In [17]:

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

In [2]:

```
df = pd.read_csv(r'C:\Users\nikde\Downloads\hospital_readmission.csv')
```

In [3]:

```
df.head(5)
```

Out[3]:

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	

5 rows × 50 columns

In [4]:

```
df.info()
```

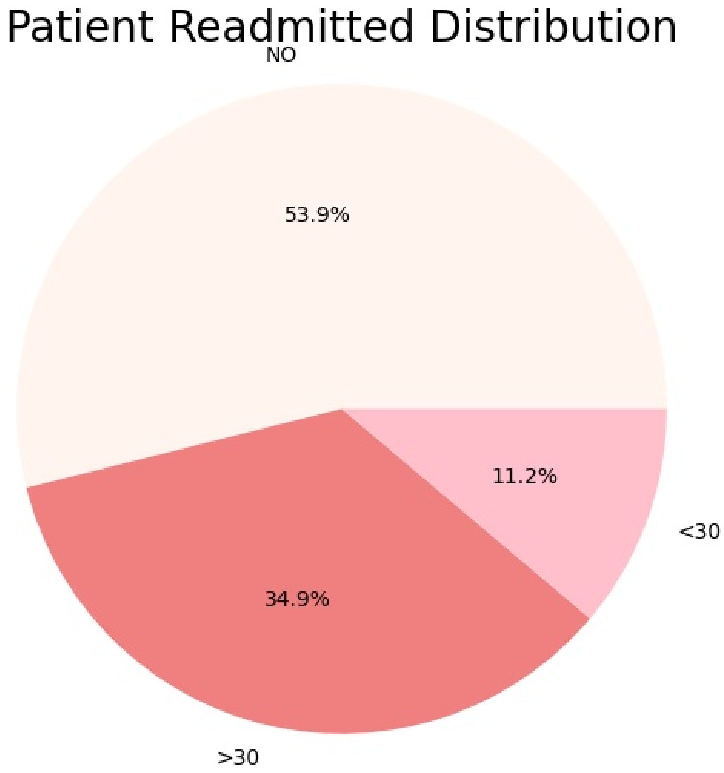
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 50 columns):
 #   Column                    Non-Null Count   Dtype 
---  ------                    --------------   ----- 
 0   encounter_id              101766 non-null  int64 
 1   patient_nbr               101766 non-null  int64 
 2   race                      101766 non-null  object
 3   gender                    101766 non-null  object
 4   age                       101766 non-null  object
 5   weight                    101766 non-null  object
 6   admission_type_id         101766 non-null  int64 
 7   discharge_disposition_id  101766 non-null  int64 
 8   admission_source_id       101766 non-null  int64 
 9   time_in_hospital          101766 non-null  int64 
 10  payer_code                101766 non-null  object
 11  medical_specialty         101766 non-null  object
 12  num_lab_procedures        101766 non-null  int64 
 13  num_procedures            101766 non-null  int64 
 14  num_medications           101766 non-null  int64 
 15  number_outpatient         101766 non-null  int64 
 16  number_emergency          101766 non-null  int64 
 17  number_inpatient          101766 non-null  int64 
 18  diag_1                    101766 non-null  object
 19  diag_2                    101766 non-null  object
 20  diag_3                    101766 non-null  object
 21  number_diagnoses          101766 non-null  int64 
 22  max_glu_serum             5346 non-null    object
 23  A1Cresult                 17018 non-null   object
 24  metformin                 101766 non-null  object
 25  repaglinide               101766 non-null  object
 26  nateglinide               101766 non-null  object
 27  chlorpropamide            101766 non-null  object
 28  glimepiride               101766 non-null  object
 29  acetohexamide             101766 non-null  object
 30  glipizide                 101766 non-null  object
 31  glyburide                 101766 non-null  object
 32  tolbutamide               101766 non-null  object
 33  pioglitazone              101766 non-null  object
 34  rosiglitazone             101766 non-null  object
 35  acarbose                  101766 non-null  object
 36  miglitol                  101766 non-null  object
 37  troglitazone              101766 non-null  object
 38  tolazamide                101766 non-null  object
 39  examide                   101766 non-null  object
 40  citoglipton               101766 non-null  object
 41  insulin                   101766 non-null  object
 42  glyburide-metformin       101766 non-null  object
 43  glipizide-metformin       101766 non-null  object
 44  glimepiride-pioglitazone  101766 non-null  object
 45  metformin-rosiglitazone   101766 non-null  object
 46  metformin-pioglitazone    101766 non-null  object
 47  change                    101766 non-null  object
 48  diabetesMed               101766 non-null  object
 49  readmitted                101766 non-null  object
dtypes: int64(13), object(37)
memory usage: 38.8+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

	encounter_id	patient_nbr	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_p
count	1.017660e+05	1.017660e+05	101766.000000	101766.000000	101766.000000	101766.000000	1017
mean	1.652016e+08	5.433040e+07	2.024006	3.715642	5.754437	4.395987	
std	1.026403e+08	3.869636e+07	1.445403	5.280166	4.064081	2.985108	
min	1.252200e+04	1.350000e+02	1.000000	1.000000	1.000000	1.000000	
25%	8.496119e+07	2.341322e+07	1.000000	1.000000	1.000000	2.000000	
50%	1.523890e+08	4.550514e+07	1.000000	1.000000	7.000000	4.000000	
75%	2.302709e+08	8.754595e+07	3.000000	4.000000	7.000000	6.000000	
max	4.438672e+08	1.895026e+08	8.000000	28.000000	25.000000	14.000000	1

In [6]:
```
readmitted_counts = df['readmitted'].value_counts(normalize=True)
readmitted_counts
plt.figure(figsize=(6, 6))
plt.pie(readmitted_counts.values,
        labels=readmitted_counts.index,
        autopct='%1.1f%%',
        colors=['seashell', 'lightcoral', 'pink'])

plt.title('Patient Readmitted Distribution', size = 20)
plt.axis('equal')
plt.show()
```



So, here we get clear idea that 46.1 percent of patient get readmitted and out of readmitted patients 11.2 % get readmitted within a month and rest 34.9 % get readmitted but after a month.

.

Ques 2 :- How does patient age impact readmission rates ??

What is the distribution of patients across different medical specialties?

In [7]:

```
Speciality_counts = df['medical_specialty'].value_counts(normalize=True)
Speciality_counts
```

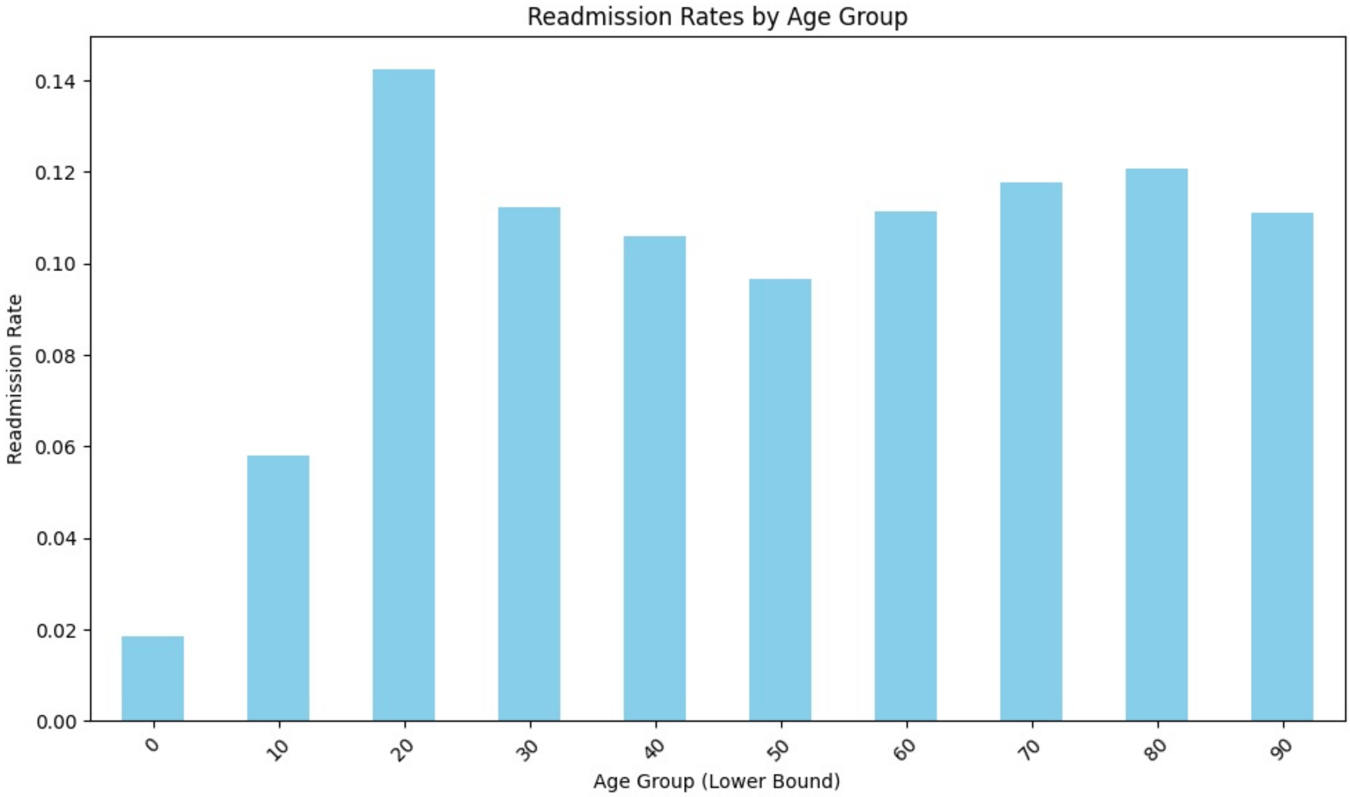
Out[7]:

```
medical_specialty
not known                           0.490822
InternalMedicine                    0.143810
Emergency/Trauma                    0.074337
Family/GeneralPractice              0.073109
Cardiology                          0.052591
                                      ...
SportsMedicine                      0.000010
Speech                              0.000010
Perinatology                        0.000010
Neurophysiology                     0.000010
Pediatrics-InfectiousDiseases       0.000010
Name: proportion, Length: 73, dtype: float64
```

How does patient age impact readmission rates?

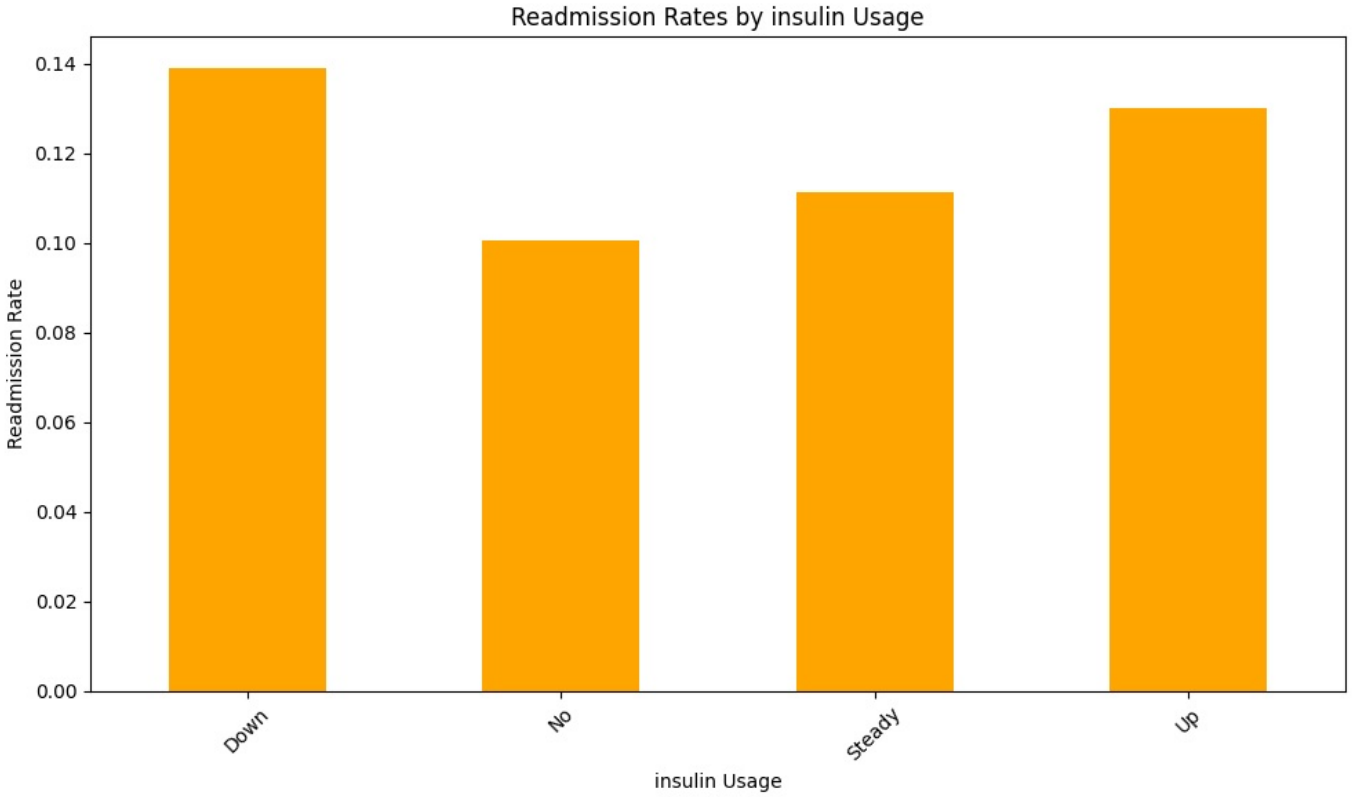
In [9]:

```
# Step 1: Data cleaning (example)
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 0, 'NO': 0})  # Encode readmission as 1 (yes) and
df['age'] = df['age'].str.strip('[]').str.split('-').str[0].astype(int)  # Convert age ranges to numerical lowe

# Step 2: Group by age ranges
age_groups = df.groupby('age')['readmitted'].mean()  # Calculate mean readmission rate by age group

# Step 3: Plotting
plt.figure(figsize=(10, 6))
age_groups.plot(kind='bar', color='skyblue')
plt.title('Readmission Rates by Age Group')
plt.xlabel('Age Group (Lower Bound)')
plt.ylabel('Readmission Rate')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



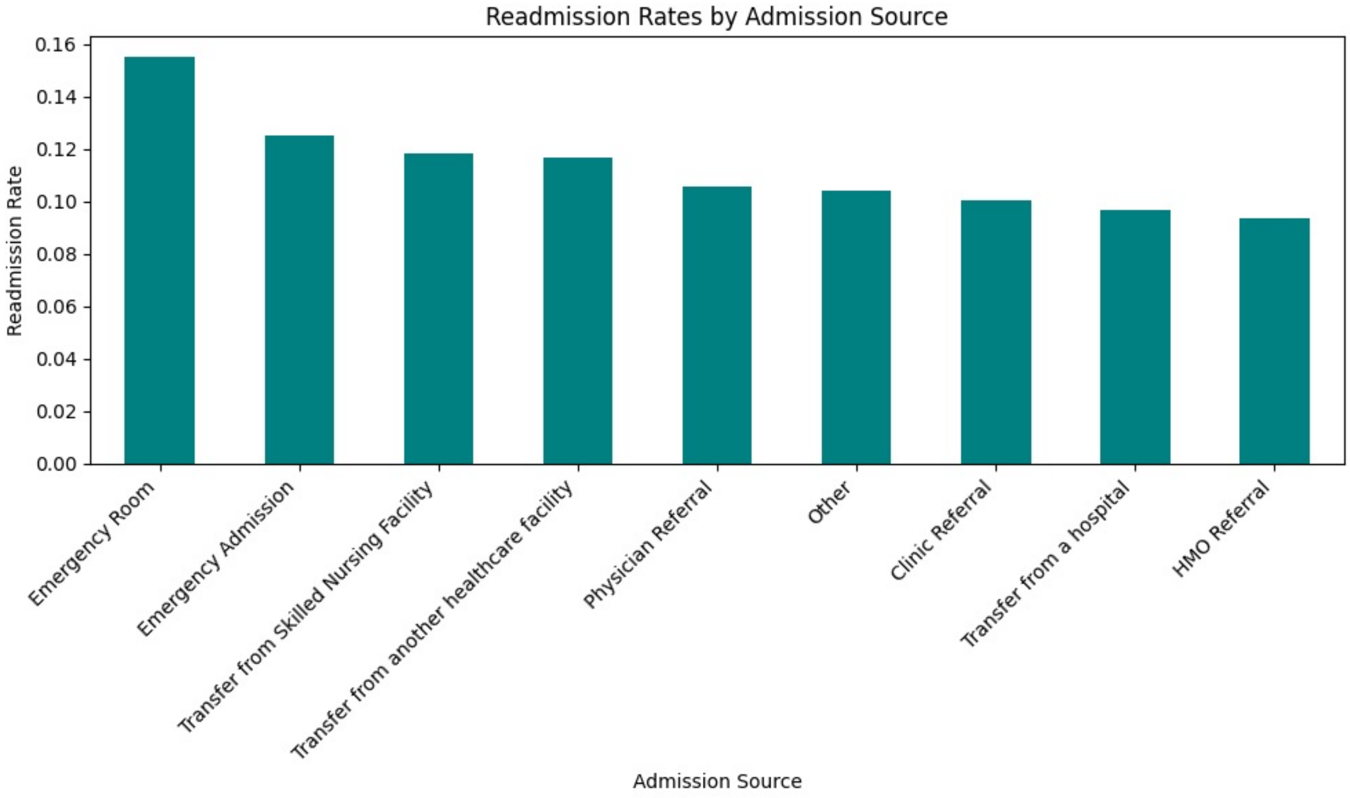
In [22]:

```
# Step 1: Preprocessing
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 1, 'NO': 0})  # Encode readmitted as 1 (yes) and (

# Step 2: Focus on procedure and medication columns
procedure_medication_cols = [
    'num_lab_procedures', 'num_procedures', 'num_medications',
    'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
    'glimepiride', 'acetohexamide', 'glipizide', 'glyburide',
    'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose',
    'miglitol', 'troglitazone', 'tolazamide', 'insulin'
]

# Step 3: Aggregate readmission rates
results = {}
for col in procedure_medication_cols:
    if df[col].dtype == 'object':  # If categorical (e.g., medications)
        rates = df.groupby(col)['readmitted'].mean()  # Calculate mean readmission rate per category
        results[col] = rates
    else:  # If numerical (e.g., counts of procedures)
        correlation = df[[col, 'readmitted']].corr().iloc[0, 1]  # Correlation with readmission
        results[col] = correlation

# Step 4: Visualization (Example for a medication column like insulin)
medication = 'insulin'
readmission_rates = results[medication]

plt.figure(figsize=(10, 6))
readmission_rates.plot(kind='bar', color='orange')
plt.title(f'Readmission Rates by {medication} Usage')
plt.xlabel(f'{medication} Usage')
plt.ylabel('Readmission Rate')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [13]:

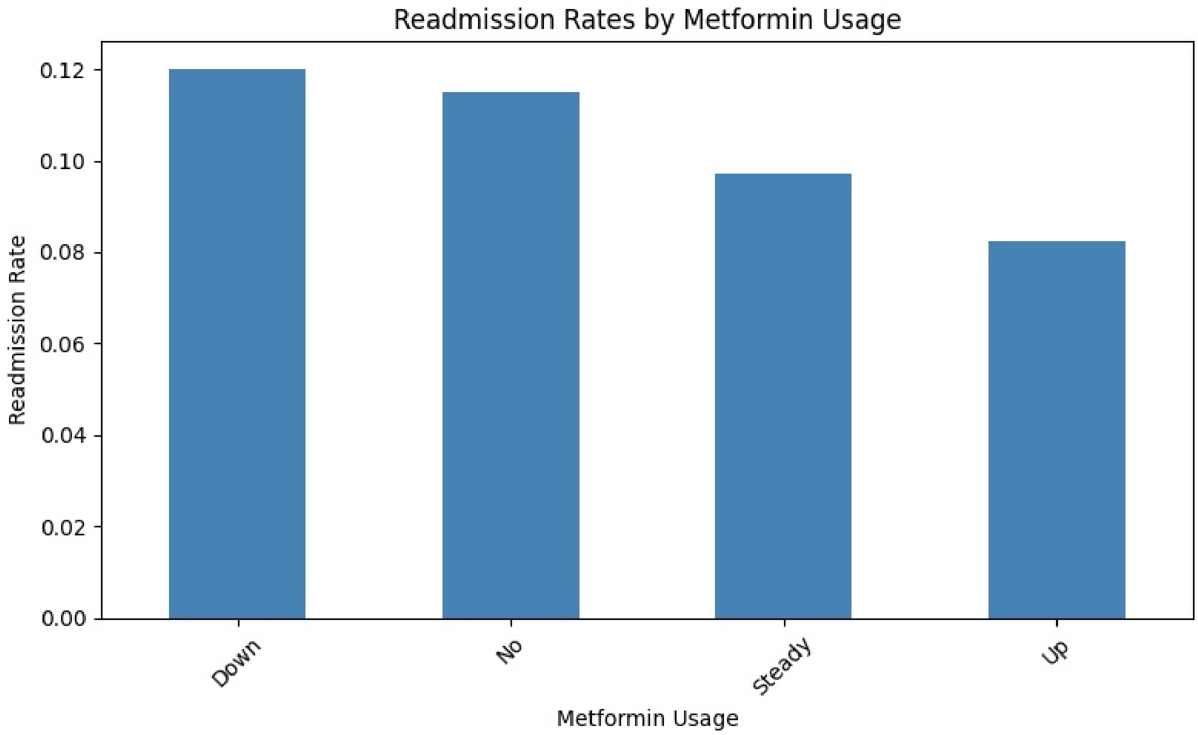
```
# Step 1: Preprocess the data
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 1, 'NO': 0})  # Encode readmitted as 1 (yes) and (

# Optional: Map admission_source_id to descriptive labels if a mapping is available
admission_source_mapping = {
    1: 'Physician Referral', 2: 'Clinic Referral', 3: 'Emergency Room',
    4: 'Transfer from a hospital', 5: 'Transfer from Skilled Nursing Facility',
    6: 'HMO Referral', 7: 'Transfer from another healthcare facility',
    8: 'Emergency Admission', 9: 'Other'
}
df['admission_source'] = df['admission_source_id'].map(admission_source_mapping)

# Step 2: Group by admission source and calculate readmission rates
readmission_rates = df.groupby('admission_source')['readmitted'].mean()

# Step 3: Visualize the results
plt.figure(figsize=(10, 6))
readmission_rates.sort_values(ascending=False).plot(kind='bar', color='teal')
plt.title('Readmission Rates by Admission Source')
plt.xlabel('Admission Source')
plt.ylabel('Readmission Rate')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Display the readmission rates for reference
print(readmission_rates.sort_values(ascending=False))
```

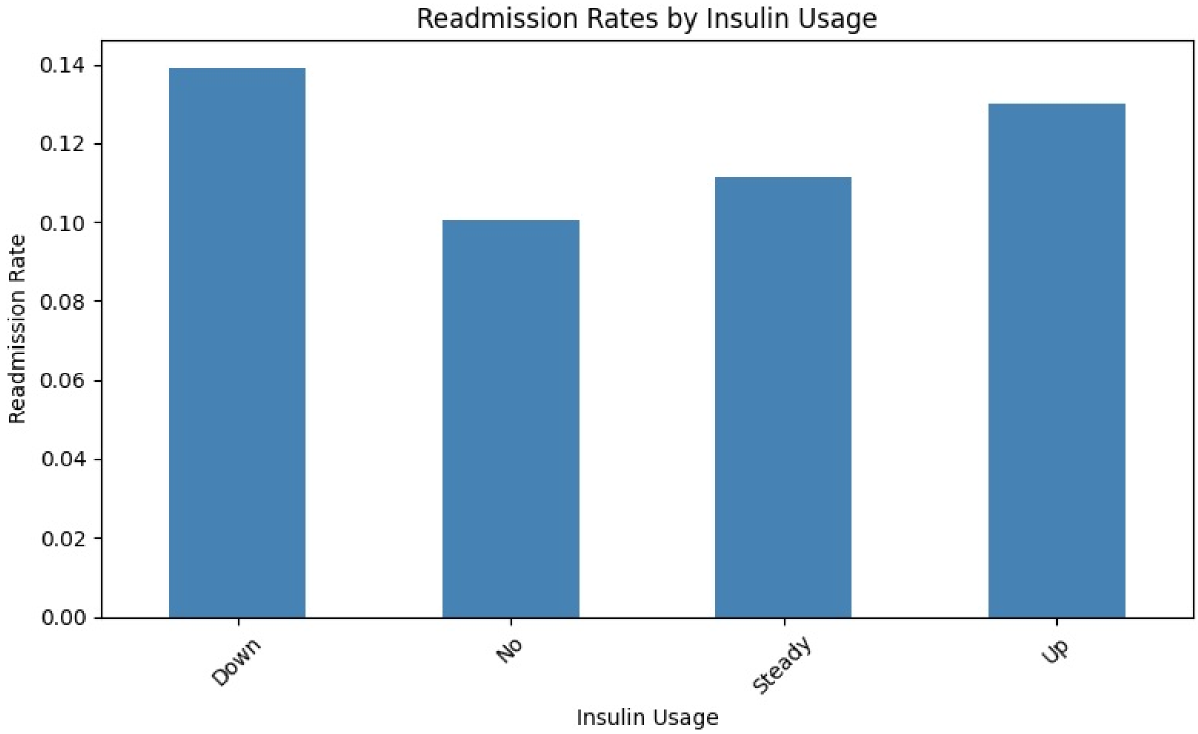


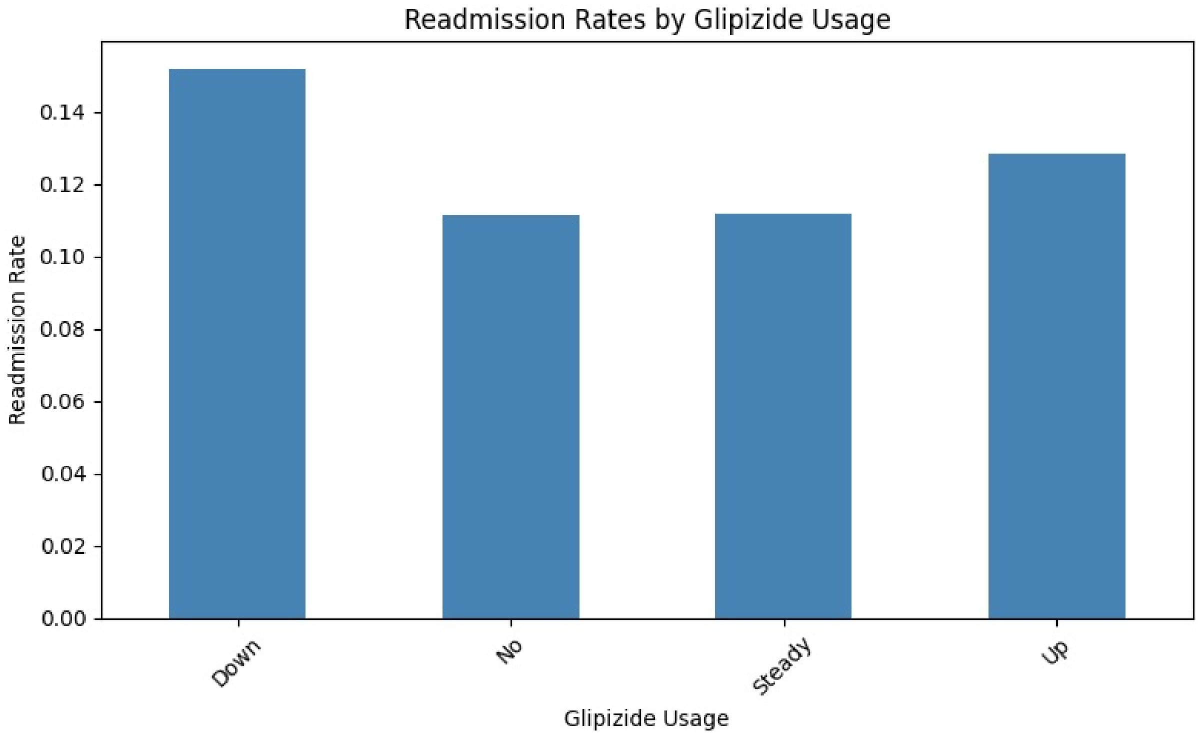
```
admission_source
Emergency Room                               0.155080
Emergency Admission                          0.125000
Transfer from Skilled Nursing Facility       0.118129
Transfer from another healthcare facility    0.116882
Physician Referral                           0.105868
Other                                        0.104000
Clinic Referral                              0.100543
Transfer from a hospital                     0.096956
HMO Referral                                 0.093640
Name: readmitted, dtype: float64
```

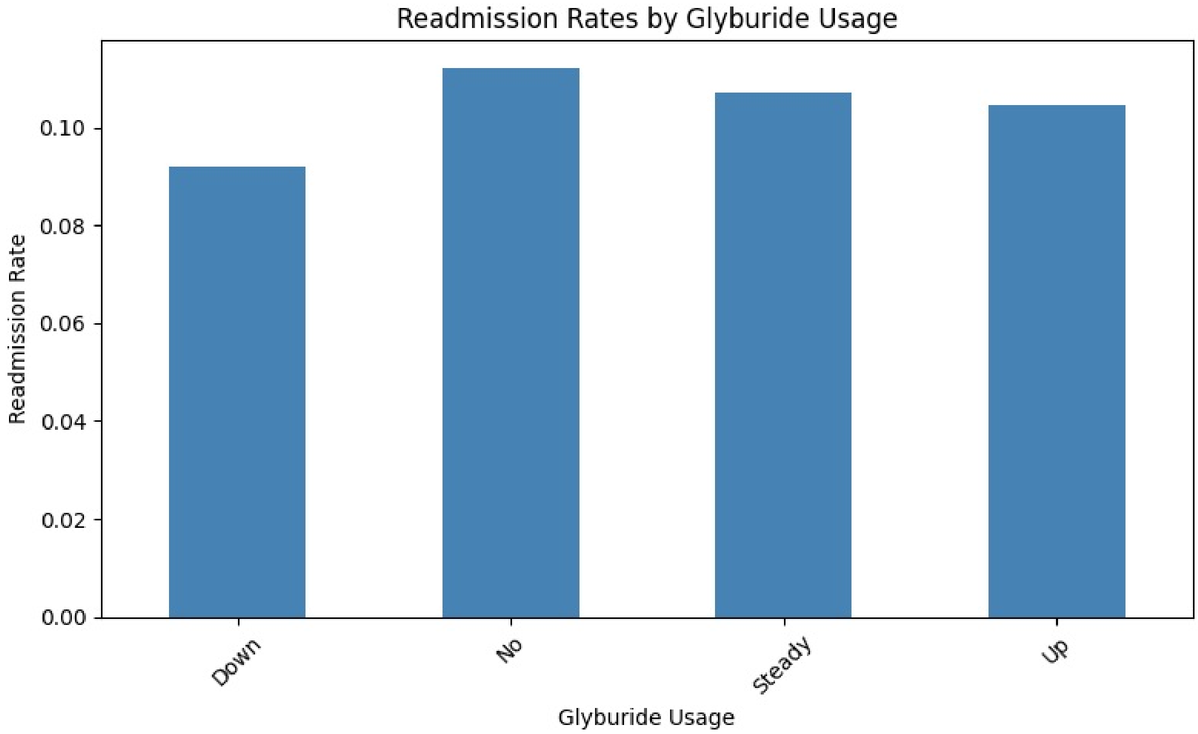
In [15]:

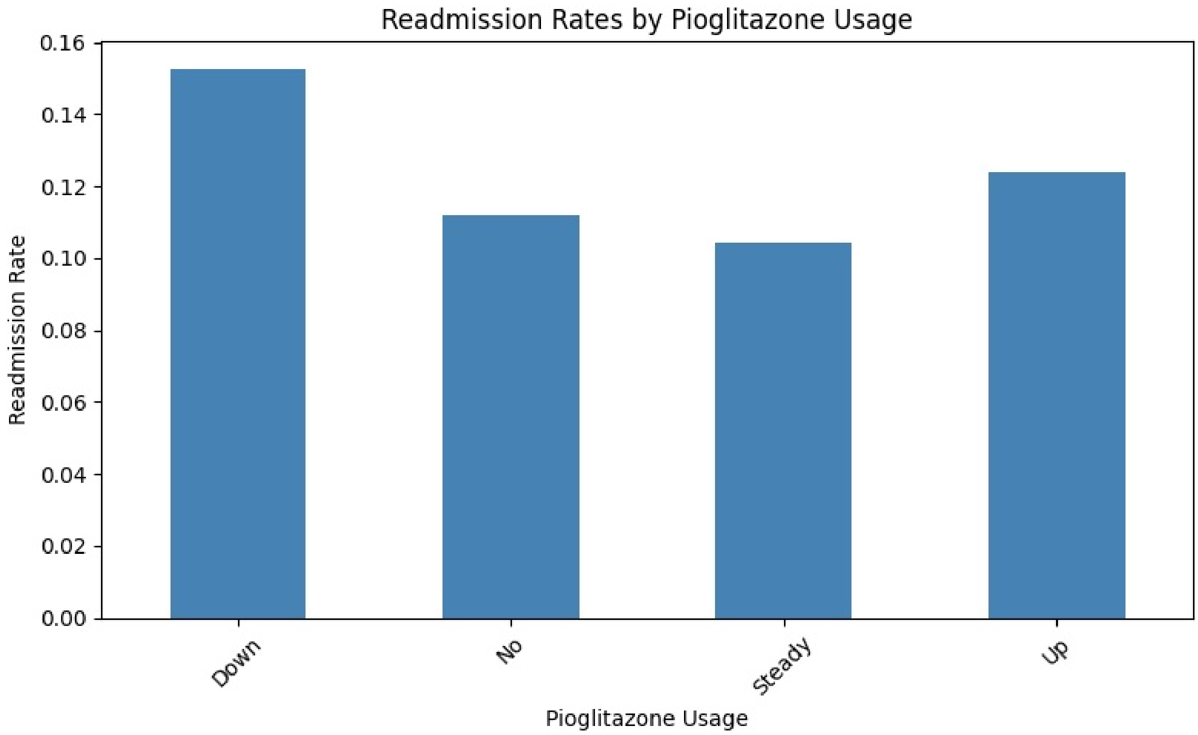
```python
# Step 1: Preprocessing
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 1, 'NO': 0})  # Encode readmitted as 1 or 0

# List of medication columns to analyze
medications = ['metformin', 'insulin', 'glipizide', 'glyburide', 'pioglitazone']

# Step 2: Calculate readmission rates for each medication
medication_effectiveness = {}
for med in medications:
    rates = df.groupby(med)['readmitted'].mean()  # Mean readmission rate by usage category
    medication_effectiveness[med] = rates

# Step 3: Visualization
for med, rates in medication_effectiveness.items():
    plt.figure(figsize=(8, 5))
    rates.plot(kind='bar', color='steelblue')
    plt.title(f'Readmission Rates by {med.capitalize()} Usage')
    plt.xlabel(f'{med.capitalize()} Usage')
    plt.ylabel('Readmission Rate')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# Step 4: Summary Table
summary_table = pd.DataFrame(medication_effectiveness).T
summary_table.columns = ['No', 'Steady', 'Up', 'Down']  # Adjust based on actual categories in the dataset
print("Medication Effectiveness Summary:")
print(summary_table)
```


Readmission Rates by Metformin Usage
Readmission Rate
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Down
No
Steady
Up
Metformin Usage


Readmission Rates by Insulin Usage
Readmission Rate
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Down
No
Steady
Up
Insulin Usage


Readmission Rates by Glipizide Usage
Readmission Rate
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Down
No
Steady
Up
Glipizide Usage

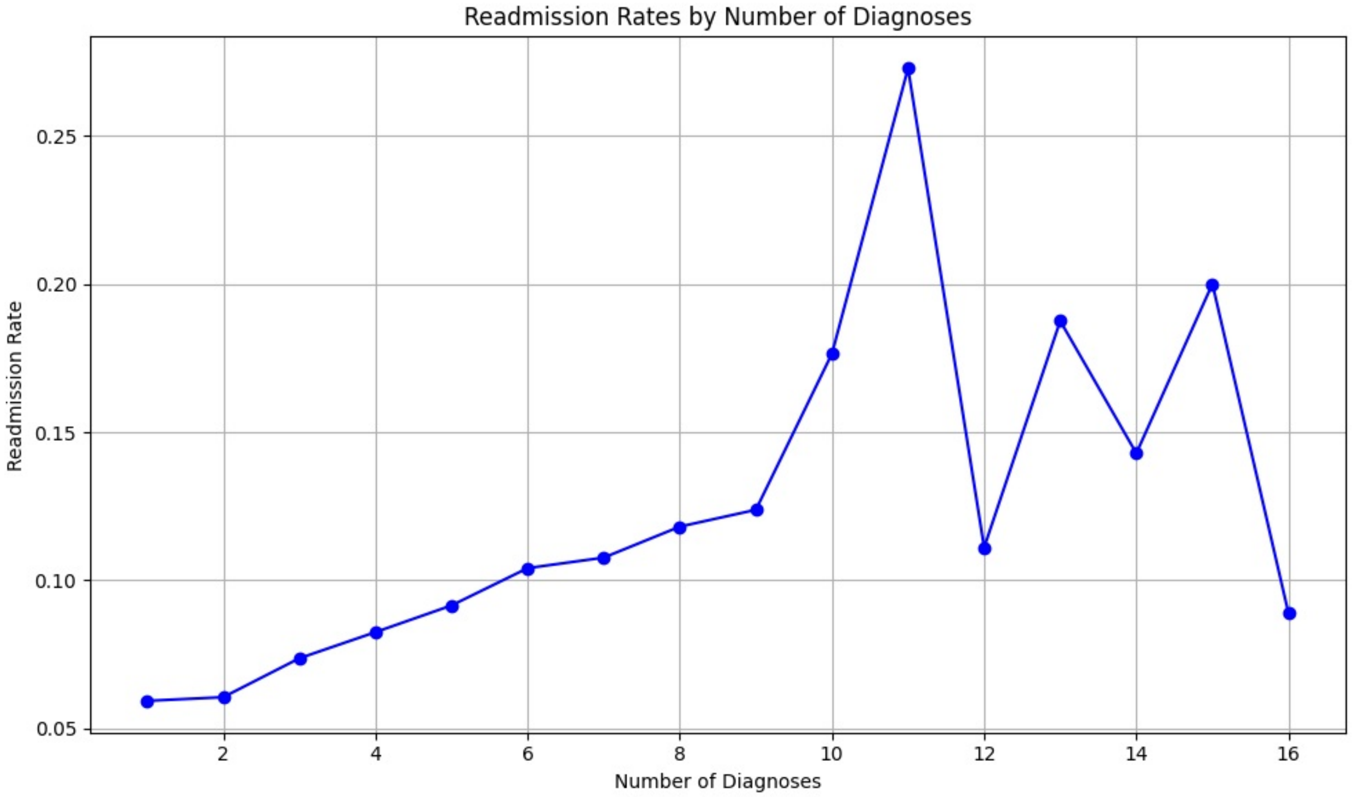

Readmission Rates by Glyburide Usage
Readmission Rate
0.10
0.08
0.06
0.04
0.02
0.00
Down
No
Steady
Up
Glyburide Usage



```
Medication Effectiveness Summary:
                    No    Steady        Up      Down
metformin     0.120000  0.115165  0.097133  0.082474
insulin       0.138975  0.100374  0.111284  0.129905
glipizide     0.151786  0.111192  0.111659  0.128571
glyburide     0.092199  0.112220  0.107289  0.104680
pioglitazone  0.152542  0.112063  0.104214  0.123932
```

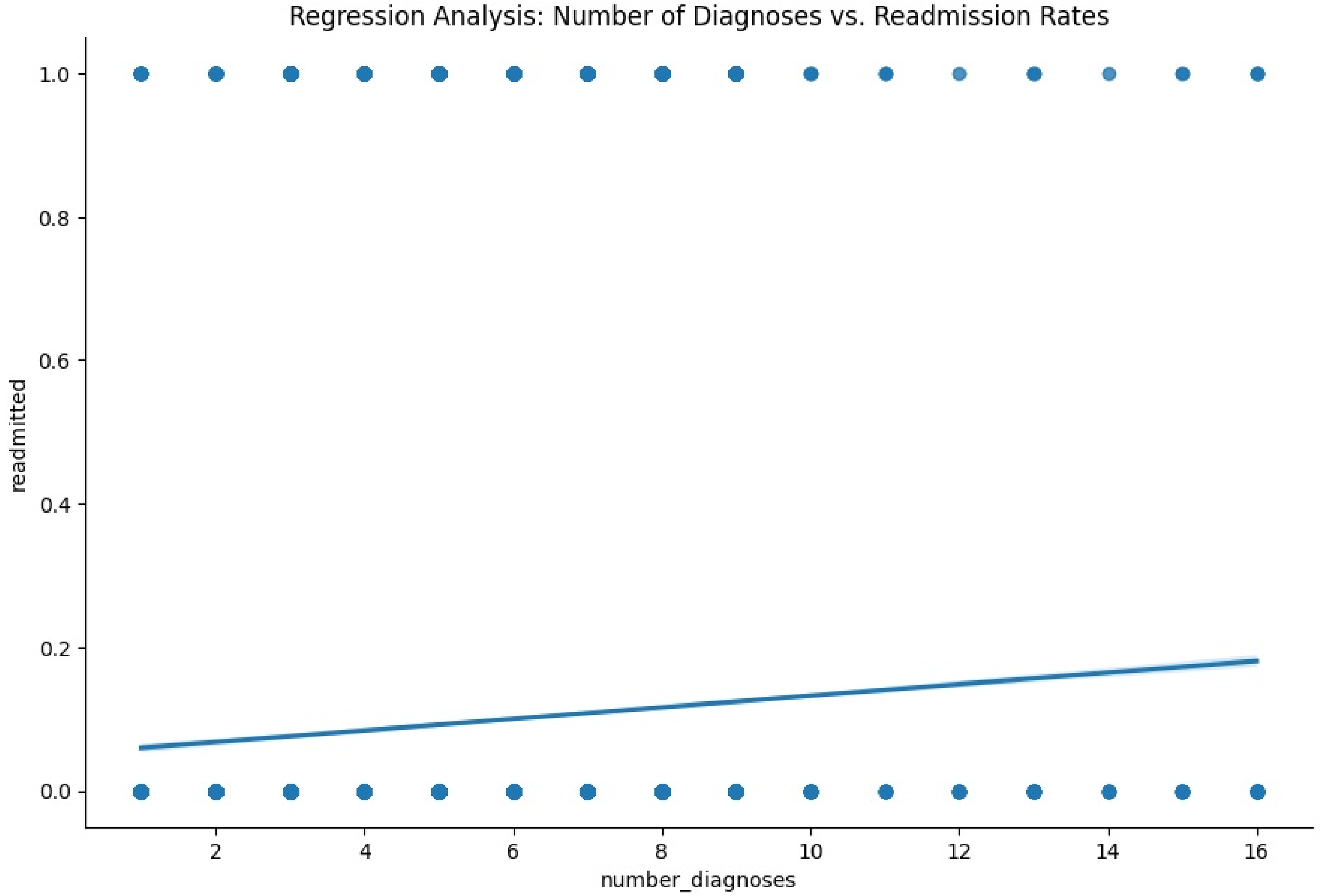
In [18]:

```
# Step 1: Preprocessing
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 1, 'NO': 0})  # Encode readmitted as 1 or 0

# Step 2: Calculate readmission rates by number of diagnoses
diagnosis_readmission_rates = df.groupby('number_diagnoses')['readmitted'].mean()

# Step 3: Visualization
plt.figure(figsize=(10, 6))
plt.plot(diagnosis_readmission_rates.index, diagnosis_readmission_rates.values, marker='o', color='blue')
plt.title('Readmission Rates by Number of Diagnoses')
plt.xlabel('Number of Diagnoses')
plt.ylabel('Readmission Rate')
plt.grid(True)
plt.tight_layout()
plt.show()

# Step 4: Correlation analysis
correlation = df[['number_diagnoses', 'readmitted']].corr().iloc[0, 1]
print(f"Correlation between number of diagnoses and readmission rates: {correlation:.2f}")

# Optional: Regression analysis
sns.lmplot(x='number_diagnoses', y='readmitted', data=df, height=6, aspect=1.5)
plt.title('Regression Analysis: Number of Diagnoses vs. Readmission Rates')
plt.show()
```



```
Correlation between number of diagnoses and readmission rates: 0.05
```

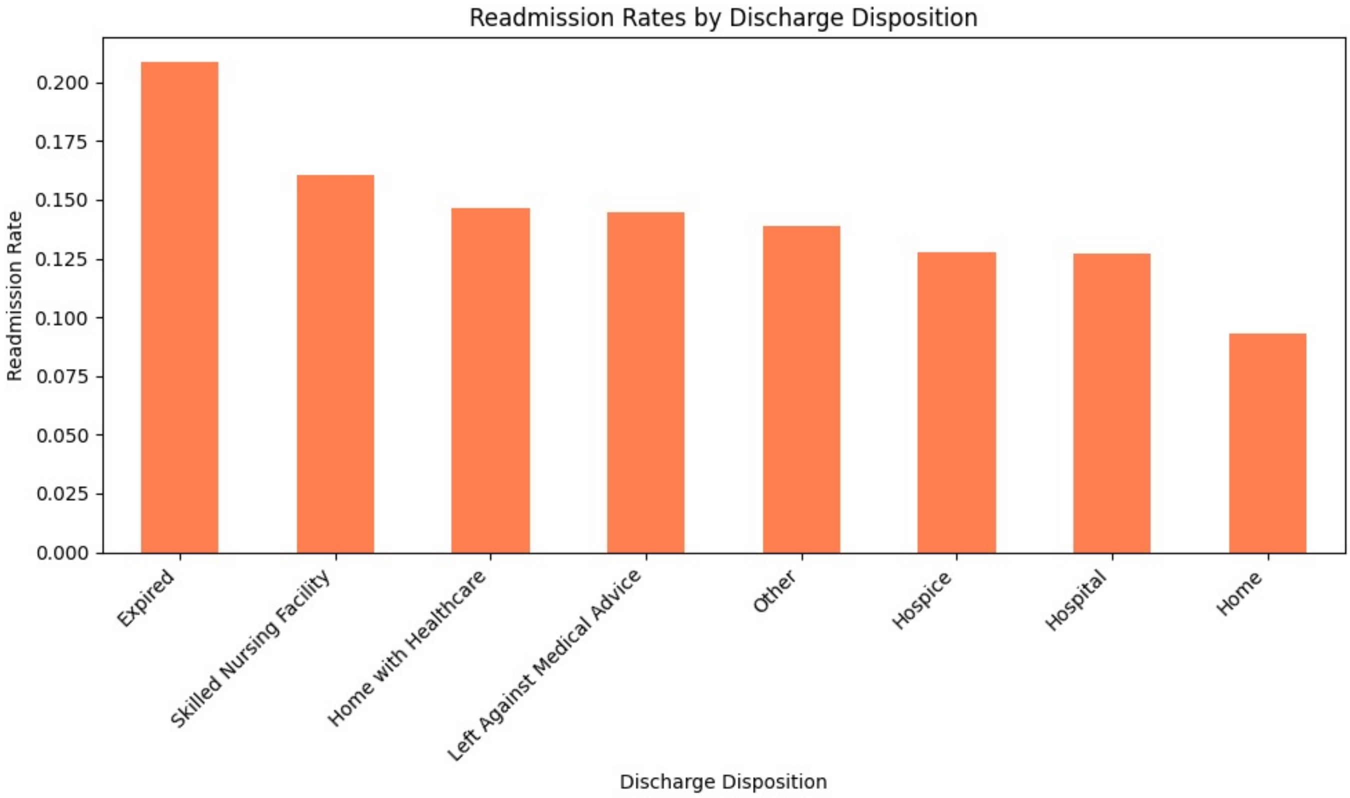
```
C:\Users\nikde\AppData\Roaming\Python\Python311\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure l
ayout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [20]:

```
# Step 1: Preprocessing
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 1, 'NO': 0})  # Encode readmitted as 1 or 0

# Optional: Map discharge_disposition_id to descriptive labels
discharge_mapping = {
    1: 'Home', 2: 'Skilled Nursing Facility', 3: 'Home with Healthcare',
    4: 'Hospice', 5: 'Expired', 6: 'Hospital', 7: 'Left Against Medical Advice',
    8: 'Other'
}
df['discharge_disposition'] = df['discharge_disposition_id'].map(discharge_mapping)
```

```
# Step 2: Calculate readmission rates by discharge disposition
readmission_rates = df.groupby('discharge_disposition')['readmitted'].mean()

# Step 3: Visualization
plt.figure(figsize=(10, 6))
readmission_rates.sort_values(ascending=False).plot(kind='bar', color='coral')
plt.title('Readmission Rates by Discharge Disposition')
plt.xlabel('Discharge Disposition')
plt.ylabel('Readmission Rate')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Step 4: Correlation Analysis
# Assuming discharge_disposition_id is numeric, calculate correlation
correlation = df[['discharge_disposition_id', 'readmitted']].corr().iloc[0, 1]
print(f"Correlation between discharge disposition ID and readmission rates: {correlation:.2f}")

# Print the readmission rates for reference
print("Readmission Rates by Discharge Disposition:")
print(readmission_rates.sort_values(ascending=False))
```

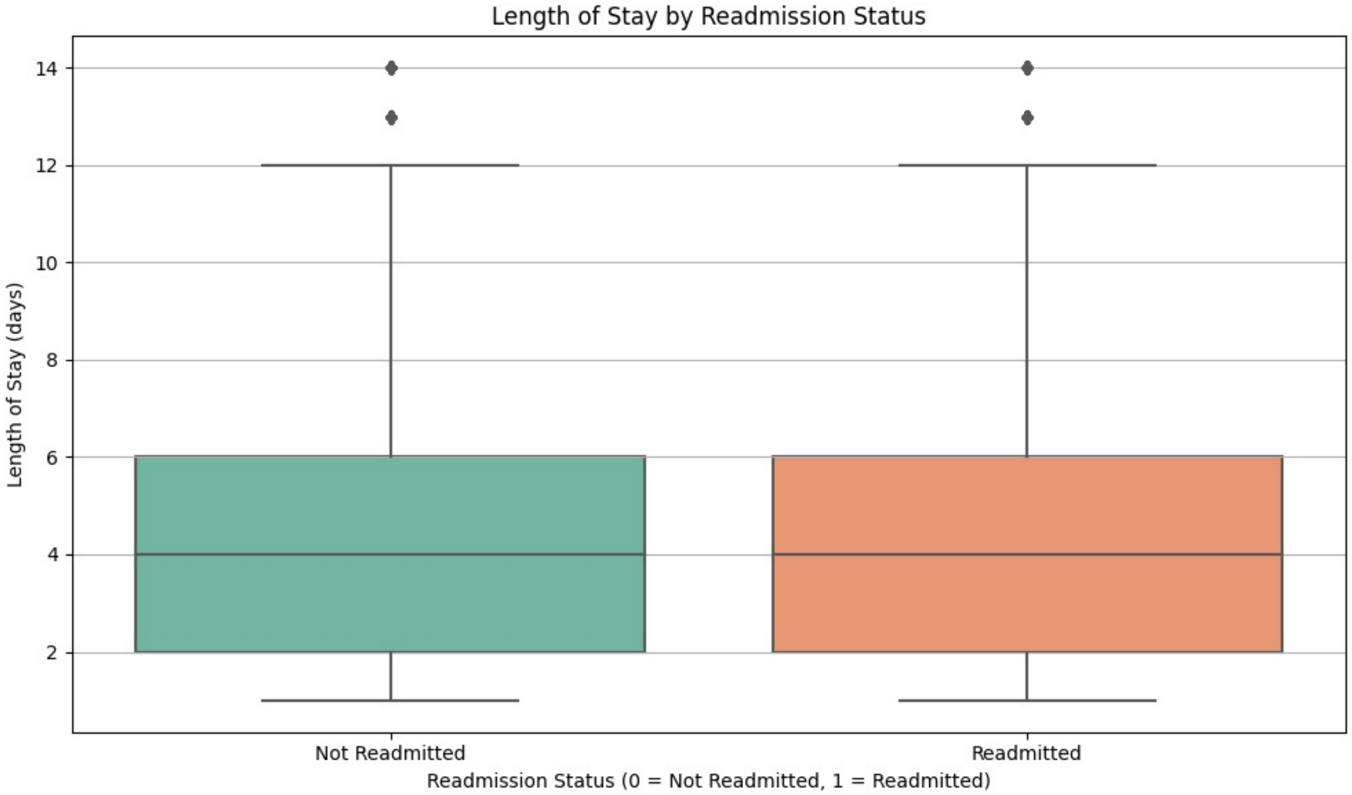


```
Correlation between discharge disposition ID and readmission rates: 0.05
Readmission Rates by Discharge Disposition:
discharge_disposition
Expired                        0.208615
Skilled Nursing Facility       0.160714
Home with Healthcare           0.146625
Left Against Medical Advice    0.144462
Other                          0.138889
Hospice                        0.127607
Hospital                       0.126957
Home                           0.093004
Name: readmitted, dtype: float64
```

In [24]:

```
# Step 1: Preprocessing
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 1, 'NO': 0})  # Encode readmitted as 1 or 0

# Step 2: Group by readmission status and calculate summary statistics
summary_stats = df.groupby('readmitted')['time_in_hospital'].describe()

# Display summary statistics
print("Summary Statistics for Length of Stay by Readmission Status:")
print(summary_stats)

# Step 3: Visualization
plt.figure(figsize=(10, 6))
sns.boxplot(x='readmitted', y='time_in_hospital', data=df, palette='Set2')
plt.title('Length of Stay by Readmission Status')
plt.xlabel('Readmission Status (0 = Not Readmitted, 1 = Readmitted)')
plt.ylabel('Length of Stay (days)')
plt.xticks([0, 1], ['Not Readmitted', 'Readmitted'])
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Optional: Statistical test (t-test)
from scipy.stats import ttest_ind

# Split the data
readmitted = df[df['readmitted'] == 1]['time_in_hospital']
not_readmitted = df[df['readmitted'] == 0]['time_in_hospital']

# Perform t-test
t_stat, p_value = ttest_ind(readmitted, not_readmitted, equal_var=False)
print(f"T-test results: t-statistic = {t_stat:.2f}, p-value = {p_value:.4f}")
```

```
Summary Statistics for Length of Stay by Readmission Status:
              count      mean       std  min  25%  50%  75%   max
readmitted
0           90409.0  4.349224  2.976382  1.0  2.0  4.0  6.0  14.0
1           11357.0  4.768249  3.028165  1.0  2.0  4.0  6.0  14.0
```



```
T-test results: t-statistic = 13.93, p-value = 0.0000
```

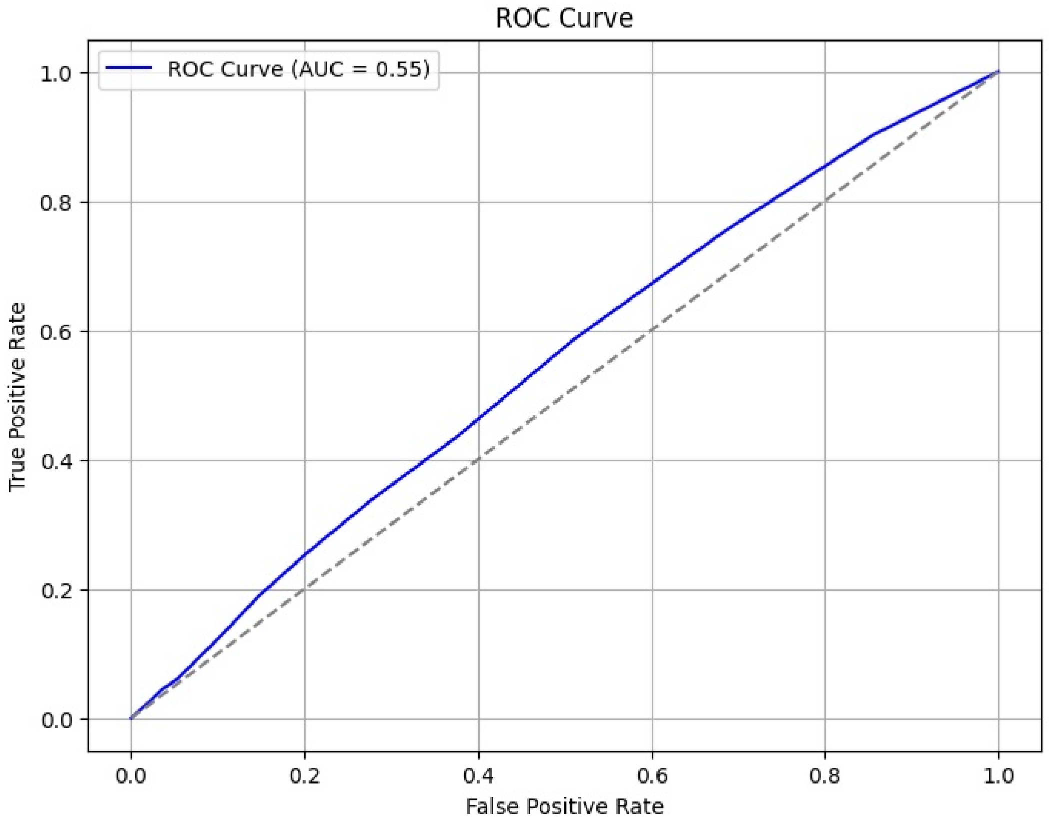
In [26]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score, roc_curve
import matplotlib.pyplot as plt


# Step 1: Preprocessing
df['readmitted'] = df['readmitted'].replace({'<30': 1, '>30': 1, 'NO': 0})  # Encode readmitted as binary

# Feature and target variables
X = df[['time_in_hospital']]  # Use length of stay as the predictor
y = df['readmitted']          # Readmission status as the target

# Step 2: Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Step 3: Build and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 4: Evaluate the model
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
# AUC-ROC score
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f"AUC-ROC Score: {roc_auc:.2f}")

# Plot the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})', color='blue')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid()
plt.show()

# Step 5: Interpretation
coef = model.coef_[0][0]
print(f"Logistic Regression Coefficient for time_in_hospital: {coef:.2f}")
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     27123
           1       0.00      0.00      0.00      3407

    accuracy                           0.89     30530
   macro avg       0.44      0.50      0.47     30530
weighted avg       0.79      0.89      0.84     30530

AUC-ROC Score: 0.55
```

```
C:\Users\nikde\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\_classification.py:1471: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. U
se `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nikde\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\_classification.py:1471: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. U
se `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\nikde\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\_classification.py:1471: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. U
se `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



```
Logistic Regression Coefficient for time_in_hospital: 0.04
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js