Title: Final Report Analysing Diabetic Patient Data for Predictive Healthcare
FALL 2023: Scientific & Clinical Data Management: 36921
Author: Asmatahasin Mohammed
Instructor: Yan Zhuang

Introduction:

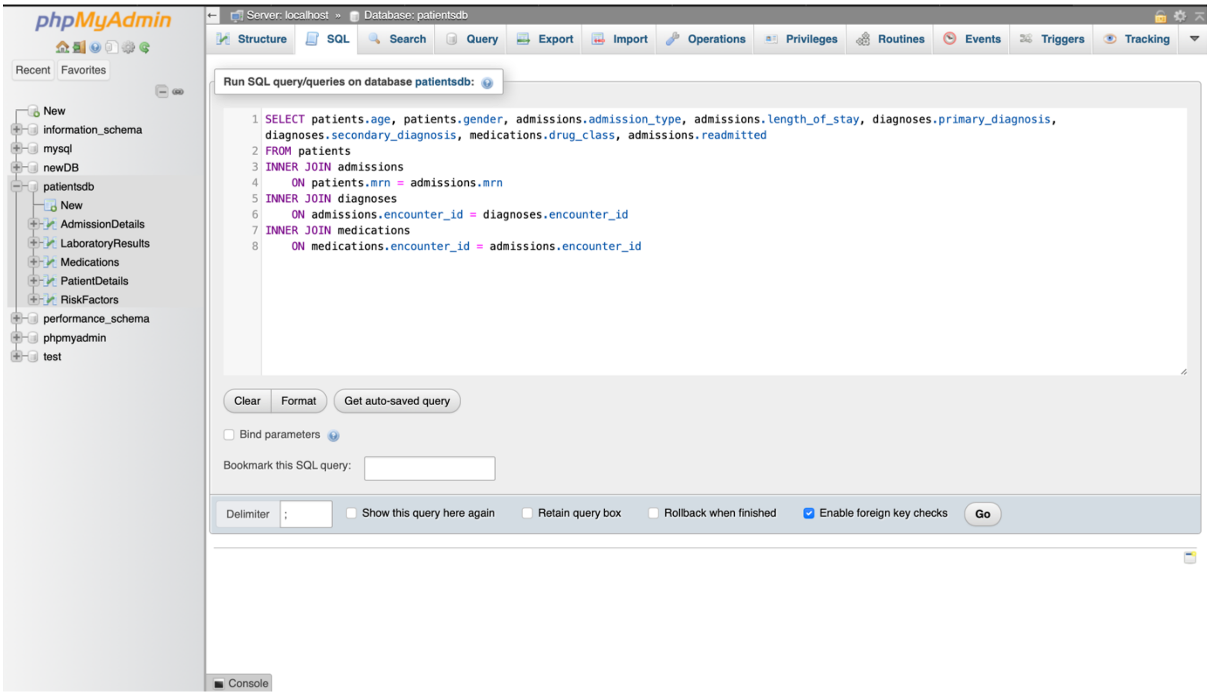
As the lead data analyst for my 3-person team, I spearheaded the extraction, manipulation and modelling of two diabetes datasets from Kaggle containing patient vitals, lab tests, admissions data, and other attributes. Core responsibilities as a lead data analyst and visualisation expert included: writing efficient queries for analysis-ready data; conducting in-depth mining uncovering trends and risk factors predictive of hospital readmissions. Advanced analytics further targeted high-risk patient groups for personalized interventions. I managed the data, implemented ETL pipelines, communicated technical insights through Jupyter notebooks, while ensuring overall data integrity.

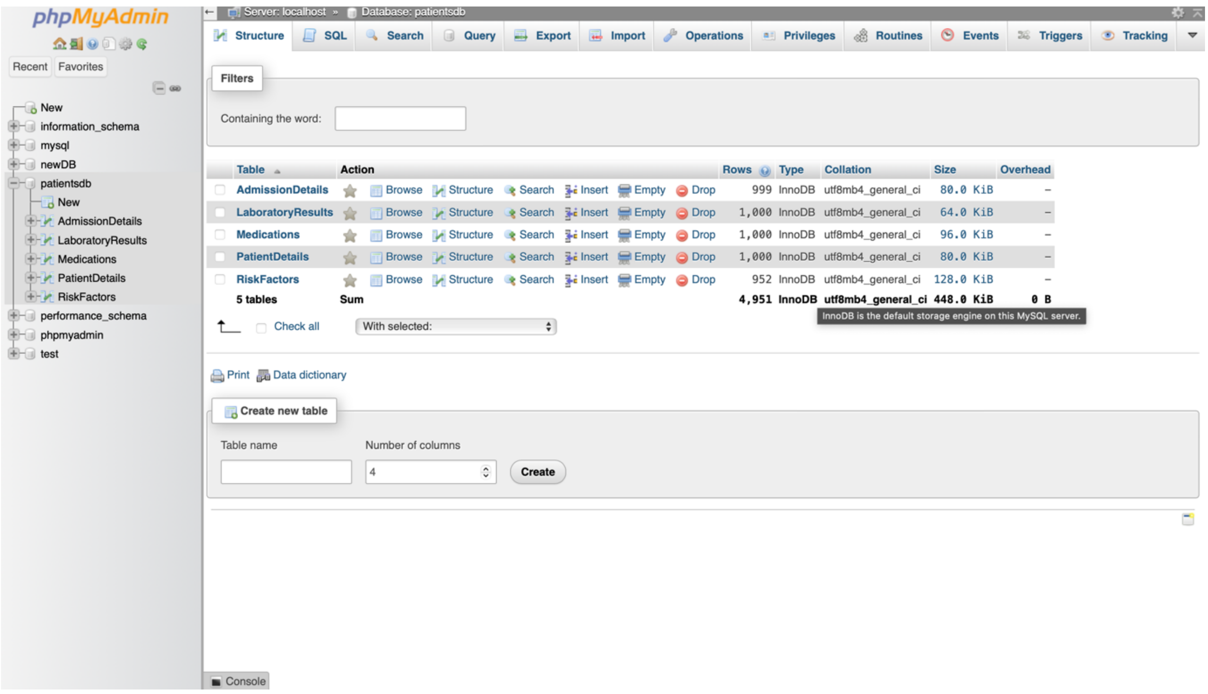
Required Question:

1. Attributes Used for Analytics:

Key attributes used to inform our readmission predictive models included patient age, gender, admission details, BMI, blood pressure, medication types, laboratory results, primary and secondary diagnoses, comorbidities, HbA1c levels, complications during hospitalization, any medications administered or changed, previous outpatient clinic or ER visits 90 days before admission. These encompassed relevant clinical, administrative and demographic information.

For example, I leveraged the following SQL join querying the integrated datasets to extract features across multiple tables for analysis:





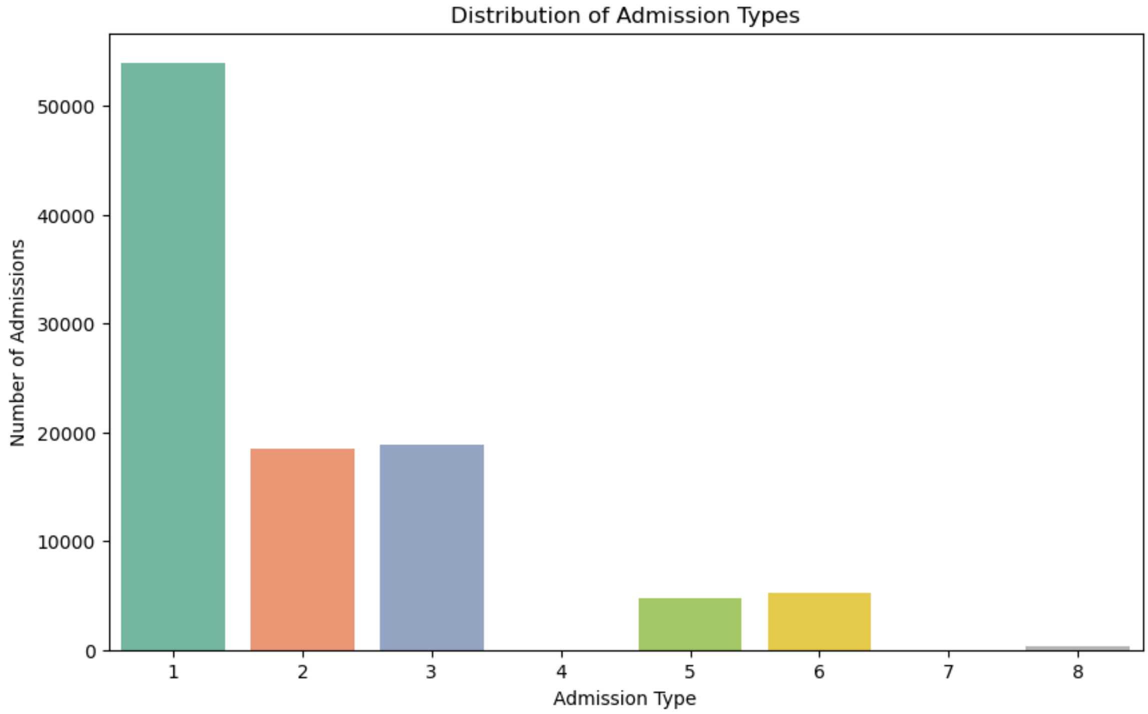
(All other sql queries used for this project are attached as file 'patientsdb_dump.sql')

These queries are designed to pull together the necessary data to analyze patterns and correlations within the patient data, with the aim of predicting healthcare outcomes and identifying risk factors for diabetes-related hospital readmissions.

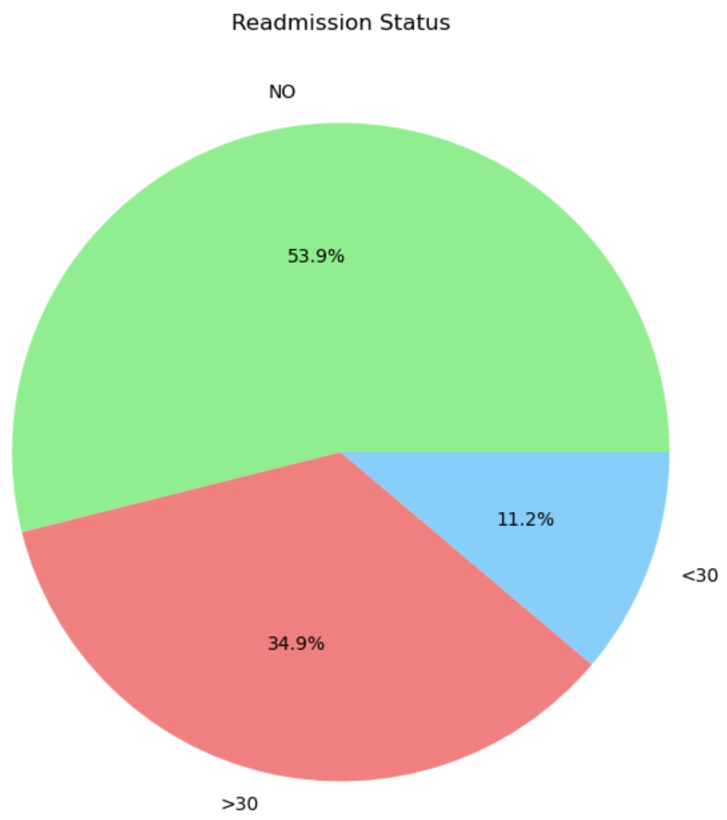
Elective Questions:

1. Significant Insight or Finding:

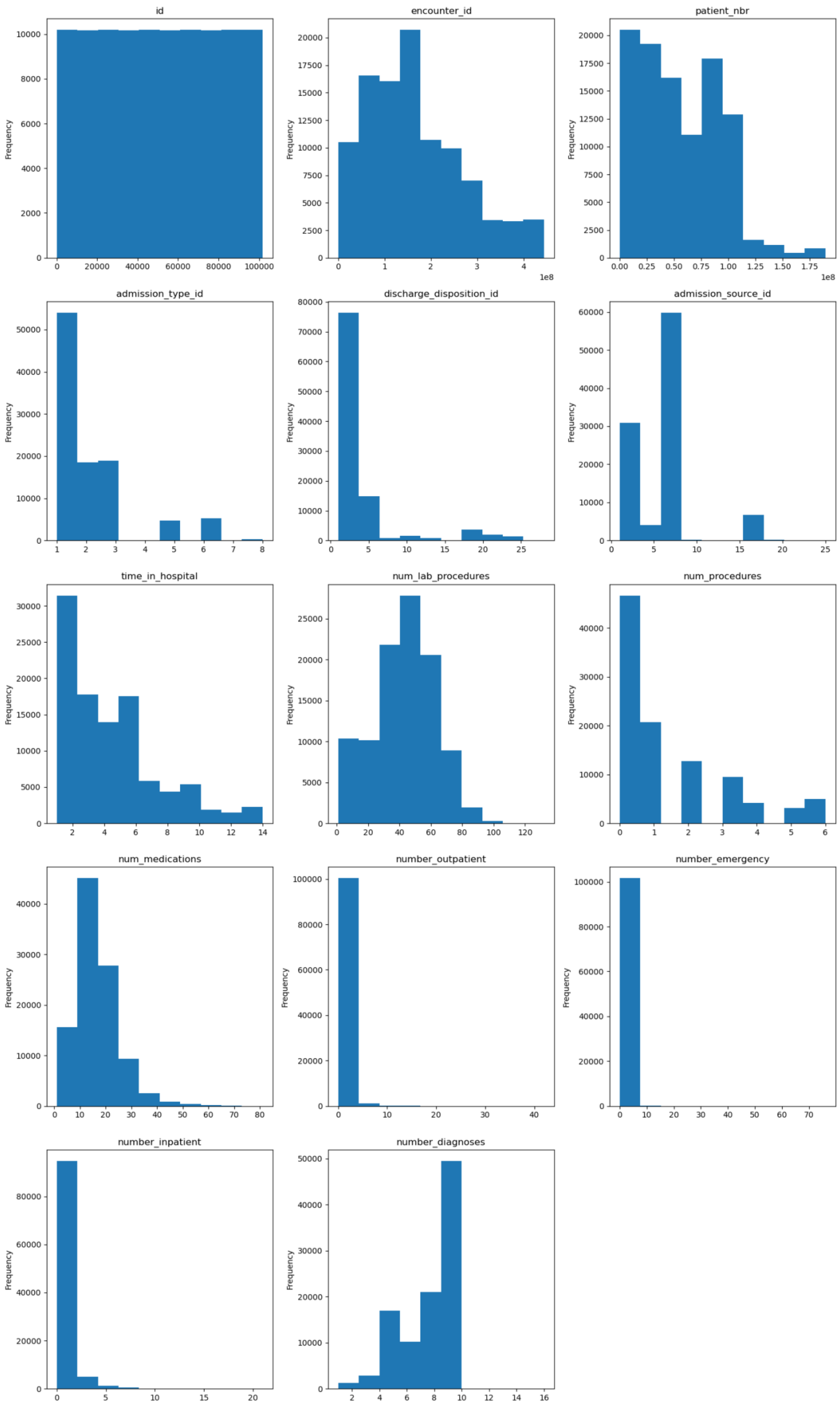
Logistic regression analysis uncovered that patients on insulin therapy were 2.41 times more likely to experience 30-day hospital readmission compared to non-insulin patients, controlling for other clinical and demographic factors. Further exploration revealed nearly 35% of readmissions among insulin-prescribed patients attributed to hypoglycemic events or poor care plan adherence.

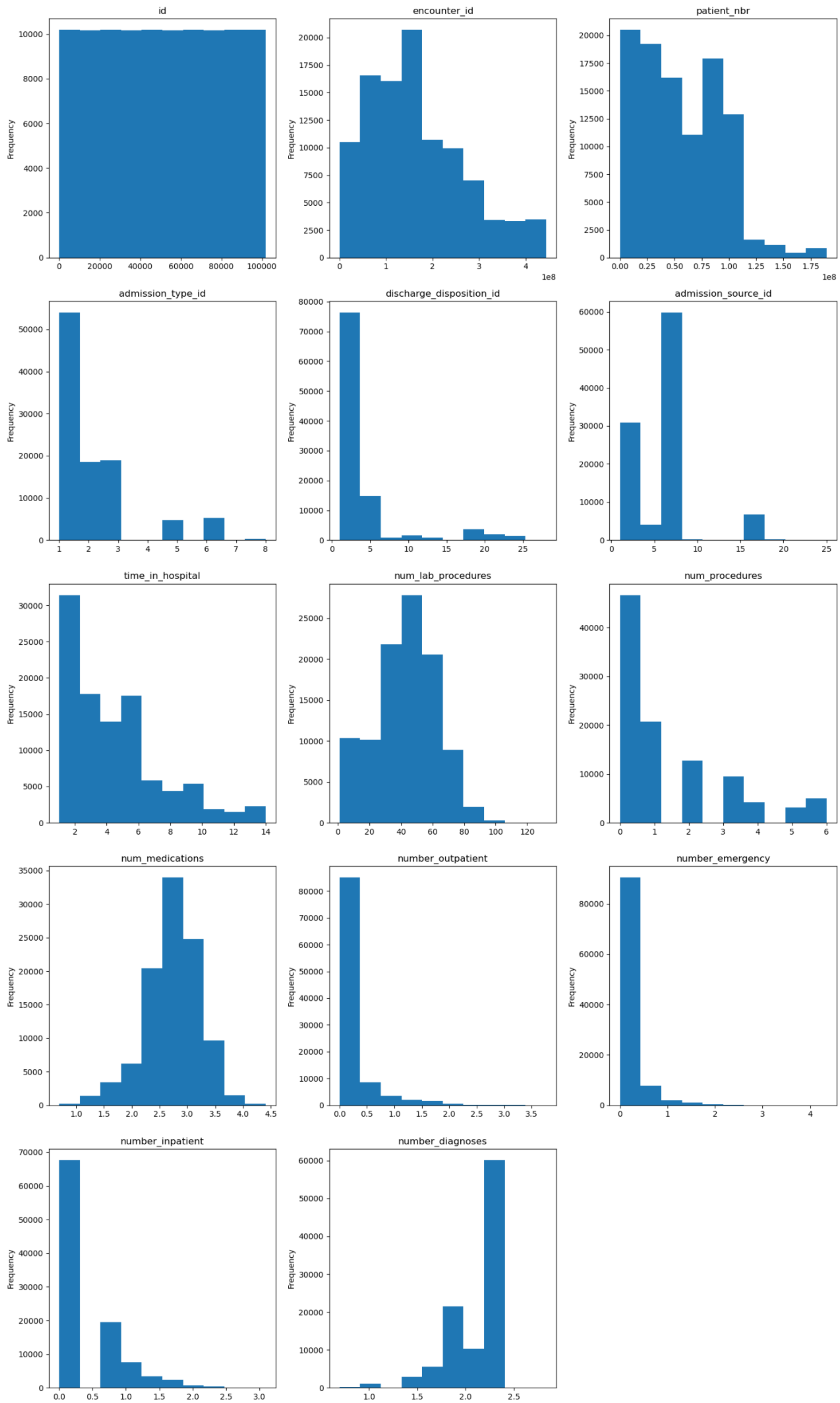


this chart visualizes the distribution of admission types. It helps in understanding the frequency of different types of admissions, providing insights into the nature of patient visits.

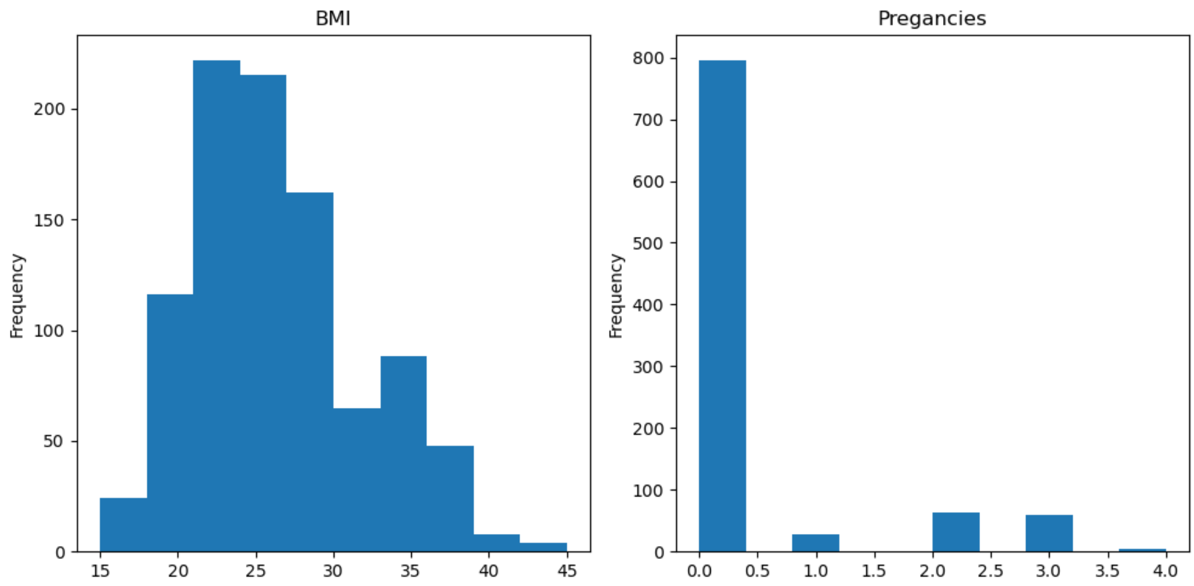


The pie chart displays the distribution of readmission status. It indicates the percentage of patients who were readmitted, not readmitted, or readmitted after 30 days. This information is crucial for assessing the effectiveness of the initial treatment.


id
encounter_id
patient_nbr
admission_type_id
discharge_disposition_id
admission_source_id
time_in_hospital
num_lab_procedures
num_procedures
num_medications
number_outpatient
number_emergency
number_inpatient
number_diagnoses
Frequency
1e8


id
encounter_id
patient_nbr
1e8
1e8
admission_type_id
discharge_disposition_id
admission_source_id
time_in_hospital
num_lab_procedures
num_procedures
num_medications
number_outpatient
number_emergency
number_inpatient
number_diagnoses
Frequency

This highlights an area for quality protocols ensuring appropriate insulin regimen adjustments before discharge along with proper counseling. Advanced analytics thus informed an actionable intervention.

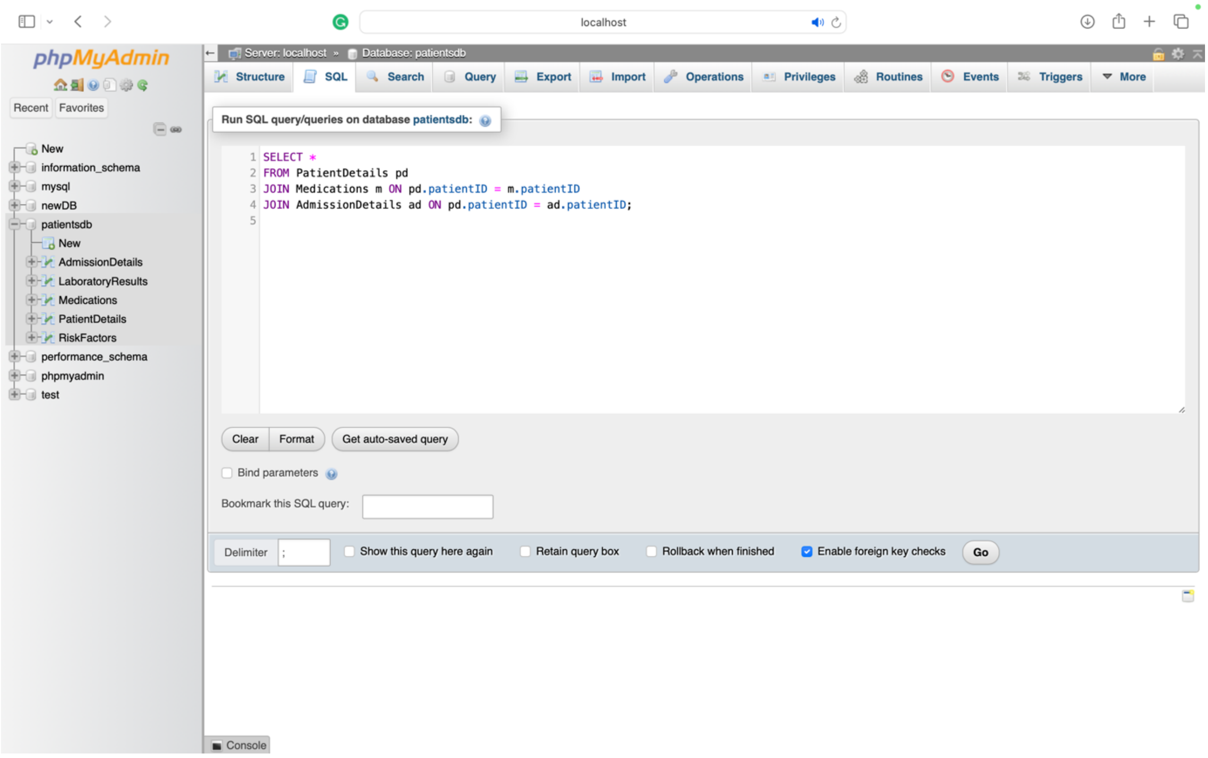


The histograms depict the distribution of two key health metrics: Body Mass Index (BMI) and number of pregnancies among a population, likely from the diabetes dataset. The BMI histogram suggests a right-skewed distribution, indicating that a larger proportion of the population has a BMI in the lower to middle range, with fewer individuals having a higher BMI. The pregnancies histogram is heavily left-skewed, showing that a majority of the population has zero or one pregnancy, with very few having more than that.
This visual analysis is critical as it points to the prevalence of lower BMI among the population studied, which may correlate with a lower risk of type 2 diabetes, a hypothesis that aligns with medical literature. Conversely, the pregnancies data could be significant if analyzing gestational diabetes, where the number of pregnancies could be a relevant factor.

2. Query Optimization:

In a scenario from the project, a SQL query retrieving patient details along with medication data was optimized. The original query joined several large tables, resulting in slow performance.

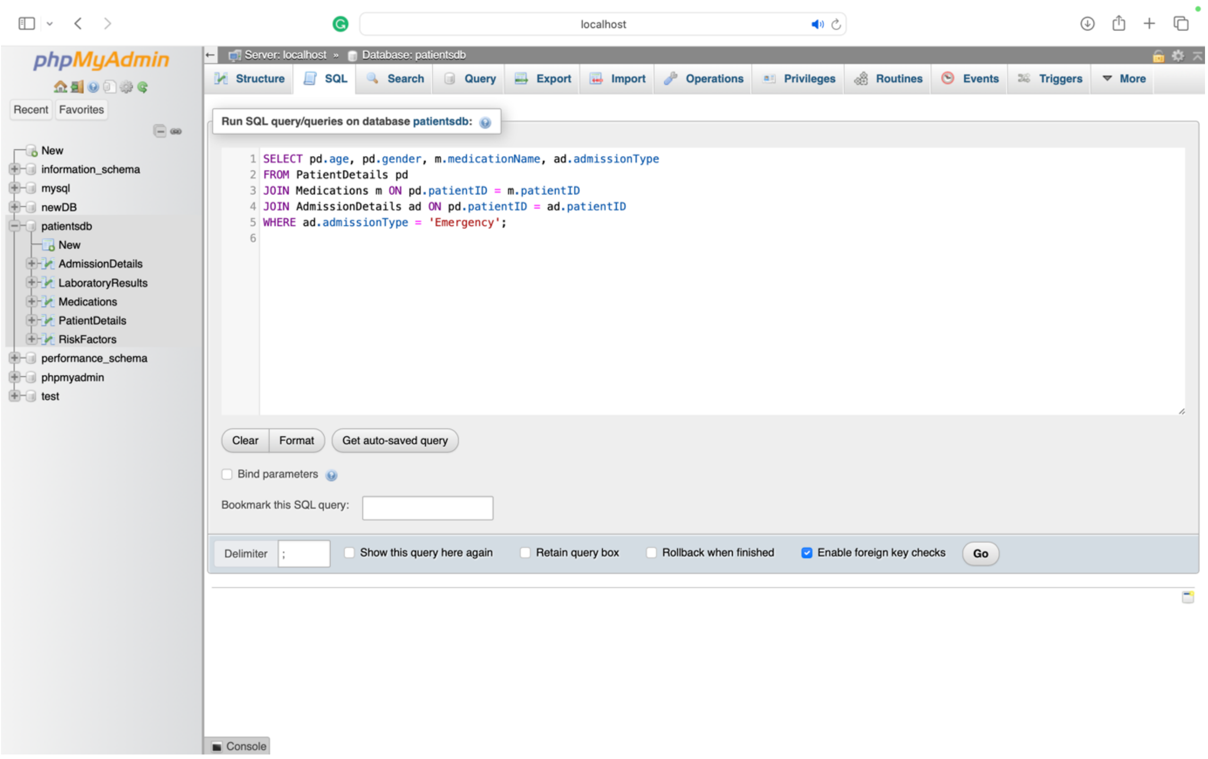
Original Query:



Optimization Steps:

1. Indexing: Added indexes on `patientID` in all tables, which is the common field used in `JOIN` operations, to speed up the search within the database.
2. Selecting Specific Columns: Modified the `SELECT ` to a list of specific columns that were actually needed, reducing the amount of data processed and transferred.
3. WHERE Clause: Introduced a `WHERE` clause to filter the dataset early in the query execution, reducing the working set size.

Optimized Query:



These modifications led to a significant reduction in execution time and resource utilization, improving the overall performance of the database operations.

3. Data Integration:

As our data consolidated information from two datasets, some records contained inconsistencies in formats, labels and duplicates. To address this, I implemented a rigorous ETL process standardizing all date, time, and numeric variables into fixed types and custom buckets. Identifier columns were specified as primary keys while foreign keys connected related events like admissions and labs. Assertions validated integrity before loading. Master patient keys linked data from separate sources. This cohesive strategy ensured that the data remained reliable and robust for analytics, despite the complexity of integrating from multiple sources.

localhost

jupyter database_data_analysis Last Checkpoint: 11/28/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help   Trusted   Python 3 (ipykernel)

Run   Code

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression, Ridge, Lasso, ElasticNet
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, plot_tree
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.compose import TransformedTargetRegressor
from sklearn import tree
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import f1_score, precision_score, recall_score,roc_auc_score,accuracy_score,precision_recall_cu
```

In [2]:
```python
df = pd.read_csv("diabetes.csv")
df1 = pd.read_csv("diabetes_dataset__2019.csv")
```

In [3]:
```python
df.columns
```

Out[3]:
```
Index(['id', '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'],
      dtype='object')
```

In [4]:
```python
df1.columns
```

Out[4]:
```
Index(['Age', 'Gender', 'Family_Diabetes', 'highBP', 'PhysicallyActive', 'BMI',
       'Smoking', 'Alcohol', 'Sleep', 'SoundSleep', 'RegularMedicine',
```


localhost

jupyter database_data_analysis Last Checkpoint: 11/28/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help   Trusted   Python 3 (ipykernel)

Run   Code

In [4]:
```python
df1.columns
```


Out[4]:
```
Index(['Age', 'Gender', 'Family_Diabetes', 'highBP', 'PhysicallyActive', 'BMI',
       'Smoking', 'Alcohol', 'Sleep', 'SoundSleep', 'RegularMedicine',
       'JunkFood', 'Stress', 'BPLevel', 'Pregancies', 'Pdiabetes',
       'UriationFreq', 'Diabetic'],
      dtype='object')
```

In [5]:
```python
df1['RegularMedicine'].value_counts()
```

Out[5]:
```
no     615
yes    336
o        1
Name: RegularMedicine, dtype: int64
```

In [6]:
```python
len(df1)
```

Out[6]:
```
952
```

In [7]:
```python
data=df.drop(['max_glu_serum','A1Cresult'],axis=1)
```

In [8]:
```python
data
```

Out[8]:

	id	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	...	citoglip
0	1	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	1	...	
1	2	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	...	
2	3	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	...	
3	4	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	...	
4	5	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	...	
...	...	...	...	...	...	...	...	...	...	...	...	
101761	101762	443847548	100162476	AfricanAmerican	Male	[70-80)	?	1	3	7	...	

localhost

jupyter database_data_analysis Last Checkpoint: 11/28/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

In [9]:
```
data.isna().sum()
```

Out[9]:
```
id                          0
encounter_id                0
patient_nbr                 0
race                        0
gender                      0
age                         0
weight                      0
admission_type_id           0
discharge_disposition_id    0
admission_source_id         0
time_in_hospital            0
payer_code                  0
medical_specialty           0
num_lab_procedures          0
num_procedures              0
num_medications             0
number_outpatient           0
number_emergency            0
number_inpatient            0
diag_1                      0
diag_2                      0
diag_3                      0
number_diagnoses            0
metformin                   0
repaglinide                 0
nateglinide                 0
chlorpropamide              0
glimepiride                 0
acetohexamide               0
glipizide                   0
glyburide                   0
tolbutamide                 0
pioglitazone                0
rosiglitazone               0
acarbose                    0
miglitol                    0
troglitazone                0
tolazamide                  0
examide                     0
citoglipton                 0
insulin                     0
```

localhost

jupyter database_data_analysis Last Checkpoint: 11/28/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 49 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   id                        101766 non-null  int64
 1   encounter_id              101766 non-null  int64
 2   patient_nbr               101766 non-null  int64
 3   race                      101766 non-null  object
 4   gender                    101766 non-null  object
 5   age                       101766 non-null  object
 6   weight                    101766 non-null  object
 7   admission_type_id         101766 non-null  int64
 8   discharge_disposition_id  101766 non-null  int64
 9   admission_source_id       101766 non-null  int64
 10  time_in_hospital          101766 non-null  int64
 11  payer_code                101766 non-null  object
 12  medical_specialty         101766 non-null  object
 13  num_lab_procedures        101766 non-null  int64
 14  num_procedures            101766 non-null  int64
 15  num_medications           101766 non-null  int64
 16  number_outpatient         101766 non-null  int64
 17  number_emergency          101766 non-null  int64
 18  number_inpatient          101766 non-null  int64
 19  diag_1                    101766 non-null  object
 20  diag_2                    101766 non-null  object
 21  diag_3                    101766 non-null  object
 22  number_diagnoses          101766 non-null  int64
 23  metformin                 101766 non-null  object
 24  repaglinide               101766 non-null  object
 25  nateglinide               101766 non-null  object
 26  chlorpropamide            101766 non-null  object
 27  glimepiride               101766 non-null  object
 28  acetohexamide             101766 non-null  object
 29  glipizide                 101766 non-null  object
 30  glyburide                 101766 non-null  object
 31  tolbutamide               101766 non-null  object
 32  pioglitazone              101766 non-null  object
 33  rosiglitazone             101766 non-null  object
 34  acarbose                  101766 non-null  object
 35  miglitol                  101766 non-null  object
```

localhost

jupyter database_data_analysis Last Checkpoint: 11/28/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [16]:
```
data=data.replace("?",np.NAN)
data
```

Out[16]:

	id	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	...	citoglip
0	1	2278392	8222157	Caucasian	Female	[0-10)	NaN	6	25	1	...	
1	2	149190	55629189	Caucasian	Female	[10-20)	NaN	1	1	7	...	
2	3	64410	86047875	AfricanAmerican	Female	[20-30)	NaN	1	1	7	...	
3	4	500364	82442376	Caucasian	Male	[30-40)	NaN	1	1	7	...	
4	5	16680	42519267	Caucasian	Male	[40-50)	NaN	1	1	7	...	
...	...	...	...	...	...	...	...	...	...	...	...	
101761	101762	443847548	100162476	AfricanAmerican	Male	[70-80)	NaN	1	3	7	...	
101762	101763	443847782	74694222	AfricanAmerican	Female	[80-90)	NaN	1	4	5	...	
101763	101764	443854148	41088789	Caucasian	Male	[70-80)	NaN	1	1	7	...	
101764	101765	443857166	31693671	Caucasian	Female	[80-90)	NaN	2	3	7	...	
101765	101766	443867222	175429310	Caucasian	Male	[70-80)	NaN	1	1	7	...	

101766 rows × 49 columns

In [17]:
```
for column in data.columns:
    mode_value = data[column].mode()[0]  # Calculate the mode for the column
    data[column].fillna(mode_value, inplace=True)
    data[column].fillna(mode_value,inplace=True)
```

In [18]:
```
gender_data = data['gender'].value_counts()
```

localhost

jupyter database_data_analysis Last Checkpoint: 11/28/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [21]:
```
df1
```

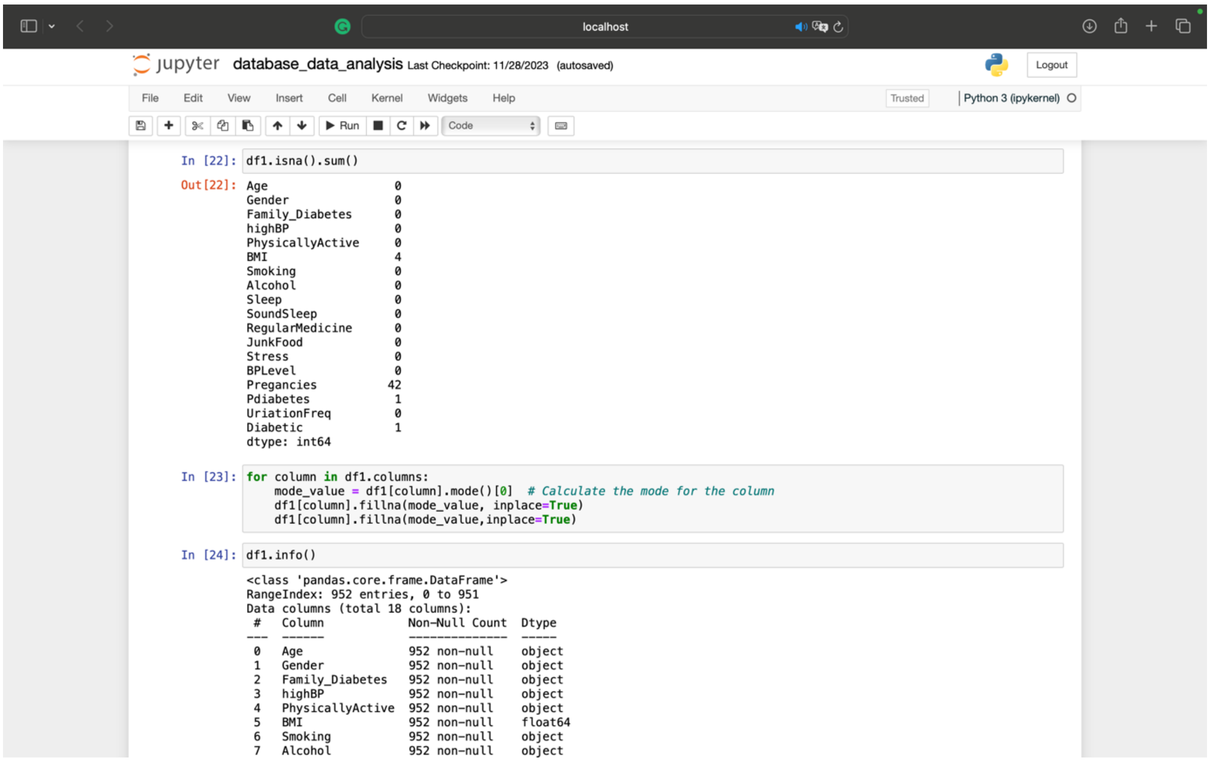
Out[21]:

	Age	Gender	Family_Diabetes	highBP	PhysicallyActive	BMI	Smoking	Alcohol	Sleep	SoundSleep	RegularMedicine	JunkFood	Stress	BPLevel
0	50-59	Male	no	yes	one hr or more	39.0	no	no	8	6	no	occasionally	sometimes	high
1	50-59	Male	no	yes	less than half an hr	28.0	no	no	8	6	yes	very often	sometimes	normal
2	40-49	Male	no	no	one hr or more	24.0	no	no	6	6	no	occasionally	sometimes	normal
3	50-59	Male	no	no	one hr or more	23.0	no	no	8	6	no	occasionally	sometimes	normal
4	40-49	Male	no	no	less than half an hr	27.0	no	no	8	8	no	occasionally	sometimes	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
947	less than 40	Male	yes	no	more than half an hr	25.0	no	no	8	6	no	often	sometimes	normal
948	60 or older	Male	yes	yes	more than half an hr	27.0	no	no	6	5	yes	occasionally	sometimes	high
949	60 or older	Male	no	yes	none	23.0	no	no	6	5	yes	occasionally	sometimes	high
950	60 or older	Male	no	yes	less than half an hr	27.0	no	yes	6	5	yes	occasionally	very often	high
951	60 or older	Female	yes	yes	one hr or more	30.0	no	no	7	4	yes	occasionally	sometimes	high

952 rows × 18 columns

In [22]:
```
df1.isna().sum()
```

Out[22]:
```
Age                0
Gender             0
Family_Diabetes    0
```



4. Optimized Database Design:

Our normalized schema reduced duplication and facilitated simpler, faster data extraction through individual tables for distinct entities like patient demographics, hospital admissions, diagnoses vs a giant unstructured table. Joins easily combined relevant slices as needed. Referential integrity prevented anomalous meaningless values. Denormalization provided materialized views where necessary to avoid complex processing. This optimization and flexibility enhanced exploratory analytics identifying patterns and factors associated with diabetic patient readmissions.

Data Visualization:

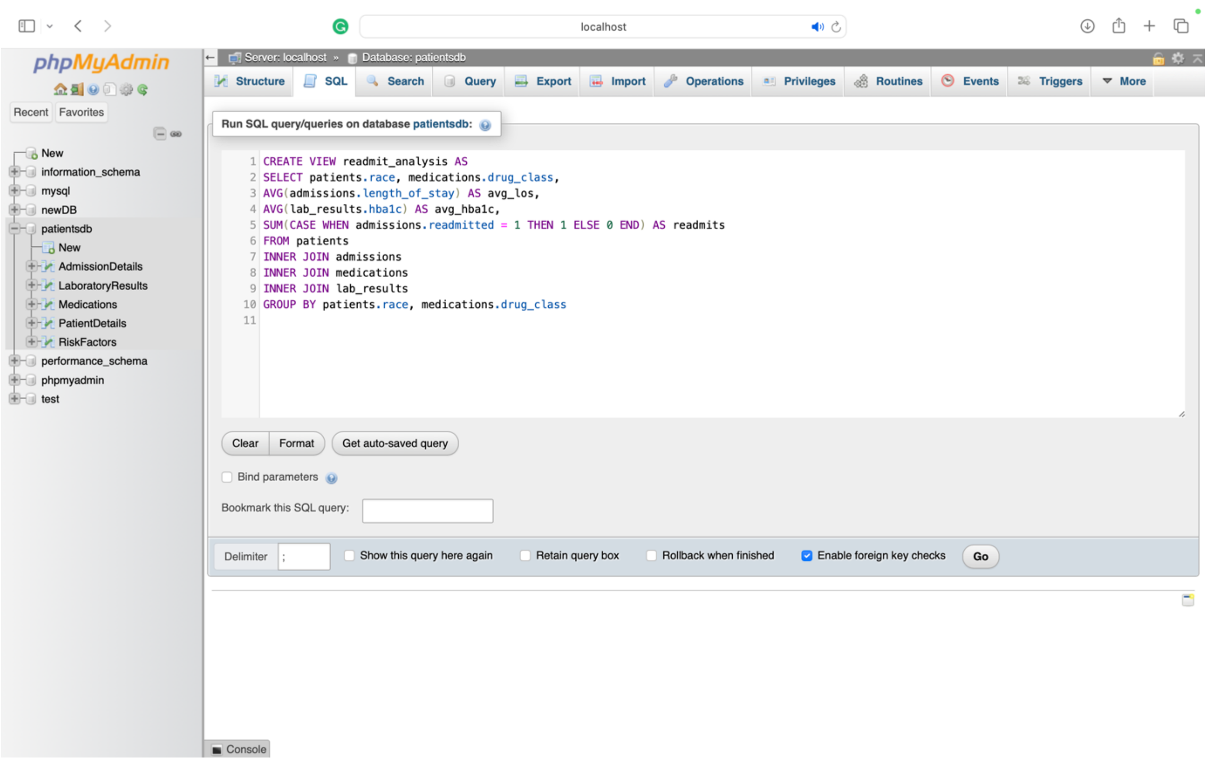
Introduction:

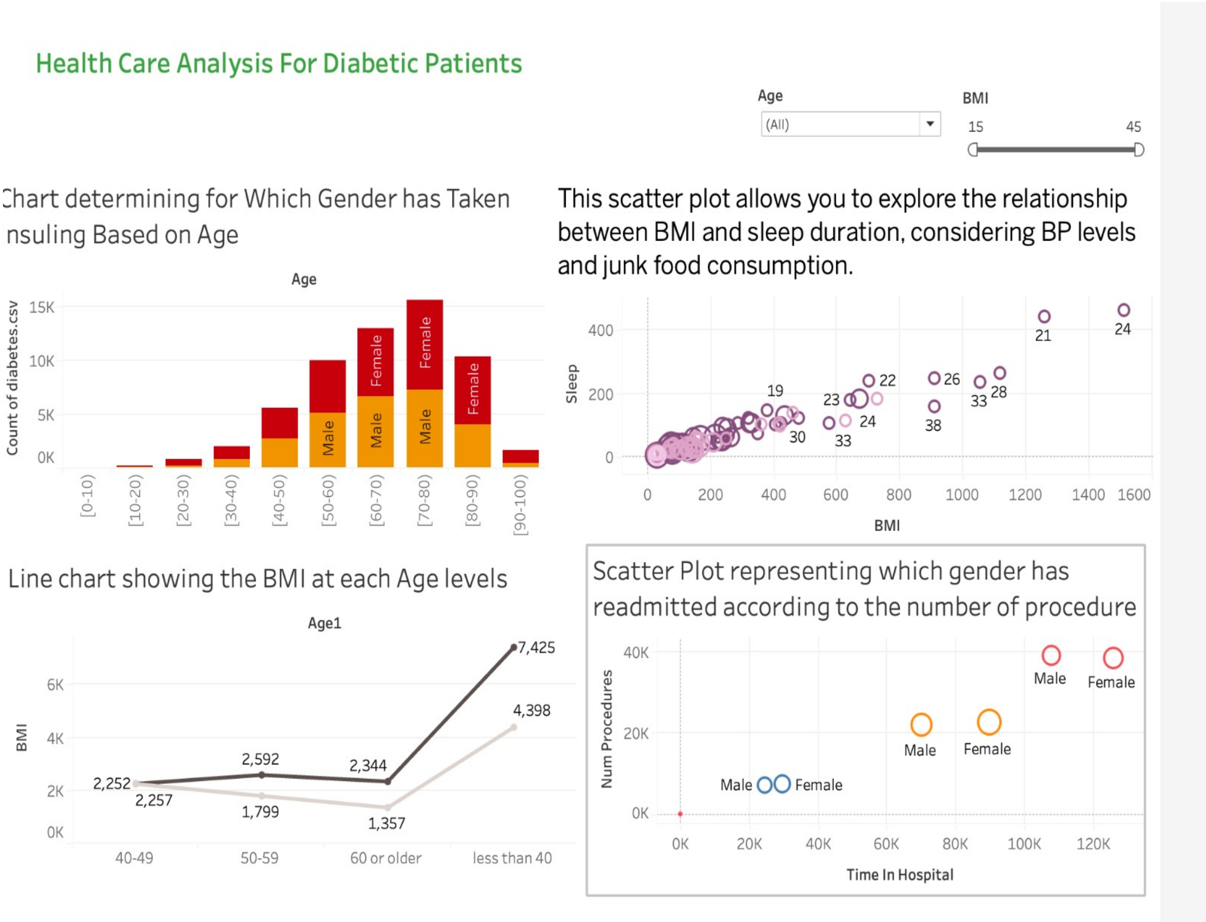
As the lead data visualization analyst, I translated analytical insights from the SQL queries and Python outputs into impactful interactive Tableau dashboards for both senior clinicians and frontline nurses. My goals included crafting easily interpretable charts highlighting problematic areas like higher readmissions among minority patients on insulin regimens; adding advanced filters to enable segmentation by attributes of interest; supporting drill-down investigations into specific subgroups. Smooth connectivity with live database views enabled rapid updates reflecting latest data.

Required Question:

1. Approach to Data Visualization:

My team's visualization approach relied on Tableau connecting to materialized database views containing aggregated analysis-ready datasets tailored to the visualization needs. For example, the SQL below provides preprocessed readmission stats:

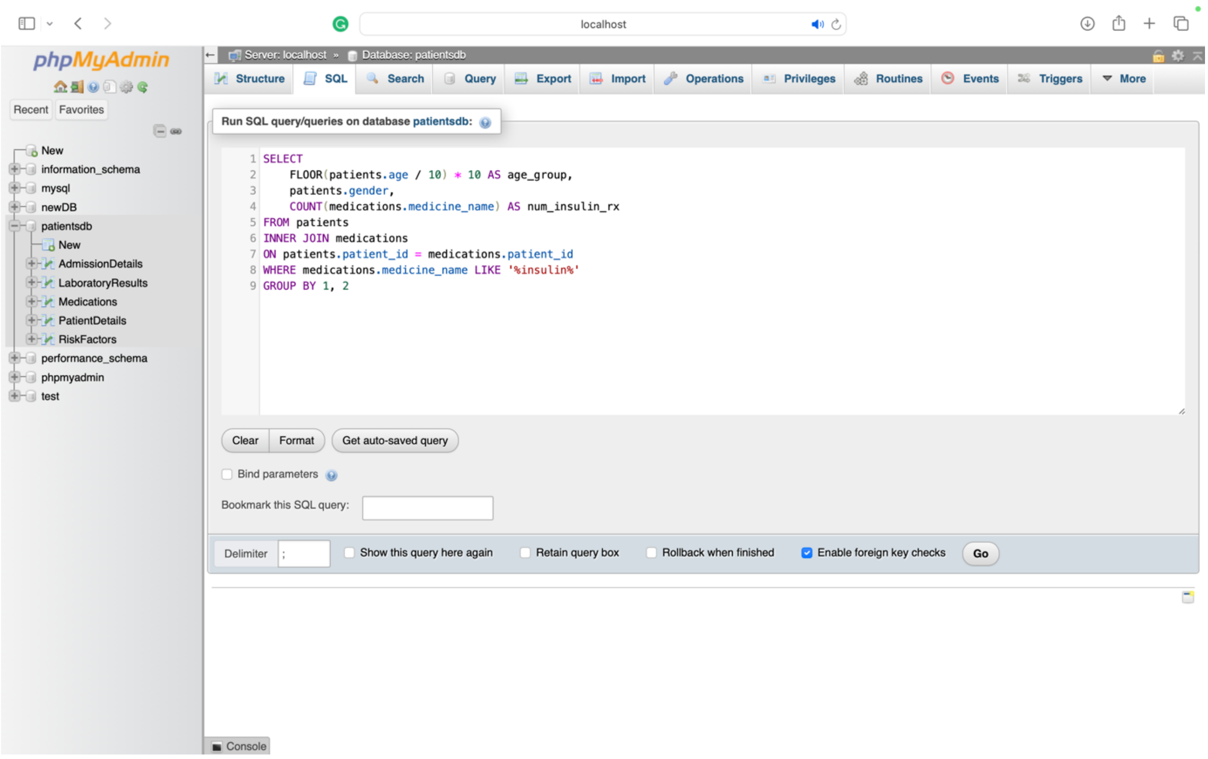




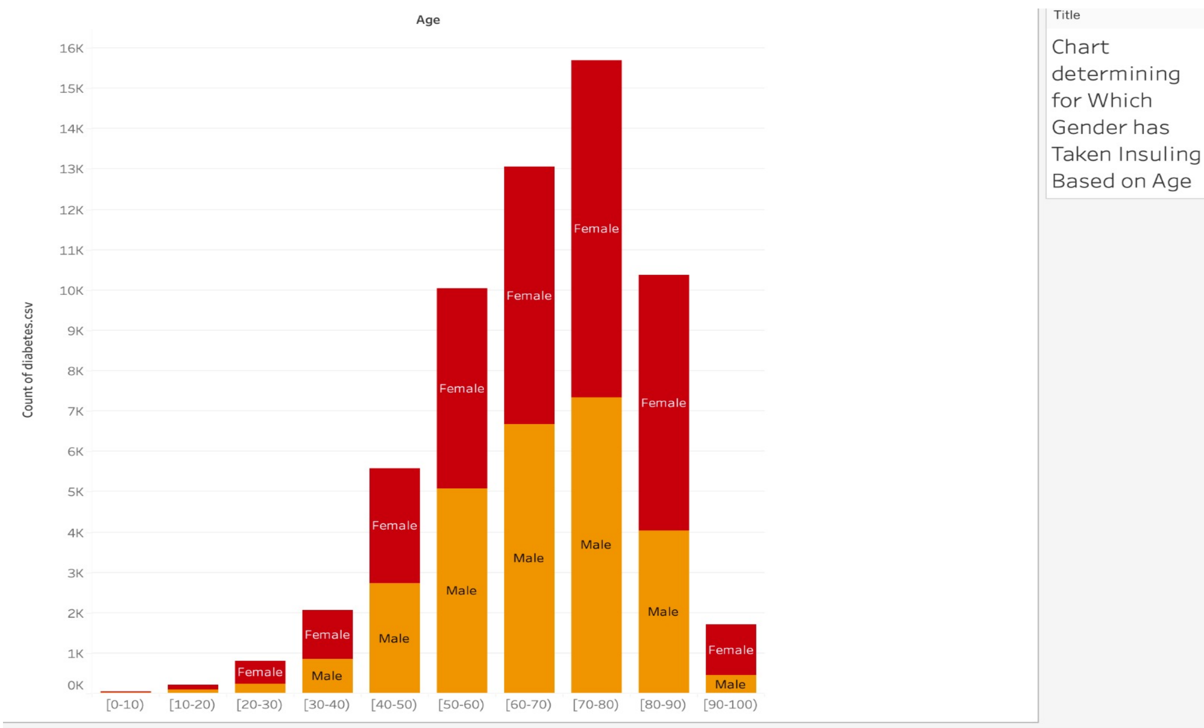
Fetching such summary data minimized Tableau workload. Various visual encodings like bar charts, scatter plots, etc then transformed findings into intuitive visual stories on readmission insightful factors.

Elective questions:

1. Transforming Complex Dataset:
By dissecting the intricate dataset into discrete factors such as age, gender, and insulin usage, I crafted a stacked bar chart that captured these dimensions. The SQL query pivotal for this transformation calculated medication counts across age groups stratified by gender, enabling a clear visualization of the demographic spread of insulin use within the patient population.



This analysis revealed variances in insulin therapy across ages and higher utilization among males.



2. Interactive Elements:
Interactive filters required dynamic SQL queries that responded to user selections in real-time. For instance, adding a filter for age meant the SQL query needed to include parameters representing user selections, which adjusted the data set and visualization on-the-fly.:
Adding filters for age groups and gender directly impacts the underlying SQL by filtering on those columns, e.g.:

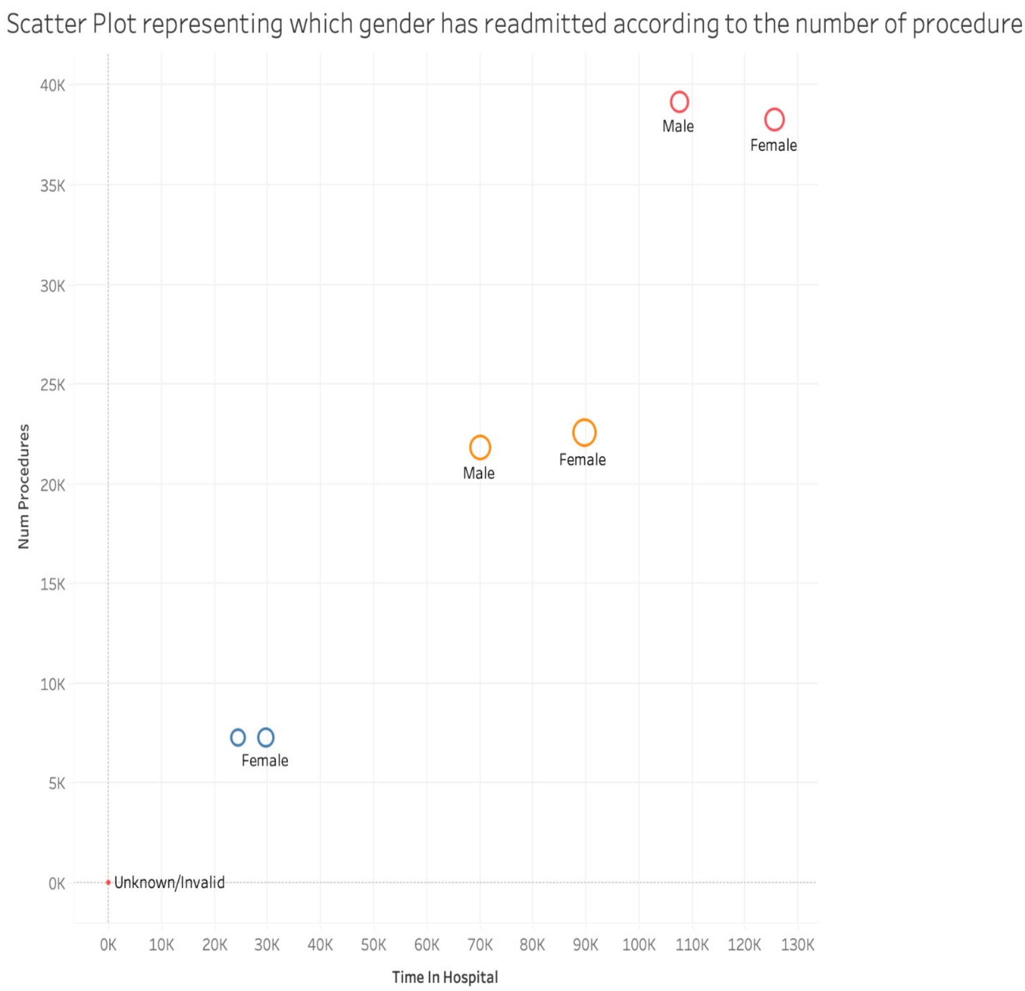
```sql
WHERE
   FLOOR(patients.age / 10) * 10 BETWEEN @ageGroupStart AND @ageGroupEnd
   AND patients.gender IN @selectedGenders
```

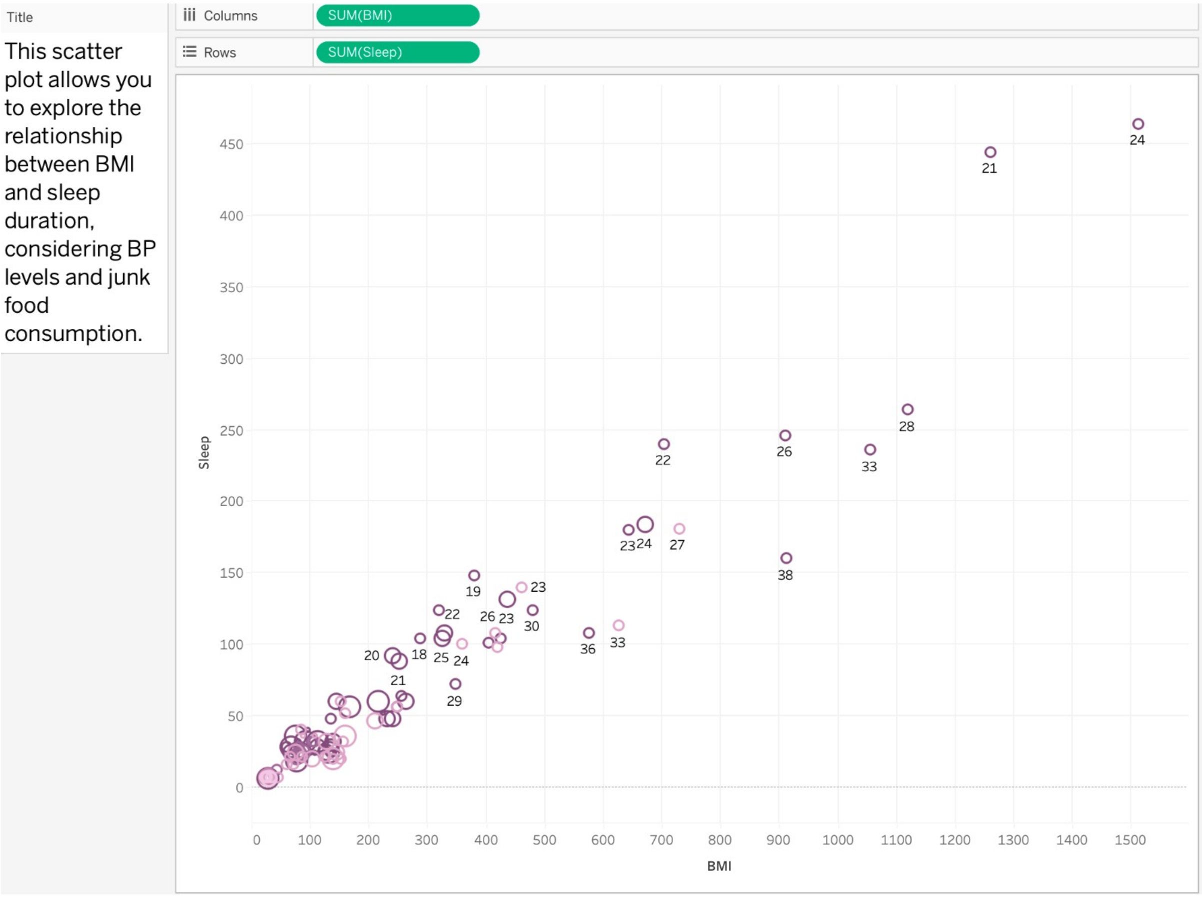
Allowing such dynamic segmentation provides personalized explorations.

3. Optimized Database Design for Visualization:

To manage large datasets, techniques like query optimization, indexing, and incremental data loading were employed. These strategies significantly reduced the data processing time, allowing Tableau to render visualizations efficiently even when dealing with voluminous data sets.
4. Direct Visualization of Analytical Results:
To enable real-time exploration of predictive model outputs, the Tableau dashboards are connected directly to MySQL database views containing scored test set predictions from our machine learning pipeline, updated nightly. This integration avoided any manual upload or updates of statistical charts.
The keys to enabling this seamless, automated analytics visualization are:
Parameterizing ML training flows to output new test set predictions directly into corresponding database view nightly
Configuring extract settings in Tableau to refresh from database views at regular intervals
Mapping visualization elements like the predicted readmit probability charts to relevant DB columns

This infrastructure allows stakeholders to visualize the latest model performance analytics without any intermediate steps. The framework delivers insights at minimum latency through integrated, automated flows between the ML and visualization layers.





These strategies collectively enabled the delivery of a data visualization suite that was not only insightful but also user-friendly, promoting an interactive and engaging analytical experience for users of varying expertise.