

UNCOVERING HEALTHCARE PATTERNS AND PATIENT INSIGHTS THROUGH MYSQL DATA ANALYSIS

MARK ANTHONY A. BUNA
Data Analyst

PROBLEM STATEMENT

- Healthcare organizations handle large volumes of patient data, yet actionable insights are often hidden in scattered records. Without efficient analysis, trends in admissions, diagnoses, treatments, and billing go unnoticed. This limits the potential for improving operations and delivering better patient outcomes. This project uses MySQL to uncover key healthcare patterns by exploring patient demographics, hospital performance, doctor workloads, and insurance billing data.

OBJECTIVE

- To utilize MySQL for analyzing healthcare data, with the goal of identifying patient trends, optimizing hospital operations, and enhancing data-driven insights.
- Uncover insights into healthcare trends by tackling these analytical questions:
 - Basic Queries (Data Exploration)*
 - How many unique patients are recorded in the dataset?*
 - Retrieve a list of distinct hospitals where patients were admitted.*
 - Find all patients diagnosed with Diabetes, displaying their name, age, and hospital.*
 - Count the number of male and female patients in the dataset.*
 - What are the different types of admissions, and how many patients fall under each category?*
 - Aggregation & Grouping*
 - Count the number of patients admitted per hospital, ordered by highest admissions.*
 - What is the average billing amount for each admission type?*

8. Identify the hospital with the highest total billing amount across all admissions.
 9. Retrieve the top 5 most common medical conditions, sorted by number of patients diagnosed.
 10. Which doctor has treated the most patients, and how many have they treated?
- **Date & Time Analysis**
 11. What is the earliest and latest admission date recorded?
 12. Count the number of patients admitted in the last 30 days from the most recent admission date.
 13. Calculate the average length of stay for patients (difference between discharge and admission date).
 14. Find the month and year with the highest number of admissions and the corresponding count.
 15. Count how many emergency admissions happened on weekends.
 - **Advanced Queries (Joins, Subqueries, and Window Functions)**
 16. Identify the most prescribed medication, along with the number of times it was prescribed.
 17. Retrieve details of patients with "Abnormal" test results, including their medical condition and assigned doctor.
 18. Which insurance provider has covered the highest total billing amount?
 19. Identify patients prescribed both "Aspirin" and "Ibuprofen" during their admission.
 20. Find the room number with the highest patient occupancy and how many patients stayed there.



FINAL PORTFOLIO DELIVERABLES

- ✓ Built a MySQL database to analyze healthcare trends and billing, developed SQL queries to explore patient data and uncover insights, and documented the entire project in a GitHub repository.



LINKS

- Dataset: <https://www.kaggle.com/datasets/parthgosavi/healthcare-dataset>
- Github: <https://github.com/marktheanalyst103/mysql-scripts/blob/main/Uncovering%20Healthcare%20Patterns%20and%20Patient%20Insights%20through%20Mysql%20Data%20Analysis.sql>

PROJECT STEPS

I. SET UP MySQL DATABASE

This screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Navigator' pane on the left shows the 'SCHEMAS' section with 'personal_finance', 'sakila', 'sys', and 'world' listed. The main SQL editor window contains the following SQL script:

```

1 # CREATION OF A DATABASE TO USE
2 • create database healthcare_data;
3 • use healthcare_data;

5 # CREATING THE TABLE WHERE WE WILL INSERT THE EXCEL FILE DATA
6 • create table patient_records (
7     id INT AUTO_INCREMENT PRIMARY KEY,
8     patient_name VARCHAR(100),
9     age INT,
10    gender ENUM('Male', 'Female'),
11    blood_type VARCHAR(5),
12    medical_condition VARCHAR(255),
13    date_of_admission DATE,
14    doctor VARCHAR(100),
15    hospital VARCHAR(100),
16    insurance_provider VARCHAR(100),
17    billing_amount DECIMAL(10,2),
18    room_number INT,
19    admission_type ENUM('Emergency', 'Elective', 'Urgent'),
20    discharge_date DATE,
21    medication VARCHAR(255),
22    test_results ENUM ('Normal', 'Abnormal', 'Inconclusive')

```

The 'Output' pane at the bottom shows the execution results:

Action	Time	Message	Duration / Fetch
select room_number, COUNT(*) as patient_count from patient_records group by room_number order by patient_count desc;	154 16:18:00	400 row(s) returned	0.031 sec / 0.000 sec
select room_number, COUNT(*) as patient_count from patient_records group by room_number order by patient_count desc;	155 16:18:04	1 row(s) returned	0.032 sec / 0.000 sec
use healthcare_data;	156 16:13:31	0 row(s) affected	0.000 sec

SQL script saved to 'G:\My Drive\10 Data Analyst\7 DataSets\MySQL Portfolio Files\Portfolio 2.2>Data\Portfolio 2.2.sql'

II. PREPARATION OF THE MySQL TABLE FOR INSERTING EXCEL FILE DATA

This screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Navigator' pane on the left shows the 'SCHEMAS' section with 'personal_finance', 'sakila', 'sys', and 'world' listed. The main SQL editor window contains the same SQL script as the previous screenshot:

```

1 # CREATION OF A DATABASE TO USE
2 • create database healthcare_data;
3 • use healthcare_data;

5 # CREATING THE TABLE WHERE WE WILL INSERT THE EXCEL FILE DATA
6 • create table patient_records (
7     id INT AUTO_INCREMENT PRIMARY KEY,
8     patient_name VARCHAR(100),
9     age INT,
10    gender ENUM('Male', 'Female'),
11    blood_type VARCHAR(5),
12    medical_condition VARCHAR(255),
13    date_of_admission DATE,
14    doctor VARCHAR(100),
15    hospital VARCHAR(100),
16    insurance_provider VARCHAR(100),
17    billing_amount DECIMAL(10,2),
18    room_number INT,
19    admission_type ENUM('Emergency', 'Elective', 'Urgent'),
20    discharge_date DATE,
21    medication VARCHAR(255),
22    test_results ENUM ('Normal', 'Abnormal', 'Inconclusive')

```

The 'Output' pane at the bottom shows the execution results:

Action	Time	Message	Duration / Fetch
select room_number, COUNT(*) as patient_count from patient_records group by room_number order by patient_count desc;	154 16:18:00	400 row(s) returned	0.031 sec / 0.000 sec
select room_number, COUNT(*) as patient_count from patient_records group by room_number order by patient_count desc;	155 16:18:04	1 row(s) returned	0.032 sec / 0.000 sec
use healthcare_data;	156 16:13:31	0 row(s) affected	0.000 sec

SQL script saved to 'G:\My Drive\10 Data Analyst\7 DataSets\MySQL Portfolio Files\Portfolio 2.2>Data\Portfolio 2.2.sql'

III. INSERTION OF EXCEL FILE DATA

A screenshot of Microsoft Excel showing a spreadsheet titled "healthcare_dataset - Excel". The spreadsheet contains data for patients, including columns for patient name, age, gender, blood type, medical condition, date of admission, doctor, hospital, insurance provider, billing amount, room number, admission type, discharge date, medication, and test results. The data spans multiple rows and columns, with various medical terms and names listed.

- The data from the Microsoft Excel file that I need to upload and use for analysis in MySQL

A screenshot of MySQL Workbench showing a Python script named "Portfolio 2.2.sql". The script is intended to import data from an Excel file into a MySQL database. It includes comments explaining the connection settings and data cleaning steps. A specific section of the script, which enables local INFILE via connection settings, is highlighted with a red box.

```

# ENABLE LOCAL INFILE VIA CONNECTION SETTINGS
# SHOW VARIABLES LIKE 'local_infile'; -- This allows us to upload the Excel file
# [It's ON already, but still not working...]
# Turns out I had to use a Python script to fix this part

## DATA CLEANING
# CLEANING THE NAMES (Capitalize the first letter of each word and remove titles such as Mr., Ms., Mrs., Dr., etc.)

DELIMITER $$

Output:
  Action Output
    # Time Action
    154 16:18:00 select room_number, COUNT(*) as patient_count from patient_records group by room_number order by patient_count DESC;
    155 16:18:04 select room_number, COUNT(*) as patient_count from patient_records group by room_number order by patient_count DESC;
    156 18:13:31 use healthcare_data
  
```

- For certain reasons, MySQL was not allowing me to upload the excel file so I had to find ways how to do it, and using python script is one that worked for me.

jupyter MySQL (healthcare_data importation) Last Checkpoint: 24 days ago Trusted

```
[5]: import pandas as pd
import mysql.connector

# 1. Load the CSV
file_path = "C:/Users/markb/Documents/healthcare_dataset.csv"
df = pd.read_csv(file_path)

# 2. Rename columns to match MySQL table
df.columns = [
    'patient_name', 'age', 'gender', 'blood_type', 'medical_condition',
    'date_of_admission', 'doctor', 'hospital', 'insurance_provider',
    'billing_amount', 'room_number', 'admission_type',
    'discharge_date', 'medication', 'test_results'
]

# 3. Convert date columns to YYYY-MM-DD format
date_cols = ['date_of_admission', 'discharge_date']
for col in date_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce').dt.strftime('%Y-%m-%d')

# 4. Connect to MySQL
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="09273271736143#Abcd", # Replace with your actual MySQL password
    database="healthcare_data"
)
cursor = conn.cursor()

# 5. Insert each row
for index, row in df.iterrows():
    sql = """
        INSERT INTO patient_records
        (patient_name, age, gender, blood_type, medical_condition,
        date_of_admission, doctor, hospital, insurance_provider,
        billing_amount, room_number, admission_type,
        discharge_date, medication, test_results)
        VALUES (%s, %s, %s)
    """
    values = tuple(row)
    cursor.execute(sql, values)

# 6. Finalize
conn.commit()
cursor.close()
conn.close()

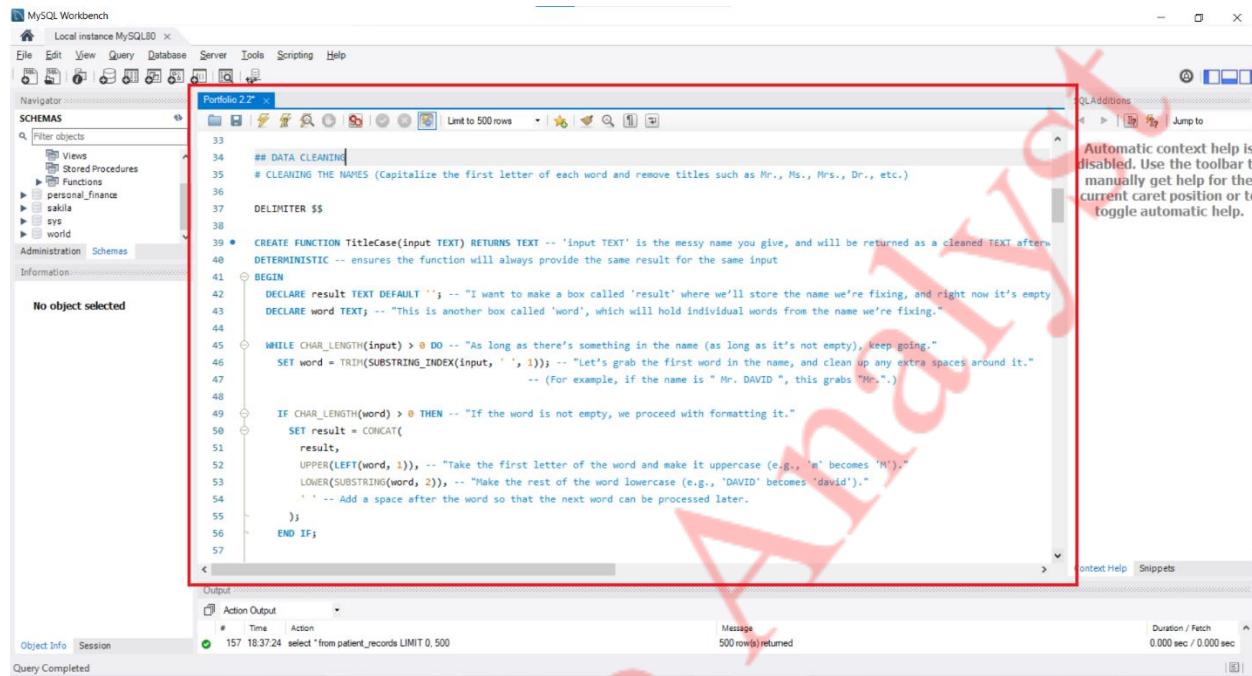
print("✅ Data imported successfully into patient_records.")

✅ Data imported successfully into patient_records.
```

- *The Python script I used to upload the Excel file to MySQL for later analysis.*

IV. DATA CLEANING

- Cleaning the names (Capitalize the first letter of each word and remove titles such as Mr., Ms., Mrs., Dr., etc.)



```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
Views
Stored Procedures
Functions
personal_finance
sakila
sys
world
Administration Schemas
Information
No object selected
Output
Action Output
# Time Action
157 18:37:24 selected * from patient_records LIMIT 0, 500
Message 500 rows returned
Duration / Fetch 0.000 sec / 0.000 sec
Object Info Session
Query Completed

```

SQL Editor Content:

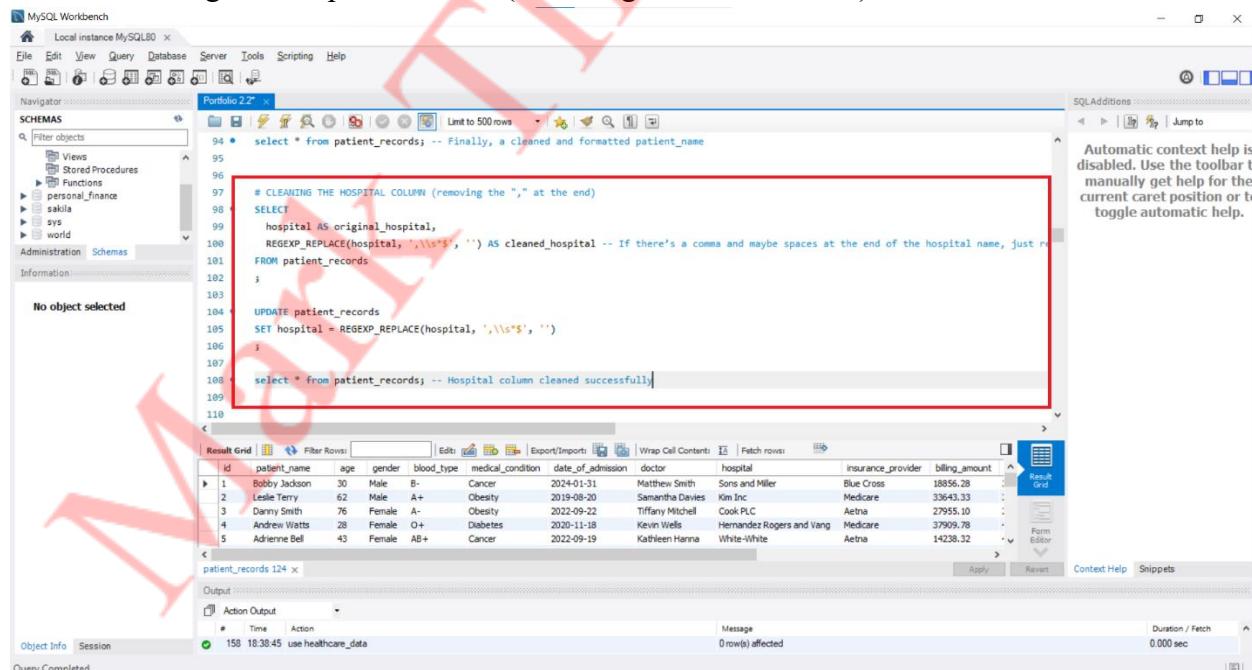
```

Portfolio 2.2" >
33
34 ## DATA CLEANING
35 # CLEANING THE NAMES (Capitalize the first letter of each word and remove titles such as Mr., Ms., Mrs., Dr., etc.)
36
37 DELIMITER $$

38
39 • CREATE FUNCTION TitleCase(input TEXT) RETURNS TEXT -- 'input TEXT' is the messy name you give, and will be returned as a cleaned TEXT after
DETERMINISTIC -- ensures the function will always provide the same result for the same input
40 BEGIN
41     DECLARE result TEXT DEFAULT ''; -- "I want to make a box called 'result' where we'll store the name we're fixing, and right now it's empty"
42     DECLARE word TEXT; -- "This is another box called 'word', which will hold individual words from the name we're fixing."
43
44
45     WHILE CHAR_LENGTH(input) > 0 DO -- "As long as there's something in the name (as long as it's not empty), keep going."
46         SET word = TRIM(SUBSTRING_INDEX(input, ' ', 1)); -- "Let's grab the first word in the name, and clean up any extra spaces around it."
47         -- (For example, if the name is " Mr. DAVID ", this grabs "Mr.".)
48
49         IF CHAR_LENGTH(word) > 0 THEN -- "If the word is not empty, we proceed with formatting it."
50             SET result = CONCAT(
51                 result,
52                 UPPER(LEFT(word, 1)), -- "Take the first letter of the word and make it uppercase (e.g., 'm' becomes 'M')."
53                 LOWER(SUBSTRING(word, 2)), -- "Make the rest of the word lowercase (e.g., 'DAVID' becomes 'david')."
54                 ' ' -- Add a space after the word so that the next word can be processed later.
55             );
56         END IF;
57     END WHILE;
58
59     RETURN result;
60 END$$
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

- Cleaning the Hospital Column (removing the "," at the end)



```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
Views
Stored Procedures
Functions
personal_finance
sakila
sys
world
Administration Schemas
Information
No object selected
Output
Action Output
# Time Action
158 18:38:45 use healthcare_data
Message 0 row(s) affected
Duration / Fetch 0.000 sec
Object Info Session
Query Completed

```

SQL Editor Content:

```

Portfolio 2.2" >
94 • select * from patient_records; -- Finally, a cleaned and formatted patient_name
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

Result Grid:

ID	Patient_Name	Age	Gender	Blood_Type	Medical_Condition	Date_of_Admission	Doctor	Hospital	Insurance_Provider	Billing_Amount
1	Bobby Johnson	30	Male	A+	Cancer	2014-01-31	Matthew Smith	Sons and Miller	Blue Cross	18856.28
2	Leddy Terry	62	Male	A+	Obeesity	2019-08-20	Samantha Davies	Kim Inc	Medicare	33643.33
3	Danny Smith	76	Female	A-	Obeesity	2022-09-22	Tiffany Mitchell	Cook PLC	Aetna	27955.10
4	Andrew Watts	28	Female	O+	Diabetes	2020-11-18	Kevin Wells	Hernandez Rogers and Vang	Medicare	37909.78
5	Adrienne Bell	43	Female	AB+	Cancer	2022-09-19	Kathleen Hanna	White-White	Aetna	14238.32

V. MySQL QUERIES

Basic Queries (Data Exploration)

- How many unique patients are recorded in the dataset?

The screenshot shows the MySQL Workbench interface with a query editor containing the following code:

```
## BASIC QUERIES (Data Exploration)
#1. How many unique patients are recorded in the dataset?
select COUNT(patient_name) as unique_patients
from patient_records
;

-- or,
#2. Retrieve a list of distinct hospitals where patients were admitted.
select COUNT(DISTINCT id) as unique_id
from patient_records
;

#2. Retrieve a list of distinct hospitals where patients were admitted.
select DISTINCT hospital as distinct_hospital
from patient_records
order by hospital -- not really necessary
;
```

The Result Grid shows the output for the first query:

unique_patients
55500

The Output pane shows the execution details:

#	Time	Action
159	18:40:15	select * from patient_records LIMIT 0, 500

Message: 500 row(s) returned

Duration / Fetch: 0.000 sec / 0.000 sec

- Retrieve a list of distinct hospitals where patients were admitted.

The screenshot shows the MySQL Workbench interface with a query editor containing the following code:

```
-- or,
#2. Retrieve a list of distinct hospitals where patients were admitted.
select COUNT(DISTINCT id) as unique_id
from patient_records
;

#2. Retrieve a list of distinct hospitals where patients were admitted.
select DISTINCT hospital as distinct_hospital
from patient_records
order by hospital -- not really necessary
;

#3. Find all patients diagnosed with Diabetes, displaying their name, age, and hospital.
select
    patient_name,
    age,
    hospital
from patient_records
;
```

The Result Grid shows the output for the second query:

distinct_hospital
Abbott and Thompson, Sullivan
Abbott, Inc
Abbott Ltd
Abbott Moore and Williams
Abbott-Castillo
Abbott-Coleman

The Output pane shows the execution details:

#	Time	Action
160	19:34:03	select COUNT(patient_name) as unique_patients from patient_records LIMIT 0, 500

Message: 1 row(s) returned

Duration / Fetch: 0.015 sec / 0.000 sec

3. Find all patients diagnosed with **Diabetes**, displaying their name, age, and hospital.

The screenshot shows the MySQL Workbench interface with a query editor window titled "Portfolio 22". The code in the editor is:

```
121
122 #2. Retrieve a list of distinct hospitals where patients were admitted.
123 • select DISTINCT hospital as distinct_hospital
124   from patient_records
125   order by hospital -- not really necessary
126 ;
127
128 #3. Find all patients diagnosed with Diabetes, displaying their name, age, and hospital.
129 select
130   patient_name,
131   age,
132   hospital
133   from patient_records
134   where
135     medical_condition LIKE '%Diabetes%' -- anything that contains the word 'Diabetes'
136 ;
137
```

A red box highlights the last section of the code. Below the editor is the "Result Grid" pane, which displays the results of the query:

patient_name	age	hospital
Andrew Watts	28	Hernandez Rogers and Vang
Edward Edwards	21	Group Middleton
Connor Hansen	75	Powers Miller, and Flores
Mr. Kenneth Moore	34	Serrano-Dixon
Nicole Rodriguez	30	Poole Inc
Denise Torres	33	LLC Martin

The "Output" pane at the bottom shows the message: "Message 500 row(s) returned".

4. Count the number of **male** and **female** patients in the dataset.

The screenshot shows the MySQL Workbench interface with a query editor window titled "Portfolio 22". The code in the editor is:

```
131
132   age,
133   hospital
134   from patient_records
135   where
136     medical_condition LIKE '%Diabetes%' -- anything that contains the word 'Diabetes'
137
138 #4. Count the number of male and female patients in the dataset.
139 • select COUNT(gender) as unique_female
140   from patient_records
141   where
142     gender LIKE 'Female'
143
144 • select COUNT(gender) as unique_male
145   from patient_records
146   where
147
```

A red box highlights the section starting with "#4. Count the number of male and female patients in the dataset.". Below the editor is the "Result Grid" pane, which displays the results of the query:

unique_female
27726

The "Output" pane at the bottom shows the message: "Message 0.000 sec / 0.000 sec".

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Portfolios

SCHEMAS

Views

Stored Procedures

Functions

personal_finance

sakila

sys

world

No object selected

```

136
137
138 #4. Count the number of male and female patients in the dataset.
139 • select COUNT(gender) as unique_female
140   from patient_records
141   where
142     gender LIKE 'Female'
143
144
145 • select COUNT(gender) as unique_male
146   from patient_records
147   where
148     gender LIKE 'Male'
149
150
151 #5. What are the different types of admissions, and how many patients fall under each category?
152 • select

```

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

unique_male
27774

Result 129 x

Action Output

Time Action

163 19:37:56 select COUNT(gender) as unique_female from patient_records where gender LIKE 'Female' LIMIT 0, 500

Message 1 row(s) returned

Duration / Fetch 0.031 sec / 0.000 sec

Object Info Session

Query Completed

5. What are the different types of admissions, and how many patients fall under each category?

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Portfolios

SCHEMAS

Views

Stored Procedures

Functions

personal_finance

sakila

sys

world

No object selected

```

141
142   where
143     gender LIKE 'Female'
144
145 • select COUNT(gender) as unique_male
146   from patient_records
147   where
148     gender LIKE 'Male'
149
150
151 #5. What are the different types of admissions, and how many patients fall under each category?
152 • select
153   admission_type, COUNT(*) as patient_count
154   from patient_records
155   group by admission_type
156
157

```

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

admission_type	patient_count
Urgent	18576
Emergency	18269
Elective	18655

Result 130 x

Action Output

Time Action

164 19:38:19 select COUNT(gender) as unique_male from patient_records where gender LIKE 'Male' LIMIT 0, 500

Message 1 row(s) returned

Duration / Fetch 0.000 sec / 0.000 sec

Object Info Session

Query Completed

Aggregation & Grouping

6. Count the number of patients admitted per hospital, ordered by the highest admissions.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
151 #5. What are the different types of admissions, and how many patients fall under each category?
152 • select
153     admission_type, COUNT(*) as patient_count
154     from patient_records
155     group by admission_type
156
157
158 ## AGGREGATION & GROUPING
159 #6. Count the number of patients admitted per hospital, ordered by the highest admissions.
160 • select
161     hospital, COUNT(*) as total_admission
162     from patient_records
163     group by hospital
164     order by total_admission desc
165
166
```

The results grid shows the following data:

hospital	total_admission
LLC Smith	44
Ltd Smith	39
Johnson PLC	38
Smith Ltd	37
Smith Group	36
Smith PLC	36

The message bar at the bottom indicates "1 row(s) returned".

7. What is the average billing amount for each admission type?

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
159 #6. Count the number of patients admitted per hospital, ordered by the highest admissions.
160 • select
161     hospital, COUNT(*) as total_admission
162     from patient_records
163     group by hospital
164     order by total_admission desc
165
166
167 #7. What is the average billing amount for each admission type?
168 • select
169     admission_type, ROUND(AVG(billing_amount),2) as avg_bill -- to display only two decimal places
170     from patient_records
171     group by admission_type
172
173
174 #8. Identify the hospital with the highest total billing amount across all admissions.
175 • select
```

The results grid shows the following data:

admission_type	avg_bill
Urgent	25517.36
Emergency	25497.40
Elective	25602.23

The message bar at the bottom indicates "500 row(s) returned".

8. Identify the hospital with the highest total billing amount across all admissions.

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, the following SQL query is displayed:

```
169     admission_type, ROUND(AVG(billing_amount),2) as avg_bill -- to display only two decimal places
170   from patient_records
171   group by admission_type
172   ;
173
174 #8. Identify the hospital with the highest total billing amount across all admissions.
175   select
176     hospital, SUM(billing_amount) as total_bill
177   from patient_records
178   group by hospital
179   order by total_bill desc
180   limit 1
181   ;
182
183 #9. Retrieve the top 5 most common medical conditions, sorted by the number of patients diagnosed.
184 •   select
185     medical_condition, COUNT(*) as patients_diagnosed
```

The results grid shows the output of the query:

hospital	total_bill
Johnson PLC	1084202.70

The message area at the bottom indicates 3 row(s) returned.

9. Retrieve the top 5 most common medical conditions, sorted by the number of patients diagnosed.

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, the following SQL query is displayed:

```
175 •   select
176     hospital, SUM(billing_amount) as total_bill
177   from patient_records
178   group by hospital
179   order by total_bill desc
180   limit 1
181   ;
182
183 #9. Retrieve the top 5 most common medical conditions, sorted by the number of patients diagnosed.
184 •   select
185     medical_condition, COUNT(*) as patients_diagnosed
186   from patient_records
187   group by medical_condition
188   order by patients_diagnosed desc
189   limit 5
190   ;
```

The results grid shows the output of the query:

medical_condition	patients_diagnosed
Arthritis	9308
Diabetes	9304
Hypertension	9245
Obesity	9231
Cancer	9227

The message area at the bottom indicates 1 row(s) returned.

10. Which doctor has treated the most patients, and how many have they treated?

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Query Editor:** Portfolio 2.2 window containing the following SQL code:

```
185 medical_condition, COUNT(*) as patients_diagnosed
186 from patient_records
187 group by medical_condition
188 order by patients_diagnosed desc
189 limit 5
190
191
192 #10. Which doctor has treated the most patients, and how many have they treated?
193 select
194     doctor, COUNT(*) as patients_treated
195 from patient_records
196 group by doctor
197 order by patients_treated desc
198 limit 1
199
200
201 ## DATE & TIME ANALYSIS
```
- Result Grid:** Shows the output of the query:

doctor	patients_treated
Michael Smith	27
- Output:** Action Output pane shows the message: "5 row(s) returned".
- Message Bar:** "Duration / Fetch 0.031 sec / 0.000 sec".

Date & Time Analysis

11. What is the earliest and latest admission date recorded?

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Query Editor:** Portfolio 2.2 window containing the following SQL code:

```
195 from patient_records
196 group by doctor
197 order by patients_treated desc
198 limit 1
199
200
201 ## DATE & TIME ANALYSIS
202 #11. What is the earliest and latest admission date recorded?
203 select
204     MIN(date_of_admission) as earliest_admission,
205     MAX(discharge_date) as latest_admission
206 from patient_records
207
208
209 #12. Count the number of patients admitted in the last 30 days from the most recent admission date.
210
211 select
212     COUNT(*) as patients_last_30days
```
- Result Grid:** Shows the output of the query:

earliest_admission	latest_admission
2019-05-08	2024-06-06
- Output:** Action Output pane shows the message: "1 row(s) returned".
- Message Bar:** "Duration / Fetch 0.062 sec / 0.000 sec".

12. Count the number of patients admitted in the last 30 days from the most recent admission date.

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code is as follows:

```
205 MAX(discharge_date) as latest_admission
206 from patient_records
207 ;
208
209 #12. Count the number of patients admitted in the last 30 days from the most recent admission date.
210 select
211 COUNT(*) as patients_last_30days
212 from patient_records
213 where
214 date_of_admission >=
215 (
216 select MAX(date_of_admission) - INTERVAL 30 day
217 from patient_records
218 )
219 ;
220
221 #13. Calculate the average length of stay for patients (difference between discharge and admission date).
222
```

The result grid shows the output of the query:

patients_last_30days
978

The message area at the bottom indicates the query completed successfully:

Object Info Session Query Completed

Result 138 x

Action Output

Time Action Message Duration / Fetch

172 19:42:44 select MIN(date_of_admission) as earliest_admission, MAX(discharge_date) as latest_admission from patient_records 1 row(s) returned 0.016 sec / 0.000 sec

Context Help Snippets

13. Calculate the average length of stay for patients (difference between discharge and admission date).

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code is as follows:

```
210 select
211 COUNT(*) as patients_last_30days
212 from patient_records
213 where
214 date_of_admission >=
215 (
216 select MAX(date_of_admission) - INTERVAL 30 day
217 from patient_records
218 )
219 ;
220
221 #13. Calculate the average length of stay for patients (difference between discharge and admission date).
222 select
223 ROUND(AVG(DATEDIFF(discharge_date, date_of_admission)),2) as avg_length_of_stay
224 from patient_records
225 ;
```

The result grid shows the output of the query:

avg_length_of_stay
15.51

The message area at the bottom indicates the query completed successfully:

Object Info Session Query Completed

Result 139 x

Action Output

Time Action Message Duration / Fetch

173 19:43:10 select COUNT(*) as patients_last_30days from patient_records where date_of_admission >= (select MAX(d... 1 row(s) returned 0.031 sec / 0.000 sec

Context Help Snippets

14. Find the month and year with the highest number of admissions and the corresponding count.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the `healthcare_data` database selected.
- SQL Editor:** Contains the following SQL code:

```
221 #13. Calculate the average length of stay for patients (difference between discharge and admission date).
222 select
223     ROUND(AVG(DATEDIFF(discharge_date, date_of_admission)),2) as avg_length_of_stay
224 from patient_records
225
226
227 #14. Find the month and year with the highest number of admissions and the corresponding count.
228 select
229     DATE_FORMAT(date_of_admission, '%Y-%M') as admission_month,
230     COUNT(*) as admission_count
231 from patient_records
232 group by admission_month
233 order by admission_count desc
234 limit 1
235
236
```
- Result Grid:** Displays the result of the query:

admission_month	admission_count
2020-August	1014
- Output:** Shows the execution message: "20 20:12:47 select DATE_FORMAT(date_of_admission, '%Y-%M') as admission_month, COUNT() as admission_count ... 1 row(s) returned".

15. Count how many emergency admissions happened on weekends.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the `healthcare_data` database selected.
- SQL Editor:** Contains the following SQL code:

```
230 COUNT(*) as admission_count
231 from patient_records
232 group by admission_month
233 order by admission_count desc
234 limit 1
235
236
237 #15. Count how many emergency admissions happened on weekends.
238 select
239     COUNT(*) as emergency_weekend_admissions
240     from patient_records
241     where
242         admission_type = 'Emergency'
243         and
244         DAYOFWEEK(date_of_admission) IN (1,7) -- 1 means SUNDAY, 7 means SATURDAY
245
246
```
- Result Grid:** Displays the result of the query:

emergency_weekend_admissions
5193
- Output:** Shows the execution message: "175 19:44:10 select DATE_FORMAT(date_of_admission, '%Y-%M') as admission_month, COUNT() as admission_count ... 1 row(s) returned".

Advanced Queries (Joins, Subqueries, and Window Functions)

16. Identify the most prescribed medication, along with the number of times it was prescribed.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the "Schemas" section with "personal_finance" selected.
- SQL Editor:** Contains the following SQL code:

```
## ADVANCED QUERIES (JOINS, SUBQUERIES, AND WINDOW FUNCTIONS)
#16. Identify the most prescribed medication, along with the number of times it was prescribed.
select
    medication,
    COUNT(*) as prescription_count
from patient_records
group by medication
order by prescription_count desc
limit 1;
```
- Result Grid:** Displays the result of the query:

medication	prescription_count
Lipitor	11140
- Output:** Shows the execution message: "176 19:44:41 select COUNT(*) as emergency_weekend_admissions from patient_records where admission_type = 'Emergency' 1 row(s) returned".

17. Retrieve details of patients with "Abnormal" test results, including their medical condition and assigned doctor.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the "Schemas" section with "personal_finance" selected.
- SQL Editor:** Contains the following SQL code:

```
#16. Identify the most prescribed medication, along with the number of times it was prescribed.
#17. Retrieve details of patients with "Abnormal" test results, including their medical condition and assigned doctor.
select *
from patient_records
where
    test_results = 'Abnormal';
```
- Result Grid:** Displays the result of the query:

ID	Patient Name	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount
4	Andrew Watts	28	Female	O+	Diabetes	2020-11-18	Kevin Wells	Hernandez Rogers and Vang	Medicare	37909.78
5	Adrienne Bell	43	Female	AB+	Cancer	2022-09-19	Kathleen Hanna	White-White	Aetna	14238.32
9	Jasmine Aguilar	82	Male	AB+	Asthma	2020-07-01	Daniel Ferguson	Sons Rich and	Cigna	50119.22
13	Connor Hansen	75	Female	A+	Diabetes	2019-12-12	Kenneth Fletcher	Powers Miller, and Flores	Cigna	43282.28
18	Mrs. Jamie Campbell	38	Male	AB-	Obesity	2020-03-08	Justin Kim	Torres, and Harrison Jones	Cigna	17440.47
- Output:** Shows the execution message: "177 19:45:10 select medication, COUNT(*) as prescription_count from patient_records group by medication order by pr... 1 row(s) returned".

18. Which insurance provider has covered the highest total billing amount?

```
#18. Which insurance provider has covered the highest total billing amount?  
select  
    insurance_provider,  
    SUM(billing_amount) as total_bill_covered  
from patient_records  
group by insurance_provider  
order by total_bill_covered desc  
limit 1  
;
```

insurance_provider	total_bill_covered
Oigna	287139345.20

19. Identify patients prescribed both "Aspirin" and "Ibuprofen" during their admission.

```
select patient_name  
from patient_records  
where medication IN ('Aspirin', 'Ibuprofen')  
group by patient_name  
HAVING COUNT(DISTINCT medication) = 2 -- ensures both were prescribed
```

patient_name
Aaron Lopez
Aaron Moore
Aaron Smith
Adam Gonzalez
Adam Hernandez
Alex Williams

20. Find the room number with the highest patient occupancy and how many patients stayed there.

The screenshot shows the MySQL Workbench interface. The SQL Editor tab contains the following code:

```
276 • select patient_name
277   from patient_records
278   where medication IN ('Aspirin','Ibuprofen')
279   group by patient_name
280   HAVING COUNT(DISTINCT medication) = 2 -- ensures both were prescribed
281
282
283 #20. Find the room number with the highest patient occupancy and how many patients stayed there.
284
285 select
286   room_number,
287   COUNT(*) as patient_count
288   from patient_records
289   group by room_number
290   order by patient_count desc
291
292
```

The result grid shows the following data:

room_number	patient_count
393	181

The Output pane shows the message: "180 19:48:03 select patient_name from patient_records where medication IN ('Aspirin','Ibuprofen') group by patient_name H... 500 row(s) returned".

KEY INSIGHTS

- The dataset includes **55,000 unique patients**, with a near-even gender split — **27,726 female** patients.
- **Admission types** are evenly distributed: **Urgent (18,576)**, **Emergency (18,269)**, **Elective (18,655)**.
- **LLC Smith Hospital** recorded the highest number of patient admissions (44), while **Johnson PLC** had the **highest total billing at ₦1,084,202.70**.
- Among **medical conditions**, the most common are **Arthritis (9,308)**, **Diabetes (9,304)**, and **Hypertension (9,245)**.
- **Michael Smith** stands out as the doctor with the most treated patients (27).
- Patients stay in the hospital for an **average of 15.51 days**, and **978** admissions occurred in the last **30 days**.
- The **busiest month** was **August 2020** with **1,014 admissions**.
- **5,193 emergency admissions** took place during weekends, suggesting high weekend demand.
- The **most prescribed medication** appears over **11,140 times**, and **Cigna** was the top insurance provider by billing at **₦287.1M**.
- **Room 393** had the highest occupancy, accommodating **181 patients**.

RECOMMENDATIONS

- **Staffing & Resource Allocation:** Given the even distribution across admission types and frequent weekend emergencies, hospitals should ensure adequate weekend staffing and resources, especially for emergency cases.
- **Doctor Workload Monitoring:** With Michael Smith treating the most patients, reviewing doctor-patient ratios may help avoid potential burnout or improve efficiency.
- **Facility Usage:** High usage of Room 393 may point to availability issues or operational preference — deeper analysis could ensure better space optimization.
- **Medical Supply Planning:** Frequent prescriptions and top medical conditions (e.g., arthritis, diabetes) highlight areas to prioritize in medication inventory and chronic care programs.

- **Insurance & Billing Optimization:** Cigna's large billing coverage may indicate a strong partnership — but reviewing claim approval rates and processing times could help manage financial flows more effectively.



REFLECTION & NEXT STEPS

- This project sharpened my skills in writing complex SQL queries and analyzing real-world healthcare data. It gave me practical experience uncovering trends in patient care and billing. Moving forward, I plan to connect the data to a visualization tool to present insights more clearly and explore long-term patterns.