# **Critical Care Database**

1. Write a query to count the number of columns in the nursing chart table.

**QUERY:**

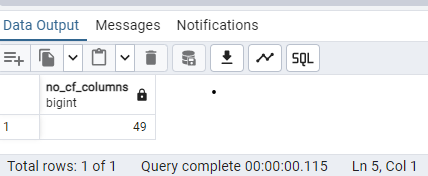
SELECT COUNT(\*) as no\_of\_columns

/\* information\_schema.columns contain metadata which stores all columns in all table of the database.\*/

FROM information\_schema.columns

WHERE table\_name = 'nursingchart' AND table\_schema = 'public';

**OUTPUT:**

****

2. Using a recursive query, show a list of patients that were transferred to various departments after they were admitted, along with the departments and the time of transfer. Hint, use earliest start time as time of admission

**QUERY:**

WITH RECURSIVE Transfer\_Report AS (

-- STEP 1: Identify the Base Case (First Transfer as Admission)

SELECT

t.patient\_id,

t.transferDept AS Department,

t.starttime AS Transfer\_Time,

1 AS level -- used to initialize the recursion for each patient

FROM transfer t

JOIN baseline b

ON t.patient\_id = b.patient\_id

WHERE t.starttime = (

SELECT min(starttime)

FROM transfer

WHERE patient\_id = t.patient\_id

)

-- STEP 2: Recursive Case - Find Subsequent Transfers

UNION ALL

SELECT

t.patient\_id,

t.transferdept AS Department,

t.starttime AS Transfer\_Time,

TR.level + 1 AS level -- This increments the level for subsequent transfers

FROM transfer t

JOIN Transfer\_Report TR

ON t.patient\_id = TR.patient\_id

WHERE t.starttime > TR.Transfer\_Time

AND TR.level < 5 -- Recursion depth limit

)

-- STEP 3: Apply ROW\_NUMBER to ensure unique records

SELECT

patient\_id,

Department,

Transfer\_Time

FROM (

SELECT

patient\_id,

Department,

Transfer\_Time,

ROW\_NUMBER() OVER (PARTITION BY patient\_id,Transfer\_Time ORDER BY Transfer\_Time) AS row\_num

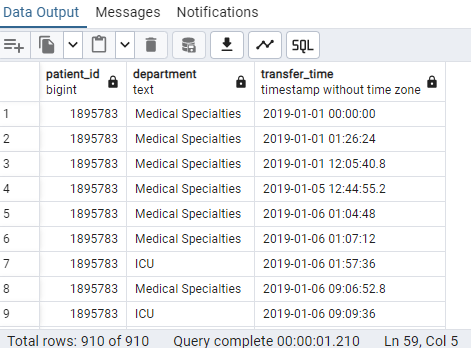
FROM Transfer\_Report

) AS Transfer\_Rank

WHERE row\_num =1 ---select only the first transfer per patient

ORDER BY patient\_id, Transfer\_Time;

**OUTPUT:**



NOTE:

TR.level < 5 ensures that recursion doesn't continue endlessly, limiting the query to a reasonable depth.

By using **ROW\_NUMBER()**, this query ensures that it returns only the **first department and transfer time** after admission for each patient, eliminating duplicates and maintaining a unique list of transfers.

3. List all patients who had a systolic blood pressure higher than the median value in the ICU.

Use windows functions to achieve this.

**QUERY:**

-- STEP 1: Rank the SBP values for all patients

WITH RankSBP AS (

SELECT

b.patient\_id,

n.invasive\_sbp AS SBP,

ROW\_NUMBER() OVER (ORDER BY n.invasive\_sbp) AS row\_num, -- Assign unique ranks

COUNT(\*) OVER() AS total\_count

FROM nursingchart n

JOIN baseline b ON n.inp\_no = b.inp\_no

WHERE n.invasive\_sbp IS NOT NULL

AND b.admitdept = 'ICU'

),

-- STEP 2: Identify median SBP value

MedianSBP AS (

SELECT

total\_count,

CASE

WHEN total\_count % 2 = 1 THEN

MAX(CASE WHEN row\_num = (total\_count + 1) / 2 THEN SBP END) -- Odd count, single middle value

ELSE

AVG(CASE WHEN row\_num IN (total\_count / 2, total\_count / 2 + 1) THEN SBP END) -- Even count, average of two middle values

END AS median\_value

FROM RankSBP

GROUP BY total\_count

)

-- STEP 3: Retrieve rows where SBP is higher than the median

SELECT DISTINCT

r.patient\_id,

r.SBP

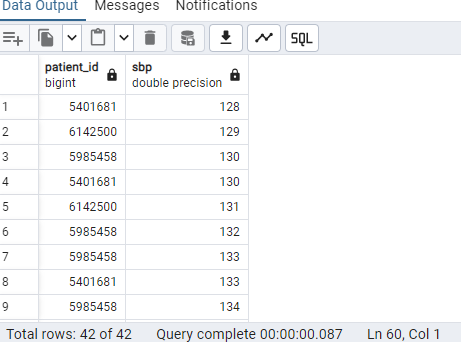
FROM RankSBP r

CROSS JOIN MedianSBP m

WHERE r.SBP > m.median\_value

ORDER BY r.SBP;

**OUTPUT:**

****

**NOTE:To filter the systolic blood pressure (SBP) values as requested in the task, I have used the invasive\_sbp field instead of other blood pressure metrics. This choice was made because the question specifically focuses on ICU patients, and invasive\_sbp is the standard and relevant measurement for blood pressure in ICU settings. Hence, only the invasive\_sbp data is used in this query to align with the ICU context.**

4. Create a function to fetch the details of the last recorded drug for a patient

**QUERY:**

-- STEP 1: Define the function signature

CREATE OR REPLACE FUNCTION last\_recorded\_drug\_details()

RETURNS TABLE(patient\_id INT,drugname VARCHAR, formula VARCHAR, drug\_time TIMESTAMP)

AS $$

--STEP 2: Begin the function definition

BEGIN

-- STEP 3: Write the query to fetch the last recorded drug

RETURN QUERY

SELECT

d.patient\_id::INT,

d.drugname::VARCHAR, -- Casting to VARCHAR

d.formula::VARCHAR, -- Casting to VARCHAR

d.drug\_time::TIMESTAMP -- Casting to TIMESTAMP

FROM drugs d

ORDER BY d.drug\_time DESC

LIMIT 1;

-- STEP 4: End the function definition

END;

-- STEP 5: Specify the language

$$ LANGUAGE plpgsql;

**OUTPUT:**

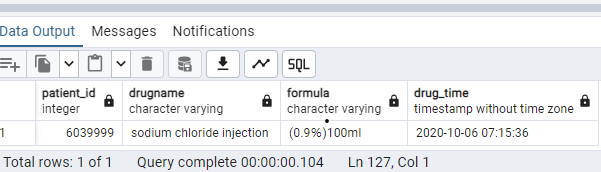


STEP 6: Call the function to execute it

**QUERY:**

SELECT \* FROM last\_recorded\_drug\_details();

**OUTPUT:**

****

5. List the 5 most recent transfers.

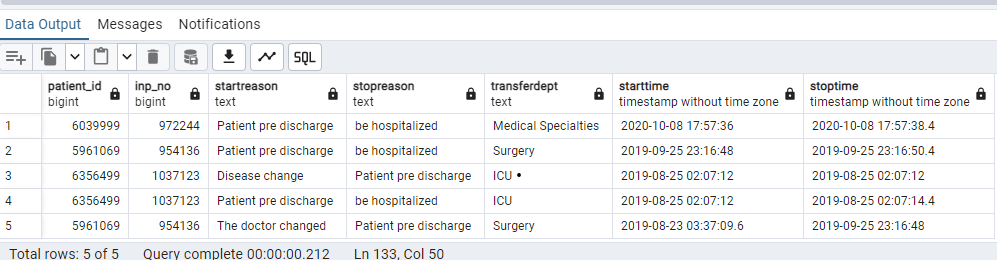
**QUERY:**

SELECT \* FROM transfer

ORDER BY starttime DESC –Sorts the results by the starttime column in descending order, so the most recent transfers appear first.

LIMIT 5; –Limits the output to the top 5 rows.

**OUTPUT:**

****

6. Use a window function to calculate the rolling average of heart rate for each patient.

**QUERY:**

WITH RollingAverage AS (

SELECT

b.patient\_id,

n.charttime,

n.heart\_rate,

AVG(n.heart\_rate) OVER (PARTITION BY b.patient\_id ORDER BY n.charttime ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

) AS Rolling\_Avg\_Heart\_Rate,

ROW\_NUMBER() OVER (PARTITION BY b.patient\_id ORDER BY n.charttime DESC

) AS row\_num

FROM

nursingchart n

JOIN

baseline b

ON

b.inp\_no = n.inp\_no

WHERE

n.heart\_rate IS NOT NULL

)

SELECT

patient\_id,

charttime,

heart\_rate,

Rolling\_Avg\_Heart\_Rate

FROM

RollingAverage

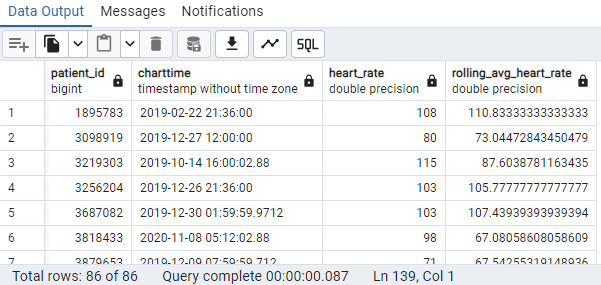
WHERE

row\_num = 1

ORDER BY

patient\_id;

**OUTPUT:**

****

7. List patients who were transferred back into surgery after they were discharged.

**QUERY:**

SELECT DISTINCT

t.patient\_id

FROM Transfer t

JOIN Baseline b ON b.PATIENT\_ID = t.PATIENT\_ID

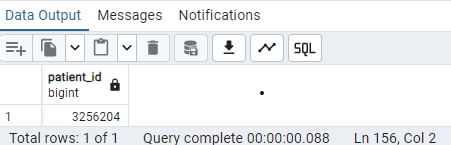
WHERE

t.TransferDept = 'Surgery' -- Transfer to surgery

AND t.StartTime > b.ICU\_discharge\_time

ORDER BY t.PATIENT\_ID;

**OUTPUT:**

****

8.Find the average age in each department by gender.

**QUERY:**

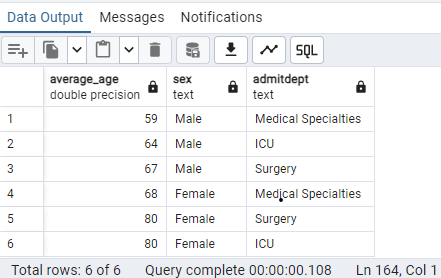
SELECT ROUND(AVG(age)) As average\_age, sex, admitdept ----Round function used to round the nearest whole number.

FROM baseline

GROUP BY admitdept,sex

ORDER BY average\_age;

**OUTPUT:**

****

9. Show all patients whose blood sugar is in the 99th percentile, and the time when it was recorded.

**QUERY:**

SELECT DISTINCT ON (b.patient\_id) ---distinct on used to remove the dulpicate records of patient and time

b.patient\_id,

n.charttime,

n.blood\_sugar

FROM nursingchart n

JOIN baseline b ON b.inp\_no = n.inp\_no

WHERE n.blood\_sugar >= (

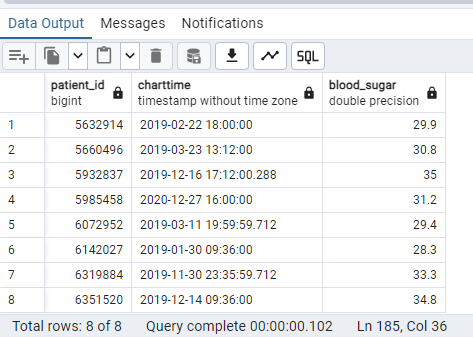
SELECT PERCENTILE\_CONT(0.99) WITHIN GROUP (ORDER BY blood\_sugar) AS Percentile\_99 ---calculate 99th perentile for blood sugar

FROM nursingchart

)

ORDER BY b.patient\_id, n.charttime;

**OUTPUT:**

****

10.Show the last 6 letters of disease names.

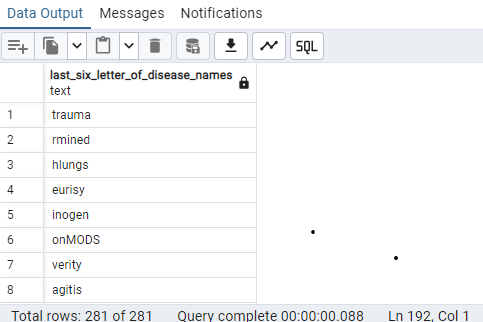
**QUERY:**

SELECT DISTINCT RIGHT(REGEXP\_REPLACE(icd\_desc,'[^a-zA-Z]','', 'g'),6) As last\_six\_letter\_of\_disease\_names

---REGEXP used to remove non-aplhabetic characters

FROM icd

**OUTPUT:**

****

11.Show the most commonly administered drug for each department, and the number of times it was administered. Use windows function to achieve this.

**QUERY:**

WITH RankedDrugs As (

SELECT d.drugname,b.admitdept,

COUNT(d.drug\_time) As drug\_administrated\_count,

RANK() OVER (PARTITION BY b.admitdept ORDER BY COUNT(d.drug\_time) DESC) As rank

FROM drugs d

JOIN baseline b ON b.patient\_id =d.patient\_id

GROUP BY b.admitdept,d.drugname

)

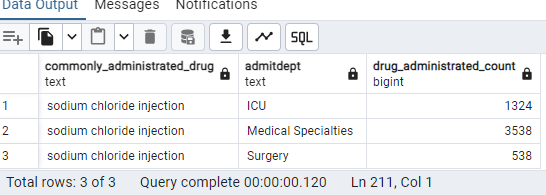
SELECT drugname As commonly\_administrated\_drug, admitdept, drug\_administrated\_count

FROM RankedDrugs

WHERE rank = 1 ----most commonly administered drugs in each department

ORDER BY admitdept;

**OUTPUT:**



12.Show the position of the letter y in disease name if it exists.

**QUERY:**

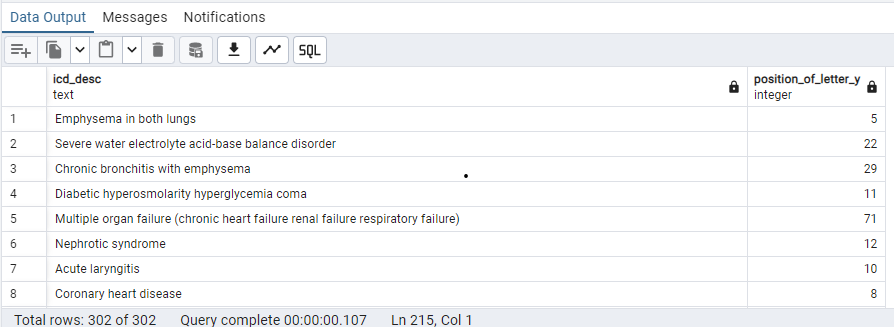
SELECT DISTINCT icd\_desc,

POSITION('y' IN icd\_desc) AS position\_of\_letter\_y

FROM icd

WHERE POSITION('y' IN icd\_desc) > 0;

**OUTPUT:**

****

13. Using windows function rank and display the 3 oldest patients admitted into each department.

**QUERY:**

WITH Rank\_Oldest\_patient AS (

SELECT patient\_id, age As patient\_age,admitdept,

RANK() OVER(PARTITION BY admitdept ORDER BY age DESC) As rankage

FROM baseline

)

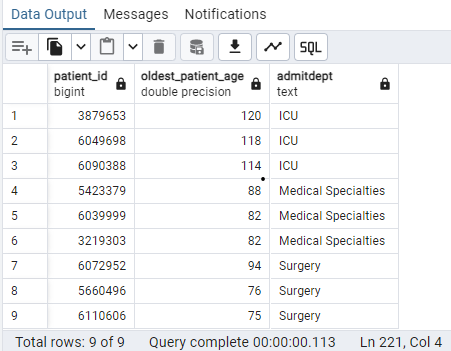
SELECT patient\_id, patient\_age As Oldest\_Patient\_age,admitdept

FROM Rank\_Oldest\_Patient

WHERE rankage<=3 ---provide top 3 patient for each department

ORDER BY admitdept,rankage

**OUTPUT:**



14. Show the number of patients that were discharged in 2020.

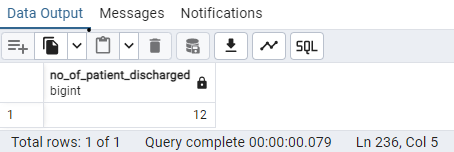
**QUERY:**

SELECT COUNT(patient\_id) As no\_of\_patient\_discharged

FROM baseline

WHERE EXTRACT(YEAR FROM icu\_discharge\_time) = 2020;

**OUTPUT:**



15.Show the total ICU stay in days for each patient who was transferred at least once.

**QUERY:**

SELECT patient\_id,

ROUND(SUM(EXTRACT(EPOCH FROM stoptime - starttime) / 86400), 2) AS Total\_ICU\_Stay

FROM transfer

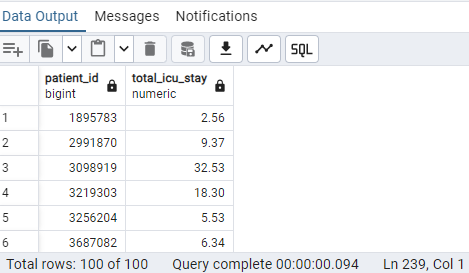
WHERE transferdept = 'ICU'

GROUP BY patient\_id

HAVING COUNT(\*) > 0

ORDER BY patient\_id;

**OUTPUT:**

****

16.Find the average, minimum, and maximum systolic blood pressure for patients in each department.

**QUERY:**

SELECT

b.admitdept AS department,

ROUND(AVG(COALESCE(n.blood\_pressure\_high, n.invasive\_sbp))::NUMERIC,2) AS avg\_systolic\_bp,

MIN(COALESCE(n.blood\_pressure\_high, n.invasive\_sbp)) AS min\_systolic\_bp,

MAX(COALESCE(n.blood\_pressure\_high, n.invasive\_sbp)) AS max\_systolic\_bp

FROM

nursingchart n

JOIN

baseline b ON n.inp\_no = b.inp\_no

WHERE

COALESCE(n.blood\_pressure\_high, n.invasive\_sbp) > 0

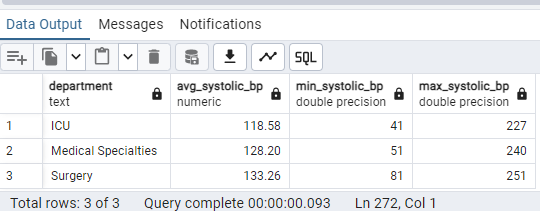
GROUP BY

b.admitdept

ORDER BY

b.admitdept;

**OUTPUT:**

****

17. Write a stored procedure to calculate the total number of patients per department and return the results as a table.

**QUERY:**

DROP TABLE IF EXISTS PatientRecords ;

-- Step 1: Create the permanent table to store results

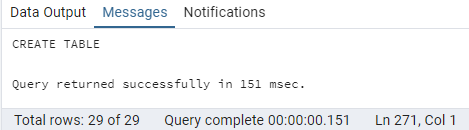
CREATE TABLE PatientRecords (

Patient\_Count INT,

Department TEXT

);

**OUTPUT:**

****

-- Step 2: Create the stored procedure

**QUERY:**

CREATE OR REPLACE PROCEDURE Patient\_Department\_Details()

LANGUAGE plpgsql

AS $$

BEGIN

-- Insert the data into the Created table

INSERT INTO PatientRecords (Patient\_Count, Department)

SELECT COUNT(patient\_id) AS Patient\_Count, admitdept AS Department

FROM baseline

GROUP BY admitdept

ORDER BY Patient\_Count DESC;

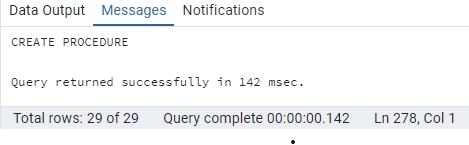
-- Raise a notice for confirmation

RAISE NOTICE 'Patient count per department has been updated in Patient\_Records.';

END;

$$;

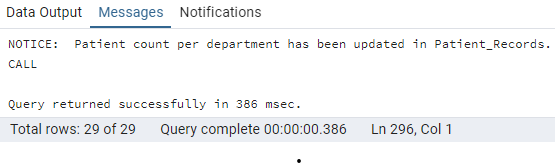
**OUTPUT:**

****

-- Step 3: Call the procedure to populate the table

**QUERY:**

CALL Patient\_Department\_Details();

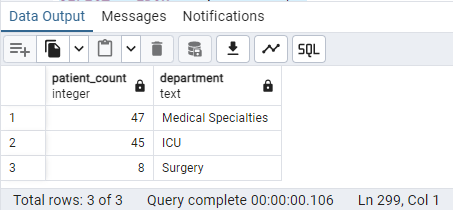
****

-- Step 4: Retrieve the result

**QUERY:**

SELECT \* FROM PatientRecords;

**OUTPUT:**

****

18.Show the top 3 patients who went into surgery the most number of times

**QUERY:**

SELECT patient\_id,COUNT(transferdept) As No\_of\_times\_in\_Surgery

FROM transfer

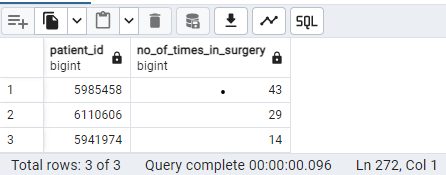
WHERE transferdept = 'Surgery'

GROUP BY patient\_id

ORDER BY No\_of\_times\_in\_Surgery DESC,patient\_id

LIMIT 3

**OUTPUT:**

****

19. Show patients whose critical-care pain observation tool score is 0.

**QUERY:**

SELECT DISTINCT b.patient\_id,

n.Cpot\_pain\_score AS Critical\_care\_pain\_observation\_score

FROM nursingchart n

JOIN baseline b

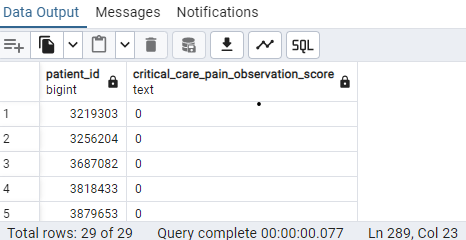
ON n.inp\_no = b.inp\_no

WHERE CAST(n.Cpot\_pain\_score As INT) = 0 ---cpot\_pain\_score column can be compared as an integer.

AND n.Cpot\_pain\_score IS NOT NULL

ORDER BY b.patient\_id;

**OUTPUT:**

****

20.Use windows functions to find BP measurements for 3 consecutive days.

List all patients who experienced a drop in blood pressure measurements for 3 continuous days.

**QUERY:**

WITH consecutive\_days AS (

SELECT

b.patient\_id,

DATE(n.charttime) AS reading\_date,

COALESCE( n.blood\_pressure\_high,n.invasive\_sbp) AS high\_bp,

COALESCE( n.blood\_pressure\_low,n.invasive\_diastolic\_blood\_pressure) AS low\_bp,

LAG(DATE(n.charttime)) OVER (PARTITION BY b.patient\_id ORDER BY DATE(n.charttime)) AS one\_day\_ago\_date,

LAG(COALESCE(n.blood\_pressure\_low,n.invasive\_diastolic\_blood\_pressure)) OVER (PARTITION BY b.patient\_id ORDER BY DATE(n.charttime)) AS one\_day\_ago\_bp\_low,

LAG(COALESCE(n.blood\_pressure\_high,n.invasive\_sbp)) OVER (PARTITION BY b.patient\_id ORDER BY DATE(n.charttime)) AS one\_day\_ago\_bp\_high,

LAG(DATE(n.charttime), 2) OVER (PARTITION BY b.patient\_id ORDER BY DATE(n.charttime)) AS two\_days\_ago\_date,

LAG(COALESCE(n.blood\_pressure\_low,n.invasive\_diastolic\_blood\_pressure, 2)) OVER (PARTITION BY b.patient\_id ORDER BY DATE(n.charttime)) AS two\_days\_ago\_bp\_low,

LAG(COALESCE(n.blood\_pressure\_high,n.invasive\_sbp, 2)) OVER (PARTITION BY b.patient\_id ORDER BY DATE(n.charttime)) AS two\_days\_ago\_bp\_high

FROM nursingchart n

JOIN baseline b

ON b.inp\_no = n.inp\_no

WHERE COALESCE( n.blood\_pressure\_high,n.invasive\_sbp) IS NOT NULL AND COALESCE( n.blood\_pressure\_low,n.invasive\_diastolic\_blood\_pressure) IS NOT NULL

),

drops\_over\_consecutive\_days AS (

SELECT

patient\_id,

reading\_date,

high\_bp,

low\_bp,

one\_day\_ago\_date,

one\_day\_ago\_bp\_high,

one\_day\_ago\_bp\_low,

two\_days\_ago\_date,

two\_days\_ago\_bp\_high,

two\_days\_ago\_bp\_low

FROM consecutive\_days

WHERE

-- Ensure the dates are consecutive

one\_day\_ago\_date = reading\_date - INTERVAL '1 day'

AND two\_days\_ago\_date = reading\_date - INTERVAL '2 day'

-- Check for a drop in blood pressure over 3 consecutive days

AND high\_bp < one\_day\_ago\_bp\_high

AND one\_day\_ago\_bp\_high < two\_days\_ago\_bp\_high

AND low\_bp < one\_day\_ago\_bp\_low

AND one\_day\_ago\_bp\_low < two\_days\_ago\_bp\_low

)

SELECT

patient\_id,

reading\_date,

high\_bp,

low\_bp,

one\_day\_ago\_date,

one\_day\_ago\_bp\_high,

one\_day\_ago\_bp\_low,

two\_days\_ago\_date,

two\_days\_ago\_bp\_high,

two\_days\_ago\_bp\_low,

CASE

WHEN COUNT(\*) = 0 THEN 'No records for consecutive days'

ELSE 'Records found for consecutive days'

END AS record\_status

FROM drops\_over\_consecutive\_days

GROUP BY patient\_id, reading\_date, high\_bp, low\_bp, one\_day\_ago\_date, one\_day\_ago\_bp\_high, one\_day\_ago\_bp\_low, two\_days\_ago\_date, two\_days\_ago\_bp\_high, two\_days\_ago\_bp\_low

UNION ALL

SELECT

NULL AS patient\_id,

NULL AS reading\_date,

NULL AS high\_bp,

NULL AS low\_bp,

NULL AS one\_day\_ago\_date,

NULL AS one\_day\_ago\_bp\_high,

NULL AS one\_day\_ago\_bp\_low,

NULL AS two\_days\_ago\_date,

NULL AS two\_days\_ago\_bp\_high,

NULL AS two\_days\_ago\_bp\_low,

'No records for consecutive days' AS record\_status

FROM (SELECT 1) AS no\_data\_check

WHERE NOT EXISTS (SELECT 1 FROM drops\_over\_consecutive\_days)

ORDER BY patient\_id;

**OUTPUT:**

****

**NOTE:**  No records for consecutive days with a significant drop in blood pressure for any patient, indicating no matching data in the dataset.

21. How was general health of patients who had a breathing rate > 20?

**QUERY:**

SELECT distinct outcome.patient\_id AS "Patien ID",

nursingchart.inp\_no AS "Inp No",

outcome.sf36\_generalhealth AS "generaihealth",

nursingchart.breathing

FROM outcome

JOIN baseline ON baseline.patient\_id = outcome.patient\_id

JOIN nursingchart On nursingchart.inp\_no = baseline.inp\_no

WHERE breathing >20

GROUP BY outcome.patient\_id, outcome.sf36\_generalhealth, nursingchart.breathing, nursingchart.inp\_no;

**OUTPUT:**

A screenshot of a computer

Description automatically generated

22. List patients with heart\_rate more than two standard deviations from the average.

**QUERY:**

SELECT DISTINCT baseline.patient\_id , nursingchart.heart\_rate

FROM baseline

JOIN nursingchart on nursingchart.inp\_no = baseline.inp\_no

WHERE nursingchart.heart\_rate > (SELECT AVG(heart\_rate) + 2 \* STDDEV(heart\_rate)

FROM nursingchart)

OR nursingchart.heart\_rate <(SELECT AVG(heart\_rate) - 2 \* STDDEV(heart\_rate)

FROM nursingchart**);**

**OUTPUT**

A screenshot of a data output

Description automatically generated

23. Create a trigger to raise notice and prevent deletion of a record from baseline table.

**QUERY:**

CREATE OR REPLACE FUNCTION prevent\_deletion()

RETURNS TRIGGER AS $$

BEGIN

RAISE NOTICE 'Deletion of Records Not Allowed ';

RETURN NULL;

END;

$$

LANGUAGE plpgsql;

**OUTPUT**

A screenshot of a computer

Description automatically generated

CREATE TRIGGER deletion\_not\_allowed

BEFORE DELETE ON baseline

FOR EACH ROW

EXECUTE PROCEDURE prevent\_deletion();

A screenshot of a computer

Description automatically generated

**QUERY TO VERIFY TRIGGER**

DELETE FROM baseline

WHERE patient\_ID = 6291268;

**OUTPUT:**

A screenshot of a computer

Description automatically generated

24. Use a CTE to get all patients with temperature readings above 38.

**QUERY:**

WITH Temperature\_Recording AS (

SELECT DISTINCT baseline.patient\_id,

nursingchart.temperature

FROM nursingchart

JOIN baseline On baseline.inp\_no = nursingchart.inp\_no

WHERE nursingchart.temperature > 38

)

SELECT \* FROM Temperature\_Recording

**OUTPUT:**

A screenshot of a computer

Description automatically generated

25. Develop a stored procedure to insert a new patient into the patients table and return the new patient ID.

**QUERY:**

CREATE OR REPLACE PROCEDURE insert\_new\_patient(

--b\_patient\_id bigint,

b\_inp\_no bigint,

b\_age integer,

b\_sex text,

b\_admindept text,

b\_infectionsite text,

b\_patient\_id OUT bigint

)

LANGUAGE plpgsql

AS $$

DECLARE

M\_PATIENT\_ID bigint;

BEGIN

SELECT INTO M\_PATIENT\_ID MAX(Patient\_ID)+1 from baseline;

INSERT INTO baseline (PATIENT\_ID, inp\_no,age,sex,admitdept,infectionsite,icu\_discharge\_time)

VALUES(m\_patient\_ID, b\_inp\_no,b\_age,b\_sex,b\_admindept,b\_infectionsite,CURRENT\_DATE);

b\_patient\_id:=M\_PATIENT\_ID;

END;

$$;

**OUTPUT: 1**

A screenshot of a computer

Description automatically generated

**QUERY:2**

**EXECUTE PROCEDURE:**

CALL insert\_new\_patient(

100000,90,'Male','icu','Liver', null --6357027

);

**OUTPUT: 2**

**A screenshot of a computer

Description automatically generated**

26. Find a correlation between blood sugar levels and discharge time from ICU.

**QUERY:**

SELECT

CORR (blood\_sugar,(DATE\_PART('HOUR',(ICU\_discharge\_time))))

AS correlation

FROM

baseline

JOIN nursingchart ON nursingchart.inp\_no = baseline.inp\_no;

**OUTPUT:**

A screenshot of a computer

Description automatically generated

27. Divide the patients into 3 age groups.

**QUERY:**

SELECT

CASE

WHEN age < 18 THEN 'Child(0-17)'

WHEN age BETWEEN 18 AND 64 THEN 'Adult(18-64)'

ELSE 'Senior (65+)'

END AS AgeGroup, \*

FROM baseline

ORDER BY ageGroup;

**OUTPUT:**

A screenshot of a computer

Description automatically generated

28. Show the hour(as a time slot like 9 AM - 10 AM) when least discharges happen.

**QUERY:**

SELECT COUNT(PATIENT\_ID) AS DISCHARGE\_COUNT,

concat(DATE\_PART('HOUR',(ICU\_discharge\_time)),' Oclock to ',(DATE\_PART('HOUR',(ICU\_discharge\_time))+1),' Oclock' )

AS Time\_Range

FROM baseline

GROUP BY DATE\_PART('HOUR',(ICU\_discharge\_time))

ORDER BY DISCHARGE\_COUNT ASC;

**OUTPUT:**

A screenshot of a computer

Description automatically generated

29. Display 3 random patients who had UTI.

**QUERY:**

SELECT infectionsite,patient\_id

FROM baseline

WHERE infectionsite = 'UTI'

ORDER BY RANDOM()

LIMIT 3 ;

**OUTPUT:**

A screenshot of a computer

Description automatically generated

30. List the average length of stay for patients diagnosed with soft tissue infections.

**QUERY**:

SELECT baseline.InfectionSite,

AVG(Age(baseline.icu\_discharge\_time, StartTime)) AS average\_stays

FROM transfer

JOIN baseline ON baseline.inp\_no = transfer.inp\_no

WHERE InfectionSite = 'Soft Tissue' and startreason= 'Admission'

GROUP BY baseline.InfectionSite;

**OUTPUT:**

A screenshot of a computer

Description automatically generated

31. Create a table called Patient1952 to store all patients born in 1952 with their age and and sex info.

**QUERY**:

Create TABLE Patient1952

AS

SELECT Patient\_ID, Age,sex

FROM baseline

WHERE EXTRACT('YEAR' FROM (CURRENT\_DATE - CAST (age\*365 AS INTEGER)))=1952;

**OUTPUT:1**

A screenshot of a phone

Description automatically generated

**QUERY TO CHECK THE VIEW**

SELECT \* FROM Patient1952;

**OUTPUT:2**

A screenshot of a computer

Description automatically generated

32. Give the highest temperature, and highest heart rate recorded of all the patients

in surgery for each day.

**QUERY**:

SELECT

DATE(nursingchart.charttime )AS DATE,

MAX(nursingchart.temperature) AS highest\_temprature,

MAX(nursingchart.heart\_rate ) AS highest\_heart\_rate,

transfer.patient\_id

FROM

transfer

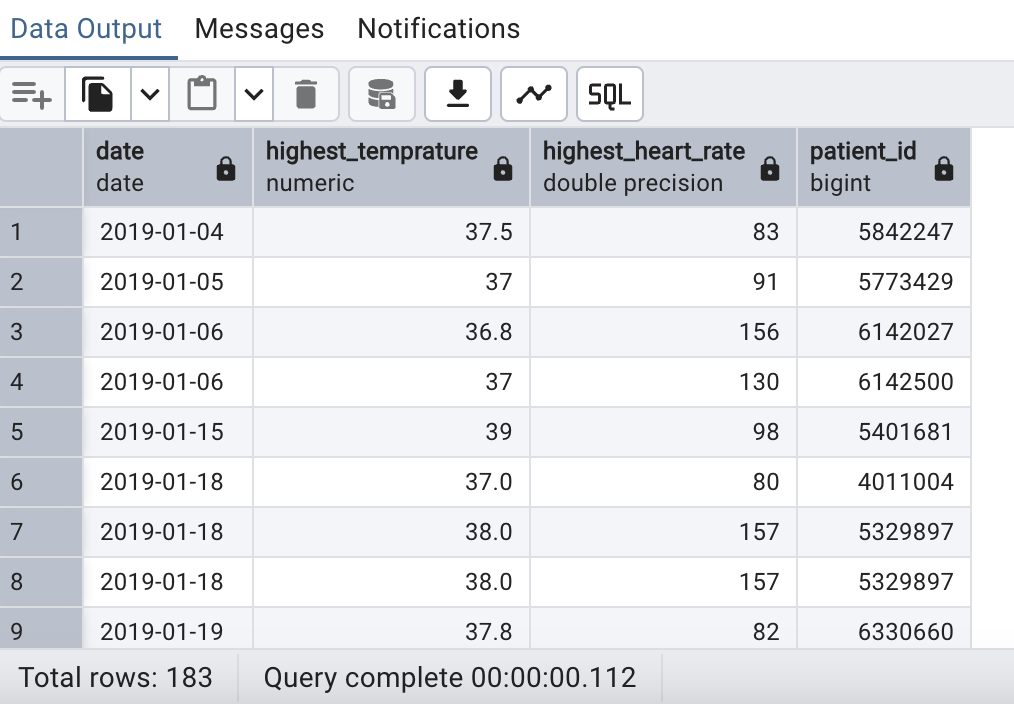
JOIN nursingchart ON nursingchart.inp\_no = transfer.inp\_no

WHERE UPPER(transfer.transferdept)='SURGERY'

GROUP BY nursingchart.charttime, transfer.patient\_id

ORDER BY DATE;

**OUTPUT**



33. List all patients whose heart rate increased by over 30% from the previous reading and the time when it happened.

List all occurences of heart rate increase. Use Windows functions to achieve this.

**QUERY**

SELECT CAST(((Test.heart\_rate - TEst.prev\_heart\_rate) / TEst.prev\_heart\_rate)\* 100 as bigint) AS "%INCREASE", \*

FROM

(SELECT

baseline.patient\_id,

nursingchart.charttime,

nursingchart.heart\_rate,

LAG(nursingchart.heart\_rate)OVER (PARTITION BY baseline.patient\_id

ORDER BY nursingchart.charttime) prev\_heart\_rate

FROM nursingchart

JOIN baseline ON baseline.inp\_no = nursingchart.inp\_no) AS TEst

WHERE

Test.heart\_rate IS NOT NULL AND Test.prev\_heart\_rate <>0

AND TEst.prev\_heart\_rate IS NOT NULL

AND ((Test.heart\_rate - TEst.prev\_heart\_rate) / TEst.prev\_heart\_rate)\* 100 >30;

**OUTPUT**

A screenshot of a computer

Description automatically generated

34. List patients who had milk and soft food but produced no urine.

**QUERY**

SELECT DISTINCT baseline.patient\_id, nursingchart.milk AS milk\_intake,

nursingchart.soft\_food AS soft\_food\_intake,

nursingchart.urine\_volume

FROM baseline

JOIN nursingchart ON nursingchart.inp\_no = baseline.inp\_no

WHERE nursingchart.milk >0

AND nursingchart.soft\_food >0

AND (urine\_volume = 0 or urine\_volume IS NULL);

**OUTPUT**

A screenshot of a computer

Description automatically generated

35. Using crosstab, show number of times each patient was transferred to each department.

**QUERY**

CREATE EXTENSION IF NOT EXISTS tablefunc;

SELECT \*

FROM CROSSTAB (

'SELECT patient\_id,transferdept,COUNT(\*) AS transfer\_count

FROM transfer

GROUP BY patient\_id,transferdept

ORDER BY patient\_id,transferdept',

'SELECT DISTINCT transferdept FROM transfer ORDER BY transferdept'

) AS (

patient\_id integer,

"ICU" integer,

"MEDICAL SPECIALITIES" integer,

"SURGERY" integer

);

**OUTPUT**

**A screenshot of a data output

Description automatically generated**

36. Produce a list of 100 normally distributed age values. Set the mean as the 3rd lowest age in the table, and assume the

standard deviation from the mean is 3.

**QUERY**

WITH sorted\_ages AS (

SELECT

age,

ROW\_NUMBER() OVER (ORDER BY age) AS

row\_num

FROM baseline

),

third\_lowest\_age AS(

SELECT

age AS mean

FROM sorted\_ages

WHERE row\_num = 3

)

SELECT

ROUND((mean + 3 \* sqrt(-2 \* lN(RANDOM())) \*

cos(2\* pi() \* RANDOM())) )AS age

FROM

third\_lowest\_age,

baseline;

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

37. Display the patients who engage in vigorous physical activity and have no body pain.

**QUERY**

SELECT patient\_id,SF36\_ActivityLimit\_VigorousActivity,

SF36\_Pain\_BodyPainPast4wk

FROM public.outcome

WHERE sf36\_activitylimit\_vigorousactivity ='3\_not limited at all'

AND UPPER(SF36\_Pain\_BodyPainPast4wk) ='1\_NONE';

**OUTPUT**

A screenshot of a computer

Description automatically generated

38. Create a view on outcome table to show patients with poor health.

**QUERY**

CREATE VIEW poor\_health\_condition AS

SELECT

patient\_id,

sf36\_generalhealth

FROM outcome

WHERE UPPER(sf36\_generalhealth) ='5\_POOR';

**OUTPUT-1**

**A screenshot of a computer

Description automatically generated**

**QUERY TO VERIFY THE VIEW**

SELECT \* FROM poor\_health\_condition;

**OUTPUT-2**

**A screenshot of a computer

Description automatically generated**

39. Create a procedure to check if a disease code exists.

**QUERY**

CREATE OR REPLACE PROCEDURE

Check\_Disease\_Code (patient\_disease\_code text,

code\_exists OUT BOOLEAN )

AS $$

BEGIN

SELECT EXISTS(

SELECT 1

FROM icd

WHERE icd\_code = patient\_disease\_code

)INTO code\_exists;

END ;

$$ LANGUAGE plpgsql;

**OUTPUT-1**

**A screenshot of a computer

Description automatically generated**

**QUERY TO VERIFY THE PROCEDURE**

CALL Check\_Disease\_Code ('G93.812',null)

**OUTPUT-2**

**A screenshot of a computer

Description automatically generated**

40. Which drug was most administered among patients who have never been intubated?

**QUERY**

SELECT

drugs.DrugName,

COUNT(\*)AS administered\_count

FROM drugs WHERE

drugs.patient\_id IN (SELECT DISTINCT patient\_id

FROM nursingchart

JOIN baseline ON baseline.inp\_no = nursingchart.inp\_no

WHERE nursingchart.Endotracheal\_intubation IS NOT NULL

AND nursingchart.Gastric\_tube IS NOT NULL)

GROUP BY drugs.DrugName

order by drugs.DrugName;

**OUTPUT**

A screenshot of a computer

Description automatically generated

1. Add a column birthyear to baseline column based on age.

**Step 1: Add birthyear column to baseline table.**

**QUERY:**

ALTER TABLE baseline ADD COLUMN IF NOT EXISTS birthyear INT;

**OUTPUT :**

A screenshot of a computer

Description automatically generated

**Step 2: Update the birthyear column based on age**

**QUERY:**

UPDATE baseline

SET birthyear = EXTRACT(YEAR FROM CURRENT\_DATE) - age;

**OUTPUT :**

A screenshot of a message

Description automatically generated

**Step 3: Verify if birthyear column updated correctly.**

**QUERY:**

SELECT \* FROM baseline;

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**



1. Use regular expression to find disease names that end in 'itis'.

**QUERY:**

SELECT DISTINCT icd\_desc

FROM icd

WHERE icd\_desc ~ 'itis$';

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**



1. Write a stored procedure to generate a summary report for a patient ID specified by user, including blood sugar, temperature, heart rate and drug administration.

**Step 1: Create the stored procedure:**

**QUERY:**

CREATE OR REPLACE PROCEDURE generate\_patient\_summary(patient\_id\_input BIGINT)

LANGUAGE plpgsql

AS $$

DECLARE

summary\_cursor CURSOR FOR

SELECT

b.patient\_id,

b.inp\_no,

MAX(n.blood\_sugar) AS blood\_sugar,

AVG(n.temperature) AS avg\_temperature,

AVG(n.heart\_rate) AS avg\_heart\_rate

FROM nursingchart n

JOIN baseline b ON n.inp\_no = b.inp\_no

WHERE b.patient\_id = patient\_id\_input

GROUP BY b.patient\_id, b.inp\_no;

drug\_cursor CURSOR FOR

SELECT

d.drugname,

d.formula,

d.drug\_time

FROM drugs d

WHERE d.patient\_id = patient\_id\_input

ORDER BY d.drug\_time DESC;

summary\_row RECORD;

drug\_row RECORD;

BEGIN

OPEN summary\_cursor;

LOOP

FETCH summary\_cursor INTO summary\_row;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'Patient ID: %, INP No: %, Blood Sugar: %, Avg Temperature: %, Avg Heart Rate: %',

summary\_row.patient\_id,

summary\_row.inp\_no,

summary\_row.blood\_sugar,

ROUND(summary\_row.avg\_temperature::NUMERIC, 2),

ROUND(summary\_row.avg\_heart\_rate::NUMERIC, 2);

END LOOP;

CLOSE summary\_cursor;

RAISE NOTICE '------------------------------------------------------------------------------------------------------';

RAISE NOTICE 'Drug Name Formula Drug Time';

RAISE NOTICE '-------------------------------------------------------------------------------------------------------';

OPEN drug\_cursor;

LOOP

FETCH drug\_cursor INTO drug\_row;

EXIT WHEN NOT FOUND;

RAISE NOTICE '% % %',

RPAD(drug\_row.drugname, 35, ' '),

RPAD(drug\_row.formula, 30, ' '),

drug\_row.drug\_time;

END LOOP;

CLOSE drug\_cursor;

RAISE NOTICE '---------------------------------------------------------------------------------------------------------';

END;

$$;

**OUTPUT:**

A screenshot of a message

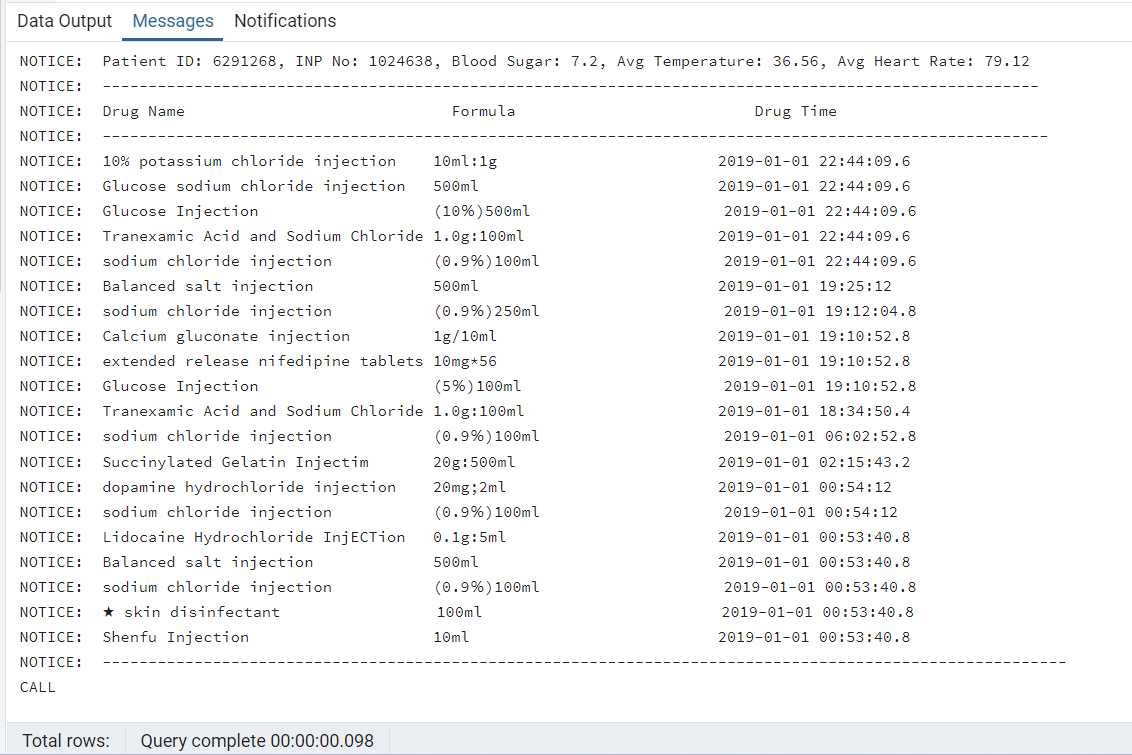
Description automatically generated

**Step 2: Call the stored procedure:**

**QUERY:**

CALL generate\_patient\_summary(6291268);

**OUTPUT:**





1. Create an index on any column in outcome table and also write a query to delete that index.

**Step 1: Create an index:**

**QUERY:**

CREATE INDEX IF NOT EXISTS idx\_outcome\_patient\_id

ON outcome (patient\_id);

**OUTPUT:**

**A screenshot of a message

Description automatically generated**

**Step 2: Check if index is created properly:**

**QUERY:**

SELECT indexname, indexdef

FROM pg\_indexes

WHERE tablename = 'outcome';

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**Step 3: Drop the index:**

**QUERY:**

DROP INDEX IF EXISTS idx\_outcome\_patient\_id;

**OUTPUT:**

**A screenshot of a message

Description automatically generated**

**Step 4: Check if index is dropped properly:**

**QUERY:**

SELECT indexname, indexdef

FROM pg\_indexes

WHERE tablename = 'outcome';

**OUTPUT:**

**A screenshot of a message

Description automatically generated**



1. Display the sf36\_generalhealth of all patients whose blood sugar has a standard deviation of more than 2 from the average.

**QUERY:**

SELECT

o.patient\_id,

temp.inp\_no,

temp.stddev\_blood\_sugar,

o.sf36\_generalhealth

FROM

(

SELECT inp\_no, STDDEV(blood\_sugar) AS stddev\_blood\_sugar

FROM nursingchart

GROUP BY inp\_no

)temp

JOIN baseline b ON temp.inp\_no = b.inp\_no

JOIN outcome o ON b.patient\_id = o.patient\_id

WHERE temp.stddev\_blood\_sugar > 2;

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**



1. Show the average time spent across different departments among alive patients, and among dead patients.

**QUERY:**

SELECT

t.transferdept,

o.follow\_vital,

AVG(EXTRACT(EPOCH FROM (t.stoptime - t.starttime)) / 3600) AS avg\_hours\_spent

FROM transfer t

JOIN outcome o ON t.patient\_id = o.patient\_id

WHERE o.follow\_vital IS NOT NULL

GROUP BY transferdept, follow\_vital

ORDER BY transferdept, follow\_vital;

**OUTPUT:**

A screenshot of a computer

Description automatically generated



1. Write a query to list all the users in the database.

**QUERY:**

SELECT usename FROM pg\_catalog.pg\_user;

**OUTPUT:**

**A screenshot of a message

Description automatically generated**



1. For each patient, find their maximum blood oxygen saturation while they were in the ICU , and display if it is above or below the average value among all patients.

**QUERY:**

WITH avg\_blood\_oxygen\_saturation AS

(

SELECT AVG(blood\_oxygen\_saturation) AS avg\_blood\_oxygen\_saturation

FROM nursingchart

),

max\_blood\_oxygen\_saturation\_by\_patient AS

(

SELECT n.inp\_no, MAX(blood\_oxygen\_saturation) as max\_blood\_oxygen\_saturation

FROM nursingchart n

JOIN baseline b ON n.inp\_no = b.inp\_no

JOIN transfer t ON b.patient\_id = t.patient\_id

WHERE t.transferdept = 'ICU'

GROUP BY n.inp\_no

)

SELECT inp\_no,

max\_blood\_oxygen\_saturation,

avg\_blood\_oxygen\_saturation,

CASE WHEN max\_blood\_oxygen\_saturation > avg\_blood\_oxygen\_saturation

THEN 'Above Average' ELSE 'Below Average' END AS comparison

FROM max\_blood\_oxygen\_saturation\_by\_patient

CROSS JOIN avg\_blood\_oxygen\_saturation;

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**



1. For each department, find the percentage of alive patients whose general health was poor after discharge.

**QUERY:**

WITH alive\_patients AS (

SELECT discharge\_dept, COUNT(\*) AS total\_alive\_patients

FROM outcome

WHERE follow\_vital = 'Alive'

GROUP BY discharge\_dept

),

poor\_health\_patients AS (

SELECT discharge\_dept, COUNT(\*) AS poor\_health\_count

FROM outcome

WHERE follow\_vital = 'Alive' AND sf36\_generalhealth = '5\_Poor'

GROUP BY discharge\_dept

)

SELECT

a.discharge\_dept,

a.total\_alive\_patients,

COALESCE(p.poor\_health\_count, 0) AS poor\_health\_count,

ROUND(COALESCE(p.poor\_health\_count, 0) \* 100.0 / a.total\_alive\_patients, 2) AS percentage\_poor\_health

FROM

alive\_patients a

LEFT JOIN

poor\_health\_patients p

ON

a.discharge\_dept = p.discharge\_dept

ORDER BY

a.discharge\_dept;

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**



1. Write a function that takes a date and returns the average temperature recorded for that day.

**Step 1: Create the function:**

**QUERY:**

CREATE OR REPLACE FUNCTION get\_avg\_temperature\_for\_date(date DATE)

RETURNS NUMERIC AS $$

DECLARE

avg\_temperature NUMERIC;

BEGIN

SELECT ROUND(AVG(temperature),2) INTO avg\_temperature

FROM nursingchart

WHERE charttime::DATE = date;

RETURN avg\_temperature;

END;

$$ LANGUAGE plpgsql;

**OUTPUT:**

A screenshot of a computer message

Description automatically generated

**Step 2 : Execute the function:**

**QUERY:**

SELECT get\_avg\_temperature\_for\_date(

'2019-02-23'

);

**OUTPUT:**

A screenshot of a message

Description automatically generated



1. Show the time spent in ICU for each patient that transferred to ICU from surgery.

**QUERY:**

WITH icu\_stay AS (

SELECT

patient\_id,

starttime AS icu\_start\_time,

stoptime AS icu\_stop\_time

FROM transfer

WHERE transferdept = 'ICU'

),

surgery\_stay AS (

SELECT

patient\_id,

stoptime AS surgery\_stoptime

FROM transfer

WHERE transferdept = 'Surgery'

AND stopreason = 'Transfer'

)

SELECT

i.patient\_id,

SUM(EXTRACT(EPOCH FROM i.icu\_stop\_time - i.icu\_start\_time) / 3600) AS icu\_hours

FROM icu\_stay i

JOIN surgery\_stay s

ON i.patient\_id = s.patient\_id

WHERE i.icu\_start\_time >= s.surgery\_stoptime -- Check if patient stayed in ICU after Surgery

GROUP BY i.patient\_id;

**OUTPUT:**

A screenshot of a computer

Description automatically generated



1. List all the drugs that were administered between 4 and 5 AM.

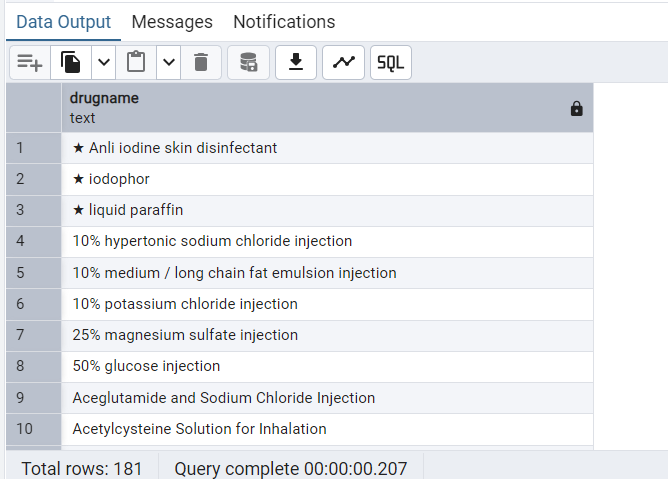
**QUERY:**

SELECT DISTINCT drugname

FROM drugs

WHERE drug\_time::time BETWEEN '04:00:00' AND '04:59:59';

**OUTPUT:**





1. Rank each patient based on the number of times they went to ICU.

**QUERY:**

SELECT

  patient\_id,

  icu\_visits,

  RANK() OVER (ORDER BY icu\_visits DESC) AS icu\_visits\_rank

FROM

(

  SELECT patient\_id, COUNT(\*) AS icu\_visits

  FROM transfer

  WHERE transferdept = 'ICU'

  GROUP BY patient\_id

) temp

ORDER BY icu\_visits\_rank;

**OUTPUT:**

A screenshot of a computer

Description automatically generated



1. Create a function to calculate the percentage of patients admitted into each department.

**Step 1: Create the function:**

**QUERY:**

CREATE OR REPLACE FUNCTION calculate\_patient\_percentage()

RETURNS TABLE (

department TEXT,

percentage NUMERIC

) AS $$

BEGIN

RETURN QUERY

WITH totalPatients AS (

SELECT COUNT(\*)::DECIMAL AS totCount

FROM baseline

),

PatientsByDept AS (

SELECT admitdept, COUNT(patient\_id) AS countByDept

FROM baseline

GROUP BY admitdept

)

SELECT

p.admitdept,

ROUND((p.countByDept / t.totCount) \* 100,2) AS percentage

FROM PatientsByDept p

CROSS JOIN totalPatients t;

END;

$$ LANGUAGE plpgsql;

**OUTPUT:**

A screenshot of a message

Description automatically generated

**Step 2: Execute the function:**

**QUERY:**

SELECT \* FROM calculate\_patient\_percentage();

**OUTPUT:**

A screenshot of a computer

Description automatically generated



1. Calculate the variance and standard deviation of oxygen flow readings across different admit departments.

**QUERY:**

SELECT

b.admitdept,

VARIANCE(n.oxygen\_flow::NUMERIC) AS variance\_oxygen\_flow,

STDDEV(n.oxygen\_flow::NUMERIC) AS stddev\_oxygen\_flow

FROM baseline b

JOIN nursingchart n ON b.inp\_no = n.inp\_no

GROUP BY b.admitdept;

**OUTPUT:**

A screenshot of a computer

Description automatically generated



1. Use a nested query to calculate the max blood\_sugar among patients whose average is below 120.

**QUERY:**

SELECT MAX(n.blood\_sugar) AS max\_blood\_sugar

FROM (

SELECT inp\_no, AVG(blood\_sugar) AS avg\_blood\_sugar

FROM nursingchart

GROUP BY inp\_no

HAVING AVG(blood\_sugar) < 120

) temp

JOIN nursingchart n ON temp.inp\_no = n.inp\_no;

**OUTPUT:**

**A screenshot of a message

Description automatically generated**

Note: The lab table also contains blood sugar values for some patients. But it has very limited records, so we are not considering that table. Also, we noticed that there could be some error in the question, as the average blood sugar is around 10.5. A blood sugar value of 120 is considered very high. So, no filtering happens here since all the patients blood sugar is less than 120.



1. List all transfers that started due to a change in disease.

**QUERY:**

SELECT patient\_id,

inp\_no,

startreason,

stopreason,

transferdept,

starttime,

stoptime

FROM transfer

WHERE startreason = 'Disease change';

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**



1. Show the number of drugs administered to every patient aged 65 or older.

**QUERY:**

SELECT d.patient\_id,COUNT(d.patient\_id) AS no\_of\_drugs

FROM baseline b

JOIN drugs d ON b.patient\_id = d.patient\_id

WHERE b.age >= 65

GROUP BY d.patient\_id

ORDER BY no\_of\_drugs DESC;

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**



1. Find the patients report feeling happy all the time.

**QUERY:**

SELECT patient\_id, sf36\_emotional\_happyperson

FROM outcome

WHERE sf36\_emotional\_happyperson = '1\_All of the time';

**OUTPUT:**

A screenshot of a computer

Description automatically generated



1. List the patients that were discharged in December of any year.

**QUERY:**

SELECT patient\_id, icu\_discharge\_time

FROM baseline

WHERE EXTRACT(MONTH FROM icu\_discharge\_time) = 12;

**OUTPUT:**

A screenshot of a message

Description automatically generated

61. List the last 100 patients that were discharged.

**QUERY:**

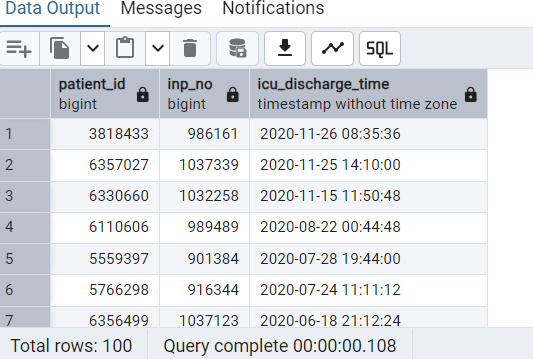
SELECT patient\_id, inp\_no, icu\_discharge\_time

FROM baseline

ORDER BY icu\_discharge\_time DESC

LIMIT 100;

**OUTPUT:**

****

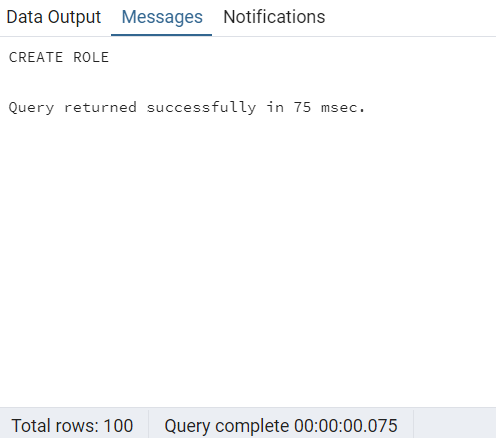
62. Create a role that cannot create other roles and expires on 12/31/24.

**QUERY:**

DROP ROLE IF EXISTS read\_only\_user;

CREATE ROLE read\_only\_user nocreaterole VALID UNTIL '2024-12-31';

**OUTPUT:**

****

**Step 2:**

To check the read\_only\_user role is created and check whether there is no create role and the validity.

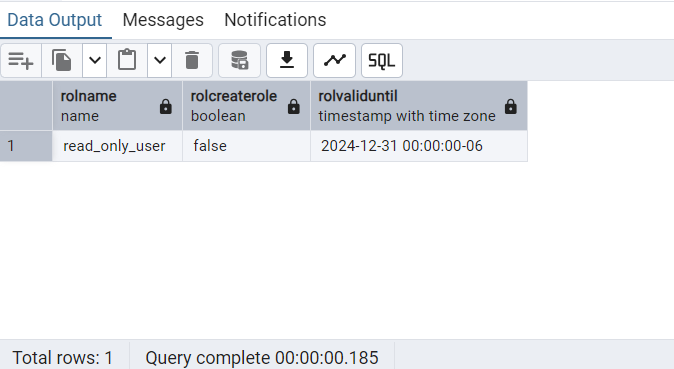
**QUERY:**

SELECT rolname, rolcreaterole, rolvaliduntil

FROM pg\_roles

WHERE rolname = 'read\_only\_user';

**OUTPUT:**

****

63. Find instances where a patient was transferred into the same

department twice within a day.

**QUERY:**

SELECT

patient\_id,

transferdept,

CAST(starttime AS DATE) AS transfer\_date,

COUNT(\*) AS transfer\_count

FROM transfer

GROUP BY

patient\_id,

transferdept,

CAST(starttime AS DATE)

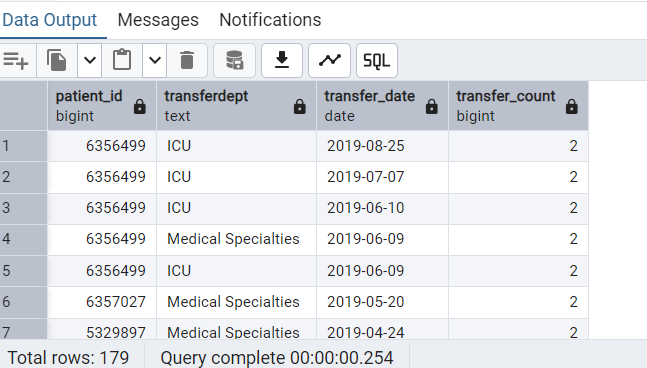
HAVING

COUNT(\*) = 2

ORDER BY

transfer\_date DESC;

**OUTPUT:**

****

64. Use nested CTEs to calculate the median temperature of patients

over the 60 years old while in ICU.

**QUERY:**

WITH temperature\_of\_patients\_above\_60\_in\_ICU AS

(

SELECT nc.temperature

FROM nursingchart nc

JOIN baseline b ON nc.inp\_no = b.inp\_no

JOIN transfer t ON b.patient\_id = t.patient\_id

WHERE b.age > 60 AND t.transferDept = 'ICU'

)

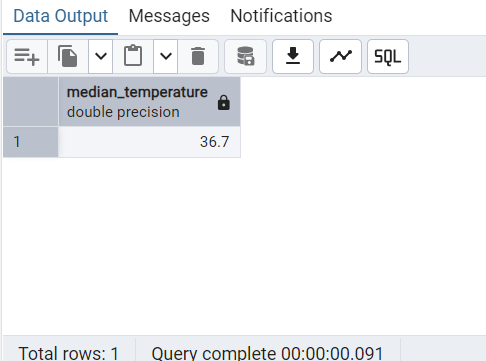
SELECT PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY temperature) AS median\_temperature

FROM temperature\_of\_patients\_above\_60\_in\_ICU;

--The percentile\_cont() function returns an interpolated value between multiple values based on the distribution.

--It is more accurate, but it may return a fractional value between two values in the input set.

**OUTPUT:**



65. Show the average sodium value for each patient.

**QUERY:**

SELECT ba.patient\_id, la.inp\_no, la.category,la.item, AVG(la.labvalue :: decimal) as avg\_sodium\_value

FROM lab la

JOIN baseline ba ON ba.inp\_no = la.inp\_no

WHERE la.item LIKE 'Sodium (NA)%'

AND la.labvalue NOT LIKE '?%'

GROUP BY ba.patient\_id, la.inp\_no, la.category, la.item

ORDER BY avg\_sodium\_value;

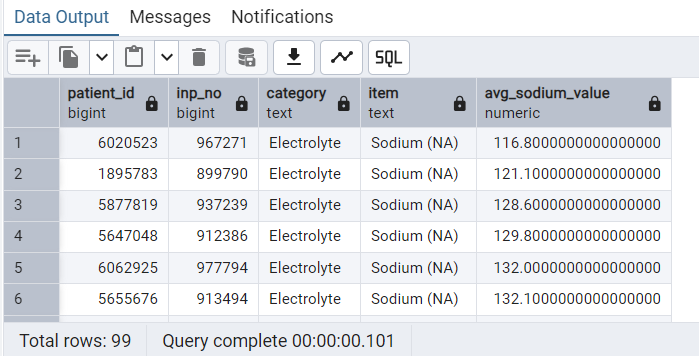
To Convert the String into Number Eg: AVG(CAST(la.labValue as decimal))

AVG(la.labvalue :: decimal)

AVG(la.labvalue :: double precision) We need

to use double precision instead double

**OUTPUT:**

****

66. For each department show the count of patients whose condition

got worse a year after discharge.

**QUERY:**

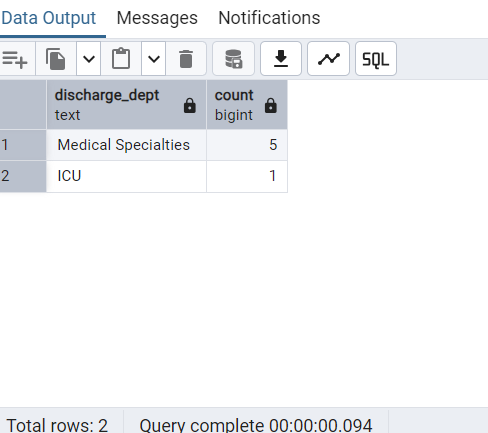
SELECT discharge\_dept, COUNT(patient\_id)

FROM outcome

WHERE sf36\_oneyearcomparehealthcondition LIKE '5\_Much worse now than one year ago%'

GROUP BY discharge\_dept;

**OUTPUT:**

****

67. Identify the patients who have the same systolic blood pressure values

recorded for more than two consecutive days.

**QUERY:**

WITH consecutive\_days\_bp AS (

SELECT

inp\_no,

DATE(charttime) AS reading\_date,

blood\_pressure\_high,

LAG(DATE(charttime)) OVER (PARTITION BY inp\_no ORDER BY DATE(charttime)) AS one\_day\_ago\_date,

LAG(blood\_pressure\_high) OVER (PARTITION BY inp\_no ORDER BY DATE(charttime)) AS one\_day\_ago\_bp\_high,

LAG(DATE(charttime), 2) OVER (PARTITION BY inp\_no ORDER BY DATE(charttime)) AS two\_days\_ago\_date,

LAG(blood\_pressure\_high, 2) OVER (PARTITION BY inp\_no ORDER BY DATE(charttime)) AS two\_days\_ago\_bp\_high

FROM nursingchart

WHERE blood\_pressure\_high IS NOT NULL

)

SELECT

inp\_no,

reading\_date,

blood\_pressure\_high,

one\_day\_ago\_date,

one\_day\_ago\_bp\_high,

two\_days\_ago\_date,

two\_days\_ago\_bp\_high

FROM consecutive\_days\_bp

WHERE

-- Ensure the dates are consecutive

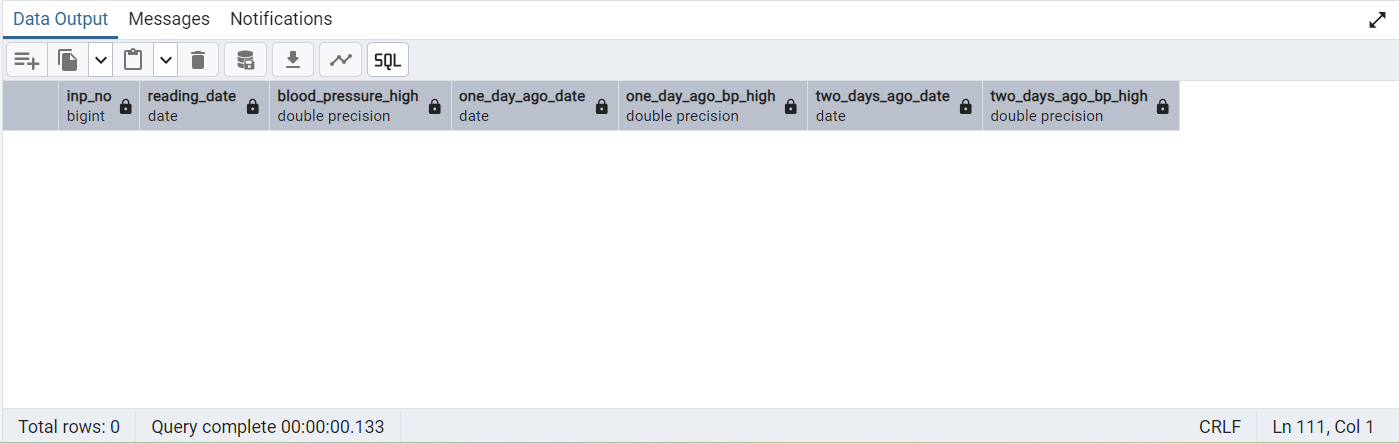
one\_day\_ago\_date = reading\_date - INTERVAL '1 day'

AND two\_days\_ago\_date = reading\_date - INTERVAL '2 day'

AND blood\_pressure\_high = one\_day\_ago\_bp\_high

AND blood\_pressure\_high = two\_days\_ago\_bp\_high -- Check if the consecutive days bp readings are same

**OUTPUT:**

****

68. Show the 9th youngest patient and if they are alive or not.

**QUERY:**

WITH ninth\_youngest\_patient AS (

SELECT patient\_id, age

FROM baseline

ORDER BY age ASC, patient\_id ASC

LIMIT 1

OFFSET 8

)

SELECT n.patient\_id,age,follow\_vital

FROM outcome o

JOIN ninth\_youngest\_patient n ON o.patient\_id = n.patient\_id;

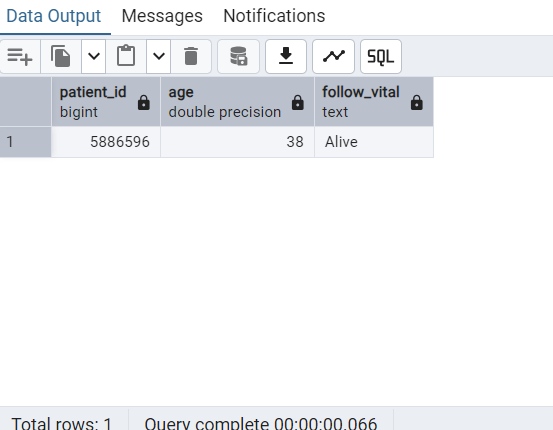
-- Limit 1 OFFSET 8 is used to take the 9 th row of the result set.

-- OFFSET says to skip that many rows before beginning to return rows.

-- LIMIT 1 : - Return the 1 row.

-- OFFSEET 8 : - It will skips the first 8 rows of the result set.

**OUTPUT:**

****

69. Show the bar distribution of ventilator modes for Pneumonia patients.

Hint: Do not consider null values.

**QUERY:**

SELECT n.breathing\_pattern, COUNT(DISTINCT patient\_id) AS patient\_count

FROM baseline b

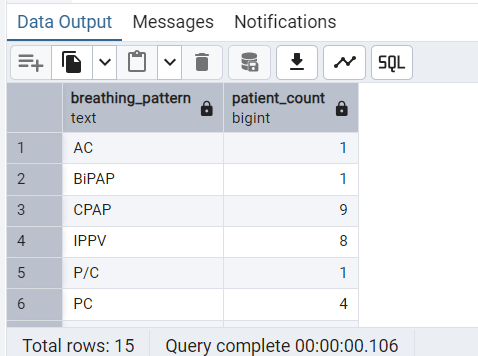
JOIN nursingchart n on b.inp\_no = n.inp\_no

WHERE infectionsite = 'Pneumonia'

AND breathing\_pattern IS NOT NULL

GROUP BY n.breathing\_pattern;

**OUTPUT:**

****

70. Create a view on baseline table with a check option on admit

department .

**QUERY - 1**

DROP VIEW IF EXISTS baseline\_view;

CREATE VIEW baseline\_view AS

SELECT patient\_id, inp\_no, age, sex, admitdept, infectionsite, icu\_discharge\_time

FROM baseline

WHERE admitdept = 'ICU'

WITH CHECK OPTION;

-- CHECK OPTION clause means to ensure that the changes to the base tables through

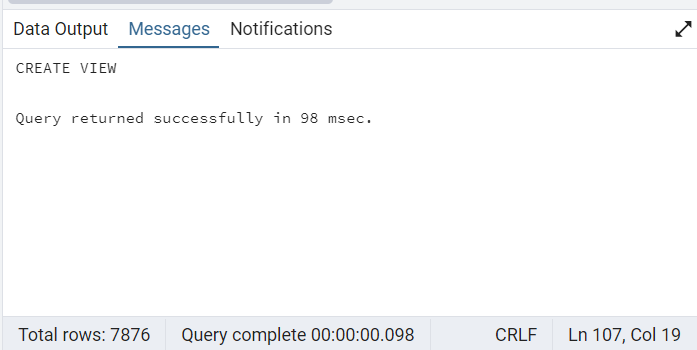
the view satisfy the view-defining condition.

--When you create a view WITH CHECK OPTION, PostgreSQL will ensure that you can

only modify data of the view that satisfies the condition in the view’s defining query

(query).

**OUTPUT - 1**

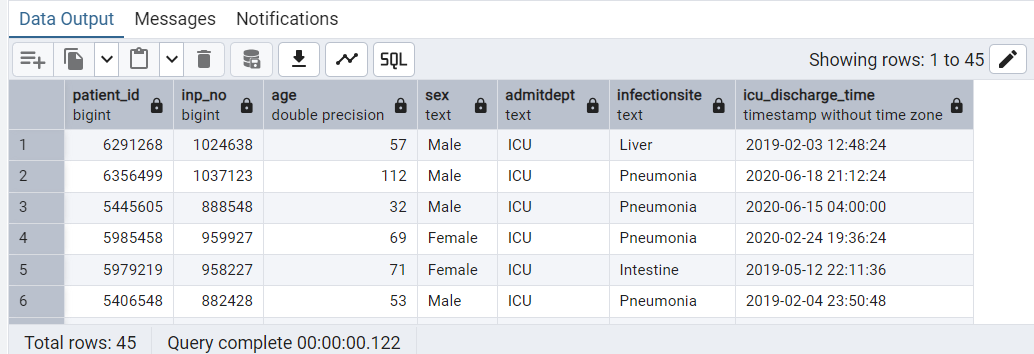
****

**QUERY - 2**

select \* from baseline\_view;

-- to view the baseline table only with admitdept= ICU, the created baseline\_view is successful.

**OUTPUT - 2**

****

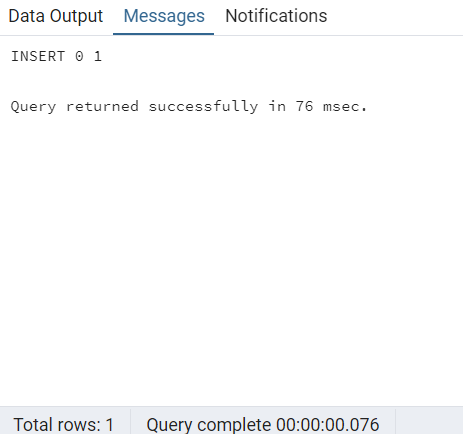
**QUERY - 3**

--Insert the data in to Baseline\_View values

INSERT Into baseline\_view (patient\_id, inp\_no, age, sex, admitdept, infectionsite, icu\_discharge\_time)

VALUES (50344432,3453344,45,'Female','ICU','Liver','2024-12-2 07:45:01');

**OUTPUT - 3**

****

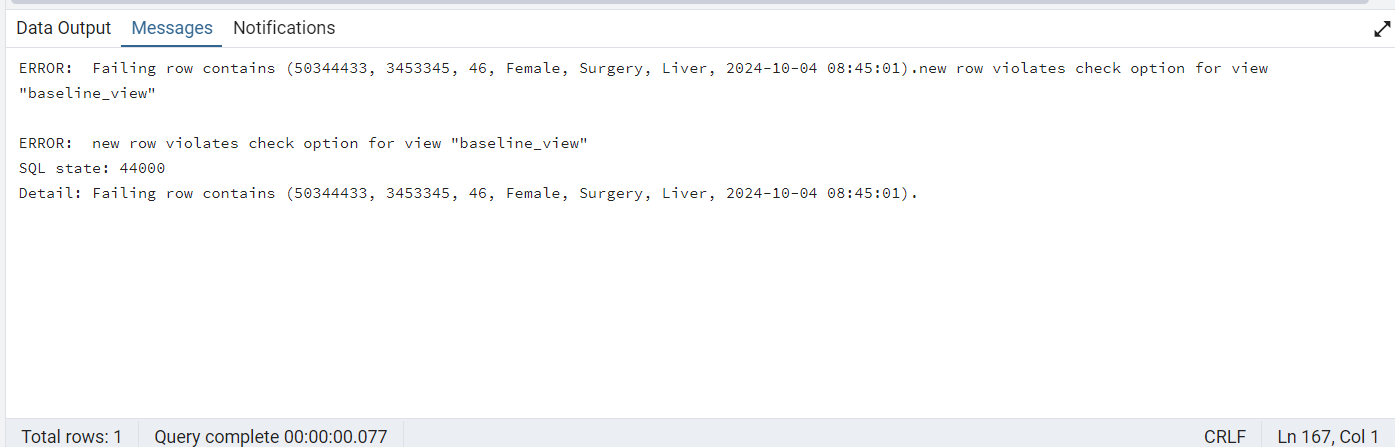
**QUERY - 4**

--Insert the invalid data in to Baseline\_View values

INSERT INTO baseline\_view (patient\_id, inp\_no, age, sex, admitdept, infectionsite, icu\_discharge\_time)

VALUES (50344433,3453345,46,'Female','Surgery','Liver','2024-10-4 08:45:01');

**OUTPUT - 4**

****

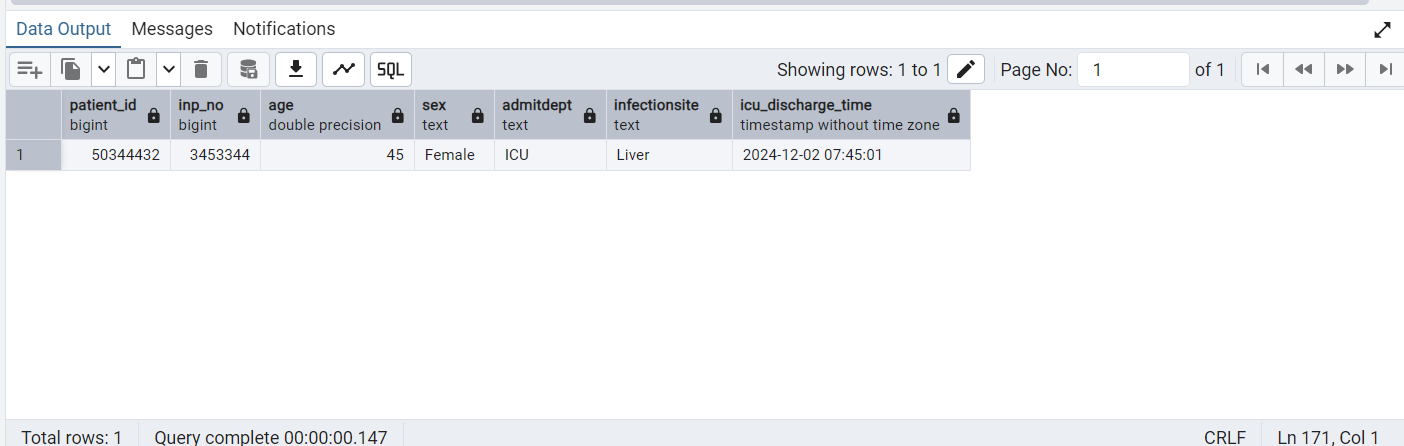
**QUERY - 5**

--Check Option after add the valid data.

SELECT \* from baseline\_view

WHERE patient\_id='50344432';

**OUTPUT - 5**

****

**QUERY - 6**

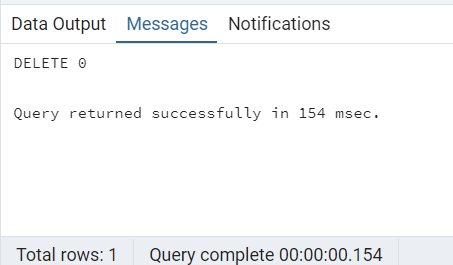
--Delete the patient\_id = '50344432' from the baseline\_view.

DELETE FROM baseline\_view

WHERE patient\_id = 50344432

AND inp\_no = 3453344;

**OUTPUT - 6**

****

71. How many patients were admitted to Surgery within 30 days of getting

discharged?

**QUERY:**

SELECT COUNT(DISTINCT b.patient\_id) AS patient\_count

FROM baseline b

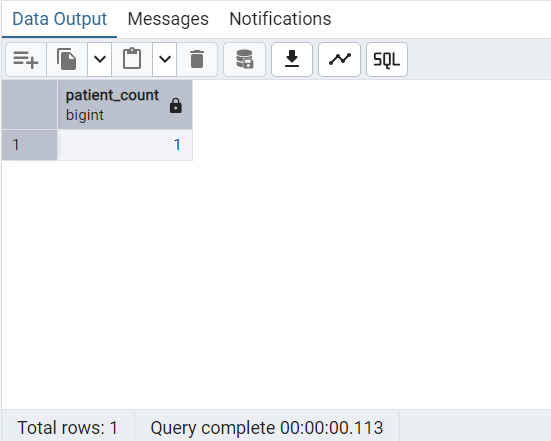
JOIN transfer t ON b.patient\_id = t.patient\_id

WHERE b.admitDept = 'Surgery'

AND t.starttime >= b.icu\_discharge\_time

AND t.starttime <= b.icu\_discharge\_time + INTERVAL '30 days';

**OUTPUT:**

****

72. What percentage of the patients with blood oxygen saturation < 90 are

alive?

**QUERY:**

WITH low\_oxygen\_patients AS (

SELECT b.patient\_id

FROM nursingchart nc

JOIN baseline b ON nc.inp\_no = b.inp\_no

WHERE nc.blood\_oxygen\_saturation < 90

),

low\_oxygen\_alive\_patients AS (

SELECT COUNT(DISTINCT lop.patient\_id) AS alive\_count

FROM low\_oxygen\_patients lop

JOIN outcome o ON lop.patient\_id = o.patient\_id

WHERE o.follow\_vital = 'Alive'

),

total\_low\_oxygen\_patients AS (

SELECT COUNT(DISTINCT patient\_id) AS total\_count

FROM low\_oxygen\_patients

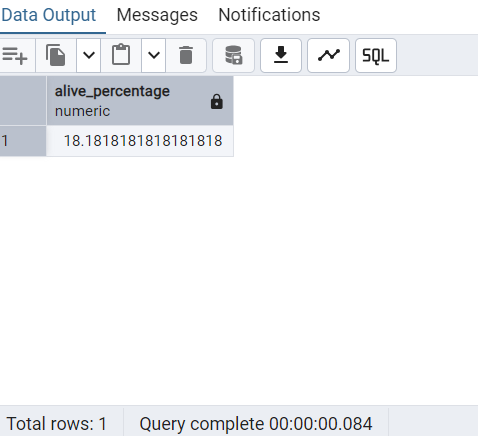
)

SELECT (a.alive\_count \* 100.0 / t.total\_count) AS alive\_percentage

FROM low\_oxygen\_alive\_patients a

CROSS JOIN total\_low\_oxygen\_patients t;

**OUTPUT:**

****

73. List the tables where column Patient\_ID is present.(display column

position number with respective table also)

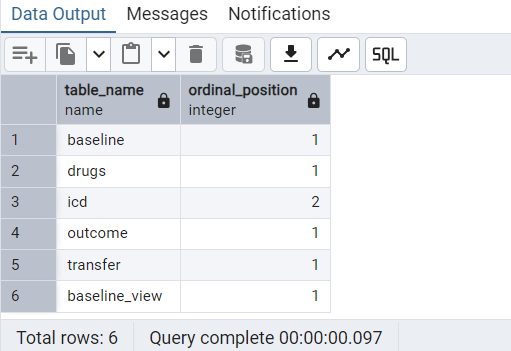
**QUERY:**

SELECT table\_name, ordinal\_position AS patient\_id\_column\_position

FROM information\_schema.columns

WHERE column\_name = 'patient\_id';

**OUTPUT:**

****

74. Find the average heart rate of patients under 40.

**QUERY:**

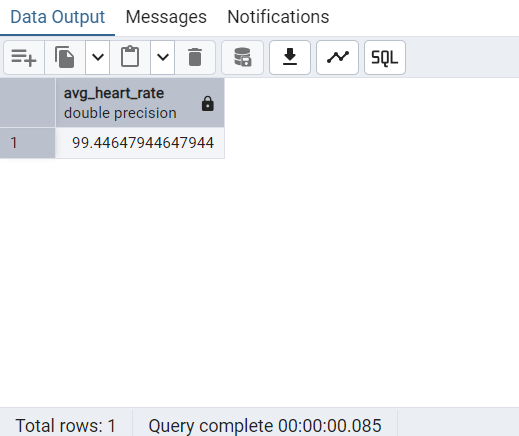
SELECT AVG(nc.heart\_rate) as avg\_heart\_rate

FROM nursingchart nc

JOIN baseline b on nc.inp\_no = b.inp\_no

WHERE b.age < 40;

**OUTPUT:**

****

75. Use CTE to calculate the percentage of patients with hypoproteinemia

who had to be intubated.

**QUERY:**

WITH patients\_with\_hypoproteinemia AS (

SELECT DISTINCT inp\_no

FROM lab

WHERE item LIKE '%Total protein (TP)%'

AND CAST(labvalue AS DOUBLE PRECISION) < 65 -- Patients with Total protein (TP) value less than the reference value 65 are taken as hypoproteinemia patients

),

patients\_with\_hypoproteinemia\_intubated AS (

SELECT DISTINCT n.inp\_no

FROM nursingchart n

JOIN patients\_with\_hypoproteinemia h ON n.inp\_no = h.inp\_no

WHERE endotracheal\_intubation IS NOT NULL -- Patients with hypoproteinemia who had to be intubated

)

SELECT

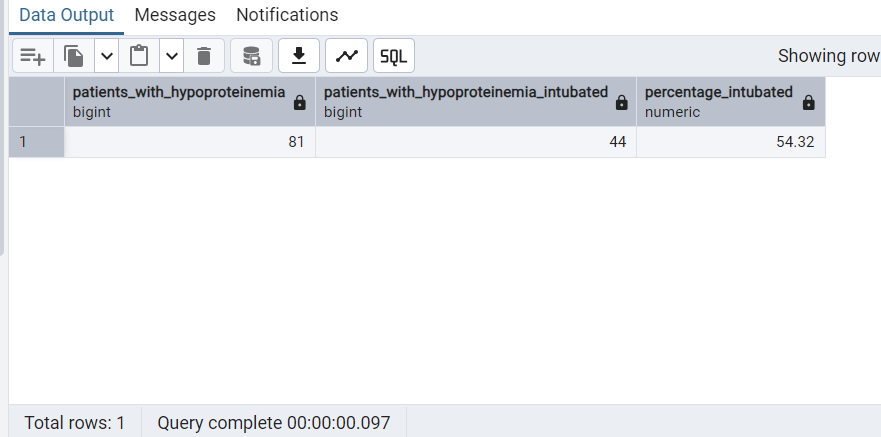
COUNT(DISTINCT pwh.inp\_no) AS patients\_with\_hypoproteinemia,

COUNT(DISTINCT hi.inp\_no) AS patients\_with\_hypoproteinemia\_intubated,

ROUND( COUNT(DISTINCT hi.inp\_no) \* 100.0 / COUNT(DISTINCT pwh.inp\_no), 2) AS percentage\_intubated

FROM patients\_with\_hypoproteinemia pwh

LEFT JOIN patients\_with\_hypoproteinemia\_intubated hi ON pwh.inp\_no = hi.inp\_no;

**OUTPUT:**

76. Identify patients whose breathing tube has been removed.

**QUERY:**

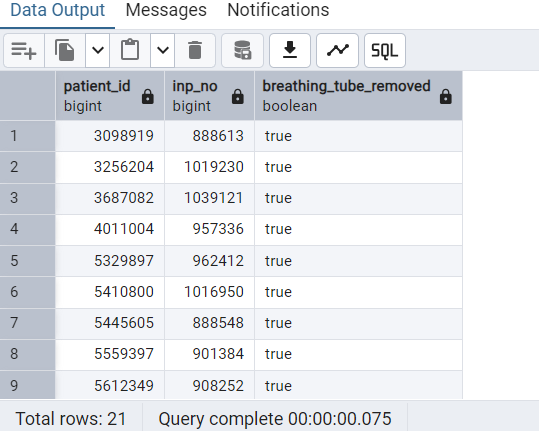
SELECT DISTINCT bl.patient\_id, nc.inp\_no, nc.extubation AS breathing\_tube\_removed

FROM nursingchart nc, baseline bl

WHERE bl.inp\_no = nc.inp\_no

AND nc.extubation = 'true'

ORDER BY bl.patient\_id;

**OUTPUT:**

77. Compare each diastolic blood pressure value with the previous reading.

And show previous and current value.

**QUERY:**

SELECT bl.patient\_id,

nc.inp\_no,

nc.invasive\_diastolic\_blood\_pressure AS blood\_pressure\_current\_value,

LAG(nc.invasive\_diastolic\_blood\_pressure)

OVER(

PARTITION BY bl.patient\_id

ORDER BY nc.inp\_no) AS blood\_pressure\_previous\_value

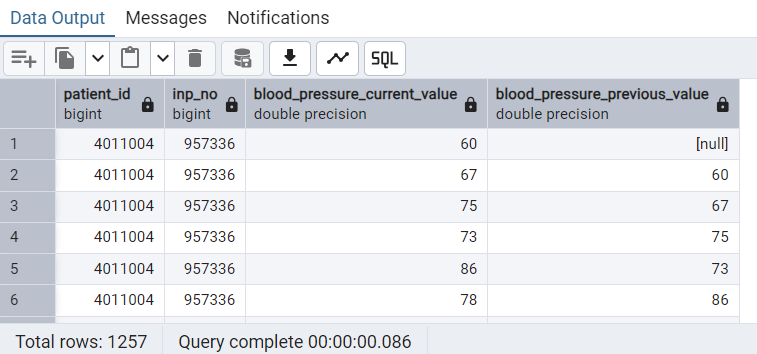
FROM nursingchart nc

JOIN baseline bl ON bl.inp\_no = nc.inp\_no

WHERE nc.invasive\_diastolic\_blood\_pressure IS NOT NULL

ORDER BY bl.patient\_id;

**OUTPUT:**

****

78. List patients who have more than 500 entries in the nursing chart.

**QUERY:**

SELECT bl.patient\_id, COUNT (nc.inp\_no) patient\_entries

FROM nursingchart nc

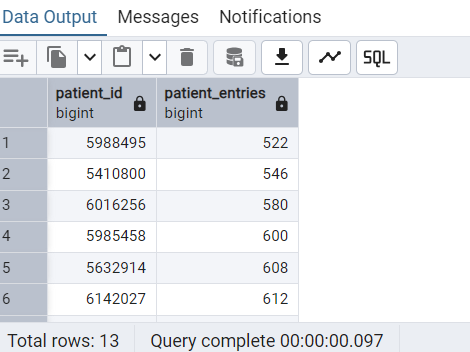
JOIN baseline bl ON nc.inp\_no = bl.inp\_no

GROUP BY bl.patient\_id

HAVING COUNT(nc.inp\_no) > 500

ORDER BY patient\_entries;

**OUTPUT:**

****

79. Display month name and the number of patients discharged from the

ICU in that month.

**QUERY:**

SELECT EXTRACT(MONTH FROM icu\_discharge\_time) AS month,

TO\_CHAR(bl.icu\_discharge\_time,'Month') AS month\_name,

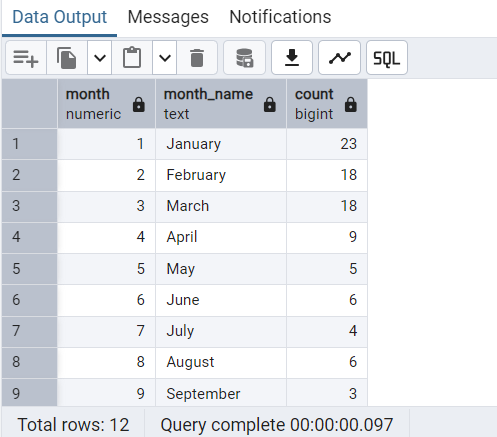
COUNT(bl.patient\_id)

FROM baseline bl

GROUP BY EXTRACT(MONTH FROM icu\_discharge\_time),TO\_CHAR(bl.icu\_discharge\_time,'Month')

ORDER BY month;

**OUTPUT:**

****

80. Write a function that calculates the percentage of people who had

moderate body pain after 4 weeks.

**QUERY:**

CREATE OR REPLACE FUNCTION calculate\_moderate\_pain\_percentage()

RETURNS NUMERIC AS $$

DECLARE

total\_patients INT;

moderate\_pain\_patients INT;

percentage NUMERIC;

BEGIN

-- Total number of patients

SELECT COUNT(DISTINCT patient\_id) INTO total\_patients

FROM outcome;

-- Number of patients with moderate pain after 4 weeks

SELECT COUNT(\*) INTO moderate\_pain\_patients

FROM outcome

WHERE SF36\_Pain\_BodyPainPast4wk ='4\_Moderate';

-- Calculate the percentage

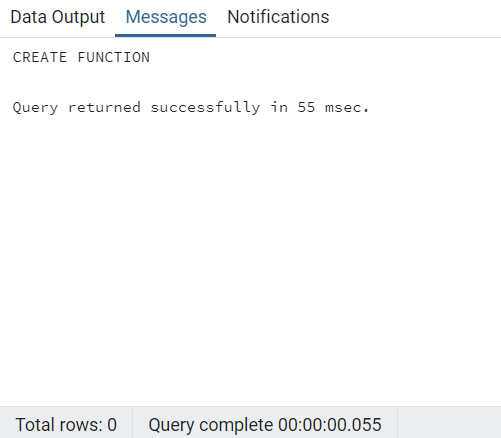
percentage := (moderate\_pain\_patients::NUMERIC / total\_patients) \* 100;

RETURN percentage;

END;

$$ LANGUAGE plpgsql;

**OUTPUT:**

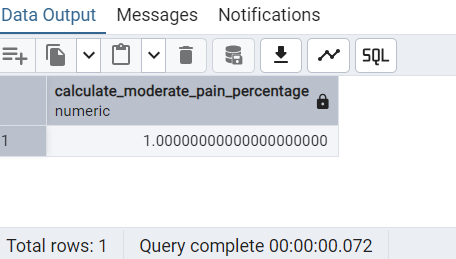
****

**QUERY:**

To view the function that calculates the percentage of people who had the moderate pain.

SELECT calculate\_moderate\_pain\_percentage();

**OUTPUT:**

****