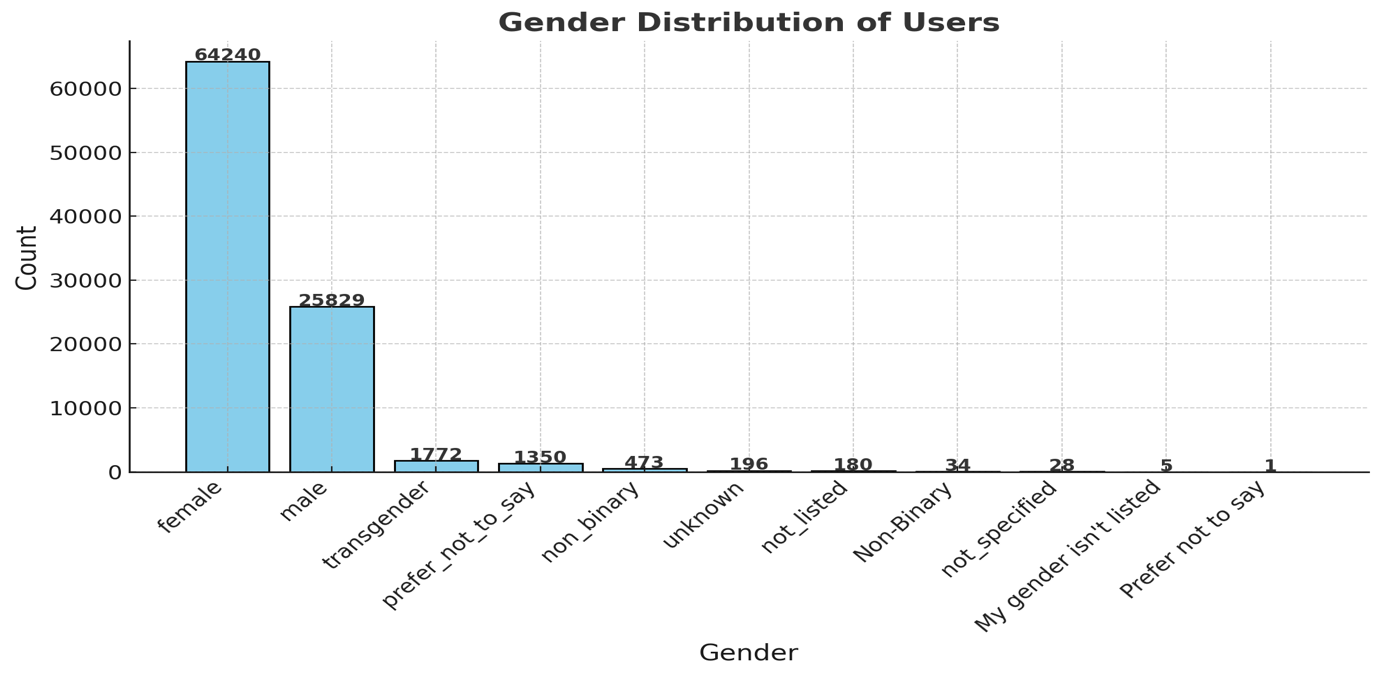
Description automatically generated

Chart, histogram

Description automatically generated

Chart

Description automatically generated



**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  
  
Text

Description automatically generated with medium confidence