

# **FINAL DATABASE PROJECT**

Members:

Yash Revadekar : 002738776

Krutik Kanakia : 002787847

Ikra Bagwan : 002794307

## **Project Description:**

An integrated system comprising numerous entities makes up the hospital management system. The management of all these elements would be difficult. We are therefore creating a single database for hospitals to address this issue and improve system efficiency by maintaining all the information needed to administer hospitals. Additionally, with the aid of this project, we will use visualization tools to exhibit and convey the data for prevalent trends for the breakouts. This would make it easier to comprehend analytics based on data or information collected from several hospitals.

### **Hospital Admin Table:**

We manually input the data for the hospital admin table. We have 4 columns in our table, and using Jupyter, we were able to read the hospital admin's csv file and remove all of the null values. We have verified the accuracy of the data for this table by displaying all the rows with null values. After finishing everything, we entered this information into our database.

### **Doctor Table:**

We used python programs to web scrape the data for the doctor table from the Browse AI website. Our table has eight columns, and using Jupyter, we were able to read the doctor's csv file and remove all of the null entries. By grouping by one column (primary spec) and count (doc id), we were able to see the data. Next, we used matplotlib to plot the following bar graph. Finally, we group by primary spec once more, got the mean for the age, and plot the graph for age. After finishing everything, we entered this information into our database.

### **Patient Table:**

We manually input the data for the patients table. We have 13 columns in our table, and using Jupyter, we were able to read the patient's csv file and remove all of the null values. We have verified the accuracy of the data for this table by displaying all the rows with null values. By converting the admit date into a month and year using the date-time format, we then grouped the data by monthyear, counted the patient IDs, and generated a line graph. After finishing everything, we entered this information into our database.

### **Payment Table:**

The management of all patient and hospital financial information would fall under the purview of this module. All patient payment information and financial records for hospital expenses are stored and shown.

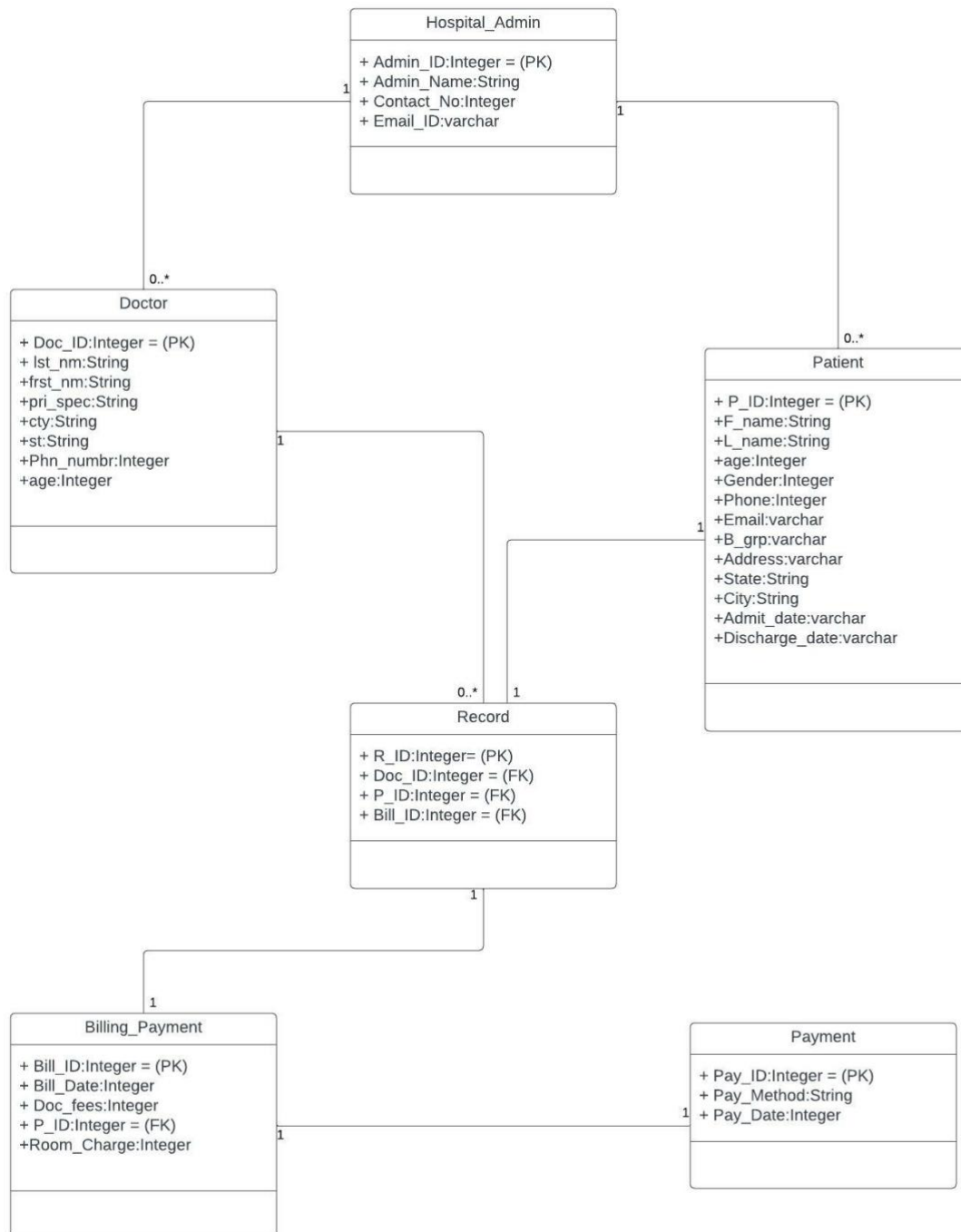
#### Bill Payment Table:

We manually entered the data for the bill payment table. Our database has five columns, and using Jupyter, we were able to read the bill payment csv file and remove all of the null entries. When grouping by monthyear and taking the count of the room charge, we turned the bill date into a month year by date-time before plotting the line graph. After finishing everything, we entered this information into our database.

#### Record Table:

We manually entered the data into the Record table. Our database has three columns, and using Jupyter, we were able to read the payment's csv file and remove all of the null entries. After finishing everything, we entered this information into our database.

## ER Diagram:



### Snippets From Database:

1) Hospital Admin Table:

```
CREATE TABLE `admin` (  
  `Admin_ID` int NOT NULL,  
  `Admin_Name` varchar(255) DEFAULT NULL,  
  `Contact_No` varchar(255) DEFAULT NULL,  
  `Email_ID` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`Admin_ID`)  
)
```

Result Grid		Filter Rows:		Edit:
	Admin_ID	Admin_Name	Contact_No	Email_ID
▶	1	David	8578456321	david1@gmail.com
	2	Darshan	8541253987	darshan23@gmail.com
	3	Nick	8745213698	nickb@gmail.com
	4	Joe	7896541236	joe2@gmail.com
	5	Kapil	8574563254	kapil@gmail.com
	6	Gayle	7845214639	gayle3@gmail.com
	7	Otis	8547896523	otis@gmail.com
	8	Olson	7135486254	olson4@gmail.com

## 2) Doctor Table:

```
CREATE TABLE `doctor` (  
  `Doc_ID` int NOT NULL,  
  `lst_nm` varchar(255) DEFAULT NULL,  
  `frst_nm` varchar(255) DEFAULT NULL,  
  `pri_spec` varchar(255) DEFAULT NULL,  
  `cty` varchar(255) DEFAULT NULL,  
  `st` varchar(255) DEFAULT NULL,  
  `phn_numbr` varchar(255) DEFAULT NULL,  
  `age` int DEFAULT NULL,  
  PRIMARY KEY (`Doc_ID`)  
)
```

Result Grid		Filter Rows:		Edit:		Export/Import:		Wrap Cell Content:	
	Doc_ID	lst_nm	frst_nm	pri_spec	cty	st	phn_numbr	age	
▶	1407160575	KERR	BEATRICE	CLINICAL SOCIAL WORKER	FREDERICKSBURG	VA	5403611556.0	37	
	1407164437	KORENCHEN	PATRICIA	CLINICAL SOCIAL WORKER	ALBUQUERQUE	NM	5053858496.0	34	
	1407172521	WONG	ANDREW	NEUROLOGY	SAN ANTONIO	TX	8751452635.0	71	
	1407182421	MOSLEY	STEPHEN	PHYSICIAN ASSISTANT	SANTA ROSA BEACH	FL	8751452635.0	48	
	1407230097	BROCK	SONYA	CLINICAL SOCIAL WORKER	BARRE	MA	2149126784.0	34	
	1407234792	BRADBURY	MATTHEW	PSYCHIATRY	TEMPLE	TX	8751452635.0	39	
	1407243538	LU	JOSEPH	NEUROLOGY	SAN GABRIEL	CA	6264580181.0	28	



### 3) Patient Table:

```
CREATE TABLE `patient` (
  `P_ID` int NOT NULL,
  `F_name` varchar(255) DEFAULT NULL,
  `L_name` varchar(255) DEFAULT NULL,
  `age` varchar(255) DEFAULT NULL,
  `Gender` varchar(255) DEFAULT NULL,
  `Phone` varchar(255) DEFAULT NULL,
  `Email` varchar(255) DEFAULT NULL,
  `B_Grp` varchar(255) DEFAULT NULL,
  `Address` varchar(255) DEFAULT NULL,
  `State` varchar(255) DEFAULT NULL,
  `City` varchar(255) DEFAULT NULL,
  `Admit_Date` varchar(255) DEFAULT NULL,
  `Discharge_Date` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`P_ID`)
)
```

P_ID	F_name	L_name	age	Gender	Phone	Email	B_Grp	Address	State	City	Admit_Date	Discharge_Date
1	Chris	Brown	56.0	M	8578452154	cb@gmail.com	A+	56, Washington Street	MA	Boston	11/10/2021	11/25/2021
2	Nick	Jonas	45.0	M	8756984125	nj@gmail.com	B-	45, Lambert Street	TX	Dallas	10/20/2021	11/15/2021
3	Nikki	Brook	67.0	F	8796521453	nb@gmail.com	O+	15, Huntington Ave	FL	Tampa	1/7/2022	1/10/2022
4	Robert	Costa	79.0	M	8763214589	rc@gmail.com	A-	25, Breaking Rocks	MA	Lowell	2/6/2022	2/12/2022
5	Kelly	Mcardle	35.0	F	8796542314	km@gmail.com	A+	88, Center Street	MA	Boston	3/15/2022	3/20/2022
6	James	Johnson	46.0	M	8578963254	jj@gmail.com	B+	726, Brighton Ave	NY	Rochester	11/7/2021	11/15/2021
7	William	Smith	52.0	M	7856452125	ws@hotmail.com	O-	63, Braintree	NV	Las Vegas	9/11/2021	10/5/2021




4) Record Table:

```
CREATE TABLE `record` (
  `R_ID` int NOT NULL,
  `Doc_ID` int DEFAULT NULL,
  `P_ID` int DEFAULT NULL,
  `Bill_ID` int DEFAULT NULL,
  PRIMARY KEY (`R_ID`),
  KEY `Doc_ID` (`Doc_ID`),
  KEY `P_ID` (`P_ID`),
  KEY `Bill_ID` (`Bill_ID`),
  CONSTRAINT `record_ibfk_1` FOREIGN KEY (`Doc_ID`)
REFERENCES `doctor` (`Doc_ID`),
  CONSTRAINT `record_ibfk_2` FOREIGN KEY (`P_ID`)
REFERENCES `patient` (`P_ID`),
  CONSTRAINT `record_ibfk_3` FOREIGN KEY (`Bill_ID`)
REFERENCES `billpayment` (`Bill_ID`)
)
```

Result Grid				Filter Rows:	
	R_ID	Doc_ID	P_ID	Bill_ID	
▶	1	1407160575	1	1	
	2	1407164437	2	2	
	3	1407172521	3	3	
	4	1407182421	4	4	
	5	1407230097	5	5	
	6	1407234792	6	6	
	7	1407243538	7	7	
	8	1407245285	8	8	
	9	1407254881	9	9	
	10	1407256159	10	10	



5) Billing Payment Table:

```
CREATE TABLE `billpayment` (
  `Bill_ID` int NOT NULL,
  `Bill_Date` varchar(255) DEFAULT NULL,
  `P_ID` int DEFAULT NULL,
  `Doc_Fees` varchar(255) DEFAULT NULL,
  `Room_Charge` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`Bill_ID`),
  KEY `P_ID` (`P_ID`),
  CONSTRAINT `billpayment_ibfk_1` FOREIGN KEY (`P_ID`)
REFERENCES `patient` (`P_ID`),
)
```

Result Grid				Filter Rows:		Edit:	
	Bill_ID	Bill_Date	P_ID	Doc_Fees	Room_Charge		
▶	1	11/25/2021	1	50000	10000		
	2	11/15/2021	2	80000	15000		
	3	1/10/2022	3	75000	12000		
	4	2/12/2022	4	45000	8000		
	5	3/20/2022	5	60000	7500		
	6	11/15/2021	6	57500	4500		
	7	10/5/2021	7	56000	8500		
	8	1/10/2022	8	54500	4500		

6) Payment Table:

```
CREATE TABLE `payment` (
  `Pay_ID` int NOT NULL,
  `Pay_Method` varchar(255) DEFAULT NULL,
  `Pay_Date` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`Pay_ID`)
)
```

Result Grid				Filter Rows:
	Pay_ID	Pay_Method	Pay_Date	
	1	Cash	11/28/2021	
	2	Card	11/19/2021	
	3	Cash	1/13/2022	
▶	4	Cheque	2/20/2022	
	5	Card	2/28/2022	
	6	Cash	11/15/2021	
	7	Card	10/15/2021	
	8	Cash	1/12/2022	



## Output of Views:

1) CREATE VIEW `bill\_date\_patients` AS

Select p.F\_name  
,p.L\_name ,(Bill\_Date) as bill\_date  
From patient p  
Left join billpayment bp on  
bp.P\_ID = p.P\_ID Group by 1,2,3

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar. The left sidebar contains a 'SCHEMAS' tree with 'Views' expanded, listing various views including 'bill\_date\_patients'. Below the tree is an 'Information' panel showing the definition of the 'bill\_date\_patients' view:

```
View:
bill_date_patients

Columns:
F_name  varchar(255)
L_name  varchar(255)
bill_date  varchar(255)
```

The main query editor displays the SQL statement: `select * from bill_date_patients;`. The 'Result Grid' at the bottom shows the output of the query, which is a table with three columns: F\_name, L\_name, and bill\_date. The data is as follows:

F_name	L_name	bill_date
Chris	Brown	11/25/2021
Nick	Jonas	11/15/2021
Nikki	Brook	1/10/2022
Robert	Costa	2/12/2022
Kelly	Mcardle	3/20/2022
James	Johnson	11/15/2021
William	Smith	10/5/2021
Robert	Johnson	1/10/2022
John	Smith	3/12/2022
Maria	Gracia	6/20/2022
Mary	Smith	1/25/2021
Maria	Rodriguez	7/5/2021
Michael	Smith	1/5/2022
Maria	Martinez	3/12/2022
Laura	Fernan...	3/20/2022
Tomas	Hughes	11/25/2021

2) CREATE VIEW `avg\_bill\_patient` AS

```
SELECT
    `d`.`pri_spec` AS `pri_spec`,
    `p`.`State` AS `state`,
    AVG(`bp`.`Room_Charge`) AS `doc_fees`
FROM
    (((`patient` `p`
    LEFT JOIN `record` `r` ON ((`r`.`P_ID` = `p`.`P_ID`)))
    LEFT JOIN `doctor` `d` ON ((`d`.`Doc_ID` = `r`.`Doc_ID`)))
    LEFT JOIN `billpayment` `bp` ON ((`bp`.`Bill_ID` =
    `r`.`Bill_ID`)))
GROUP BY `d`.`pri_spec`, `p`.`State`
```

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of database objects, including 'Views' and 'Functions'. The 'avg\_bill\_patient' view is highlighted. Below the tree, the 'View: avg\_bill\_patient' section shows the columns: 'pri\_spec' (varchar(255)), 'state' (varchar(255)), and 'doc\_fees' (double). The main window displays the SQL query for the view: 'select \* from avg\_bill\_patient;'. The 'Result Grid' shows the output of the query, which is a table with three columns: 'pri\_spec', 'state', and 'doc\_fees'. The data rows are as follows:

pri_spec	state	doc_fees
CLINICAL SOCIAL WORKER	MA	8750
CLINICAL SOCIAL WORKER	TX	15000
NEUROLOGY	FL	12000
PHYSICIAN ASSISTANT	MA	8000
PSYCHIATRY	NY	4500
NEUROLOGY	NV	8500
NURSE PRACTITIONER	WA	4500
NURSE PRACTITIONER	IL	6500
CLINICAL SOCIAL WORKER	CA	9500
CLINICAL SOCIAL WORKER	MI	2500
CHIROPRACTIC	MA	3500
OPTOMETRY	CT	7500
CHIROPRACTIC	NY	4500

3) CREATE VIEW `doctor\_patient\_perstate` AS

SELECT

`p`.`State` AS `state`, COUNT(0) AS `count\_of\_doctors`

FROM

(`patient` `p`

LEFT JOIN `record` `r` ON ((`r`.`P\_ID` = `p`.`P\_ID`)))

GROUP BY `p`.`State`

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane lists various database objects, including a view named 'doctor\_patient\_perstate'. The main editor window displays the SQL query: `select * from doctor_patient_perstate;`. Below the editor, the 'Result Grid' shows the output of the query, which is a table with two columns: 'state' and 'count\_of\_doctors'. The data is as follows:

state	count_of_doctors
MA	6
TX	3
FL	3
NY	2
NV	3
WA	2
IL	3
CA	2
MI	2
CT	1
NH	1

Below the result grid, the 'View: doctor\_patient\_perstate' section shows the columns and their data types: 'state' (varchar) and 'count\_of\_doctors' (bigint).

4) CREATE VIEW `highest\_fees` AS

```
SELECT
    `d`.`frst_nm` AS `frst_nm`,
    `d`.`lst_nm` AS `lst_nm`,
    `d`.`st` AS `st`,
    SUM(`bp`.`Doc_Fees`) AS `doc_fees`
FROM
    ((`doctor` `d`
    LEFT JOIN `record` `r` ON ((`r`.`Doc_ID` = `d`.`Doc_ID`)))
    LEFT JOIN `billpayment` `bp` ON ((`bp`.`Bill_ID` =
`r`.`Bill_ID`)))
GROUP BY `d`.`frst_nm` , `d`.`lst_nm` , `d`.`st`
```

The screenshot shows a SQL IDE interface with a Navigator pane on the left, a SQL editor at the top, and a Result Grid at the bottom. The Navigator pane shows a tree of database objects, including Views, Triggers, Stored Procedures, and Functions. The 'highest\_fees' view is highlighted. The SQL editor shows the SQL query for creating the view. The Result Grid shows the output of the query, displaying a list of doctors and their highest fees.

**View: highest\_fees**

**Columns:**

frst_nm	lst_nm	st	doc_fees
BEATRICE	KERR	VA	50000
PATRICIA	KORENCHEN	NM	80000
ANDREW	WONG	TX	75000
STEPHEN	MOSLEY	FL	45000
SONYA	BROCK	MA	60000
MATTHEW	BRADBURY	TX	57500
JOSEPH	LU	CA	56000
STACI	DUNCAN	WA	54500
KESHA	SAUCIER	CA	53000
EDITH	MANN	NY	51500
SAKEENA	MIRZA	CA	50000
PATRICK	HOLLAND	PA	48500
ABRAHAM	AVNILOV	NY	47000

5) CREATE VIEW `highest\_patients` AS

SELECT

`d`.`frst\_nm` AS `frst\_nm`,

`d`.`lst\_nm` AS `lst\_nm`,

`d`.`st` AS `st`,

COUNT(0) AS `count\_of\_patients`

FROM

(`doctor` `d`

LEFT JOIN `record` `r` ON ((`r`.`Doc\_ID` = `d`.`Doc\_ID`)))

GROUP BY `d`.`frst\_nm`, `d`.`lst\_nm`, `d`.`st`

The screenshot shows a database management interface with a 'SCHEMAS' panel on the left and a 'Result Grid' on the right. The 'SCHEMAS' panel lists various database objects, including a view named 'highest\_patients'. The 'Result Grid' displays the output of a SQL query, showing a list of patients with their first name, last name, state, and the count of patients. The query is: `select * from highest_patients;`

frst_nm	lst_nm	st	count_of_patients
BEATRICE	KERR	VA	1
PATRICIA	KORENCHEN	NM	1
ANDREW	WONG	TX	1
STEPHEN	MOSLEY	FL	1
SONYA	BROCK	MA	1
MATTHEW	BRADBURY	TX	1
JOSEPH	LU	CA	1
STACI	DUNCAN	WA	1
KESHA	SAUCIER	CA	1
EDITH	MANN	NY	1
SAKEENA	MIRZA	CA	1
PATRICK	HOLLAND	PA	1
ABRAHAM	AVNILOV	NY	1

```

6) CREATE VIEW `highest_room_charges` AS
SELECT
    `d`.`pri_spec` AS `pri_spec`,
    `d`.`st` AS `st`,
    SUM(`bp`.`Room_Charge`) AS `doc_fees`
FROM
    ((`doctor` `d`
    LEFT JOIN `record` `r` ON ((`r`.`Doc_ID` = `d`.`Doc_ID`)))
    LEFT JOIN `billpayment` `bp` ON ((`bp`.`Bill_ID` = `r`.`Bill_ID`)))
GROUP BY `d`.`pri_spec` , `d`.`st`

```

The screenshot shows a SQL IDE interface with a Navigator pane on the left, a SQL editor at the top, and a Result Grid at the bottom.

**Navigator:** The 'Schemas' tab is active. Under 'Views', the view 'highest\_room\_charges' is selected. Below the view list, the 'View' section shows the details for 'highest\_room\_charges':

- Columns:**
  - `pri_spec` varchar(255)
  - `st` varchar(255)
  - `doc_fees` double

**SQL Editor:** The SQL File 8\* tab contains the following query:

```
1 select * from highest_room_charges;
```

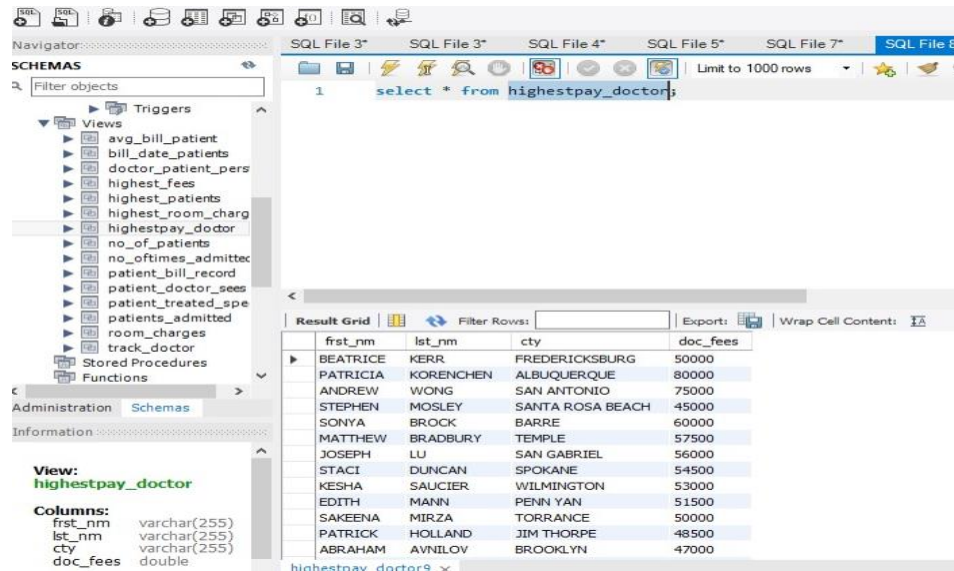
**Result Grid:** The query results are displayed in a table with 3 columns: `pri_spec`, `st`, and `doc_fees`. The results are as follows:

pri_spec	st	doc_fees
CLINICAL SOCIAL WORKER	VA	10000
CLINICAL SOCIAL WORKER	NM	15000
NEUROLOGY	TX	12000
PHYSICIAN ASSISTANT	FL	8000
CLINICAL SOCIAL WORKER	MA	7500
PSYCHIATRY	TX	4500
NEUROLOGY	CA	8500
NURSE PRACTITIONER	WA	4500
NURSE PRACTITIONER	CA	6500
CLINICAL SOCIAL WORKER	NY	9500
CLINICAL SOCIAL WORKER	CA	2500
CHIROPRACTIC	PA	3500
OPTOMETRY	NY	7500

```

7) CREATE VIEW `highestpay_doctor` AS
SELECT
    `d`.`frst_nm` AS `frst_nm`,
    `d`.`lst_nm` AS `lst_nm`,
    `d`.`cty` AS `cty`,
    SUM(`bp`.`Doc_Fees`) AS `doc_fees`
FROM
    ((`doctor` `d`
    LEFT JOIN `record` `r` ON ((`r`.`Doc_ID` = `d`.`Doc_ID`)))
    LEFT JOIN `billpayment` `bp` ON ((`bp`.`Bill_ID` = `r`.`Bill_ID`)))
GROUP BY `d`.`frst_nm` , `d`.`lst_nm` , `d`.`cty`

```



The screenshot shows a database management interface. On the left, the 'SCHEMAS' pane lists various database objects, with 'highestpay\_doctor' selected under the 'Views' category. Below this, the 'View: highestpay\_doctor' section displays its columns: 'frst\_nm' (varchar(255)), 'lst\_nm' (varchar(255)), 'cty' (varchar(255)), and 'doc\_fees' (double). The main window shows the SQL query: 'select \* from highestpay\_doctor;'. Below the query, the 'Result Grid' displays the data returned by the query.

frst_nm	lst_nm	cty	doc_fees
BEATRICE	KERR	FREDERICKSBURG	50000
PATRICIA	KORENCHEN	ALBUQUERQUE	80000
ANDREW	WONG	SAN ANTONIO	75000
STEPHEN	MOSLEY	SANTA ROSA BEACH	45000
SONYA	BROCK	BARRE	60000
MATTHEW	BRADBURY	TEMPLE	57500
JOSEPH	LU	SAN GABRIEL	56000
STACI	DUNCAN	SPOKANE	54500
KESHA	SAUCIER	WILMINGTON	53000
EDITH	MANN	PENN YAN	51500
SAKEENA	MIRZA	TORRANCE	50000
PATRICK	HOLLAND	JIM THORPE	48500
ABRAHAM	AVNILOV	BROOKLYN	47000

```

8) CREATE VIEW `no_of_patients` AS
SELECT
    COUNT(`patient`.`P_ID`) AS `COUNT(P_ID)`,
    `patient`.`City` AS `City`
FROM
    `patient`
GROUP BY `patient`.`City`

```

The screenshot shows a SQL IDE interface with a 'SCHEMAS' panel on the left and a 'Result Grid' on the right. The 'SCHEMAS' panel lists various database objects, including a view named 'no\_of\_patients'. The 'Result Grid' displays the output of a query that selects all data from the 'no\_of\_patients' view.

**View: no\_of\_patients**

**Columns:**

- COUNT(P\_ID) bigint
- City varchar(255)

**Result Grid:**

COUNT(P_ID)	City
3	Boston
1	Dallas
1	Tampa
1	Lowell
1	Rochester
1	Las Vegas
1	Seattle
2	Chicago
1	Los Angeles
1	Jackson
1	Worcester
1	Greenwich
1	Norwich



```

9) CREATE VIEW `patient_bill_record` AS
SELECT
    `record`.`R_ID` AS `R_ID`,
    `billpayment`.`Bill_ID` AS `Bill_ID`
FROM
    (`record`
    JOIN `billpayment` ON ((`record`.`P_ID` = `billpayment`.`P_ID`)))

```

The screenshot shows a database management interface with a 'Schemas' pane on the left and a 'SQL File 3\*' editor on the right. The 'Schemas' pane lists various views, with 'patient\_bill\_record' selected. Below the schema list, the 'View: patient\_bill\_record' is defined with columns 'R\_ID' (int) and 'Bill\_ID' (int). The SQL editor contains the query: `select * from patient_bill_record;`. The 'Result Grid' shows 18 rows of data, with columns 'R\_ID' and 'Bill\_ID'.

R_ID	Bill_ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18

```

10) CREATE VIEW `patient_doctor_sees` AS
SELECT
    `p`.`F_name` AS `F_name`,
    `p`.`L_name` AS `L_name`,
    `p`.`City` AS `City`,
    COUNT(0) AS `count_of_doctors`
FROM
    (`patient` `p`
    LEFT JOIN `record` `r` ON ((`r`.`P_ID` = `p`.`P_ID`)))
GROUP BY `p`.`F_name`, `p`.`L_name`, `p`.`City`

```

The screenshot shows a database management interface with a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar. The 'Schemas' pane on the left lists various database objects, including Views and Triggers. The main window displays a SQL query in a text editor: `select * from patient_doctor_sees;`. Below the editor, the 'Result Grid' shows the output of the query, which is a table with four columns: F\_name, L\_name, City, and count\_of\_doctors. The table contains 20 rows of data, including names like Chris Brown, Nick Jonas, and Robert Costa, along with their cities and a count of 1 for each.

F_name	L_name	City	count_of_doctors
Chris	Brown	Boston	1
Nick	Jonas	Dallas	1
Nikki	Brook	Tampa	1
Robert	Costa	Lowell	1
Kelly	Mcardle	Boston	1
James	Johnson	Rochester	1
William	Smith	Las Vegas	1
Robert	Johnson	Seattle	1
John	Smith	Chicago	1
Maria	Gracia	Los Angeles	1
Mary	Smith	Jackson	1
Maria	Rodriguez	Worcester	1
Michael	Smith	Greenwich	1
Maria	Martinez	Norwich	1
Laura	Fernan...	Virginia City	1
Tomas	Hughes	Boston	1
Andrew	Symonds	Houston	1
Joe	Root	Miami	1

```

11) CREATE VIEW `patient_treated_speciality` AS
SELECT
    `d`.`pri_spec` AS `pri_spec`,
    COUNT(0) AS `count_of_patients`
FROM
    (`doctor` `d`
    LEFT JOIN `record` `r` ON ((`r`.`Doc_ID` = `d`.`Doc_ID`)))
GROUP BY `d`.`pri_spec`

```

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree of database objects, including 'Views' and 'Functions'. The 'patient\_treated\_speciality' view is selected. The main pane shows the SQL query: `select * from patient_treated_speciality;`. Below the query, the 'Result Grid' displays the results of the query, showing two columns: 'pri\_spec' and 'count\_of\_patients'.

pri_spec	count_of_patients
CLINICAL SOCIAL WORKER	5
NEUROLOGY	2
PHYSICIAN ASSISTANT	1
PSYCHIATRY	1
NURSE PRACTITIONER	3
CHIROPRACTIC	3
OPTOMETRY	1
CLINICAL PSYCHOLOGIST	3
INTERNAL MEDICINE	2
FAMILY PRACTICE	3
PHYSICAL THERAPY	1
OTOLARYNGOLOGY	1
PHYSICAL MEDICINE AND ...	1
PODIATRY	1

```

12) CREATE VIEW `patients_admitted` AS
SELECT
    COUNT(`patient`.`P_ID`) AS `COUNT(P_ID)`,
    `patient`.`Admit_Date` AS `Admit_Date`
FROM
    `patient`
GROUP BY `patient`.`Admit_Date`

```

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar. The left sidebar contains a 'SCHEMAS' tree with 'Triggers', 'Views', 'Stored Procedures', and 'Functions'. The 'Views' folder is expanded, showing a list of views including 'patients\_admitted'. The main editor displays the SQL query: `select * from patients_admitted;`. Below the editor, the 'Result Grid' shows the output of the query, which is a table with two columns: 'COUNT(P\_ID)' and 'Admit\_Date'. The table contains 15 rows of data.

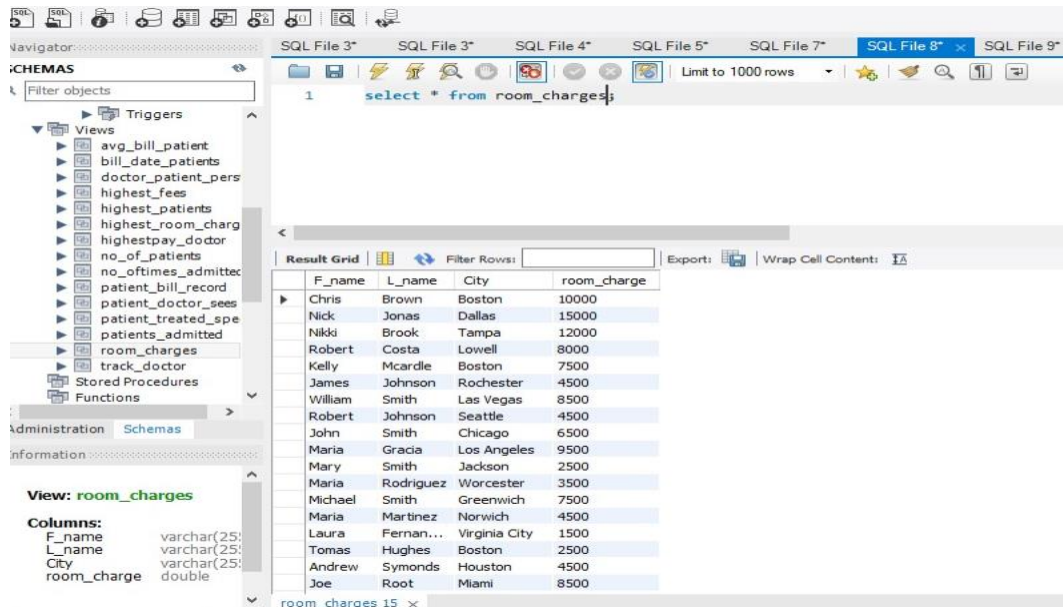
COUNT(P_ID)	Admit_Date
4	11/10/2021
3	10/20/2021
3	1/7/2022
3	2/6/2022
3	3/15/2022
1	11/7/2021
1	9/11/2021
1	5/8/2022
1	2/9/2022
1	6/4/2022
1	1/10/2021
1	6/22/2021
1	1/3/2022
1	3/7/2022
1	3/11/2022
1	7/5/2021
1	3/17/2021

Below the result grid, the 'View: patients\_admitted' section shows the columns: 'COUNT(P\_ID)' (bigint) and 'Admit\_Date' (varchar(25)).

```

13) CREATE VIEW `room_charges` AS
SELECT
    `p`.`F_name` AS `F_name`,
    `p`.`L_name` AS `L_name`,
    `p`.`City` AS `City`,
    SUM(`bp`.`Room_Charge`) AS `room_charge`
FROM
    ((`patient` `p`
    LEFT JOIN `record` `r` ON ((`r`.`P_ID` = `p`.`P_ID`)))
    LEFT JOIN `billpayment` `bp` ON ((`bp`.`Bill_ID` = `r`.`Bill_ID`)))
GROUP BY `p`.`F_name`, `p`.`L_name`, `p`.`City`

```



The screenshot shows a SQL IDE interface. On the left, the 'Navigation' pane displays the database schema 'CHEMAS' with various tables and views. The 'room\_charges' view is selected. The top pane shows the SQL query being executed: 'select \* from room\_charges;'. The bottom pane shows the 'Result Grid' with 15 rows of data. The columns are 'F\_name', 'L\_name', 'City', and 'room\_charge'.

F_name	L_name	City	room_charge
Chris	Brown	Boston	10000
Nick	Jonas	Dallas	15000
Nikki	Brook	Tampa	12000
Robert	Costa	Lowell	8000
Kelly	Mcardle	Boston	7500
James	Johnson	Rochester	4500
William	Smith	Las Vegas	8500
Robert	Johnson	Seattle	4500
John	Smith	Chicago	6500
Maria	Gracia	Los Angeles	9500
Mary	Smith	Jackson	2500
Maria	Rodriguez	Worcester	3500
Michael	Smith	Greenwich	7500
Maria	Martinez	Norwich	4500
Laura	Fernan...	Virginia City	1500
Tomas	Hughes	Boston	2500
Andrew	Symonds	Houston	4500
Joe	Root	Miami	8500

```

14) CREATE VIEW `track_doctor` AS
SELECT
    `doctor`.`Doc_ID` AS `Doc_ID`,
    `record`.`R_ID` AS `R_ID`,
    `doctor`.`first_nm` AS `first_nm`
FROM
    (`doctor`
JOIN `record` ON ((`doctor`.`Doc_ID` = `record`.`Doc_ID`)))

```

The screenshot shows a SQL IDE interface with a database named 'HEMAS'. The left sidebar displays a tree view of database objects, including Triggers, Views, Stored Procedures, and Functions. The 'Views' folder is expanded, showing a list of views, with 'track\_doctor' selected. Below the tree, the 'View: track\_doctor' section displays the columns: Doc\_ID (int), R\_ID (int), and first\_nm (varchar(255)).

The main editor window shows the SQL query: `select * from track_doctor;`. The 'Result Grid' at the bottom displays the results of the query, showing 18 rows of data. The columns are Doc\_ID, R\_ID, and first\_nm.

Doc_ID	R_ID	first_nm
1407160575	1	BEATRICE
1407164437	2	PATRICIA
1407172521	3	ANDREW
1407182421	4	STEPHEN
1407230097	5	SONYA
1407234792	6	MATTHEW
1407243538	7	JOSEPH
1407245285	8	STACI
1407254881	9	KESHA
1407256159	10	EDITH
1407257652	11	SAKEENA
1407266570	12	PATRICK
1407289168	13	ABRAHAM
1407292550	14	NATHAN
1407355266	15	CHANA
1407467475	16	TERRI
1407491400	17	REBECCA
1407800154	18	FRANCIS

```

15) CREATE VIEW `no_oftimes_admitted` AS
SELECT
    `patient`.`P_ID` AS `P_ID`,
    `patient`.`F_name` AS `F_name`,
    COUNT(`patient`.`Admit_Date`) AS `count(Admit_Date)`
FROM
    `patient`
GROUP BY `patient`.`P_ID`
LIMIT 1

```

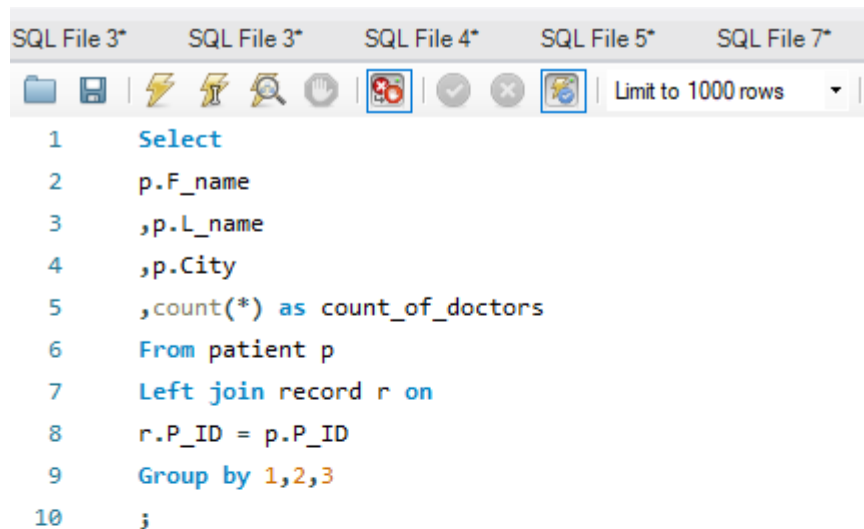
The screenshot shows a SQL IDE interface with the following components:

- Navigator:** Displays a tree of database objects. Under the 'Views' folder, 'no\_oftimes\_admitted' is highlighted.
- SQL Editor:** Contains the SQL query: `select * from no_oftimes_admitted;`
- Result Grid:** Displays the output of the query as a table with the following data:
 

P_ID	F_name	count(Admit_Date)
1	Chris	1
- Information Panel:** Shows details for the selected view:
  - View:** no\_oftimes\_admitted
  - Columns:**
    - P\_ID: int
    - F\_name: varchar
    - count(Admit\_Date): bigint

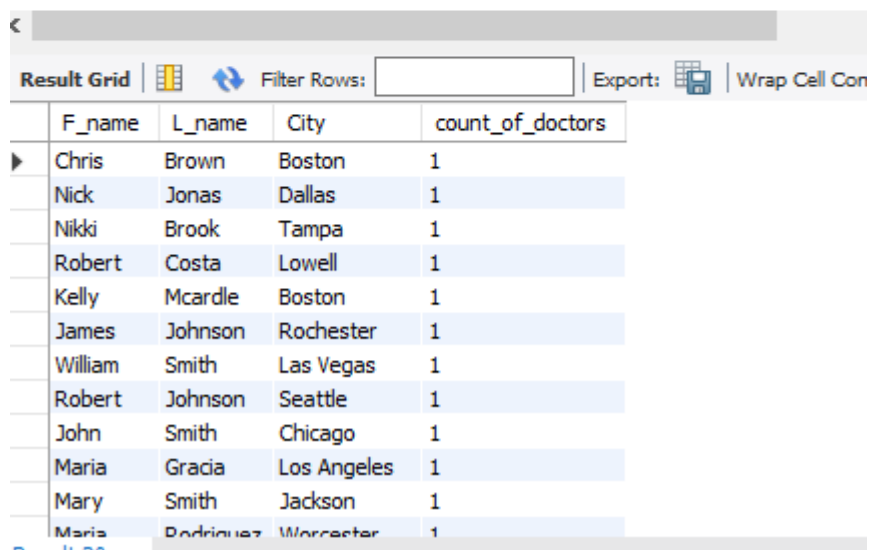
## USE-CASES:

- 1) To show number of patients a doctor serves in each city :



The screenshot shows a SQL query editor with five tabs labeled 'SQL File 3\*', 'SQL File 3\*', 'SQL File 4\*', 'SQL File 5\*', and 'SQL File 7\*'. The active tab is 'SQL File 4\*'. The query is as follows:

```
1  Select
2  p.F_name
3  ,p.L_name
4  ,p.City
5  ,count(*) as count_of_doctors
6  From patient p
7  Left join record r on
8  r.P_ID = p.P_ID
9  Group by 1,2,3
10 ;
```

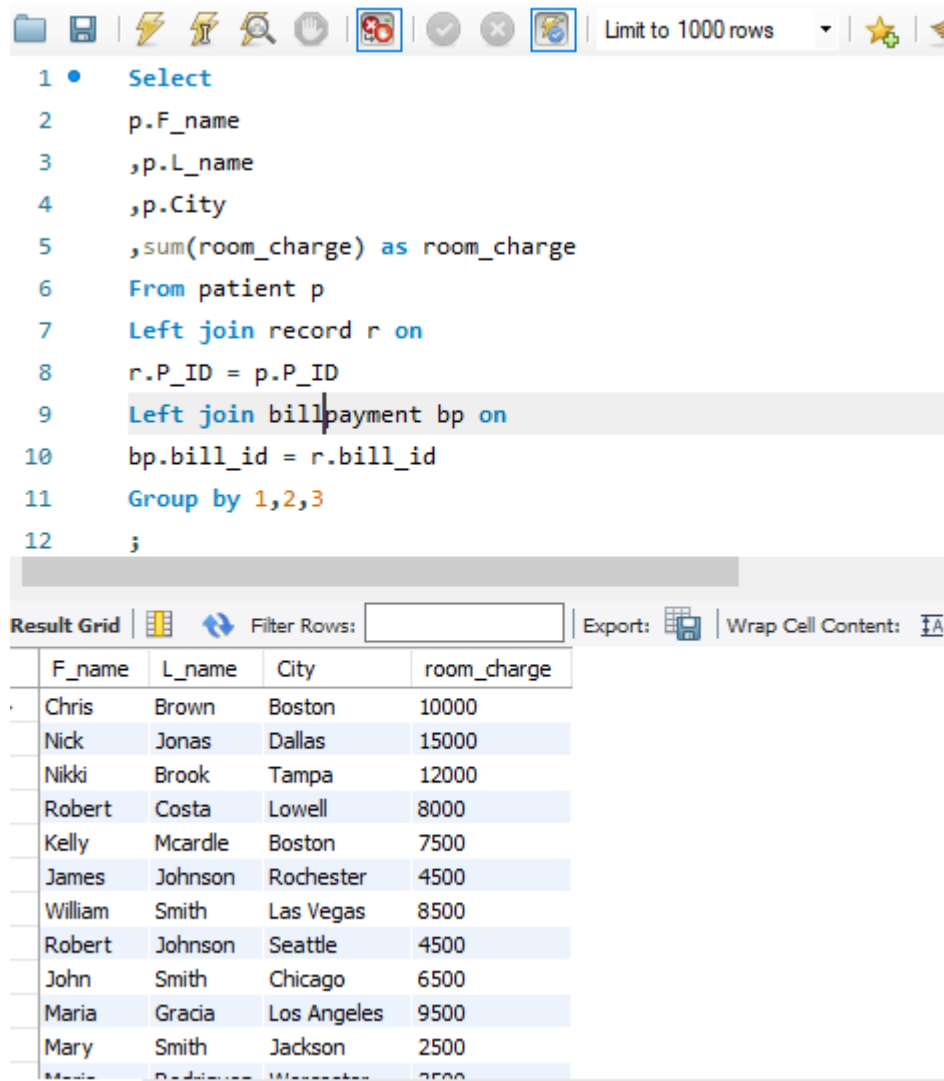


The screenshot shows the 'Result Grid' of the SQL query. It has a toolbar with 'Filter Rows', 'Export', and 'Wrap Cell Content' options. The results are displayed in a table with the following columns: F\_name, L\_name, City, and count\_of\_doctors. The data is as follows:

F_name	L_name	City	count_of_doctors
Chris	Brown	Boston	1
Nick	Jonas	Dallas	1
Nikki	Brook	Tampa	1
Robert	Costa	Lowell	1
Kelly	Mcardle	Boston	1
James	Johnson	Rochester	1
William	Smith	Las Vegas	1
Robert	Johnson	Seattle	1
John	Smith	Chicago	1
Maria	Gracia	Los Angeles	1
Mary	Smith	Jackson	1
Maria	Rodriguez	Worcester	1



2) To show amount of room charged in each city for Patient :



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • Select
2   p.F_name
3   ,p.L_name
4   ,p.City
5   ,sum(room_charge) as room_charge
6 From patient p
7 Left join record r on
8   r.P_ID = p.P_ID
9 Left join billpayment bp on
10  bp.bill_id = r.bill_id
11 Group by 1,2,3
12 ;
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with the following data:

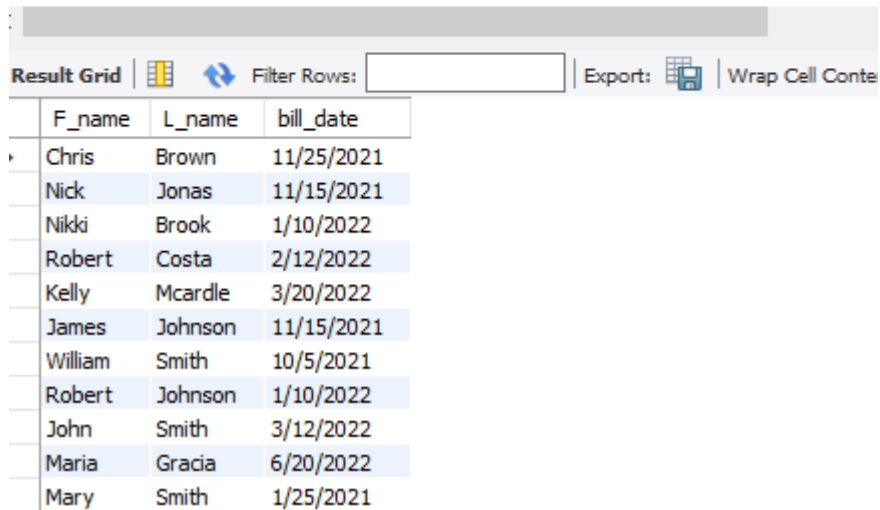
F_name	L_name	City	room_charge
Chris	Brown	Boston	10000
Nick	Jonas	Dallas	15000
Nikki	Brook	Tampa	12000
Robert	Costa	Lowell	8000
Kelly	Mcardle	Boston	7500
James	Johnson	Rochester	4500
William	Smith	Las Vegas	8500
Robert	Johnson	Seattle	4500
John	Smith	Chicago	6500
Maria	Gracia	Los Angeles	9500
Mary	Smith	Jackson	2500
Maria	Padilla	Worcester	3500

3) Show billing date of patients:



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and settings. The query text is as follows:

```
1 • Select
2     p.F_name
3     ,p.L_name
4     ,(Bill_Date) as bill_date
5 From patient p
6 Left join billpayment bp on
7 bp.P_ID = p.P_ID
8 Group by 1,2,3
9 ;
10
```



The screenshot shows a 'Result Grid' window with a toolbar for filtering, exporting, and wrapping text. The results are displayed in a table with the following data:

	F_name	L_name	bill_date
•	Chris	Brown	11/25/2021
	Nick	Jonas	11/15/2021
	Nikki	Brook	1/10/2022
	Robert	Costa	2/12/2022
	Kelly	Mcardle	3/20/2022
	James	Johnson	11/15/2021
	William	Smith	10/5/2021
	Robert	Johnson	1/10/2022
	John	Smith	3/12/2022
	Maria	Gracia	6/20/2022
	Mary	Smith	1/25/2021

4) View the number of patients served by doctors in a particular state:

```
1 • select p.state, count(*) as count_of_doctors From Patient p LEFT JOIN record r
2 on r.P_ID=p.P_ID Group by 1;
3
```

Result Grid

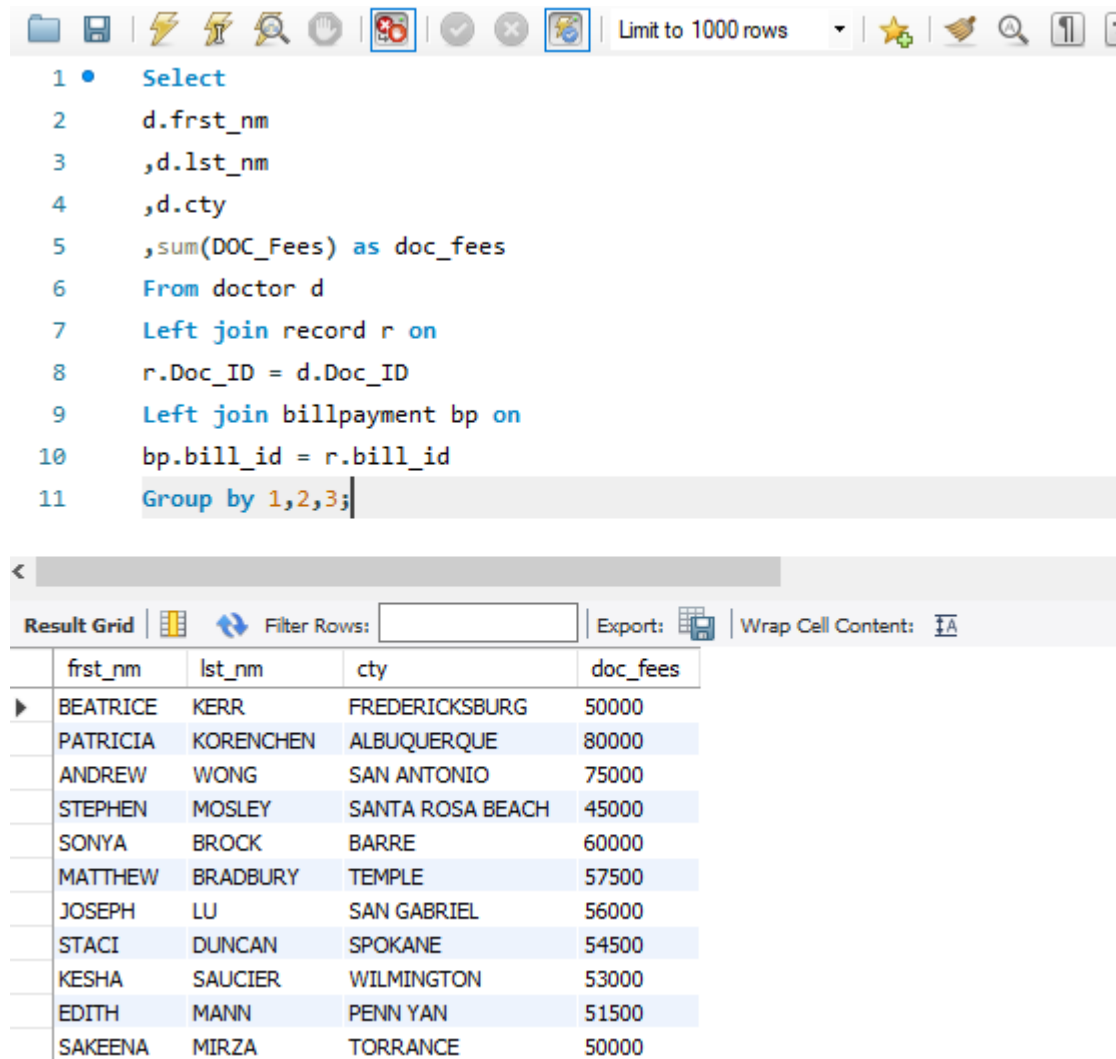
Filter Rows:

Export:

Wrap Cell Content:

	state	count_of_doctors
	MA	6
	TX	3
	FL	3
	NY	2
	NV	3
	WA	2
	IL	3
	CA	2
	MI	2
	CT	1
	NH	1

5) To show highest-paid medical professional in each city:



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • Select
2   d.frst_nm
3   ,d.lst_nm
4   ,d.cty
5   ,sum(DOC_Fees) as doc_fees
6 From doctor d
7 Left join record r on
8   r.Doc_ID = d.Doc_ID
9 Left join billpayment bp on
10  bp.bill_id = r.bill_id
11 Group by 1,2,3;
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with the following columns: frst\_nm, lst\_nm, cty, and doc\_fees.

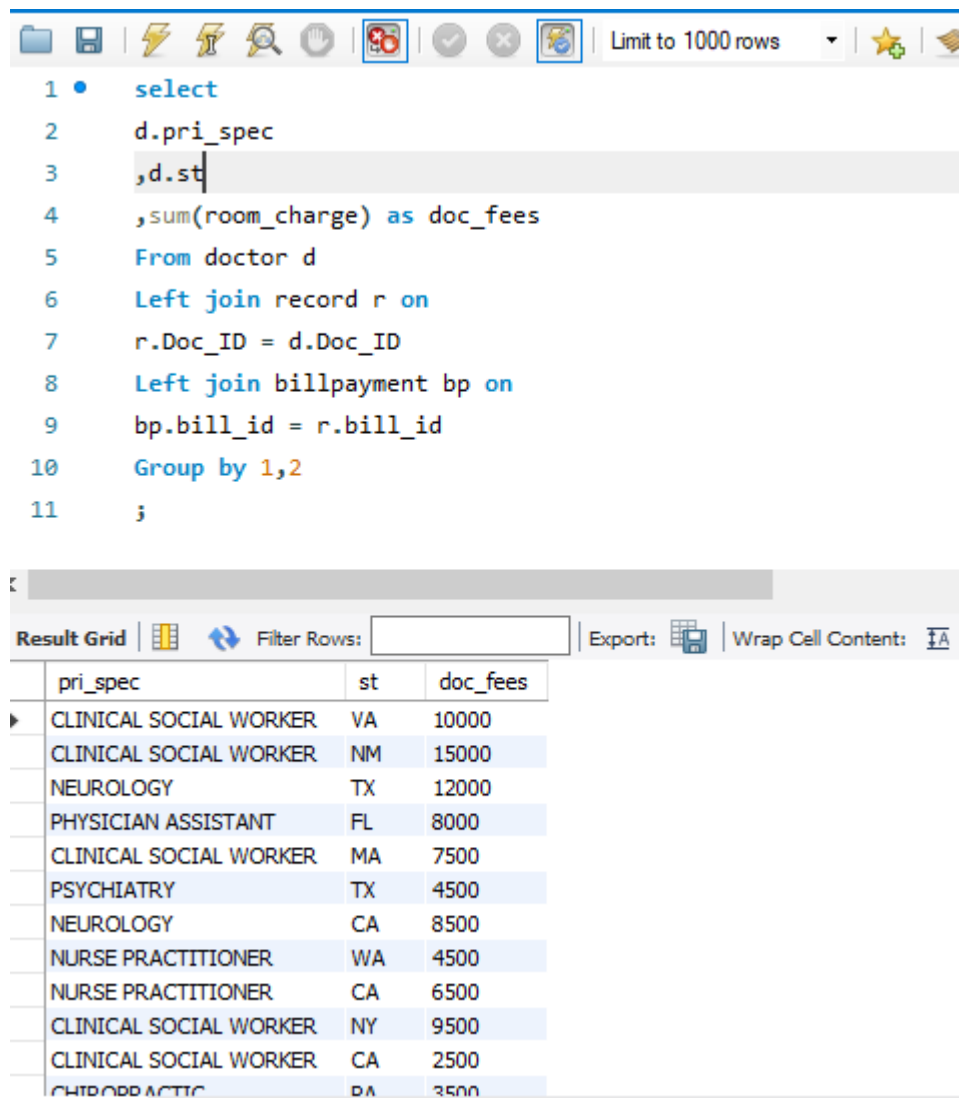
frst_nm	lst_nm	cty	doc_fees
BEATRICE	KERR	FREDERICKSBURG	50000
PATRICIA	KORENCHEN	ALBUQUERQUE	80000
ANDREW	WONG	SAN ANTONIO	75000
STEPHEN	MOSLEY	SANTA ROSA BEACH	45000
SONYA	BROCK	BARRE	60000
MATTHEW	BRADBURY	TEMPLE	57500
JOSEPH	LU	SAN GABRIEL	56000
STACI	DUNCAN	SPOKANE	54500
KESHA	SAUCIER	WILMINGTON	53000
EDITH	MANN	PENN YAN	51500
SAKEENA	MIRZA	TORRANCE	50000

6) Number of patients treated in each speciality :

```
1 • select d.pri_spec, count(*) as count_of_patients From Doctor d LEFT JOIN
2 record r on r.Doc_ID=d.Doc_ID Group by 1;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
pri_spec	count_of_patients			
CLINICAL SOCIAL WORKER	5			
NEUROLOGY	2			
PHYSICIAN ASSISTANT	1			
PSYCHIATRY	1			
NURSE PRACTITIONER	3			
CHIROPRACTIC	3			
OPTOMETRY	1			
CLINICAL PSYCHOLOGIST	3			
INTERNAL MEDICINE	2			
FAMILY PRACTICE	3			
PHYSICAL THERAPY	1			

7) Specialty bringing in highest room charges per state:




The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • select
2   d.pri_spec
3   ,d.st
4   ,sum(room_charge) as doc_fees
5 From doctor d
6 Left join record r on
7   r.Doc_ID = d.Doc_ID
8 Left join billpayment bp on
9   bp.bill_id = r.bill_id
10 Group by 1,2
11 ;
```

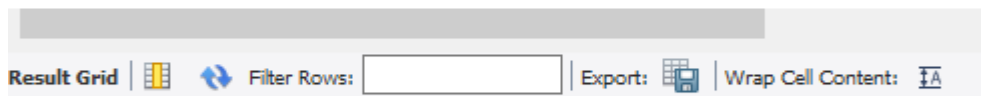
Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with three columns: 'pri\_spec', 'st', and 'doc\_fees'.

pri_spec	st	doc_fees
CLINICAL SOCIAL WORKER	VA	10000
CLINICAL SOCIAL WORKER	NM	15000
NEUROLOGY	TX	12000
PHYSICIAN ASSISTANT	FL	8000
CLINICAL SOCIAL WORKER	MA	7500
PSYCHIATRY	TX	4500
NEUROLOGY	CA	8500
NURSE PRACTITIONER	WA	4500
NURSE PRACTITIONER	CA	6500
CLINICAL SOCIAL WORKER	NY	9500
CLINICAL SOCIAL WORKER	CA	2500
CHIROPRACTIC	PA	3500

8) Doctor with highest fees in each state:



```
1 • Select
2   d.frst_nm
3   ,d.lst_nm
4   ,d.st
5   ,sum(DOC_Fees) as doc_fees
6 From doctor d
7 Left join record r on
8   r.Doc_ID = d.Doc_ID
9 Left join billpayment bp on
10  bp.bill_id = r.bill_id
11 Group by 1,2,3;
```



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

	frst_nm	lst_nm	st	doc_fees
1	BEATRICE	KERR	VA	50000
2	PATRICIA	KORENCHEN	NM	80000
3	ANDREW	WONG	TX	75000
4	STEPHEN	MOSLEY	FL	45000
5	SONYA	BROCK	MA	60000
6	MATTHEW	BRADBURY	TX	57500
7	JOSEPH	LU	CA	56000
8	STACI	DUNCAN	WA	54500
9	KESHA	SAUCIER	CA	53000
10	EDITH	MANN	NY	51500
11	SAKEENA	MIRZA	CA	50000
12	DATDICK	HOLLAND	PA	48500

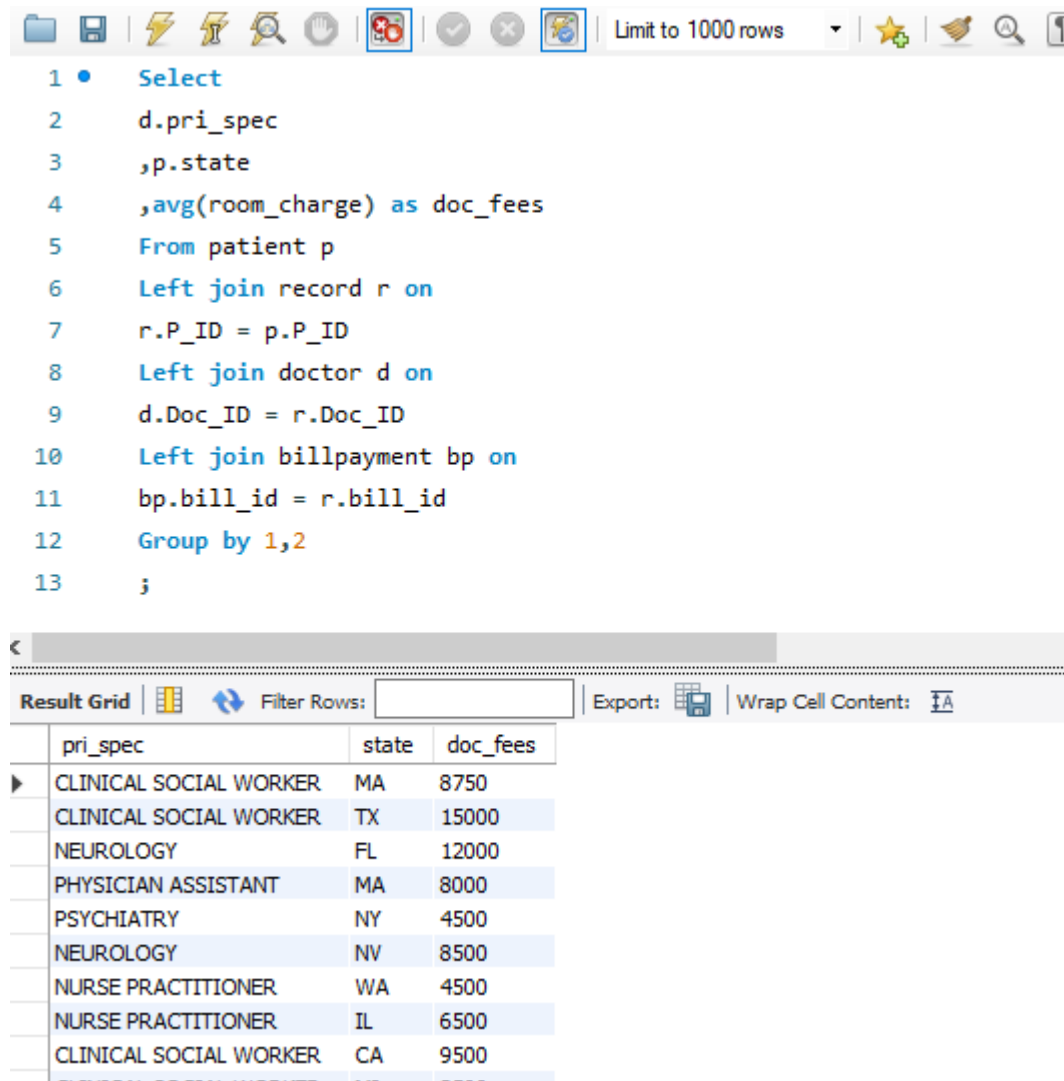
9) Doctor served the highest patients in every state :

```
1 • select d.frst_nm,d.lst_nm,d.st,count(*) as count_of_patients From Doctor d Left
2 JOIN record r on r.Doc_ID=d.Doc_ID GROUP BY 1,2,3;
```

Result Grid				
Filter Rows: <input type="text"/>				
Export: <input type="button" value=""/>				
Wrap Cell Content: <input type="button" value=""/>				
	frst_nm	lst_nm	st	count_of_patients
▶	BEATRICE	KERR	VA	1
	PATRICIA	KORENCHEN	NM	1
	ANDREW	WONG	TX	1
	STEPHEN	MOSLEY	FL	1
	SONYA	BROCK	MA	1
	MATTHEW	BRADBURY	TX	1
	JOSEPH	LU	CA	1
	STACI	DUNCAN	WA	1
	KESHA	SAUCIER	CA	1
	EDITH	MANN	NY	1
	SAKEENA	MIRZA	CA	1
	DATBICK	HOLLAND	PA	1



10) Average bill per patient per state:



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • Select
2   d.pri_spec
3   ,p.state
4   ,avg(room_charge) as doc_fees
5 From patient p
6 Left join record r on
7   r.P_ID = p.P_ID
8 Left join doctor d on
9   d.Doc_ID = r.Doc_ID
10 Left join billpayment bp on
11   bp.bill_id = r.bill_id
12 Group by 1,2
13 ;
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with the following data:

	pri_spec	state	doc_fees
▶	CLINICAL SOCIAL WORKER	MA	8750
	CLINICAL SOCIAL WORKER	TX	15000
	NEUROLOGY	FL	12000
	PHYSICIAN ASSISTANT	MA	8000
	PSYCHIATRY	NY	4500
	NEUROLOGY	NV	8500
	NURSE PRACTITIONER	WA	4500
	NURSE PRACTITIONER	IL	6500
	CLINICAL SOCIAL WORKER	CA	9500
	CLINICAL SOCIAL WORKER	MT	8500

11) Patient Admitted the maximum time:

```
1 • select P_ID,F_name,count(Admit_Date) as number_ofTimes_admitted
2 from patient
3 group by patient.P_ID
4 limit 1;
```

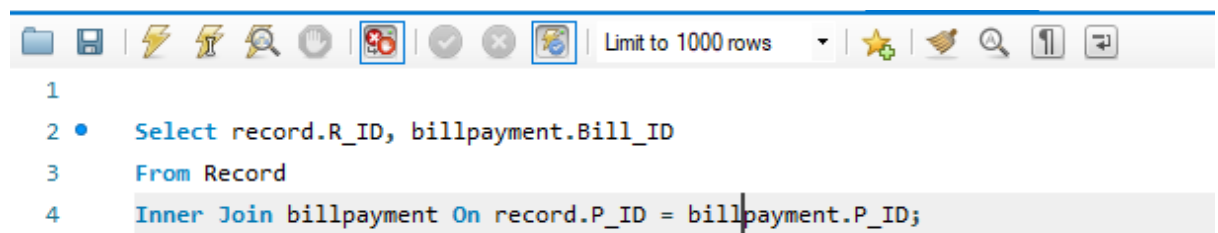
	P_ID	F_name	number_ofTimes_admitted
▶	1	Chris	1

12) Count patient admit date:

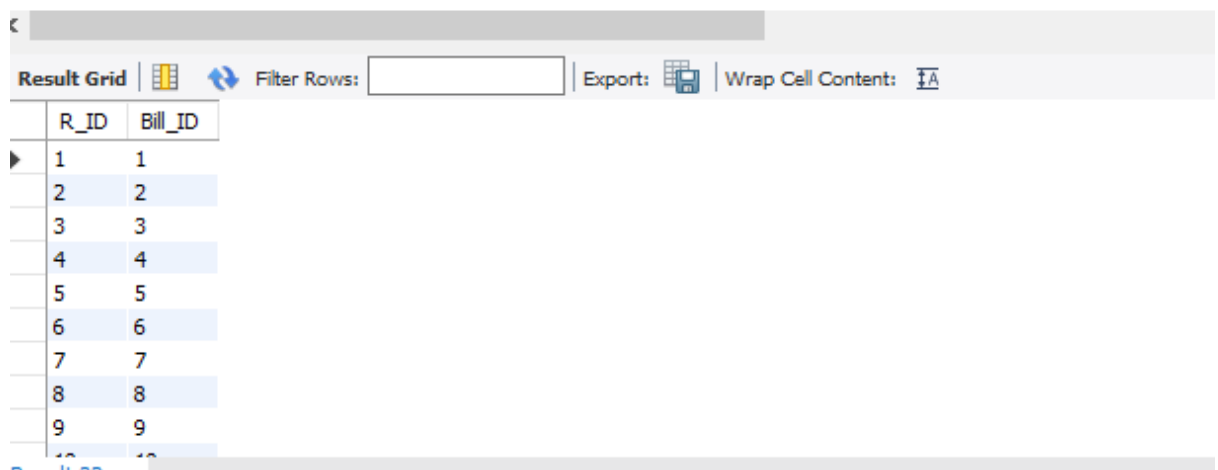
```
1 • SELECT COUNT(P_ID) , Admit_Date
2 FROM patient
3 GROUP BY Admit_Date;
```

result Grid		Filter Rows:	Export:	Wrap Cell Content:
COUNT(P_ID)	Admit_Date			
4	11/10/2021			
3	10/20/2021			
3	1/7/2022			
3	2/6/2022			
3	3/15/2022			
1	11/7/2021			
1	9/11/2021			
1	5/8/2022			
1	2/9/2022			
1	6/1/2022			

13) Count Bill ID :

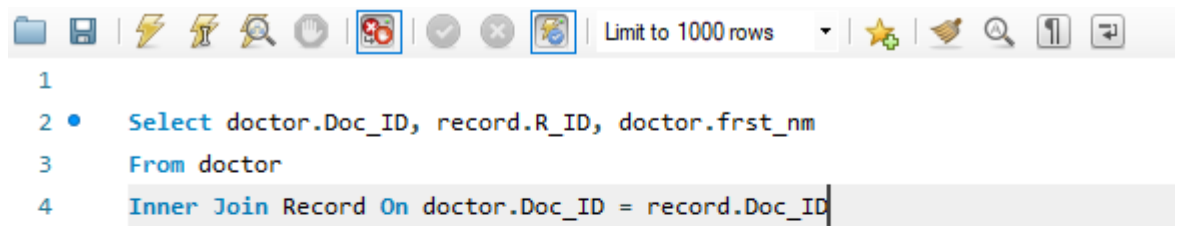


```
1
2 • Select record.R_ID, billpayment.Bill_ID
3 From Record
4 Inner Join billpayment On record.P_ID = billpayment.P_ID;
```



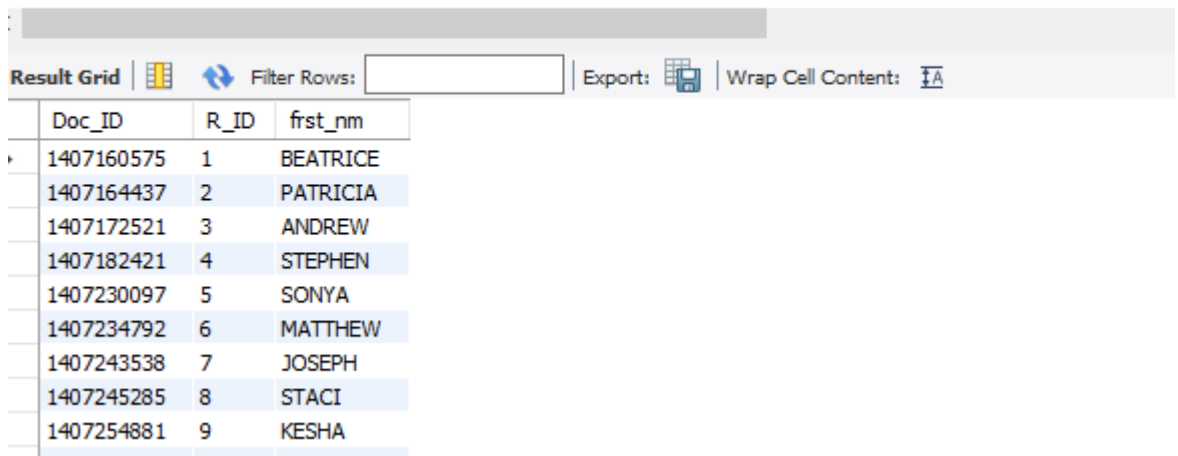
	R_ID	Bill_ID
▶	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	10	10

14) To show doctor and record ID:



The screenshot shows a SQL query editor interface. The toolbar at the top includes icons for file operations, execution, and search. The query text is as follows:


```
1
2 • Select doctor.Doc_ID, record.R_ID, doctor.frst_nm
3 From doctor
4 Inner Join Record On doctor.Doc_ID = record.Doc_ID
```



The screenshot shows the result grid of the SQL query. The grid has a toolbar with options for filtering, exporting, and wrapping cell content. The data is displayed in a table with the following columns: Doc\_ID, R\_ID, and frst\_nm.


Doc_ID	R_ID	frst_nm
1407160575	1	BEATRICE
1407164437	2	PATRICIA
1407172521	3	ANDREW
1407182421	4	STEPHEN
1407230097	5	SONYA
1407234792	6	MATTHEW
1407243538	7	JOSEPH
1407245285	8	STACI
1407254881	9	KESHA



15) To show total count of patient per city:



```
1 • SELECT COUNT(P_ID) , City
2 FROM patient
3 GROUP BY City;
```

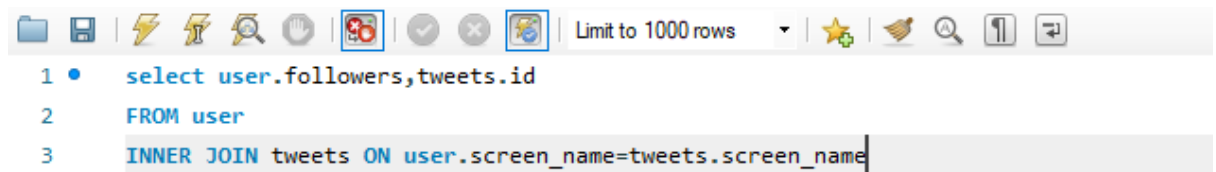
Limit to 1000 rows



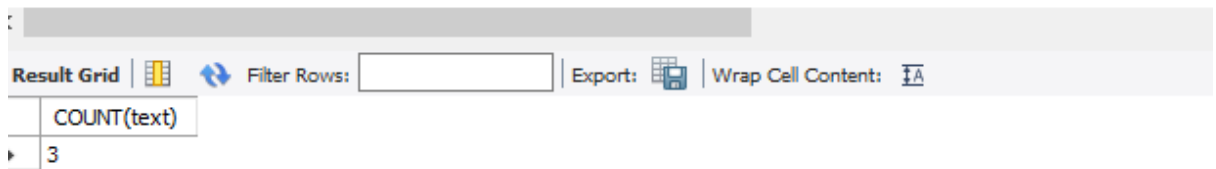
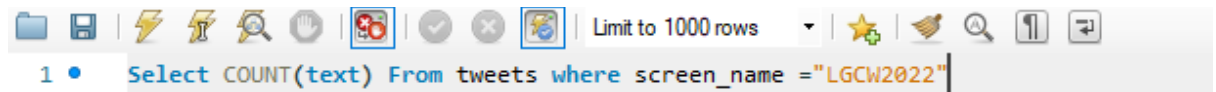
Result Grid |  Filter Rows:  | Export:  | Wr

	COUNT(P_ID)	City
3		Boston
1		Dallas
1		Tampa
1		Lowell
1		Rochester
1		Las Vegas
1		Seattle
2		Chicago
1		Los Angeles

16) View the follower and tweet id:

[illegible]

17) View the total number of tweets by a particular user:





## **Steps for choosing datasets to get final DB:**

<https://data.cms.gov/provider-data/topics/doctors-clinicians>

We first located the dataset on this website, and then we scraped the data using Python scripts. The data for the doctor table was web scraped from the Browse AI website using python programs, and using Jupyter, we were able to read the doctor's csv file and eliminate all of the null values. We were able to see the data by grouping by one column (main spec) and count (doc id). We then plotted the following bar graph using matplotlib. Finally, we re-group by primary spec, calculate the age mean, and plot the age graph. After completing everything, we manually entered the data in other tables and through our database after entering this information by Python programs. The data was entered into my sql workbench. We obtained our final database after normalizing the data from the prior assignment.