


Healthcare Project - Data and Transformations

Four tables with data for this project: claims, patients, providers, and procedures.

```
import pandas as pd
```

▼ 1. Loading data

```
# Table 1: Claims data
path1 = "C:/Users/rvrei/Documents/claims.csv"
claims_df = pd.read_csv(path1)
claims_df.head()
```



	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	pr
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
1	CLM0002	PAT0002	15	2024-02-06	4763.43	approved	Blue Shield	Lab Test	
2	CLM0003	PAT0003	38	2023-02-25	3713.57	denied	Blue Shield	X-Ray	
3	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	Blue Shield	Lab Test	
4	CLM0005	PAT0005	46	2023-02-17	948.89	pending	Cigna	MRI	

```
# Table 2: Patients data
path2 = "C:/Users/rvrei/Documents/patients.csv"
patients_df = pd.read_csv(path2)
patients_df.head()
```



patient_id	age	gender	risk_factor	admission_date	readmission
------------	-----	--------	-------------	----------------	-------------

0	PAT0001	74	M	low	2023-02-01	Yes
1	PAT0002	29	M	low	2024-02-24	No
2	PAT0003	80	F	low	2022-12-05	No
3	PAT0004	25	M	medium	2023-03-08	Yes
4	PAT0005	67	F	medium	2023-02-17	No

```
# Table 3: Providers data
```

```
path3 = "C:/Users/rvrei/Documents/providers.csv"
```

```
providers_df = pd.read_csv(path3)
```

```
providers_df.head()
```



	provider_id	specialty	claims_handled
0	PRV001	Cardiology	98
1	PRV002	General Medicine	58
2	PRV003	Orthopedics	90
3	PRV004	Radiology	58
4	PRV005	Neurology	46

```
# Table 4: Procedures data
```

```
path4 = "C:/Users/rvrei/Documents/procedures.csv"
```

```
procedures_df = pd.read_csv(path4)
```

```
procedures_df.head()
```



	procedure_id	revenue	procedure_cost	time_spent_on_patient_care	time_spent_on_administration	doctor_id
0	48	7545.37	4017.55	161	57	10
1	22	2100.00	1100.00	22	50	5

1	22	6400.81	14424.00	90	58	5
2	15	17098.44	4768.25	41	30	8
3	40	17746.07	3873.94	136	55	10
4	13	15775.19	7607.74	132	30	3

▼ 2. Checking for Missing Values

Checking missing values for all four tables.

```
# Create a dictionary to map names to DataFrames
data_dict = {
    "claims_df": claims_df,
    "patients_df": patients_df,
    "providers_df": providers_df,
    "procedures_df": procedures_df
}

# Iterate through the dictionary to check for missing values
for name, df in data_dict.items():
    if df.isnull().sum().all() == 0: # Check if there are no missing values
        print(f"There are no missing values in: {name}")

        There are no missing values in: claims_df
        There are no missing values in: patients_df
        There are no missing values in: providers_df
        There are no missing values in: procedures_df

# Verify there is no missing values in the "claims_df" dataframe
claims_df.isnull().sum()

claim_id      0
patient id    0
```

```
procedure_id      0
claim_date        0
claim_amount      0
claim_status      0
insurance_provider 0
procedure_type    0
provider_id       0
dtype: int64
```

▼ 3. Checking for Duplicates

```
# Create a dictionary to map names to DataFrames
data_dict = {
    "claims_df": claims_df,
    "patients_df": patients_df,
    "providers_df": providers_df,
    "procedures_df": procedures_df
}

# Iterate through the dictionary to check for duplicates
for name, df in data_dict.items():
    if df.duplicated().sum().all() == 0: # Check if there are no duplicates
        print(f"There are no duplicates in {name}")

        There are no duplicates in claims_df
        There are no duplicates in patients_df
        There are no duplicates in providers_df
        There are no duplicates in procedures_df

# Verify there is no duplicates in the "claims_df" dataframe
claims_df.duplicated().sum()

0
```

▼ 4. Outlier Detection

Identify and handle outliers in numerical columns using statistical methods, such as the IQR method or z-scores.

▼ 4.1. Outliers with the z-scores method

```
# Detecting outliers with the z-score method for the "claim_amount" data
from scipy import stats
```

```
z_scores = stats.zscore(claims_df['claim_amount'])
claims_df[(z_scores<3) & (z_scores>-3)].head() # This range of z-scores includes 99.7% of the data
```

	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	pr
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
1	CLM0002	PAT0002	15	2024-02-06	4763.43	approved	Blue Shield	Lab Test	
2	CLM0003	PAT0003	38	2023-02-25	3713.57	denied	Blue Shield	X-Ray	
3	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	Blue Shield	Lab Test	
4	CLM0005	PAT0005	46	2023-02-17	948.89	pending	Cigna	MRI	

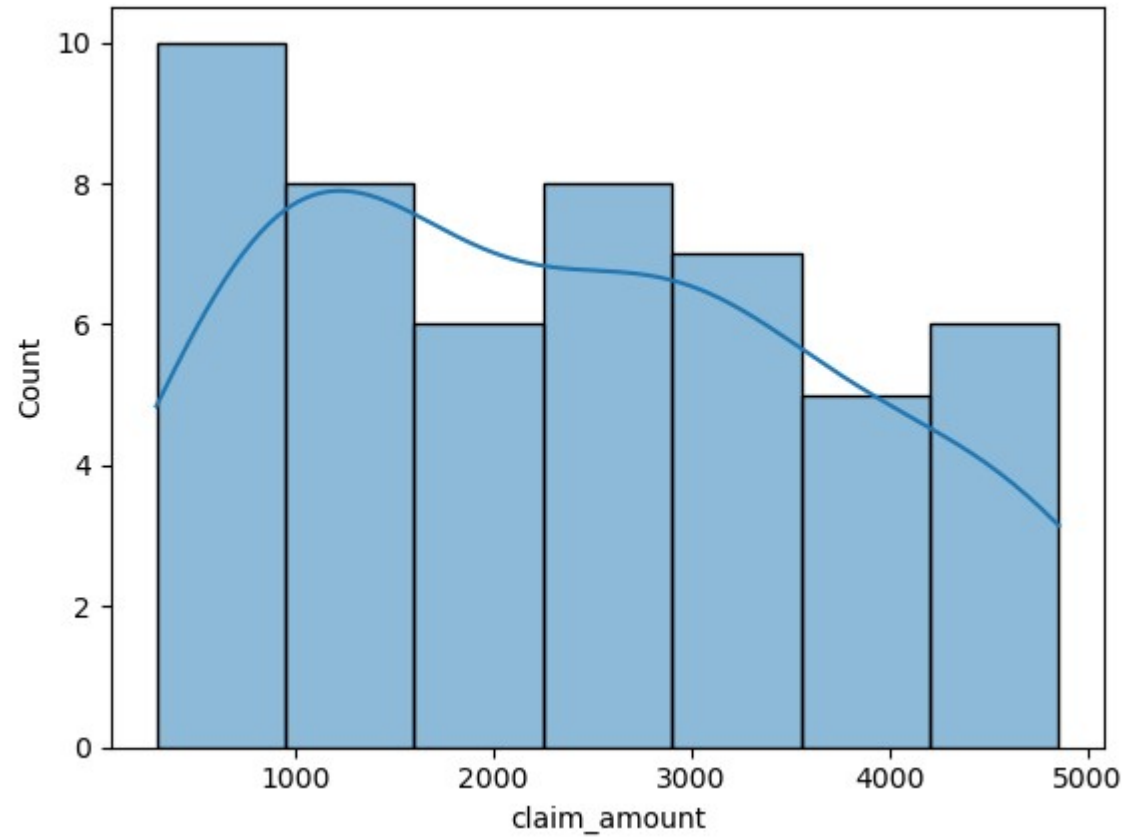
Any data outside of the above range is considered an outlier.

```
# Outlier data
outlier = claims_df[(z_scores>3) & (z_scores<-3)]
outlier
```

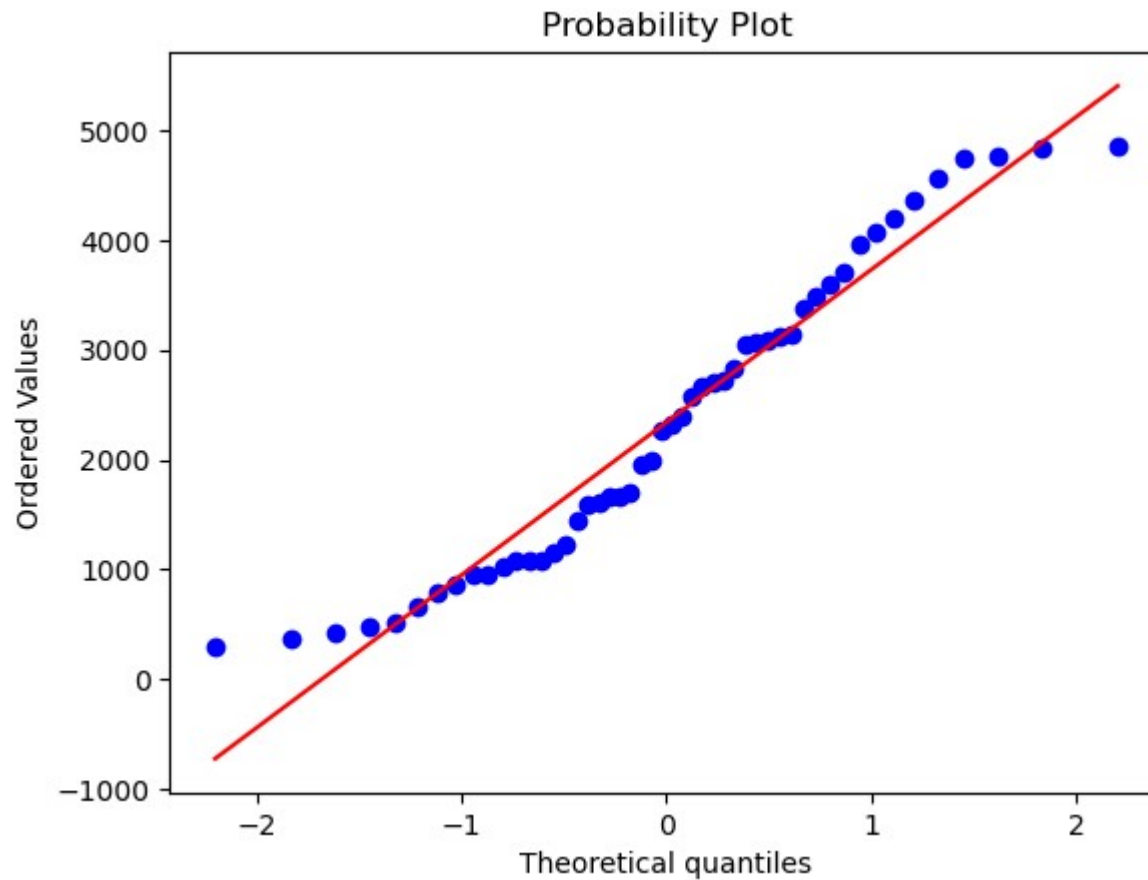
	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	prc
--	----------	------------	--------------	------------	--------------	--------------	--------------------	----------------	-----

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

data = claims_df['claim_amount']
sns.histplot(data, kde=True) # Histogram with a KDE curve
plt.show()
```



```
# Q-Q plot
stats.probplot(data, dist="norm", plot=plt) # Q-Q plot
plt.show()
```



Diagonal line: if the data points lie exactly on this line, the data is perfectly normally distributed.

Points deviate upward or downward at the tails, and the slightly s-shape curve of the data indicate a positive skew of the distribution, longer tail on the right.

✓ 4.2. Outliers with the IQR method

```
import pandas as pd
import numpy as np
import seaborn as sns
```

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

# Calculate Q1, Q3, and IQR
Q1 = claims_df['claim_amount'].quantile(0.25) # 25th percentile
Q3 = claims_df['claim_amount'].quantile(0.75) # 75th percentile
IQR = Q3 - Q1 # Interquartile Range

# Define lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter outliers
filtered_df = claims_df[(claims_df['claim_amount'] >= lower_bound) & (claims_df['claim_amount'] <= upper_bound)]

# Identify outliers
outliers = claims_df[(claims_df['claim_amount'] < lower_bound) | (claims_df['claim_amount'] > upper_bound)]

# Display results
print("Lower Bound:", lower_bound)
print("Upper Bound:", upper_bound)
print("\nFiltered Data (No Outliers):\n", filtered_df.head())
print("\nOutliers:\n", outliers)
```

```
Lower Bound: -2273.7575000000006
Upper Bound: 6675.142500000001
```

```
Filtered Data (No Outliers):
```

	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	\
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	
1	CLM0002	PAT0002	15	2024-02-06	4763.43	approved	
2	CLM0003	PAT0003	38	2023-02-25	3713.57	denied	
3	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	
4	CLM0005	PAT0005	46	2023-02-17	948.89	pending	

	insurance_provider	procedure_type	provider_id
0	Blue Shield	CT Scan	PRV004
1	Blue Shield	Lab Test	PRV002
2	Blue Shield	X-Ray	PRV002
3	Blue Shield	Lab Test	PRV004

4 Cigna MRI PRV008

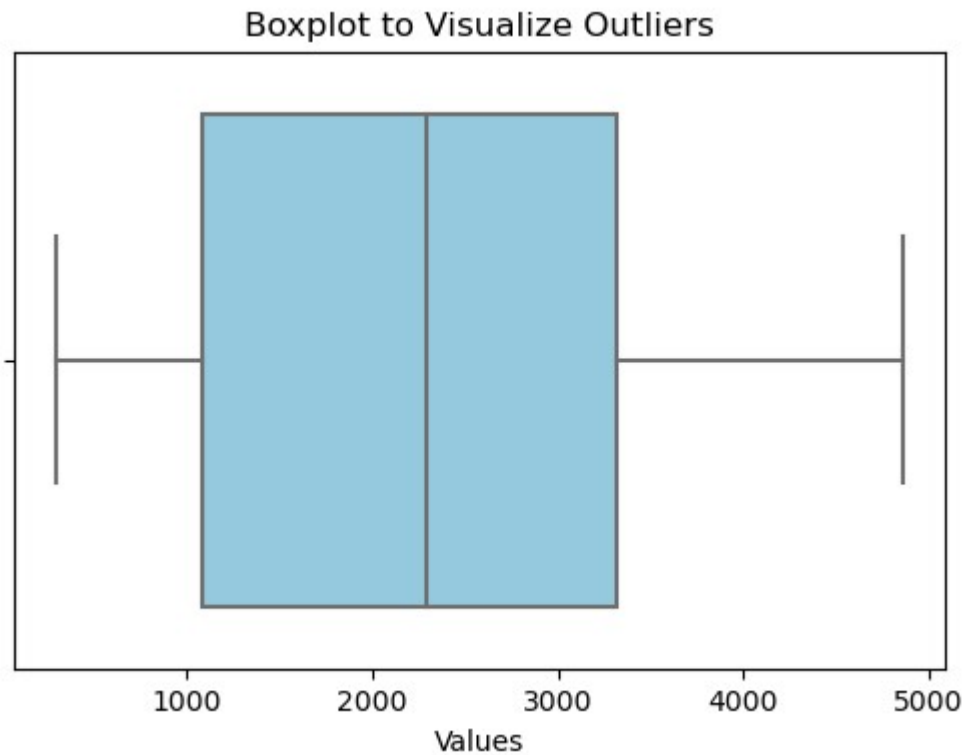
Outliers:

Empty DataFrame

Columns: [claim_id, patient_id, procedure_id, claim_date, claim_amount, claim_status, insurance_provider, procedure_

Index: []

```
# Plot boxplot to visualize outliers
plt.figure(figsize=(6, 4))
sns.boxplot(x=claims_df['claim_amount'], color="skyblue")
plt.title("Boxplot to Visualize Outliers")
plt.xlabel("Values")
plt.show()
```



5. Merging DataFrames

Combine all four dataframes into one for future analysis: "healthcare_df"

```
# Merge claims_df and patients_df into merged_1
merged_1 = pd.merge(claims_df, patients_df, on='patient_id', how='inner')
merged_1.head()
```

	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	pr
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
1	CLM0002	PAT0002	15	2024-02-06	4763.43	approved	Blue Shield	Lab Test	
2	CLM0003	PAT0003	38	2023-02-25	3713.57	denied	Blue Shield	X-Ray	
3	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	Blue Shield	Lab Test	
4	CLM0005	PAT0005	46	2023-02-17	948.89	pending	Cigna	MRI	

```
# Merge merged_1 and providers_df into merged_2
merged_2 = pd.merge(merged_1, providers_df, on='provider_id', how='inner')
merged_2.head()
```

	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	pr
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
1	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	Blue Shield	Lab Test	
2	CLM0011	PAT0011	19	2022-12-26	298.81	pending	United Healthcare	Lab Test	
3	CLM0040	PAT0040	22	2024-09-05	2312.73	approved	Aetna	Lab Test	
4	CLM0042	PAT0042	3	2023-08-08	2576.85	denied	Blue Shield	Lab Test	

```
# Merge merged_2 and procedures_df into healthcare_df:
healthcare_df = pd.merge(merged_2, procedures_df, on='procedure_id', how='inner')
healthcare_df.head()
```

	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	pr
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
1	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
2	CLM0016	PAT0016	41	2023-09-16	1080.34	approved	Aetna	MRI	
3	CLM0016	PAT0016	41	2023-09-16	1080.34	approved	Aetna	MRI	
4	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	Blue Shield	Lab Test	

5 rows × 21 columns

```
healthcare_df.shape
```

(47, 21)

```
# Save the DataFrame to a CSV file
healthcare_df.to_csv('healthcare_df.csv', index=False)
```

```
print("DataFrame saved as 'healthcare_df.csv'")
```

DataFrame saved as 'healthcare_df.csv'

```
# Statistical Summary of the dataframe
round(healthcare_df.describe(),2)
```

	procedure_id	claim_amount	age	claims_handled	revenue	procedure_cost	time_spent_on_patient_care	time_!
count	47.00	47.00	47.00	47.00	47.00	47.00		47.00

mean	24.17	2205.03	46.06	61.19	14053.87	8245.10	101.17
std	13.84	1426.14	19.09	16.07	3894.54	3173.26	41.49
min	1.00	298.81	19.00	46.00	6400.81	3344.40	40.00
25%	14.50	785.78	28.00	47.00	11055.26	5573.21	67.00
50%	22.00	2312.73	48.00	58.00	14752.16	7989.29	96.00
75%	35.00	3073.56	60.00	61.00	17473.98	10363.37	130.50
max	50.00	4835.03	74.00	98.00	19949.47	14424.00	177.00