



BERLIN SC

Assignment Title: Relational Database Design, Implementation, and Analytics

Assignment Type: Set exercise

Programme Title: Enterprise Data Warehouses and Database Management Systems

Name: Jinesh Kumar Kanakmal Jain

Year:2025

1 CONTENTS

Table of Content

1. Content.....	2
2. Introduction.....	4
3. Task 1:-	
1. Domain selection & Description.....	
2. Create Table Scripts, Constraints and the schemas.....	
3. Entity Identification and Relationship Design (ERD).....	
4. Entity-Relationship Diagram (ERD).....	
5. Normalization to third normal form (3NF).....	
4. Task 2:-	
1. Sample Data Population.....	
2. View Creation.....	
5. Task 3:-	
1. Query: Identification Doctors who Generate the Highest Revenue.....	
2. Query Hospitals With an Average Length of Stay Above 5 Days.....	
3. Query Categorising Patients Based on Billing Amount...	
4. Query Identifying Patients with Billing above the Overall Average.....	

5. Query: Ranking Patients by Billing Amount (Window Function).....
6. Stored Procedure: Calculating Total Billing for a Specific Patient
6. Task 4:-
1. Components of the CAP Theorem.....
2. Why no distributed Database can guarantee all three simultaneously.....
3. Applying CAP to your Healthcare Database System.....
4. Reflection: How CAP theorem influences Healthcare Database Design.....
7. Task 5:-
Query Performance and Optimization.....
8. Bibliography.....
9. Appendix.....

Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people texts I clearly declare it through the good use of references following academic ethics.

(In the case that is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the program).

Jinesh Kumar Kanakmal Jain

Name and Surname (Capital letters):

Date: 30/01/2026

2 INTRODUCTION

In the Healthcare system Data management system is necessary and strategically important to have an idea what is currently running in Background, where operational decisions, Financial planning, service quality of hospital depend the accuracy and with the actual information.

Hospital works on the environment managing patient admission, insurance process, Billing activities, Medical Condition, Past History of patient, Doctors presence in the hospital and other activities.

Without a structure Database system all of the above process is not possible, it will be so conjusted that it can lead to the failure of the system which we don't want. Relational database address these challenges by providing a reliable foundation for data consistency, operational transperancy and analytical insight (Connolly & Begg, 2015).

This project develops a complete relational database solution for healthcare domain and to be more precise for a chain of group of hospitals using a real dataset sourced from Kaggle without any tempering of data before downloading. The dataset Initially provided in excel .csv format 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 , Length of Stay. After importing the dataset into Mysql, the was cleaned, organized and transformed into a normalized schema. This will emphasis in building the system that will support Healthcare business need and changes requirement for hospitals and improving the data accuracy and which will enable reporting at Managerial level for decision making.

The assignment follow a practical industry workflow. It begins with domain understanding and entity identification, followed (3NF) normalized form, core entities , Junction table for many to many relationship, constrains, views, advanced queries, performance tracking, Billing analysis which will improve and increase patient flow with priority patients and business giving patient for growth of Hospital.

The project also explained the CAP Theorem(Consistency, Availability, Partition Tolerance) Query optimization strategies, stored procedures for immediate search or result. By integration database design, analytics & performance evaluation, categorizing the patient, this report demonstrate well- engineered relational database can transform raw Healthcare data into a strategic asset that supports informed decision making and operational excellence.

DOMAIN:- HEALTHCARE



FIG 1

3 TASK ONE

1. Domain Selection & Description.

The Healthcare Hospital Management Domain is selected for the project/assignment. It's a growing sector every year which requires accurate, consistent data to support hospital Management, hospital records, patient Medical condition, Doctor status and administrative decision making.

Hospital manages complex structure of the workflow involving Patient, admission, diagnosis, rooms, treatments, insurance claims and financial transaction as backbone for the industry. If the structured database system is not there it will break the system leading to the inefficiencies, duplicate of records. If a relational database is there it will provide a stable information with a strong foundation for the organization, which will ensure Consistency, reducing redundancy and enabling reports for managerial decision making.

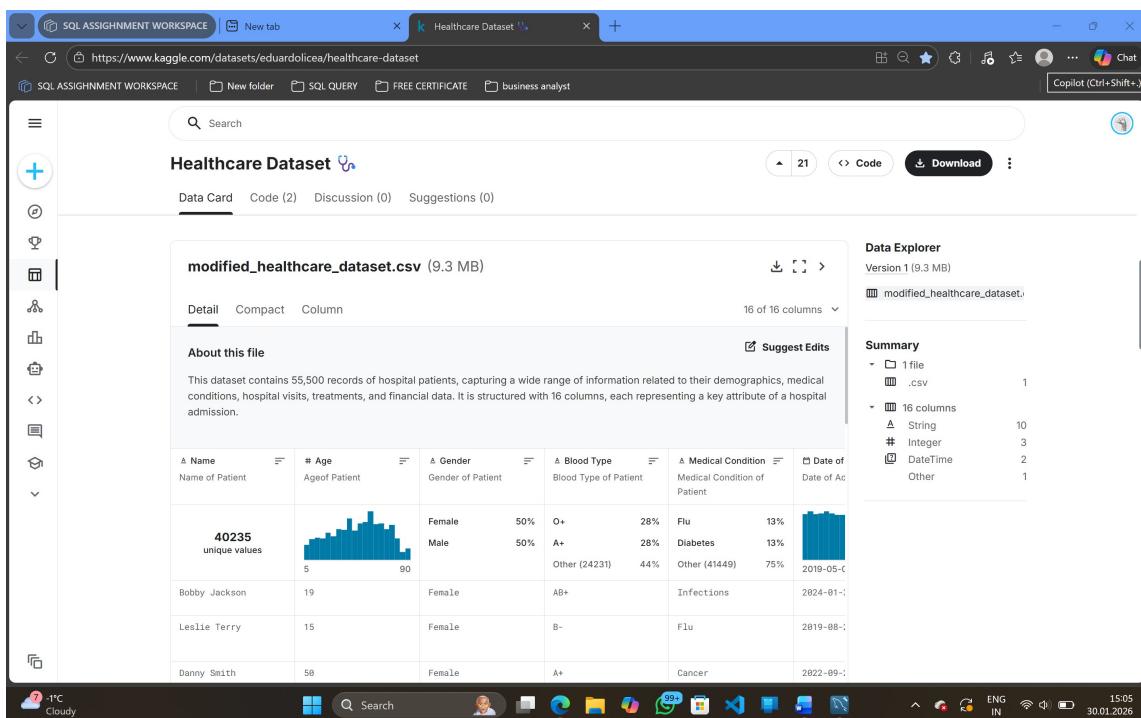


Fig 2

DATASET :- <https://www.kaggle.com/datasets/eduardolicea/healthcare-dataset>

A real world Healthcare dataset is taken from the Kaggle. The dataset originally in excel format which contain 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 , Length of Stay. The raw dataset was imported into MYSQL as a starting table (Raw_Healthcare_data_tb) and served as basis for designning a fully normalized relational database.

2. Create Table Scripts, Constraints and the schemas.

The database (Healthcare_db) was created in Mysql in beginning with a raw data table, which stores the all raw data exactly as it is provided from the original Kaggle dataset. From this the data was decomposed or segregate into multiple normalized entities.

CORE ENTITY Tables created. And Relationship / Transaction tables

Table Name	Purpose
Patient_tb	Stores patient demographic information
Doctor_tb	Stores doctor names
Hospital_tb	Stores hospital names
Insurance_tb	Stores insurance providers
Room_tb	Stores room numbers
Medication_tb	Stores Medication names
Medical_Condition_tb	Stores medical conditions
Test_tb	Stores Diagnostic Test names

Table Name	Purpose
Admission_tb	Links patient, doctor, hospital, insurance, room, and admission details
Billing_tb	Stores billing amounts and payment status for each admission
Patient_Medical_Condition_tb	Junction table for patient-condition (M:N)
Prescription_tb	Junction table for patient-medication (M:N)
Test_Result_tb	Junction table for patient-test results (M:N)

Constrain used are used

Primary keys: Auto_increment fro unique identification

Foreign keys: for reference integrity.

Not null: data is filled no empty columns.

Default: used for Payment_status in billing for no duplication.

3. Entity Identification and Relationship Design (ERD).

For these I have 12+ entity and with their relationshop.

3.1 One to Many (1:M) Relationship are

Patient ---> Admission.

Doctor ---> Admission.

Hospital ---> Admission.

Insurance ---> Admission.

Room ---> Admission.

3.2 Many to Many (M:N) Relationship are

Patient ---> Medical condition.

Junction: Patient_Medical_Condition_tb

Patient ---> Medication.

Junction: Prescription_tb

Patient ---> Test.

Junction: Test_Result_tb

In the Many to many relationship you can see a patient can receive multiple medication, test diagnostics results etc.

4. Entity-Relationship Diagram (ERD).

Enttiy relationship diagram represents the full structure of the Healthcare database which has total 13 entites, each of them stores the specific category of heahrthcare information. The database uses foreighn key to maintain the the integrity.

Core Entites are Patient_tb – Stores patient demographic details, Doctor_tb – Stores doctor names, Hospital_tb – Stores hospital names, Insurance_tb – Stores insurance providers, Room_tb – Stores room numbers, Medication_tb – Stores medication names, Medical_Condition_tb – Stores medical conditions, Test_tb – Stores diagnostic test names.

Transactional Entities are Admission_tb – Links patient, doctor, hospital, insurance, and room for each admission, Billing_tb – Stores billing amount and payment status.

Junction Tables are Patient_Medical_Condition_tb – Patient ↔ Condition, Prescription_tb – Patient ↔ Medication, Test_Result_tb – Patient ↔ Test.

The purpose of the erd diagram is to get a clear visual structure of data how it flows in healthcare system. It shows all the constraints and data present in the structure for clear understanding in a visual form. How the patient interact with the healthcare system like patient doctor, hospital , insurancce provider, billing, prescription diagnostic reports etc.

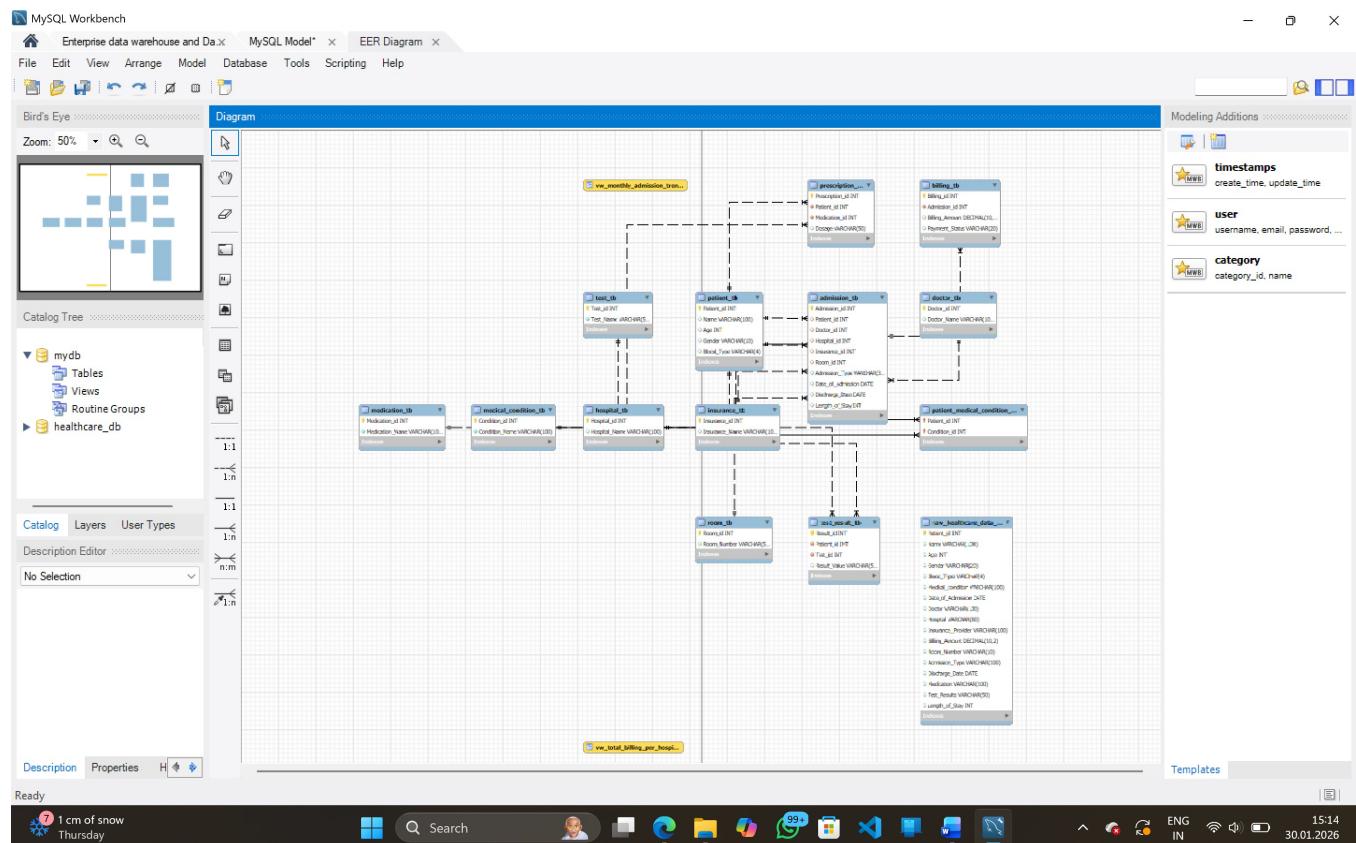


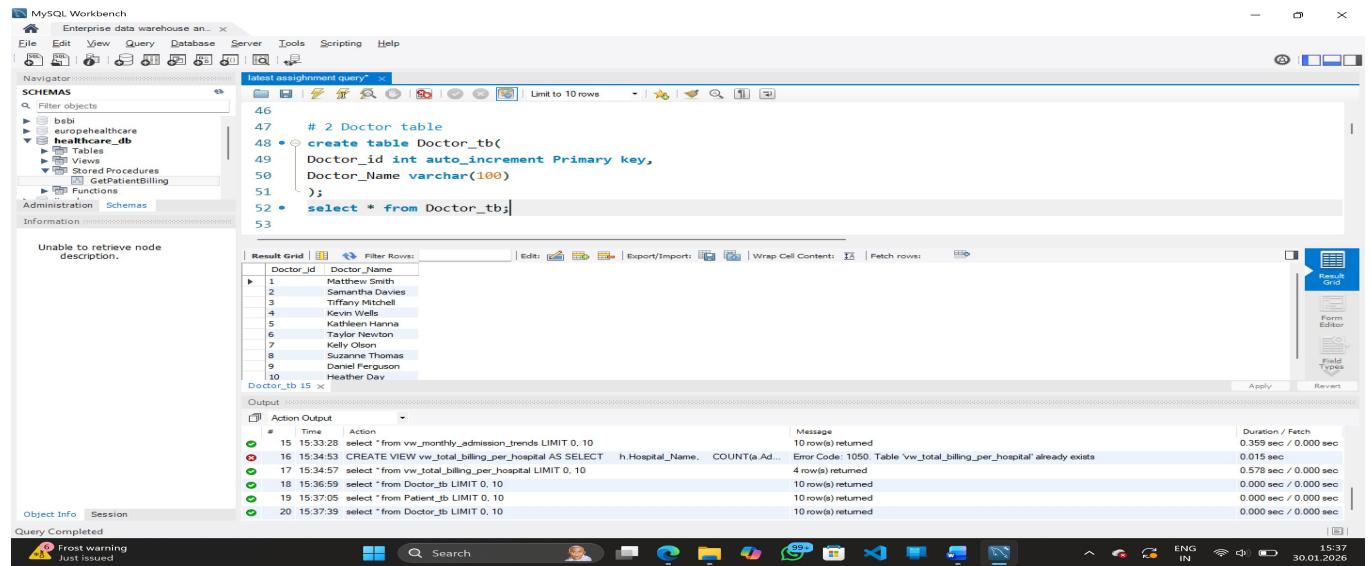
FIG 3

5. Normalization to Third Normal Form (3NF).

The raw Kaggle data set was used. It contained repeated values and multi values attributes, and inconsistent structure. Therefore the schema was normalized to 3NF.

Normalization:- Patient, Name, Age, Gender, Doctor, Blood_Type, Medical_condition, Date_of_Admission, Doctor, Hospital, Insurance_Provider, Billing_Amount decimal, Room_Number, Admission_Type, Discharge_Date date, Medication, Test_Results, Length_of_Stay. Before Normalization all attributes were stored in a single raw table causing

duplication. After Normalisation: separate tables were created as entities required like patient, doctor, insurance, billing etc. foreign keys were used to maintain data integrity in admission table.



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

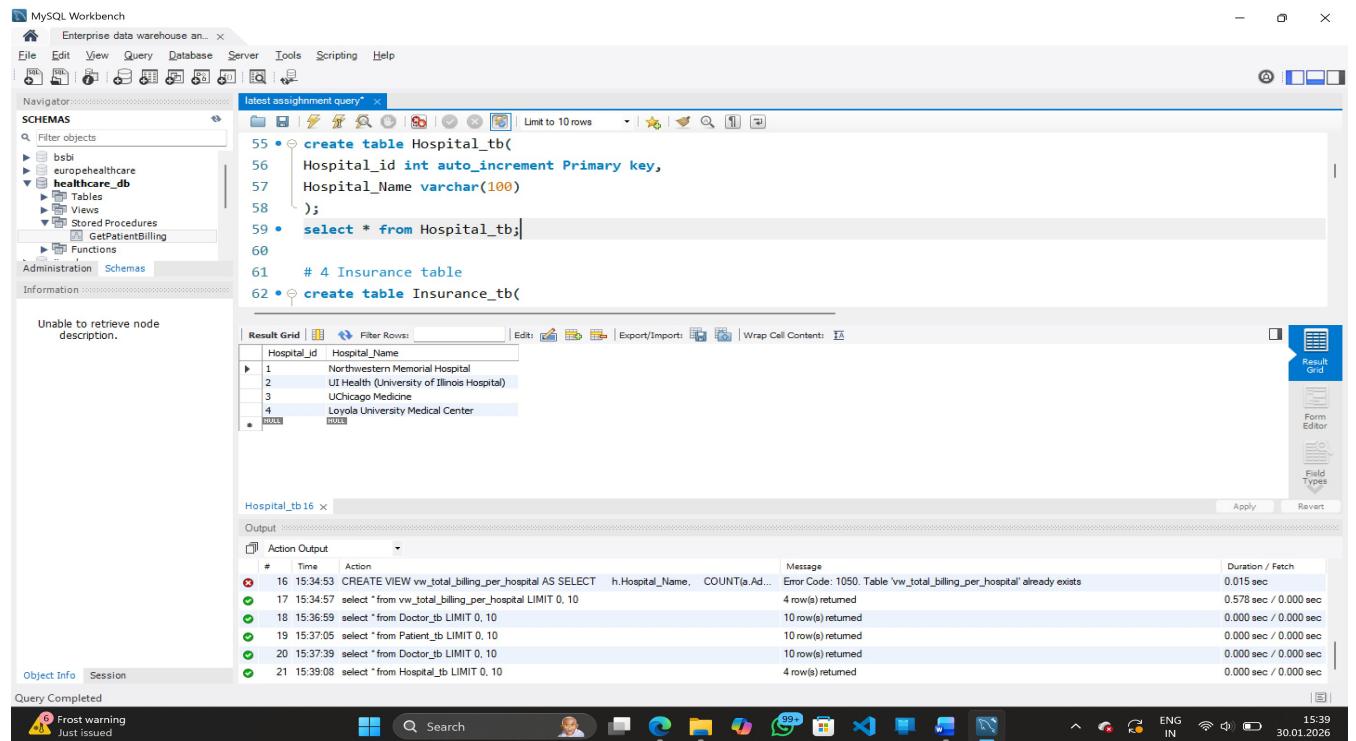
- Schemas:** healthcare_db
- Tables:** Doctor_tb
- Code:**

```

46
47. # 2 Doctor table
48. • create table Doctor_tb(
49. | Doctor_id int auto_increment Primary key,
50. | Doctor_Name varchar(100)
51. );
52. • select * from Doctor_tb;
53

```
- Result Grid:** Displays 10 rows of data from the Doctor_tb table, listing names such as Matthew Smith, Samantha Davies, Timothy Mellish, Kevin Wells, Kathleen Hanna, Taylor Newton, Kelly Olson, Suzanne Thomas, Daniel Ferguson, and Heather Day.
- Action Output:** Shows the execution history with actions like SELECT, CREATE VIEW, and various INSERT/UPDATE operations, along with their timestamps, durations, and error messages.
- Session:** Shows a warning about a frost warning.

FIG 4



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

- Schemas:** healthcare_db
- Tables:** Hospital_tb, Insurance_tb
- Code:**

```

55. • create table Hospital_tb(
56. | Hospital_id int auto_increment Primary key,
57. | Hospital_Name varchar(100)
58. );
59. • select * from Hospital_tb;
60
61. # 4 Insurance table
62. • create table Insurance_tb(

```
- Result Grid:** Displays 4 rows of data from the Hospital_tb table, listing Northwestern Memorial Hospital, UI Health (University of Illinois Hospital), UChicago Medicine, and Loyola University Medical Center.
- Action Output:** Shows the execution history with actions like SELECT, CREATE VIEW, and various INSERT/UPDATE operations, along with their timestamps, durations, and error messages.
- Session:** Shows a warning about a frost warning.

FIG 5

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

- bbei
- europehealthcare
- healthcare_db**
 - Tables
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information: Unable to retrieve node description.

latest assignment query*

```

58  );
59 • select * from Hospital_tb;
60
61 # 4 Insurance table
62 • create table Insurance_tb(
63 Insurance_id int auto_increment Primary key,
64 Insurance_Name varchar(100)
65 );
66 • select * from Insurance_tb;
67

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid

Insurance_id	Insurance_Name
1	Blue Cross
2	UnitedHealthcare
3	Aetna
4	Cigna
5	Medicare

Insurance_tb 17 x

Output: Action Output

#	Time	Action	Message	Duration / Fetch
17	15:34:57	select * from vx_total_billing_per_hospital LIMIT 0, 10	4 row(s) returned	0.578 sec / 0.000 sec
18	15:36:59	select * from Doctor_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
19	15:37:05	select * from Patient_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
20	15:37:39	select * from Doctor_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
21	15:39:08	select * from Hospital_tb LIMIT 0, 10	4 row(s) returned	0.000 sec / 0.000 sec
22	15:39:44	select * from Insurance_tb LIMIT 0, 10	5 row(s) returned	0.000 sec / 0.000 sec

Object Info Session Query Completed

Frost warning In effect

Windows Taskbar: Search, File Explorer, Microsoft Edge, VS Code, etc.

FIG 6

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

- bbei
- europehealthcare
- healthcare_db**
 - Tables
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information: Unable to retrieve node description.

latest assignment query*

```

67
68 # 5 Room table
69 • create table Room_tb(
70 Room_id int auto_increment primary key,
71 Room_Number varchar(50)
72 );
73 • select* from Room_tb;
74
75 # 6 Medication table
76 • CREATE TABLE Medication_tb(

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows: | Result Grid

Room_id	Room_Number
1	328
2	265
3	205
4	450
5	458

Room_tb 18 x

Output: Action Output

#	Time	Action	Message	Duration / Fetch
18	15:36:59	select * from Doctor_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
19	15:37:05	select * from Patient_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
20	15:37:39	select * from Doctor_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
21	15:39:08	select * from Hospital_tb LIMIT 0, 10	4 row(s) returned	0.000 sec / 0.000 sec
22	15:39:44	select * from Insurance_tb LIMIT 0, 10	5 row(s) returned	0.000 sec / 0.000 sec
23	15:40:46	select* from Room_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec

Object Info Session Query Completed

Frost warning In effect

Windows Taskbar: Search, File Explorer, Microsoft Edge, VS Code, etc.

FIG 7

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

- File Menu:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Navigator pane shows the schema structure. The **healthcare_db** schema contains:
 - Tables: Room_tb, Medication_tb, Doctor_tb, Hospital_tb, Insurance_tb.
 - Views: None.
 - Stored Procedures: GetPatientBilling.
 - Functions: None.
- Information:** Unable to retrieve node description.
- Query Editor:** Title: latest assignment query*. The code is as follows:

```
73 • select* from Room_tb;
74
75 # 6 Medication table
76 • CREATE TABLE Medication_tb(
77 Medication_id INT AUTO_INCREMENT PRIMARY KEY,
78 Medication_Name VARCHAR(100) NOT NULL
79 );
80 • select * from Medication_tb;
81
82 # 7 Medical Condition table
```
- Result Grid:** Title: Medication_tb 19 x. It displays the following data:

Medication_Id	Medication_Name
1	Azithromycin
2	Tamiflu
3	Cisplatin
4	Prednisone
5	Beta-blockers
- Action Output:** Shows the history of recent actions:

#	Time	Action	Message	Duration / Fetch
19	15:37:05	select * from Patient_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
20	15:37:39	select * from Doctor_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
21	15:39:00	select * from Hospital_tb LIMIT 0, 10	4 row(s) returned	0.000 sec / 0.000 sec
22	15:39:44	select * from Insurance_tb LIMIT 0, 10	5 row(s) returned	0.000 sec / 0.000 sec
23	15:40:46	select * from Room_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
24	15:41:38	select * from Medication_tb LIMIT 0, 10	10 row(s) returned	0.031 sec / 0.000 sec
- Status Bar:** Black ice warning in effect.

FIG 8

MySQL Workbench

Enterprise data warehouse an... x

File Edit View Query Database Server Tools Scripting Help

Navigator: latest assignment query*

SCHEMAS

Filter objects

- b3b1
- europehealthcare
- healthcare_db
 - Tables
 - Views
 - Stored Procedures
 - GetPatientBilling
 - Functions

Administration Schemas

Information

Unable to retrieve node description.

Result Grid | Filter Rows: Condition_id Condition_Name

Condition_id	Condition_Name
1	Infections
2	Flu
3	Cancer
4	Asthma
5	Heart Disease

Medical_Condition_tb_20 x

Action Output

#	Time	Action	Message	Duration / Fetch
20	15:37:39	select * from Doctor_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
21	15:39:08	select * from Hospital_tb LIMIT 0, 10	4 row(s) returned	0.000 sec / 0.000 sec
22	15:39:44	select * from Insurance_tb LIMIT 0, 10	5 row(s) returned	0.000 sec / 0.000 sec
23	15:40:46	select * from Room_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
24	15:41:38	select * from Medication_tb LIMIT 0, 10	10 row(s) returned	0.031 sec / 0.000 sec
25	15:42:25	Select* from Medical_Condition_tb LIMIT 0, 10	8 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Query Completed

Black ice warning
In effect

Search

ENG IN

15:42 30.01.2026

FIG 9

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the database schema, including the 'healthcare_db' schema which contains a 'Tables' node. The main area shows a query editor with the following SQL code:

```

88
89  # 8 Test table table
90 • CREATE TABLE Test_tb(
91   Test_id INT AUTO_INCREMENT PRIMARY KEY,
92   Test_Name VARCHAR(50) NOT NULL
93 );
94 • Select* from Test_tb;
95
96 # Relation ship tables or Entity tables
97 # 1 Admission_tb

```

Below the code, a Result Grid shows the data from the 'Test_tb' table:

Test_Id	Test_Name
1	Normal
2	Abnormal
3	Inconclusive
*	NULL

The Output pane at the bottom displays the Action Output log:

#	Time	Action	Message	Duration / Fetch
21	15:39:08	select * from Hospital_tb LIMIT 0, 10	4 row(s) returned	0.000 sec / 0.000 sec
22	15:39:44	select * from Insurance_tb LIMIT 0, 10	5 row(s) returned	0.000 sec / 0.000 sec
23	15:40:46	select* from Room_tb LIMIT 0, 10	10 row(s) returned	0.000 sec / 0.000 sec
24	15:41:38	select * from Medication_tb LIMIT 0, 10	10 row(s) returned	0.031 sec / 0.000 sec
25	15:42:25	Select from Medical_Condition_tb LIMIT 0, 10	8 row(s) returned	0.000 sec / 0.000 sec
26	15:43:03	Select* from Test_tb LIMIT 0, 10	3 row(s) returned	0.031 sec / 0.000 sec

FIG 10

4 Task TWO

1. Sample Data Population

The data was taken from Kaggle in csv format and downloaded in laptop after that I load the data set into the mysql tables which was created and the data was loaded directly into the my sql workbench and created healthcare database.

I used the database and created several tables according to the need of the entity for the project and to the business requirement required. and operation of data was done and insert procedure begin from here there were total 50585 rows in the database. and more than 3 columns as 3 nf formulation was also needed to maintain the data requirement of the project.

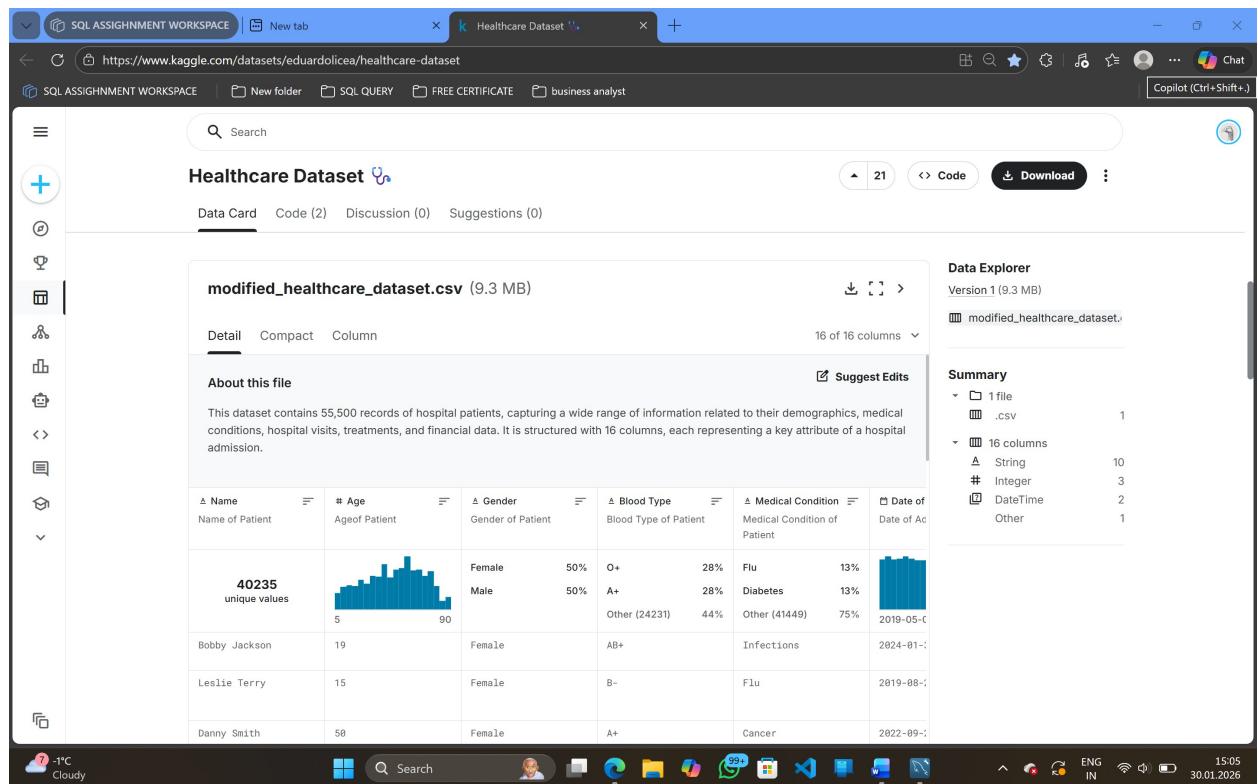


FIG 11

2. View Creation

2.1 View 1

The view one contain total billing per hospital which was calculated with the the total admission, total revenue, average length of stay with all of these I was able to calculate the total billing per hospital.

These help us tpo make financial planning performance benchm, arking and strategic Decision making for future and decide the budget of the hospital.

```
MySQL Workbench
Enterprise data warehouse an...
File Edit View Query Database Server Tools Scripting Help
latest assignment query*
268 *
269 ## VIEW 1 - Summary / Aggregation View
270 CREATE VIEW vw_total_billing_per_hospital AS
271 SELECT
272     h.Hospital_Name,
273     COUNT(a.Admission_id) AS Total_Admissions,
274     SUM(b.Billing_Amount) AS Total_Revenue,
275     AVG(c.Length_of_Stay) AS Avg_Stay_Days
276 FROM Admission_tb a
277 INNER JOIN Hospital_tb h ON a.Hospital_id = h.Hospital_id
278 INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id
279 GROUP BY h.Hospital_Name
280 ORDER BY Total_Revenue DESC;
281 select * from vw_total_billing_per_hospital;
```

Hospital_Name	Total_Admissions	Total_Revenue	Avg_Stay_Days
Northwestern Memorial Hospital	14208	317565464.36	18.3264
UI Health (University of Illinois Hospital)	14304	315834802.10	18.0960
Loyola University Medical Center	13908	309145622.79	18.0037
UChicago Medicine	14348	307026138.49	17.4054

Action Output

#	Time	Action	Message	Duration / Fetch
12	15:31:32	SELECT p.Name, b.Billing_Amount FROM Patient_tb p INNER JOIN Admission_tb a ON p... 10 row(s) returned	10 row(s) returned	0.047 sec / 0.000 sec
13	15:32:24	SELECT p.Name, b.Billing_Amount, RANK() OVER (ORDER BY b.Billing_Amount DESC)... 56768 row(s) returned	56768 row(s) returned	0.421 sec / 0.094 sec
14	15:33:22	CREATE VIEW vw_monthly_admission_trends AS SELECT DATE_FORMAT(a.Date_of_Admi... Error Code: 1050. Table 'vw_monthly_admission_trends' already exists	Error Code: 1050. Table 'vw_monthly_admission_trends' already exists	0.031 sec
15	15:33:28	select * from vw_monthly_admission_trends LIMIT 0, 10	10 row(s) returned	0.359 sec / 0.000 sec
16	15:34:53	CREATE VIEW vw_total_billing_per_hospital AS SELECT h.Hospital_Name, COUNT(a.Ad... Error Code: 1050. Table 'vw_total_billing_per_hospital' already exists	Error Code: 1050. Table 'vw_total_billing_per_hospital' already exists	0.015 sec
17	15:34:57	select * from vw_total_billing_per_hospital LIMIT 0, 10	4 row(s) returned	0.578 sec / 0.000 sec

FIG 12

2.2 View 2

Task 2 or view 2 contain monthly admission trend of patient which help us to determine in which month hospital require more staffs and Doctors it help us to understand the peak season and financial stability of hospital.

With this we can monitor the hospital performance.

We can track revenue and financial outcomes.Understand the Patient flow and seasonal demand. All the 3 will make us help to make data driven decision making.
this step enables healthcare organization to operates more effectively and strategically.

The screenshot shows the MySQL Workbench interface. In the top pane, a query editor titled "latest assignment query" displays the SQL code for creating a view named "vw_monthly_admission_trends". The code includes selecting data from "Admission_tb" and "Billing_tb" based on "Admission_id", grouping by month, and calculating total admissions, monthly revenue, and average stay. The bottom pane shows the results of the query, a history of actions, and the status bar indicating the date and time.

```
284 ## VIEW 2 - Trend / Performance View
285 CREATE VIEW vw_monthly_admission_trends AS
286 SELECT
287     DATE_FORMAT(a.Date_of_Admission, '%Y-%m') AS Month,
288     COUNT(a.Admission_id) AS Total_Admissions,
289     SUM(b.Billing_Amount) AS Monthly_Revenue,
290     AVG(a.Length_of_Stay) AS Avg_Stay
291 FROM Admission_tb a
292 INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id
293 GROUP BY DATE_FORMAT(a.Date_of_Admission, '%Y-%m')
294 ORDER BY Month;
295 select * from vw_monthly_admission_trends;
```

Month	Total_Admissions	Monthly_Revenue	Avg_Stay
2019-05	649	14419594.62	18.2712
2019-06	953	19553942.68	17.2267
2019-07	1008	22485054.39	18.5476
2019-08	1048	22104302.95	18.8416
2019-09	998	23622391.80	18.9469
2019-10	1057	22411969.87	17.1069

vw_monthly_admission_trends11 x

Action Output

#	Time	Action	Message	Duration / Fetch
10	15:28:14	SELECT h.Hospital_Name, AVG(a.Length_of_Stay) AS Avg_Stay FROM Hospital_tb h INNER JOIN Admission_tb a ON h.Hospital_ID = a.Hospital_ID GROUP BY h.Hospital_Name	4 row(s) returned	0.282 sec / 0.000 sec
11	15:30:09	SELECT p.Name, b.Billing_Amount, CASE WHEN b.Billing_Amount > 5000 THEN 'High' ELSE 'Low' END AS Category FROM Patient_tb p INNER JOIN Admission_tb a ON p.Patient_ID = a.Patient_ID INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE p.Name LIKE '%John%'	10 row(s) returned	0.000 sec / 0.000 sec
12	15:31:32	SELECT p.Name, b.Billing_Amount FROM Patient_tb p INNER JOIN Admission_tb a ON p.Patient_ID = a.Patient_ID INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE p.Name LIKE '%John%' ORDER BY b.Billing_Amount DESC	10 row(s) returned	0.047 sec / 0.000 sec
13	15:32:24	SELECT p.Name, b.Billing_Amount, RANK() OVER (ORDER BY b.Billing_Amount DESC)	56768 row(s) returned	0.421 sec / 0.094 sec
14	15:33:22	CREATE VIEW vw_monthly_admission_trends AS SELECT DATE_FORMAT(a.Date_of_Admission, '%Y-%m') AS Month, COUNT(a.Admission_id) AS Total_Admissions, SUM(b.Billing_Amount) AS Monthly_Revenue, AVG(a.Length_of_Stay) AS Avg_Stay FROM Admission_tb a INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id GROUP BY DATE_FORMAT(a.Date_of_Admission, '%Y-%m')	Error Code: 1050. Table 'vw_monthly_admission_trends' already exists	0.031 sec
15	15:33:28	select * from vw_monthly_admission_trends LIMIT 0, 10	10 row(s) returned	0.359 sec / 0.000 sec

FIG 13

5 Task 3

1. Query: Identifying Doctors Who Generate the Highest Revenue.

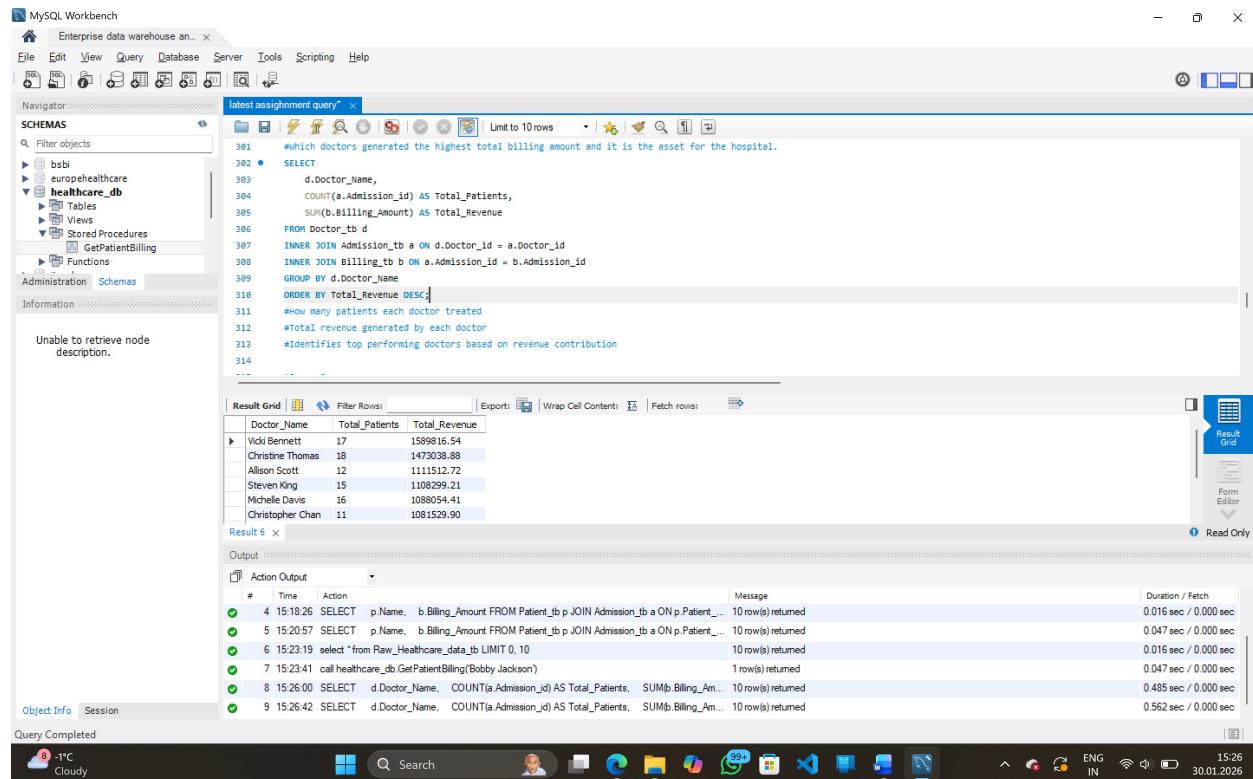
This query evaluates the doctors who generated the highest revenue and treated the patient. It is ordered in descending order so the highest the revenue top one is the doctor and it can limit to the top 10 performing doctors.

The top 3 are :-

Vicki Bennett who treated 17 patient with total revenue is 1589816.54

Christine Thomas who treated 18 patient with total revenue is 1473038.88

Alison Scott who treated 12 patient with total revenue is 1111512.72



The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 Which doctors generated the highest total billing amount and it is the asset for the hospital.
2
3 SELECT
4     d.Doctor_Name,
5     COUNT(a.Admission_id) AS Total_Patients,
6     SUM(b.Billing_amount) AS Total_Revenue
7
8     FROM doctor_tb d
9     INNER JOIN Admission_tb a ON d.Doctor_Id = a.Doctor_id
10    INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id
11    GROUP BY d.Doctor_Name
12    ORDER BY Total_Revenue DESC;
13
14    #How many patients each doctor treated
15    #Total revenue generated by each doctor
16    #Identifies top performing doctors based on revenue contribution
```

The Result Grid shows the following data:

Doctor_Name	Total_Patients	Total_Revenue
Vicki Bennett	17	1589816.54
Christine Thomas	18	1473038.88
Alison Scott	12	1111512.72
Steven King	15	1108299.21
Michele Davis	16	1088054.41
Christopher Chan	11	1081529.90

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	15:18:26	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id WHERE a.Doctor_id = 1	10 row(s) returned	0.016 sec / 0.000 sec
5	15:20:57	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id WHERE a.Doctor_id = 2	10 row(s) returned	0.047 sec / 0.000 sec
6	15:23:19	select * from Raw_Healthcare_data_tb LIMIT 0, 10	10 row(s) returned	0.016 sec / 0.000 sec
7	15:23:41	call healthcare_db.GetPatientBilling('Bobby Jackson')	1 row(s) returned	0.047 sec / 0.000 sec
8	15:26:00	SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Am... 10 row(s) returned		0.485 sec / 0.000 sec
9	15:26:42	SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Am... 10 row(s) returned		0.562 sec / 0.000 sec

FIG 14

These are the top 3 performing doctors who created a big revenue for the hospital.

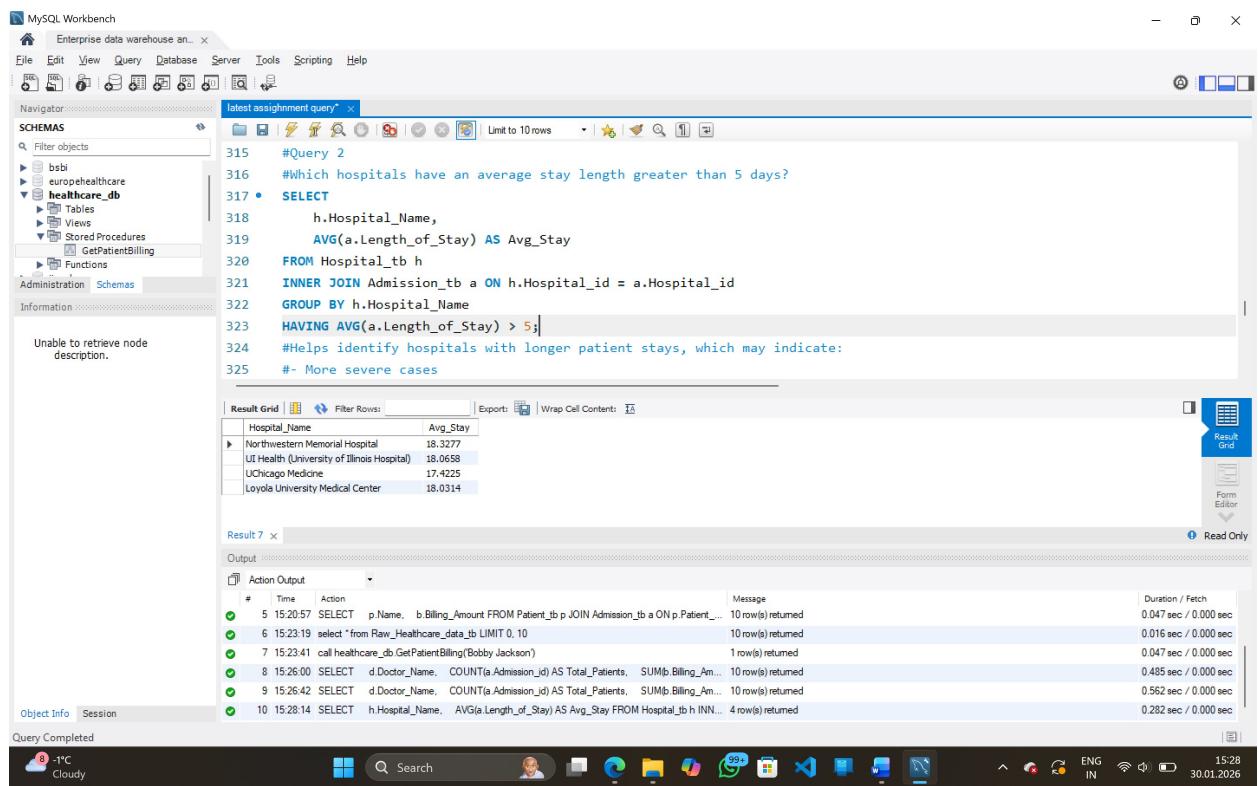
Management can use these insight for top performing doctors, who handled the complex case and contribute significantly to hospital contribution.

2. Query Hospitals With an Average Length of Stay Above 5 Days.

The query tell us that in which hospital is the grater length of stay. However the length of stay is greater the revenue for the hospital is big.

with these query we can see in which all hospital patient frequency of loger stay is more to indentified the top performing hospital.

There are total 4 hospitals whose length of stay is more than average 17 - 18 days. These are the names of hospitals.



The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```

315  #Query 2
316  #Which hospitals have an average stay length greater than 5 days?
317  • SELECT
318      h.Hospital_Name,
319      AVG(a.Length_of_Stay) AS Avg_Stay
320  FROM Hospital_tb h
321  INNER JOIN Admission_tb a ON h.Hospital_id = a.Hospital_id
322  GROUP BY h.Hospital_Name
323  HAVING AVG(a.Length_of_Stay) > 5
324  #Helps identify hospitals with longer patient stays, which may indicate:
325  #- More severe cases

```

The result grid shows:

Hospital_Name	Avg_Stay
Northwestern Memorial Hospital	18.3277
UI Health (University of Illinois Hospital)	18.0658
UChicago Medicine	17.4225
Loyola University Medical Center	18.0314

The output pane shows the execution history:

#	Time	Action	Message	Duration / Fetch
5	15:20:57	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id WHERE a.Hospital_id = 1	10 row(s) returned	0.047 sec / 0.000 sec
6	15:23:19	select * from Raw_Healthcare_data_tb LIMIT 0, 10	10 row(s) returned	0.016 sec / 0.000 sec
7	15:23:41	call healthcare_db.GetPatientBilling('Bobby Jackson')	1 row(s) returned	0.047 sec / 0.000 sec
8	15:26:00	SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Amount) AS Total_Billing FROM Doctor_tb d JOIN Patient_tb p ON d.Doctor_id = p.Doctor_id JOIN Admission_tb a ON p.Patient_id = a.Patient_id JOIN Hospital_tb h ON a.Hospital_id = h.Hospital_id WHERE d.Doctor_id = 1	10 row(s) returned	0.485 sec / 0.000 sec
9	15:26:42	SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Amount) AS Total_Billing FROM Doctor_tb d JOIN Patient_tb p ON d.Doctor_id = p.Doctor_id JOIN Admission_tb a ON p.Patient_id = a.Patient_id JOIN Hospital_tb h ON a.Hospital_id = h.Hospital_id WHERE d.Doctor_id = 2	10 row(s) returned	0.562 sec / 0.000 sec
10	15:28:14	SELECT h.Hospital_Name, AVG(a.Length_of_Stay) AS Avg_Stay FROM Hospital_tb h INNER JOIN Admission_tb a ON h.Hospital_id = a.Hospital_id WHERE h.Hospital_Name != 'Northwestern Memorial Hospital'	4 row(s) returned	0.282 sec / 0.000 sec

FIG 15

Northwestern Memorial Hospital – 18 days.

UI Health (University of Illinois Hospital – 18 days.

UChicago Medicine – 17 days.

Loyola university Medical Center – 18 days

3. Query Categorising Patients Based on Billing Amount.

In these Query you can see I have done Categorical division on billing amount. I have categorized in 3 division higher amount, medium amount, lower billing amount. whose Billing Amount is greater than 5000 are considered as higher billing bracket and we can give some schemes, Discount coupans or we can tell them to take our Membership card. So in future they choose our hospital for treatments.

For Medium bracket people or middleclass people we can tell them to take our life insurance policy so that they don't need to go anywhere whatever the treatment would we will do and give them a value package so we can make a customer segmentaionand tell them in future to tell their neighbours , relatives friends to take the policy.

through these we can get a large group of people who will come to us.

we wil, get Mothly, Quaterly, and Yearly premium amount which can boost our financial staus as well we will a new segment of crowd and we can handle them with our benefits.

Lower middle class peoplewhere I don't have any schemes but in future we segregate these patient so that we can find out our potential clients and non Potential clients.

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

- Schemas:** healthcare_db
- Query Editor:** latest assignment query
- Query Text:**

```

129  #Recognize patients based on billing amount.
130
131  SELECT
132      p.Name,
133      b.Billing_Amount,
134
135      CASE
136          WHEN b.Billing_Amount > 5000 THEN 'High Billing'
137          WHEN b.Billing_Amount BETWEEN 2000 AND 5000 THEN 'Medium Billing'
138          ELSE 'Low Billing'
139      END AS Billing_Catagory
140
141      FROM Patient_tb P
142      INNER JOIN Admission_tb A ON P.Patient_Id = A.Patient_Id
143      INNER JOIN Billing_tb B ON A.Admission_Id = B.Admission_Id;
144
145  #This classifies patients into High, Medium, or Low Billing categories.
146  #Identifying highest cost patients
147  #Financial segmentation
148
149  #Query 4 - Subquery (Correlated)

```
- Result Grid:**

Name	Billing_Amount	Billing_Catagory
Bobby Jackson	2212.27	Medium Billing
Leslie Terry	3185.16	Medium Billing
Danny Smith	72055.21	High Billing
Andrew Watts	4092.60	Medium Billing
Adrienne Bell	4985.66	High Billing
Emily Johnson	37140.37	High Billing
- Output:**

Action	Time	Message	Duration / Fetch
select * from Raw_Healthcare_data_tb LIMIT 0, 10	15:23:19	10 row(s) returned	0.016 sec / 0.000 sec
call healthcare_db.GetPatientBilling('Bobby Jackson')	15:23:41	1 row(s) returned	0.047 sec / 0.000 sec
SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Amount) AS Avg_Billing_Amount FROM Patient_tb P JOIN Admission_tb A ON P.Patient_Id = A.Patient_Id JOIN Billing_tb B ON A.Admission_Id = B.Admission_Id GROUP BY d.Doctor_Name	15:26:00	10 row(s) returned	0.485 sec / 0.000 sec
SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Amount) AS Avg_Billing_Amount FROM Patient_tb P JOIN Admission_tb A ON P.Patient_Id = A.Patient_Id JOIN Billing_tb B ON A.Admission_Id = B.Admission_Id WHERE d.Doctor_Name = 'Dr. Smith' GROUP BY d.Doctor_Name	15:26:42	10 row(s) returned	0.562 sec / 0.000 sec
SELECT h.Hospital_Name, AVG(a.Length_of_Stay) AS Avg_Stay FROM Hospital_tb H JOIN Admission_tb A ON H.Hospital_Id = A.Hospital_Id GROUP BY h.Hospital_Name	15:28:14	4 row(s) returned	0.282 sec / 0.000 sec
SELECT p.Name, b.Billing_Amount, CASE WHEN b.Billing_Amount > 5000 THEN 'High Billing' WHEN b.Billing_Amount BETWEEN 2000 AND 5000 THEN 'Medium Billing' ELSE 'Low Billing' END AS Billing_Catagory FROM Patient_tb P JOIN Admission_tb A ON P.Patient_Id = A.Patient_Id JOIN Billing_tb B ON A.Admission_Id = B.Admission_Id	15:30:09	10 row(s) returned	0.000 sec / 0.000 sec

FIG 16

4. Query Identifying Patients with Billing above the Overall Average.

In these query we can segregate the overall average billing amount patient. It is segregate according to the average billing amount across all patient. These help us for financial forecasting , risk assessment and insurance cordinance. Through these we can plan hospital budget and negotiate the insurance reimbursement and monitor financial potential risks.

```

MySQL Workbench - Enterprise data warehouse an...
File Edit View Query Database Server Tools Scripting Help
Navigator: latest assignment query*
SCHEMAS
Filter objects
bebi
eurohealthcare
healthcare_db
Tables
Views
Stored Procedures
GetPatientBilling
Functions
Administration Schemas
Information
Unable to retrieve node description.
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows: | Result Grid
Result 9 ×
Output: Action Output
# Time Action Message Duration / Fetch
1 7 15:23:41 call healthcare_db.GetPatientBilling('Bobby Jackson') 1 row(s) returned 0.047 sec / 0.000 sec
2 8 15:26:00 SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Amount) AS Avg_Billing_Amount FROM Patient_tb p INNER JOIN Admission_tb a ON p.Patient_id = a.Patient_id INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > (SELECT AVG(Billing_Amount) FROM Billing_tb)
3 9 15:26:42 SELECT d.Doctor_Name, COUNT(a.Admission_id) AS Total_Patients, SUM(b.Billing_Amount) AS Avg_Billing_Amount FROM Patient_tb p INNER JOIN Admission_tb a ON p.Patient_id = a.Patient_id INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > (SELECT AVG(Billing_Amount) FROM Billing_tb)
4 10 15:28:14 SELECT h.Hospital_Name, AVG(a.Length_of_Stay) AS Avg_Stay FROM Hospital_tb h INNER JOIN Admission_tb a ON h.Hospital_id = a.Hospital_id GROUP BY h.Hospital_Name
5 11 15:30:09 SELECT p.Name, b.Billing_Amount, CASE WHEN b.Billing_Amount > 5000 THEN 'High Risk' ELSE 'Normal' END AS Risk_Level FROM Patient_tb p INNER JOIN Admission_tb a ON p.Patient_id = a.Patient_id INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id
6 12 15:31:32 SELECT p.Name, b.Billing_Amount FROM Patient_tb p INNER JOIN Admission_tb a ON p.Patient_id = a.Patient_id INNER JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > (SELECT AVG(Billing_Amount) FROM Billing_tb)

```

FIG 17

5. Query: Ranking Patients by Billing Amount (Window Function).

Based on their billing amount, patients are ranked from highest to lowest in this query. Ranking makes it easy to see which patients bring in the most money for the hospital. Understanding revenue concentration.

We can manage VIP patients, and think strategically for financial year all benefit from this knowledge. This knowledge can help hospitals get good.

FIG 18

6. Stored Procedure: Calculating Total Billing for a Specific Patient

The process of figuring out the final billing amount for a particular patient is required directly by administrative staff.

Then with the help of stored procedure we can get it directly. Administrative personnel that often need to get billing summaries for insurance claims, patient inquiries, or financial audits will find this helpful. Financial billing gets accelerated, accuracy is increased, and human work or administrative staff work is decreased when this activity is automated.

```

1 • call healthcare_db.GetPatientBilling('Bobby Jackson');
2

```

Name	Total_Billing
Bobby Jackson	2212.2

Action Output

#	Time	Action	Message	Duration / Fetch
1				
2	15:18:12	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_ID = a.Patient_ID WHERE a.Admission_ID = 10; 10 row(s) returned	0 row(s) affected	0.062 sec / 0.000 sec
3	15:18:22	SET @avg_bill := (SELECT AVG(Billing_Amount) FROM Billing_tb)	0 row(s) affected	0.156 sec
4	15:18:26	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_ID = a.Patient_ID WHERE a.Admission_ID = 10; 10 row(s) returned	0 row(s) affected	0.016 sec / 0.000 sec
5	15:20:57	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_ID = a.Patient_ID WHERE a.Admission_ID = 10; 10 row(s) returned	0 row(s) affected	0.047 sec / 0.000 sec
6	15:23:19	select * from Raw_Healthcare_data_tb LIMIT 0, 10	10 row(s) returned	0.015 sec / 0.000 sec
7	15:23:41	call healthcare_db.GetPatientBilling('Bobby Jackson')	1 row(s) returned	0.047 sec / 0.000 sec

FIG 19

Task 4



Cap Theorem Application in Distributed Systems:

1) Components of the CAP Theorem.

Consistency:- It ensures all nodes in the Healthcare database system ensures that all nodes return the same information all over and in a consistent manner not a single data can't be inconsistent. It applies to all tables such as patient_tb, billing_tb, Prescription_tb, Doctor_tb etc. Where Clinical, billing etc must remain accurate all over the replicas.

Availability:- It ensures that all data present and available after every request made. Even if some nodes are down then also it should be available. This means that if there is any emergency the patient record, lab result, medication, billing etc all this information should be available and executed after every request is made by hospital elite staff.

Partition Tolerance:- in this when there is crash of servers, routers failures, regional connectivity problems etc happen then also it should able to operate when nodes are disturbed. In the Healthcare sector partition always happen as different Hospitals are located to different places some primary Healthcare sector secondary Healthcare sector are located to different areas and some are in villages where the systems break, power cuts, server crash happen but partition tolerance over it still work and after connection it updates the data which was lost due to the server failure connection.

this crucial in Healthcare where downtime can directly affects the patient safety, clinical errors and diagnostic reports so partition tolerance helps and ensure it.

2) Why no distributed Database can guarantee all three simultaneously.

In any distributed system either of 3 consistency, Availability and Partition Tolerance only 2 will show as in my domain is Healthcare so for me consistency and Partition tolerance is important and I sacrificed Availability.

Consistency:- Stop serving some Request until all Nodes are connected or arranged .

Availability: - Keep serving request even if some data is outdated.

Partition tolerance:- The system must keep running despite of network failures. It becomes impossible to get both Consistency and Availability at the same time.

3) Applying CAP to your Healthcare Database System.

I have chosen Healthcare database domain, where the data stores critical information like 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 etc.

because of these data it directly affects the patient safety.

As Doctors, Nurses should get correct information as we know nurses in hospitals are directed by doctors to give timely the medicine check other vitals organs etc that's why in my field or I would say in Healthcare domain Consistency and Partition Tolerance (CP) I have prioritised and sacrificed the availability (A).

REASONS To choose CP:-

CONSISTENCY ?:- as outdated information can't work it should be consistent, as the doctors nurses Lab reports etc are important to see by the doctor so it is important to have consistent data for Patient safety.

PARTITION TOLERANCE:- As we know hospitals are there in cities as well as in villages and at least primary, Secondary or tertiary Hospitals in our Cities, villages across all over the country which are located in Various location where sometime Network issues, servers crashed, light source or any other kind of problem which disturb the network and cause failure for that moment. At that time system should function and work even if some nodes cannot communicate at that time and update it when servers restore it, because of it there will be no errors and zero patient risk.

Why I have sacrificed the Availability?

During a network failure happens the server must temporarily block the access rather showing with an incomplete or inconsistent data or back dated information which can lead to Patient risk. for eg:- a patient was having an operation of Angioplasty and during that time servers got crashed due to some technical issues and if availability is there then same name of two patients are there one has done its Angiography and waiting for angioplasty at that time person A with angiography get replaced with Patient B who is waiting for angioplasty and doctors put the stent without knowing that he/ she has to go through angiography before angioplasty at that time risk is there of death if person A has to put 2 stents. What will doctor do he will put the stent and as heart has 3 arteries from there both side part of heart has to get stent where blockage is there and reduce the risk of Heart Failure but patient which doctor don't know at which location does the stent has to be placed if he placed at wrong location and later after operation patient died it will be sole responsibility of doctor but who was wrong over here no one .

With that wrong information patient risk gets higher, thus why I sacrificed the availability. and I choose the Consistency and Partitioned Tolerance. Delayed data And Consistent data will only work in Healthcare System.

4) Reflection: How CAP theorem influences Healthcare Database Design.

The cap theorem affects how distributed databases are built. This forces designers to choose which two properties are most important.

This influences the system.

Design:- The system must be consistent or available when failures/ Crash etc happens.

Scaling:- It affects scaling because when new hospitals are added in different locations, network partition becomes more important and common and must be planned according to it.

Resilience:- the system needs recovery updates and backup to keep working even when network fails or due to any condition you are not connected.

Task 5



Query Performance and Optimization:-

I think this query need optimization as system required to scan many rows every time for calculation so this make me to optimize the the query as patient's billing amount with the overall average billing amount. This query took longer time and had no index.

To improve the performance I used two methods.

The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The main area displays a query titled "latest assignment query" with the following code:

```
398  #
399  # query which is need to be optimized.
400  SELECT
401      p.Name,
402      b.Billing_Amount
403  FROM Patient_tb p
404  JOIN Admission_tb a ON p.Patient_id = a.Patient_id
405  JOIN Billing_tb b ON a.Admission_id = b.Admission_id
406  WHERE b.Billing_Amount > (SELECT AVG(Billing_Amount) FROM Billing_tb);
407
```

The results grid shows the following data:

Name	Billing_Amount
Victoria Reyes	22016.54
Ryan Powell	22022.22
Jennifer Barnett	22025.80
Brian Ramirez	22034.20
Joseph Cameron	22038.30
William Gardner	22042.06
James Thompson	22058.95

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	15:13:09	use Healthcare_db	0 row(s) affected	0.000 sec
2	15:18:12	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > (SELECT AVG(Billing_Amount) FROM Billing_tb);	10 row(s) returned	0.062 sec / 0.000 sec
3	15:18:22	SET @avg_bill = (SELECT AVG(Billing_Amount) FROM Billing_tb)	0 row(s) affected	0.156 sec
4	15:18:26	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > @avg_bill;	10 row(s) returned	0.016 sec / 0.000 sec
5	15:20:57	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > @avg_bill;	10 row(s) returned	0.047 sec / 0.000 sec

FIG 20

Task 1:- The optimization task 1 is adding indexing in patient_id and billing amount. Index are like shortcuts as they help database find information without checking row.

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The main area has a 'latest assignment query' tab open. The left sidebar shows the Navigator with 'SCHEMAS' expanded, revealing 'babi', 'eurohealthcare', and 'healthcare_db'. Under 'Tables', there are 'Patient_tb', 'Admission_tb', and 'Billing_tb'. The right pane contains the query code for 'Task 5'.

```

396
397
398     #
399     # query which is need to be optimised.
400
401     SELECT
402         p.Name,
403         b.Billing_Amount
404     FROM Patient_tb p
405     JOIN Admission_tb a ON p.Patient_id = a.Patient_id
406     JOIN Billing_tb b ON a.Admission_id = b.Admission_id
407     WHERE b.Billing_Amount > (SELECT AVG(Billing_Amount) FROM Billing_tb);
408
409     # 1st optomisation
410     • CREATE INDEX idx_billing_amount ON Billing_tb(Billing_Amount);
411     • CREATE INDEX idx_patient ON Admission_tb(Patient_id);
412
413     # 2nd optimisation
414     • SET @avg_bill = (SELECT AVG(Billing_Amount) FROM Billing_tb);
415
416     • SELECT

```

The 'Output' section shows the execution log:

Action	Time	Action	Message	Duration / Fetch
use Healthcare_db	15:13:09		0 row(s) affected	0.000 sec
SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > (SELECT AVG(Billing_Amount) FROM Billing_tb);	15:18:12		10 row(s) returned	0.062 sec / 0.000 sec
SET @avg_bill = (SELECT AVG(Billing_Amount) FROM Billing_tb);	15:18:22		0 row(s) affected	0.156 sec
SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_id = a.Patient_id JOIN Billing_tb b ON a.Admission_id = b.Admission_id WHERE b.Billing_Amount > @avg_bill;	15:18:26		10 row(s) returned	0.016 sec / 0.000 sec

The bottom status bar shows the system tray, taskbar, and system information like temperature (1°C), battery level, and date/time (30.01.2026).

FIG 21

Task 2:- Second, I rewrote the query so that the average billing amount was calculated only once instead of repeatedly, which reduced unnecessary work for the system.

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the schema 'healthcare_db' under 'Schemas'. The main area contains a query editor titled 'latest assignment query*' with the following SQL code:

```

411
412
413    # 2nd optimization
414  SET @avg_bill = (SELECT AVG(Billing_Amount) FROM Billing_tb);
415
416  SELECT
417      p.Name,
418      b.Billing_Amount
419  FROM Patient_tb p
420  JOIN Admission_tb a ON p.Patient_Id = a.Patient_Id
421  JOIN Billing_tb b ON a.Admission_Id = b.Admission_Id
422  WHERE b.Billing_Amount > @avg_bill;
423

```

Below the query editor is a 'Result Grid' showing a list of patients with their names and billing amounts. The results are as follows:

Name	Billing_Amount
Victoria Reyes	22015.54
Ryan Powell	22032.22
Jennifer Barnett	22035.80
Brian Ramirez	22034.20
Joseph Cameron	22038.30
William Gardner	22042.06
James Thompson	22038.95

At the bottom of the interface, the 'Output' section shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	15:13:09	use Healthcare_db	0 row(s) affected	0.000 sec
2	15:18:12	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_Id = a.Patient_Id JOIN Billing_tb b ON a.Admission_Id = b.Admission_Id WHERE b.Billing_Amount > @avg_bill;	10 row(s) returned	0.062 sec / 0.000 sec
3	15:18:22	SET @avg_bill = (SELECT AVG(Billing_Amount) FROM Billing_tb);	0 row(s) affected	0.156 sec
4	15:18:26	SELECT p.Name, b.Billing_Amount FROM Patient_tb p JOIN Admission_tb a ON p.Patient_Id = a.Patient_Id JOIN Billing_tb b ON a.Admission_Id = b.Admission_Id WHERE b.Billing_Amount > @avg_bill;	10 row(s) returned	0.016 sec / 0.000 sec

FIG 22

After applying these optimization the query was running much faster because database did less scanning and few repeated calculation and it saves time reduce the server load and ensures that important info like patient and billing data can be accessible quickly.

6 Bibliography.

Licea, E. (2022) *Healthcare Dataset*. Kaggle. Available at:

<https://www.kaggle.com/datasets/eduardolicea/healthcare-dataset> (Accessed: 30 January 2026).

Connolly, T. and Begg, C. (2015) *Database System: A Practical Approach to Design, Implementation, and Management*. 6th edn. Harlow: Pearson.

Elmasri, R. and Navathe, S.B. (2016) *Fundamentals of Database Systems*. 7th edn. Boston, MA: Pearson.

Oracle (2024) *MySQL – The world's most popular open-system of database*. Available at:

<https://www.mysql.com/de/> (Accessed: 11 November 2025).

7 APPENDIX (if necessary)

All Sql scripts Raw data theory etc are uploaded on the git hub repository

jinu5360 (2026) *hospital-chain-database-3nf*. GitHub. Available at: <https://github.com/jinu5360/hospital-chain-database-3nf> (Accessed: 30 January 2026).

GitHub Repository:

<https://github.com/jinu5360/hospital-chain-database-3nf>