Notebook 1: Data Loading & Preprocessing

Purpose: Load Amazon Electronics Reviews, clean, and save for other notebooks.

Output: `amazon_electronics_cleaned.csv`

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Import libraries
import pandas as pd
import numpy as np
import gzip
import json
import urllib.request
import warnings
warnings.filterwarnings('ignore')

print("✅ Libraries imported")
```

```
✅ Libraries imported
```

Step 1: Load Dataset

```
# ============================================================================
# CONFIGURATION
# ============================================================================

SAMPLE_SIZE = 50000  # Number of reviews to load (increase for final run)
RANDOM_STATE = 42

print(f"Configuration:")
print(f"  Sample size: {SAMPLE_SIZE:,}")
print(f"  Random state: {RANDOM_STATE}")
```

```
Configuration:
  Sample size: 50,000
  Random state: 42
```

```
# ============================================================================
# LOAD AMAZON ELECTRONICS REVIEWS (2014 Dataset - Reliable)
# ============================================================================

print("=" * 70)
print("LOADING AMAZON ELECTRONICS REVIEWS DATASET")
print("Source: UCSD McAuley Lab (Stanford SNAP)")
print("=" * 70)

url = "http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Electronics_5.json.gz"

print(f"\nDownloading {SAMPLE_SIZE:,} reviews...")
print("⏳ This may take 1-2 minutes...\n")

try:
    # Download file
    urllib.request.urlretrieve(url, 'electronics.json.gz')

    # Load JSONL.gz file
    reviews = []
    with gzip.open('electronics.json.gz', 'rt', encoding='utf-8') as f:
        for i, line in enumerate(f):
            if i >= SAMPLE_SIZE:
                break
            if i % 10000 == 0 and i > 0:
                print(f"  Loaded {i:,} reviews...")
            reviews.append(json.loads(line))

    df_raw = pd.DataFrame(reviews)
    print(f"\n✅ SUCCESS! Loaded {len(df_raw):,} reviews")
```

```
    print(f"   Columns: {df_raw.columns.tolist()}")

except Exception as e:
    print(f"❌ Error: {e}")
    df_raw = None
```

```
======================================================================
LOADING AMAZON ELECTRONICS REVIEWS DATASET
Source: UCSD McAuley Lab (Stanford SNAP)
======================================================================

Downloading 50,000 reviews...
⏳ This may take 1-2 minutes...

  Loaded 10,000 reviews...
  Loaded 20,000 reviews...
  Loaded 30,000 reviews...
  Loaded 40,000 reviews...

✅ SUCCESS! Loaded 50,000 reviews
   Columns: ['reviewerID', 'asin', 'reviewerName', 'helpful', 'reviewText', 'overall', 'summary', 'unixReviewTime', 'reviewTime'
```

```
# Preview raw data
print("\n📋 Raw Data Preview:")
df_raw.head(3)
```

📋 Raw Data Preview:

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	AO94DHGC771SJ	0528881469	amazdnu	[0, 0]	We got this GPS for my husband who is an (OTR)...	5.0	Gotta have GPS!	1370131200	06 2, 2013
1	AMO214LNFCEI4	0528881469	Amazon Customer	[12, 15]	I'm a professional OTR truck driver, and I bou...	1.0	Very Disappointed	1290643200	11 25, 2010

Next steps: Generate code with df_raw | New interactive sheet

Step 2: Data Cleaning

```
# ======================================================================
# DATA CLEANING
# ======================================================================

print("=" * 70)
print("DATA CLEANING")
print("=" * 70)

# Rename columns to standard names
df = df_raw.rename(columns={
    'overall': 'rating',
    'reviewText': 'text'
}).copy()

print(f"\nOriginal size: {len(df):,} reviews")

# Keep only relevant columns
df = df[['text', 'rating']].copy()

# Step 1: Remove missing values
missing_before = df.isna().sum()
print(f"\nMissing values: text={missing_before['text']}, rating={missing_before['rating']}")
df = df.dropna(subset=['text', 'rating'])
print(f"After removing nulls: {len(df):,} reviews")

# Step 2: Remove very short reviews
df = df[df['text'].str.len() >= 10]
print(f"After removing short reviews (<10 chars): {len(df):,} reviews")

# Step 3: Convert rating to integer
df['rating'] = df['rating'].astype(int)

# Step 4: Verify ratings are 1-5
```

```
df = df[df['rating'].between(1, 5)]
print(f"After rating validation: {len(df):,} reviews")

# Reset index
df = df.reset_index(drop=True)
```

```
======================================================================
DATA CLEANING
======================================================================

Original size: 50,000 reviews

Missing values: text=0, rating=0
After removing nulls: 50,000 reviews
After removing short reviews (<10 chars): 49,960 reviews
After rating validation: 49,960 reviews

✅ Final cleaned dataset: 49,960 reviews
```

```
# Preview cleaned data
print("\n📋 Cleaned Data Preview:")
df.head()
```

📋 Cleaned Data Preview:

	text	rating
0	We got this GPS for my husband who is an (OTR)...	5
1	I'm a professional OTR truck driver, and I bou...	1
2	Well, what can I say. I've had this unit in m...	3
3	Not going to write a long review, even thought...	2
4	I've had mine for a year and here's what we go...	1

Next steps: Generate code with df | New interactive sheet

Step 3: Class Distribution

```
# ======================================================================
# CLASS DISTRIBUTION
# ======================================================================

print("=" * 70)
print("CLASS DISTRIBUTION")
print("=" * 70)

rating_counts = df['rating'].value_counts().sort_index()

print("\n📊 Rating Distribution:")
for rating, count in rating_counts.items():
    pct = count / len(df) * 100
    bar = '█' * int(pct / 2)
    print(f"   {rating} ⭐: {count:>6,} ({pct:>5.1f}%) {bar}")

print(f"\n   Total: {len(df):,} reviews")
```

```
======================================================================
CLASS DISTRIBUTION
======================================================================

📊 Rating Distribution:
   1 ⭐:  2,835 (  5.7%) ██
   2 ⭐:  2,161 (  4.3%) ██
   3 ⭐:  3,964 (  7.9%) ███
   4 ⭐: 10,103 ( 20.2%) ██████████
   5 ⭐: 30,897 ( 61.8%) ██████████████████████████████

   Total: 49,960 reviews
```

```
# Check class imbalance
print("\n⚠️ Class Imbalance Analysis:")
majority_class = rating_counts.max()
minority_class = rating_counts.min()
```

```
imbalance_ratio = majority_class / minority_class

print(f"   Majority class (5-star): {majority_class:,}")
print(f"   Minority class: {minority_class:,}")
```

```
⚠️ Class Imbalance Analysis:
   Majority class (5-star): 30,897
   Minority class: 2,161
   Imbalance ratio: 14.3:1
```

Step 4: Text Statistics

```
# ============================================================================
# TEXT STATISTICS
# ============================================================================

print("=" * 70)
print("TEXT STATISTICS")
print("=" * 70)

# Calculate text length
df['text_length'] = df['text'].str.len()
df['word_count'] = df['text'].str.split().str.len()

print("\n📏 Review Length (characters):")
print(f"   Min:    {df['text_length'].min():,}")
print(f"   Max:    {df['text_length'].max():,}")
print(f"   Mean:   {df['text_length'].mean():,.0f}")
print(f"   Median: {df['text_length'].median():,.0f}")

print("\n📝 Word Count:")
print(f"   Min:    {df['word_count'].min():,}")
print(f"   Max:    {df['word_count'].max():,}")
print(f"   Mean:   {df['word_count'].mean():,.0f}")
print(f"   Median: {df['word_count'].median():,.0f}")

# Drop helper columns before saving
df = df.drop(columns=['text_length', 'word_count'])
```

```
======================================================================
TEXT STATISTICS
======================================================================

📏 Review Length (characters):
   Min:    10
   Max:    15,567
   Mean:   577
   Median: 337

📝 Word Count:
   Min:    2
   Max:    2,845
   Mean:   105
   Median: 62
```

Step 5: Save Cleaned Data

```
# ============================================================================
# SAVE CLEANED DATA
# ============================================================================

print("=" * 70)
print("SAVING CLEANED DATA")
print("=" * 70)

import os

# Define output path
drive_path = '/content/drive/MyDrive/Fall 2025/Foundations of Artificial Intelligence/Final Project/data'
output_file = os.path.join(drive_path, 'amazon_electronics_cleaned.csv')

# Create directory if it doesn't exist
os.makedirs(drive_path, exist_ok=True)

# Save to CSV
```

```
df.to_csv(output_file, index=False)

print(f"\n✅ Successfully saved cleaned data to: {output_file}")

```

```
======================================================================
SAVING CLEANED DATA
======================================================================

✅ Successfully saved cleaned data to: /content/drive/MyDrive/Fall 2025/Foundations of Artificial Intelligence/Final Project/dat
```

```
# Download for Google Colab
try:
    from google.colab import files
    files.download(output_file)
    print("📥 Download started...")
except:
    print("Not in Colab - file saved locally")
```

```
📥 Download started...
```

✅ Summary

Data loaded and cleaned!

Metric	Value
Total reviews	See output above
Columns	text, rating
Rating range	1-5
Output file	amazon_electronics_cleaned.csv

Next: Run `2_EDA_Visualization.ipynb`

```
# Final summary
print("\n" + "=" * 70)
print("📋 NOTEBOOK 1 COMPLETE")
print("=" * 70)
print(f"\nDataset: Amazon Electronics Reviews")
print(f"Source: UCSD McAuley Lab")
print(f"Reviews: {len(df):,}")
print(f"Output: {output_file}")
print("\n→ Next: Run Notebook 2 (EDA & Visualization)")
```

```
======================================================================
📋 NOTEBOOK 1 COMPLETE
======================================================================

Dataset: Amazon Electronics Reviews
Source: UCSD McAuley Lab
Reviews: 49,960
Output: /content/drive/MyDrive/Fall 2025/Foundations of Artificial Intelligence/Final Project/data/amazon_electronics_cleaned.cs

→ Next: Run Notebook 2 (EDA & Visualization)
```