D207 - Exploratory Data Analysis

By Krista Moik

In [1]:

```
#import packages
import pandas as pd
import numpy as np
from scipy.stats import chi2_contingency
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [2]:

```
#Load medical_clean CSV
df = pd.read_csv('C:/Users/Kmoik WGU/Desktop/D207/medical_clean.csv')
```

In [3]:

```
#View data set
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CaseOrder           10000 non-null  int64  
 1   Customer_id         10000 non-null  object 
 2   Interaction         10000 non-null  object 
 3   UID                 10000 non-null  object 
 4   City                10000 non-null  object 
 5   State               10000 non-null  object 
 6   County              10000 non-null  object 
 7   Zip                 10000 non-null  int64  
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64  
 11  Area                10000 non-null  object 
 12  TimeZone            10000 non-null  object 
 13  Job                 10000 non-null  object 
 14  Children            10000 non-null  int64  
 15  Age                 10000 non-null  int64  
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object 
 18  Gender              10000 non-null  object 
 19  ReAdmis             10000 non-null  object 
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64  
 22  Full_meals_eaten    10000 non-null  int64  
 23  vitD_supp           10000 non-null  int64  
 24  Soft_drink          10000 non-null  object 
 25  Initial_admin       10000 non-null  object 
 26  HighBlood           10000 non-null  object 
 27  Stroke              10000 non-null  object 
 28  Complication_risk   10000 non-null  object 
 29  Overweight          10000 non-null  object 
 30  Arthritis           10000 non-null  object 
 31  Diabetes            10000 non-null  object 
 32  Hyperlipidemia      10000 non-null  object 
 33  BackPain            10000 non-null  object 
 34  Anxiety             10000 non-null  object 
 35  Allergic_rhinitis   10000 non-null  object 
 36  Reflux_esophagitis  10000 non-null  object 
 37  Asthma              10000 non-null  object 
 38  Services            10000 non-null  object 
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64  
 43  Item2               10000 non-null  int64  
 44  Item3               10000 non-null  int64  
 45  Item4               10000 non-null  int64  
 46  Item5               10000 non-null  int64  
 47  Item6               10000 non-null  int64  
 48  Item7               10000 non-null  int64  
 49  Item8               10000 non-null  int64  
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [4]:

```
#View variable ReAdmis
```

```
df.ReAdmis.describe()
```

Out[4]:

```
count      10000
unique         2
top           No
freq        6331
Name: ReAdmis, dtype: object
```

In [5]:

```
#Count responses in ReAdmis
df['ReAdmis'].value_counts(normalize=True, sort=True, dropna=True)
```

Out[5]:

```
ReAdmis
No     0.6331
Yes    0.3669
Name: proportion, dtype: float64
```

In [6]:

```
#View variable Complication_risk
df.Complication_risk.describe()
```

Out[6]:

```
count       10000
unique          3
top        Medium
freq         4517
Name: Complication_risk, dtype: object
```

In [7]:

```
#Count responses in Complication_risk
df['Complication_risk'].value_counts(normalize=True, sort=True, dropna=True)
```

Out[7]:

```
Complication_risk
Medium    0.4517
High      0.3358
Low       0.2125
Name: proportion, dtype: float64
```

In [8]:

```
#Create Contingency Table for Chi-Square using code from WGU course materials
table=pd.crosstab(df.ReAdmis, df.Complication_risk, margins=True)
print(table)
```

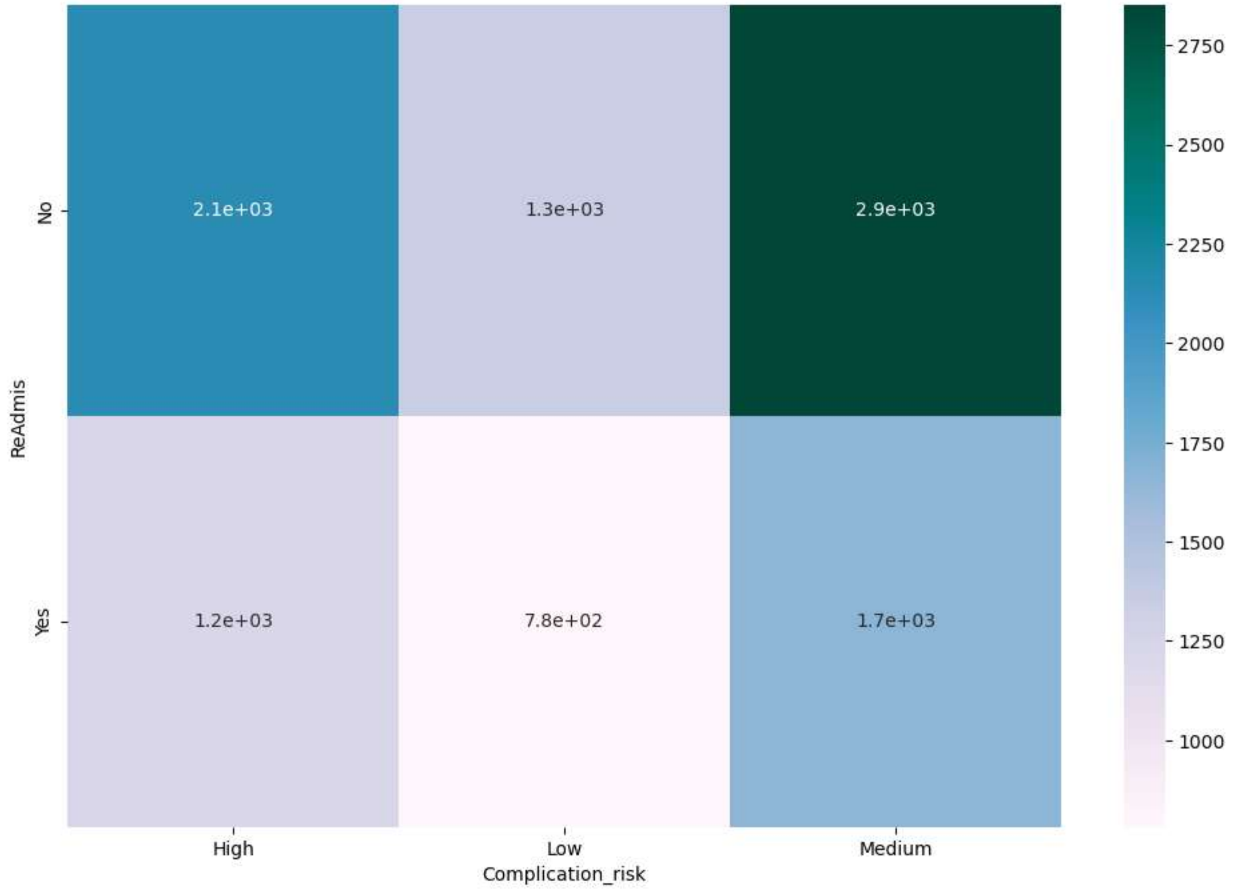
```
Complication_risk  High   Low  Medium    All
ReAdmis
No                 2135  1343    2853   6331
Yes                1223   782    1664   3669
All                3358  2125    4517  10000
```

In [9]:

```
#Obtain P-Value using code from WGU course materials
df.head()
contingency=pd.crosstab(df['ReAdmis'], df['Complication_risk'])
contingency
contingency_pct=pd.crosstab(df['ReAdmis'], df['Complication_risk'], normalize='index')
contingency_pct
plt.figure(figsize=(12,8))
sns.heatmap(contingency, annot=True, cmap="PuBuGn")
```

Out[9]:

```
<Axes: xlabel='Complication_risk', ylabel='ReAdmis'>
```



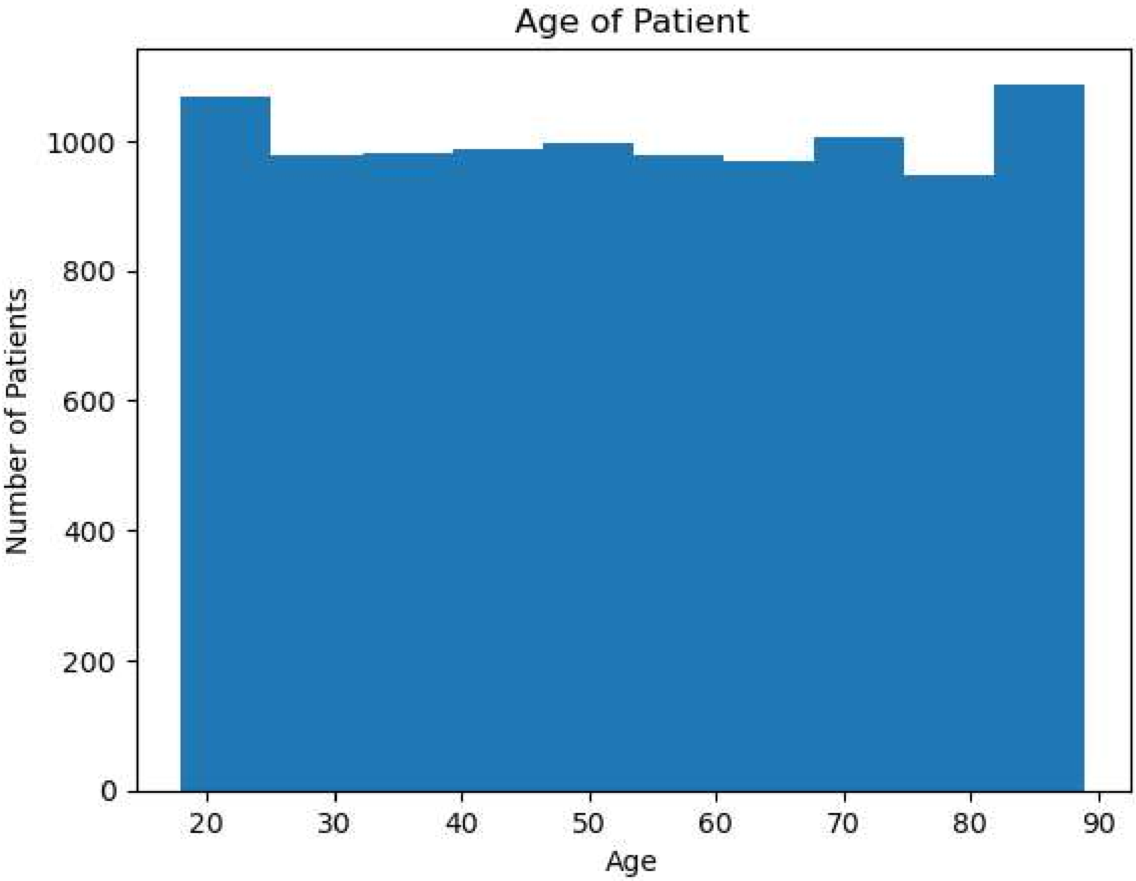
In [10]:
```
#Chi-Square test of independence using code from WGU course materials
c, p, dof, expected=chi2_contingency(contingency)
```

In [11]:
```
#Print P-value
print(p)
```

```
0.923567890607327
```

Univariate Statistics

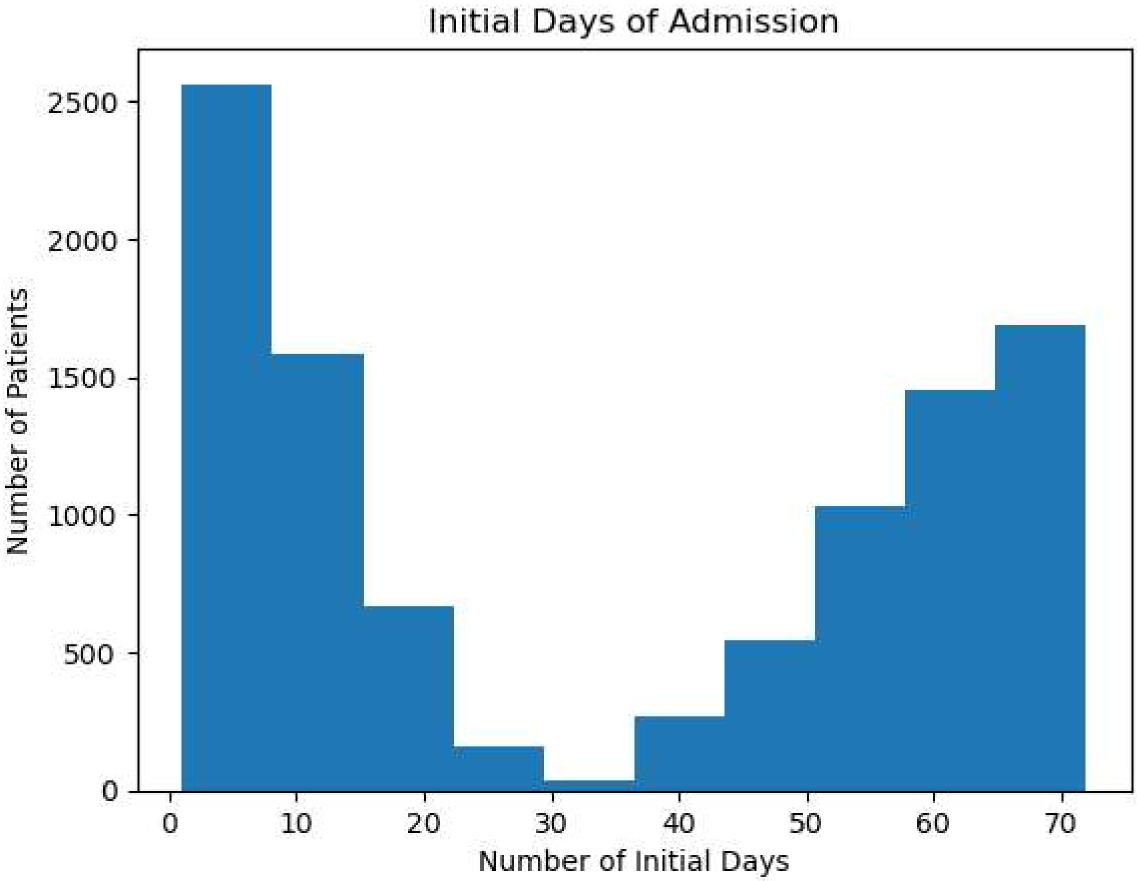
In [12]:
```
#Histogram of Age column - continuous
plt.hist(df['Age'])
plt.title("Age of Patient")
plt.xlabel("Age")
plt.ylabel("Number of Patients")
plt.show()
```



In [13]:
```
df.Age.describe()
```

Out[13]:
```
count    10000.000000
mean        53.511700
std         20.638538
min         18.000000
25%         36.000000
50%         53.000000
75%         71.000000
max         89.000000
Name: Age, dtype: float64
```

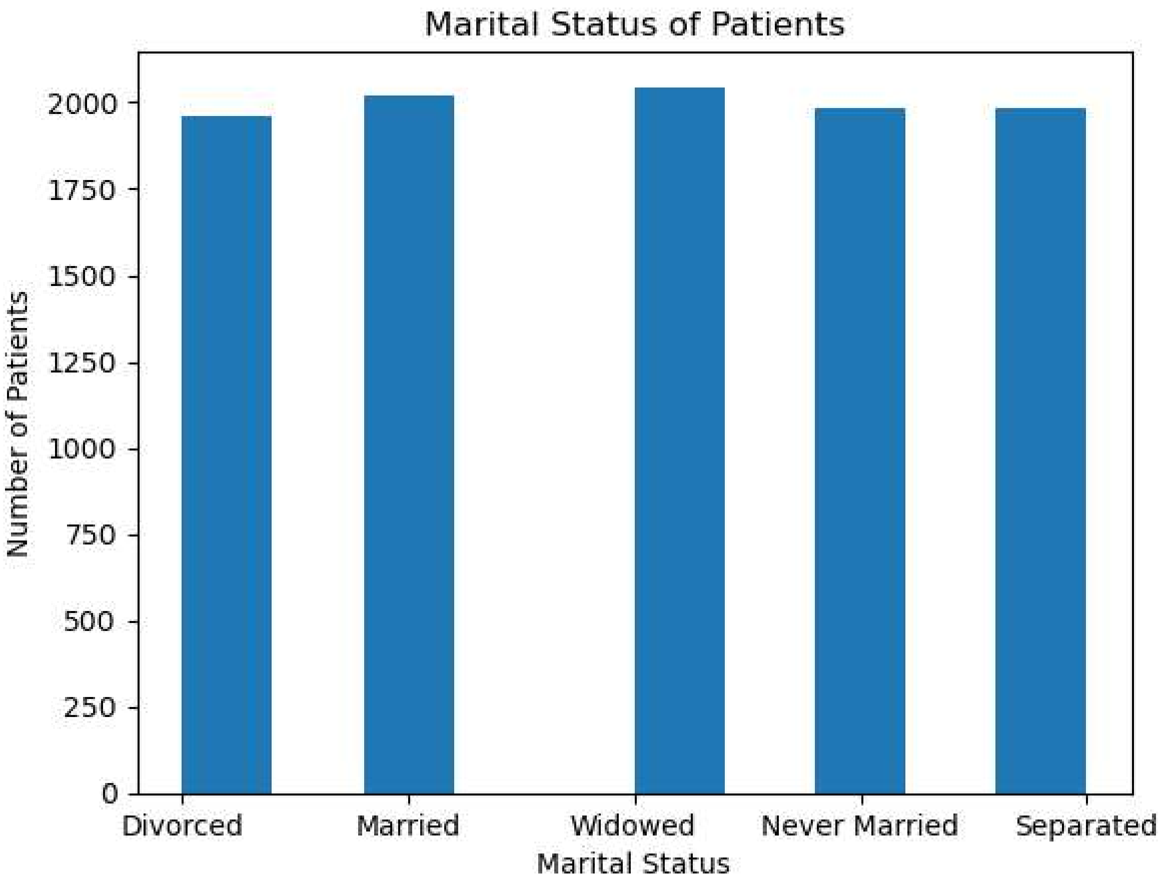
In [14]:
```
#Histogram of Initial_days column - continuous
plt.hist(df['Initial_days'])
plt.title("Initial Days of Admission")
plt.xlabel("Number of Initial Days")
plt.ylabel("Number of Patients")
plt.show()
```



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

Out[15]:
```
count    10000.000000
mean        34.455299
std         26.309341
min          1.001981
25%          7.896215
50%         35.836244
75%         61.161020
max         71.981490
Name: Initial_days, dtype: float64
```

In [16]:
```
#Histogram of Marital - categorical
plt.hist(df['Marital'])
plt.title("Marital Status of Patients")
plt.xlabel("Marital Status")
plt.ylabel("Number of Patients")
plt.show()
```



In [17]:

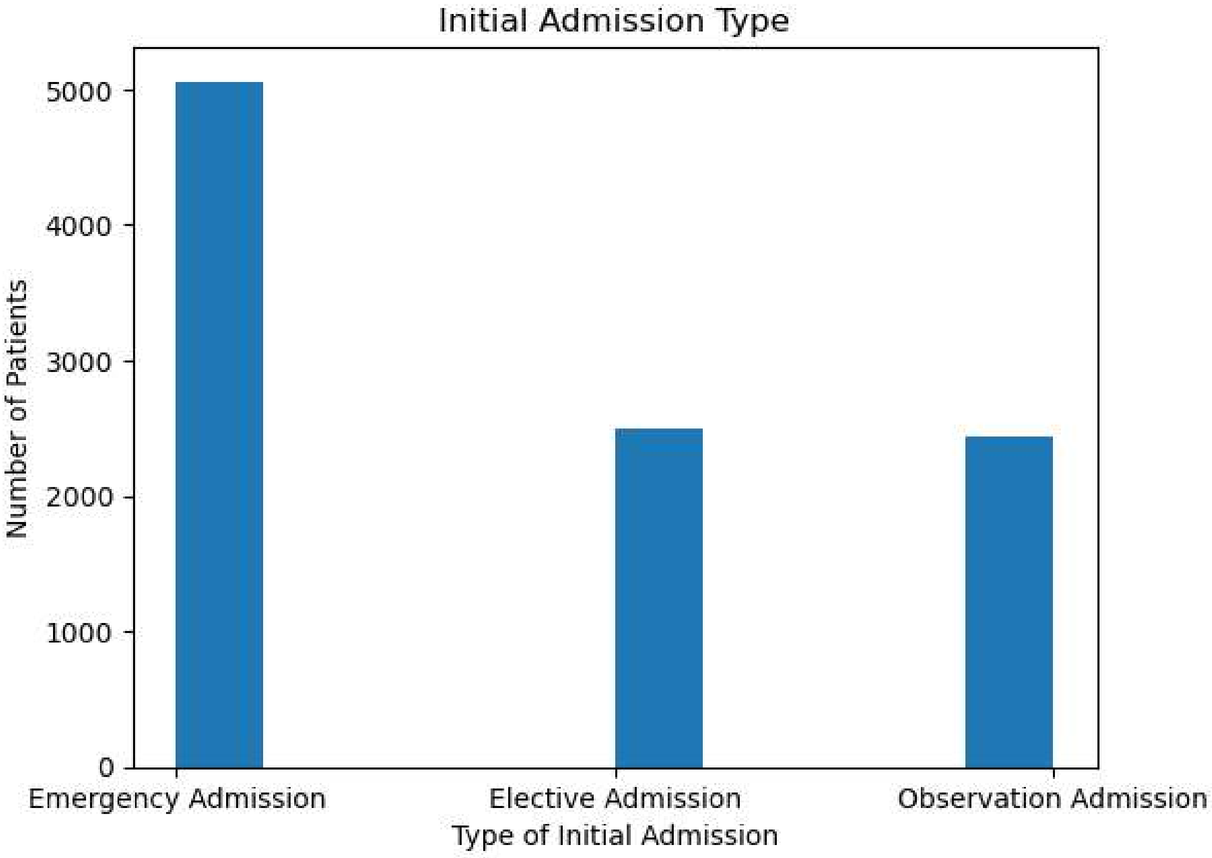
```
df.Marital.describe()
```

Out[17]:

```
count       10000
unique          5
top       Widowed
freq         2045
Name: Marital, dtype: object
```

In [18]:

```
#Histogram of Initial_admin - categorical
plt.hist(df['Initial_admin'])
plt.title("Initial Admission Type")
plt.xlabel("Type of Initial Admission")
plt.ylabel("Number of Patients")
plt.show()
```

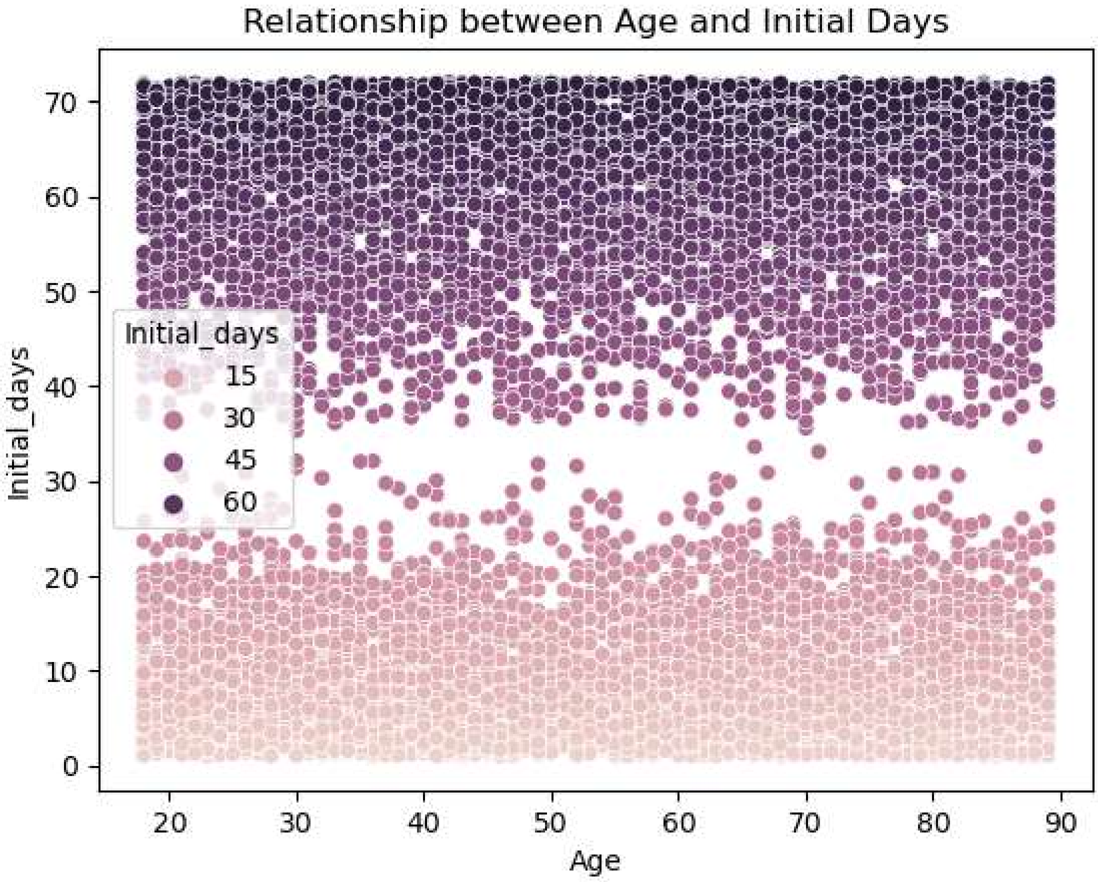


In [19]:
```
df.Initial_admin.describe()
```

Out[19]:
```
count                   10000
unique                      3
top       Emergency Admission
freq                     5060
Name: Initial_admin, dtype: object
```

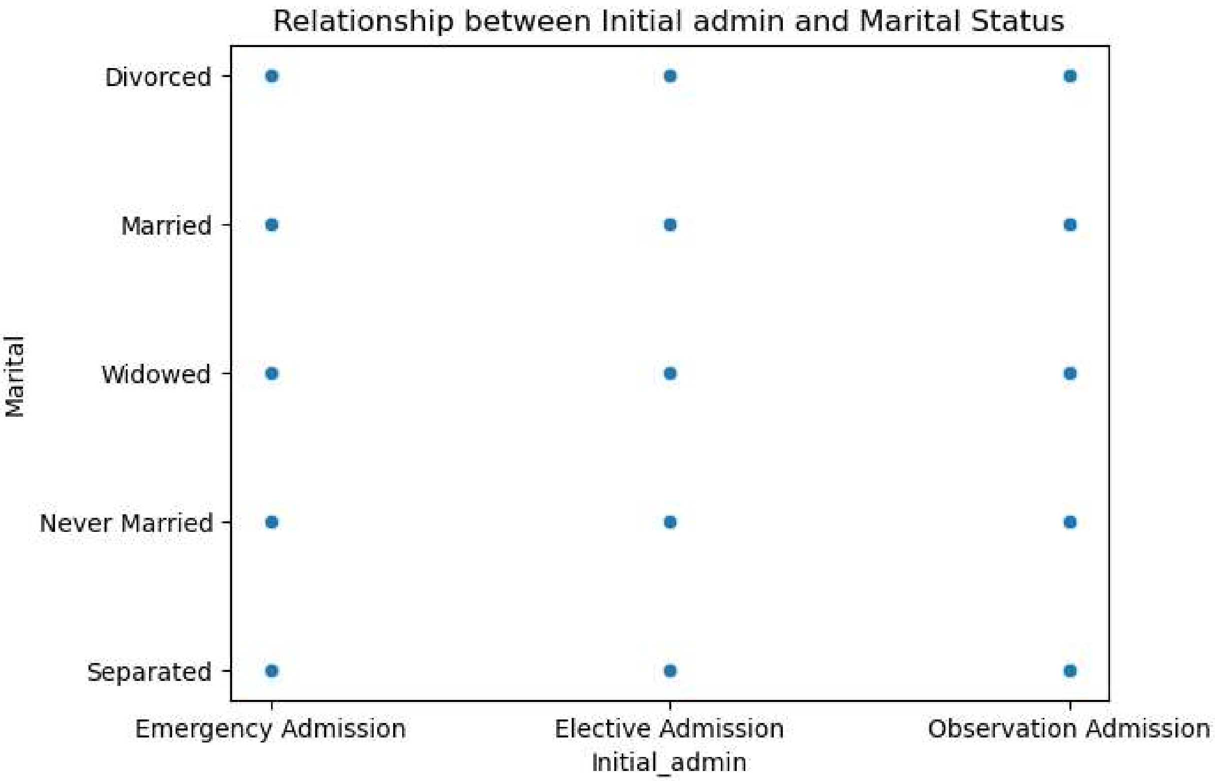
Bivariate Statistics

In [20]:
```
#Scatterplot of Age and Initial_days - continuous
sns.scatterplot(x='Age', y='Initial_days', hue='Initial_days', data=df).set(title='Rel
plt.show()
```

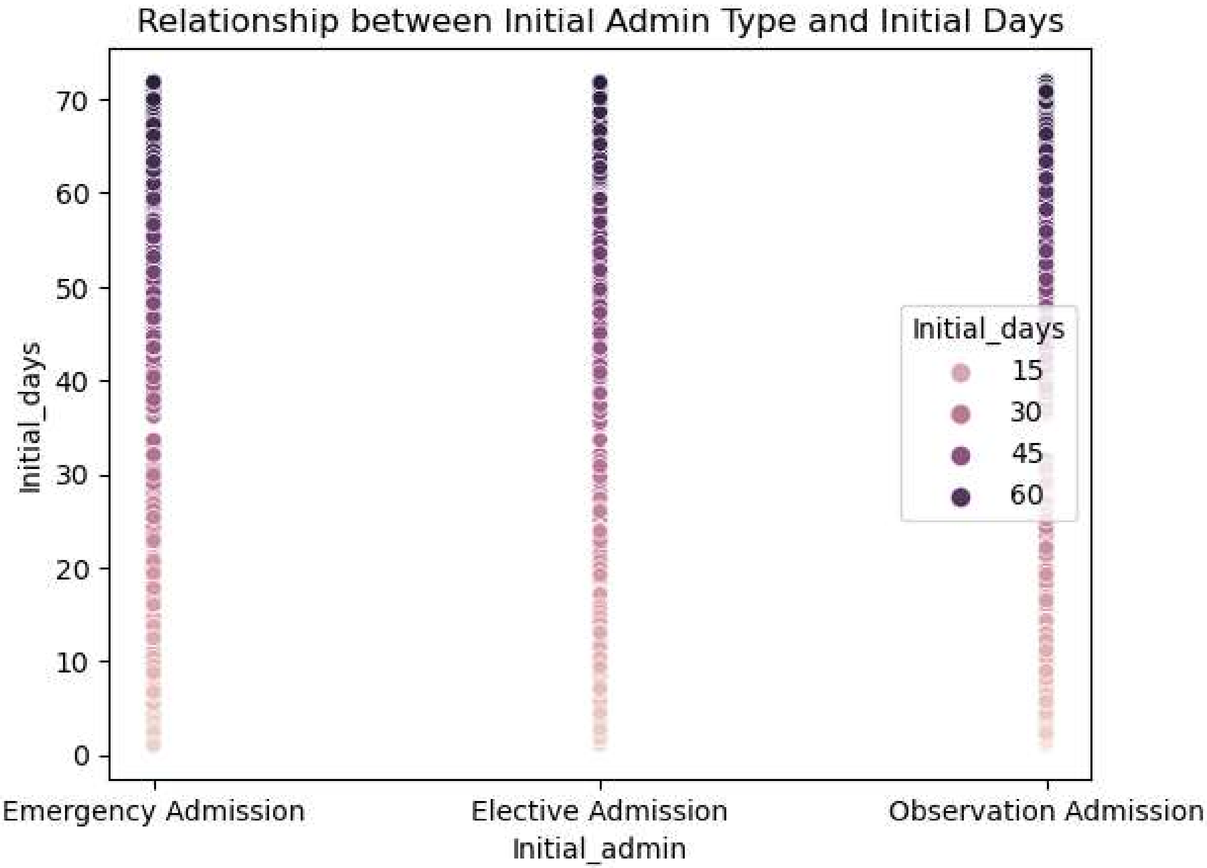


In [21]:
```
#Scatter plot of Initial_admin and Marital - categorical
sns.scatterplot(x='Initial_admin', y='Marital', data=df).set(title='Relationship betwe
```

Out[21]:
```
[Text(0.5, 1.0, 'Relationship between Initial admin and Marital Status')]
```

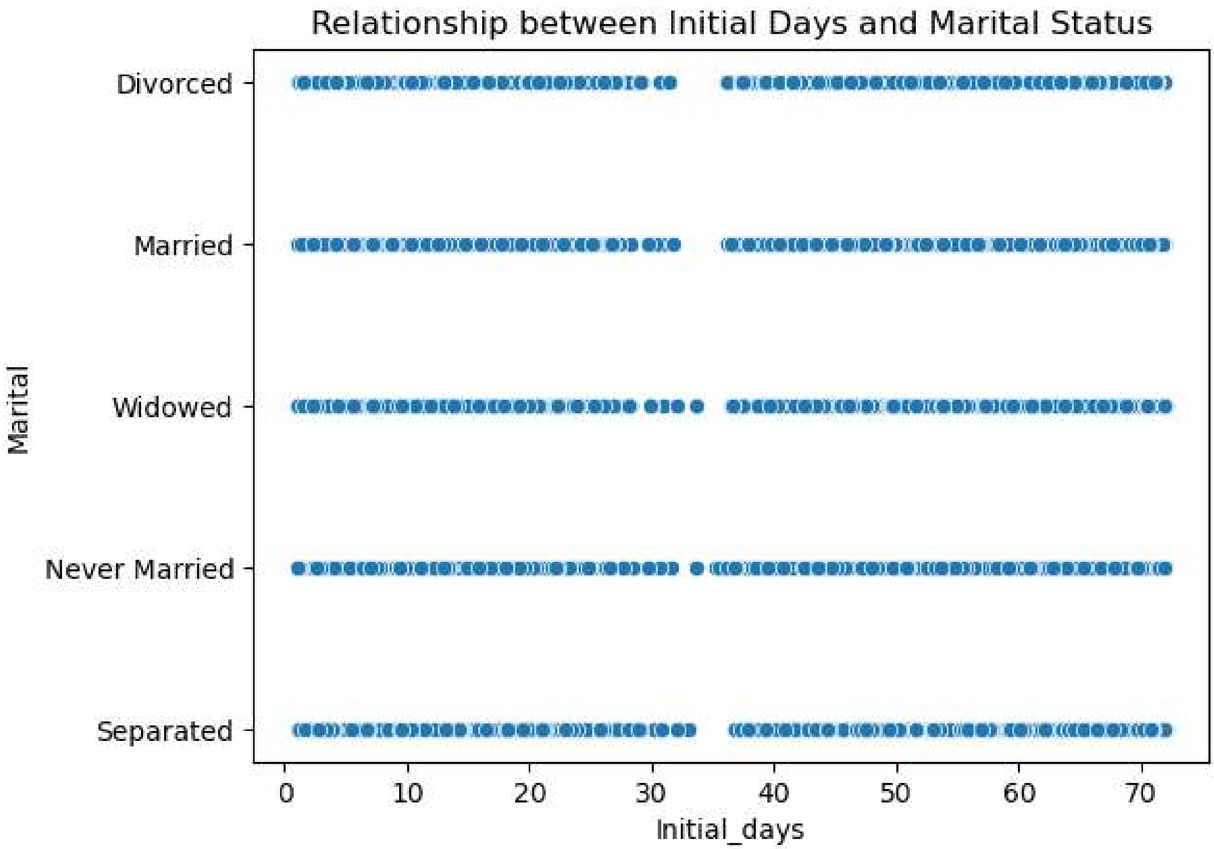


In [22]:
```
#Scatterplot of Initial_admin and Initial_days - categorical and continuous
sns.scatterplot(x='Initial_admin', y='Initial_days', hue='Initial_days', data=df).set(
plt.show()
```



In [23]:
```
#Scatter plot of Initial_admin and Marital - categorical
sns.scatterplot(x='Initial_days', y='Marital', data=df).set(title='Relationship betwee
```

Out[23]:
```
[Text(0.5, 1.0, 'Relationship between Initial Days and Marital Status')]
```


Relationship between Initial Days and Marital Status
Marital
Divorced
Married
Widowed
Never Married
Separated
0
10
20
30
40
50
60
70
Initial_days

```
In [ ]:
```