Retail Data Preprocessing — Cleaned & Detailed

1-- imports

```
# Common libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

These imports cover loading data, visualization, numeric operations and common preprocessing tools.

---

2 — Load dataset

```
# Load CSV
sales =pd.read_csv(r"C:\Users\juwer\Downloads\Ratails Analytics\sales_data.csv")

customers=pd.read_csv(r"C:\Users\juwer\Downloads\Ratails Analytics\customers.csv")

products=pd.read_csv(r"C:\Users\juwer\Downloads\Ratails Analytics\products.csv")

returns=pd.read_csv(r"C:\Users\juwer\Downloads\Ratails Analytics\returns.csv")

stores=pd.read_csv(r"C:\Users\juwer\Downloads\Ratails Analytics\stores.csv")
```

---

3 — Quick sanity checks

```
df.shape         # rows, columns
df.columns       # column names
df.head()        # first rows
df.info()        # dtypes and non-null counts
df.describe(include='all')to  Get a feel for data size, dtypes and obvious issues (e.g., lots of nulls).
```

---

4 — Clean column names

```
# make column names consistent and python-friendly
df.columns = (df.columns
              .str.strip()  # remove extra spaces
              .str.lower()  # converts string into lower_case
              .str.upper() # converts into upper_case
```

Consistent names prevent mistakes when referencing columns.

---

5 — Duplicates

```
# check duplicates (all columns)
df.duplicated().sum()
# if duplicates are present extract duplicates, drop them
df = df.drop_duplicates()
```

Exact duplicate rows can skew aggregates and counts.

---

6 — Missing value audit

example: customers['age'] = customers['age'].fillna(customers['age'].median())

Columns with >50% missing: consider dropping (unless business-critical) - Numeric columns: impute with median

Categorical columns: filling with 'Unknown' or a new category, or using of mode when appropriate

---

7 — Type conversions and datetime handling

```
# fixing of data type

customers['Signup_Date'] = pd.to_datetime(customers['Signup_Date'])
```

Correct dtypes enable date feature engineering.

---

8 — Derived columns

 # Creating derived column age_group from customers data

```
def Age_group(age):
    if age < 25:
        return 'Youth'
    elif age < 40:
        return 'Adult'
    elif age < 60:
        return 'Mid-Age'
    else:
        return 'Senior'
customers['Age_group'] = customers['Age'].apply(Age_group)
```

 # Caluclating profit in sales column from sales data

```
sales = sales.merge(products[['Product_Id', 'Cost_Price']], on='Product_Id', how='left')
sales['Profit'] = (sales['Total_Amount'] - (sales['Cost_Price'] * sales['Quantity'])).round(2)
```

---

9 — Outlier detection & treatment

IQR method (robust):

```
# Fixing of Outliers
# Columns to check
cols = ['Quantity', 'Unit_Price', 'Discount_Pct', 'Total_Amount']

# Loop through each column
for col in cols:
```

```
    Q1 = sales[col].quantile(0.25)
    Q3 = sales[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    # Count outliers
    outlier_count = ((sales[col] < lower) | (sales[col] > upper)).sum()
    print(f"Outliers in '{col}': {outlier_count}")

    # Fix outliers by capping
    sales[col] = sales[col].clip(lower, upper)
# Extreme values can bias mean/variance-based models and visualizations.
```

---

12 — Saving cleaned dataset

```
# csv
sales.to_csv('sales_cleaned.csv',index=False)
customers.to_csv('customers_cleaned.csv',index=False)
products.to_csv('products_cleaned.csv',index=False)
returns.to_csv('returns_cleaned.csv',index=False)
stores.to_csv('stores_cleaned.csv',index=False)
```

---

14 — Quick automated checklist

1. df.isnull().sum().sum() -> should be low/zero for modelling.
2. df.duplicated().sum() -> zero.
3. df.dtypes -> correct types.
4. df.describe() -> check ranges.