Data Exploring

Load and View Data

In [17]:
```
# import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

In [18]:
```
# read data files
products = pd.read_csv('PRODUCTS.csv')
tran = pd.read_csv('TRANSACTION.csv')
user = pd.read_csv('USER.csv')
```

In [19]:
```
# View Dataframe and with its shape
print(products.shape)
products.head()
```

(845552, 7)

Out[19]:

	CATEGORY_1	CATEGORY_2	CATEGORY_3	CATEGORY_4	MANUFACTURER	BRAND	BARCODE
0	Health & Wellness	Sexual Health	Conductivity Gels & Lotions	NaN	NaN	NaN	7.964944e+11
1	Snacks	Puffed Snacks	Cheese Curls & Puffs	NaN	NaN	NaN	2.327801e+10
2	Health & Wellness	Hair Care	Hair Care Accessories	NaN	PLACEHOLDER MANUFACTURER	ELECSOP	4.618178e+11
3	Health & Wellness	Oral Care	Toothpaste	NaN	COLGATE-PALMOLIVE	COLGATE	3.500047e+10
4	Health & Wellness	Medicines & Treatments	Essential Oils	NaN	MAPLE HOLISTICS AND HONEYDEW PRODUCTS INTERCHA...	MAPLE HOLISTICS	8.068109e+11

In [20]:
```
print(tran.shape)
tran.head()
```

(50000, 8)

Out[20]:

	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	

In [21]:
```
print(user.shape)
user.head()
```

(100000, 6)

Out[21]:

	ID	CREATED_DATE	BIRTH_DATE	STATE	LANGUAGE	GENDER
0	5ef3b4f17053ab141787697d	2020-06-24 20:17:54.000 Z	2000-08-11 00:00:00.000 Z	CA	es-419	female
1	5ff220d383fcfc12622b96bc	2021-01-03 19:53:55.000 Z	2001-09-24 04:00:00.000 Z	PA	en	female
2	6477950aa55bb77a0e27ee10	2023-05-31 18:42:18.000 Z	1994-10-28 00:00:00.000 Z	FL	es-419	female
3	658a306e99b40f103b63ccf8	2023-12-26 01:46:22.000 Z	NaN	NC	en	NaN
4	653cf5d6a225ea102b7ecdc2	2023-10-28 11:51:50.000 Z	1972-03-19 00:00:00.000 Z	PA	en	female

I noticed that the transaction table has emplty cell but not showing miss value.

In [22]:
```
# Replace empty strings with NaN for correct missing value detection
tran.replace(" ", np.nan, inplace=True)
products.replace(" ", np.nan, inplace=True)
user.replace(" ", np.nan, inplace=True)

# Recheck missing values after cleaning empty strings
tran.head()
```

Out[22]:

	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	NaN
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	NaN
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	NaN

Data Quality Checking

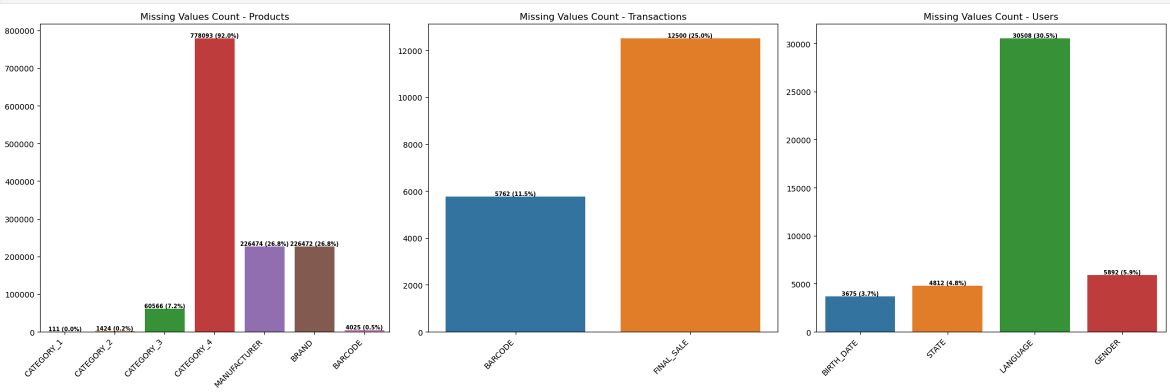
Unique Values in each coloms at each file

In [23]:
```
print(f"\nUnique Values Report for PRODUCTS:")
print(products.nunique().sort_values(ascending=True))
print(f"\nUnique Values Report for TRANSACTION:")
print(tran.nunique().sort_values(ascending=True))
print(f"\nUnique Values Report for USER:")
print(user.nunique().sort_values(ascending=True))
```

```
Unique Values Report for PRODUCTS:
CATEGORY_1          27
CATEGORY_2         121
CATEGORY_4         127
CATEGORY_3         344
MANUFACTURER      4354
BRAND             8122
BARCODE         841342
dtype: int64

Unique Values Report for TRANSACTION:
FINAL_QUANTITY       87
PURCHASE_DATE        89
STORE_NAME          954
FINAL_SALE         1434
BARCODE           11027
USER_ID           17694
RECEIPT_ID        24440
SCAN_DATE         24440
dtype: int64

Unique Values Report for USER:
LANGUAGE             2
GENDER              11
STATE               52
BIRTH_DATE       54721
CREATED_DATE     99942
ID              100000
dtype: int64
```

Visualize Missing Data

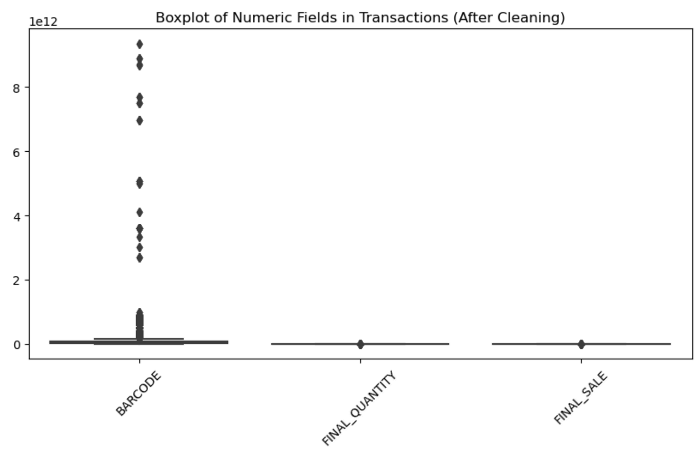
In [24]:
```python
def visualize_missing_data_pie(df, name, ax):
    missing_counts = df.isnull().sum()[df.isnull().sum() > 0]
    missing_percentages = (missing_counts / len(df)) * 100
    sns.barplot(x=missing_counts.index, y=missing_counts.values, ax=ax)
    ax.set_title(f'Missing Values Count - {name}')
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
    for p, percentage in zip(ax.patches, missing_percentages):
        ax.annotate(f'{int(p.get_height())} ({percentage:.1f}%)',
                    (p.get_x() + p.get_width() / 2, p.get_height()),
                    ha='center', va='bottom', fontsize=7, fontweight='bold')

# Plot missing data using pie charts
fig, axes = plt.subplots(1, 3, figsize=(21, 7))
visualize_missing_data_pie(products, "Products", axes[0])
visualize_missing_data_pie(tran, "Transactions", axes[1])
visualize_missing_data_pie(user, "Users", axes[2])
plt.tight_layout()
plt.show()
```



Anomalies in Numeric Columns

From previous results, there are no numerical value in User table, and 'BARCODE' is the only numerical columns in products table. Since 'BARCODE' appears in both Products table and Transactions table, I only check Transactions table.

In [25]:
```python
# Replace 'zero' with '0' in FINAL_QUANTITY and convert to numeric
tran["FINAL_QUANTITY"] = tran["FINAL_QUANTITY"].replace("zero", "0")
tran["FINAL_QUANTITY"] = pd.to_numeric(tran["FINAL_QUANTITY"], errors="coerce")

# Convert FINAL_SALE to numeric (handling empty strings and non-numeric values)
tran["FINAL_SALE"] = tran["FINAL_SALE"].replace(["", " "], np.nan)
tran["FINAL_SALE"] = pd.to_numeric(tran["FINAL_SALE"], errors="coerce")

# Detecting anomalies in numeric columns after cleaning
numeric_cols = tran.select_dtypes(include=['number']).columns

# Plot boxplot for numeric columns
plt.figure(figsize=(10, 5))
sns.boxplot(data=tran[numeric_cols])
plt.xticks(rotation=45)
plt.title("Boxplot of Numeric Fields in Transactions (After Cleaning)")
plt.show()
```



In [26]:
```
# Return updated data types to confirm changes
tran.dtypes
```

Out[26]:
```
RECEIPT_ID         object
PURCHASE_DATE      object
SCAN_DATE          object
STORE_NAME         object
USER_ID            object
BARCODE           float64
FINAL_QUANTITY    float64
FINAL_SALE        float64
dtype: object
```

Duplicate Checking

In [27]:
```
# check duplicates
print(f"\nDuplicate Records in PRODUCTS: ", products.duplicated().sum())
print(f"\nDuplicate Records in TRANSACTIONS: ", tran.duplicated().sum())
print(f"\nDuplicate Records in USER: ", tran.duplicated().sum())
```

```
Duplicate Records in PRODUCTS:  215

Duplicate Records in TRANSACTIONS:  171

Duplicate Records in USER:  171
```

Exploring Categorical Fields

In [28]:
```
# Understanding categorical fields
print("\nExample of categorical fields in Products dataset:")
print(products[['CATEGORY_1', 'CATEGORY_2', 'CATEGORY_3']].drop_duplicates().head(10))
```

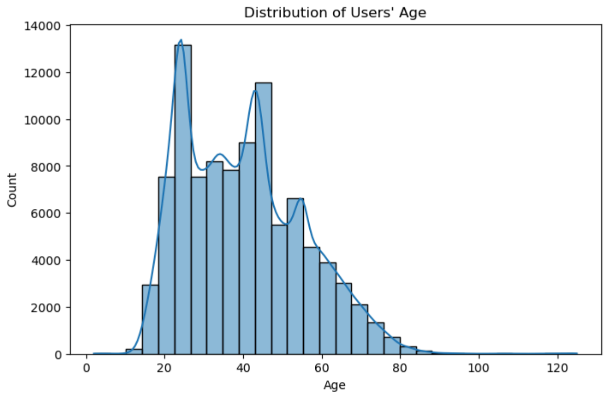
```
Example of categorical fields in Products dataset:
           CATEGORY_1                  CATEGORY_2  \
0   Health & Wellness               Sexual Health
1              Snacks               Puffed Snacks
2   Health & Wellness                   Hair Care
3   Health & Wellness                   Oral Care
4   Health & Wellness      Medicines & Treatments
6   Health & Wellness      Medicines & Treatments
7   Health & Wellness  Deodorant & Antiperspirant
8              Snacks                  Snack Bars
9   Health & Wellness                         NaN
11  Health & Wellness      Medicines & Treatments

                             CATEGORY_3
0        Conductivity Gels & Lotions
1               Cheese Curls & Puffs
2              Hair Care Accessories
3                         Toothpaste
4                     Essential Oils
6      Vitamins & Herbal Supplements
7   Men's Deodorant & Antiperspirant
8                       Granola Bars
9                                NaN
11                   Skin Treatments
```

In [29]:
```
# Since too many nan in Category_4, sperate this out
print(products[['CATEGORY_4']].drop_duplicates().head(10))
```

```
                              CATEGORY_4
0                                    NaN
15                   Hair Brushes & Combs
25             Women's Shaving Gel & Cream
31                              Lip Balms
39                    Already Popped Popcorn
109                           Men's Razors
115                           Snoring Aids
142  Popcorn Kernels & Popcorn Seasonings
143                             Sleep Aids
200                    Hair Straighteners
```

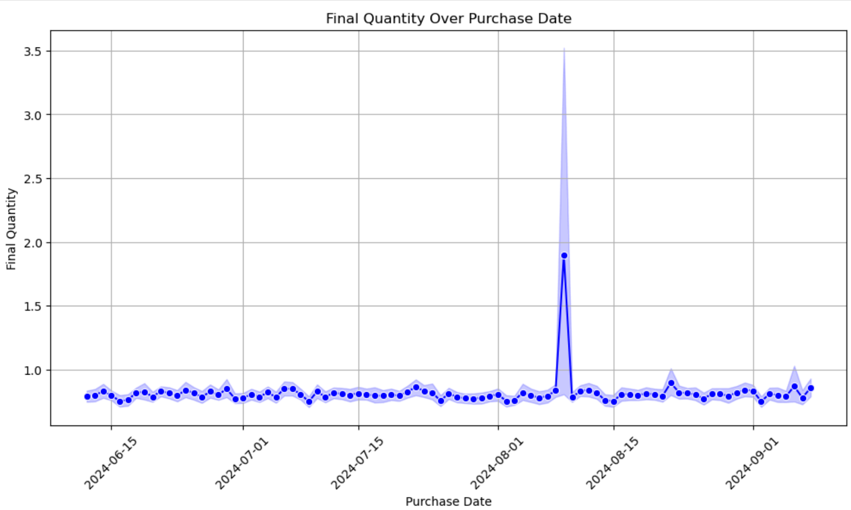
Visualizating Age Distribution

In [30]:
```
# Analyzing age distribution

user['BIRTH_DATE'] = pd.to_datetime(user['BIRTH_DATE'], errors='coerce')
# If BIRTH_DATE has timezone, remove it
if user['BIRTH_DATE'].dt.tz is not None:
    user['BIRTH_DATE'] = user['BIRTH_DATE'].dt.tz_localize(None)

# Ensure current date is also timezone-naive
current_date = datetime.now()

# Calculate age
user['AGE'] = (current_date - user['BIRTH_DATE']).dt.days // 365

plt.figure(figsize=(8, 5))
sns.histplot(user['AGE'].dropna(), bins=30, kde=True)
plt.title("Distribution of Users' Age")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```



Visualize Transaction Table about Time

In [33]:
```
# Convert date fields to datetime format
tran["PURCHASE_DATE"] = pd.to_datetime(tran["PURCHASE_DATE"], errors="coerce")
tran['SCAN_DATE'] = pd.to_datetime(tran['SCAN_DATE'], errors='coerce')

# Plot purchase date vs final quantity
plt.figure(figsize=(12, 6))
sns.lineplot(data=tran, x="PURCHASE_DATE", y="FINAL_QUANTITY", marker="o", color="blue")

plt.title("Final Quantity Over Purchase Date")
plt.xlabel("Purchase Date")
plt.ylabel("Final Quantity")
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



Summary of Findings

1. Missing values exist in various datasets, particularly in the Products dataset (CATEGORY_4, MANUFACTURER, BRAND).
2. The Transactions dataset has missing values in both BARCODE and FINAL_SALE, with FINAL_SALE being highly missing, which may impact sales analysis.
3. The FINAL_QUANTITY field contains values like "zero", which should be converted to numeric for accurate calculations.
4. The USERS dataset has missing values in the GENDER, LANGUAGE, and STATE fields, which may affect demographic insights. Additionally, LANGUAGE has a high percentage of missing values.
5. The dataset contains some duplicate records, which may require de-duplication.
6. The AGE distribution suggests potential outliers, with extreme values reaching 100+ years, which may indicate incorrect or missing birth dates.

In [34]:
```
# Save the cleaned datasets as rough_clean version
products.to_csv("rough_clean_products.csv", index=False)
tran.to_csv("rough_clean_transactions.csv", index=False)
user.to_csv("rough_clean_user.csv", index=False)
```