Healthcare Fraud Detection

Overview

In this project, we will delve into anomaly detection within healthcare data, focusing on identifying unusual patterns or outliers that could signify potential health issues, fraudulent activities, or errors in the data. Through a comprehensive Exploratory Data Analysis (EDA) process, we aim to uncover hidden anomalies and understand their significance in the healthcare domain. By leveraging data visualization techniques and robust analysis methodologies, we seek to extract actionable insights that can aid in early detection, prevention, and intervention strategies. Our ultimate goal is to discern the factors influencing anomalies within healthcare data and develop effective anomaly detection models to safeguard patient health, ensure data integrity, and optimize healthcare operations.

Dataset Info

The dataset is an imaginary collection of healthcare fraud records, comprising 1,000,000 entries. It features fields such as Provider ID, Claim ID, Patient ID, Diagnosis Code, Claim Amount, Paid Amount, and more. Values for Claim Amount, Paid Amount, Coverage Amount, and Total Charges are generated randomly between 100 and 10,000.

Importing Libraries

In [9]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Reading the dataset

In [10]:
```
df = pd.read_csv('Healthcare_Fraud_Imaginary_Asia_Dataset.csv')
```

In [11]:
```
df.shape
```

Out[11]:
```
(1000000, 29)
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 29 columns):
 #   Column                Non-Null Count    Dtype  
---  ------                --------------    -----  
 0   Provider_ID           1000000 non-null  object 
 1   Claim_ID              1000000 non-null  object 
 2   Patient_ID            1000000 non-null  object 
 3   Diagnosis_Code        1000000 non-null  object 
 4   Procedure_Code        1000000 non-null  object 
 5   Claim_Date            1000000 non-null  object 
 6   Admission_Date        1000000 non-null  object 
 7   Discharge_Date        1000000 non-null  object 
 8   Claim_Amount          1000000 non-null  float64
 9   Paid_Amount           1000000 non-null  float64
 10  Provider_Specialty    1000000 non-null  object 
 11  Patient_Age           1000000 non-null  int64  
 12  Patient_Gender        1000000 non-null  object 
 13  Fraud_Label           1000000 non-null  int64  
 14  Investigation_Details 1000000 non-null  object 
 15  Policy_Type           1000000 non-null  object 
 16  Coverage_Amount       1000000 non-null  float64
 17  Total_Charges         1000000 non-null  float64
 18  Payment_Type          1000000 non-null  object 
 19  State                 1000000 non-null  object 
 20  Email                 1000000 non-null  object 
 21  Phone_Number          1000000 non-null  object 
 22  Address               1000000 non-null  object 
 23  Nationality           1000000 non-null  object 
 24  Passport_Number       1000000 non-null  object 
 25  Employer              1000000 non-null  object 
 26  Occupation            1000000 non-null  object 
 27  Marital_Status        1000000 non-null  object 
 28  Education_Level       1000000 non-null  object 
dtypes: float64(4), int64(2), object(23)
memory usage: 221.3+ MB
```

In [13]:
```
df.head()
```

Out[13]:

	Provider_ID	Claim_ID	Patient_ID	Diagnosis_Code	Procedure_Code	Claim_Date	Admission_Date	Discharge_Date	Claim_Amount	Paid_A
0	Asian Medical Center	CLAIM_1	Darrell Blair	DX_714	PROC_2648	2024-04-24	2024-03-26	2024-05-08	1077.86	4
1	Sky Hospital	CLAIM_2	William Young	DX_885	PROC_9084	2024-04-24	2024-04-07	2024-05-03	4998.88	5
2	Moon Healthcare	CLAIM_3	Keith Reynolds	DX_988	PROC_9747	2024-04-24	2024-04-01	2024-05-24	7058.21	8
3	Sky Hospital	CLAIM_4	Andre Kelly	DX_779	PROC_4334	2024-04-24	2024-03-31	2024-04-27	1628.67	8
4	Sun Clinic	CLAIM_5	Terry Gonzales	DX_644	PROC_8408	2024-04-24	2024-03-27	2024-05-12	1480.43	4

In [14]:
```
df.tail()
```

Out[14]:

	Provider_ID	Claim_ID	Patient_ID	Diagnosis_Code	Procedure_Code	Claim_Date	Admission_Date	Discharge_Date	Claim_Amo
999995	Asian Medical Center	CLAIM_999996	Evelyn Rivers	DX_590	PROC_1543	2024-04-24	2024-03-25	2024-05-12	9576
999996	Moon Healthcare	CLAIM_999997	Robert Woods	DX_954	PROC_6870	2024-04-24	2024-03-26	2024-05-09	4600
999997	Sun Clinic	CLAIM_999998	David Thomas	DX_302	PROC_9405	2024-04-24	2024-03-25	2024-05-20	7103
999998	Asian Medical Center	CLAIM_999999	Samantha Hubbard	DX_517	PROC_4916	2024-04-24	2024-04-05	2024-05-21	313
999999	Sun Clinic	CLAIM_1000000	Amy Edwards	DX_699	PROC_7914	2024-04-24	2024-03-29	2024-05-08	1540

In [15]:
```
df.describe()
```

Out[15]:

	Claim_Amount	Paid_Amount	Patient_Age	Fraud_Label	Coverage_Amount	Total_Charges
count	1000000.000000	1000000.000000	1000000.000000	1000000.000000	1000000.000000	1000000.000000
mean	5049.924917	5023.608723	53.988752	0.499551	2999.730097	5047.055361
std	2856.368535	2870.834799	21.079818	0.500000	1154.205850	2859.124481
min	100.020000	50.050000	18.000000	0.000000	1000.000000	100.030000
25%	2577.415000	2538.360000	36.000000	0.000000	2000.050000	2568.770000
50%	5051.705000	5023.160000	54.000000	0.000000	2999.905000	5045.890000
75%	7522.192500	7509.830000	72.000000	1.000000	3998.232500	7524.647500
max	9999.990000	10000.000000	90.000000	1.000000	5000.000000	9999.990000

Exploratory Data Analysis (EDA)

Missing Values

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
Provider_ID              0
Claim_ID                 0
Patient_ID               0
Diagnosis_Code           0
Procedure_Code           0
Claim_Date               0
Admission_Date           0
Discharge_Date           0
Claim_Amount             0
Paid_Amount              0
Provider_Specialty       0
Patient_Age              0
Patient_Gender           0
Fraud_Label              0
Investigation_Details    0
Policy_Type              0
Coverage_Amount          0
Total_Charges            0
Payment_Type             0
State                    0
Email                    0
Phone_Number             0
Address                  0
Nationality              0
Passport_Number          0
Employer                 0
Occupation               0
Marital_Status           0
Education_Level          0
dtype: int64
```

Duplicate Values

In [9]:
```
df.duplicated().any()
```

Out[9]:
```
False
```

Constant Values

In [10]:
```
constant_columns = df.columns[df.nunique() == 1]
constant_columns
```

Out[10]:
```
Index(['Claim_Date'], dtype='object')
```

Conclusion:- Our dataset has not any null/missing/duplicate values but contains a column having constant value.

So, removing the column 'Claim_Date' from the dataset.

In [11]:
```
df = df.drop('Claim_Date', axis=1)
```

In [12]:
```
df.shape
```

Out[12]:
```
(1000000, 28)
```

Correlation Matrix Representation

In [13]:
```
df_corr = df.select_dtypes(include=['number']).corr()
df_corr
```

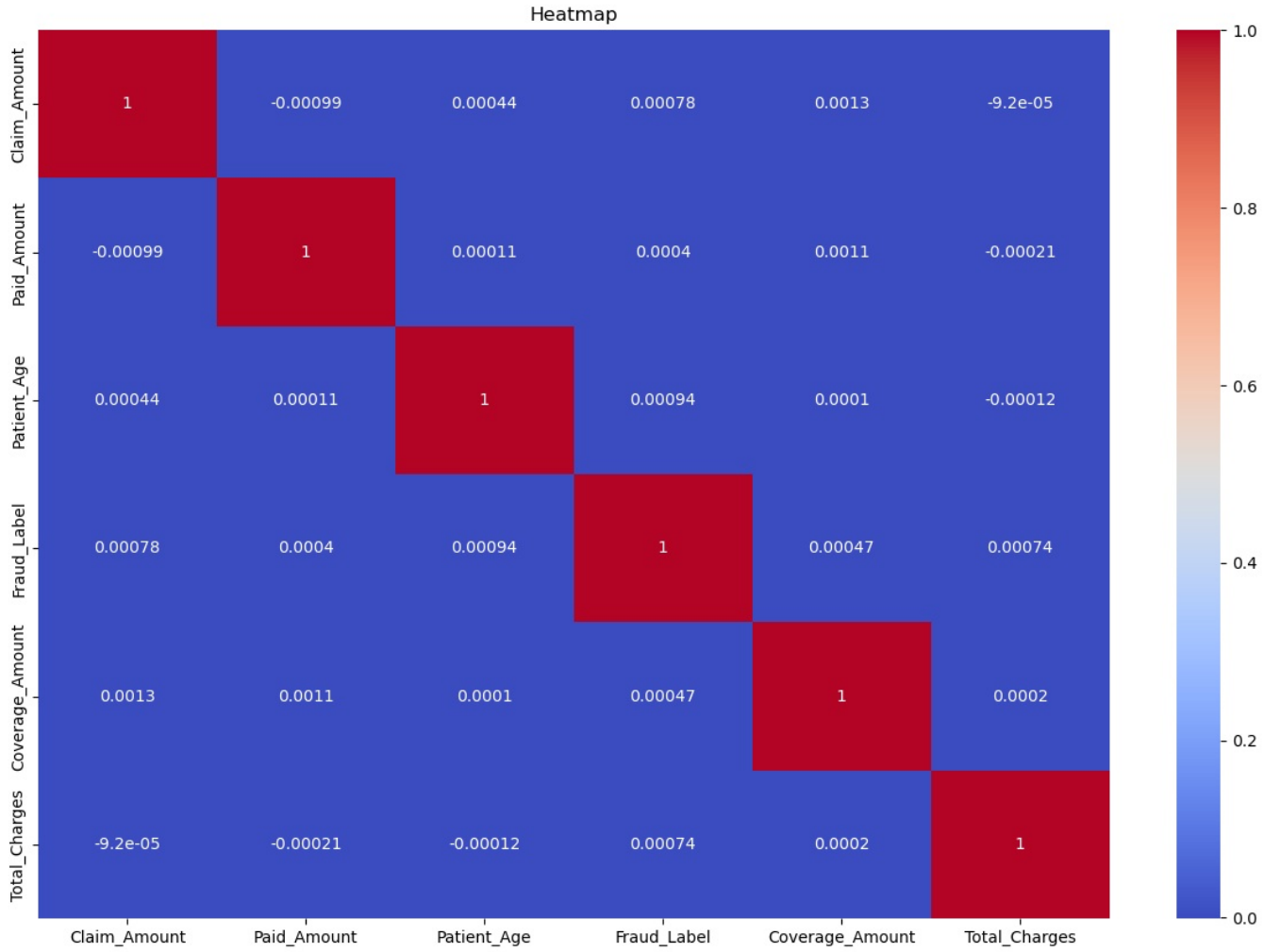
Out[13]:

	Claim_Amount	Paid_Amount	Patient_Age	Fraud_Label	Coverage_Amount	Total_Charges
Claim_Amount	1.000000	-0.000991	0.000442	0.000777	0.001259	-0.000092
Paid_Amount	-0.000991	1.000000	0.000114	0.000399	0.001060	-0.000206
Patient_Age	0.000442	0.000114	1.000000	0.000938	0.000100	-0.000122
Fraud_Label	0.000777	0.000399	0.000938	1.000000	0.000471	0.000738
Coverage_Amount	0.001259	0.001060	0.000100	0.000471	1.000000	0.000197
Total_Charges	-0.000092	-0.000206	-0.000122	0.000738	0.000197	1.000000

Drawing Heatmap

In [14]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df_corr, annot=True, cmap='coolwarm')
plt.title('Heatmap')
```

Out[14]:
```
Text(0.5, 1.0, 'Heatmap')
```



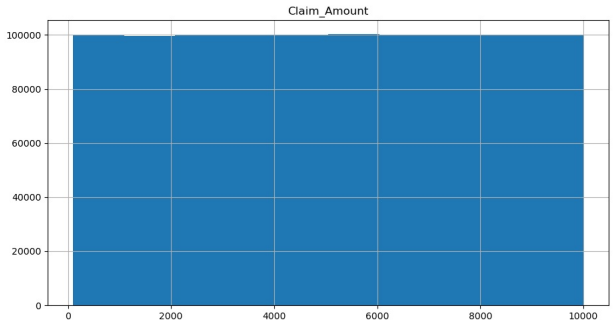
In [15]:
```
df.columns
```

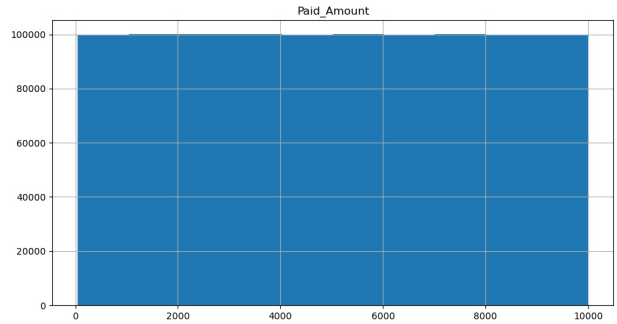
Out[15]:
```
Index(['Provider_ID', 'Claim_ID', 'Patient_ID', 'Diagnosis_Code',
       'Procedure_Code', 'Admission_Date', 'Discharge_Date', 'Claim_Amount',
       'Paid_Amount', 'Provider_Specialty', 'Patient_Age', 'Patient_Gender',
       'Fraud_Label', 'Investigation_Details', 'Policy_Type',
       'Coverage_Amount', 'Total_Charges', 'Payment_Type', 'State', 'Email',
       'Phone_Number', 'Address', 'Nationality', 'Passport_Number', 'Employer',
       'Occupation', 'Marital_Status', 'Education_Level'],
      dtype='object')
```

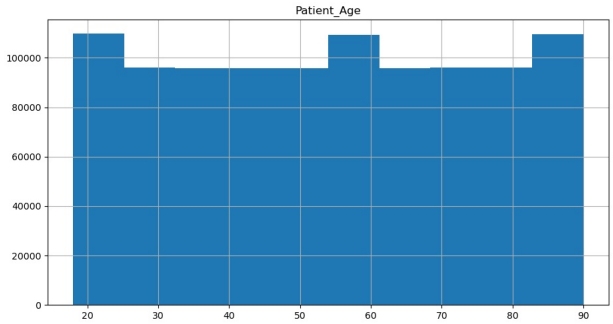
Histogram Representation

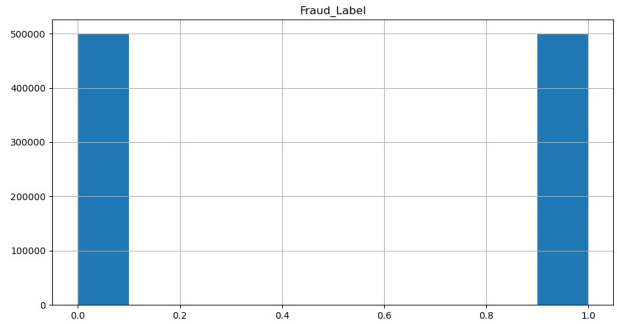
In [16]:
```
df.hist(figsize=(25,20))
```

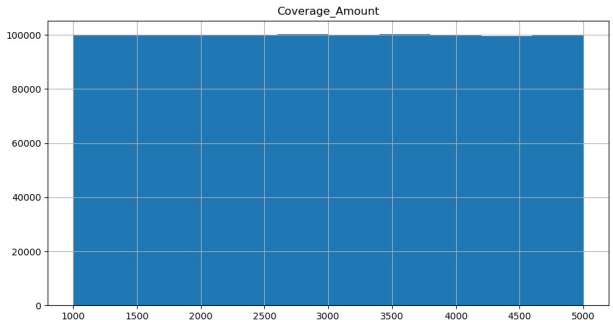
Out[16]:
```
array([[<Axes: title={'center': 'Claim_Amount'}>,
        <Axes: title={'center': 'Paid_Amount'}>],
       [<Axes: title={'center': 'Patient_Age'}>,
        <Axes: title={'center': 'Fraud_Label'}>],
       [<Axes: title={'center': 'Coverage_Amount'}>,
        <Axes: title={'center': 'Total_Charges'}>]], dtype=object)
```

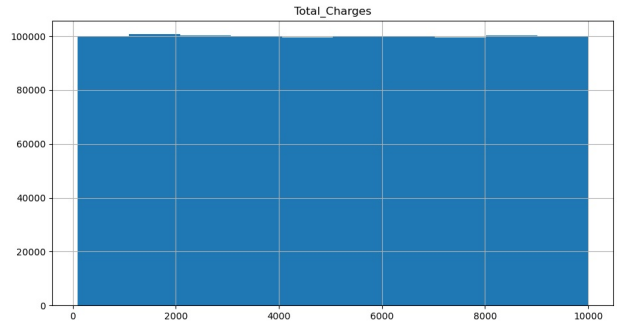






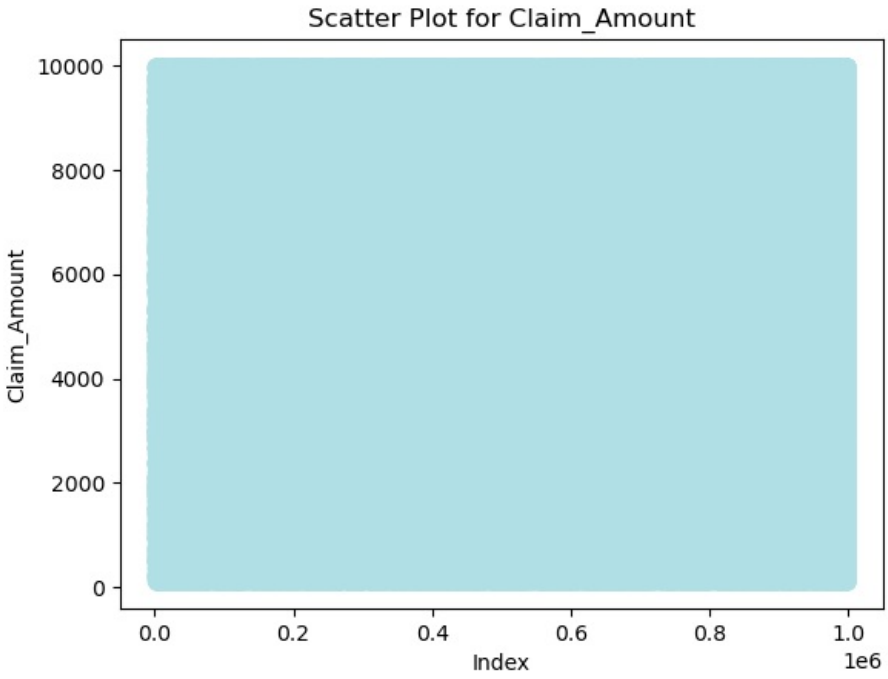


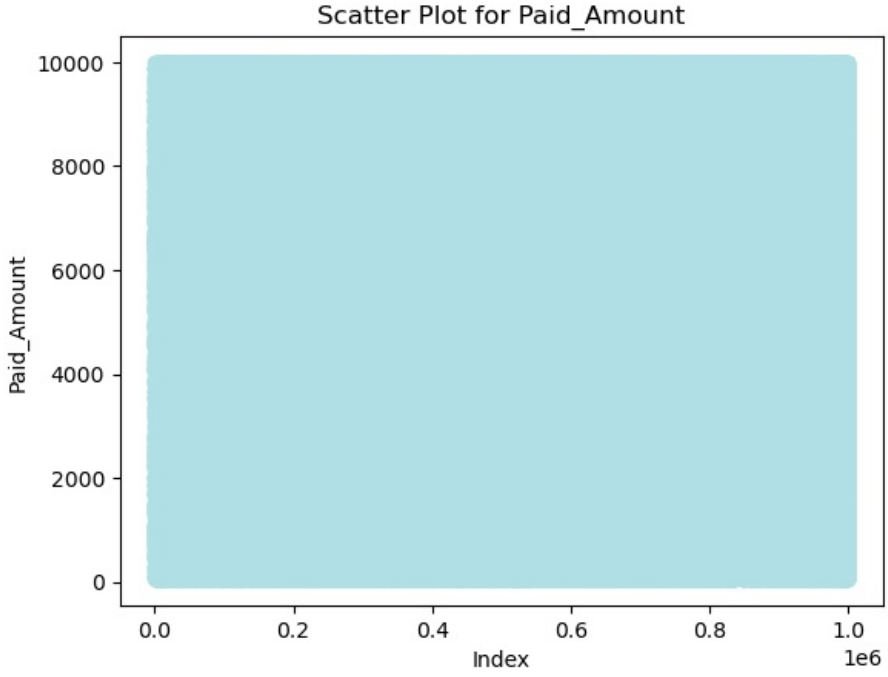




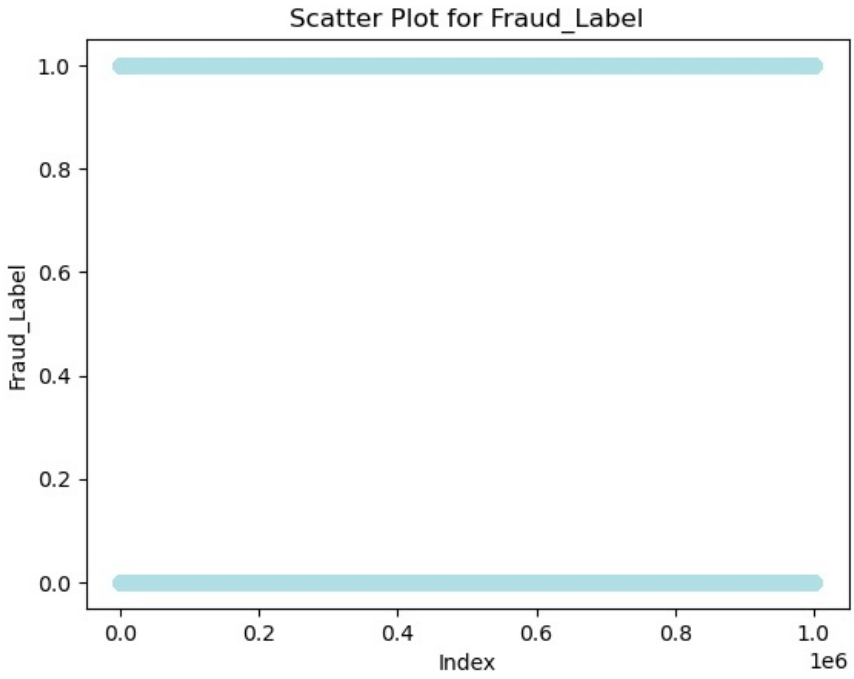
Scatter Plot for all numeric columns

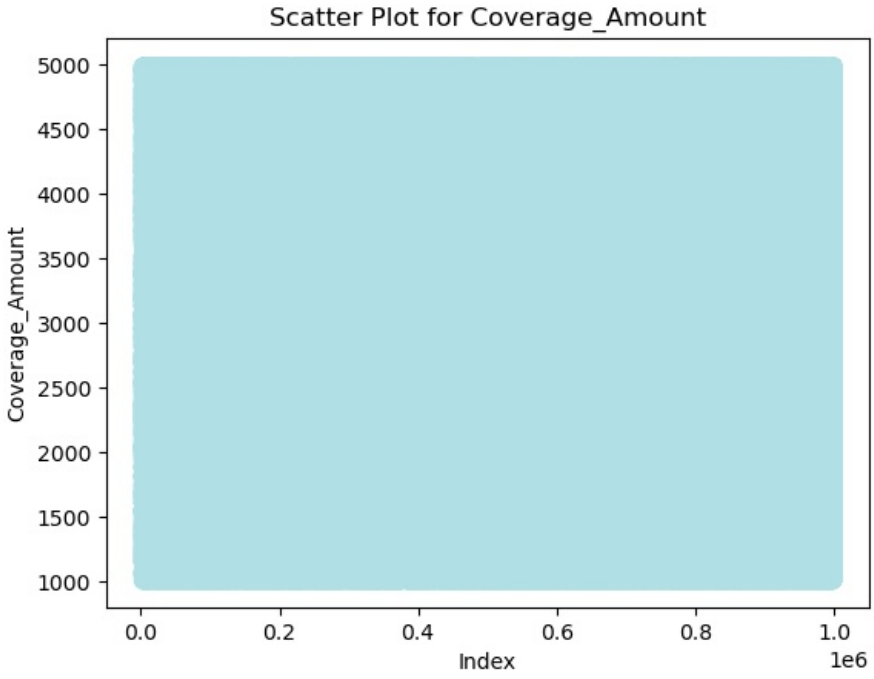
In [17]:
```
numerical_cols = df.select_dtypes(include=[np.number])

for col in numerical_cols.columns:
    plt.figure()
    plt.scatter(df.index, df[col], c='powderblue')
    plt.title(f"Scatter Plot for {col}")
    plt.xlabel("Index")
    plt.ylabel(col)

plt.show()
```

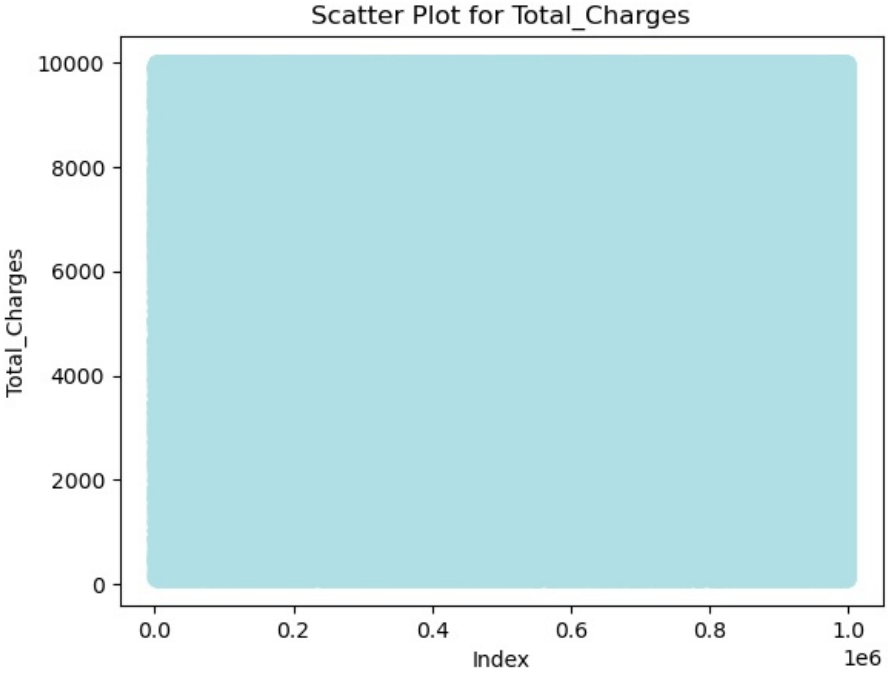

Scatter Plot for Claim_Amount
Claim_Amount
10000
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Index


Scatter Plot for Paid_Amount
Paid_Amount
10000
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Index


Scatter Plot for Patient_Age
Patient_Age
90
80
70
60
50
40
30
20
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Index



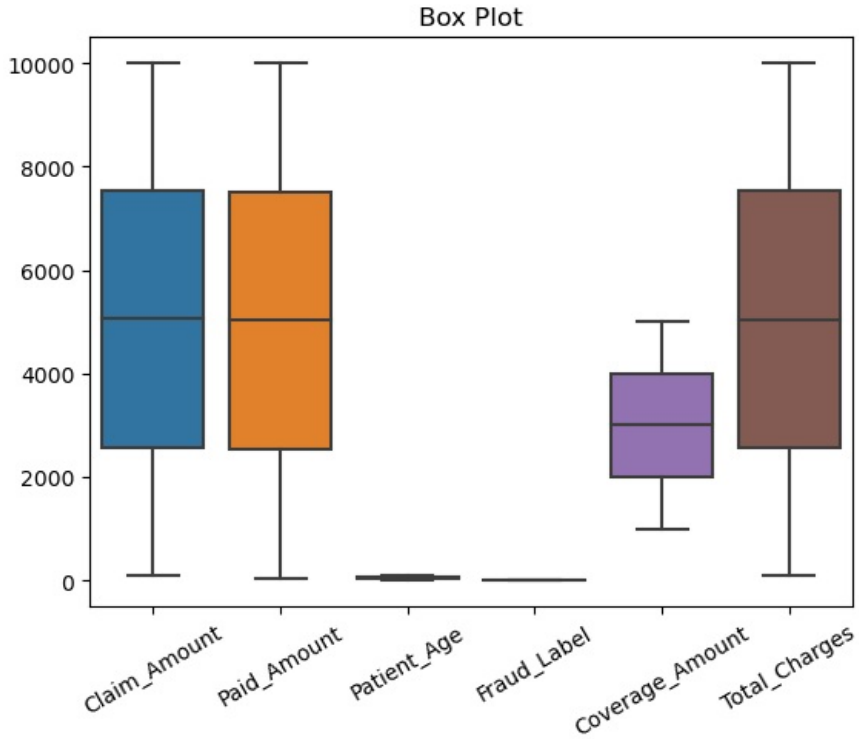




Box Plot Representation

In [18]:
```
sns.boxplot(data=df)
plt.title('Box Plot')
plt.xticks(rotation=30)
```

Out[18]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Claim_Amount'),
  Text(1, 0, 'Paid_Amount'),
  Text(2, 0, 'Patient_Age'),
  Text(3, 0, 'Fraud_Label'),
  Text(4, 0, 'Coverage_Amount'),
  Text(5, 0, 'Total_Charges')])
```



Automated EDA

Using Pandas Profiling (ydata profiling)

from ydata_profiling import ProfileReport

profile = ProfileReport(df) profile.to_file("ProfileReport.html")

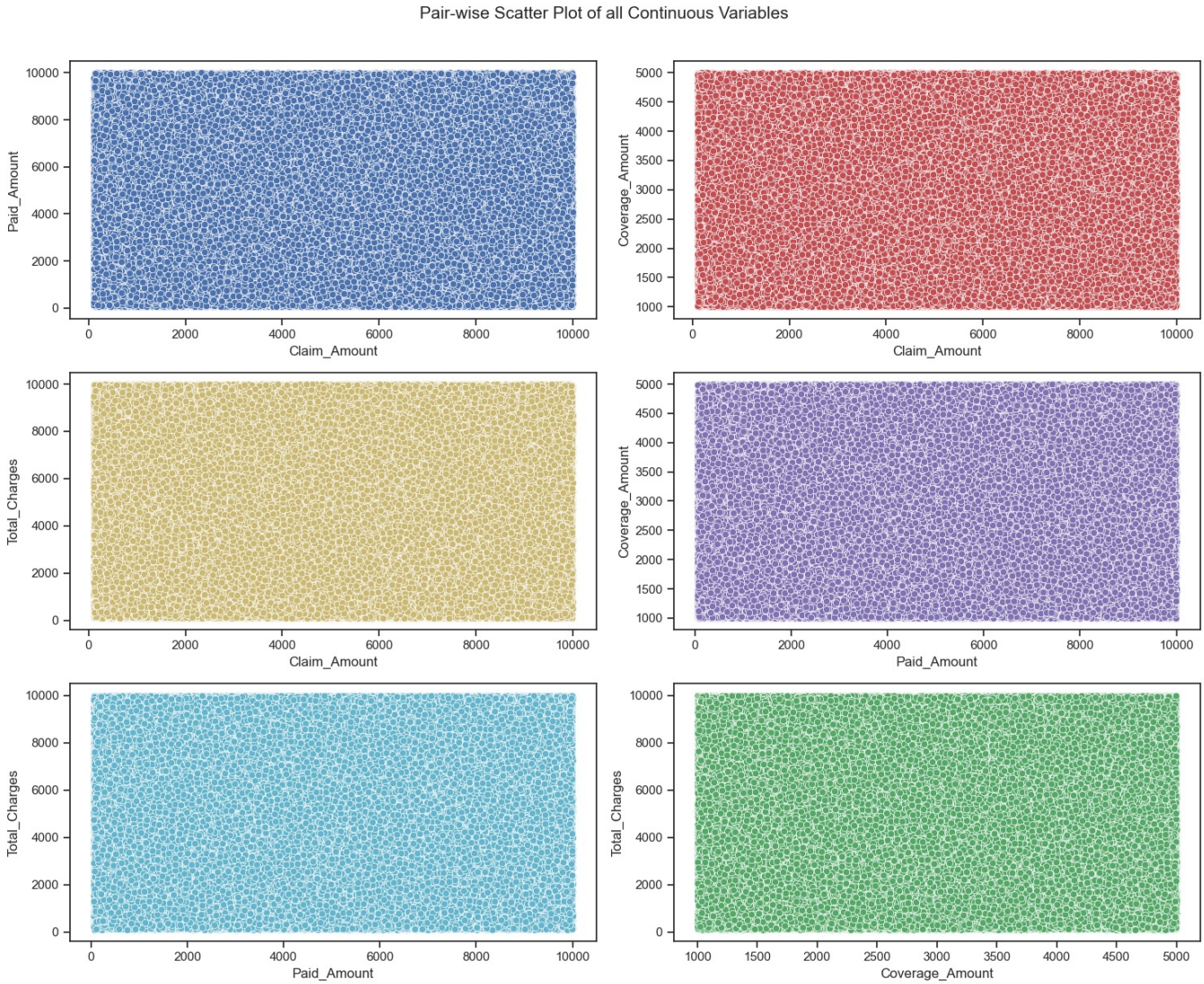
Using AutoViz Library

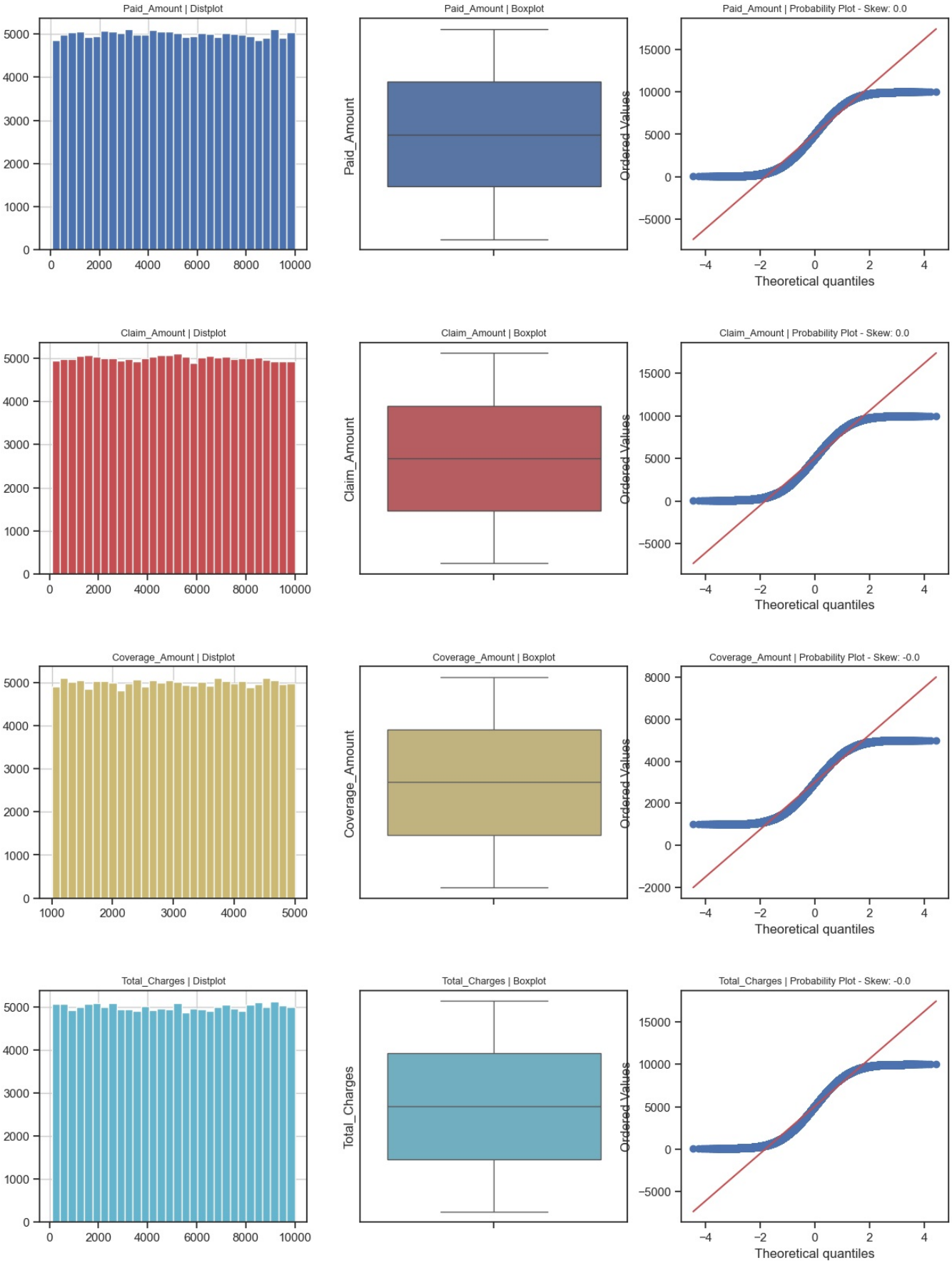
In [8]:
```
%matplotlib inline
from autoviz import AutoViz_Class
AV = AutoViz_Class()

dft = AV.AutoViz(df, chart_format="svg")
```

```
    Since nrows is smaller than dataset, loading random sample of 150000 rows into pandas...
Shape of your Data Set loaded: (150000, 29)
################################################################################################
######################## C L A S S I F Y I N G  V A R I A B L E S  ####################
################################################################################################
Classifying variables in data set...
    Number of Numeric Columns =  4
    Number of Integer-Categorical Columns =  1
    Number of String-Categorical Columns =  13
    Number of Factor-Categorical Columns =  0
    Number of String-Boolean Columns =  2
    Number of Numeric-Boolean Columns =  1
    Number of Discrete String Columns =  1
    Number of NLP String Columns =  3
    Number of Date Time Columns =  0
    Number of ID Columns =  3
    Number of Columns to Delete =  1
    29 Predictors classified...
        4 variable(s) removed since they were ID or low-information variables
        List of variables removed: ['Claim_ID', 'Phone_Number', 'Passport_Number', 'Claim_Date']
Since Number of Rows in data 150000 exceeds maximum, randomly sampling 150000 rows for EDA...
To fix these data quality issues in the dataset, import FixDQ from autoviz...
    All variables classified into correct types.
```

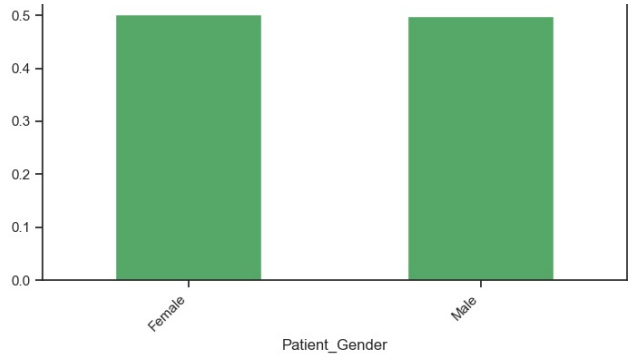
	Data Type	Missing Values%	Unique Values%	Minimum Value	Maximum Value	DQ Issue
Provider_ID	object	0.000000	0			No issue
Claim_ID	object	0.000000	100			Possible ID column: drop before modeling step.
Patient_ID	object	0.000000	64			No issue
Diagnosis_Code	object	0.000000	0			Possible high cardinality column with 900 unique values: Use hash encoding or text embedding to reduce dimension.
Procedure_Code	object	0.000000	6			No issue
Claim_Date	object	0.000000	0			Possible Zero-variance or low information colum: drop before modeling step.
Admission_Date	object	0.000000	0			No issue
Discharge_Date	object	0.000000	0			No issue
Claim_Amount	float64	0.000000	NA	100.080000	9999.940000	No issue
Paid_Amount	float64	0.000000	NA	50.050000	9999.960000	No issue
Provider_Specialty	object	0.000000	0			No issue
Patient_Age	int64	0.000000	0	18.000000	90.000000	No issue
Patient_Gender	object	0.000000	0			No issue
Fraud_Label	int64	0.000000	0	0.000000	1.000000	No issue
Investigation_Details	object	0.000000	0			No issue
Policy_Type	object	0.000000	0			No issue
Coverage_Amount	float64	0.000000	NA	1000.110000	4999.940000	No issue
Total_Charges	float64	0.000000	NA	100.060000	9999.940000	No issue
Payment_Type	object	0.000000	0			No issue
State	object	0.000000	0			No issue
Email	object	0.000000	81			No issue
Phone_Number	object	0.000000	100			Possible ID column: drop before modeling step.
Address	object	0.000000	100			No issue
Nationality	object	0.000000	0			No issue
Passport_Number	object	0.000000	99			149987 rare categories: Too many to list. Group them into a single category or drop the categories.
Employer	object	0.000000	65			No issue
Occupation	object	0.000000	0			No issue
Marital_Status	object	0.000000	0			No issue
Education_Level	object	0.000000	0			No issue

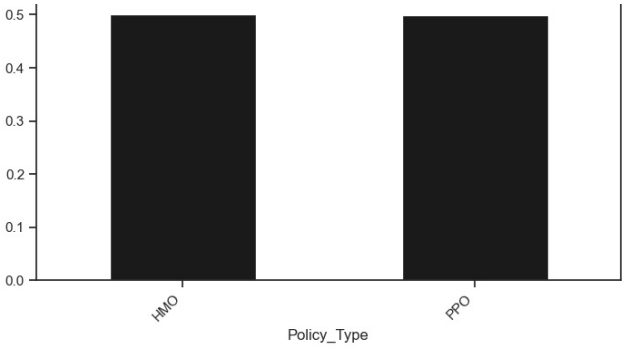
```
Number of All Scatter Plots = 10
```

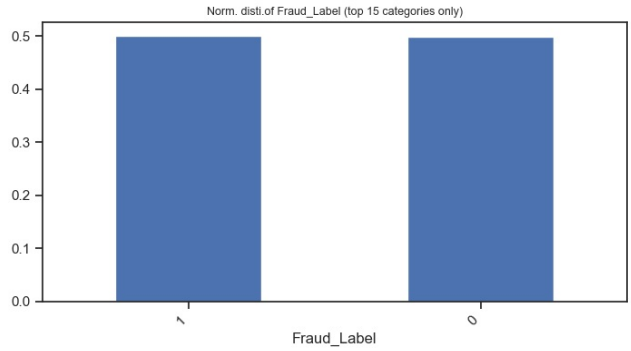

Pair-wise Scatter Plot of all Continuous Variables
Paid_Amount
Claim_Amount
Coverage_Amount
Claim_Amount
Total_Charges
Claim_Amount
Coverage_Amount
Paid_Amount
Total_Charges
Paid_Amount
Total_Charges
Coverage_Amount

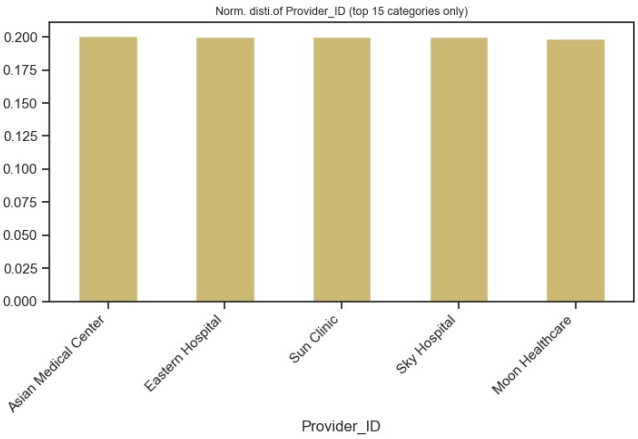


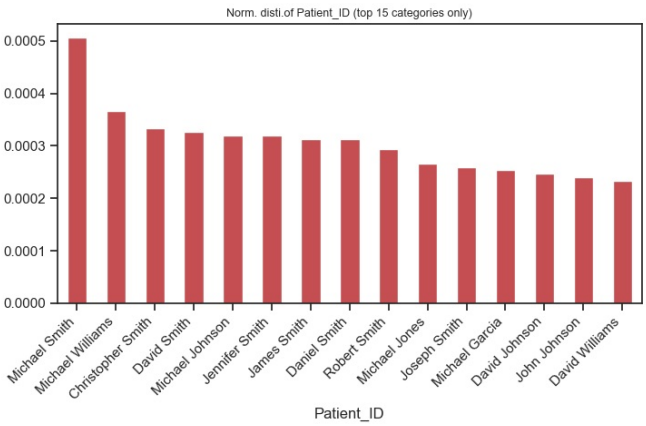
Histograms and Normalized distribitions of all variables

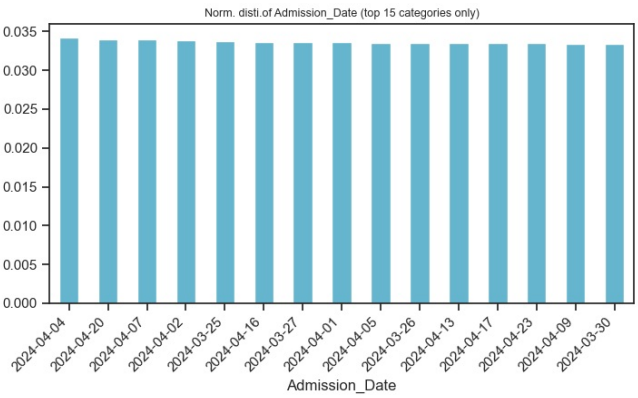

0.5
0.4
0.3
0.2
0.1
0.0
Female
Male
Patient_Gender

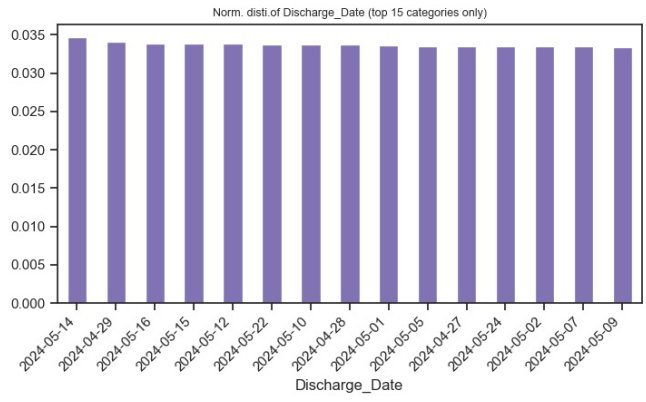

0.5
0.4
0.3
0.2
0.1
0.0
HMO
PPO
Policy_Type

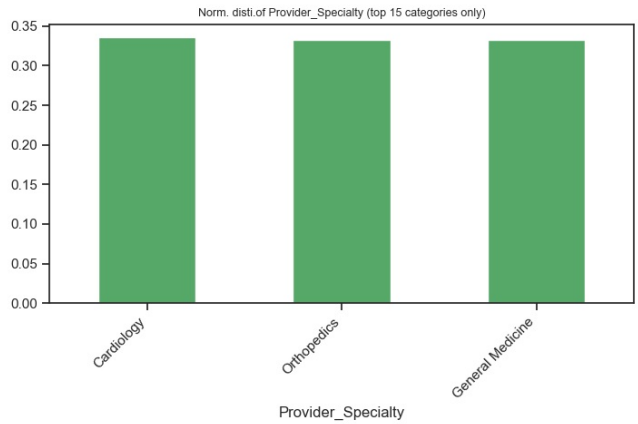

Norm. disti.of Fraud_Label (top 15 categories only)
0.5
0.4
0.3
0.2
0.1
0.0
1
0
Fraud_Label

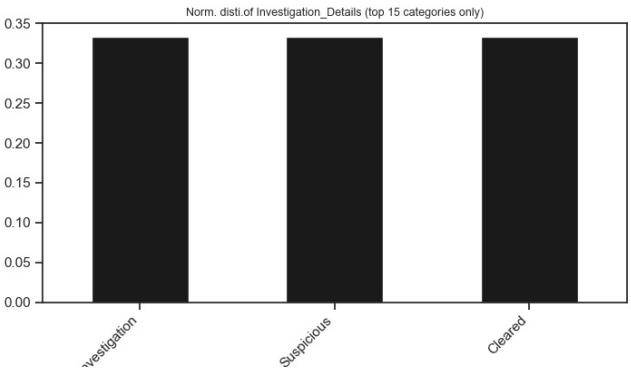

Norm. disti.of Provider_ID (top 15 categories only)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Asian Medical Center
Eastern Hospital
Sun Clinic
Sky Hospital
Moon Healthcare
Provider_ID

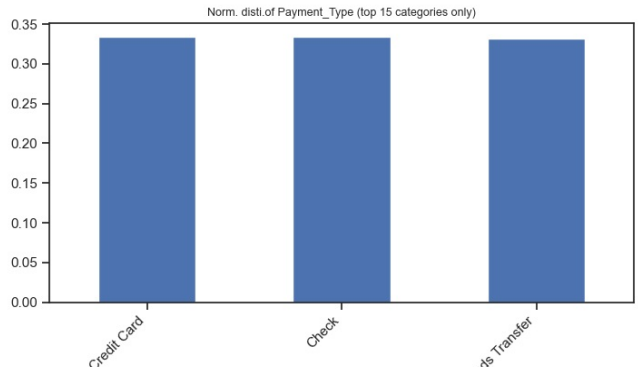

Norm. disti.of Patient_ID (top 15 categories only)
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
Michael Smith
Michael Williams
Christopher Smith
David Smith
Michael Johnson
Jennifer Smith
James Smith
Daniel Smith
Robert Smith
Michael Jones
Joseph Smith
Michael Garcia
David Johnson
John Johnson
David Williams
Patient_ID

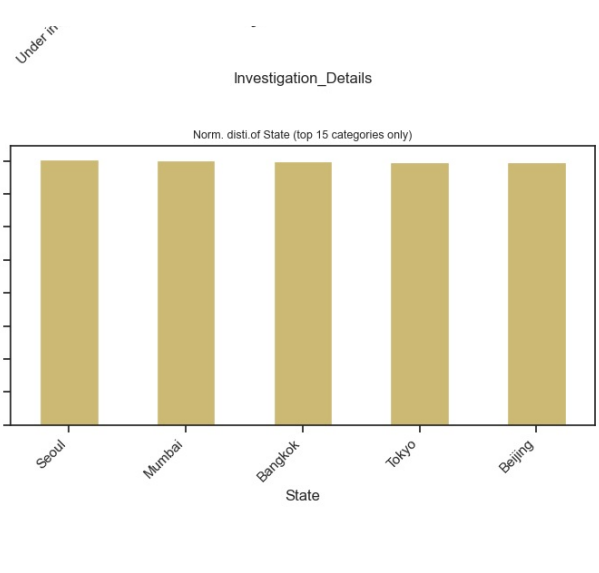

Norm. disti.of Admission_Date (top 15 categories only)
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
2024-04-04
2024-04-20
2024-04-07
2024-04-02
2024-03-25
2024-04-16
2024-03-27
2024-04-01
2024-04-05
2024-03-26
2024-04-13
2024-04-17
2024-04-23
2024-04-09
2024-03-30
Admission_Date

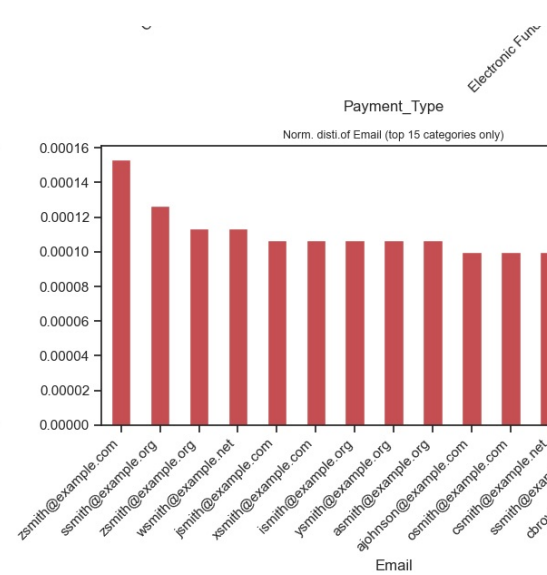

Norm. disti.of Discharge_Date (top 15 categories only)
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
2024-05-14
2024-04-29
2024-05-16
2024-05-15
2024-05-12
2024-05-22
2024-05-10
2024-04-28
2024-05-01
2024-05-05
2024-04-27
2024-05-24
2024-05-02
2024-05-07
2024-05-09
Discharge_Date


Norm. disti.of Provider_Specialty (top 15 categories only)
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Cardiology
Orthopedics
General Medicine
Provider_Specialty


Norm. disti.of Investigation_Details (top 15 categories only)
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Suspicious
Cleared

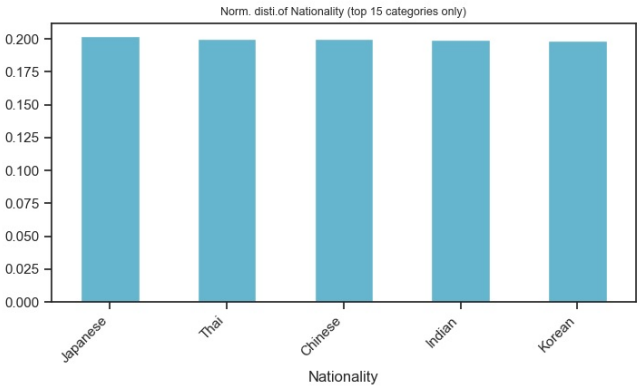

Norm. disti.of Payment_Type (top 15 categories only)
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Check

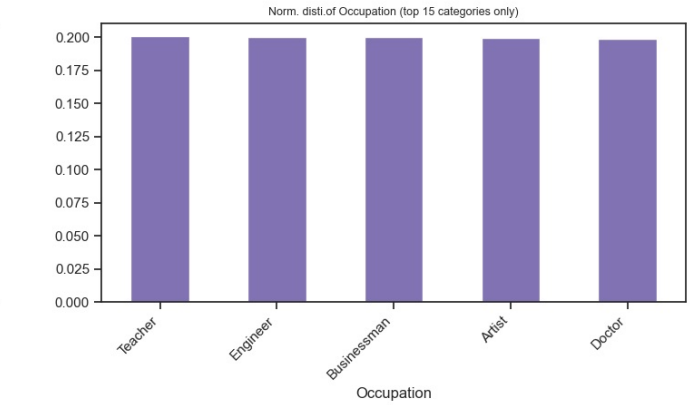

Under in
Investigation_Details

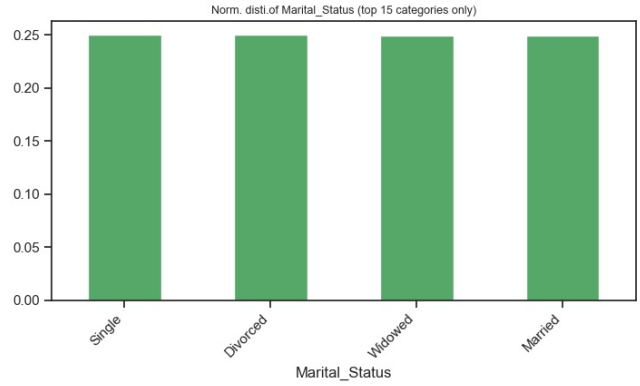

Electronic Fund
Payment_Type

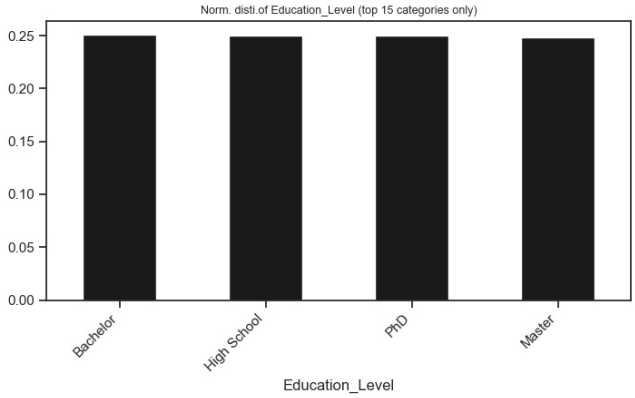
Norm. disti.of State (top 15 categories only)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Seoul
Mumbai
Bangkok
Tokyo
Beijing
State

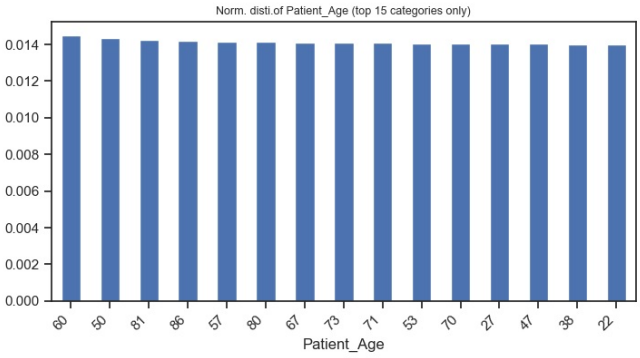
Norm. disti.of Email (top 15 categories only)
0.00016
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
zsmith@example.com
ssmith@example.org
zsmith@example.org
wsmith@example.net
jsmith@example.com
xsmith@example.com
ismith@example.org
ysmith@example.org
asmith@example.org
ajohnson@example.com
osmith@example.com
csmith@example.com
ssmith@example.net
cbrown@example.net
ismith@example.net
Email


Norm. disti.of Nationality (top 15 categories only)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Japanese
Thai
Chinese
Indian
Korean
Nationality

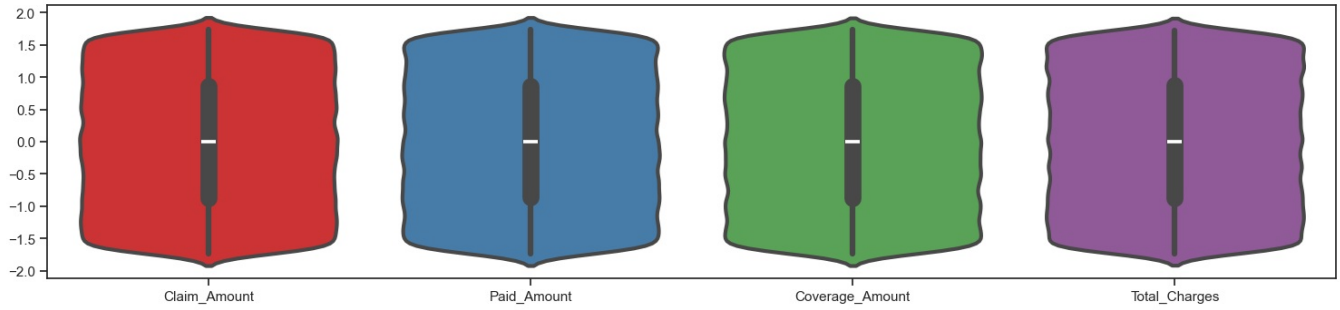

Norm. disti.of Occupation (top 15 categories only)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Teacher
Engineer
Businessman
Artist
Doctor
Occupation


Norm. disti.of Marital_Status (top 15 categories only)
0.25
0.20
0.15
0.10
0.05
0.00
Single
Divorced
Widowed
Married
Marital_Status

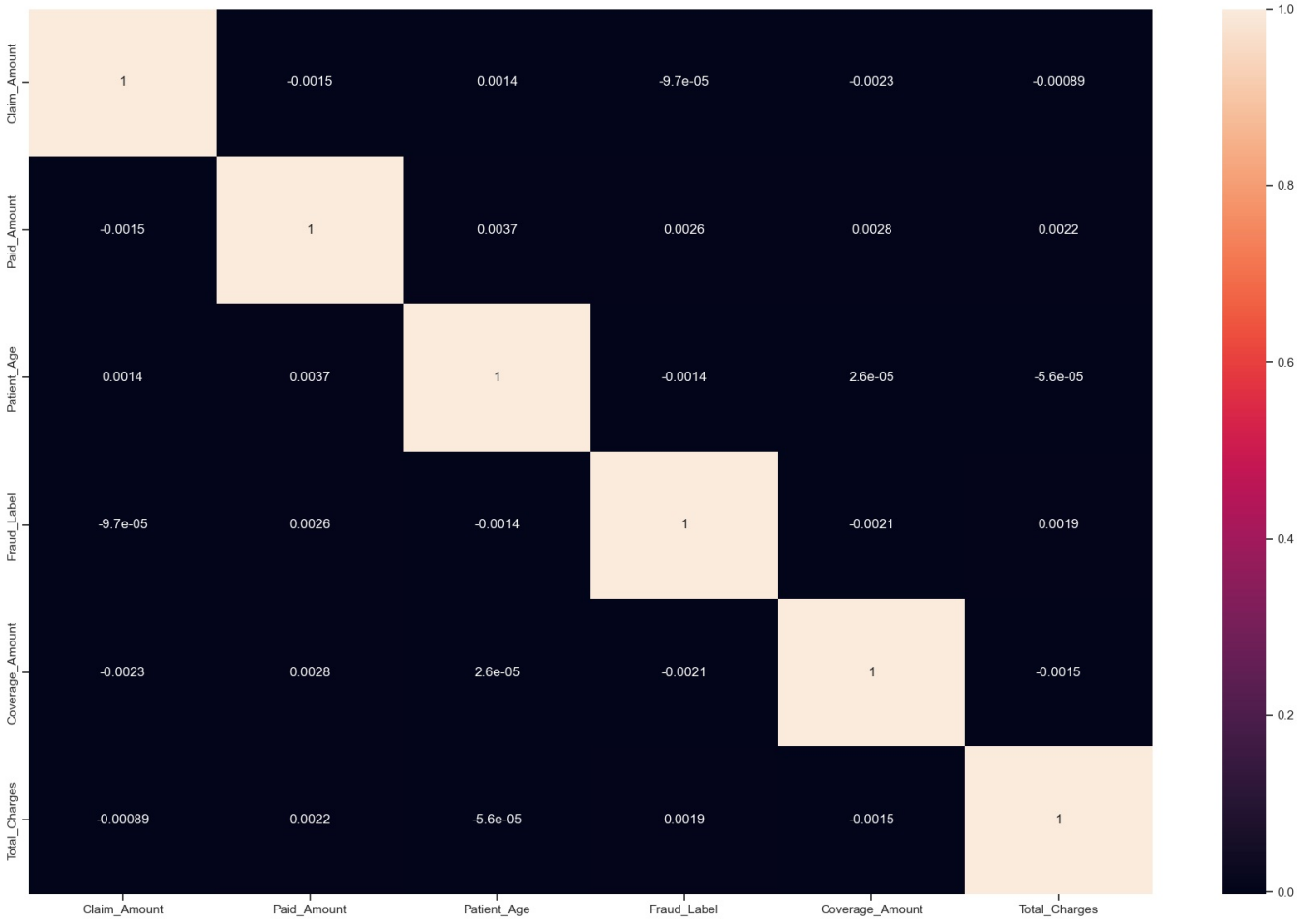

Norm. disti.of Education_Level (top 15 categories only)
0.25
0.20
0.15
0.10
0.05
0.00
Bachelor
High School
PhD
Master
Education_Level


Norm. disti.of Patient_Age (top 15 categories only)
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
60
50
81
86
57
80
67
73
71
53
70
27
47
38
22
Patient_Age

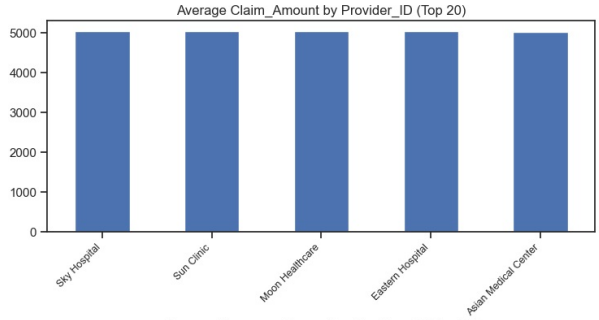
Violin Plot of all Continuous Variables

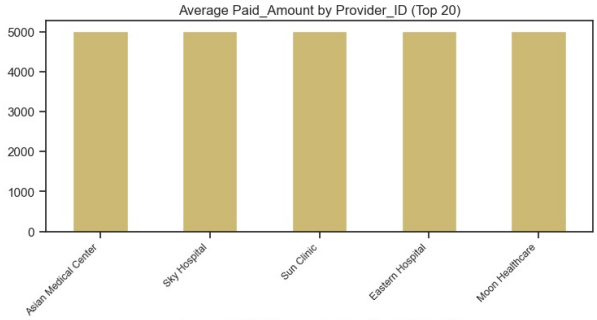


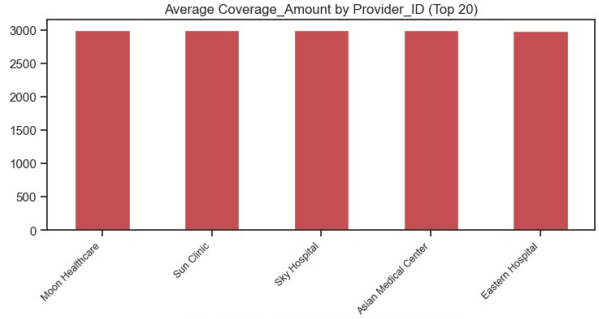
Heatmap of all Numeric Variables including target:

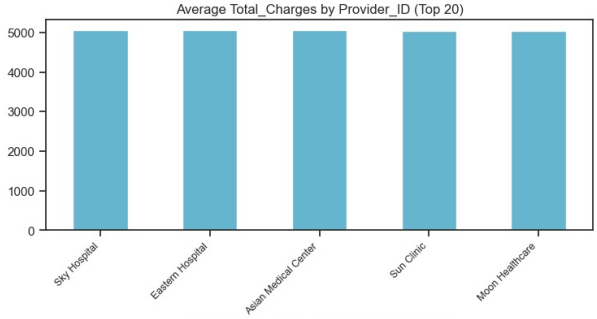


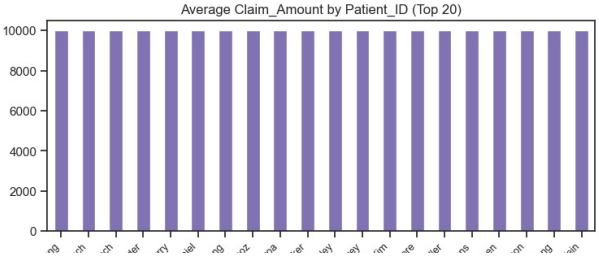
Bar plots for each Continuous by each Categorical variable

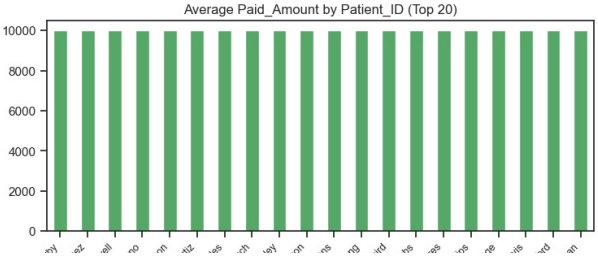

Average Claim_Amount by Provider_ID (Top 20)
5000
4000
3000
2000
1000
0
Sky Hospital
Sun Clinic
Moon Healthcare
Eastern Hospital
Asian Medical Center

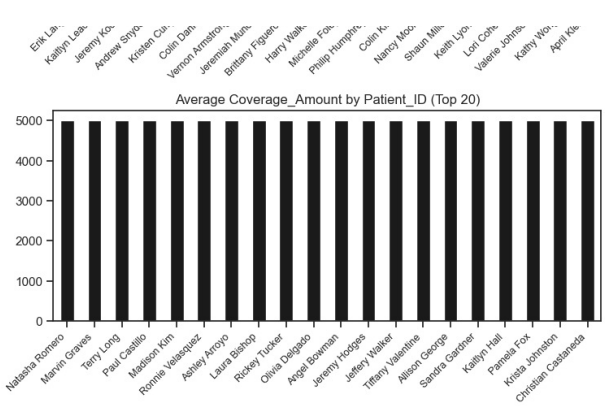

Average Paid_Amount by Provider_ID (Top 20)
5000
4000
3000
2000
1000
0
Asian Medical Center
Sky Hospital
Sun Clinic
Eastern Hospital
Moon Healthcare

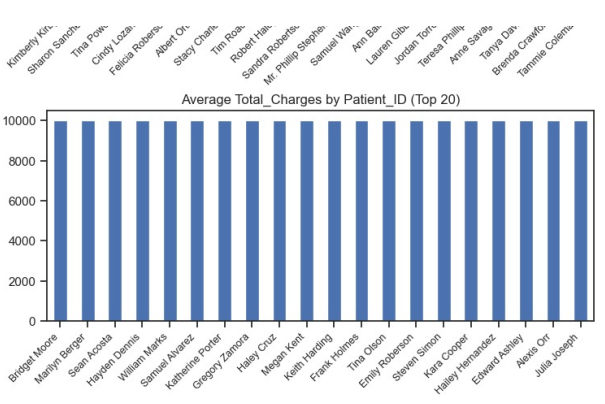

Average Coverage_Amount by Provider_ID (Top 20)
3000
2500
2000
1500
1000
500
0
Moon Healthcare
Sun Clinic
Sky Hospital
Asian Medical Center
Eastern Hospital

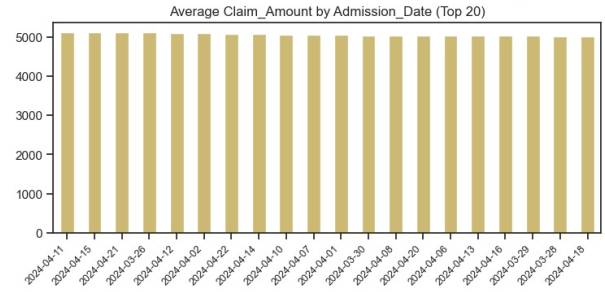

Average Total_Charges by Provider_ID (Top 20)
5000
4000
3000
2000
1000
0
Sky Hospital
Eastern Hospital
Asian Medical Center
Sun Clinic
Moon Healthcare

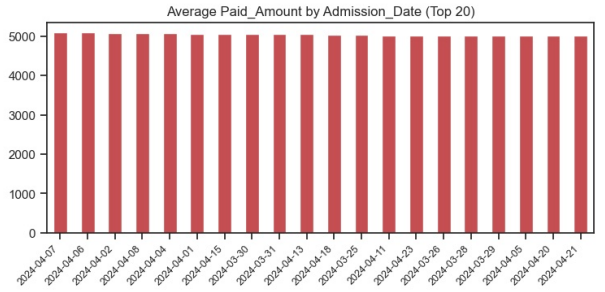

Average Claim_Amount by Patient_ID (Top 20)
10000
8000
6000
4000
2000
0

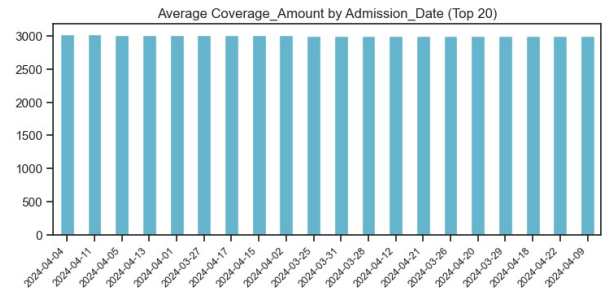

Average Paid_Amount by Patient_ID (Top 20)
10000
8000
6000
4000
2000
0

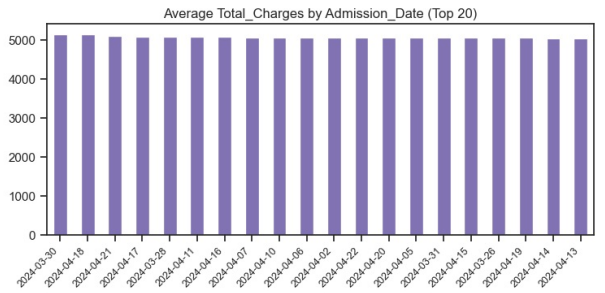

Average Coverage_Amount by Patient_ID (Top 20)
Natasha Romero
Marvin Graves
Terry Long
Paul Castillo
Madison Kim
Ronnie Velasquez
Ashley Arroyo
Laura Bishop
Rickey Tucker
Olivia Delgado
Angel Bowman
Jeremy Hodges
Jeffery Walker
Tiffany Valentine
Allison George
Sandra Gardner
Kaitlyn Hall
Pamela Fox
Krista Johnston
Christian Castaneda

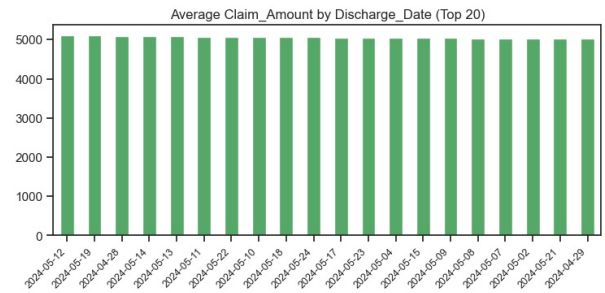

Average Total_Charges by Patient_ID (Top 20)
Bridget Moore
Marilyn Berger
Sean Acosta
Hayden Dennis
William Marks
Samuel Alvarez
Katherine Porter
Gregory Zamora
Haley Cruz
Megan Kent
Keith Harding
Frank Holmes
Tina Olson
Emily Roberson
Steven Simon
Kara Cooper
Hailey Hernandez
Edward Ashley
Alexis Orr
Julia Joseph

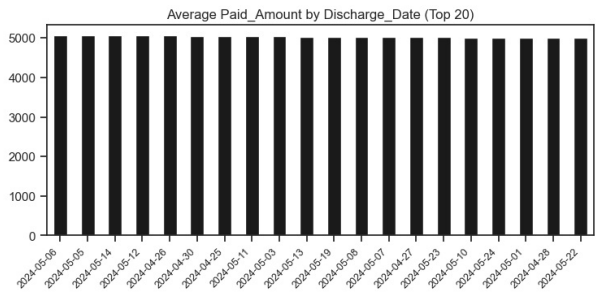

Average Claim_Amount by Admission_Date (Top 20)
2024-04-11
2024-04-15
2024-04-21
2024-03-26
2024-04-12
2024-04-02
2024-04-22
2024-04-14
2024-04-10
2024-04-07
2024-04-01
2024-03-30
2024-04-08
2024-04-20
2024-04-06
2024-04-13
2024-04-16
2024-03-29
2024-03-28
2024-04-18

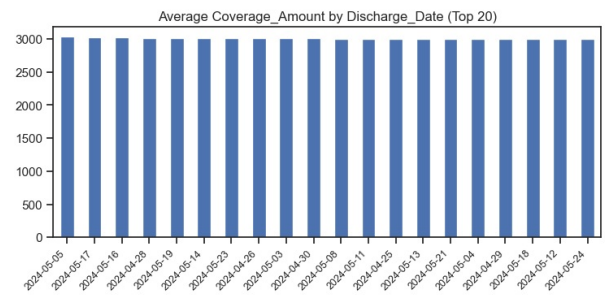

Average Paid_Amount by Admission_Date (Top 20)
2024-04-07
2024-04-06
2024-04-02
2024-04-08
2024-04-04
2024-04-01
2024-04-15
2024-03-30
2024-03-31
2024-04-13
2024-04-18
2024-03-25
2024-04-11
2024-04-23
2024-03-26
2024-03-28
2024-03-29
2024-04-05
2024-04-20
2024-04-21

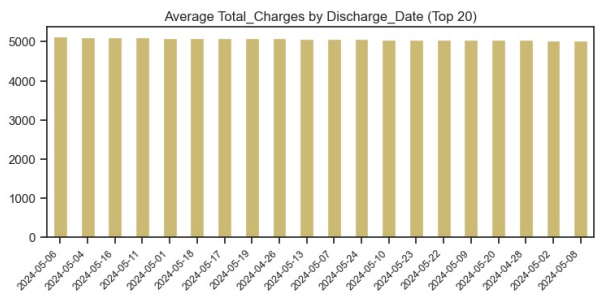

Average Coverage_Amount by Admission_Date (Top 20)
2024-04-04
2024-04-11
2024-04-05
2024-04-13
2024-04-01
2024-03-27
2024-04-17
2024-04-15
2024-04-02
2024-03-25
2024-03-31
2024-03-28
2024-04-12
2024-04-21
2024-03-26
2024-04-20
2024-03-29
2024-04-18
2024-04-22
2024-04-09

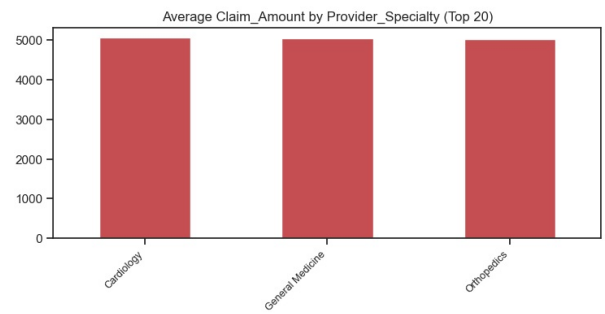

Average Total_Charges by Admission_Date (Top 20)
2024-03-30
2024-04-18
2024-04-21
2024-04-17
2024-03-28
2024-04-11
2024-04-16
2024-04-07
2024-04-10
2024-04-06
2024-04-02
2024-04-22
2024-04-20
2024-04-05
2024-03-31
2024-04-15
2024-03-26
2024-04-19
2024-04-14
2024-04-13

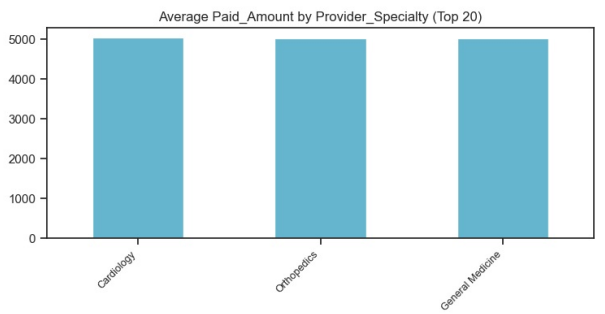

Average Claim_Amount by Discharge_Date (Top 20)
2024-05-12
2024-05-19
2024-04-28
2024-05-14
2024-05-13
2024-05-11
2024-05-22
2024-05-10
2024-05-18
2024-05-24
2024-05-17
2024-05-23
2024-05-04
2024-05-15
2024-05-09
2024-05-08
2024-05-07
2024-05-02
2024-05-21
2024-04-29

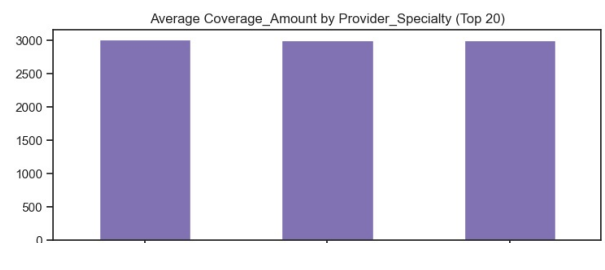

Average Paid_Amount by Discharge_Date (Top 20)
2024-05-06
2024-05-05
2024-05-14
2024-05-12
2024-04-26
2024-04-30
2024-04-25
2024-05-11
2024-05-03
2024-05-13
2024-05-19
2024-05-08
2024-05-07
2024-04-27
2024-05-23
2024-05-10
2024-05-24
2024-05-01
2024-04-28
2024-05-22

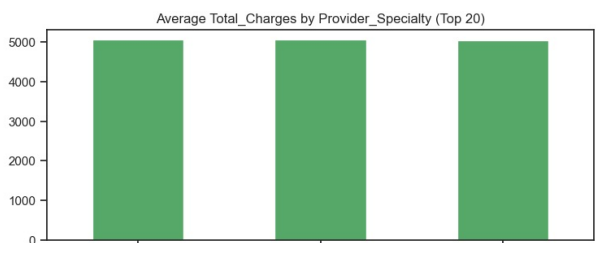

Average Coverage_Amount by Discharge_Date (Top 20)
2024-05-05
2024-05-17
2024-05-16
2024-04-28
2024-05-19
2024-05-14
2024-05-23
2024-04-26
2024-05-03
2024-04-30
2024-05-08
2024-05-11
2024-04-25
2024-05-13
2024-05-21
2024-05-04
2024-04-29
2024-05-18
2024-05-12
2024-05-24


Average Total_Charges by Discharge_Date (Top 20)
2024-05-06
2024-05-04
2024-05-16
2024-05-11
2024-05-01
2024-05-18
2024-05-17
2024-05-19
2024-04-26
2024-05-13
2024-05-07
2024-05-24
2024-05-10
2024-05-23
2024-05-22
2024-05-09
2024-05-20
2024-04-28
2024-05-02
2024-05-08


Average Claim_Amount by Provider_Specialty (Top 20)
Cardiology
General Medicine
Orthopedics

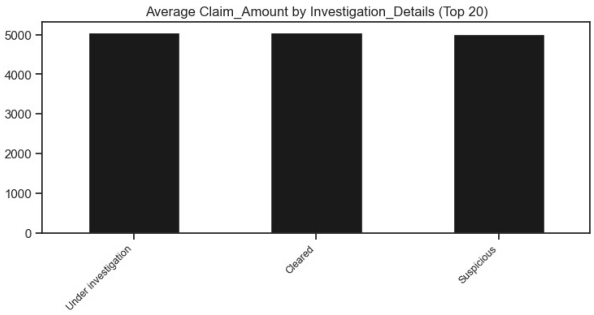

Average Paid_Amount by Provider_Specialty (Top 20)
Cardiology
Orthopedics
General Medicine

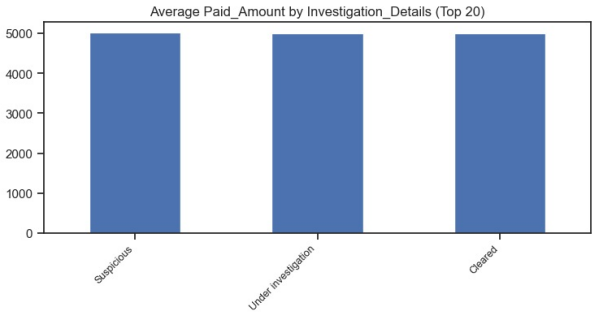

Average Coverage_Amount by Provider_Specialty (Top 20)

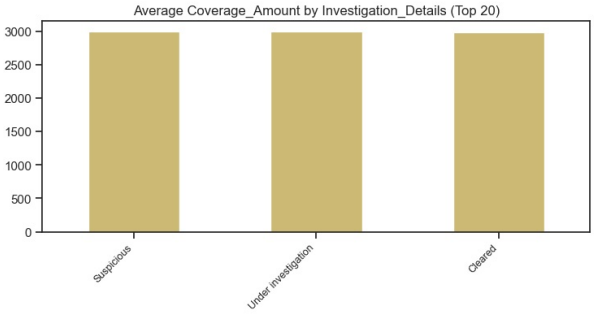

Average Total_Charges by Provider_Specialty (Top 20)

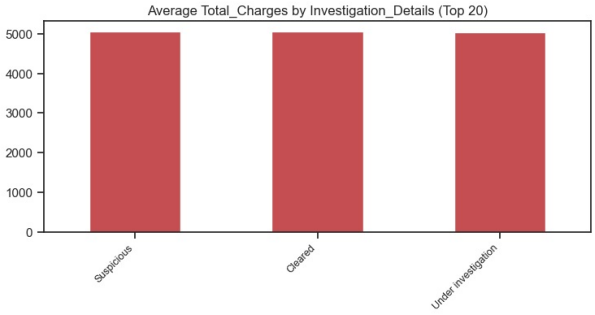

Cardiology
General Medicine
Orthopedics

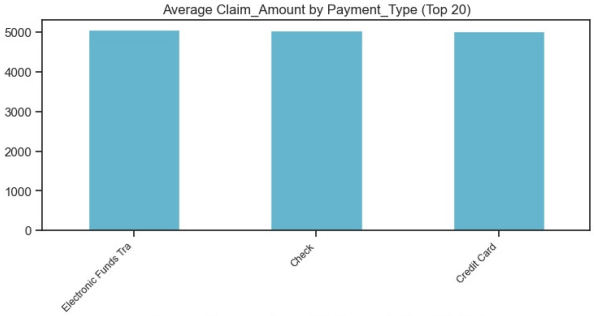

Cardiology
Orthopedics
General Medicine

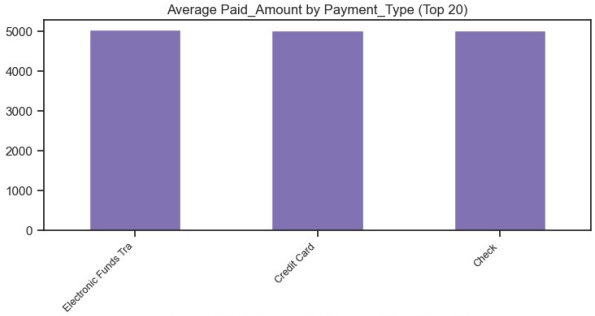

Average Claim_Amount by Investigation_Details (Top 20)
5000
4000
3000
2000
1000
0
Under investigation
Cleared
Suspicious

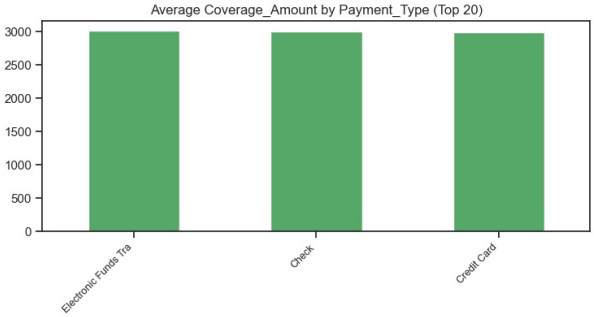

Average Paid_Amount by Investigation_Details (Top 20)
5000
4000
3000
2000
1000
0
Suspicious
Under investigation
Cleared

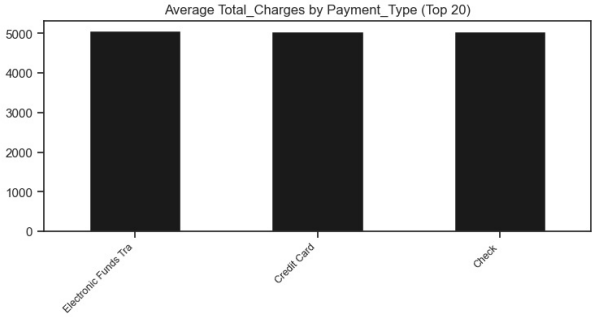

Average Coverage_Amount by Investigation_Details (Top 20)
3000
2500
2000
1500
1000
500
0
Suspicious
Under investigation
Cleared

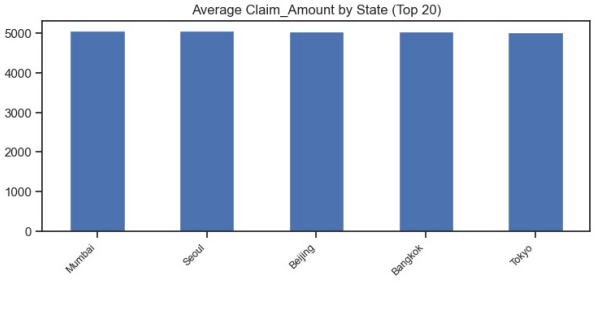

Average Total_Charges by Investigation_Details (Top 20)
5000
4000
3000
2000
1000
0
Suspicious
Cleared
Under investigation

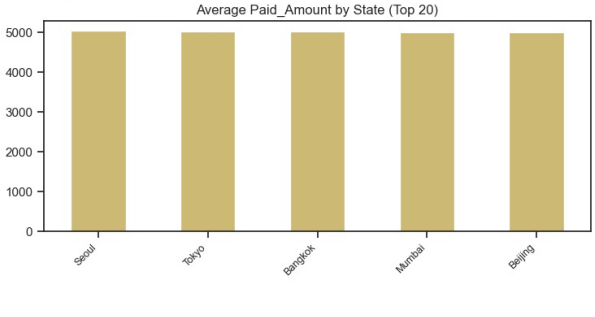

Average Claim_Amount by Payment_Type (Top 20)
5000
4000
3000
2000
1000
0
Electronic Funds Tra
Check
Credit Card

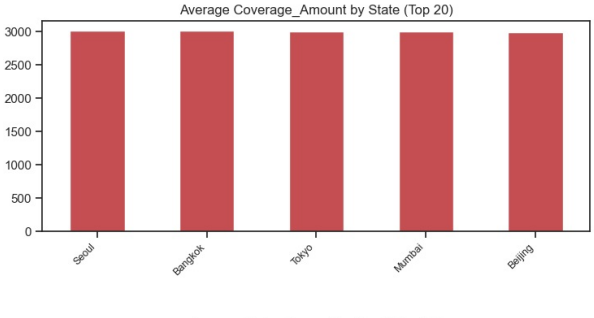

Average Paid_Amount by Payment_Type (Top 20)
5000
4000
3000
2000
1000
0
Electronic Funds Tra
Credit Card
Check

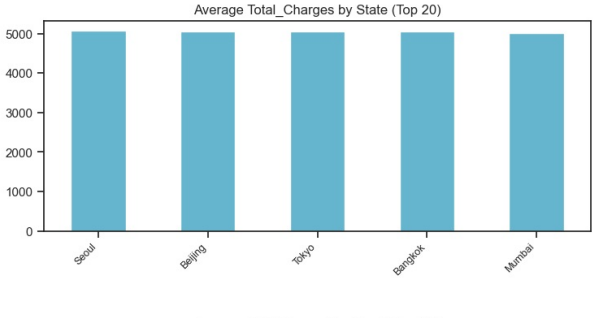

Average Coverage_Amount by Payment_Type (Top 20)
3000
2500
2000
1500
1000
500
0
Electronic Funds Tra
Check
Credit Card

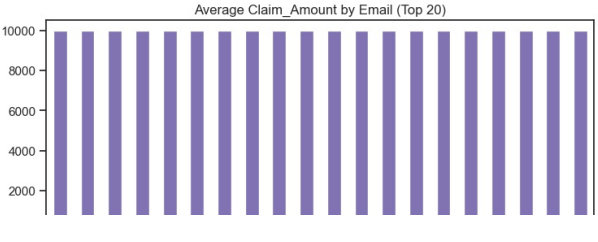

Average Total_Charges by Payment_Type (Top 20)
5000
4000
3000
2000
1000
0
Electronic Funds Tra
Credit Card
Check

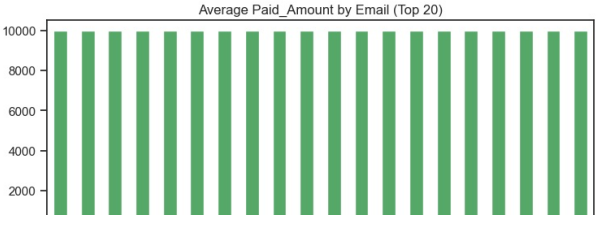

Average Claim_Amount by State (Top 20)
5000
4000
3000
2000
1000
0
Mumbai
Seoul
Beijing
Bangkok
Tokyo

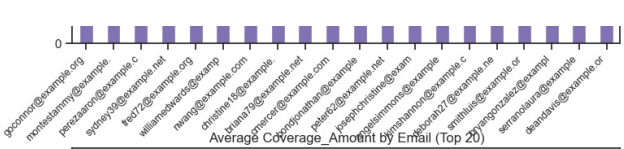

Average Paid_Amount by State (Top 20)
5000
4000
3000
2000
1000
0
Seoul
Tokyo
Bangkok
Mumbai
Beijing

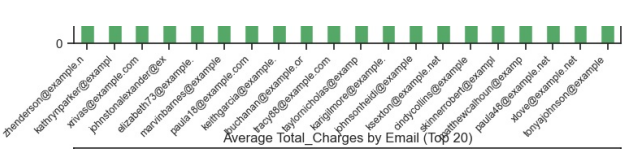

Average Coverage_Amount by State (Top 20)
3000
2500
2000
1500
1000
500
0
Seoul
Bangkok
Tokyo
Mumbai
Beijing

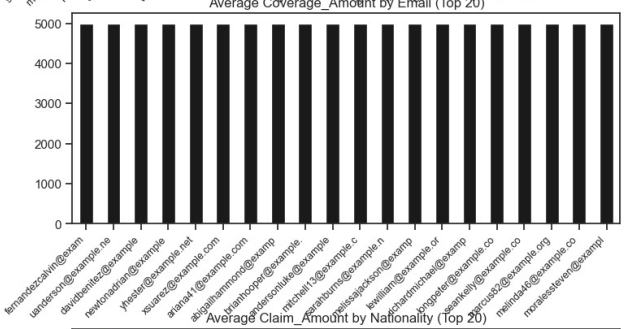

Average Total_Charges by State (Top 20)
5000
4000
3000
2000
1000
0
Seoul
Beijing
Tokyo
Bangkok
Mumbai

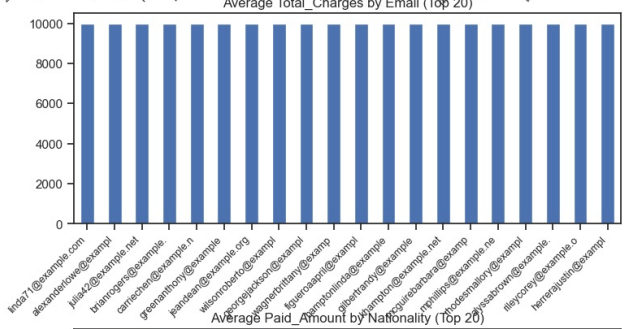

Average Claim_Amount by Email (Top 20)
10000
8000
6000
4000
2000

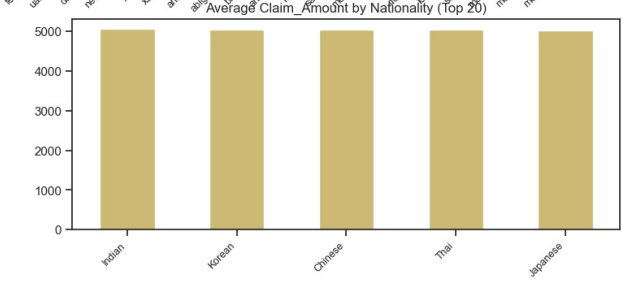

Average Paid_Amount by Email (Top 20)
10000
8000
6000
4000
2000

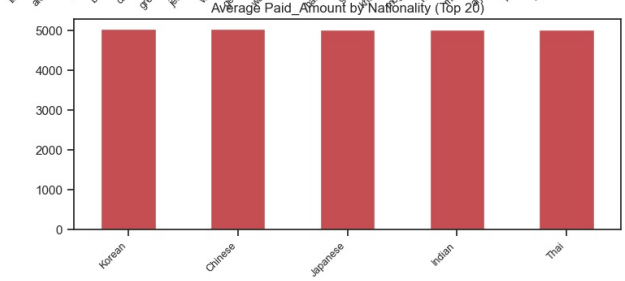


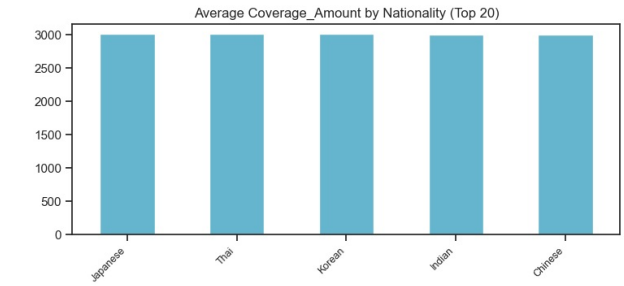


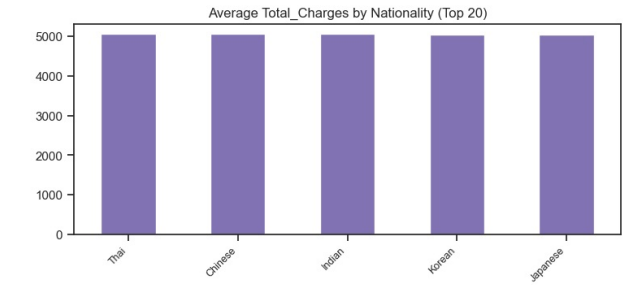

Average Coverage_Amount by Email (Top 20)

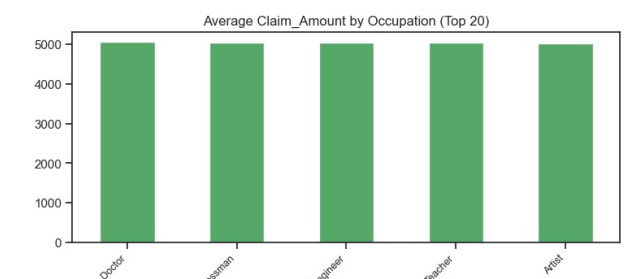

Average Total_Charges by Email (Top 20)

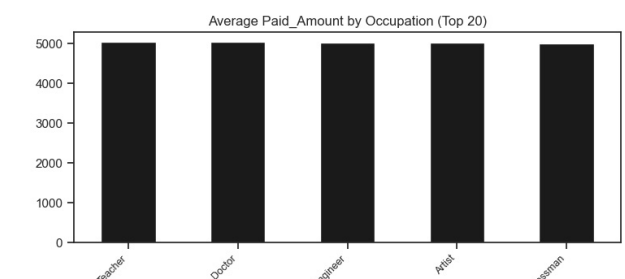

Average Claim_Amount by Nationality (Top 20)
Indian
Korean
Chinese
Thai
Japanese

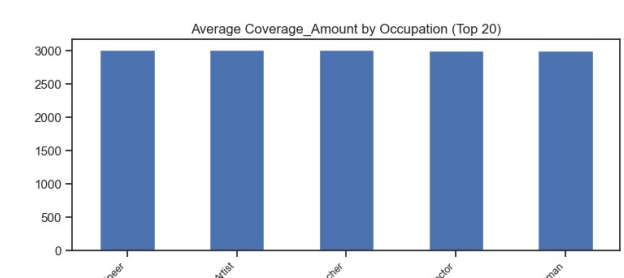

Average Paid_Amount by Nationality (Top 20)
Korean
Chinese
Japanese
Indian
Thai

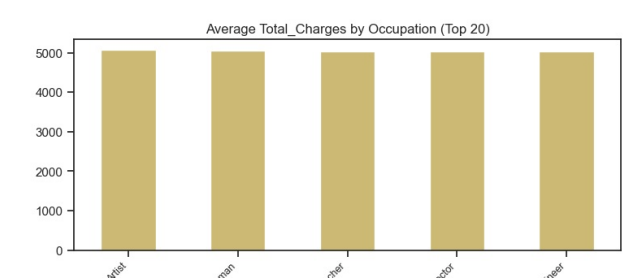

Average Coverage_Amount by Nationality (Top 20)
Japanese
Thai
Korean
Indian
Chinese

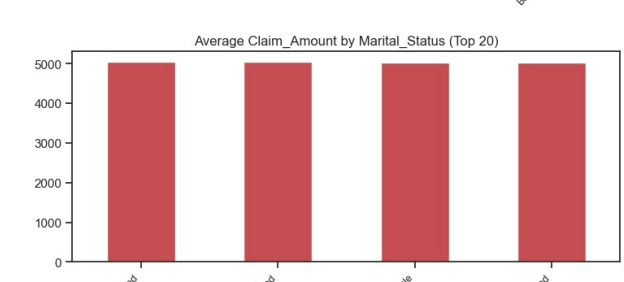

Average Total_Charges by Nationality (Top 20)
Thai
Chinese
Indian
Korean
Japanese

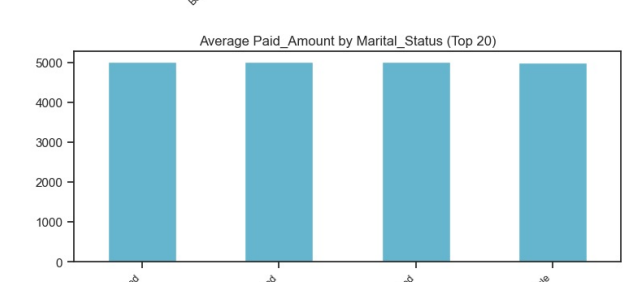

Average Claim_Amount by Occupation (Top 20)
Doctor
Businessman
Engineer
Teacher
Artist

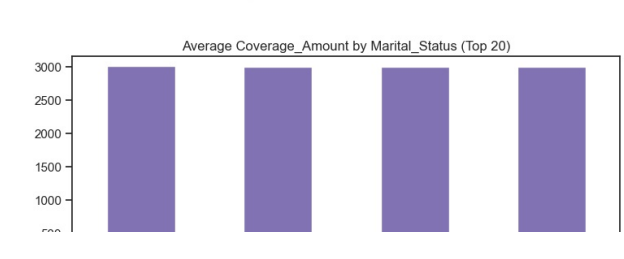

Average Paid_Amount by Occupation (Top 20)
Teacher
Doctor
Engineer
Artist
Businessman

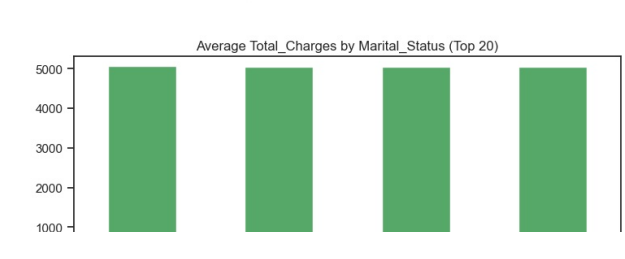

Average Coverage_Amount by Occupation (Top 20)
Engineer
Artist
Teacher
Doctor
Businessman

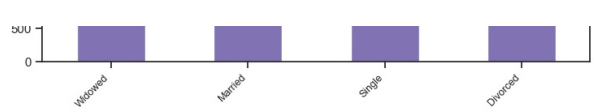

Average Total_Charges by Occupation (Top 20)
Artist
Businessman
Teacher
Doctor
Engineer

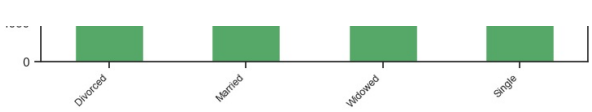

Average Claim_Amount by Marital_Status (Top 20)
Widowed
Divorced
Single
Married

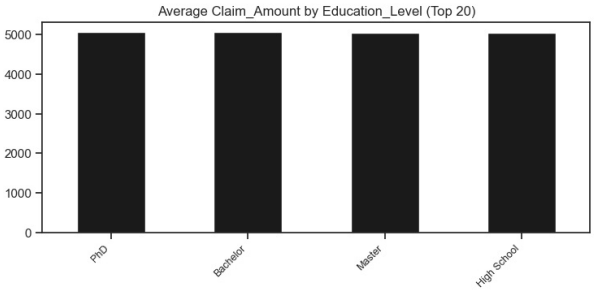

Average Paid_Amount by Marital_Status (Top 20)
Widowed
Divorced
Married
Single

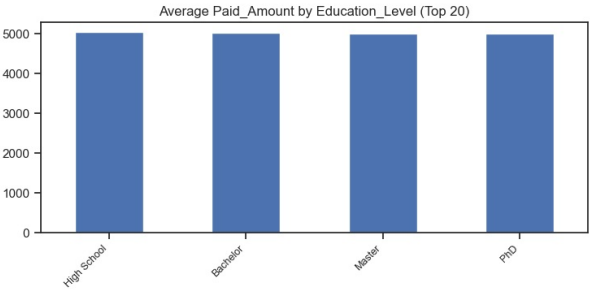

Average Coverage_Amount by Marital_Status (Top 20)

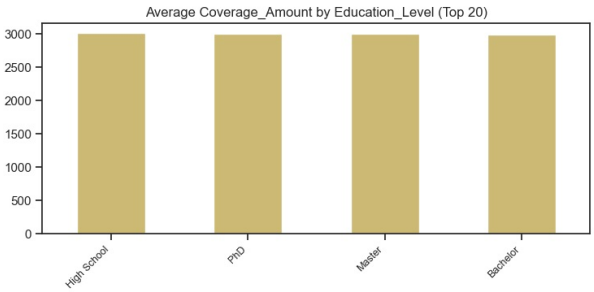

Average Total_Charges by Marital_Status (Top 20)

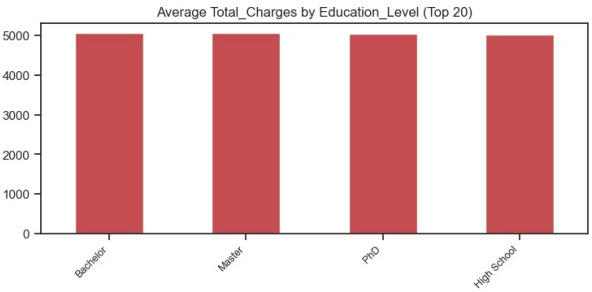

Widowed
Married
Single
Divorced

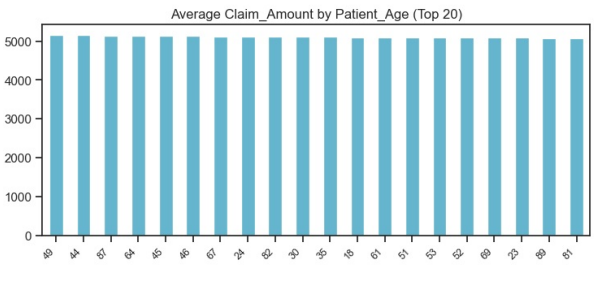

Divorced
Married
Widowed
Single

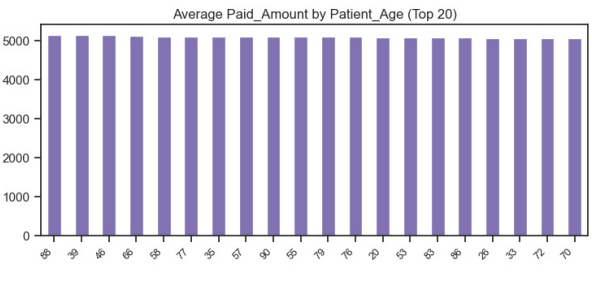

Average Claim_Amount by Education_Level (Top 20)
PhD
Bachelor
Master
High School

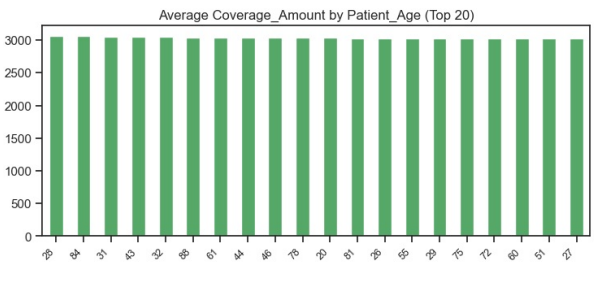

Average Paid_Amount by Education_Level (Top 20)
High School
Bachelor
Master
PhD

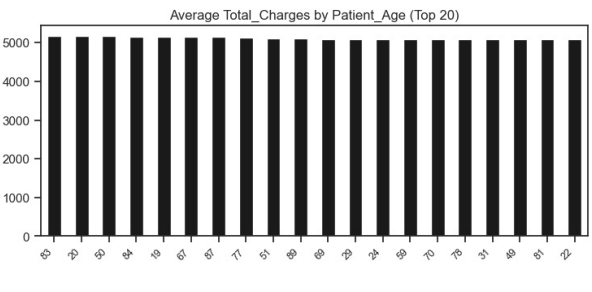

Average Coverage_Amount by Education_Level (Top 20)
High School
PhD
Master
Bachelor

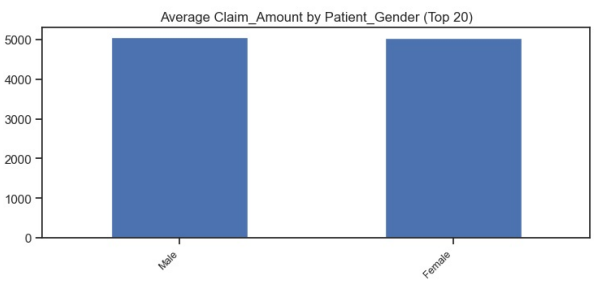

Average Total_Charges by Education_Level (Top 20)
Bachelor
Master
PhD
High School

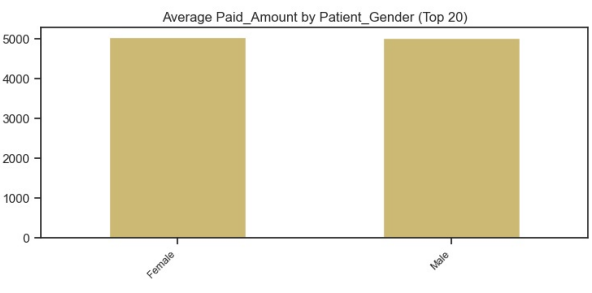

Average Claim_Amount by Patient_Age (Top 20)
49
44
67
64
45
46
67
24
82
30
35
18
61
51
53
52
69
23
89
81

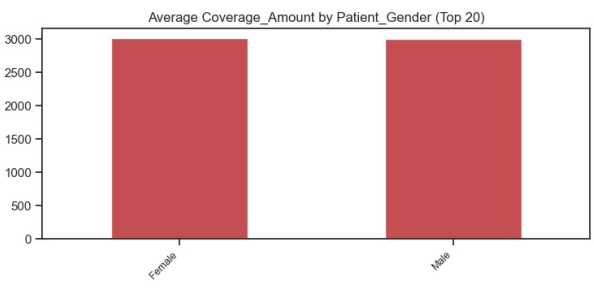

Average Paid_Amount by Patient_Age (Top 20)
88
39
46
66
58
77
35
57
90
55
79
76
20
53
83
86
26
33
72
70

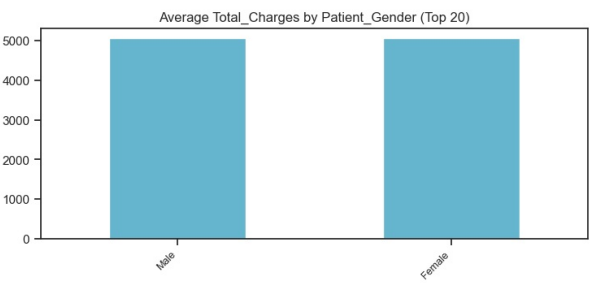

Average Coverage_Amount by Patient_Age (Top 20)
28
84
31
43
32
88
61
44
46
78
20
81
26
55
29
75
72
60
51
27

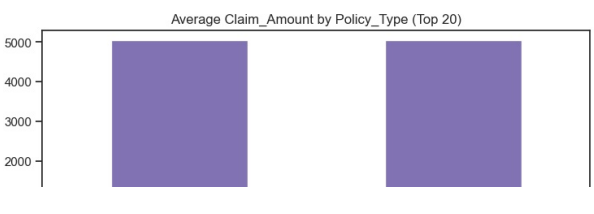

Average Total_Charges by Patient_Age (Top 20)
83
20
50
84
19
67
87
77
51
89
69
29
24
59
70
78
31
49
81
22

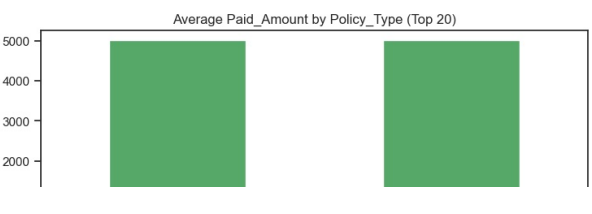

Average Claim_Amount by Patient_Gender (Top 20)
Male
Female

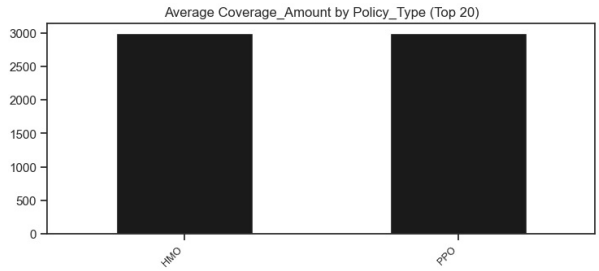

Average Paid_Amount by Patient_Gender (Top 20)
Female
Male

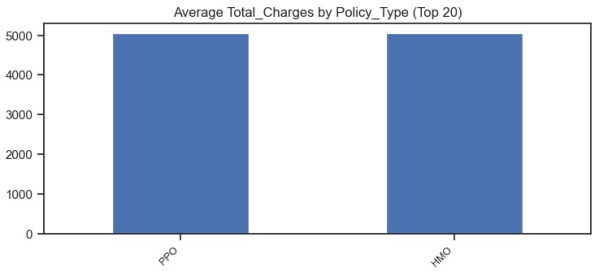

Average Coverage_Amount by Patient_Gender (Top 20)
Female
Male

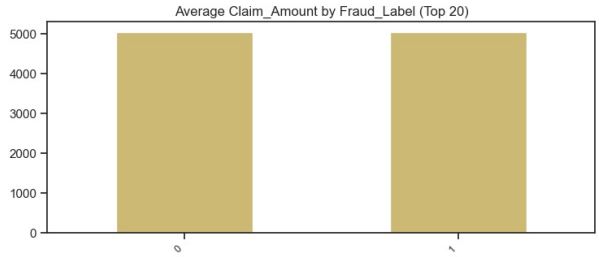

Average Total_Charges by Patient_Gender (Top 20)
Male
Female

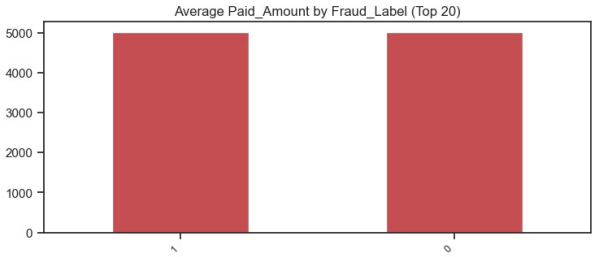

Average Claim_Amount by Policy_Type (Top 20)

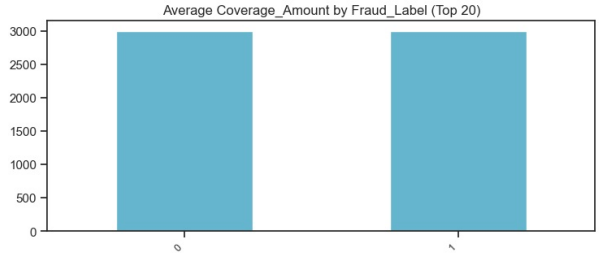

Average Paid_Amount by Policy_Type (Top 20)









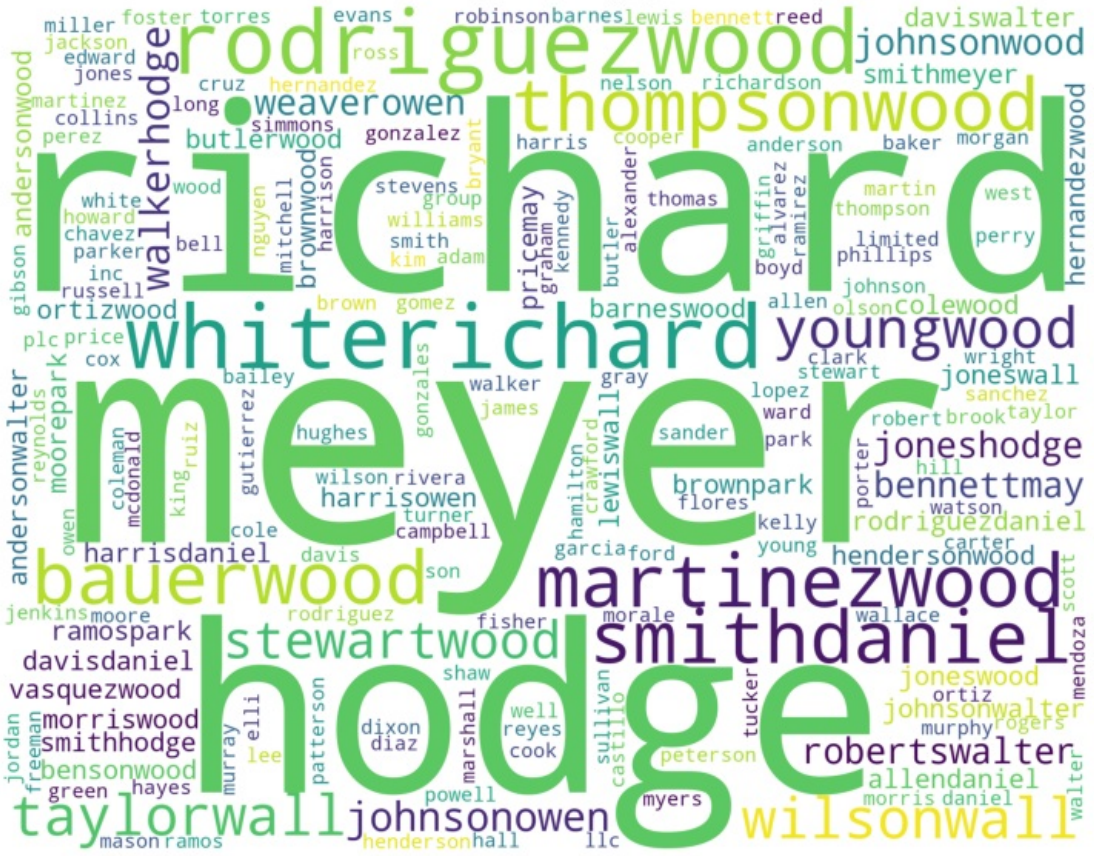



```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
```

```
[nltk_data]    | Downloading package omw to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package omw-1.4 to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package omw-1.4 is already up-to-date!
[nltk_data]    | Downloading package wordnet to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package punkt to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     C:\Users\Pushpendra.PUSHPENDRA-
[nltk_data]    |     PC.001\AppData\Roaming\nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection popular
```

Wordcloud for Procedure_Code

Wordcloud for Employer




```
All Plots done
Time to run AutoViz = 217 seconds

 ###################### AUTO VISUALIZATION Completed ########################
```

In [ ]:

In [ ]:

Analyzing the Fraud Records

In [19]:
```
df['Fraud_Label'].value_counts()
```

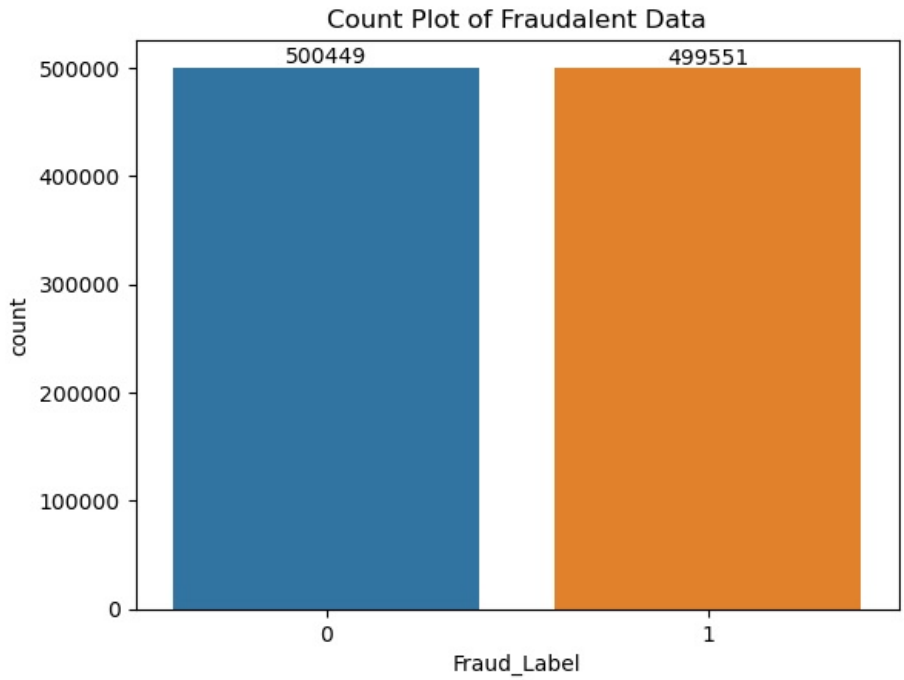
Out[19]:
```
Fraud_Label
0    500449
1    499551
Name: count, dtype: int64
```

0 -> Genuine Data

1 -> Fraud Data

In [20]:
```
# Drawing Count Plot
ax = sns.countplot(data=df, x='Fraud_Label')
ax.bar_label(ax.containers[0])
plt.title('Count Plot of Fraudalent Data')
```

Out[20]:
```
Text(0.5, 1.0, 'Count Plot of Fraudalent Data')
```



Features and Target Creation

In [21]:
```
target_name = 'Fraud_Label'
feature_names = [col_name for col_name in df.columns if col_name != target_name]
```

In [22]:
```
features = df[feature_names].copy()
target = df[target_name].copy()
```

In [23]:
```
features.head()
```

Out[23]:

	Provider_ID	Claim_ID	Patient_ID	Diagnosis_Code	Procedure_Code	Admission_Date	Discharge_Date	Claim_Amount	Paid_Amount	Provi
0	Asian Medical Center	CLAIM_1	Darrell Blair	DX_714	PROC_2648	2024-03-26	2024-05-08	1077.86	4362.78	
1	Sky Hospital	CLAIM_2	William Young	DX_885	PROC_9084	2024-04-07	2024-05-03	4998.88	5867.30	
2	Moon Healthcare	CLAIM_3	Keith Reynolds	DX_988	PROC_9747	2024-04-01	2024-05-24	7058.21	8526.15	
3	Sky Hospital	CLAIM_4	Andre Kelly	DX_779	PROC_4334	2024-03-31	2024-04-27	1628.67	8317.18	
4	Sun Clinic	CLAIM_5	Terry Gonzales	DX_644	PROC_8408	2024-03-27	2024-05-12	1480.43	4136.33	

5 rows × 27 columns

In [24]:
```
target.head()
```

Out[24]:
```
0    1
1    1
2    0
3    0
4    0
Name: Fraud_Label, dtype: int64
```

In [25]:
```
target.value_counts()
```

Out[25]:
```
Fraud_Label
0    500449
1    499551
Name: count, dtype: int64
```

Forming categorical and numerical columns

On reading and analysing the dataset, the list of categorical and numerical columns are formed.

Categorical Columns:

'Provider_ID', 'Claim_ID', 'Patient_ID', 'Diagnosis_Code', 'Procedure_Code', 'Admission_Date', 'Discharge_Date', 'Provider_Specialty', 'Patient_Gender', 'Investigation_Details', 'Policy_Type', 'Payment_Type', 'State', 'Email', 'Phone_Number', 'Address', 'Nationality', 'Passport_Number', 'Employer', 'Occupation', 'Marital_Status', 'Education_Level'.

Numerical Columns:

'Claim_Amount', 'Paid_Amount', 'Patient_Age', 'Coverage_Amount', 'Total_Charges'.

In [26]:
```
#Categorical Columns Creation
categorical_cols = ['Provider_ID', 'Claim_ID', 'Patient_ID', 'Diagnosis_Code','Procedure_Code', 'Admission_Date
                    'Discharge_Date','Provider_Specialty', 'Patient_Gender', 'Investigation_Details','Policy_Ty
                    'Payment_Type', 'State', 'Email', 'Phone_Number','Address', 'Nationality', 'Passport_Number
                    'Employer', 'Occupation','Marital_Status', 'Education_Level']
categorical_cols
```

Out[26]:
```
['Provider_ID',
 'Claim_ID',
 'Patient_ID',
 'Diagnosis_Code',
 'Procedure_Code',
 'Admission_Date',
 'Discharge_Date',
 'Provider_Specialty',
 'Patient_Gender',
 'Investigation_Details',
 'Policy_Type',
 'Payment_Type',
 'State',
 'Email',
 'Phone_Number',
 'Address',
 'Nationality',
 'Passport_Number',
 'Employer',
 'Occupation',
 'Marital_Status',
 'Education_Level']
```

In [27]:
```
#Numerical Columns Creation
numerical_cols = ['Claim_Amount', 'Paid_Amount', 'Patient_Age','Coverage_Amount', 'Total_Charges']
numerical_cols
```

Out[27]:
```
['Claim_Amount',
 'Paid_Amount',
 'Patient_Age',
 'Coverage_Amount',
 'Total_Charges']
```

Scaling the numerical columns

In [28]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
features[numerical_cols] = scaler.fit_transform(features[numerical_cols])
```

In [29]:
```
# Checking the data types of columns
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 27 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   Provider_ID            1000000 non-null  object
 1   Claim_ID               1000000 non-null  object
 2   Patient_ID             1000000 non-null  object
 3   Diagnosis_Code         1000000 non-null  object
 4   Procedure_Code         1000000 non-null  object
 5   Admission_Date         1000000 non-null  object
 6   Discharge_Date         1000000 non-null  object
 7   Claim_Amount           1000000 non-null  float64
 8   Paid_Amount            1000000 non-null  float64
 9   Provider_Specialty     1000000 non-null  object
 10  Patient_Age            1000000 non-null  float64
 11  Patient_Gender         1000000 non-null  object
 12  Investigation_Details  1000000 non-null  object
 13  Policy_Type            1000000 non-null  object
 14  Coverage_Amount        1000000 non-null  float64
 15  Total_Charges          1000000 non-null  float64
 16  Payment_Type           1000000 non-null  object
 17  State                  1000000 non-null  object
 18  Email                  1000000 non-null  object
 19  Phone_Number           1000000 non-null  object
 20  Address                1000000 non-null  object
 21  Nationality            1000000 non-null  object
 22  Passport_Number        1000000 non-null  object
 23  Employer               1000000 non-null  object
 24  Occupation             1000000 non-null  object
 25  Marital_Status         1000000 non-null  object
 26  Education_Level        1000000 non-null  object
dtypes: float64(5), object(22)
memory usage: 206.0+ MB
```

In [30]:
```
# Converting categorical column data types to category
for col in categorical_cols:
    features[col] = features[col].astype('category')
```

In [31]:
```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 27 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   Provider_ID            1000000 non-null  category
 1   Claim_ID               1000000 non-null  category
 2   Patient_ID             1000000 non-null  category
 3   Diagnosis_Code         1000000 non-null  category
 4   Procedure_Code         1000000 non-null  category
 5   Admission_Date         1000000 non-null  category
 6   Discharge_Date         1000000 non-null  category
 7   Claim_Amount           1000000 non-null  float64
 8   Paid_Amount            1000000 non-null  float64
 9   Provider_Specialty     1000000 non-null  category
 10  Patient_Age            1000000 non-null  float64
 11  Patient_Gender         1000000 non-null  category
 12  Investigation_Details  1000000 non-null  category
 13  Policy_Type            1000000 non-null  category
 14  Coverage_Amount        1000000 non-null  float64
 15  Total_Charges          1000000 non-null  float64
 16  Payment_Type           1000000 non-null  category
 17  State                  1000000 non-null  category
 18  Email                  1000000 non-null  category
 19  Phone_Number           1000000 non-null  category
 20  Address                1000000 non-null  category
 21  Nationality            1000000 non-null  category
 22  Passport_Number        1000000 non-null  category
 23  Employer               1000000 non-null  category
 24  Occupation             1000000 non-null  category
 25  Marital_Status         1000000 non-null  category
 26  Education_Level        1000000 non-null  category
dtypes: category(22), float64(5)
memory usage: 291.8 MB
```

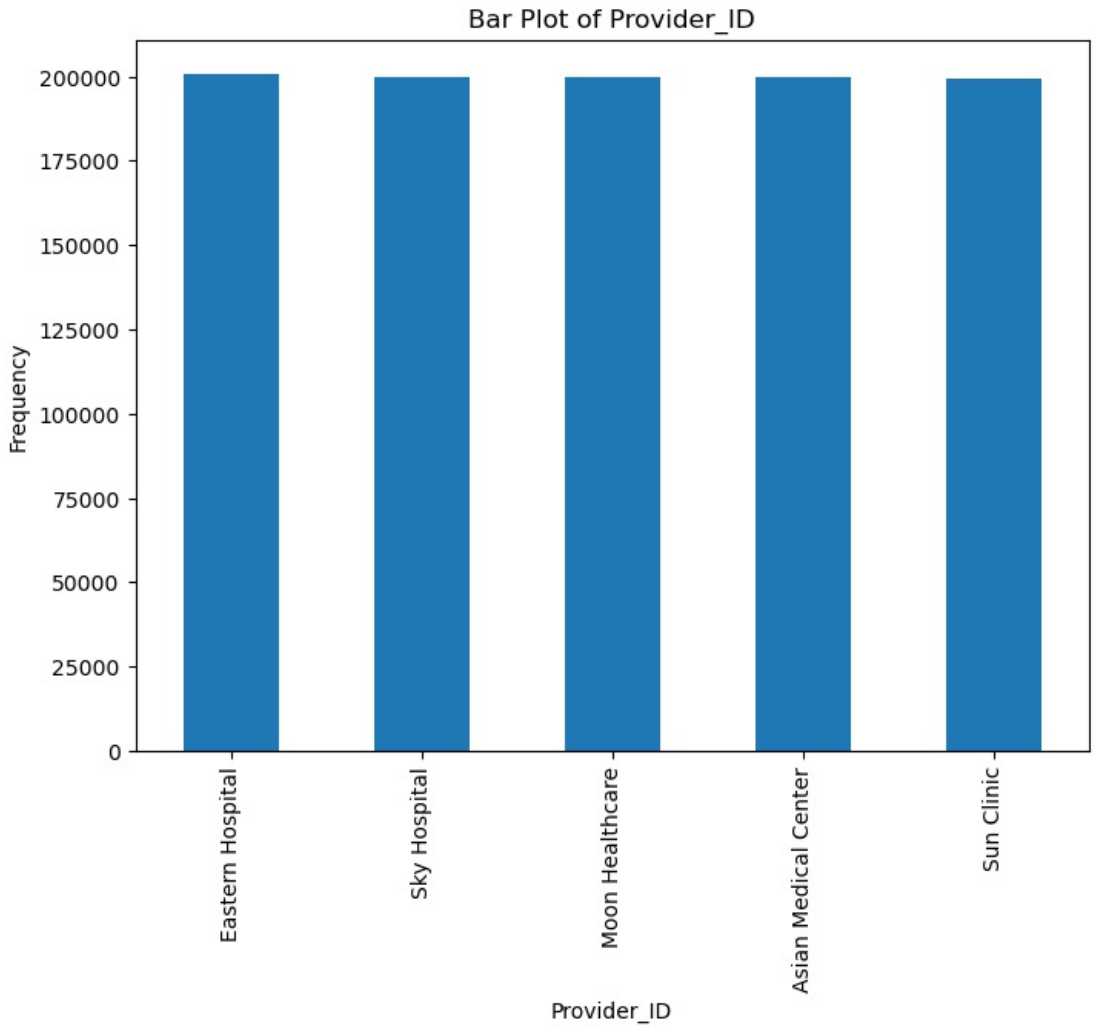
In [33]:
```
features[numerical_cols].head()
```

Out[33]:

	Claim_Amount	Paid_Amount	Patient_Age	Coverage_Amount	Total_Charges
0	-1.390600	-0.230187	1.044186	0.763088	1.702065
1	-0.017871	0.293884	0.901870	-1.263813	0.936243
2	0.703091	1.220043	-0.948242	-0.824871	1.617463
3	-1.197764	1.147253	0.190289	0.172578	0.993485
4	-1.249663	-0.309067	1.708329	-0.055276	-1.648847

Visualization of Categorical Columns

In [40]:
```
# List of categorical columns
cat_columns = ['Provider_ID', 'Claim_ID', 'Patient_ID', 'Diagnosis_Code', 'Procedure_Code',
               'Admission_Date', 'Discharge_Date', 'Provider_Specialty', 'Patient_Gender',
               'Investigation_Details', 'Policy_Type', 'Payment_Type', 'State', 'Email',
               'Phone_Number', 'Address', 'Nationality', 'Passport_Number', 'Employer',
               'Occupation', 'Marital_Status', 'Education_Level']

# Loop through each categorical column and plot a bar plot
for col in cat_columns:
    plt.figure(figsize=(8, 6))
    df[col].value_counts().plot(kind='bar')
    plt.title(f'Bar Plot of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```




```
KeyboardInterrupt
```

```
Error in callback <function _draw_all_if_interactive at 0x000001EBD7B46D40> (for post_execute):
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\pyplot.py:120, in _draw_all_if_interactive()
    118 def _draw_all_if_interactive():
    119     if matplotlib.is_interactive():
--> 120         draw_all()

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\_pylab_helpers.py:132, in Gcf.draw_all(cls, force)
    130 for manager in cls.get_all_fig_managers():
    131     if force or manager.canvas.figure.stale:
--> 132         manager.canvas.draw_idle()

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\backend_bases.py:2082, in FigureCanvasBase.draw_idle
(self, *args, **kwargs)
   2080 if not self._is_idle_drawing:
   2081     with self._idle_draw_cntx():
-> 2082         self.draw(*args, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\backends\backend_agg.py:400, in FigureCanvasAgg.draw
(self)
    396 # Acquire a lock on the shared font cache.
    397 with RendererAgg.lock, \
    398      (self.toolbar._wait_cursor_for_draw_cm() if self.toolbar
```

```
    399       else nullcontext()):
--> 400     self.figure.draw(self.renderer)
    401     # A GUI class may be need to update a window using this draw, so
    402     # don't forget to call the superclass.
    403     super().draw()

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\artist.py:95, in _finalize_rasterization.<locals>.dr
aw_wrapper(artist, renderer, *args, **kwargs)
     93 @wraps(draw)
     94 def draw_wrapper(artist, renderer, *args, **kwargs):
---> 95     result = draw(artist, renderer, *args, **kwargs)
     96     if renderer._rasterizing:
     97         renderer.stop_rasterizing()

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\artist.py:72, in allow_rasterization.<locals>.draw_w
rapper(artist, renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\figure.py:3175, in Figure.draw(self, renderer)
   3172         # ValueError can occur when resizing a window.
   3174 self.patch.draw(renderer)
-> 3175 mimage._draw_list_compositing_images(
   3176     renderer, self, artists, self.suppressComposite)
   3178 for sfig in self.subfigs:
   3179     sfig.draw(renderer)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\image.py:131, in _draw_list_compositing_images(rende
rer, parent, artists, suppress_composite)
    129 if not_composite or not has_images:
    130     for a in artists:
--> 131         a.draw(renderer)
    132 else:
    133     # Composite any adjacent images together
    134     image_group = []

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\artist.py:72, in allow_rasterization.<locals>.draw_w
rapper(artist, renderer)
     69     if artist.get_agg_filter() is not None:
     70         renderer.start_filter()
---> 72     return draw(artist, renderer)
     73 finally:
     74     if artist.get_agg_filter() is not None:

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:3035, in _AxesBase.draw(self, renderer
)
   3032         artists.remove(_axis)
   3034 if not self.figure.canvas.is_saving():
-> 3035     artists = [
   3036         a for a in artists
   3037         if not a.get_animated() or isinstance(a, mimage.AxesImage)]
   3038 artists = sorted(artists, key=attrgetter('zorder'))
   3040 # rasterize artists with negative zorder
   3041 # if the minimum zorder is negative, start rasterization

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:3035, in <listcomp>(.0)
   3032         artists.remove(_axis)
   3034 if not self.figure.canvas.is_saving():
-> 3035     artists = [
   3036         a for a in artists
   3037         if not a.get_animated() or isinstance(a, mimage.AxesImage)]
   3038 artists = sorted(artists, key=attrgetter('zorder'))
   3040 # rasterize artists with negative zorder
   3041 # if the minimum zorder is negative, start rasterization

KeyboardInterrupt:
Error in callback <function flush_figures at 0x000001EBDA045800> (for post_execute):
KeyboardInterrupt
```

In [ ]: