Insurance Data Wrangling & Cleaning

Data wrangling ensures the raw insurance dataset is accurate, consistent, and ready for exploratory analysis and modeling.
This notebook details the full pipeline used to clean, impute, encode, and scale the data before EDA and machine-learning tasks.

1. Project Overview

- **Business Context**
  An insurance company collects rich customer data (demographics, policies, risk factors). Before modeling premium drivers or customer behavior, the raw data must be cleaned and standardized.
- **Goal**
  Detect and fix inconsistencies, handle missing values, encode categorical variables, and scale numeric features to produce a machine-learning–ready dataset.
- **Deliverable**
  `Insurance_cleaned.csv` – fully cleaned, encoded, and scaled dataset.

2. Data Source

- **Source:** https://www.kaggle.com/datasets/ethancollins9786/us-health-insurance-premiums-and-risk-factors
- **Original File:** `Insurance.csv`
- **Size:** 2,082 rows × 23 columns
- **Key Fields:**
  `Customer ID`, `Age`, `Gender`, `Marital Status`, `Occupation`, `Income Level`, `Education Level`, `Coverage Amount`, `Premium Amount`, `Deductible`, `Policy Type`, `Preferred Communication Channel`, `Risk Profile`, `Credit Score`, `Driving Record`, `Life Events`.

3. Cleaning & Transformation Steps

3.1 Load the Dataset

In [9]:
```
import pandas as pd
import numpy as np
from sqlalchemy import create_engine
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import SimpleImputer
```

```
from sklearn.impute import IterativeImputer

# Read the CSV
df = pd.read_csv("data/Insurance.csv")
df.head()
```

Out[9]:

	Customer ID	Age	Gender	Marital Status	Occupation	Income Level	Education Level	Geographic Information	Locati
0	15043	48	Female	Single	Engineer	72654	Associate Degree	Karnataka	751
1	88777	50	Male	Divorced	Manager	93448	Master's Degree	Karnataka	567
2	62911	53	Male	Widowed	Doctor	92558	Doctorate	Arunachal Pradesh	602
3	38955	38	Male	Widowed	Salesperson	78536	High School Diploma	Andhra Pradesh	347
4	3935	42	Male	Married	Salesperson	90220	High School Diploma	Puducherry	142

5 rows × 23 columns

In [2]:
```
print("Shape:", df.shape)
```

```
Shape: (2082, 23)
```

In [3]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2082 entries, 0 to 2081
Data columns (total 23 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Customer ID                      2082 non-null   int64
 1   Age                              2082 non-null   int64
 2   Gender                           2082 non-null   object
 3   Marital Status                   2082 non-null   object
 4   Occupation                       2082 non-null   object
 5   Income Level                     2082 non-null   int64
 6   Education Level                  2082 non-null   object
 7   Geographic Information           2082 non-null   object
 8   Location                         2082 non-null   int64
 9   Purchase History                 2082 non-null   object
 10  Policy Start Date                2082 non-null   object
 11  Policy Renewal Date              2082 non-null   object
 12  Claim History                    2082 non-null   int64
 13  Coverage Amount                  2082 non-null   int64
 14  Premium Amount                   2082 non-null   int64
 15  Deductible                       2082 non-null   int64
 16  Policy Type                      2082 non-null   object
 17  Preferred Communication Channel  2082 non-null   object
 18  Preferred Contact Time           2082 non-null   object
 19  Risk Profile                     2082 non-null   int64
 20  Credit Score                     2082 non-null   int64
 21  Driving Record                   2082 non-null   object
 22  Life Events                      2082 non-null   object
dtypes: int64(10), object(13)
memory usage: 374.2+ KB
None
```

3.2 Identify & Remove Inconsistent Entries

Examples of “inconsistency” include impossible ages, negative premiums, or unexpected categories.

In [4]:
```python
# Quick sanity checks
print("Age range:", df['Age'].min(), "to", df['Age'].max())
print("Premium range:", df['Premium Amount'].min(), "to", df['Premium Amount'].max(

# Remove impossible values
df = df[df['Age'].between(18, 100)]
df = df[df['Premium Amount'] > 0]
df = df[df['Coverage Amount'] > 0]

# Check category typos
for col in ['Gender','Marital Status','Policy Type']:
    print(f"\nUnique values in {col}:", sorted(df[col].unique()))
```

```
Age range: 27 to 63
Premium range: 600 to 4800

Unique values in Gender: ['Female', 'Male']

Unique values in Marital Status: ['Divorced', 'Married', 'Separated', 'Single', 'Wid
owed']

Unique values in Policy Type: ['Business', 'Family', 'Group', 'Individual']
```

Rows with negative premiums, zero coverage, or unrealistic ages are dropped. For text fields, reviewing unique values helps spot typos or inconsistent casing.

3.3 Handle Missing Values

In [11]:
```
# Numeric columns
num_cols = df.select_dtypes(include=[np.number]).columns
# Simple median imputation
median_imputer = SimpleImputer(strategy='median')
df[num_cols] = median_imputer.fit_transform(df[num_cols])

# Categorical columns
cat_cols = df.select_dtypes(exclude=[np.number]).columns
mode_imputer = SimpleImputer(strategy='most_frequent')
df[cat_cols] = mode_imputer.fit_transform(df[cat_cols])

print("Missing values after imputation:\n", df.isnull().sum().sum())
```

```
Missing values after imputation:
 0
```

For number columns, missing values are first filled with the median of that column. Then an Iterative Imputer goes back and improves those numbers by predicting each column based on the others. For text or category columns, missing values are filled with the most common category in that column.

3.4 Encode Categorical Variables

Encoding converts text categories to numbers for analysis or machine learning.

In [13]:
```
# One-Hot Encode high-cardinality fields carefully
df_encoded = pd.get_dummies(df,
                            columns=['Gender','Marital Status','Occupation',
                                     'Education Level','Policy Type'],
                            drop_first=True)
print("Encoded shape:", df_encoded.shape)
```

```
Encoded shape: (2082, 38)
```

Each category becomes a binary column (1 or 0), enabling numeric analysis.

3.5 Feature Scaling & Transformation

Normalize or standardize numeric features to comparable ranges.

In [15]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
scaler = StandardScaler()

scale_cols = ['Premium Amount','Coverage Amount','Deductible','Income Level','Credi
df_encoded[scale_cols] = scaler.fit_transform(df_encoded[scale_cols])
```

StandardScaler centers data to mean 0, std 1—helpful for models like logistic regression. Log transforms reduce skew for heavily right-tailed columns (e.g., Premium).

3.6 Final Check

In [18]:
```
print("Final shape:", df_encoded.shape)
print("Any remaining missing values", df_encoded.isnull().sum().sum())
```

```
Final shape: (2082, 38)
Any remaining missing values 0
```

4. Data Wrangling Summary

- Removed unrealistic ages, negative premiums, and other inconsistencies
- Imputed missing numeric values using both median and iterative multiple imputation; categorical fields filled with the most frequent category
- Encoded categorical variables with one-hot encoding for modeling readiness
- Scaled and transformed numeric features (standardization and optional log transform) to improve comparability and reduce skewness

In [19]:
```
# Save the cleaned, encoded, and scaled dataset
df_encoded.to_csv("data/Insurance_cleaned.csv", index=False)
```

In [20]:
```
df.head()
```

Out[20]:

	Customer ID	Age	Gender	Marital Status	Occupation	Income Level	Education Level	Geographic Information	Locati
0	15043.0	48.0	Female	Single	Engineer	72654.0	Associate Degree	Karnataka	7517
1	88777.0	50.0	Male	Divorced	Manager	93448.0	Master's Degree	Karnataka	5670
2	62911.0	53.0	Male	Widowed	Doctor	92558.0	Doctorate	Arunachal Pradesh	6022
3	38955.0	38.0	Male	Widowed	Salesperson	78536.0	High School Diploma	Andhra Pradesh	3470
4	3935.0	42.0	Male	Married	Salesperson	90220.0	High School Diploma	Puducherry	1422

5 rows × 23 columns

In [21]:
```
df = pd.read_csv("data/Insurance_cleaned.csv")
df.head()
```

Out[21]:

	Customer ID	Age	Income Level	Geographic Information	Location	Purchase History	Policy Start Date	Polic Renew Dat
0	15043.0	0.568694	-0.852252	Karnataka	75177.0	1/24/2020	7/23/2019	7/24/202
1	88777.0	0.928087	0.464951	Karnataka	56707.0	1/10/2023	11/10/2021	1/14/202
2	62911.0	1.467176	0.408573	Arunachal Pradesh	60225.0	12/12/2021	11/7/2023	6/2/202
3	38955.0	-1.228271	-0.479655	Andhra Pradesh	34707.0	2/1/2023	6/1/2018	2/13/202
4	3935.0	-0.509485	0.260472	Puducherry	14225.0	2/18/2021	6/4/2023	5/15/202

5 rows × 38 columns

5. Next Steps

The cleaned dataset can now feed:

- Exploratory Data Analysis (EDA) for premium drivers and customer segmentation.
- Machine learning models for churn prediction, risk scoring, or premium estimation.