Data Cleaning Workflow Template

This notebook is a reusable template for cleaning raw data:

1. Load data
2. Inspect structure
3. Handle missing values
4. Remove duplicates
5. Fix data types
6. Clean text/categories
7. Detect & cap outliers (IQR)
8. Validate & export

In [11]:

```
# === 1. IMPORTS & SETTINGS ===

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', lambda x: f'{x:.2f}')

print("Libraries loaded.")
```

```
Libraries loaded.
```

In [6]:

```
   3# === 2. LOAD RAW DATA ===
# TODO: Change the file name/path for each new project.

DATA_PATH = "/home/jkatz015/repos/personal/portfolio/projects/project-02-ap-automation/data/invoices_raw/ap_transacti
df = pd.read_csv(DATA_PATH)

print("Rows:", len(df))
df.head()
```

```
Rows: 50150
```

Out[6]:

	INVOICE_ID	VENDOR_ID	VENDOR_NAME	VENDOR_NUMBER	VENDOR_COUNTRY	VENDOR_REGION	INVOICE_DATE	DUE_DA
0	873641333	846210656	Metro Solutions	1230020547	Brunei Darussalam	North-Region	2022-04-11	2022-
1	865186916	892667166	Metro Corp	1230022657	Spain	North-Region	2022-03-09	2022-
2	914629138	921212744	Superior Enterprises	1230002613	Kenya	South-Region	2022-02-23	2022-
3	861070279	803387626	Global Manufacturing	1230014342	Turkmenistan	North-Region	2022-02-01	2022-
4	784182066	942636818	Quality Partners	1230014342	Guam	North-Region	2022-02-14	2022-

In [12]:

```
# === 3. BASIC STRUCTURE CHECK ===

df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50150 entries, 0 to 50149
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   INVOICE_ID       50150 non-null  int64  
 1   VENDOR_ID        50150 non-null  int64  
 2   VENDOR_NAME      50150 non-null  object 
 3   VENDOR_NUMBER    50150 non-null  int64  
 4   VENDOR_COUNTRY   50150 non-null  object 
 5   VENDOR_REGION    50150 non-null  object 
 6   INVOICE_DATE     50150 non-null  object 
 7   DUE_DATE         50150 non-null  object 
 8   INVOICE_AMOUNT   50150 non-null  float64
 9   CURRENCY         50150 non-null  object 
 10  GL_CODE          50150 non-null  object 
 11  PO_TYPE          50150 non-null  object 
 12  COMPANY_CODE     50150 non-null  int64  
 13  BUSINESS_UNIT    50150 non-null  int64  
 14  COST_CENTER      50150 non-null  object 
 15  PAYMENT_STATUS   50150 non-null  object 
 16  APPROVAL_STATUS  50150 non-null  float64
 17  CREDIT_LIMIT     50150 non-null  float64
 18  INVOICE_TIME     50150 non-null  int64  
 19  _ANOMALY_TYPE    50150 non-null  object 
dtypes: float64(3), int64(6), object(11)
memory usage: 7.7+ MB
```

Out[12]:

	INVOICE_ID	VENDOR_ID	VENDOR_NUMBER	INVOICE_AMOUNT	COMPANY_CODE	BUSINESS_UNIT	APPROVAL_STATU
count	50150.00	50150.00	50150.00	50150.00	50150.00	50150.00	50150.
mean	879640407.81	876513993.64	1111686783.56	6317.53	3415.36	3304.16	66.
std	86465713.92	70885396.17	357815061.23	216296.24	594.99	696.53	8.
min	753460934.00	753458315.00	12100061.00	0.01	59.00	2100.00	52.
25%	815645178.75	815235731.75	1230011793.00	189.04	3260.00	2702.00	64.
50%	877306681.50	876551085.50	1230014314.00	457.02	3660.00	3300.00	64.
75%	938984852.25	938060359.25	1230014342.00	1057.20	3660.00	3908.00	64.
max	1897434115.00	999007452.00	1230025103.00	44854887.00	4260.00	4510.00	93.

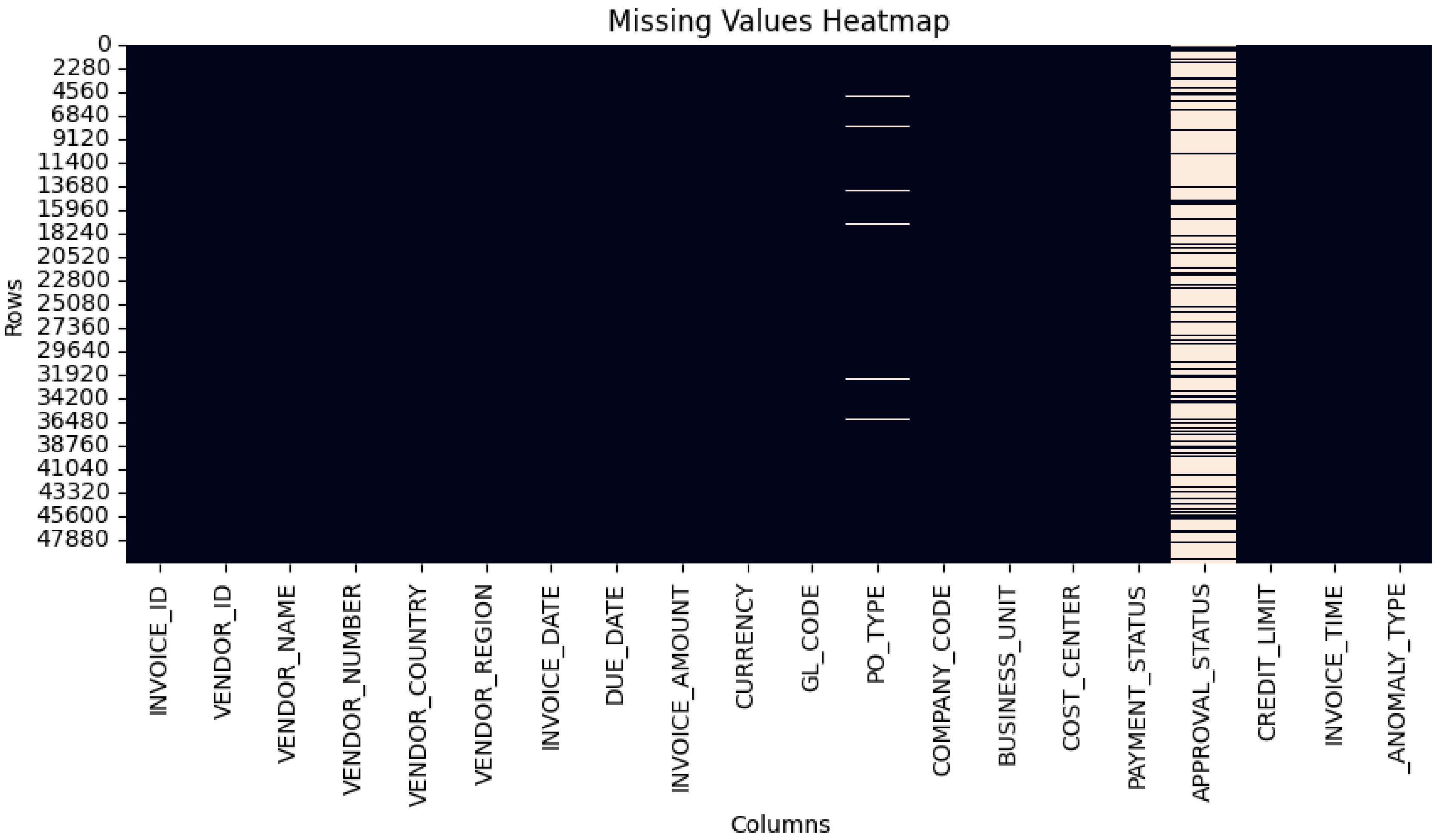
In [8]:

```
# === 4. MISSING VALUES OVERVIEW ===

print("Missing values per column:")
print(df.isnull().sum())

plt.figure(figsize=(10, 4))
sns.heatmap(df.isnull(), cbar=False)
plt.title("Missing Values Heatmap")
plt.xlabel("Columns")
plt.ylabel("Rows")
plt.show()
```

```
Missing values per column:
INVOICE_ID            0
VENDOR_ID             0
VENDOR_NAME           0
VENDOR_NUMBER         0
VENDOR_COUNTRY        0
VENDOR_REGION         0
INVOICE_DATE          0
DUE_DATE              0
INVOICE_AMOUNT        0
CURRENCY              0
GL_CODE               0
PO_TYPE            1063
COMPANY_CODE          0
BUSINESS_UNIT         0
COST_CENTER           0
PAYMENT_STATUS        0
APPROVAL_STATUS   38811
CREDIT_LIMIT          0
INVOICE_TIME          0
_ANOMALY_TYPE         0
dtype: int64
```



In [13]:

```
# === 5. IDENTIFY COLUMN TYPES ===

numeric_cols = df.select_dtypes(include=['float64', 'int64', 'Int64']).columns.tolist()
text_cols    = df.select_dtypes(include=['object']).columns.tolist()

print("Numeric columns:", numeric_cols)
print("Text columns:", text_cols)
```

```
Numeric columns: ['INVOICE_ID', 'VENDOR_ID', 'VENDOR_NUMBER', 'INVOICE_AMOUNT', 'COMPANY_CODE', 'BUSINESS_UNIT', 'APP
ROVAL_STATUS', 'CREDIT_LIMIT', 'INVOICE_TIME']
Text columns: ['VENDOR_NAME', 'VENDOR_COUNTRY', 'VENDOR_REGION', 'INVOICE_DATE', 'DUE_DATE', 'CURRENCY', 'GL_CODE',
'PO_TYPE', 'COST_CENTER', 'PAYMENT_STATUS', '_ANOMALY_TYPE']
```

In [14]:

```python
# === 6. HANDLE MISSING VALUES ===
# Adjust per project if needed.

# 6a. Numeric → fill with median
for col in numeric_cols:
    if df[col].isnull().sum() > 0:
        median_value = df[col].median()
        df.loc[:, col] = df[col].fillna(median_value)

# 6b. Text → fill with "Unknown"
for col in text_cols:
    if df[col].isnull().sum() > 0:
        df.loc[:, col] = df[col].fillna("Unknown")

print("Missing values after fill:")
print(df.isnull().sum())
```

```
Missing values after fill:
INVOICE_ID         0
VENDOR_ID          0
VENDOR_NAME        0
VENDOR_NUMBER      0
VENDOR_COUNTRY     0
VENDOR_REGION      0
INVOICE_DATE       0
DUE_DATE           0
INVOICE_AMOUNT     0
CURRENCY           0
GL_CODE            0
PO_TYPE            0
COMPANY_CODE       0
BUSINESS_UNIT      0
COST_CENTER        0
PAYMENT_STATUS     0
APPROVAL_STATUS    0
CREDIT_LIMIT       0
INVOICE_TIME       0
_ANOMALY_TYPE      0
dtype: int64
```

In [15]:

```python
# === 7. REMOVE DUPLICATES ===

dupes_before = df.duplicated().sum()
print("Duplicates before:", dupes_before)

df = df.drop_duplicates()

dupes_after = df.duplicated().sum()
print("Duplicates after:", dupes_after)
```

```
Duplicates before: 0
Duplicates after: 0
```

In [16]:

```python
# === 8. FIX DATA TYPES ===

# Example: convert columns whose name suggests date/time
date_like_cols = [c for c in df.columns if 'date' in c.lower()]
for col in date_like_cols:
    df.loc[:, col] = pd.to_datetime(df[col], errors='coerce')
```

```
# Example: convert typical integer-like columns if present
int_candidate_cols = ['age', 'quantity', 'count']  # edit per dataset
for col in int_candidate_cols:
    if col in df.columns:
        df.loc[:, col] = df[col].astype('Int64')

print(df.dtypes)
```

```
INVOICE_ID           int64
VENDOR_ID            int64
VENDOR_NAME         object
VENDOR_NUMBER        int64
VENDOR_COUNTRY      object
VENDOR_REGION       object
INVOICE_DATE        object
DUE_DATE            object
INVOICE_AMOUNT     float64
CURRENCY            object
GL_CODE             object
PO_TYPE             object
COMPANY_CODE         int64
BUSINESS_UNIT        int64
COST_CENTER         object
PAYMENT_STATUS      object
APPROVAL_STATUS    float64
CREDIT_LIMIT       float64
INVOICE_TIME         int64
_ANOMALY_TYPE       object
dtype: object
```

In [19]:

```
# === 9. CLEAN TEXT / CATEGORICAL COLUMNS ===

for col in text_cols:
    df.loc[:, col] = df[col].astype(str)
    df.loc[:, col] = df[col].str.strip()
    df.loc[:, col] = df[col].str.upper()

# Example: inspect a key category, if it exists
if 'CITY' in df.columns:
    print(df['CITY'].value_counts().head(20))
```

In [20]:

```
# === 10. OUTLIER DETECTION & CAPPING (IQR) ===

def iqr_bounds(series: pd.Series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return lower, upper

for col in numeric_cols:
    if df[col].nunique() <= 1:
        continue  # skip constant columns

    lower, upper = iqr_bounds(df[col])
    print(f"{col}: lower={lower:.2f}, upper={upper:.2f}")

    df.loc[df[col] < lower, col] = lower
    df.loc[df[col] > upper, col] = upper
```

```
INVOICE_ID: lower=630635668.50, upper=1123994362.50
VENDOR_ID: lower=630998790.50, upper=1122297300.50
VENDOR_NUMBER: lower=1230007969.50, upper=1230018165.50
INVOICE_AMOUNT: lower=-1113.21, upper=2359.44
COMPANY_CODE: lower=2660.00, upper=4260.00
BUSINESS_UNIT: lower=893.00, upper=5717.00
INVOICE_TIME: lower=-67351.12, upper=337595.88
```

In [21]:

```
# === 11. FINAL VALIDATION ===

print("Missing values per column:")
print(df.isnull().sum())

print("\nDuplicates:", df.duplicated().sum())

print("\nData types:")
print(df.dtypes)

print("\nPreview:")
df.head()
```

```
Missing values per column:
INVOICE_ID         0
VENDOR_ID          0
VENDOR_NAME        0
VENDOR_NUMBER      0
VENDOR_COUNTRY     0
VENDOR_REGION      0
INVOICE_DATE       0
DUE_DATE           0
INVOICE_AMOUNT     0
CURRENCY           0
GL_CODE            0
PO_TYPE            0
COMPANY_CODE       0
BUSINESS_UNIT      0
COST_CENTER        0
PAYMENT_STATUS     0
APPROVAL_STATUS    0
CREDIT_LIMIT       0
INVOICE_TIME       0
_ANOMALY_TYPE      0
dtype: int64

Duplicates: 0

Data types:
INVOICE_ID         float64
VENDOR_ID          float64
VENDOR_NAME         object
VENDOR_NUMBER      float64
VENDOR_COUNTRY      object
VENDOR_REGION       object
INVOICE_DATE        object
DUE_DATE            object
INVOICE_AMOUNT     float64
CURRENCY            object
GL_CODE             object
PO_TYPE             object
COMPANY_CODE         int64
BUSINESS_UNIT        int64
COST_CENTER         object
PAYMENT_STATUS      object
```

```
APPROVAL_STATUS    float64
CREDIT_LIMIT       float64
INVOICE_TIME       float64
_ANOMALY_TYPE       object
dtype: object

Preview:
```

Out[21]:

	INVOICE_ID	VENDOR_ID	VENDOR_NAME	VENDOR_NUMBER	VENDOR_COUNTRY	VENDOR_REGION	INVOICE_DATE	DU
0	873641333.00	846210656.00	METRO SOLUTIONS	1230018165.50	BRUNEI DARUSSALAM	NORTH-REGION	2022-04-11 00:00:00	20 0
1	865186916.00	892667166.00	METRO CORP	1230018165.50	SPAIN	NORTH-REGION	2022-03-09 00:00:00	20 0
2	914629138.00	921212744.00	SUPERIOR ENTERPRISES	1230007969.50	KENYA	SOUTH-REGION	2022-02-23 00:00:00	20 0
3	861070279.00	803387626.00	GLOBAL MANUFACTURING	1230014342.00	TURKMENISTAN	NORTH-REGION	2022-02-01 00:00:00	20 0
4	784182066.00	942636818.00	QUALITY PARTNERS	1230014342.00	GUAM	NORTH-REGION	2022-02-14 00:00:00	20 0

In [ ]: