```python
# Healthcare Diabetic Data - Exploratory Data Analysis (EDA)
# Building a Fraud Detection System for Healthcare Management

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from scipy import stats
from sklearn.preprocessing import LabelEncoder
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Configuration
warnings.filterwarnings('ignore')
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")

# Set up plotting parameters
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 10

print("Healthcare EDA Analysis - Fraud Detection System")
print("=" * 50)

#
================================================================================
# 1. DATA LOADING AND INITIAL EXPLORATION
#
================================================================================

def load_and_explore_data(file_path='diabetic_data.csv'):
    """
    Load the diabetic data and perform initial exploration
    """
    print("\n1. LOADING AND EXPLORING DATA")
    print("-" * 30)

    # Load the dataset
    df = pd.read_csv(file_path)

    print(f"Dataset shape: {df.shape}")
    print(f"Total encounters: {len(df):,}")

    # Display basic info
    print("\nDataset Info:")
    print(df.info())

    print("\nFirst 5 rows:")
    print(df.head())

    print("\nColumn names:")
    print(df.columns.tolist())
```

```python
    return df

# Load the data
df = load_and_explore_data()

#
================================================================================
# 2. DATA PREPROCESSING AND CLEANING
#
================================================================================

def preprocess_data(df):
    """
    Clean and preprocess the data for analysis
    """
    print("\n2. DATA PREPROCESSING")
    print("-" * 25)

    # Create a copy for processing
    df_clean = df.copy()

    # Handle missing values and '?' entries
    print("Handling missing values and '?' entries...")

    # Replace '?' with NaN
    df_clean = df_clean.replace('?', np.nan)

    # Check missing values
    missing_data = df_clean.isnull().sum()
    missing_percent = (missing_data / len(df_clean)) * 100

    missing_df = pd.DataFrame({
        'Missing Count': missing_data,
        'Missing Percentage': missing_percent
    }).sort_values('Missing Percentage', ascending=False)

    print("\nMissing Data Summary:")
    print(missing_df[missing_df['Missing Count'] > 0])

    # Convert numerical columns
    numerical_cols = ['time_in_hospital', 'num_lab_procedures', 'num_procedures',
                     'num_medications', 'number_outpatient', 'number_emergency',
                     'number_inpatient', 'number_diagnoses']

    for col in numerical_cols:
        if col in df_clean.columns:
            df_clean[col] = pd.to_numeric(df_clean[col], errors='coerce')

    print(f"Cleaned dataset shape: {df_clean.shape}")

    return df_clean

# Preprocess the data
```

```
df_clean = preprocess_data(df)

#
==========================================================================
=========
# 3. DESCRIPTIVE STATISTICAL ANALYSIS FOR NUMERICAL FEATURES
#
==========================================================================
=========

def descriptive_statistics(df):
    """
    Perform comprehensive descriptive statistical analysis
    """
    print("\n3. DESCRIPTIVE STATISTICAL ANALYSIS")
    print("-" * 35)

    # Identify numerical columns
    numerical_cols = df.select_dtypes(include=[np.number]).columns.tolist()

    if numerical_cols:
        print("Numerical Features Summary:")
        desc_stats = df[numerical_cols].describe()
        print(desc_stats)

        # Additional statistics
        print("\nAdditional Statistics:")
        additional_stats = pd.DataFrame({
            'Skewness': df[numerical_cols].skew(),
            'Kurtosis': df[numerical_cols].kurtosis(),
            'IQR': df[numerical_cols].quantile(0.75) - df[numerical_cols].quantile(0.25)
        })
        print(additional_stats)

        # Correlation matrix
        plt.figure(figsize=(12, 8))
        correlation_matrix = df[numerical_cols].corr()
        sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0,
                    square=True, linewidths=0.5)
        plt.title('Correlation Matrix of Numerical Features')
        plt.tight_layout()
        plt.show()

        return desc_stats, additional_stats, correlation_matrix
    else:
        print("No numerical columns found in the dataset")
        return None, None, None

# Perform descriptive statistics
desc_stats, additional_stats, corr_matrix = descriptive_statistics(df_clean)

#
==========================================================================
=========
# 4. VISUALIZATION OF CATEGORICAL FEATURES (RACE AND GENDER)
```

```python
#
================================================================================
========

def visualize_categorical_features(df):
    """
    Visualize the distribution of race and gender
    """
    print("\n4. CATEGORICAL FEATURES VISUALIZATION")
    print("-" * 37)

    fig, axes = plt.subplots(2, 2, figsize=(15, 12))

    # Race distribution
    if 'race' in df.columns:
        race_counts = df['race'].value_counts()
        print(f"Race Distribution:")
        print(race_counts)

        # Bar plot
        race_counts.plot(kind='bar', ax=axes[0,0])
        axes[0,0].set_title('Distribution of Race')
        axes[0,0].set_xlabel('Race')
        axes[0,0].set_ylabel('Count')
        axes[0,0].tick_params(axis='x', rotation=45)

        # Pie chart
        axes[0,1].pie(race_counts.values, labels=race_counts.index, autopct='%1.1f%%')
        axes[0,1].set_title('Race Distribution (Pie Chart)')

    # Gender distribution
    if 'gender' in df.columns:
        gender_counts = df['gender'].value_counts()
        print(f"\nGender Distribution:")
        print(gender_counts)

        # Bar plot
        gender_counts.plot(kind='bar', ax=axes[1,0])
        axes[1,0].set_title('Distribution of Gender')
        axes[1,0].set_xlabel('Gender')
        axes[1,0].set_ylabel('Count')

        # Pie chart
        axes[1,1].pie(gender_counts.values, labels=gender_counts.index, autopct='%1.1f%%')
        axes[1,1].set_title('Gender Distribution (Pie Chart)')

    plt.tight_layout()
    plt.show()

    return race_counts if 'race' in df.columns else None, gender_counts if 'gender' in df.columns else None

# Visualize categorical features
race_dist, gender_dist = visualize_categorical_features(df_clean)
```

```python
#
============================================================================
========
# 5. RELATIONSHIP BETWEEN READMISSION STATUS AND AGE
#
============================================================================
========

def analyze_readmission_age_relationship(df):
    """
    Explore the relationship between readmission status and age
    """
    print("\n5. READMISSION STATUS vs AGE ANALYSIS")
    print("-" * 36)

    if 'readmitted' in df.columns and 'age' in df.columns:
        # Cross-tabulation
        readmit_age_crosstab = pd.crosstab(df['age'], df['readmitted'], margins=True)
        print("Readmission by Age Group:")
        print(readmit_age_crosstab)

        # Percentage breakdown
        readmit_age_pct = pd.crosstab(df['age'], df['readmitted'], normalize='index') * 100
        print("\nReadmission Percentage by Age Group:")
        print(readmit_age_pct.round(2))

        # Visualization
        fig, axes = plt.subplots(2, 2, figsize=(16, 12))

        # Stacked bar chart
        readmit_age_crosstab.iloc[:-1, :-1].plot(kind='bar', stacked=True, ax=axes[0,0])
        axes[0,0].set_title('Readmission Status by Age Group (Stacked)')
        axes[0,0].set_xlabel('Age Group')
        axes[0,0].set_ylabel('Count')
        axes[0,0].legend(title='Readmitted')
        axes[0,0].tick_params(axis='x', rotation=45)

        # Percentage stacked bar chart
        readmit_age_pct.plot(kind='bar', stacked=True, ax=axes[0,1])
        axes[0,1].set_title('Readmission Percentage by Age Group')
        axes[0,1].set_xlabel('Age Group')
        axes[0,1].set_ylabel('Percentage')
        axes[0,1].legend(title='Readmitted')
        axes[0,1].tick_params(axis='x', rotation=45)

        # Heatmap
        sns.heatmap(readmit_age_pct, annot=True, fmt='.1f', cmap='YlOrRd', ax=axes[1,0])
        axes[1,0].set_title('Readmission Rate Heatmap by Age')

        # Calculate 30-day readmission rates by age
        if '<30' in df['readmitted'].values:
            df['readmitted_30'] = df['readmitted'] == '<30'
            readmit_30_by_age = df.groupby('age')['readmitted_30'].agg(['count', 'sum', 'mean'])
            readmit_30_by_age.columns = ['Total', '30-day_Readmits', '30-day_Rate']
            readmit_30_by_age['30-day_Rate'] *= 100
```

```
            readmit_30_by_age['30-day_Rate'].plot(kind='bar', ax=axes[1,1], color='red', alpha=0.7)
            axes[1,1].set_title('30-Day Readmission Rate by Age Group')
            axes[1,1].set_xlabel('Age Group')
            axes[1,1].set_ylabel('30-Day Readmission Rate (%)')
            axes[1,1].tick_params(axis='x', rotation=45)

            print("\n30-Day Readmission Analysis:")
            print(readmit_30_by_age)

        plt.tight_layout()
        plt.show()

        return readmit_age_crosstab, readmit_age_pct
    else:
        print("Required columns (readmitted, age) not found")
        return None, None

# Analyze readmission-age relationship
readmit_age_cross, readmit_age_pct = analyze_readmission_age_relationship(df_clean)

#
======================================================================
========
# 6. MEDICATION ANALYSIS
#
======================================================================
========

def analyze_medications(df):
    """
    Analyze medication changes and total medications
    """
    print("\n6. MEDICATION ANALYSIS")
    print("-" * 21)

    # Total medications analysis
    if 'num_medications' in df.columns:
        print("Total Medications Statistics:")
        print(df['num_medications'].describe())

        fig, axes = plt.subplots(2, 2, figsize=(15, 12))

        # Distribution of total medications
        df['num_medications'].hist(bins=30, ax=axes[0,0], alpha=0.7, edgecolor='black')
        axes[0,0].set_title('Distribution of Total Medications')
        axes[0,0].set_xlabel('Number of Medications')
        axes[0,0].set_ylabel('Frequency')

        # Box plot
        df.boxplot(column='num_medications', ax=axes[0,1])
        axes[0,1].set_title('Box Plot of Total Medications')
        axes[0,1].set_ylabel('Number of Medications')

    # Medication change analysis
    if 'change' in df.columns:
        change_counts = df['change'].value_counts()
```

```
        print(f"\nMedication Change Distribution:")
        print(change_counts)

        # Medication change visualization
        change_counts.plot(kind='bar', ax=axes[1,0])
        axes[1,0].set_title('Distribution of Medication Changes')
        axes[1,0].set_xlabel('Medication Change')
        axes[1,0].set_ylabel('Count')

        # Pie chart
        axes[1,1].pie(change_counts.values, labels=change_counts.index, autopct='%1.1f%%')
        axes[1,1].set_title('Medication Change Distribution')

    plt.tight_layout()
    plt.show()

    # Relationship between medication change and readmission
    if 'change' in df.columns and 'readmitted' in df.columns:
        med_change_readmit = pd.crosstab(df['change'], df['readmitted'], normalize='index') * 100
        print("\nMedication Change vs Readmission (%):")
        print(med_change_readmit.round(2))

        plt.figure(figsize=(10, 6))
        sns.heatmap(med_change_readmit, annot=True, fmt='.1f', cmap='RdYlBu')
        plt.title('Medication Change vs Readmission Rate')
        plt.show()

# Analyze medications
analyze_medications(df_clean)

#
==================================================================
========
# 7. DIAGNOSIS CATEGORIES ANALYSIS
#
==================================================================
========

def analyze_diagnoses(df):
    """
    Examine the distribution of diagnosis categories
    """
    print("\n7. DIAGNOSIS CATEGORIES ANALYSIS")
    print("-" * 31)

    diag_cols = ['diag_1', 'diag_2', 'diag_3']

    for diag_col in diag_cols:
        if diag_col in df.columns:
            print(f"\n{diag_col.upper()} Analysis:")

            # Top 10 diagnoses
            top_diag = df[diag_col].value_counts().head(10)
            print(f"Top 10 {diag_col}:")
            print(top_diag)
```

```python
        # Visualization
        plt.figure(figsize=(12, 6))
        top_diag.plot(kind='bar')
        plt.title(f'Top 10 {diag_col} Codes')
        plt.xlabel('Diagnosis Code')
        plt.ylabel('Frequency')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

    # Diabetes-related diagnosis analysis
    if 'diag_1' in df.columns:
        print("\nDiabetes-related Primary Diagnoses:")
        diabetes_diag = df[df['diag_1'].str.contains('250', na=False)]['diag_1'].value_counts()
        print(diabetes_diag.head(10))

# Analyze diagnoses
analyze_diagnoses(df_clean)

#
========================================================================
# 8. ADMISSION ANALYSIS
#
========================================================================

def analyze_admissions(df):
    """
    Explore distribution across admission types, sources, and discharge dispositions
    """
    print("\n8. ADMISSION ANALYSIS")
    print("-" * 19)

    admission_cols = ['admission_type_id', 'admission_source_id', 'discharge_disposition_id']

    fig, axes = plt.subplots(2, 2, figsize=(16, 12))
    axes = axes.flatten()

    for i, col in enumerate(admission_cols):
        if col in df.columns and i < 3:
            col_counts = df[col].value_counts().head(10)
            print(f"\n{col.replace('_', ' ').title()} Distribution:")
            print(col_counts)

            col_counts.plot(kind='bar', ax=axes[i])
            axes[i].set_title(f'Distribution of {col.replace("_", " ").title()}')
            axes[i].set_xlabel(col.replace('_', ' ').title())
            axes[i].set_ylabel('Count')
            axes[i].tick_params(axis='x', rotation=45)

    # Length of stay analysis
    if 'time_in_hospital' in df.columns:
        stay_counts = df['time_in_hospital'].value_counts().sort_index()
        print(f"\nLength of Stay Distribution:")
        print(stay_counts.head(10))
```

```python
        stay_counts.head(14).plot(kind='bar', ax=axes[3])
        axes[3].set_title('Distribution of Length of Stay')
        axes[3].set_xlabel('Days in Hospital')
        axes[3].set_ylabel('Count')

    plt.tight_layout()
    plt.show()

# Analyze admissions
analyze_admissions(df_clean)

#
=============================================================================
========
# 9. OUTLIER DETECTION AND VISUALIZATION
#
=============================================================================
========

def detect_outliers(df):
    """
    Identify and visualize outliers in numerical features
    """
    print("\n9. OUTLIER DETECTION")
    print("-" * 18)

    numerical_cols = df.select_dtypes(include=[np.number]).columns.tolist()

    if not numerical_cols:
        print("No numerical columns found for outlier detection")
        return

    # Z-score method
    print("Outlier Detection using Z-score (threshold = 3):")
    z_scores = np.abs(stats.zscore(df[numerical_cols].fillna(df[numerical_cols].median())))
    outlier_counts_z = (z_scores > 3).sum()
    print(outlier_counts_z)

    # IQR method
    print("\nOutlier Detection using IQR method:")
    outlier_counts_iqr = {}

    for col in numerical_cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers = ((df[col] < lower_bound) | (df[col] > upper_bound)).sum()
        outlier_counts_iqr[col] = outliers

    outlier_df = pd.DataFrame({
        'IQR_Outliers': outlier_counts_iqr,
        'Z_Score_Outliers': outlier_counts_z
    })
```

```python
    print(outlier_df)

    # Visualization
    n_cols = len(numerical_cols)
    n_rows = (n_cols + 2) // 3

    fig, axes = plt.subplots(n_rows, 3, figsize=(18, 4*n_rows))
    axes = axes.flatten() if n_rows > 1 else [axes] if n_rows == 1 else axes

    for i, col in enumerate(numerical_cols[:len(axes)]):
        if i < len(axes):
            df.boxplot(column=col, ax=axes[i])
            axes[i].set_title(f'Box Plot: {col}')
            axes[i].set_ylabel(col)

    # Hide empty subplots
    for i in range(len(numerical_cols), len(axes)):
        axes[i].set_visible(False)

    plt.tight_layout()
    plt.show()

    return outlier_df

# Detect outliers
outlier_summary = detect_outliers(df_clean)

#
==========================================================================
========
# 10. FRAUD DETECTION INSIGHTS
#
==========================================================================
========

def fraud_detection_analysis(df):
    """
    Analyze patterns that could indicate healthcare fraud
    """
    print("\n10. FRAUD DETECTION ANALYSIS")
    print("-" * 27)

    # Suspicious pattern 1: Unusually high number of procedures
    if 'num_procedures' in df.columns:
        high_procedures = df[df['num_procedures'] > df['num_procedures'].quantile(0.95)]
        print(f"Patients with unusually high procedures (>95th percentile): {len(high_procedures)}")
        print(f"Average procedures in this group: {high_procedures['num_procedures'].mean():.2f}")

    # Suspicious pattern 2: Multiple readmissions
    if 'patient_nbr' in df.columns and 'readmitted' in df.columns:
        patient_encounters = df['patient_nbr'].value_counts()
        multiple_encounters = patient_encounters[patient_encounters > 1]
        print(f"\nPatients with multiple encounters: {len(multiple_encounters)}")
        print(f"Average encounters per patient: {patient_encounters.mean():.2f}")

    # Suspicious pattern 3: Unusual medication combinations
```

```
    if 'num_medications' in df.columns:
        high_medications = df[df['num_medications'] > df['num_medications'].quantile(0.99)]
        print(f"\nPatients with extremely high medications (>99th percentile): {len(high_medications)}")

    # Suspicious pattern 4: Short stays with high readmission
    if 'time_in_hospital' in df.columns and 'readmitted' in df.columns:
        short_stay_readmit = df[(df['time_in_hospital'] == 1) & (df['readmitted'] == '<30')]
        print(f"\nShort stays (1 day) with 30-day readmission: {len(short_stay_readmit)}")

    # Risk score calculation
    risk_factors = []

    if 'num_procedures' in df.columns:
        df['high_procedures'] = (df['num_procedures'] > df['num_procedures'].quantile(0.95)).astype(int)
        risk_factors.append('high_procedures')

    if 'num_medications' in df.columns:
        df['high_medications'] = (df['num_medications'] > df['num_medications'].quantile(0.95)).astype(int)
        risk_factors.append('high_medications')

    if 'time_in_hospital' in df.columns:
        df['short_stay'] = (df['time_in_hospital'] <= 1).astype(int)
        risk_factors.append('short_stay')

    if risk_factors:
        df['fraud_risk_score'] = df[risk_factors].sum(axis=1)

        print(f"\nFraud Risk Score Distribution:")
        risk_distribution = df['fraud_risk_score'].value_counts().sort_index()
        print(risk_distribution)

        plt.figure(figsize=(10, 6))
        risk_distribution.plot(kind='bar')
        plt.title('Distribution of Fraud Risk Scores')
        plt.xlabel('Risk Score')
        plt.ylabel('Count')
        plt.show()

        # High-risk cases
        high_risk = df[df['fraud_risk_score'] >= 2]
        print(f"\nHigh-risk cases (score >= 2): {len(high_risk)} ({len(high_risk)/len(df)*100:.2f}%)")

# Fraud detection analysis
fraud_detection_analysis(df_clean)

# 
==============================================================================
# 11. ANALYSIS REPORT GENERATION
# 
==============================================================================

def generate_analysis_report(df, desc_stats, corr_matrix, outlier_summary):
    """
    Generate comprehensive EDA report for fraud detection system
```

```python
"""
print("\n" + "="*70)
print("HEALTHCARE EDA ANALYSIS REPORT")
print("FRAUD DETECTION SYSTEM CONTEXT")
print("="*70)

print("\n1. EXECUTIVE SUMMARY")
print("-" * 20)
print(f"• Dataset contains {len(df):,} healthcare encounters")
print(f"• Analysis covers {df.shape[1]} features including demographics, medical procedures, and outcomes")
print(f"• Key focus areas: readmission patterns, medication management, and fraud detection indicators")

print("\n2. DATA QUALITY ASSESSMENT")
print("-" * 28)
missing_data = df.isnull().sum().sum()
print(f"• Total missing values: {missing_data:,}")
print(f"• Data completeness: {((df.size - missing_data) / df.size * 100):.1f}%")

print("\n3. KEY FINDINGS")
print("-" * 15)

# Demographics
if 'age' in df.columns:
    elderly_patients = len(df[df['age'].isin(['[70-80)', '[80-90)', '[90-100)'])])
    print(f"• Elderly patients (70+): {elderly_patients:,} ({elderly_patients/len(df)*100:.1f}%)")

# Readmissions
if 'readmitted' in df.columns:
    readmit_30 = len(df[df['readmitted'] == '<30'])
    print(f"• 30-day readmissions: {readmit_30:,} ({readmit_30/len(df)*100:.1f}%)")

# High-risk indicators
if 'num_medications' in df.columns:
    high_med_patients = len(df[df['num_medications'] > df['num_medications'].quantile(0.95)])
    print(f"• High medication patients (>95th percentile): {high_med_patients:,}")

print("\n4. FRAUD DETECTION IMPLICATIONS")
print("-" * 32)
print("• Patterns identified for potential fraud detection:")
print("  - Unusually high procedure counts")
print("  - Excessive medication prescriptions")
print("  - Frequent readmissions with short stays")
print("  - Outlier patterns in treatment intensity")

print("\n5. RECOMMENDATIONS")
print("-" * 18)
print("• Implement real-time monitoring for high-risk score patients")
print("• Develop predictive models for 30-day readmission prevention")
print("• Create alerts for unusual medication/procedure combinations")
print("• Establish care transition protocols for elderly patients")
print("• Monitor patients with multiple encounters for potential fraud")

print("\n6. TECHNICAL NOTES")
print("-" * 16)
print(f"• Analysis performed using Python pandas, matplotlib, seaborn")
print(f"• Statistical methods: descriptive statistics, correlation analysis, outlier detection")
```

```python
    print(f"• Fraud risk scoring based on multiple behavioral indicators")

    print("\n" + "="*70)
    print("END OF ANALYSIS REPORT")
    print("="*70)

# Generate final report
generate_analysis_report(df_clean, desc_stats, corr_matrix, outlier_summary)

print("\n🎉 Healthcare EDA Analysis Complete!")
print("All visualizations and analyses have been generated.")
print("The notebook is ready for your fraud detection system implementation.")
```

```
Healthcare EDA Analysis - Fraud Detection System
==================================================

1. LOADING AND EXPLORING DATA
------------------------------
Dataset shape: (101766, 50)
Total encounters: 101,766

Dataset 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
None

First 5 rows:
   encounter_id  patient_nbr             race  gender      age weight  \
0       2278392      8222157        Caucasian  Female   [0-10)      ?
1        149190     55629189        Caucasian  Female  [10-20)      ?
2         64410     86047875  AfricanAmerican  Female  [20-30)      ?
3        500364     82442376        Caucasian    Male  [30-40)      ?
4         16680     42519267        Caucasian    Male  [40-50)      ?

   admission_type_id  discharge_disposition_id  admission_source_id  \
0                  6                        25                    1
1                  1                         1                    7
2                  1                         1                    7
3                  1                         1                    7
4                  1                         1                    7

   time_in_hospital  ... citoglipton insulin  glyburide-metformin  \
0                 1  ...          No      No                   No
1                 3  ...          No      Up                   No
2                 2  ...          No      No                   No
3                 2  ...          No      Up                   No
4                 1  ...          No  Steady                   No

   glipizide-metformin  glimepiride-pioglitazone  metformin-rosiglitazone  \
0                   No                        No                       No
1                   No                        No                       No
2                   No                        No                       No
3                   No                        No                       No
4                   No                        No                       No

   metformin-pioglitazone  change diabetesMed readmitted
0                      No      No          No         NO
1                      No      Ch         Yes        >30
```

```
2                No    No      Yes      NO
3                No    Ch      Yes      NO
4                No    Ch      Yes      NO

[5 rows x 50 columns]

Column names:
['encounter_id', 'patient_nbr', 'race', 'gender', 'age', 'weight', 'admission_type_id', 'discharge_disposition_id', 'admission_source_id', 'time_in_hospital', 'payer_code', 'medical_specialty', 'num_lab_procedures', 'num_procedures', 'num_medications', 'number_outpatient', 'number_emergency', 'number_inpatient', 'diag_1', 'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone', 'tolazamide', 'examide', 'citoglipton', 'insulin', 'glyburide-metformin', 'glipizide-metformin', 'glimepiride-pioglitazone', 'metformin-rosiglitazone', 'metformin-pioglitazone', 'change', 'diabetesMed', 'readmitted']

2. DATA PREPROCESSING
-------------------------
Handling missing values and '?' entries...

Missing Data Summary:
                   Missing Count  Missing Percentage
weight                     98569           96.858479
max_glu_serum              96420           94.746772
A1Cresult                  84748           83.277322
medical_specialty          49949           49.082208
payer_code                 40256           39.557416
race                        2273            2.233555
diag_3                      1423            1.398306
diag_2                       358            0.351787
diag_1                        21            0.020636
Cleaned dataset shape: (101766, 50)

3. DESCRIPTIVE STATISTICAL ANALYSIS
-----------------------------------
Numerical Features Summary:
       encounter_id   patient_nbr  admission_type_id  \
count  1.017660e+05  1.017660e+05      101766.000000
mean   1.652016e+08  5.433040e+07           2.024006
std    1.026403e+08  3.869636e+07           1.445403
min    1.252200e+04  1.350000e+02           1.000000
25%    8.496119e+07  2.341322e+07           1.000000
50%    1.523890e+08  4.550514e+07           1.000000
75%    2.302709e+08  8.754595e+07           3.000000
max    4.438672e+08  1.895026e+08           8.000000

       discharge_disposition_id  admission_source_id  time_in_hospital  \
count             101766.000000        101766.000000     101766.000000
mean                   3.715642             5.754437          4.395987
std                    5.280166             4.064081          2.985108
min                    1.000000             1.000000          1.000000
25%                    1.000000             1.000000          2.000000
50%                    1.000000             7.000000          4.000000
75%                    4.000000             7.000000          6.000000
max                   28.000000            25.000000         14.000000

       num_lab_procedures  num_procedures  num_medications  number_outpatient  \
```

```
count     101766.000000   101766.000000   101766.000000    101766.000000
mean          43.095641        1.339730       16.021844         0.369357
std           19.674362        1.705807        8.127566         1.267265
min            1.000000        0.000000        1.000000         0.000000
25%           31.000000        0.000000       10.000000         0.000000
50%           44.000000        1.000000       15.000000         0.000000
75%           57.000000        2.000000       20.000000         0.000000
max          132.000000        6.000000       81.000000        42.000000

       number_emergency  number_inpatient  number_diagnoses
count     101766.000000     101766.000000     101766.000000
mean           0.197836          0.635566          7.422607
std            0.930472          1.262863          1.933600
min            0.000000          0.000000          1.000000
25%            0.000000          0.000000          6.000000
50%            0.000000          0.000000          8.000000
75%            0.000000          1.000000          9.000000
max           76.000000         21.000000         16.000000

Additional Statistics:
                          Skewness     Kurtosis           IQR
encounter_id              0.699142    -0.102071  1.453097e+08
patient_nbr               0.471281    -0.347372  6.413273e+07
admission_type_id         1.591984     1.942476  2.000000e+00
discharge_disposition_id  2.563067     6.003347  3.000000e+00
admission_source_id       1.029935     1.744989  6.000000e+00
time_in_hospital          1.133999     0.850251  4.000000e+00
num_lab_procedures       -0.236544    -0.245074  2.600000e+01
num_procedures            1.316415     0.857110  2.000000e+00
num_medications           1.326672     3.468155  1.000000e+01
number_outpatient         8.832959   147.907736  0.000000e+00
number_emergency         22.855582  1191.686726  0.000000e+00
number_inpatient          3.614139    20.719397  1.000000e+00
number_diagnoses         -0.876746    -0.079056  3.000000e+00
```

4. CATEGORICAL FEATURES VISUALIZATION

```
-------------------------------------
Race Distribution:
race
Caucasian          76099
AfricanAmerican    19210
Hispanic            2037
Other               1506
Asian                641
Name: count, dtype: int64

Gender Distribution:
gender
Female             54708
Male               47055
Unknown/Invalid        3
Name: count, dtype: int64
```

5. READMISSION STATUS vs AGE ANALYSIS

```
------------------------------------
```

Readmission by Age Group:

readmitted	<30	>30	NO	All
age				
[0-10)	3	26	132	161
[10-20)	40	224	427	691
[20-30)	236	510	911	1657
[30-40)	424	1187	2164	3775
[40-50)	1027	3278	5380	9685
[50-60)	1668	5917	9671	17256
[60-70)	2502	7897	12084	22483
[70-80)	3069	9475	13524	26068
[80-90)	2078	6223	8896	17197
[90-100)	310	808	1675	2793
All	11357	35545	54864	101766

Readmission Percentage by Age Group:

readmitted	<30	>30	NO
age			
[0-10)	1.86	16.15	81.99
[10-20)	5.79	32.42	61.79
[20-30)	14.24	30.78	54.98
[30-40)	11.23	31.44	57.32
[40-50)	10.60	33.85	55.55
[50-60)	9.67	34.29	56.04
[60-70)	11.13	35.12	53.75
[70-80)	11.77	36.35	51.88
[80-90)	12.08	36.19	51.73
[90-100)	11.10	28.93	59.97

30-Day Readmission Analysis:

	Total	30-day_Readmits	30-day_Rate
age			
[0-10)	161	3	1.863354
[10-20)	691	40	5.788712
[20-30)	1657	236	14.242607
[30-40)	3775	424	11.231788
[40-50)	9685	1027	10.604027
[50-60)	17256	1668	9.666203
[60-70)	22483	2502	11.128408
[70-80)	26068	3069	11.773055
[80-90)	17197	2078	12.083503
[90-100)	2793	310	11.099177

6. MEDICATION ANALYSIS

Total Medications Statistics:

count	101766.000000
mean	16.021844
std	8.127566
min	1.000000
25%	10.000000
50%	15.000000
75%	20.000000
max	81.000000

Name: num_medications, dtype: float64

```
Medication Change Distribution:
change
No    54755
Ch    47011
Name: count, dtype: int64

Medication Change vs Readmission (%):
readmitted    <30    >30     NO
change
Ch          11.82  36.74  51.44
No          10.59  33.37  56.04
```

7. DIAGNOSIS CATEGORIES ANALYSIS

```
DIAG_1 Analysis:
Top 10 diag_1:
diag_1
428    6862
414    6581
786    4016
410    3614
486    3508
427    2766
491    2275
715    2151
682    2042
434    2028
Name: count, dtype: int64

DIAG_2 Analysis:
Top 10 diag_2:
diag_2
276    6752
428    6662
250    6071
427    5036
401    3736
496    3305
599    3288
403    2823
414    2650
411    2566
Name: count, dtype: int64

DIAG_3 Analysis:
Top 10 diag_3:
diag_3
250    11555
401     8289
276     5175
428     4577
427     3955
414     3664
496     2605
```

```
403    2357
585    1992
272    1969
Name: count, dtype: int64

Diabetes-related Primary Diagnoses:
diag_1
250.8     1680
250.6     1183
250.7      871
250.13     851
250.02     675
250.11     625
250.12     417
250.82     412
250.1      313
250.4      267
Name: count, dtype: int64

8. ADMISSION ANALYSIS
-------------------

Admission Type Id Distribution:
admission_type_id
1    53990
3    18869
2    18480
6     5291
5     4785
8      320
7       21
4       10
Name: count, dtype: int64

Admission Source Id Distribution:
admission_source_id
7     57494
1     29565
17     6781
4      3187
6      2264
2      1104
5       855
3       187
20      161
9       125
Name: count, dtype: int64

Discharge Disposition Id Distribution:
discharge_disposition_id
1     60234
3     13954
6     12902
18     3691
2      2128
22     1993
```

11   1642
5   1184
25   989
4   815
Name: count, dtype: int64

Length of Stay Distribution:
time_in_hospital
1   14208
2   17224
3   17756
4   13924
5   9966
6   7539
7   5859
8   4391
9   3002
10   2342
Name: count, dtype: int64

9. OUTLIER DETECTION

Outlier Detection using Z-score (threshold = 3):

encounter_id	0
patient_nbr	847
admission_type_id	341
discharge_disposition_id	3588
admission_source_id	175
time_in_hospital	1042
num_lab_procedures	43
num_procedures	0
num_medications	1361
number_outpatient	1457
number_emergency	1664
number_inpatient	2016
number_diagnoses	281

dtype: int64

Outlier Detection using IQR method:

	IQR_Outliers	Z_Score_Outliers
encounter_id	0	0
patient_nbr	247	847
admission_type_id	341	341
discharge_disposition_id	9818	3588
admission_source_id	6956	175
time_in_hospital	2252	1042
num_lab_procedures	143	43
num_procedures	4954	0
num_medications	2557	1361
number_outpatient	16739	1457
number_emergency	11383	1664
number_inpatient	7049	2016
number_diagnoses	281	281

10. FRAUD DETECTION ANALYSIS

Patients with unusually high procedures (>95th percentile): 4954
Average procedures in this group: 6.00

Patients with multiple encounters: 16773
Average encounters per patient: 1.42

Patients with extremely high medications (>99th percentile): 960

Short stays (1 day) with 30-day readmission: 1162

Fraud Risk Score Distribution:
fraud_risk_score
0   80266
1   19316
2   2181
3   3
Name: count, dtype: int64

High-risk cases (score >= 2): 2184 (2.15%)

==
HEALTHCARE EDA ANALYSIS REPORT
FRAUD DETECTION SYSTEM CONTEXT
==

1. EXECUTIVE SUMMARY

• Dataset contains 101,766 healthcare encounters
• Analysis covers 55 features including demographics, medical procedures, and outcomes
• Key focus areas: readmission patterns, medication management, and fraud detection indicators

2. DATA QUALITY ASSESSMENT

• Total missing values: 374,017
• Data completeness: 93.3%

3. KEY FINDINGS

• Elderly patients (70+): 46,058 (45.3%)
• 30-day readmissions: 11,357 (11.2%)
• High medication patients (>95th percentile): 4,525

4. FRAUD DETECTION IMPLICATIONS

• Patterns identified for potential fraud detection:
  - Unusually high procedure counts
  - Excessive medication prescriptions
  - Frequent readmissions with short stays
  - Outlier patterns in treatment intensity

5. RECOMMENDATIONS

• Implement real-time monitoring for high-risk score patients
• Develop predictive models for 30-day readmission prevention
• Create alerts for unusual medication/procedure combinations

• Establish care transition protocols for elderly patients
• Monitor patients with multiple encounters for potential fraud

6. TECHNICAL NOTES

• Analysis performed using Python pandas, matplotlib, seaborn
• Statistical methods: descriptive statistics, correlation analysis, outlier detection
• Fraud risk scoring based on multiple behavioral indicators

==
END OF ANALYSIS REPORT
==

🎉 Healthcare EDA Analysis Complete!
All visualizations and analyses have been generated.
The notebook is ready for your fraud detection system implementation.