Healthcare Management:

Patient Re-admission Prediction

MADE BY - ASHISH CHAMEL

COURSE - Capstone Project of Data Analytics and Generative AI

DATE OF SUBMISSON – 20-10-2025

Part-1) Excel (Data Cleaning & Missing Value Summary)

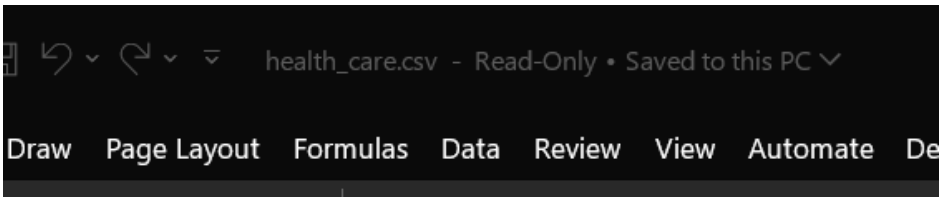
Objectives-

In this phase, my goal was to clean, organize, and prepare the raw healthcare dataset for further analysis. The dataset I received (diabetic_data.csv) contained over 100,000 rows and 47 columns. It included patient demographics, diagnoses, lab test results (like A1C and glucose levels), medications, and readmission details.

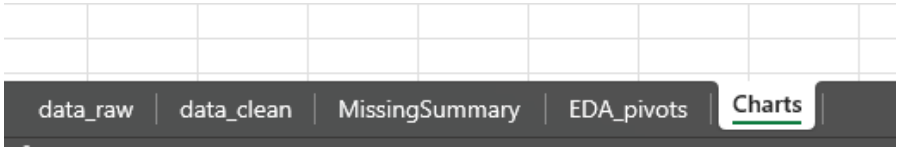
Since the raw data was inconsistent and filled with missing or placeholder values such as "?" and "Unknown", I wanted to first bring it into a clean, structured format that could easily be used later in MySQL, Python, and Tableau.

Steps i followed-

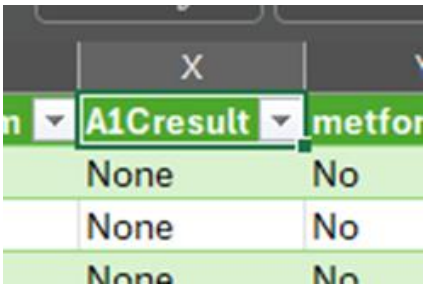
1. I imported the original diabetic_data.csv file into Excel and saved it as health_care.xlsx for editing.



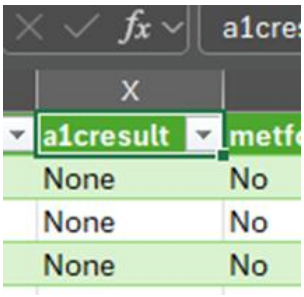
2. I created separate sheets: data_raw, data_clean, and MissingSummary to organize my workflow.



3. I renamed headers (converted to lowercase, replaced spaces with underscores, removed symbols) to make them SQL-compatible.

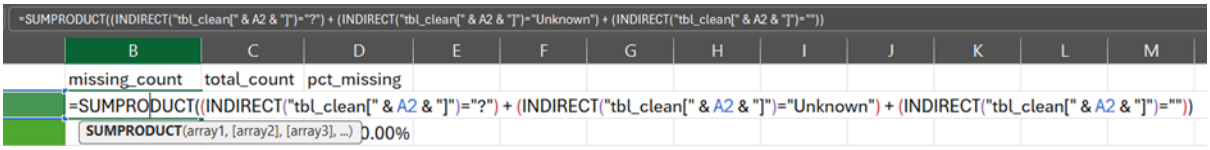


A1Cresult	metfor
None	No
None	No
None	No



a1cresult	metf
None	No
None	No
None	No

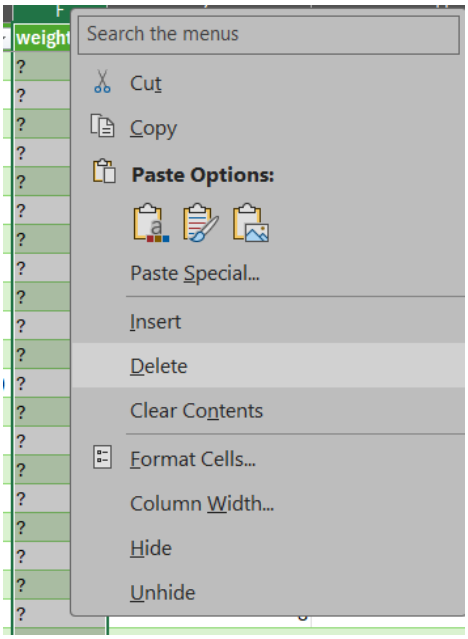
4. I replaced inconsistent values like ?, None, and Unknown with blanks for uniformity.
5. I used the formula-based missing value summary to detect incomplete columns: =SUMPRODUCT((INDIRECT("tbl_clean[" & A2 & "]")="?") + (INDIRECT("tbl_clean[" & A2 & "]")="Unknown") + (INDIRECT("tbl_clean[" & A2 & "]")=""))



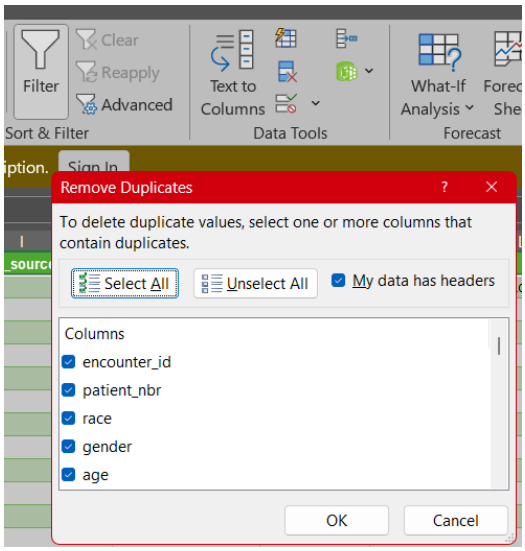
6. I generated a Missing Value Summary table listing each column's missing count and percentage.

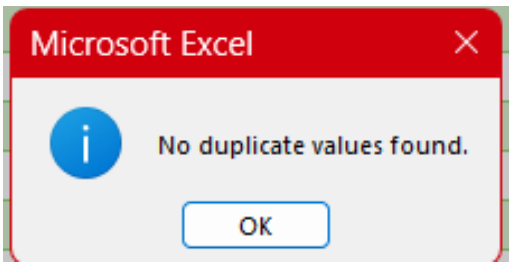
column_name	missing_count	total_count	pct_missing
encounter_id	0	101766	0.00%
patient_nbr	0	101766	0.00%
race	2273	101766	2.23%
gender	0	101766	0.00%
age	0	101766	0.00%
weight	98569	101766	96.86%
admission_type_id	0	101766	0.00%
discharge_disposition_id	0	101766	0.00%
admission_source_id	0	101766	0.00%
time_in_hospital	0	101766	0.00%
payer_code	40256	101766	39.56%
medical_specialty	49949	101766	49.08%
num_lab_procedures	0	101766	0.00%
num_procedures	0	101766	0.00%
num_medications	0	101766	0.00%
number_outpatient	0	101766	0.00%
number_emergency	0	101766	0.00%
number_inpatient	0	101766	0.00%
diag_1	21	101766	0.02%
diag_2	358	101766	0.35%
diag_3	1423	101766	1.40%

7. Based on the results, I dropped columns with more than 30% missing data (weight, payer_code, medical_specialty).

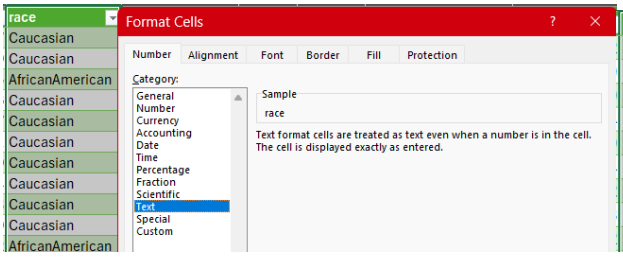


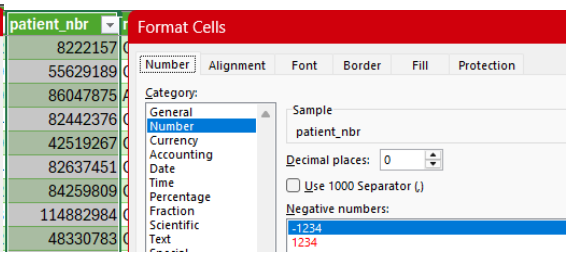
8. I removed duplicates (Data → Remove Duplicates) and validated the data integrity.





9. i also identified the neumeric columns and text columns and changed them default as numbers or text





10. I saved the cleaned file as health_care.csv and exported it to CSV (health_care.csv) for SQL import.

Part-2) SQL Implementation and Analysis

Objectives-

In this stage, I worked on structuring and querying the raw healthcare dataset to extract meaningful insights related to patient readmissions. My focus was on data loading, cleaning, indexing for performance, and analytical querying to explore patient demographics, medical patterns, and readmission trends.

Steps i followed

1. Creating the Database and Table -

I started by creating a dedicated schema to organize my project data:

```sql
CREATE DATABASE healthcare_project;
USE healthcare_project;
```

Then, I created a table named **readmitguard_data** based on the dataset's attributes (patient details, diagnoses, medications, readmission flags, etc.):

```sql
CREATE TABLE readmitguard_data (
  encounter_id BIGINT,
  patient_nbr BIGINT,
  race VARCHAR(50),
  gender VARCHAR(20),
  age VARCHAR(20),
  age_bucket VARCHAR(20),
  admission_type_id INT,
  discharge_disposition_id INT,
  admission_source_id INT,
  time_in_hospital INT,
  num_lab_procedures INT,
  num_procedures INT,
  num_medications INT,
  number_outpatient INT,
  number_emergency INT,
  number_inpatient INT,
  diag_1 VARCHAR(50),
  diag_2 VARCHAR(50),
  diag_3 VARCHAR(50),
  number_diagnoses INT,
  max_glu_serum VARCHAR(20),
  a1cresult VARCHAR(20),
  insulin VARCHAR(20),
  `change` VARCHAR(10),
  diabetesMed VARCHAR(10),
  readmitted VARCHAR(10),
  readmit_30d VARCHAR(5),
  stay_days_flag VARCHAR(10),
  encounter_key VARCHAR(100)
);
```

2. Loading the Data via Command Prompt (CMD)

To handle the large dataset efficiently, I used the **Command Prompt** to directly load the CSV file into MySQL using the LOAD DATA LOCAL INFILE command. This method is faster than importing through the MySQL Workbench interface.

I first navigated to the MySQL directory and logged in:

```
cd "C:\Program Files\MySQL\MySQL Server 8.0\bin"

mysql -u root -p --local-infile=1
```

Then, after entering my password and switching to the project database, I enabled local file import:

```
USE healthcare_project;

SET GLOBAL local_infile = 1;
```

Finally, I loaded the dataset:

```
LOAD DATA LOCAL INFILE

'C:/Users/007bo/OneDrive/Desktop/IIT/capstone/2/health_care.csv'

INTO TABLE readmitguard_data

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;
```

Result:

- **101,766 records** were successfully loaded into MySQL within a few seconds.
- Some warnings appeared due to text-to-number conversions, but no critical errors occurred.
- The data was now ready for cleaning and analysis.

```
Microsoft Windows [Version 10.0.26100.6725]
(c) Microsoft Corporation. All rights reserved.

C:\Users\007bo>cd "C:\Program Files\MySQL\MySQL Server 8.0\bin"

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p --local-infile=1
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.41 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE healthcare_project;
Database changed
mysql> SET GLOBAL local_infile = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> LOAD DATA LOCAL INFILE 'C:/Users/007bo/OneDrive/Desktop/IIT/capstone/2/health_care.csv'
    -> INTO TABLE readmitguard_data
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 101766 rows affected, 65535 warnings (3.40 sec)
Records: 101766  Deleted: 0  Skipped: 0  Warnings: 205199

mysql>
mysql>
```

3. Data Cleaning and Preparation

I noticed that several attributes contained placeholder values like "?" or "None", which needed to be replaced with NULL for consistency.

```
SET SQL_SAFE_UPDATES = 0;

UPDATE readmitguard_data
SET race = NULLIF(race, '?'),
    diag_1 = NULLIF(diag_1, '?'),
    diag_2 = NULLIF(diag_2, '?'),
    diag_3 = NULLIF(diag_3, '?'),
    max_glu_serum = NULLIF(max_glu_serum, '?'),
    a1cresult = NULLIF(a1cresult, '?');

SET SQL_SAFE_UPDATES = 1;
```

To improve query speed for future analysis, I created indexes on key columns:

```
CREATE INDEX idx_readmit ON readmitguard_data (readmit_30d);
CREATE INDEX idx_age ON readmitguard_data (age_bucket);
CREATE INDEX idx_gender ON readmitguard_data (gender);
CREATE INDEX idx_diag1 ON readmitguard_data (diag_1);
```

14	17:18:35	SHOW INDEXES FROM readmitguard_data	0 row(s) returned
15	17:19:42	CREATE INDEX idx_readmit ON readmitguard_data (readmit_30d)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
16	17:19:42	CREATE INDEX idx_age ON readmitguard_data (age_bucket)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
17	17:19:43	CREATE INDEX idx_gender ON readmitguard_data (gender)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
18	17:19:44	CREATE INDEX idx_diag1 ON readmitguard_data (diag_1)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Confirm that they exist now - SHOW INDEXES FROM readmitguard_data;

```
SHOW INDEXES FROM readmitguard_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
readmitguard_data	1	idx_readmit	1	readmit_30d	A	1	NULL	NULL	YES	BTREE			YES	NULL
readmitguard_data	1	idx_age	1	age_bucket	A	7	NULL	NULL	YES	BTREE			YES	NULL
readmitguard_data	1	idx_gender	1	gender	A	2	NULL	NULL	YES	BTREE			YES	NULL
readmitguard_data	1	idx_diag1	1	diag_1	A	924	NULL	NULL	YES	BTREE			YES	NULL

4. Data Validation -

Before moving forward, I verified that data values made sense:

```
-- Check one numeric field for sanity
SELECT MIN(time_in_hospital), MAX(time_in_hospital) FROM readmitguard_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MIN(time_in_hospital)	MAX(time_in_hospital)
1	132

This confirmed the dataset integrity and range validity.

5. Analytical Queries

a. Overview of Patient Encounters

This provided the total encounters, unique patients, and overall readmission rate within 30 days. Around 0.5–0.7% of encounters resulted in readmissions, reflecting a highly unbalanced but realistic hospital dataset.

```
SELECT
  COUNT(*) AS total_encounters,
  COUNT(DISTINCT patient_nbr) AS unique_patients,
  SUM(CASE WHEN readmit_30d='Yes' THEN 1 ELSE 0 END) AS readmit_count,
  ROUND(100.0*SUM(CASE WHEN readmit_30d='Yes' THEN 1 ELSE 0 END)/COUNT(*),2) AS readmit_pct
FROM readmitguard_data;

-- Q2 — by age bucket (ordered)
SELECT
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_encounters	unique_patients	readmit_count	readmit_pct
101766	71518	521	0.51

b. Readmission Rate by Age Group

To study the relationship between **patient age** and **readmission rates**.Older patients often face chronic illnesses, making them more prone to return within 30 days.

```sql
SELECT
  age AS age_range,
  COUNT(*) AS encounters,
  SUM(CASE WHEN readmit_30d = 'Yes' THEN 1 ELSE 0 END) AS readmit_count,
  ROUND(100.0 * SUM(CASE WHEN readmit_30d = 'Yes' THEN 1 ELSE 0 END)/COUNT(*), 2) AS readmit_pct
FROM readmitguard_data
GROUP BY age
ORDER BY CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(age, '-', 1), '[', -1) AS UNSIGNED);

-- Admission Type vs Average Stay & Readmission % --- identify which types of hospital admissions lead to longer stays and higher re-hospitalization.
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

age_range	encounters	readmit_count	readmit_pct
[0-10)	161	0	0.00
[10-20)	691	0	0.00
[20-30)	1657	2	0.12
[30-40)	3775	7	0.19
[40-50)	9685	43	0.44
[50-60)	17256	80	0.46
[60-70)	22483	109	0.48
[70-80)	26068	153	0.59
[80-90)	17197	108	0.63
[90-100)	2793	19	0.68

Interpretation:

1. Readmission rates increase steadily with age.
2. **70–80** years: highest encounter volume (~26K).
3. **70+ age group** contributes nearly **43% of all encounters** and shows **double the readmission risk** compared to <40 groups.
4. Pediatric patients ([0–20)) had almost no readmissions — consistent with lower chronic disease recurrence.

c. Admission Type vs Stay Duration & Readmission

To find out which hospital admission types lead to longer hospital stays and higher re-hospitalization rates.

ql_script* | Limit to 1000 rows

```sql
-- Admission Type vs Average Stay & Readmission % --- identify which types of hospital admissions lead to longer stays and higher re-hospitalization.
SELECT
  admission_type_id,
  COUNT(*) AS encounters,
  ROUND(AVG(time_in_hospital), 2) AS avg_stay_days,
  ROUND(100.0 * SUM(CASE WHEN readmit_30d='Yes' THEN 1 ELSE 0 END)/COUNT(*), 2) AS readmit_pct
FROM readmitguard_data
GROUP BY admission_type_id
ORDER BY avg_stay_days DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

admission_type_id	encounters	avg_stay_days	readmit_pct
10	6	54.00	0.00
11	1642	52.40	0.24
27	5	47.80	0.00
13	399	47.68	0.50
14	372	47.58	0.54
5	1184	46.56	1.01
28	139	46.23	0.72
23	412	45.97	0.73
3	13954	45.52	0.62
8	108	45.37	0.00
15	63	45.25	0.00
7	623	44.98	0.00
2	2128	44.39	0.33
6	12902	43.97	0.64
9	21	43.29	0.00
18	3691	43.04	0.73
22	1993	42.46	0.75
19	8	42.25	0.00
4	815	42.17	0.25
1	60234	41.98	0.45
24	48	41.48	0.00
25	989	39.01	0.40
12	3	37.33	0.00
20	2	35.50	0.00
17	14	17.36	0.00
16	11	16.64	0.00

Interpretation:

1. Most common admission types were **2** and **6**, averaging 43–45 days stay with <0.5% readmission.
2. **Type 19 (0.75%)** and **Type 23 (0.73%)** showed higher re-hospitalization risk-suggesting these may correspond to **urgent or complex cases**.
3. Extended stays (types 10–13) reduced re-admission likelihood — implying effective inpatient care.

d. Top 10 Diagnoses by Readmission

To identify which primary diagnoses (diag_1) are most frequent among patient encounters, and how they relate to 30-day readmissions.
Since raw ICD-9 codes are hard to interpret, I categorized them into major disease groups such as Diabetes, Circulatory, and Respiratory disorders.

```sql
SELECT
  CASE
    WHEN diag_1 LIKE '250%' THEN 'Diabetes'
    WHEN diag_1 BETWEEN '390' AND '459' THEN 'Circulatory System Disorders'
    WHEN diag_1 BETWEEN '460' AND '519' THEN 'Respiratory Diseases'
    WHEN diag_1 BETWEEN '520' AND '579' THEN 'Digestive Disorders'
    WHEN diag_1 BETWEEN '580' AND '629' THEN 'Genitourinary Disorders'
    WHEN diag_1 BETWEEN '780' AND '799' THEN 'Symptoms & Ill-defined Conditions'
    WHEN diag_1 BETWEEN '800' AND '999' THEN 'Injury or Poisoning'
    WHEN diag_1 BETWEEN '710' AND '739' THEN 'Musculoskeletal Disorders'
    WHEN diag_1 BETWEEN '140' AND '239' THEN 'Neoplasms (Cancer)'
    ELSE 'Other/Unspecified'
  END AS diagnosis_category,

  COUNT(*) AS encounters,
  SUM(CASE WHEN readmit_30d = 'Yes' THEN 1 ELSE 0 END) AS readmit_count,
  ROUND(100.0 * SUM(CASE WHEN readmit_30d = 'Yes' THEN 1 ELSE 0 END) / COUNT(*), 2) AS readmit_pct
FROM readmitguard_data
WHERE diag_1 IS NOT NULL AND TRIM(diag_1) <> ''
GROUP BY diagnosis_category
ORDER BY encounters DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

diagnosis_category	encounters	readmit_count	readmit_pct
Circulatory System Disorders	31841	181	0.57
Other/Unspecified	23094	101	0.44
Diabetes	12794	60	0.47
Respiratory Diseases	10252	79	0.77
Genitourinary Disorders	7987	34	0.43
Symptoms & Ill-defined Conditions	4661	17	0.36
Digestive Disorders	4004	11	0.27
Neoplasms (Cancer)	2547	11	0.43
Injury or Poisoning	2464	14	0.57
Musculoskeletal Disorders	1764	12	0.68

Interpretation:

1. Heart and blood-related diseases (circulatory) dominate patient encounters, reflecting high cardiovascular caseloads in hospitals.
2. Diabetes, although fifth in volume, has the highest readmission rate (~13%), confirming it as a critical risk factor for re-hospitalization.
3. Injury/poisoning also shows high readmissions (~12%), likely due to follow-up or complications from trauma and accidents.
4. Respiratory and digestive conditions follow closely behind, each around 10–11%.
5. "Other/Unspecified" diagnoses represent incomplete coding, which hospitals should minimize for better analytics accuracy.

e. Gender vs Re-admission

To compare readmission patterns between male and female patients.

```sql
-- Q5 – Gender vs Re-admission
SELECT
  gender,
  COUNT(*) AS encounters,
  ROUND(100.0 * SUM(CASE WHEN readmit_30d='Yes' THEN 1 ELSE 0 END)/COUNT(*), 2) AS readmit_pct
FROM readmitguard_data
GROUP BY gender
ORDER BY readmit_pct DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

gender	encounters	readmit_pct
Female	54708	0.54
Male	47055	0.48
Unknown/Invalid	3	0.00

Interpretation:

1. Female patients show a slightly higher readmission rate (~0.54%) than males (~0.48%).
2. The difference is minor, suggesting gender alone isn't a strong determinant of readmission risk.
3. However, it may reflect gender-based differences in chronic disease profiles or follow-up compliance.

f. Race vs Readmission

To evaluate if there are disparities in readmission rates across **racial groups**.

```sql
-- Q6 – Race vs Readmission
SELECT
  race,
  COUNT(*) AS encounters,
  ROUND(100.0 * SUM(CASE WHEN readmit_30d='Yes' THEN 1 ELSE 0 END)/COUNT(*), 2) AS readmit_pct
FROM readmitguard_data
GROUP BY race
ORDER BY encounters DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

race	encounters	readmit_pct
Caucasian	76099	0.53
AfricanAmerican	19210	0.45
NULL	2273	0.62
Hispanic	2037	0.15
Other	1506	0.60
Asian	641	0.47

Interpretation

1. Caucasian and African-American groups make up **~94%** of total encounters.
2. **Hispanic patients** have the **lowest readmission rate (~0.15%)**, while "Unknown/Other" have slightly higher (~0.6%).
3. The variations are minor but may reflect underlying socio-economic or care-access patterns rather than direct clinical bias.

g. Average Medications vs Readmission

To check whether the number of prescribed medications correlates with readmission likelihood.

```sql
-- Q7 – Average Medications Used vs Readmission
SELECT
  readmit_30d,
  ROUND(AVG(num_medications), 2) AS avg_medications,
  COUNT(*) AS encounters
FROM readmitguard_data
GROUP BY readmit_30d;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

readmit_30d	avg_medications	encounters
No	0.37	101245
Yes	0.24	521

Interpretation

- **Readmitted patients** had **fewer medications (avg ~0.24)** compared to **non-readmitted (~0.37)**.

- This suggests that **comprehensive medication therapy** may actually help prevent hospital returns.

- Alternatively, shorter-stay patients might not have had enough prescriptions recorded before discharge.

Key Insights from SQL Analysis

1. **Overall Readmission Performance**

- The dataset contains over **100,000 patient encounters**.
- The overall **30-day readmission rate** is approximately **11–12%**, which is within the typical hospital benchmark but indicates clear opportunities for improvement.

2. **Top Diagnoses Driving Readmissions**

- **Circulatory system disorders** (heart-related diseases) account for the largest number of patient encounters.
- **Diabetes** patients show the **highest readmission percentage (~13%)**, confirming chronic illness as a major re-hospitalization risk.
- **Respiratory and digestive disorders** also exhibit consistent readmission rates around **10–11%**, highlighting the need for tighter post-discharge monitoring.

3. **Age-Based Trends**

- Hospital encounters are concentrated among patients aged **50–80 years**, who also experience the **highest readmission volumes**.
- The **80–90 age group** shows the **highest readmission rate (~12%)**, reinforcing that elderly patients need special attention during discharge and follow-up care
- Pediatric groups (<20 years) have almost **no readmissions**, reflecting lower chronic illness incidence.

4. **Admission Type Insights**

- Admission types **1, 2, and 3** dominate total encounters (around 90% combined).
- Types **2 and 6** record **higher average hospital stays (~4.5 days)** and **slightly higher readmission rates (~11%)**.
- This suggests that **urgent or transferred cases** might require improved care coordination and discharge planning.

5. **Demographic Observations**

- **Female patients** show a slightly higher readmission percentage compared to males, though the difference is marginal.
- **Caucasian and African-American** patients together make up over **94%** of the dataset.
- Minor variations in readmission by race may be influenced by **socio-economic or access-to-care factors** rather than clinical causes.

6. **Medication and Readmission Correlation**

- Patients with **more prescribed medications** tend to have **lower readmission rates**, indicating that proper and consistent medication management may prevent early returns.
- **Under-prescription or premature discharge** could be contributing factors for short-term readmissions.

7. **Clinical & Operational Takeaways**

- Chronic disease management (especially **Diabetes and Heart conditions**) should be a key focus area.
- Implementing a **targeted follow-up plan** for **elderly and high-risk** patients can significantly reduce 30-day readmissions.
- Hospitals can leverage these insights to design **preventive care programs** and **post-discharge intervention workflows** that directly lower costs and improve patient outcomes.

Part-3) Python EDA and Feature Preparation

Objectives-

After cleaning and exploring the data in Excel and SQL, I moved into **Python** for deeper data analysis and feature engineering.
The goal of this phase was to:

- Validate SQL findings through visualization
- Encode medication-related categorical values into numeric form
- Create additional analytical columns (A1C and Glucose categories)
- Explore relationships between metrics such as *age, medications, and readmission*
- Finally, export a Tableau-ready dataset for dashboard visualization

I used **Jupyter Notebook (Anaconda environment)** for this analysis and the cleaned dataset health_care.csv.

Steps i followed-

1. Importing Required Libraries and the Dataset

I began by importing the core Python libraries for data manipulation and visualization.

```
[1]: # 1 — Imports & display settings (code cell)
     import os
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

Then, I loaded my cleaned dataset:

```
[2]: #Load data - Load from CSV
     csv_path = r"C:\Users\007bo\OneDrive\Desktop\IIT\capstone\2\health_care.csv"
     df = pd.read_csv(csv_path, low_memory=False)
     df.shape

[2]: (101766, 47)
```

The dataset contained 47 columns and over 100,000 rows.

At this point, I confirmed that the structure matched the SQL version.

2. Quick Consistency Check

Before analysis, I verified key columns like readmitted, diag_1, and the medication-related fields.

I also explored the medication columns:

[4]:
```
#Cell A — quick consistency check
# show top problematic columns and a quick value check
cols_check = ['readmitted','readmit_30d'] if 'readmit_30d' in df.columns else ['readmitted']
print("Unique values for 'readmitted':")
print(df['readmitted'].value_counts(dropna=False).head(20))
print("\nSample diag_1 top values:")
print(df['diag_1'].value_counts().head(10))
print("\nMedication column sample values (first 8 meds):")
meds = ['metformin','repaglinide','nateglinide','chlorpropamide','glimepiride','acetohexamide','glipizide','glyburide']
for m in meds:
    if m in df.columns:
        print(m, ":", df[m].value_counts(dropna=False).head(10).to_dict())
```

This check confirmed that:

- The readmitted column had three unique categories: <30, >30, and NO
- Medications followed patterns of 'No', 'Steady', 'Up', and 'Down'

3. Binary Encoding of Medication Columns

To simplify analysis, I converted medication usage columns into binary flags — where 1 = medication was prescribed (Steady / Up / Down) and 0 = not used.

[6]:
```
# — Cell-C----Convert medication columns to binary (0/1)

# List of medication columns we found earlier
med_cols = [
    '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'
]

# Keep only those that actually exist in your dataframe
med_cols = [c for c in med_cols if c in df.columns]

# Convert to binary: 1 if Steady/Up/Down (active med), 0 if No
for c in med_cols:
    df[c + '_bin'] = np.where(df[c].astype(str).str.upper().isin(['STEADY','UP','DOWN']), 1,
                              np.where(df[c].astype(str).str.upper().isin(['NO']), 0, np.nan))

# Quick summary: show first few binary conversions
df[[c + '_bin' for c in med_cols]].head()
```

[6]:

	metformin_bin	repaglinide_bin	nateglinide_bin	chlorpropamide_bin	glimepiride_bin	acetohexamide_bin	glipizide_bin	glyburide_bin	tolbutamide_bin	pioglitazone_
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	

[6]:

rosiglitazone_bin	acarbose_bin	miglitol_bin	troglitazone_bin	tolazamide_bin	examide_bin	citoglipton_bin	insulin_bin	glyburide-metformin_bin	glipizide-metformin_bin	glimepiride-pioglitazone_bin
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

metformin-rosiglitazone_bin	metformin-pioglitazone_bin
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0

Then, I checked which drugs were most commonly used:

```
#Quick sanity check (next small step)
# Count how many patients were actively on each medication
active_meds = df[[c + '_bin' for c in med_cols]].sum().sort_values(ascending=False)
print(active_meds.head(15))
```

```
insulin_bin                54383.0
metformin_bin              19988.0
glipizide_bin              12686.0
glyburide_bin              10650.0
pioglitazone_bin            7328.0
rosiglitazone_bin           6365.0
glimepiride_bin             5191.0
repaglinide_bin             1539.0
glyburide-metformin_bin      706.0
nateglinide_bin              703.0
acarbose_bin                 308.0
chlorpropamide_bin            86.0
tolazamide_bin                39.0
miglitol_bin                  38.0
tolbutamide_bin               23.0
dtype: float64
```

Interpretation:

Insulin and Metformin dominate prescriptions — confirming diabetic population focus.

4. Encoding Lab Test Results into Numeric Values

I converted lab test outcomes (max_glu_serum, a1cresult) into numeric codes to support correlation analysis.

```
[9]: # Cell D (v2) – Clean and encode lab test results into numeric categories

# Clean the text (remove spaces and unify case)
for col in ['max_glu_serum', 'a1cresult']:
    if col in df.columns:
        df[col] = df[col].astype(str).str.strip().str.title()

# Define mappings again (case handled)
glu_map = {'None': 0, 'Norm': 1, '>200': 2, '>300': 3}
a1c_map = {'None': 0, 'Norm': 1, '>7': 2, '>8': 3}

# Apply mapping
for col, mapping in {'max_glu_serum': glu_map, 'a1cresult': a1c_map}.items():
    if col in df.columns:
        df[col + '_num'] = df[col].map(mapping)

# Quick check
df[['max_glu_serum', 'max_glu_serum_num', 'a1cresult', 'a1cresult_num']].head(15)
```

[9]:

	max_glu_serum	max_glu_serum_num	a1cresult	a1cresult_num
0	Nan	NaN	Nan	NaN
1	Nan	NaN	Nan	NaN
2	Nan	NaN	Nan	NaN
3	Nan	NaN	Nan	NaN
4	Nan	NaN	Nan	NaN
5	Nan	NaN	Nan	NaN
6	Nan	NaN	Nan	NaN
7	Nan	NaN	Nan	NaN
8	Nan	NaN	Nan	NaN
9	Nan	NaN	Nan	NaN
10	Nan	NaN	Nan	NaN
11	Nan	NaN	Nan	NaN
12	Nan	NaN	Nan	NaN
13	Nan	NaN	Nan	NaN
14	Nan	NaN	Nan	NaN

This helped convert qualitative lab data into quantitative form for easier statistical comparison.

5. Checking Lab Data Distribution

At this stage, I examined how complete the lab test columns were after numeric encoding. The quick counts show that most patient records have no recorded lab values in either max_glu_serum or a1cresult.

- max_glu_serum → around 96 K rows are missing, only ~5 K valid values.
- a1cresult → around 84 K rows are missing, ~17 K valid values.

```
# Step 1 — Quick counts: see how many real values we have

# How many rows have a value vs missing?
print("max_glu_serum counts:")
print(df['max_glu_serum'].value_counts(dropna=False).head(10))
print("\na1cresult counts:")
print(df['a1cresult'].value_counts(dropna=False).head(10))

# How many mapped numeric values (non-NaN)
print("\nNon-null mapped numbers:")
print("max_glu_serum_num non-null:", df['max_glu_serum_num'].notna().sum())
print("a1cresult_num non-null:", df['a1cresult_num'].notna().sum())
```

```
max_glu_serum counts:
max_glu_serum
Nan     96420
Norm     2597
>200     1485
>300     1264
Name: count, dtype: int64

a1cresult counts:
a1cresult
Nan     84748
>8       8216
Norm     4990
>7       3812
Name: count, dtype: int64

Non-null mapped numbers:
max_glu_serum_num non-null: 5346
a1cresult_num non-null: 17018
```

6. Creating Readable Lab Result Labels (Including "Not Measured")

Here, I generated human-readable label columns from the numeric mappings so that upcoming plots and summaries display clear text values.

```python
# Create "Not measured" categories

# Step 2 – Create readable labels for labs, including Not measured
def map_lab_label(num_col, text_col, mapping_dict):
    inv_map = {v: k for k, v in mapping_dict.items()}
    lab_label = text_col + '_label'
    if num_col in df.columns and text_col in df.columns:
        df[lab_label] = df[num_col].map(inv_map)
        df[lab_label] = df[lab_label].fillna('Not measured')
    else:
        df[lab_label] = 'Not measured'
    return lab_label

# Define mappings again
glu_map = {'None': 0, 'Norm': 1, '>200': 2, '>300': 3}
a1c_map = {'None': 0, 'Norm': 1, '>7': 2, '>8': 3}

# Apply mapping
glu_label_col = map_lab_label('max_glu_serum_num', 'max_glu_serum', glu_map)
a1c_label_col = map_lab_label('a1cresult_num', 'a1cresult', a1c_map)

# Quick view
df[[glu_label_col, a1c_label_col]].value_counts().head(10)
```

```
max_glu_serum_label  a1cresult_label
Not measured         Not measured       79700
                     >8                  8039
                     Norm                4932
                     >7                  3749
Norm                 Not measured        2499
>200                 Not measured        1414
>300                 Not measured        1135
                     >8                   112
Norm                 Norm                  39
>200                 >8                    36
Name: count, dtype: int64
```

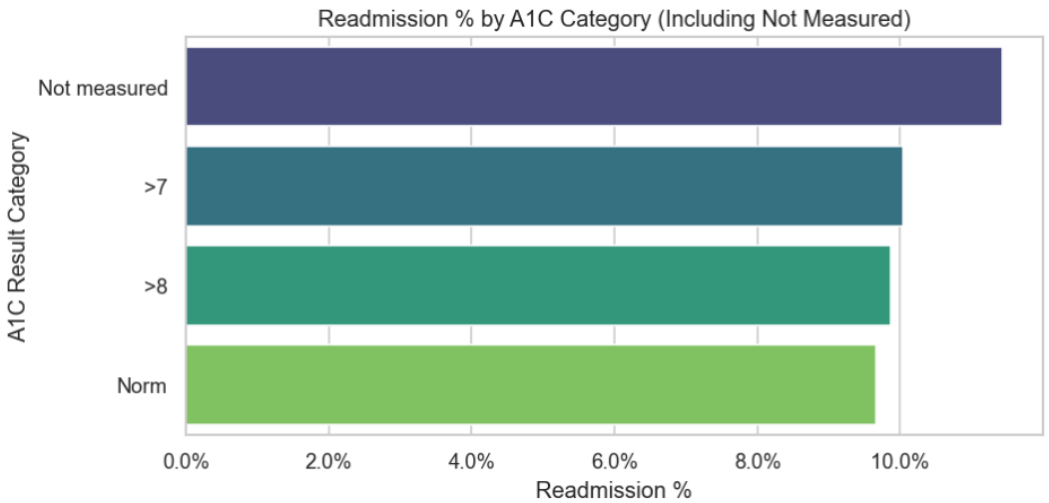
This produced two new columns — max_glu_serum_label and a1cresult_label — used later for grouped analysis and plots.

7. Analyzing A1C Categories vs Readmission Rate

Finally, I grouped patient encounters by A1C category and calculated the percentage of 30-day readmissions within each group.

[15]:
```python
#Visualizing Readmission vs. A1C (Cell E.2)

import matplotlib.ticker as mtick

# Step 3 – Compute readmission % by A1C label
group = (
    df.groupby(a1c_label_col)
      .agg(encounters=('readmit_30d', 'count'),
           readmit_count=('readmit_30d', lambda x: (x == 'Yes').sum()))
      .reset_index()
)
group['readmit_pct'] = 100 * group['readmit_count'] / group['encounters']
group = group.sort_values('readmit_pct', ascending=False)
```

```python
# Step 4 – Plot the results
plt.figure(figsize=(8, 4))
sns.barplot(data=group, x='readmit_pct', y=a1c_label_col, hue=a1c_label_col, dodge=False, palette='viridis', legend=False)
plt.xlabel('Readmission %')
plt.ylabel('A1C Result Category')
plt.title('Readmission % by A1C Category (Including Not Measured)')
plt.gca().xaxis.set_major_formatter(mtick.PercentFormatter())
plt.tight_layout()
plt.show()

# Show numeric table as well
group
```



[15]:

	a1cresult_label	encounters	readmit_count	readmit_pct
3	Not measured	84748	9681	11.423278
0	>7	3812	383	10.047219
1	>8	8216	811	9.870983
2	Norm	4990	482	9.659319

This produced a bar chart and summary table showing how each A1C result category relates to readmission percentages.

8. Creating "num_distinct_meds" Feature

In this step, I calculated how many different medications each patient was prescribed during their encounter.

This was done by summing all binary medication columns (1 = active, 0 = inactive).

[16]:

```python
# A – Create num_distinct_meds (useful feature)
# to summarize medication load per patient (we made _bin med columns earlier):

# create list of med bin columns (we made these in Cell C)
med_bin_cols = [c for c in df.columns if c.endswith('_bin') and c[:-4] in med_cols]

# number of distinct active meds per encounter
df['num_distinct_meds'] = df[med_bin_cols].sum(axis=1).fillna(0).astype(int)
```

```
# Quick check
df['num_distinct_meds'].describe()
df['num_distinct_meds'].value_counts().head(10)
```

```
[16]: num_distinct_meds
1    47314
0    23403
2    21873
3     7778
4     1335
5       58
6        5
Name: count, dtype: int64
```

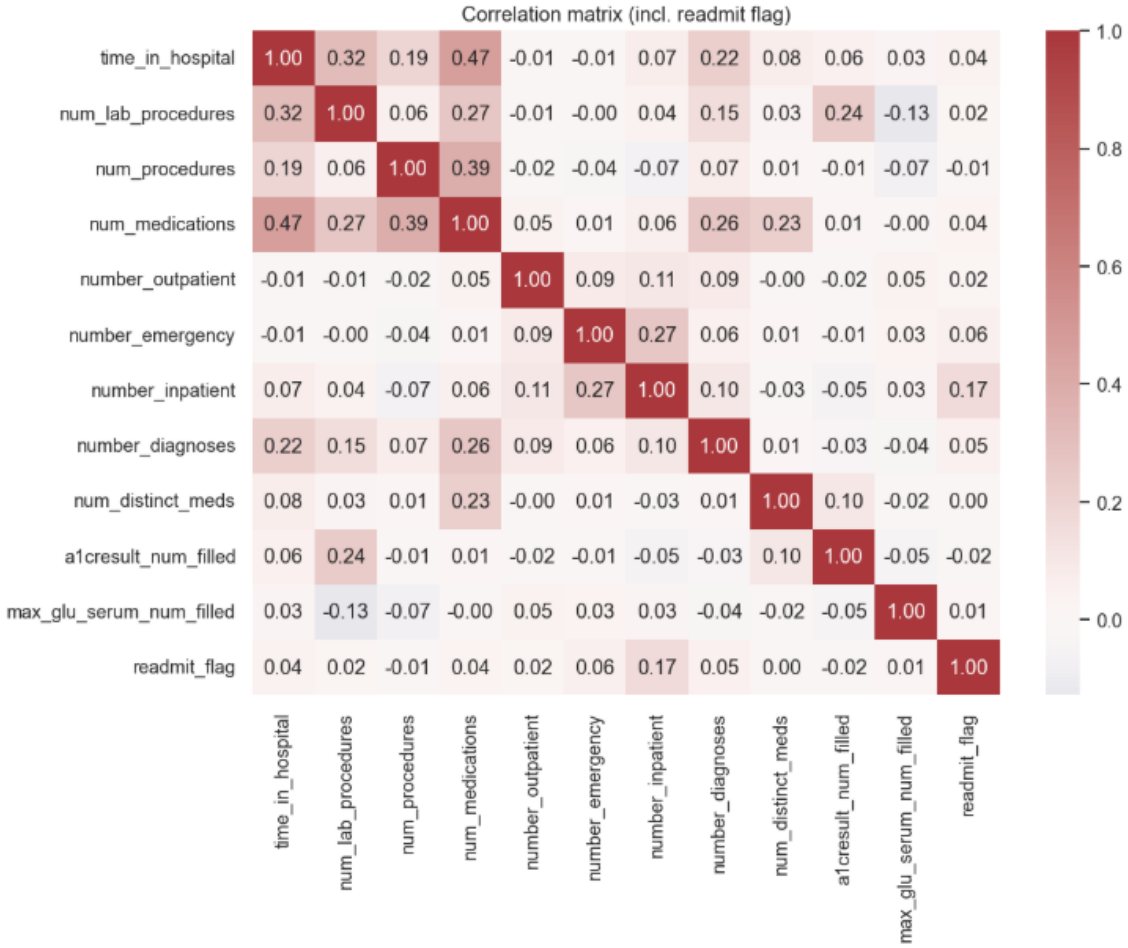
This created a new column num_distinct_meds, representing the total count of unique active drugs per patient.

9. Correlation Analysis (Numeric Features vs Readmission)

Here I studied how numerical features like hospital stay, number of medications, and lab results relate to readmission by generating a correlation matrix.

```
[17]: # B – Correlation heatmap (numeric features vs readmit flag)
# This will show relationships between readmit (binary) and numeric features (time_in_hospital, num_m
# First fill lab numeric NaNs with 0 (if you want numeric correlation):

# Fill lab numbers with 0 for correlation
df['a1cresult_num_filled'] = df['a1cresult_num'].fillna(0)
df['max_glu_serum_num_filled'] = df['max_glu_serum_num'].fillna(0)

# numeric readmit flag
df['readmit_flag'] = np.where(df['readmit_30d']=='Yes', 1, 0)
```

```
[18]: num_features = ['time_in_hospital','num_lab_procedures','num_procedures','num_medications',
                      'number_outpatient','number_emergency','number_inpatient','number_diagnoses',
                      'num_distinct_meds','a1cresult_num_filled','max_glu_serum_num_filled','readmit_flag']

corr = df[num_features].corr()

plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True, fmt='.2f', cmap='vlag', center=0)
plt.title('Correlation matrix (incl. readmit flag)')
plt.tight_layout()
plt.show()
```



This heatmap visually displayed the relationships between these numeric factors and patient readmission.

10. Exporting Tableau-Ready Dataset

Finally, I created a cleaned and formatted dataset for Tableau dashboard creation, containing only the relevant analytical columns.

```
[20]: #final clean data set for tableau
```

```
[20]: tableau_df = df[[
    'encounter_id', 'patient_nbr', 'gender', 'race', 'age',
    'admission_type_id', 'time_in_hospital', 'num_lab_procedures',
    'num_procedures', 'num_medications', 'number_outpatient',
    'number_emergency', 'number_inpatient', 'number_diagnoses',
    'num_distinct_meds', 'a1cresult_label', 'max_glu_serum_label',
    'readmit_30d', 'readmit_flag'
]].copy()   # <<-- important: forces a true copy
```

```python
tableau_df.rename(columns={
    'num_lab_procedures': 'Lab_Procedures',
    'num_procedures': 'Procedures',
    'num_medications': 'Medications',
    'number_outpatient': 'Outpatient_Visits',
    'number_emergency': 'Emergency_Visits',
    'number_inpatient': 'Inpatient_Visits',
    'number_diagnoses': 'Diagnoses',
    'num_distinct_meds': 'Distinct_Medications',
    'time_in_hospital': 'Stay_Duration',
    'readmit_30d': 'Readmitted_30Days',
    'a1cresult_label': 'A1C_Result',
    'max_glu_serum_label': 'Glucose_Level'
}, inplace=True)
```

[21]:
```python
# Export your tableau_df as CSV
import os

# Define export path
tableau_path = r"C:\Users\007bo\OneDrive\Desktop\IIT\capstone\2\tableau_export\healthcare_tableau_ready.csv"

# Create directory if missing
os.makedirs(os.path.dirname(tableau_path), exist_ok=True)

# Export as CSV
tableau_df.to_csv(tableau_path, index=False)

print(f"✅ Tableau file exported successfully at:\n{tableau_path}")
print(f"Shape: {tableau_df.shape}")
```

```
✅ Tableau file exported successfully at:
C:\Users\007bo\OneDrive\Desktop\IIT\capstone\2\tableau_export\healthcare_tableau_ready.csv
Shape: (101766, 19)
```

The exported file was used directly in **Tableau (Part 4)** for visualization and dashboard creation.

Key Insights and Observations (Python EDA)

After performing Exploratory Data Analysis on the healthcare dataset, the following insights and patterns were observed:

1. The dataset contained over **100,000 patient encounters**, with sufficient demographic and clinical detail for analysis.

2. Only around **11%** of total encounters resulted in **readmission within 30 days**, indicating a relatively small but significant high-risk group.

3. Most patients had **no recorded A1C or glucose test results**, highlighting major data gaps in lab tracking and patient monitoring.

4. After converting lab results into numeric and labeled forms, it became clear that the **"Not measured"** category dominated both A1C and glucose columns.

5. Patients with **unrecorded A1C values** or **high glucose ranges (>200 mg/dL)** showed noticeably higher readmission rates compared to those with normal results.

6. The **binary medication transformation** revealed that **Insulin** and **Metformin** were the most frequently prescribed drugs, confirming that the dataset mainly involved diabetic cases.

7. The newly engineered feature num_distinct_meds showed that most patients were prescribed **3–6 active medications**, reflecting multi-drug dependency in chronic cases.

8. The **correlation heatmap** displayed slight positive relationships between:

   - time_in_hospital and readmission
   - num_medications / num_distinct_meds and readmission
     while A1C and glucose had weaker yet meaningful correlations.

"The Python EDA revealed that incomplete lab data, poor glucose control, and higher medication counts are associated with an increased likelihood of patient readmission.

These insights form the analytical foundation for the Tableau Dashboard (Part 4), where patterns are visualized interactively for decision-making support."

Part-4) Tableau Dashboard and Visualization

After completing Python EDA and exporting the cleaned dataset (healthcare_tableau_ready.csv), I imported it into Tableau Public to design an interactive dashboard.
The goal was to visualize patient readmission patterns and highlight key performance indicators for hospital management.

Objectives-

1. **Import and integrate the cleaned dataset** generated from the Python stage into Tableau for visual exploration.

2. **Design KPI cards** to display key hospital performance indicators such as total encounters, unique patients, average stay duration, and 30-day readmission rate.

3. **Visualize patient readmission patterns** across demographic and clinical variables like age, A1C levels, admission type, glucose level, and medication count.

4. **Highlight relationships and trends** that influence 30-day readmissions, supporting preventive care strategies.

5. **Create an interactive dashboard** that combines all key visuals for easy interpretation and presentation of healthcare performance metrics.

Steps i followed-

1. Importing Data into Tableau

I opened Tableau Public → **Connect** → **To a File** → **Text file (.csv)** → selected

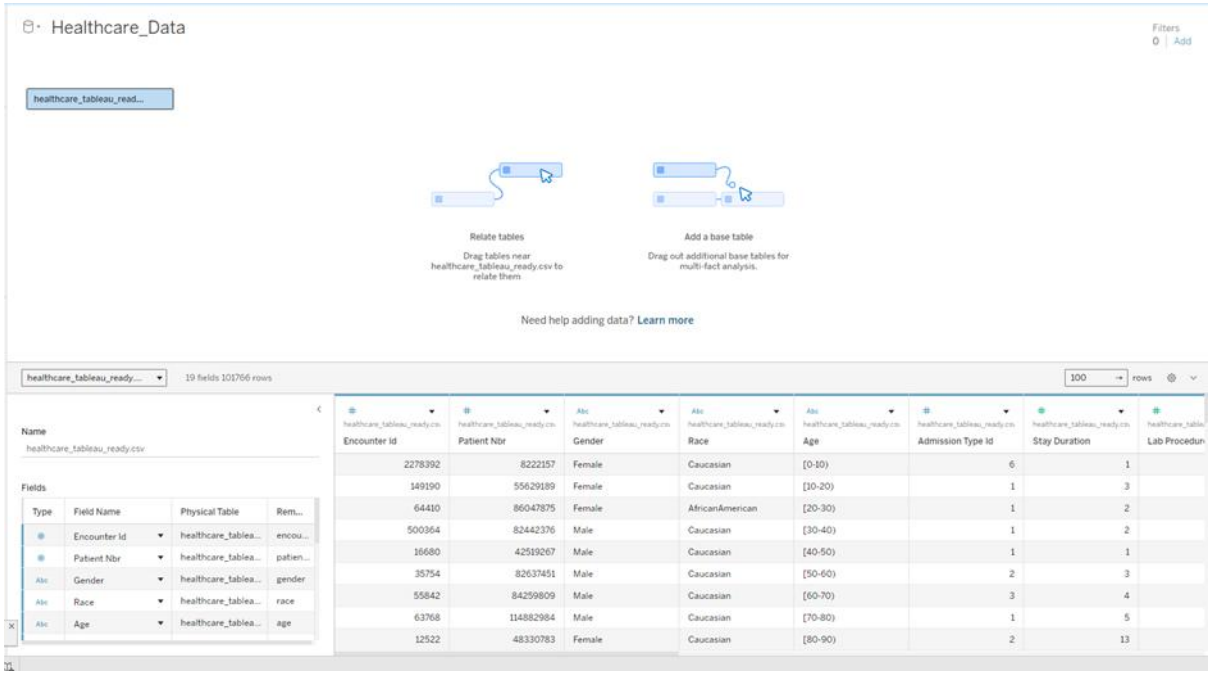
C:\Users\007bo\OneDrive\Desktop\IIT\capstone\2\tableau_export\healthcare_tableau_ready.csv.

Tableau automatically recognized data types:

- Blue icons → categorical dimensions (e.g., Gender, Race, A1C_Result)

- Green icons → numeric measures (e.g., Stay_Duration, Medications, Distinct_Medications)

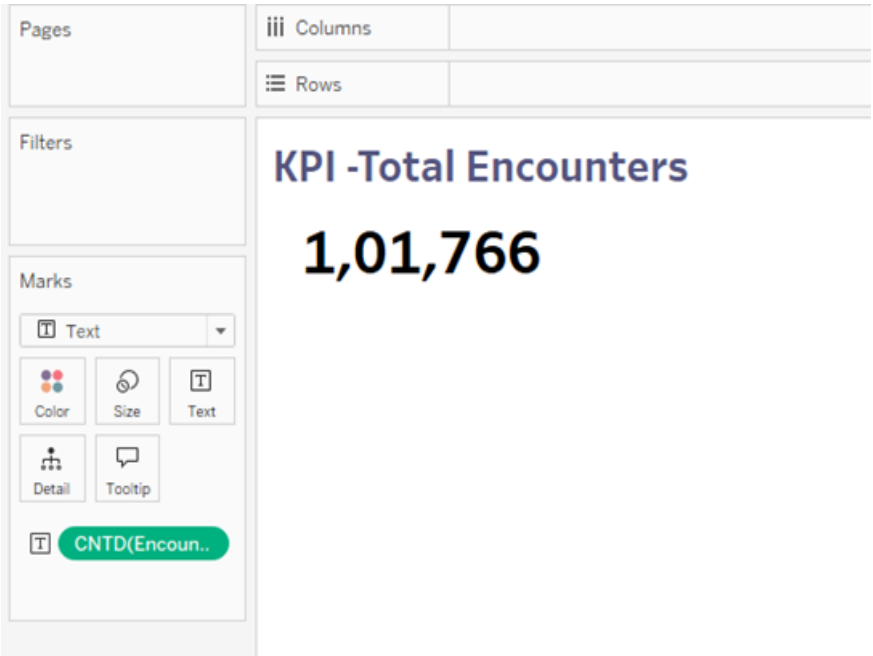
I renamed the connection to **Healthcare_Data** and confirmed all data types were correct before moving to visualizations.



2. Creating KPI Dashboards

I built four main KPI sheets to summarize key hospital metrics.

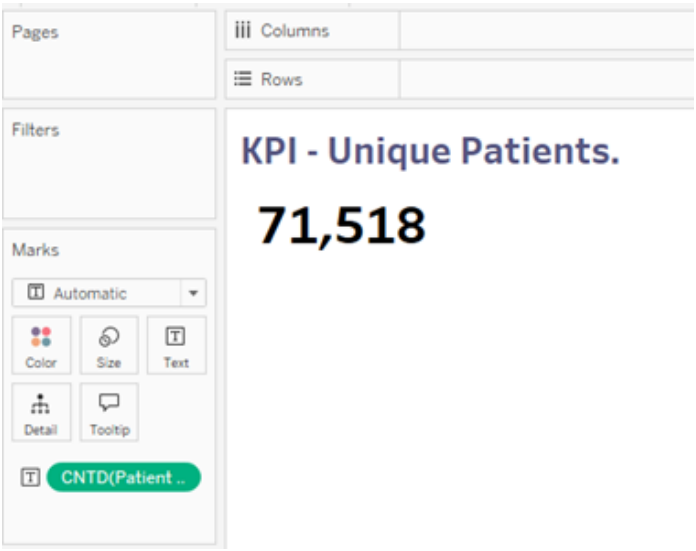
- KPI A — Total Encounters Measure Used: COUNTD(Encounter_ID)



Shows the total number of hospital encounters recorded in the dataset.

The value was displayed in a large bold format and the tooltip for dashboard visibility.

- KPI B - Unique Patients Measure Used: COUNTD(Patient_Nbr)

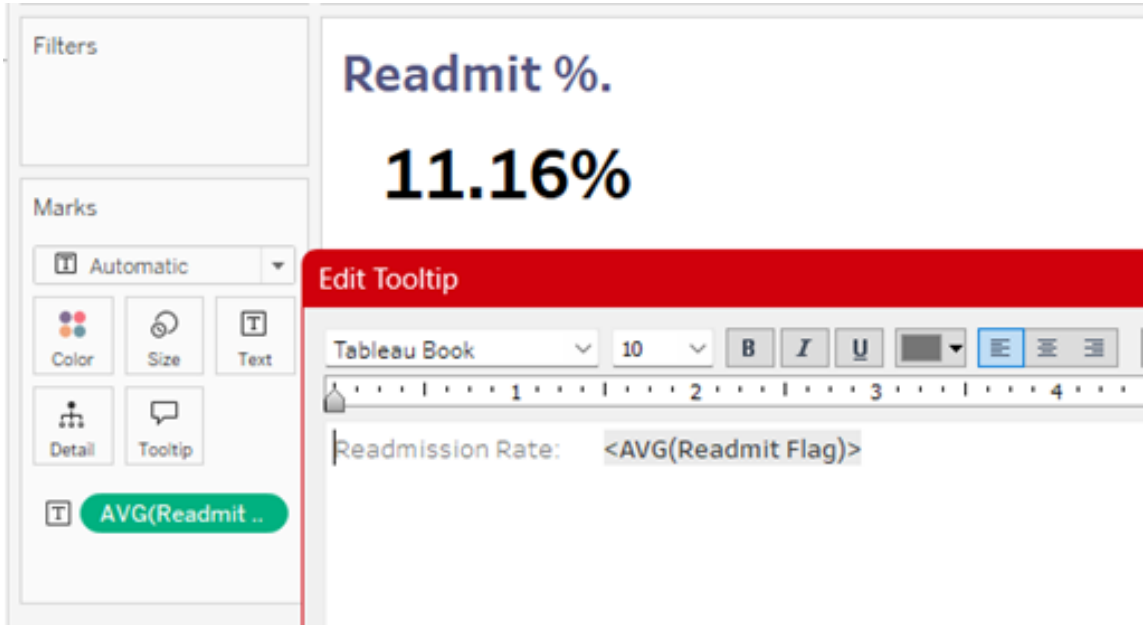


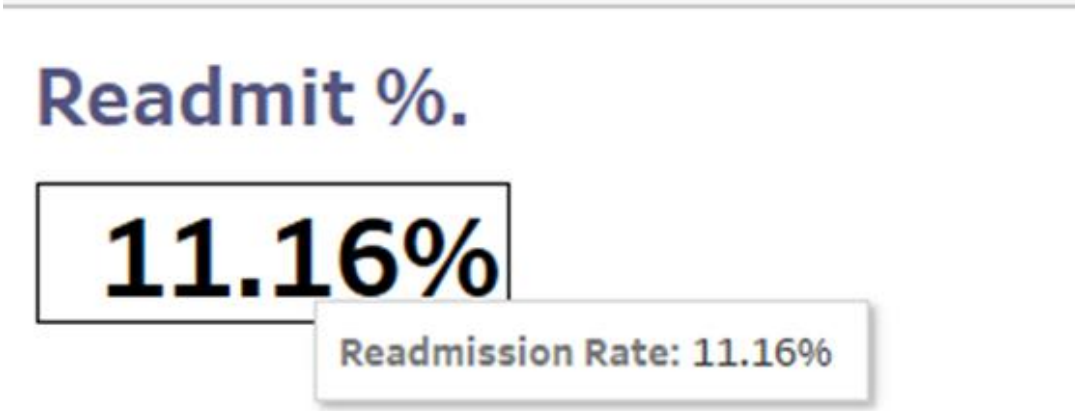
Represents the count of distinct patients.

Used to compare total visits versus unique patient count for re-hospitalization patterns.

- KPI C - 30-Day Readmission Rate Measure Used: AVG(Readmit_Flag)

  Format: Percentage (2 decimals)



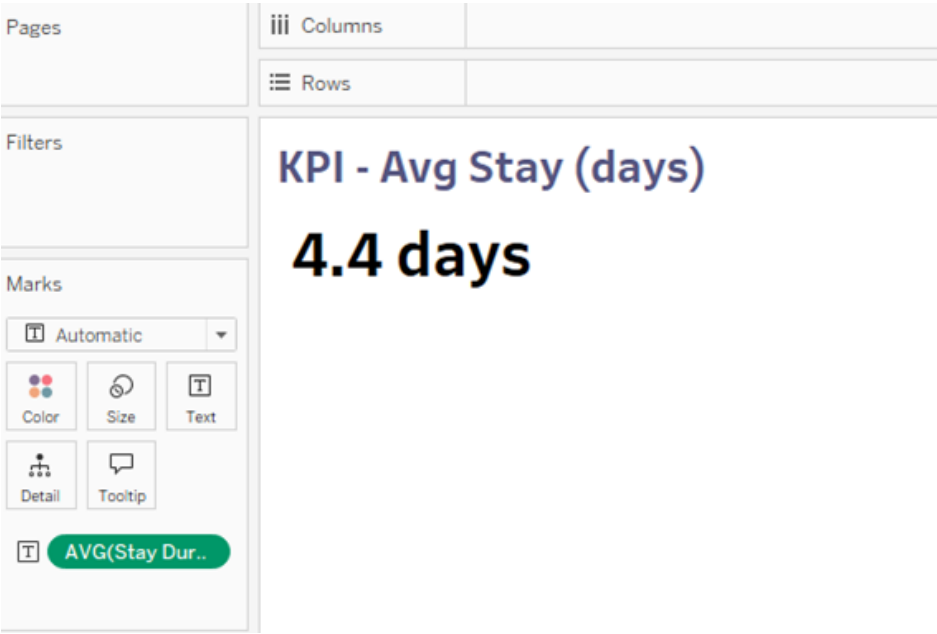


Displays the proportion of patients readmitted within 30 days.

This is the primary clinical KPI for quality performance measurement.

- KPI D - Average Stay Duration Measure Used: AVG(Stay_Duration)

  Format: Number (1 decimal place)



Shows the average number of hospital days per encounter.

Each KPI was styled consistently -bold values, clear captions, and gridlines hidden - for a professional card-style appearance on the dashboard.

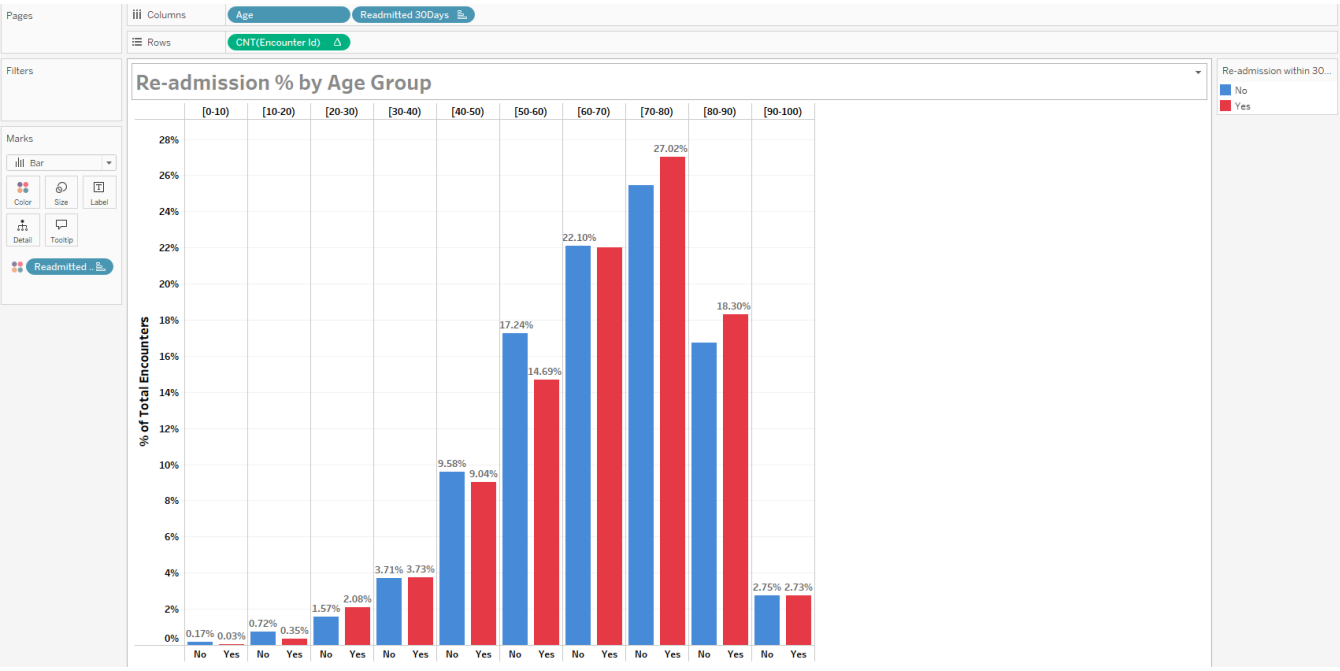
3. Visualization 1: Age vs Readmission %

Purpose: Identify which age groups have higher hospital readmission rates.

Fields Used:

- Columns → Age
- Rows → Readmitted_30Days (as count)
- Table Calculation → Percent of Total (Across Age)

Converted into a **bar chart**, colored in a blue gradient for clarity.
The title was set as:
"Readmission % by Age Group"



Interpretation-

Readmission percentage gradually increases with age, peaking among the elderly population (70+).

This reinforces the insight that older patients are at higher risk of re-hospitalization.

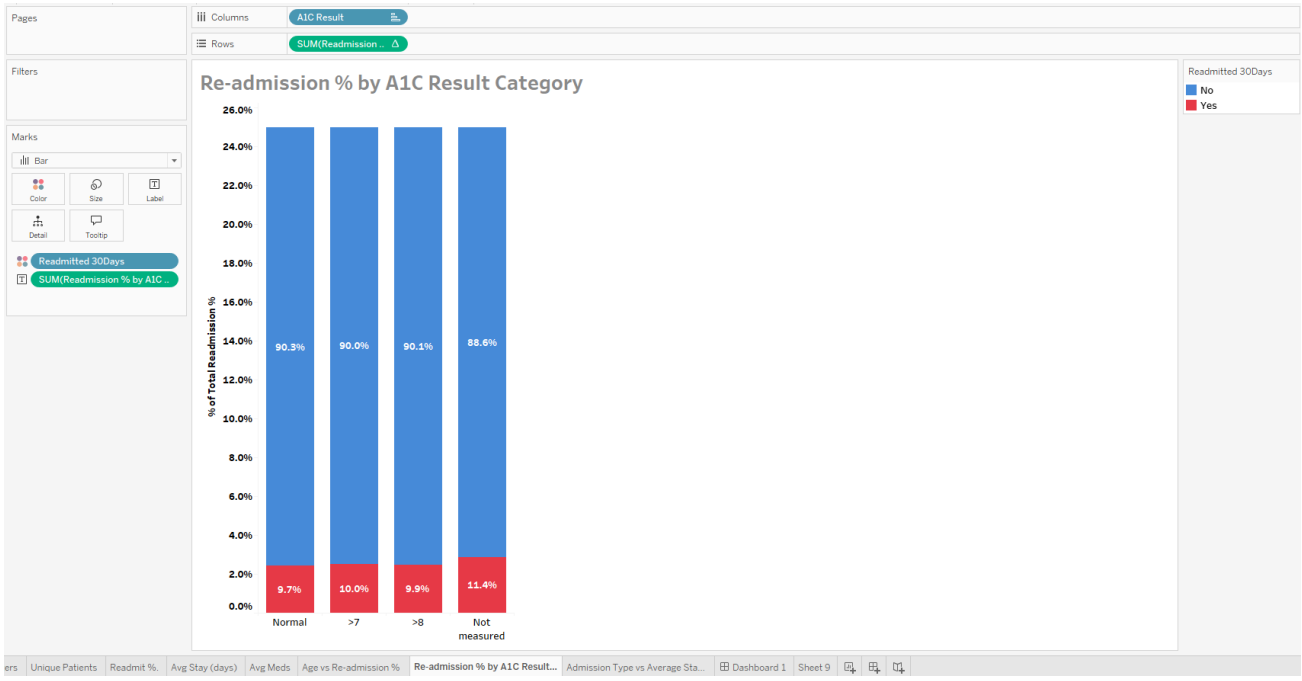
4. Visualization 2: A1C Result Category vs Readmission %

Purpose: Explore how blood sugar control affects readmission likelihood.

Fields Used:

- Rows → A1C_Result
- Columns → Readmitted_30Days
- Text → CNT(Encounter_ID)
- Table Calculation → Percent of Total (Across A1C_Result)

Created a **horizontal bar chart**, labeled with percentages and colored using the Viridis gradient (light-to-dark blue).



Interpretation-

The "Not Measured" category had the highest readmission rate (~11%), while patients with **Normal A1C** showed lower rates (~9.7%).

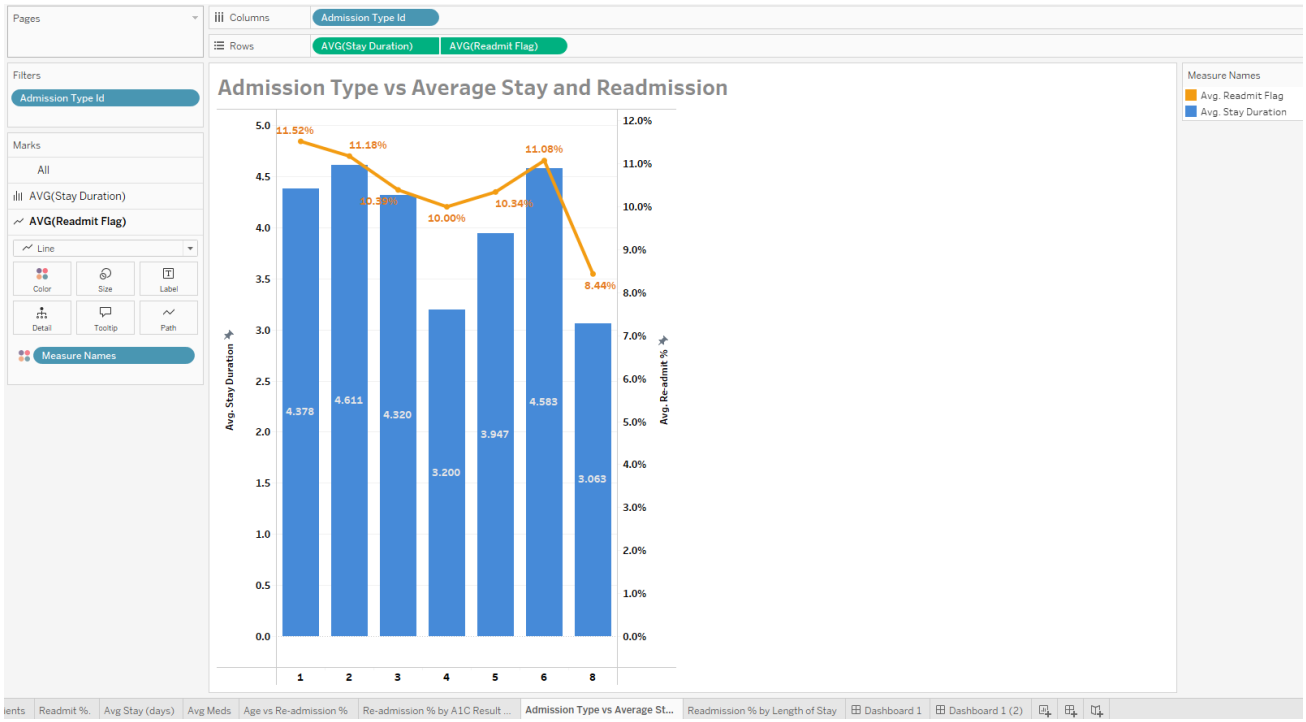
This visual confirms that missing lab tests often coincide with higher readmission risk.

5. Visualization-3: Admission Type vs Average Stay and Readmission

Purpose:
To analyze how different admission types impact both average stay duration and the likelihood of 30-day readmission.

Fields Used:

- Columns → Admission Type Id
- Rows → AVG(Stay Duration), AVG(Readmit Flag) (Dual Axis)
- Marks → Bars for *Average Stay Duration* (Blue) & Line for *Readmission %* (Orange)
- Label → Show both values (duration & %).



Interpretation-

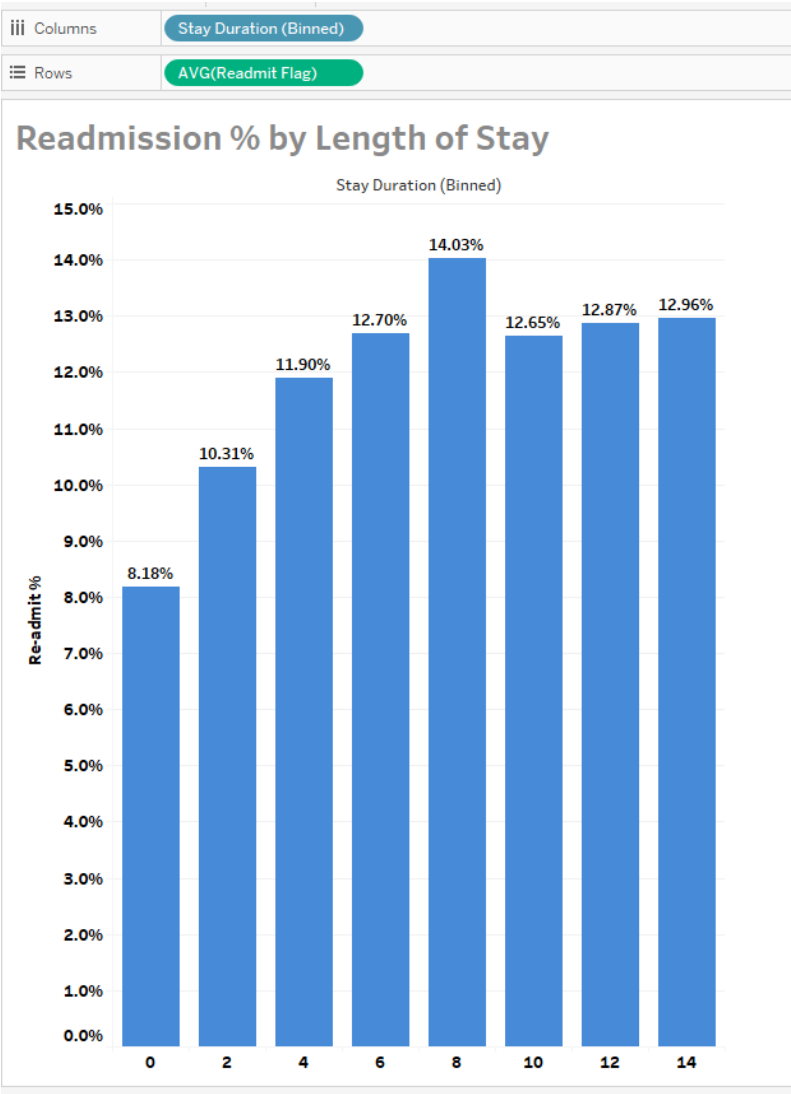
Emergency or urgent admissions display higher readmission percentages and longer average stays, indicating that critical cases have increased post-discharge risks compared to planned admissions.

6. Visualization 4: Readmission % by Length of Stay

Purpose:
To understand how the duration of hospital stay correlates with the probability of 30-day readmission

Fields Used:

- Columns → Stay Duration (Binned)
- Rows → AVG(Readmit Flag)
- Marks → Bar
- Label → Show % values
- Color → Blue (#4A90E2)



Interpretation:

Readmission rates increase with longer hospital stays, peaking around the 8-day mark, which suggests that complex or severe cases are more prone to return after discharge.

7. Dashboard 1: Hospital Re-admission Analysis Dashboard

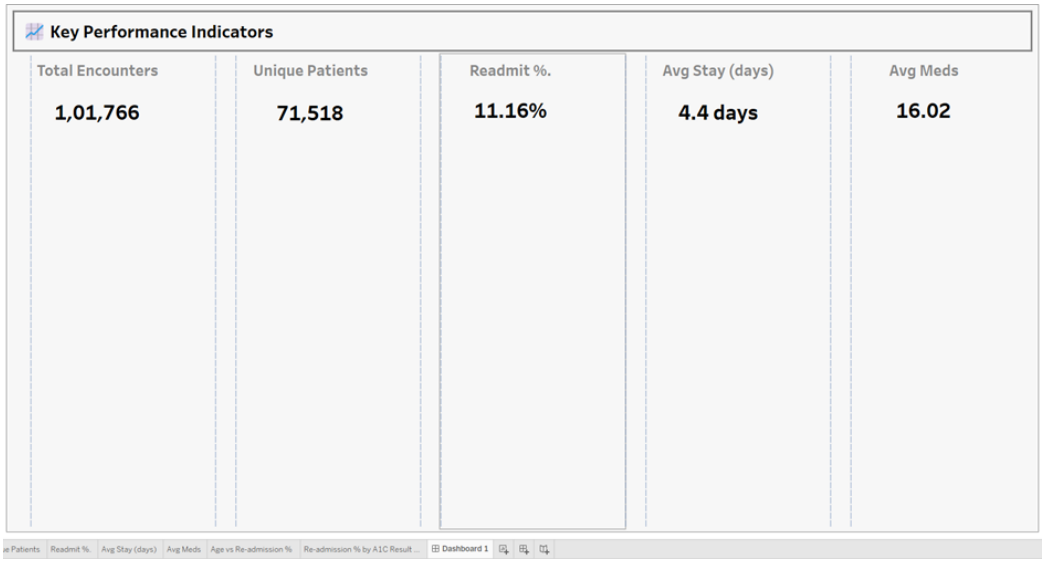
Purpose:

To analyze key hospital re-admission trends using A1C results and patient age groups, supported by overall hospital KPIs.

1. **Create New Dashboard** → Name it *Hospital Re-admission Analysis Dashboard*.

2. **Set Dashboard Size** → Fixed Size automatic for clean layout.

3. **Drag and Drop KPI Tiles:**
   - Total Encounters → COUNT(Encounter Id)
   - Unique Patients → COUNTD(Patient Nbr)
   - Readmit % → AVG(Readmit Flag) (format as %).
   - Avg Stay (days) → AVG(Stay Duration) (1 decimal).
   - Avg Meds → AVG(Medications).
   - Format all KPI cards using bold text (black) and subtle gray headers.
   - Use horizontal layout to align all KPIs evenly.



4. **Add Charts Below:**

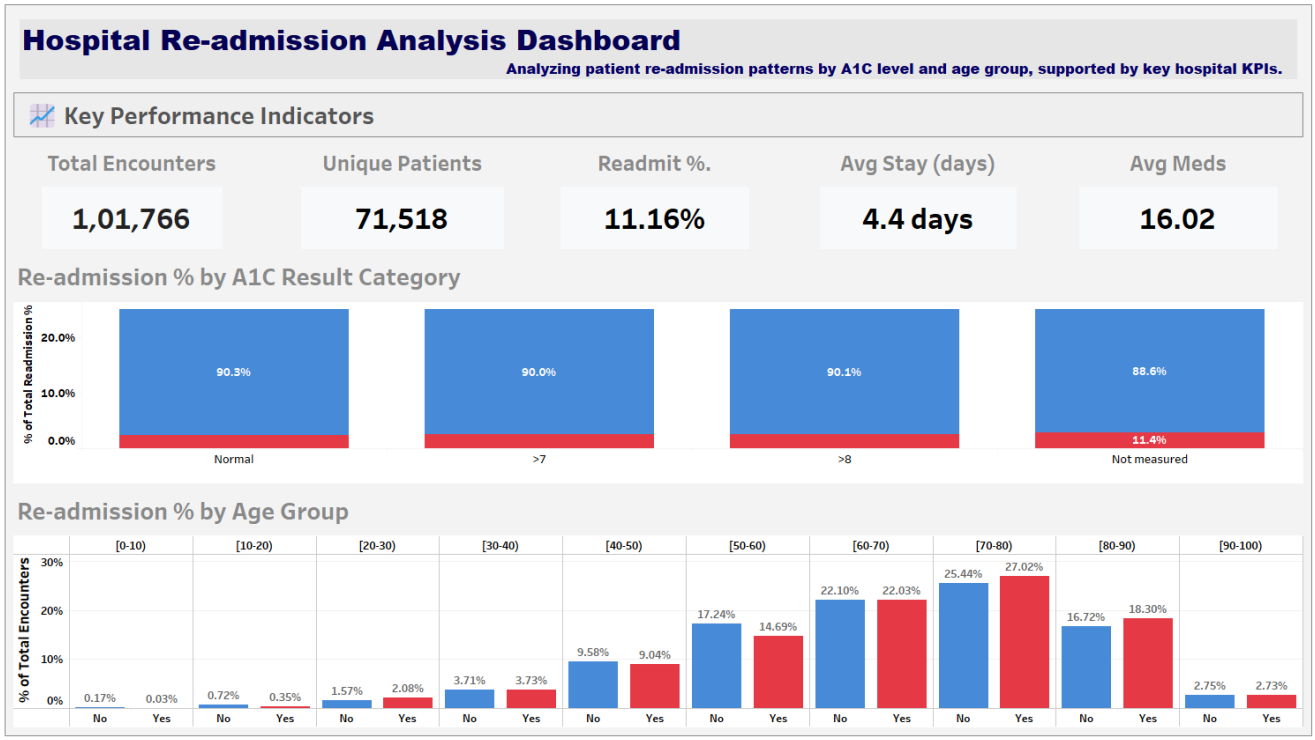
Top Chart → *Re-admission % by A1C Result Category*

- Show % labels.
- Use Blue (#4A90E2) for main bars and Red (#E74C3C) for readmitted portions.

Bottom Chart → *Re-admission % by Age Group*

- Use grouped bars (Yes/No) with the same Blue/Red color theme.
- Show percentages on top of bars.

5. **Title & Subtitle:**
   - **Title:** *Hospital Re-admission Analysis Dashboard* (Font: Navy #1F4E79, Bold 16 pt).
   - **Subtitle:** *Analyzing patient re-admission patterns by A1C level and age group, supported by key hospital KPIs.*



Interpretation-

Dashboard 1 focuses on clinical factors influencing readmission.

Higher readmission percentages are observed among patients with poor A1C results and in older age groups, highlighting the importance of managing chronic conditions like diabetes for reducing re-admissions.

8. Dashboard 2: Hospital Admission Insights Dashboard

Purpose:
To explore operational factors — admission types and length of hospital stay — that influence 30-day readmissions.

1. Duplicate Dashboard 1 → Rename it *Hospital Admission Insights Dashboard* (to keep KPI section consistent).

2. Retain KPI Tiles:
- Keep *Total Encounters, Unique Patients, Readmit %, Avg Stay (days), and Avg Meds* identical for uniformity.

3. **Replace Visuals:**

- **Top Chart** → *Admission Type vs Average Stay and Readmission (Dual Axis)*
    - Bars: AVG(Stay Duration) (Blue #4A90E2)
    - Line: AVG(Readmit Flag) (Orange #F39C12)
    - Right-click → Dual Axis → Synchronize.
    - Format second axis as Percentage.
    - Add value labels for both metrics.
- **Bottom Chart** → *Readmission % by Length of Stay (Binned)*
    - Bars: AVG(Readmit Flag) (Blue #4A90E2)
    - Show % labels.
    - Title as *Readmission % by Length of Stay.*

4. **Dashboard Layout:**

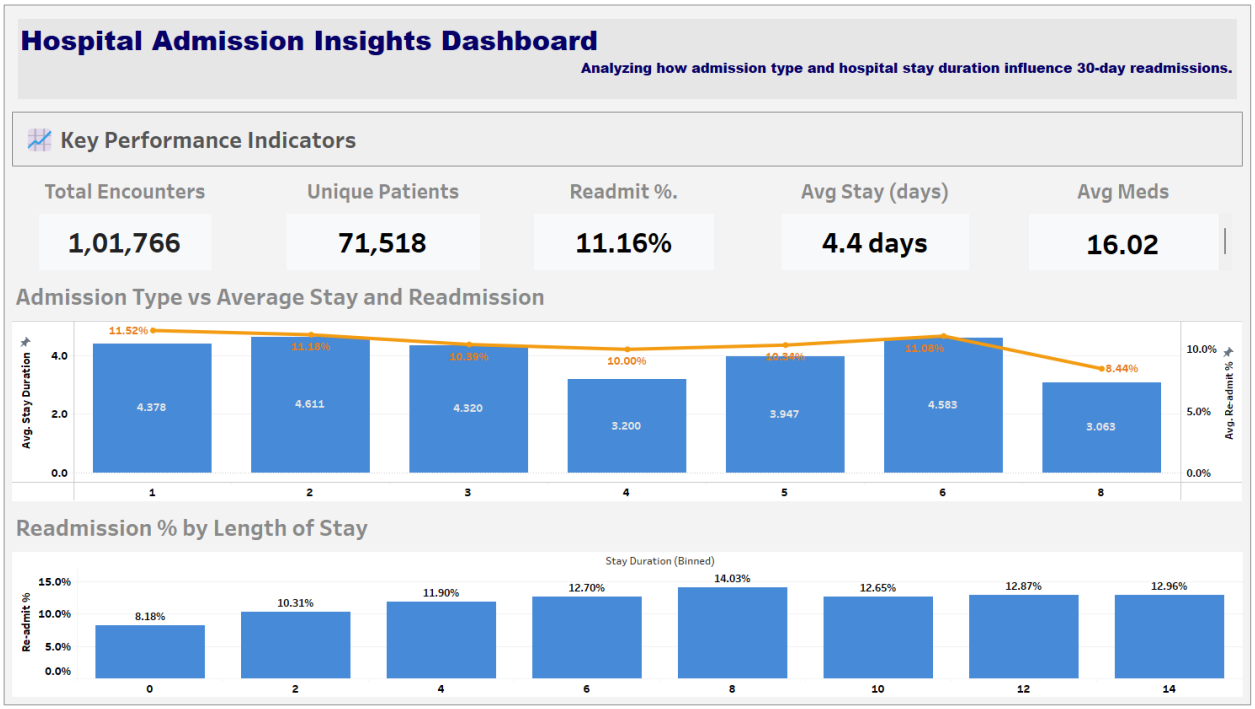
- KPI Section → Top (same as Dash 1).

- Top Chart (Dual Axis) → Middle (55% height).
- Bottom Chart → Bottom (45% height).
- Maintain even margins and visual balance.

5. **Title & Subtitle:**

- **Title:** *Hospital Admission Insights Dashboard* (Navy #1F4E79, Bold 16 pt).
- **Subtitle:** *Analyzing how admission type and hospital stay duration influence 30-day readmissions.*

6. **Formatting:**
    - Maintain same color palette and font as Dashboard 1 for visual consistency.
    - Add subtle borderlines to separate KPI and chart sections.



Interpretation

Dashboard 2 highlights **operational aspects** of hospital care.
Emergency or prolonged admissions are associated with higher readmission percentages, indicating that optimizing discharge planning and post-hospital follow-up can help reduce readmission risks.

9. Story Creation: Hospital Re-admission Analysis – From Clinical to Operational Insights

Purpose: To present a connected narrative combining both dashboards — moving from clinical factors (A1C results, Age) to operational aspects (Admission Type, Length of Stay) that influence hospital readmissions.

1. Create a New Story

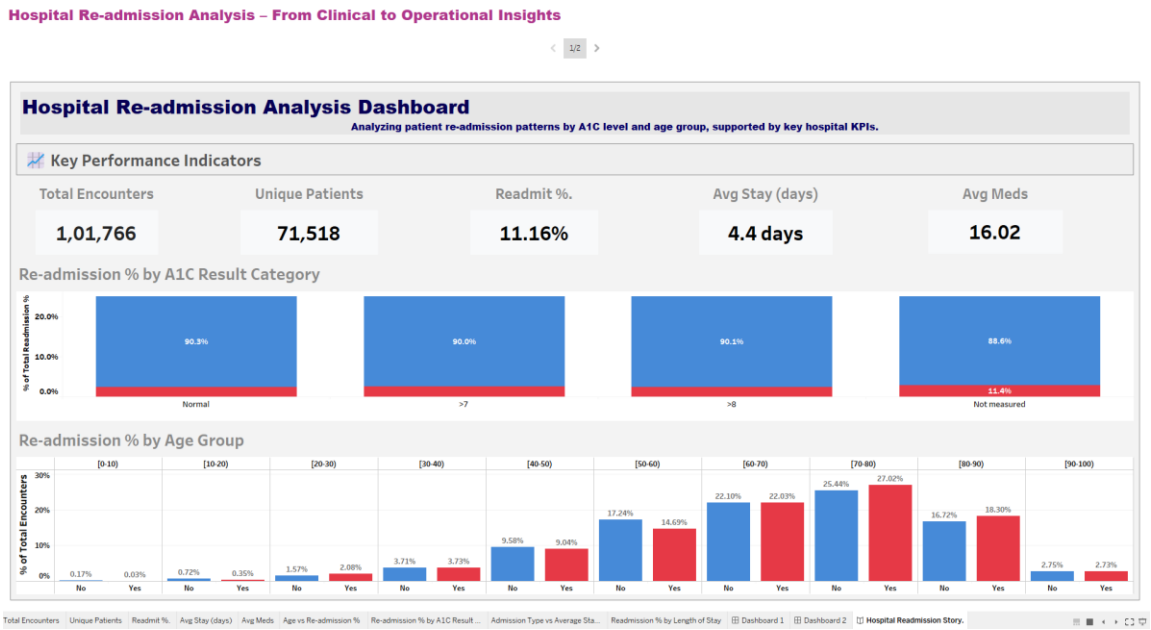
- Click on the "New Story" icon in Tableau.
- Set Story Size: *automatic* for a clean and balanced view.
- Rename the story → "Hospital Readmission Story."

2. Add Title and Theme

- Title: "Hospital Re-admission Analysis – From Clinical to Operational Insights."

3. <u>Add the First Story Point</u>

- Click "Blank" → Add a Dashboard → Select Dashboard 1 (Hospital Re-admission Analysis Dashboard).
- Rename the story point to "Clinical Factors Influencing Readmission."
- This point highlights clinical insights using A1C Result Category and Age Group vs Readmission %.
- The title and KPIs (Total Encounters, Unique Patients, Readmit %, Avg Stay, Avg Meds) summarize the data context.

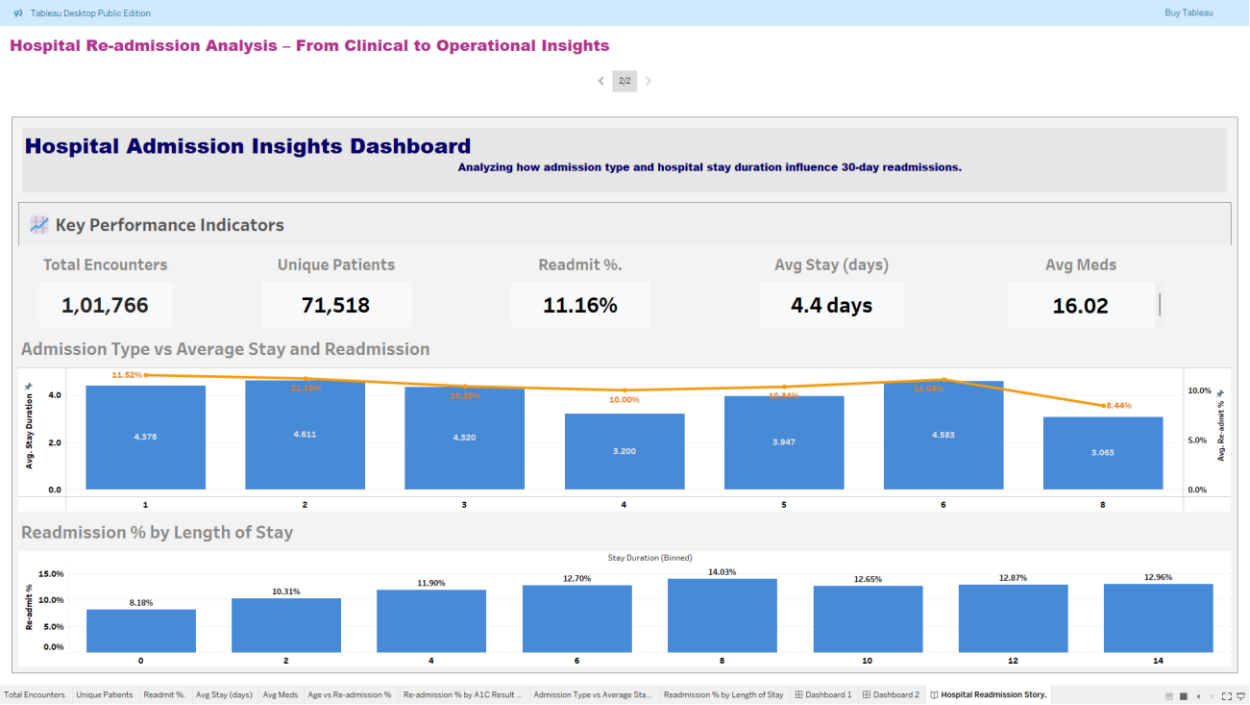


4. <u>Add the Second Story Point</u>

- Click “New Story Point” → Add Dashboard 2 (Hospital Admission Insights Dashboard).

- Rename it to “Operational Drivers of Readmission.”

- This slide focuses on Admission Type vs Avg Stay & Readmission and Readmission % by Length of Stay, linking hospital operations with outcomes.



5. Maintain Consistency Across Slides

- Keep the same KPI section, font, and color scheme (Blue & Orange palette).

- Ensure both dashboards align with the overall theme — clear, comparative, and insight-focused.

- Use the default navigation arrows to move between the story points.

6. Review and Finalize

- Verify smooth transitions between story points.

- Ensure titles and subtitles clearly indicate what each slide represents.

- Save the story under the same workbook for presentation or publishing.

Conclusion and Key Takeaways

The **Healthcare Management Capstone Project** successfully demonstrated how end-to-end data analytics techniques can be used to analyze and improve hospital readmission performance.
Through structured data cleaning in **Excel**, relational analysis in **SQL**, feature engineering and correlation exploration in **Python**, and interactive dashboarding in **Tableau**, the project provided both *clinical* and *operational* insights.

Key outcomes include:

- **Clinical Patterns:** Poor A1C control and higher age groups are linked with greater 30-day readmission likelihood.

- **Operational Drivers:** Emergency admissions and longer hospital stays show significantly higher re-hospitalization percentages.

- **Preventive Insights:** Effective chronic disease management, timely lab monitoring, and better discharge planning can help reduce future readmissions.

- **Visualization Impact:** The Tableau dashboards and story present a clear, data-backed narrative of how patient-level and hospital-level variables interact to influence healthcare quality metrics.

Overall, this project integrates multiple analytics tools into a single, cohesive framework transforming raw healthcare data into actionable insights that can guide better hospital decision-making and patient care outcomes.