Solutions for Pandas Tasks Based on customers-10000.csv

Generated for Practice

Introduction

This document provides full solutions (with code and explanation) for all Simple, Intermediate, and Hard tasks based on the dataset `customers-10000.csv`. You can run every code block directly in Jupyter Notebook.

Simple Task Solutions

S1. Load the CSV file and display the first 5 rows.

Solution:

```
import pandas as pd

df = pd.read_csv("customers-10000.csv")
df.head()
```

Explanation: Loads the dataset and shows the first 5 rows.

S2. Show the number of rows and columns in the dataset.

```
df.shape
```

Explanation: Returns (rows, columns).

S3. Print all column names.

```
df.columns
```

S4. Show summary info and statistical summary.

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

S5. Select and display only the First Name column.

```
df['First Name']
```

S6. Select Country and Email columns.

```
df[['Country', 'Email']]
```

S7. Filter customers from the United States.

```
df[df['Country'] == 'United States']
```

S8. Filter emails ending with .org.

```
df[df['Email'].str.endswith('.org')]
```

S9. Create Full Name column.

```
df['Full Name'] = df['First Name'] + " " + df['Last Name']
```

S10. Convert Subscription Date to datetime.

```
df['Subscription Date'] = pd.to_datetime(df['Subscription
   Date'], errors='coerce')
```

S11. Find earliest and latest subscription dates.

```
df['Subscription Date'].min()
df['Subscription Date'].max()
```

S12. Count missing values.

```
df.isnull().sum()
```

Intermediate Task Solutions

I1. Clean phone numbers by removing symbols and extensions.

```
df['Phone1_clean'] = df['Phone 1'].replace(r'\D+', '', regex=
   True)
df['Phone2_clean'] = df['Phone 2'].replace(r'\D+', '', regex=
   True)
```

I2. Extract email domains and count top 10.

```
df['Domain'] = df['Email'].str.split('@').str[1]
df['Domain'].value_counts().head(10)
```

I3. Extract TLD and count frequency.

```
df['TLD'] = df['Domain'].str.split('.').str[-1]
df['TLD'].value_counts()
```

I4. Group customers by country.

```
counts = df['Country'].value_counts()
percentage = (counts / len(df)) * 100
counts, percentage
```

I5. Customers per year.

```
df['Year'] = df['Subscription Date'].dt.year
df['Year'].value_counts().sort_index()
```

I6. Detect duplicate customers.

By email:

```
df[df.duplicated('Email', keep=False)]
```

By phone:

```
df[df.duplicated('Phone1_clean', keep=False)]
```

By Name + Phone:

```
df[df.duplicated(['First Name', 'Last Name', 'Phone1_clean'],
    keep=False)]
```

I7. Longest and shortest company names.

```
df['company_length'] = df['Company Name'].str.len()
df.loc[df['company_length'].idxmax()]
df.loc[df['company_length'].idxmin()]
df['company_length'].mean()
```

I8. Count customers without Phone 2.

```
df['Phone 2'].isnull().sum()
```

I9. Find customers with phone extensions.

```
ext_customers = df[df['Phone 1'].str.contains('x|ext', case=
   False, na=False)]
ext_customers
```

I10. Monthly signup trend.

```
df['Month'] = df['Subscription Date'].dt.to_period('M')
df['Month'].value_counts().sort_index()
```

I11. Outlier detection using IQR.

```
dates = df['Subscription Date'].dropna()
Q1 = dates.quantile(0.25)
Q3 = dates.quantile(0.75)
IQR = Q3 - Q1
low = Q1 - 1.5 * IQR
high = Q3 + 1.5 * IQR
outliers = df[(df['Subscription Date'] < low) | (df['
   Subscription Date'] > high)]
outliers
```

Hard Task Solutions

H1. Standardize phone numbers into international format.

```
import re

def parse_phone(p):
    if pd.isna(p): return ("", "", "")
    ext = ""
    if 'x' in p.lower():
        parts = re.split('x', p, flags=re.IGNORECASE)
        p, ext = parts[0], parts[1]
    digits = re.sub(r'\D+', '', p)
    return ("+1", digits, ext)  # assuming default country
        code

parsed = df['Phone 1'].apply(parse_phone)
df['CountryCode'], df['MainNumber'], df['Extension'] = zip(*
    parsed)
```

H2. Identify inconsistent country names.

```
unique = df['Country'].unique()
unique
# Then manually map corrections
```

H3. Customer segmentation with K-Means.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans

enc = LabelEncoder()
df['Country_enc'] = enc.fit_transform(df['Country'])
df['Company_len'] = df['Company Name'].str.len()
df['Month'] = df['Subscription Date'].dt.month

X = df[['Country_enc', 'Company_len', 'Month']].dropna()

kmeans = KMeans(n_clusters=4)
df['Cluster'] = kmeans.fit_predict(X)
```

H4. Mini EDA report components.

```
df['Country'].value_counts()
df['Month'].value_counts().sort_index()
df['Domain'].value_counts().head(10)
outliers.shape[0]
```

H5. Correlation-style encoded analysis.

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
encoded = df[['Country_enc', 'Company_len', 'Month']].corr()
sns.heatmap(encoded, annot=True)
plt.show()
```

H6. Prediction model for country.

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

X = df[['Company_len', 'Month']]
y = df['Country_enc']

X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2)

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

H7. Mini CRM dashboard code example.

```
import matplotlib.pyplot as plt

df['Country'].value_counts().head(10).plot(kind='bar')
plt.title("Top 10 Countries")
plt.show()
```

H8. Phone number parsing and pattern analysis.

```
df['NumLength'] = df['Phone1_clean'].str.len()
df['NumLength'].value_counts()
```

H9. Isolation Forest anomaly detection.

```
from sklearn.ensemble import IsolationForest

X = df[['Company_len', 'Month']].fillna(0)
iso = IsolationForest()
df['Anomaly'] = iso.fit_predict(X)
```

H10. Full data-cleaning pipeline.

```
# Load
df = pd.read_csv('customers-10000.csv')

# Fix dates
df['Subscription Date'] = pd.to_datetime(df['Subscription
   Date'], errors='coerce')

# Fill missing phone numbers
df['Phone 2'] = df['Phone 2'].fillna('')
```

```
# Clean strings
df['Company_len'] = df['Company Name'].str.len()

# Extract email domain
df['Domain'] = df['Email'].str.split('@').str[1]

# Export cleaned file
df.to_csv('cleaned_customers.csv', index=False)
```