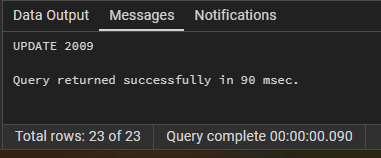
# **Team 29 – Primary Key Players**

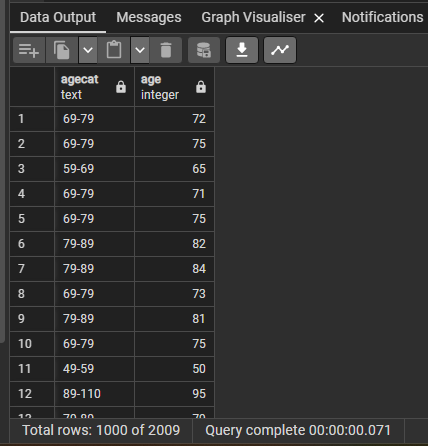
# **Main 80 Questions**

1. Update the demography table. Add a random age for each patient that falls within their respective age category. This newly added age should be an integer.

--Add New Column 'age' to demography table  
ALTER TABLE IF EXISTS public.demography  
ADD COLUMN age INTEGER;  
  
--UPDATE 'age' with random age for each patiente based on 'agecat'  
UPDATE demography  
SET age =FLOOR(CAST(LEFT(agecat, 2) AS INTEGER) + (RANDOM() \* 7));

Output:

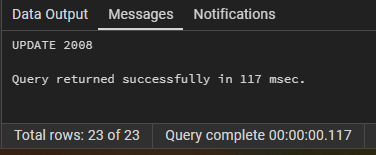




1. Calculate patient's year of birth using admission date from the hospitalization\_discharge and add to the demography table.

--Add column 'year\_of\_birth' to demography table  
ALTER TABLE IF EXISTS public.demography  
ADD COLUMN year\_of\_birth INTEGER;  
  
--calculate Update column 'year\_of\_birth' in demograhy table using 'admissin\_date' column in hospitilaition\_date' table   
UPDATE demography d  
SET year\_of\_birth = (extract(year from h.admission\_date)) - d.age  
FROM hospitalization\_discharge h  
WHERE h.inpatient\_number=d.inpatient\_number

Output:

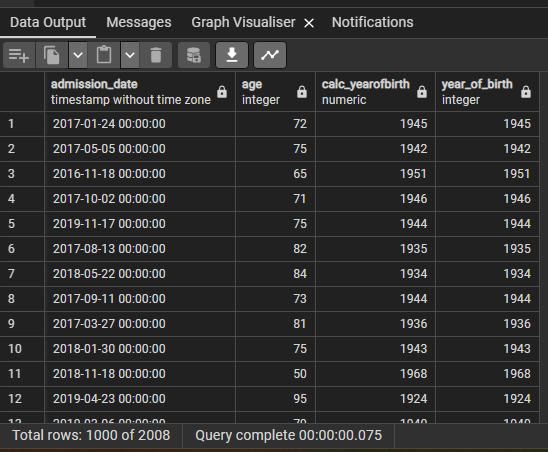


select h.admission\_date,d.age,((extract(year from h.admission\_date)) - d.age) as calc\_yearofbirth, d.year\_of\_birth

FROM hospitalization\_discharge h

JOIN demography d

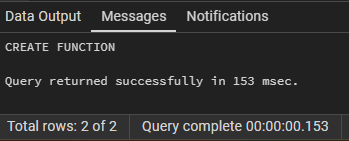
ON h.inpatient\_number=d.inpatient\_number



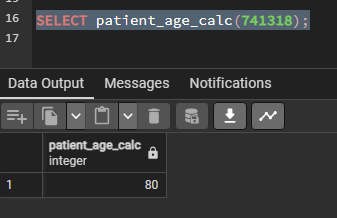
1. Create a User defined function that returns the age in years of any patient as a calculation from year of birth

CREATE OR REPLACE FUNCTION patient\_age\_calc(inpatient\_id bigint) RETURNS INTEGER  
LANGUAGE plpgsql AS $$  
DECLARE  
patient\_age INTEGER;  
BEGIN  
SELECT EXTRACT(YEAR FROM CURRENT\_DATE) - year\_of\_birth into patient\_age  
from public.demography  
where inpatient\_number = inpatient\_id;  
IF NOT FOUND THEN  
RAISE EXCEPTION 'Patient Id % Not found',inpatient\_id;  
END IF;  
RETURN patient\_age;  
END;  
$$;

Output:

  
  
SELECT patient\_age\_calc(741318);

Output:



1. What % of the dataset is male vs female?

select

COALESCE(gender,'Unknown') as gender\_category,

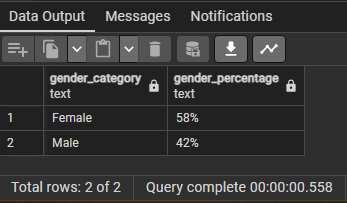
CONCAT(ROUND(count(gender) \* 100.0 / SUM(COUNT(\*)) OVER (),0),'%') AS gender\_percentage

from public.demography

where gender != 'Unknown'

group by gender\_category

Output:

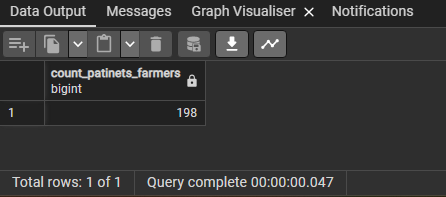


1. How many patients in this dataset are farmers?

SELECT COUNT(inpatient\_number) as count\_patinets\_farmers FROM demography

WHERE occupation = 'farmer';

Output:



1. Group the patients by age category and display it as a pie chart

select agecat, count(agecat)

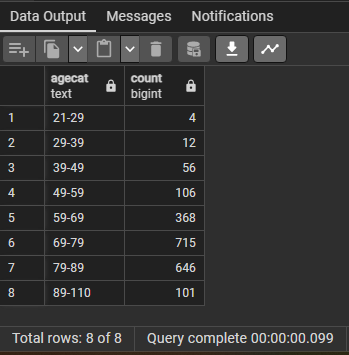
from public.demography

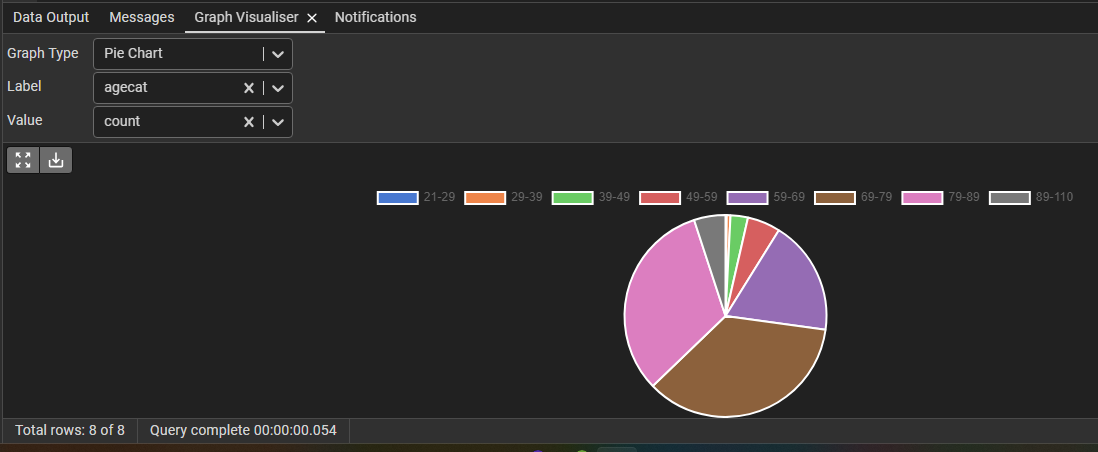
where agecat IS NOT NULL

group by agecat

order by agecat ASC

Output:





1. Divide BMI into slabs of 5 and show the count of patients within each one, without using case statements.

WITH BMI AS

(select ROUND(CAST(BMI AS numeric),2) as bmi\_rounded,count(bmi) as bmi\_count

from public.demography

group by bmi\_rounded

order by bmi\_rounded ASC)

SELECT

width\_bucket(bmi\_rounded,array[0,18.5,25,30,35]) as bmi\_bucket,

int4range(CAST(min(BMI\_rounded) as integer),CAST(max(BMI\_rounded) as integer),'()')as starting\_ending\_range,

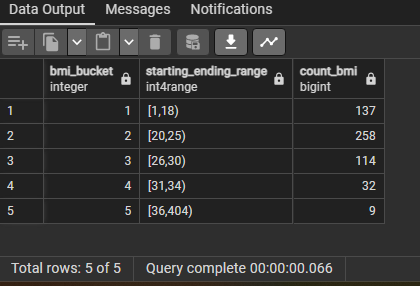
COUNT(bmi\_count) as count\_bmi

FROM BMI

GROUP BY bmi\_bucket

ORDER BY bmi\_bucket ASC

Output:



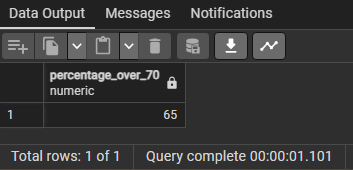
1. What % of the dataset is over 70 years old

SELECT ROUND(COUNT(age)\*100/(select count(inpatient\_number) from public.demography),0) as percentage\_over\_70

FROM public.demography

where age >70

Output:



1. What age group was least likely to be readmitted within 28 days

WITH CTE AS

(select d.agecat as least\_likely\_readmission\_agecat\_within\_28\_days,

COUNT(hd.re\_admission\_within\_28\_days)

from public.hospitalization\_discharge hd

JOIN public.demography d

ON hd.inpatient\_number = d.inpatient\_number

group by d.agecat

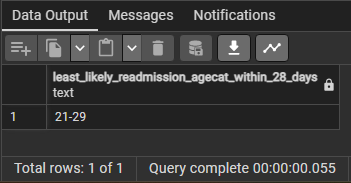
order by COUNT(hd.re\_admission\_within\_28\_days) ASC

LIMIT 1)

SELECT least\_likely\_readmission\_agecat\_within\_28\_days

from CTE

Output:



1. Create a procedure to insert a column with a serial number for all rows in demography.

CREATE OR REPLACE PROCEDURE public.generate\_series()

LANGUAGE 'sql'

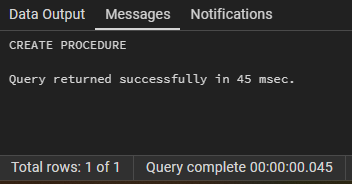
AS $BODY$

ALTER TABLE IF EXISTS public.demography

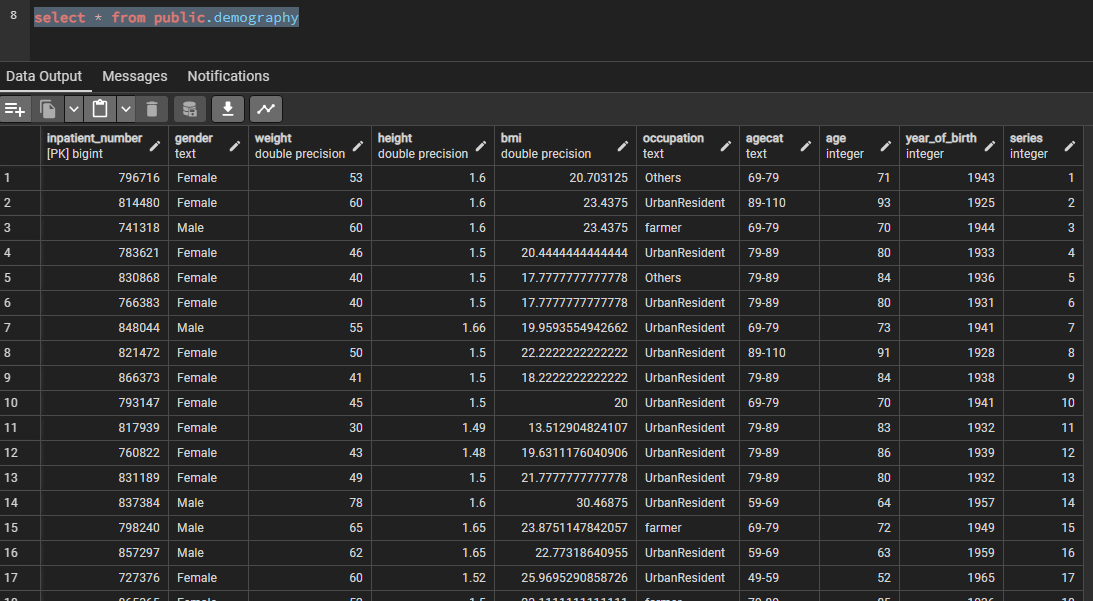
ADD COLUMN series SERIAL;

$BODY$;

Output:



Select \* from public.demography; -- Shows Series column with values generated:



1. what was the average time to readmission among men?

select ROUND(AVG(hd.readmission\_time\_days\_from\_admission)) as Avg\_readmission\_time\_days

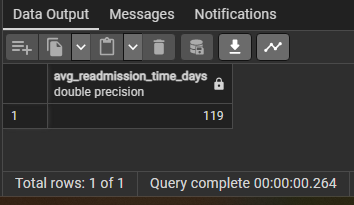
from public.hospitalization\_discharge hd

JOIN public.demography d

ON hd.inpatient\_number = d.inpatient\_number

AND d.gender='Male'

Output:



1. Display NYHA\_cardiac\_function\_classification as
   1. Class I: No symptoms of heart failure
   2. Class II: Symptoms of heart failure with moderate exertion
   3. Class III: Symptoms of heart failure with minimal exertion and show the most common type of heart failure for each classification

WITH common\_heart\_failure\_classification AS

(WITH classification AS

(select

CASE

WHEN NYHA\_cardiac\_function\_classification = 1 THEN 'Class I: No symptoms of heart failure'

WHEN NYHA\_cardiac\_function\_classification = 2 THEN 'Class II: Symptoms of heart failure with moderate exertion'

WHEN NYHA\_cardiac\_function\_classification = 3 THEN 'Class III: Symptomsof heart failure with minimal exertion'

ELSE 'Unknown'

END AS NYHA\_classification, type\_of\_heart\_failure,

count(inpatient\_number) as count\_patinet\_id

from public.cardiaccomplications

group by NYHA\_cardiac\_function\_classification, type\_of\_heart\_failure

ORDER BY NYHA\_classification, count\_patinet\_id DESC)

SELECT NYHA\_classification, type\_of\_heart\_failure,

RANK() OVER (PARTITION BY NYHA\_classification ORDER BY count\_patinet\_id DESC) as ranking

from classification

group by NYHA\_classification, type\_of\_heart\_failure, count\_patinet\_id)

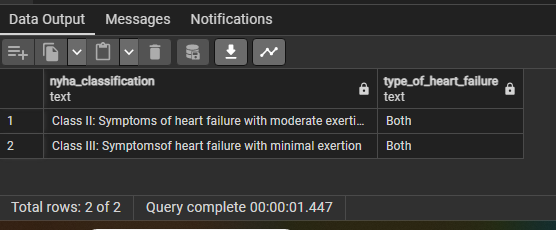
SELECT NYHA\_classification, type\_of\_heart\_failure

from common\_heart\_failure\_classification

where ranking =1

and NYHA\_classification !='Unknown'

Output:



1. Identify any columns relating to echocardiography and create a severity score for cardiac function. Add this column to the table

**Below code computes severity score for each of the columns relating to echocardiography:**

**Columns identified:** lvef, left\_ventricular\_end\_diastolic\_diameter\_lv, ea, mitral\_valve\_ems, mitral\_valve\_ams, tricuspid\_valve\_return\_velocity, tricuspid\_valve\_return\_pressure

**Scoring range:**

**0 – Good**

**1 - Normal**

**2 - Mild**

**3 - Moderate**

**4 – Severe**

**Total Severity score range:**

| **Total Score** | **Severity** |
| --- | --- |
| **0-3** | **Low Risk** or **Normal** |
| **4-6** | **Mild Dysfunction** |
| **7-9** | **Moderate Dysfunction** |
| **10-12** | **Severe Dysfunction** |
| **13+** | **Dangerous** |

WITH cardiac\_severity\_score AS

(SELECT inpatient\_number,

CASE

WHEN lvef >70 THEN 0

WHEN lvef BETWEEN 50 AND 70 THEN 1

WHEN lvef BETWEEN 40 AND 49 THEN 2

WHEN lvef BETWEEN 30 AND 39 THEN 3

WHEN lvef < 30 THEN 4

ELSE 0

END AS lvef\_score,

CASE

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv <37 THEN 0

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv BETWEEN 37 AND 56 THEN 1

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv BETWEEN 57 AND 61 THEN 2

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv BETWEEN 62 AND 65 THEN 3

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv < 65 THEN 4

ELSE 0

END AS lvedd\_score,

CASE

WHEN ea >0.8 THEN 0

WHEN ea BETWEEN 0.8 AND 1.5 THEN 1

WHEN ea BETWEEN 0.6 AND 0.8 THEN 2

WHEN ea BETWEEN 0.4 AND 0.6 THEN 3

WHEN ea < 0.4 THEN 4

ELSE 0

END AS ea\_score,

CASE

WHEN mitral\_valve\_ems <0.6 THEN 0

WHEN mitral\_valve\_ems BETWEEN 0.6 AND 0.9 THEN 1

WHEN mitral\_valve\_ems BETWEEN 1.0 AND 1.2 THEN 2

WHEN mitral\_valve\_ems BETWEEN 1.3 AND 1.5 THEN 3

WHEN mitral\_valve\_ems > 1.5 THEN 4

ELSE 0

END AS mvems\_score,

CASE

WHEN mitral\_valve\_ams >=0.12 THEN 0

WHEN mitral\_valve\_ams BETWEEN 0.09 AND 0.12 THEN 1

WHEN mitral\_valve\_ams BETWEEN 0.07 AND 0.09 THEN 2

WHEN mitral\_valve\_ams BETWEEN 0.05 AND 0.07 THEN 3

WHEN mitral\_valve\_ams < 0.05 THEN 4

ELSE 0

END AS mvams\_score,

CASE

WHEN tricuspid\_valve\_return\_velocity <=2.8 THEN 0

WHEN tricuspid\_valve\_return\_velocity =2.9 THEN 1

WHEN tricuspid\_valve\_return\_velocity BETWEEN 3.0 AND 3.1 THEN 2

WHEN tricuspid\_valve\_return\_velocity BETWEEN 3.2 AND 3.4 THEN 3

WHEN tricuspid\_valve\_return\_velocity >= 3.5 THEN 4

ELSE 0

END AS tvrv\_score,

CASE

WHEN tricuspid\_valve\_return\_pressure <=25 THEN 0

WHEN tricuspid\_valve\_return\_pressure BETWEEN 26 AND 35 THEN 1

WHEN tricuspid\_valve\_return\_pressure BETWEEN 36 AND 50 THEN 2

WHEN tricuspid\_valve\_return\_pressure BETWEEN 51 AND 70 THEN 3

WHEN tricuspid\_valve\_return\_pressure < 70 THEN 4

ELSE 0

END AS tvrp\_score

from public.cardiaccomplications)

SELECT

inpatient\_number,

lvef\_score,

lvedd\_score,

ea\_score,

mvems\_score,

mvams\_score,

tvrv\_score,

tvrp\_score,

(lvef\_score +

lvedd\_score +

ea\_score +

mvems\_score +

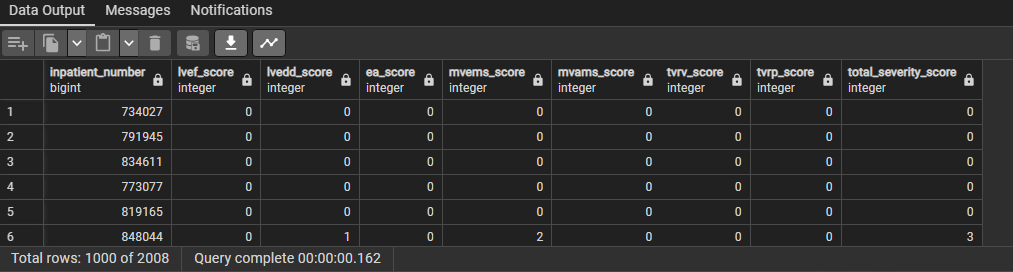
mvams\_score +

tvrv\_score+

tvrp\_score) as total\_severity\_score

from cardiac\_severity\_score

Output:



**Below statement adds a new column total\_severity\_score in cardiacomplications**

ALTER TABLE cardiaccomplications

ADD COLUMN total\_severity\_score INT;

**Below query updates total\_severity\_score score to cardiaccomplications**

UPDATE cardiaccomplications

SET total\_severity\_score =

(

CASE

WHEN lvef >70 THEN 0

WHEN lvef BETWEEN 50 AND 70 THEN 1

WHEN lvef BETWEEN 40 AND 49 THEN 2

WHEN lvef BETWEEN 30 AND 39 THEN 3

WHEN lvef < 30 THEN 4

ELSE 0

END +

CASE

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv <37 THEN 0

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv BETWEEN 37 AND 56 THEN 1

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv BETWEEN 57 AND 61 THEN 2

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv BETWEEN 62 AND 65 THEN 3

WHEN left\_ventricular\_end\_diastolic\_diameter\_lv < 65 THEN 4

ELSE 0

END +

CASE

WHEN ea >0.8 THEN 0

WHEN ea BETWEEN 0.8 AND 1.5 THEN 1

WHEN ea BETWEEN 0.6 AND 0.8 THEN 2

WHEN ea BETWEEN 0.4 AND 0.6 THEN 3

WHEN ea < 0.4 THEN 4

ELSE 0

END +

CASE

WHEN mitral\_valve\_ems <0.6 THEN 0

WHEN mitral\_valve\_ems BETWEEN 0.6 AND 0.9 THEN 1

WHEN mitral\_valve\_ems BETWEEN 1.0 AND 1.2 THEN 2

WHEN mitral\_valve\_ems BETWEEN 1.3 AND 1.5 THEN 3

WHEN mitral\_valve\_ems > 1.5 THEN 4

ELSE 0

END +

CASE

WHEN mitral\_valve\_ams >=0.12 THEN 0

WHEN mitral\_valve\_ams BETWEEN 0.09 AND 0.12 THEN 1

WHEN mitral\_valve\_ams BETWEEN 0.07 AND 0.09 THEN 2

WHEN mitral\_valve\_ams BETWEEN 0.05 AND 0.07 THEN 3

WHEN mitral\_valve\_ams < 0.05 THEN 4

ELSE 0

END +

CASE

WHEN tricuspid\_valve\_return\_velocity <=2.8 THEN 0

WHEN tricuspid\_valve\_return\_velocity =2.9 THEN 1

WHEN tricuspid\_valve\_return\_velocity BETWEEN 3.0 AND 3.1 THEN 2

WHEN tricuspid\_valve\_return\_velocity BETWEEN 3.2 AND 3.4 THEN 3

WHEN tricuspid\_valve\_return\_velocity >= 3.5 THEN 4

ELSE 0

END +

CASE

WHEN tricuspid\_valve\_return\_pressure <=25 THEN 0

WHEN tricuspid\_valve\_return\_pressure BETWEEN 26 AND 35 THEN 1

WHEN tricuspid\_valve\_return\_pressure BETWEEN 36 AND 50 THEN 2

WHEN tricuspid\_valve\_return\_pressure BETWEEN 51 AND 70 THEN 3

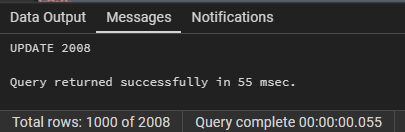
WHEN tricuspid\_valve\_return\_pressure < 70 THEN 4

ELSE 0

END )

WHERE inpatient\_number IS NOT NULL;

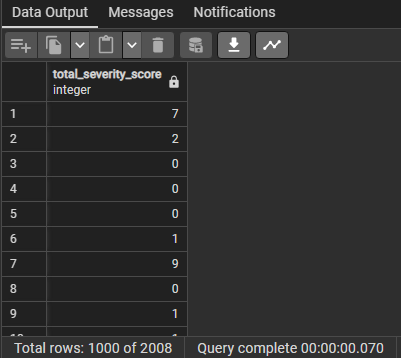
Output:



**Below is a test select query that show the score in total\_severity\_score from cardiaccomplications**

select total\_severity\_score from cardiaccomplications

Output:



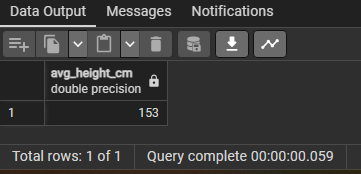
1. What is the average height of women in cms?

select ROUND(AVG(height\*100)) as avg\_height\_cm

from public.demography

where gender = 'Female'

Output:



1. Using the cardiac severity column from q13, find the correlation between hospital outcomes and cardiac severity

select

CORR(CASE WHEN hd.outcome\_during\_hospitalization = 'Dead' THEN 1 ELSE 0 END, CAST(cc.total\_severity\_score as integer)) as corr\_dead,

CORR(CASE WHEN hd.outcome\_during\_hospitalization = 'DischargeAgainstOrder' THEN 1 ELSE 0 END, CAST(cc.total\_severity\_score as integer)) as corr\_DAO,

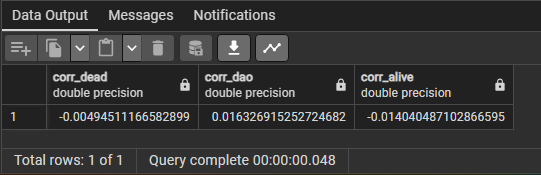
CORR(CASE WHEN hd.outcome\_during\_hospitalization = 'Alive' THEN 1 ELSE 0 END, CAST(cc.total\_severity\_score as integer)) as corr\_alive

from public.hospitalization\_discharge hd

JOIN public.cardiaccomplications cc

ON hd.inpatient\_number = cc.inpatient\_number

Output:



**-0.004945111665828978 corr\_dead Negative Corrolation**

**0.016326915252724727 corr\_DAO Very weak Correlation**

**-0.014040487102866631 corr\_alive Negative Corrolation**

Interpretation of Correlation Coefficient

0 to 0.1: Very weak positive correlation.

0.1 to 0.3: Weak positive correlation.

0.3 to 0.5: Moderate positive correlation.

0.5 to 0.7: Strong positive correlation.

0.7 to 1: Very strong positive correlation.

Negative values: Indicate a negative correlation, where as one variable increases, the other tends to decrease.

1. Show the no. of patients for everyday in March 2017. Show the date in March along with the days between the previous recorded day in march and the current.

SELECT

DATE(admission\_date) as Admission\_Date,

DATE(admission\_date) - LAG(DATE(admission\_date)) OVER (ORDER BY DATE(admission\_date)) as Difference\_in\_Days,

COUNT(inpatient\_number) as Patient\_Count

FROM public.hospitalization\_discharge

where

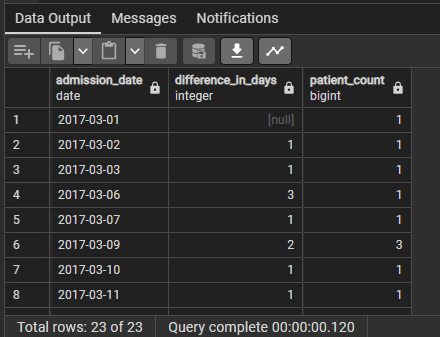
EXTRACT(YEAR FROM admission\_date) ='2017'

and EXTRACT(Month FROM admission\_date) ='03'

GROUP BY

DATE(admission\_date)

Output:



1. Create a view that combines patient demographic details of your choice along with pre-exisiting heart conditions like MI,CHF and PVD

create or replace view cardiac\_patients

as

select d.inpatient\_number, d.gender, d.occupation, d.age, c.myocardial\_infarction, c.congestive\_heart\_failure, c.peripheral\_vascular\_disease

from public.demography d

join cardiaccomplications c on c.inpatient\_number = d.inpatient\_number

where c.myocardial\_infarction = 1 or c.congestive\_heart\_failure = 1 or c.peripheral\_vascular\_disease = 1;

Output:

A screenshot of a computer

Description automatically generated

**Below is a query that selects all records from the view:**

select \* from cardiac\_patients;

Output:

A screenshot of a computer

Description automatically generated

1. Create a function to calculate total number of unique patients for every drug. Results must be returned as long as the first few characters match the user input.

create or replace function get\_count\_of\_prescribed(name\_of\_drug text)

returns table (count\_of\_users bigint, name\_of\_drugs text)

as

$$

declare first\_three\_letters text;

begin

first\_three\_letters =lower(left(name\_of\_drug,3));

return query

select count(inpatient\_number), drug\_name

from public.patient\_precriptions

where lower(drug\_name) like first\_three\_letters||'%'

group by drug\_name;

end

$$language plpgsql;

Output:

A screenshot of a computer

Description automatically generated

**Below queries are used to call the function with the drug name:**

select \* from get\_count\_of\_prescribed('furo');

Output:

A screenshot of a computer

Description automatically generated

select \* from get\_count\_of\_prescribed('a');

Output:

A screenshot of a computer

Description automatically generated

1. break up the drug names in patient\_precriptions at the ""spaces"" and display only the second string without using Substring. Show unique drug names along with newly broken up string

select distinct drug\_name, split\_part(drug\_name,' ',2) as second\_string

from patient\_precriptions;

Output:

A screenshot of a computer

Description automatically generated

1. Select the drug names starting with E and has x in any position after

select distinct drug\_name

from patient\_precriptions

where drug\_name like 'E%x%';

Output:

A screenshot of a computer

Description automatically generated

1. Create a cross tab to show the count of readmissions within 28 days, 3 months,6 months as rows and admission ward as columns

select \* from

crosstab(

'

SELECT ''180Days'' as day\_range,admission\_ward,SUM(re\_admission\_within\_6\_months) as count\_patients

FROM public.hospitalization\_discharge

GROUP BY hospitalization\_discharge.admission\_ward

UNION ALL

SELECT ''28Days'' as day\_range,admission\_ward,SUM(re\_admission\_within\_28\_days) as count\_patients

FROM public.hospitalization\_discharge

GROUP BY hospitalization\_discharge.admission\_ward

UNION ALL

SELECT ''90Days'' as day\_range,admission\_ward,SUM(re\_admission\_within\_3\_months) as count\_patients

FROM public.hospitalization\_discharge

GROUP BY hospitalization\_discharge.admission\_ward

UNION ALL

SELECT ''DWithin6Months'' as day\_range,admission\_ward,SUM(death\_within\_6\_months) as value

FROM public.hospitalization\_discharge

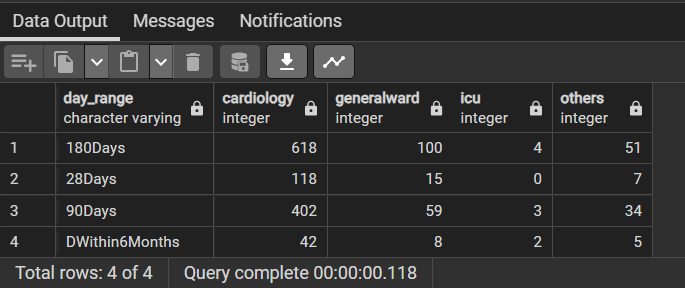
GROUP BY hospitalization\_discharge.admission\_ward

',

'SELECT DISTINCT admission\_ward FROM hospitalization\_discharge details ORDER BY admission\_ward ASC')

as adm\_columns(day\_range varchar,Cardiology int,GeneralWard int,ICU int, Others int);

Output:



1. Create a trigger to stop patient records from being deleted from the demography table

Query to create function:

create or replace function deletion\_not\_allowed()

returns trigger as

$$

begin

raise exception 'Deletion is not allowed on this table.';

return null;

end

$$ language plpgsql;

Output:

A screenshot of a computer

Description automatically generated

**Query to create trigger:**

create trigger no\_delete\_demography

before delete on public.demography

for each row execute function deletion\_not\_allowed();

Output:

A screenshot of a computer

Description automatically generated

**Query to test trigger by trying to delete a row from demography table:**

delete from demography where inpatient\_number = 5;

Output:

A screenshot of a computer screen

Description automatically generated

1. What is the total number of days between the earliest admission and the latest

select (max(admission\_date)::date-min(admission\_date)::date) as day\_interval\_between\_first\_and\_last\_admission

from hospitalization\_discharge;

Output:

A screenshot of a computer

Description automatically generated

1. Divide discharge day by visit times for any 10 patients without using mathematical operators like '/'

select inpatient\_number,dischargeday, visit\_times, div(dischargeday,visit\_times) as division\_result

from hospitalization\_discharge

limit 10;

Output:

A screenshot of a graph

Description automatically generated

1. Show the count of patients by first letter of admission\_way.

select substring(admission\_way,1,1) as first\_letter\_of\_admission\_way,count(inpatient\_number)

from hospitalization\_discharge

group by admission\_way;

Output:

A screenshot of a computer

Description automatically generated

1. Display an array of personal markers:gender, BMI, pulse, MAP for every patient. The result should look like this

select d.inpatient\_number, array[d.gender, cast(cast(d.bmi as decimal(10,2)) as text), cast(cast(l.pulse as decimal(10,2)) as text), cast(cast(l.map\_value as int) as text)] as markers

from demography d, labs l

where d.inpatient\_number = l.inpatient\_number;

Output:

A screenshot of a computer

Description automatically generated

1. Display medications With Name contains 'hydro' and display it as 'H20'.

**For this query we also considered drug names that contained ‘Hydro’ with capital H. Any drug name that contains ‘hydro’ or ‘Hydro’ will be replaced with H2O.**

select distinct drug\_name, replace(lower(drug\_name),'hydro','H2O') as modified\_drug\_name from patient\_precriptions

where drug\_name ilike '%hydro%';

Output:

A screenshot of a computer

Description automatically generated

1. Create a trigger to raise notice and prevent deletion of the view created in question 17

**Query to create function:**

create or replace function view\_deletion\_not\_allowed()

returns event\_trigger as

$$

declare

rec record;

begin

for rec in select \* from pg\_event\_trigger\_dropped\_objects()

loop

if rec.object\_type = 'view' AND rec.object\_name = 'cardiac\_patients' then

raise notice 'Deletion of view cardiac\_patients is not allowed';

end if;

end loop;

end;

$$language plpgsql;

Output:

A screenshot of a computer

Description automatically generated

**Query to create trigger:**

create event trigger cardiac\_patients\_view\_no\_deletion on sql\_drop

when TAG in ('DROP VIEW')

execute function view\_deletion\_not\_allowed();

Output:

A screenshot of a computer

Description automatically generated

**Testing the trigger by trying to drop the view cardiac\_patients:**

drop view cardiac\_patients;

Output:

A screenshot of a computer

Description automatically generated

1. How many unique patients have cancer?

**For this query, we have considered patients having leukemia or malignant lymphoma as having cancer. Solid tumor patients have not been considered as tumors may or may not be cancerous.**

select distinct count(inpatient\_number) as count\_of\_patients\_having\_cancer

from patienthistory

where leukemia = 1 or malignant\_lymphoma = 1;

Output:

A screenshot of a computer

Description automatically generated

1. Show the moving average of number of patient admitted every 3 months.

select date\_trunc('month', admission\_date) as ma\_month,

count(inpatient\_number) as patients\_admitted,

round((avg(count(inpatient\_number)) over (order by date\_trunc('month', admission\_date) rows between 2 preceding and current row)),2) as moving\_average\_3\_month

from hospitalization\_discharge

group by date\_trunc('month', admission\_date)

order by ma\_month;

Output:

A screenshot of a graph

Description automatically generated

1. Write a query to get a list of patient IDs' who recieved oxygen therapy and had a high respiration rate in February 2017

**We have considered respiration rate > 20 breaths per minute as high respiration for this query.**

select distinct hd.inpatient\_number

from hospitalization\_discharge hd, labs l

where hd.inpatient\_number = l.inpatient\_number

and l.respiration >20

and hd.oxygen\_inhalation = 'OxygenTherapy'

and extract(month from admission\_date) = 02

and extract(year from admission\_date) = 2017;

Output:

A screenshot of a computer

Description automatically generated

1. Display patients with heart failure type: "both" along with highest MAP and higest pulse without using limit

**For this question, we are considering the patients with heart failure type: “both” and amongst them we are displaying the patient with highest MAP and the patient with highest pulse.**

select cc.inpatient\_number, cc.type\_of\_heart\_failure, l.map\_value,l.pulse

from cardiaccomplications cc

JOIN labs l

ON cc.inpatient\_number = l.inpatient\_number

where cc.type\_of\_heart\_failure ='Both'

and (l.map\_value = (select max(map\_value) from labs)

or l.pulse = (select max(pulse) from labs))

Output:

A screenshot of a computer

Description automatically generated

1. Create a stored procedure that displays any message on the screen without using any tables/views.

**creating stored procedure without variable:**

create or replace procedure stored\_procedure()

language plpgsql

as $$

--declaring variable to store the value that is returning from block

declare team text;

begin

select 'Team\_29 Primary Key Players' into team;

--raise notice is used to show the output while calling

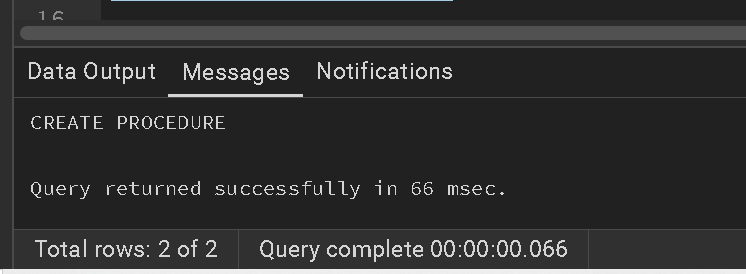
raise notice '%',team;

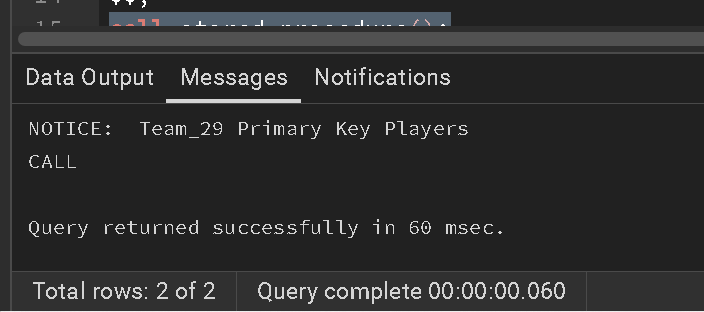
end;

$$;

– calling procedure

call stored\_procedure();

Output:



1. In healthy people, monocytes make up about 1%-9% of total white blood cells. Calculate avg monocyte percentages among each age group

select

agecat,

round(avg((monocyte\_count::numeric/white\_blood\_cell::numeric)\*100),2) as monocyte\_percentage

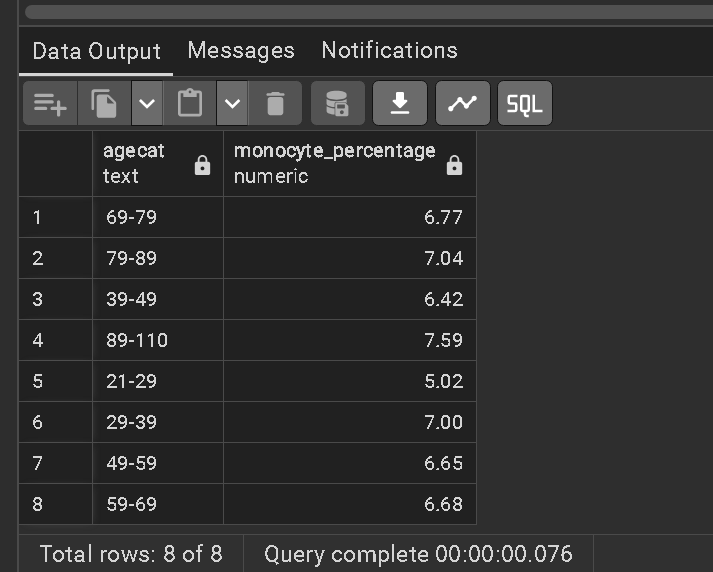
from labs lb

join demography dm

on lb.inpatient\_number=dm.inpatient\_number

group by dm.agecat

Output:

****

1. Create a table that stores any Patient Demographics of your choice as the parent table. Create a child table that contains systolic\_blood\_pressure,diastolic\_blood\_pressure per patient and inherits all columns from the parent table

**droping tables if exists:**

drop table blood\_pressure;

drop table BP\_child ;

**parent table:**

create table blood\_pressure

(inpatient\_number bigint PRIMARY KEY,

gender text,

bmi double precision,

agecat text);

**child table:**

CREATE TABLE BP\_child (

systolic\_blood\_pressure INTEGER,

diastolic\_blood\_pressure INTEGER

) INHERITS (blood\_pressure);

**insert values:**

iNSERT INTO BP\_child( inpatient\_number,gender,bmi,agecat,systolic\_blood\_pressure,diastolic\_blood\_pressure)

select

dm.inpatient\_number,

gender,

bmi,

agecat,

systolic\_blood\_pressure,

diastolic\_blood\_pressure

from demography dm

join labs lb on

dm.inpatient\_number=lb.inpatient\_number

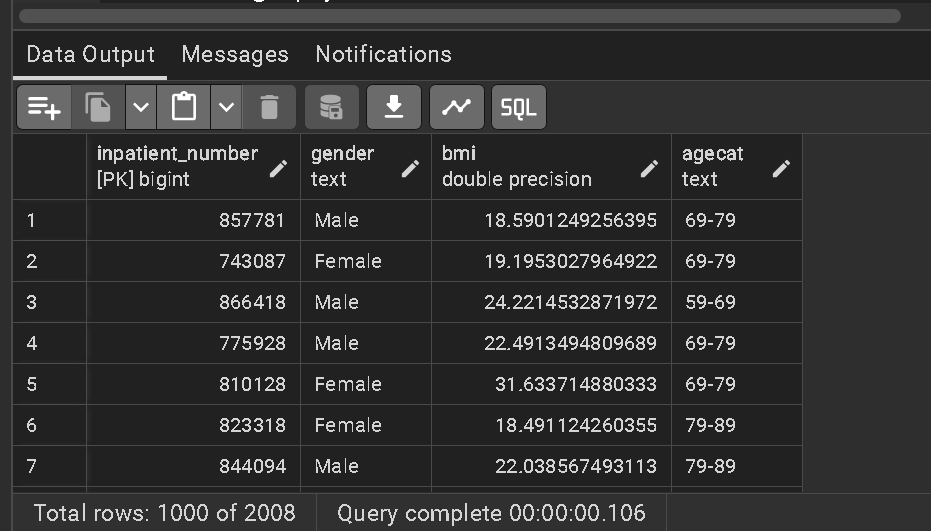
**checking the tables:**

select\* from BP\_child

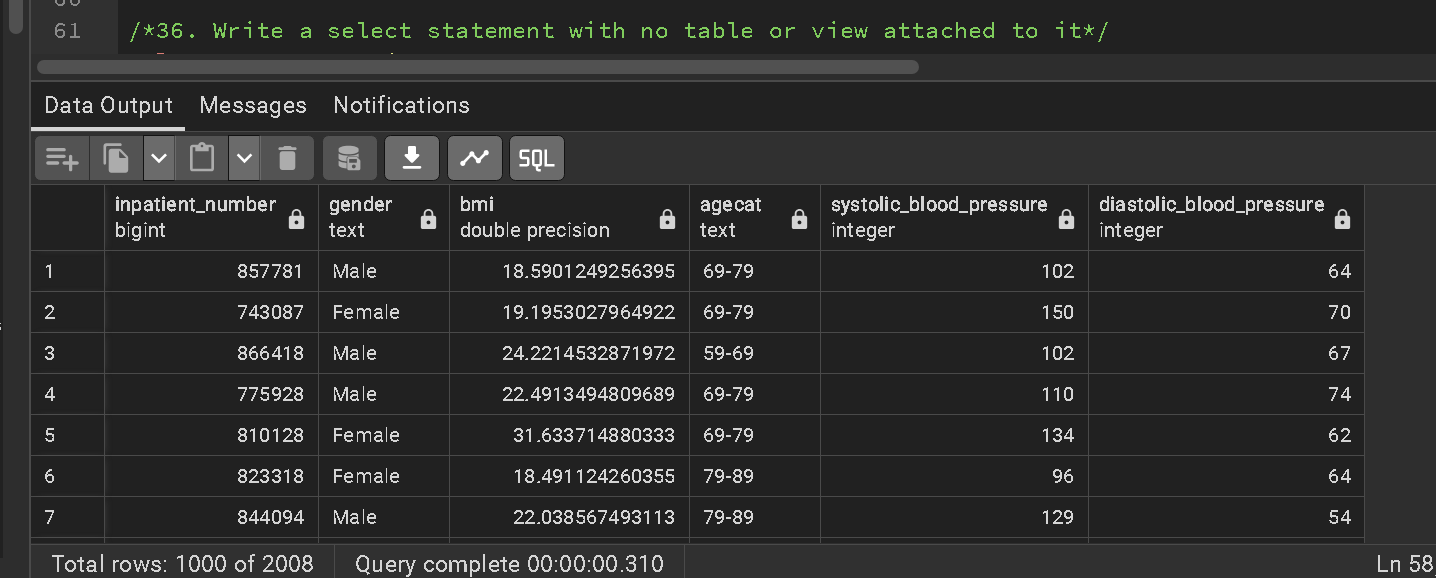
select\* from blood\_pressure

Output:

**Parent table:**



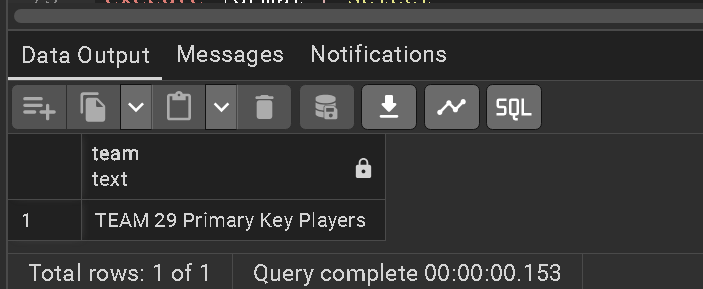
**Child table:**



1. Write a select statement with no table or view attached to it

select 'TEAM 29 Primary Key Players' as team;

Output:



1. Create a re-usable function to calculate the percentage of patients for any group. Use this function to calculate % of patients in each admission ward."

**create the reusable function:**

create or replace function percentage\_patients(tablename text ,group\_column text)

returns table (percentage numeric ,count bigint ,column\_group text) as

$$

begin

return query

/\*-- this function was unable to take the tablename and column name so used %I as

a placeholder for table name and column name\*/

execute format ('select

round((count(inpatient\_number)\*100.0/(select count(inpatient\_number) from %I)),2) as percentage,

count(inpatient\_number) as count\_by\_group,

%I

from %I

group by %I',

tablename,

group\_column,

tablename,

group\_column);

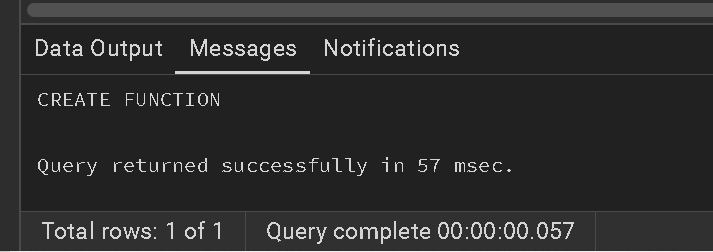
end;

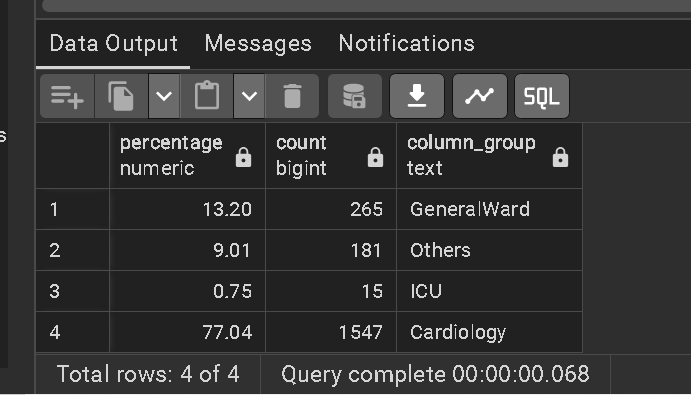
$$ language plpgsql

**calling function:**

select \* from percentage\_patients('hospitalization\_discharge','admission\_ward')

Output:

****

****

1. Write a query that shows if CCI score is an even or odd number for any 10 patients.

select

inpatient\_number,

cci\_score,

case

when(cci\_score::integer % 2)=0 then 'Even Number'

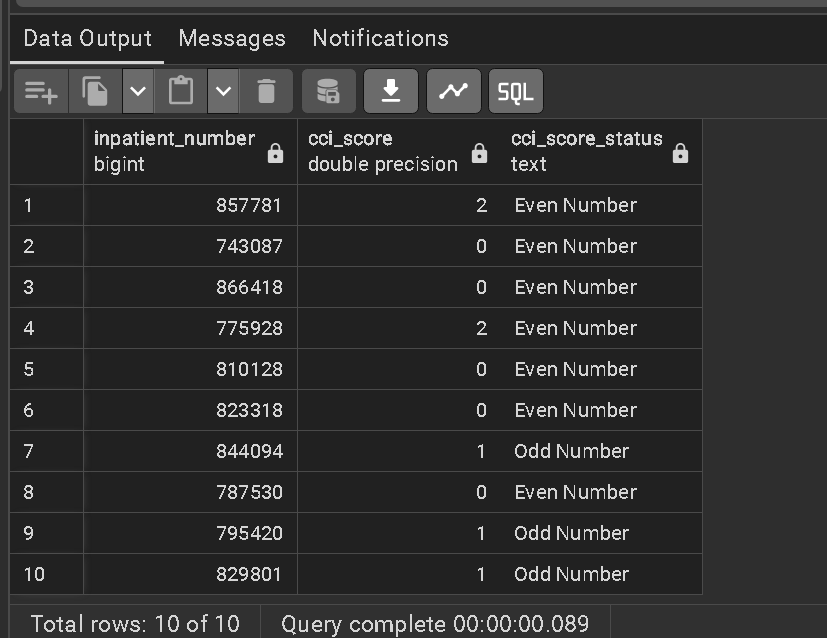
else 'Odd Number' end

as cci\_score\_status

from patienthistory

limit 10;

Output:



1. Using windows functions show the number of hospitalizations in the previous month and the next month.

with monthly\_counts as (

select

count(inpatient\_number) as number\_patients,

extract(month from admission\_date) as months

from hospitalization\_discharge

group by extract(month from admission\_date)

)

select

months,

number\_patients,

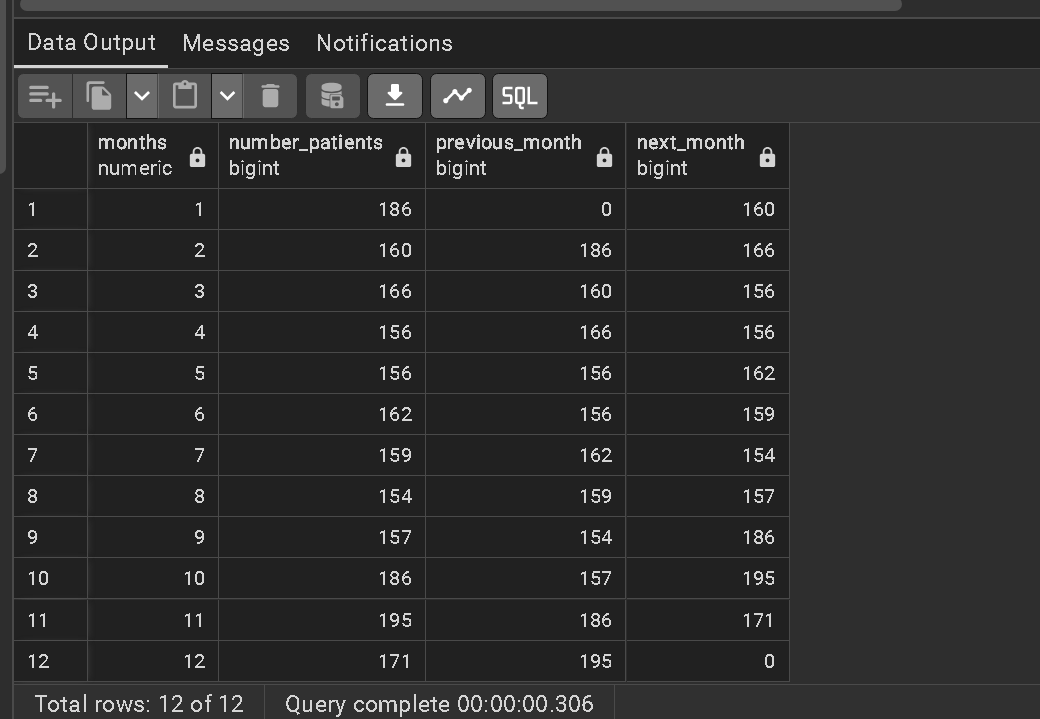
lag(number\_patients,1,0)over(order by months) as previous\_month,

lead(number\_patients,1,0)over(order by months) as next\_month

from monthly\_counts

order by months

Output:

****

1. Write a function to get comma-separated values of patient details based on patient number entered by the user. (Use a maximum of 6 columns from different tables).

**creating a function**

create or replace function comma\_separated\_patient\_details()

returns table (

patient\_details text

)

as $$

begin

return query

Select

concat\_ws(',',dm.inpatient\_number,dm.gender,cc.type\_of\_heart\_failure,hd.outcome\_during\_hospitalization,round(lb.map\_value::numeric,2),pp.drug\_name,ph.cci\_score)

from demography dm

join cardiaccomplications cc

on cc.inpatient\_number=dm.inpatient\_number

join hospitalization\_discharge hd

on hd.inpatient\_number=cc.inpatient\_number

join labs lb

on lb.inpatient\_number=hd.inpatient\_number

join patient\_precriptions pp

on pp.inpatient\_number=lb.inpatient\_number

join patienthistory ph

on ph.inpatient\_number=pp.inpatient\_number;

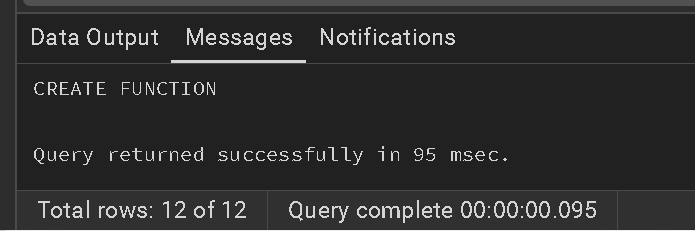
end;

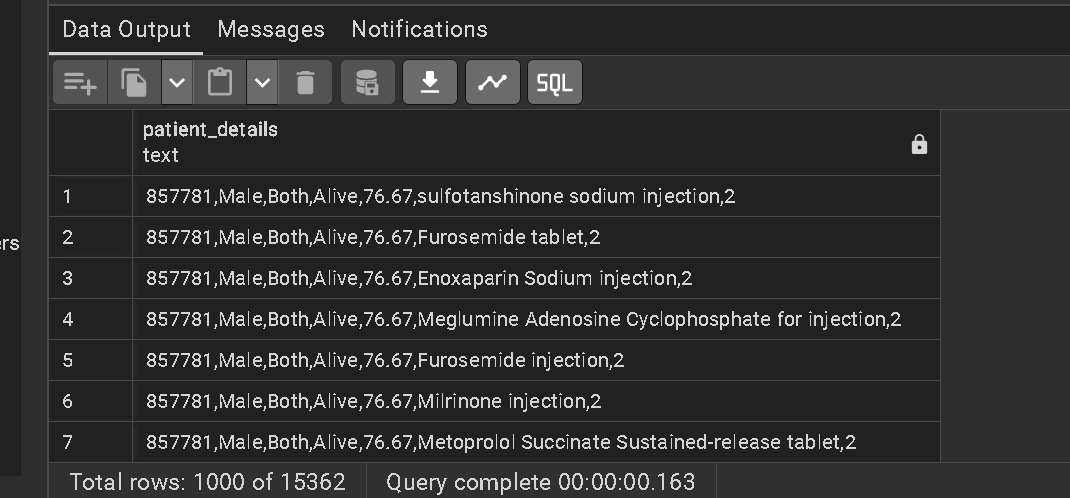
$$language plpgsql

--calling function

select \* from comma\_separated\_patient\_details()

Output:





1. Which patients were on more than 15 prescribed drugs? What was their age and outcome? show the results without using a subquery

select

pp.inpatient\_number,

age,

outcome\_during\_hospitalization,

count(drug\_name)

from patient\_precriptions pp

join demography dm on

dm.inpatient\_number=pp.inpatient\_number

join hospitalization\_discharge hd on

hd.inpatient\_number=dm.inpatient\_number

group by pp.inpatient\_number,age,outcome\_during\_hospitalization

having count(drug\_name)>15

Output:

A screenshot of a computer

Description automatically generated

1. Write a PLSQL block to return the patient ID and gender from demography for a patient if the ID exists and raise an exception if the patient id is not found. Do this without writing or storing a function.

**Patient ID can be hard-coded for the block.**

--Declaration section

do $$

declare

gender\_p text;

P\_ID bigint := 827040;

--execution section

begin

if not exists(select inpatient\_number from demography where inpatient\_number=P\_ID) then

raise notice 'The patient id is not found is %',P\_ID;

else

select

gender into gender\_p

from demography

where inpatient\_number=P\_ID;

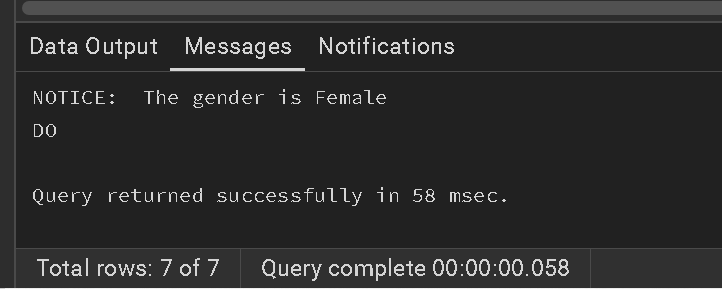
raise notice 'The gender is %',gender\_p;

end if;

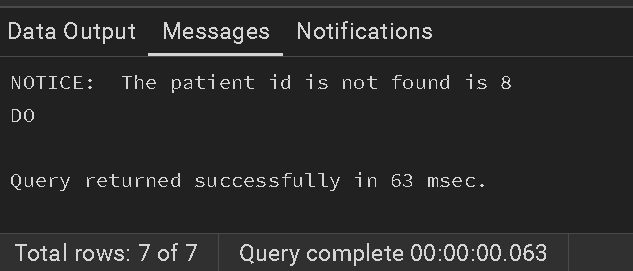
end $$;

Output:

**For the existing patient ID:**



**For the non existing patient id:**

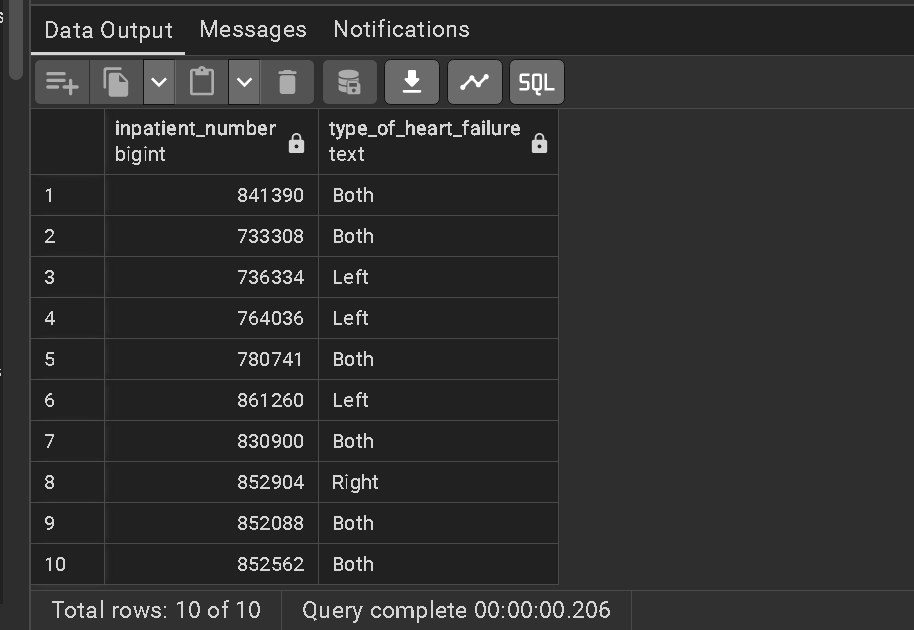


1. Display any 10 random patients along with their type of heart failure select inpatient\_number,type\_of\_heart\_failure from cardiaccomplications

order by random()

limit 10;

Output:



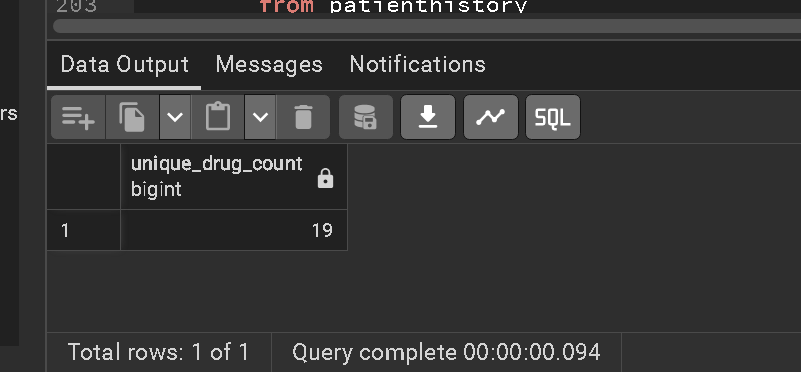
1. How many unique drug names have a length >20 letters?

SELECT COUNT(DISTINCT drug\_name) AS unique\_drug\_count

FROM patient\_precriptions

WHERE LENGTH(drug\_name) > 20;

Output:



1. Rank patients using CCI Score as your base. Use a windows function to rank them in descending order. With the highest no. of comorbidities ranked 1.

select

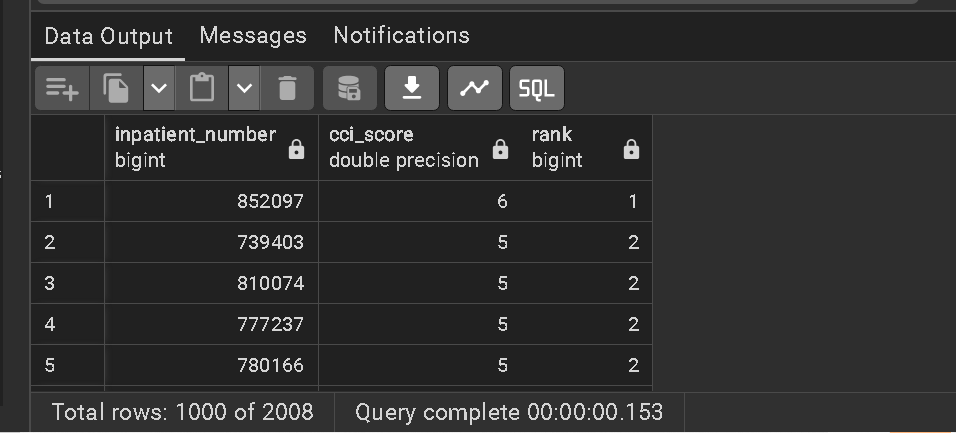
inpatient\_number,

cci\_score,

rank()over(order by coalesce (cci\_score,-1) desc)

from patienthistory

Output:



1. What ratio of patients who are responsive to sound vs pain?

select

round((select count(inpatient\_number)

from responsivenes

where consciousness='ResponsiveToSound'

group by consciousness)\*1.0/(

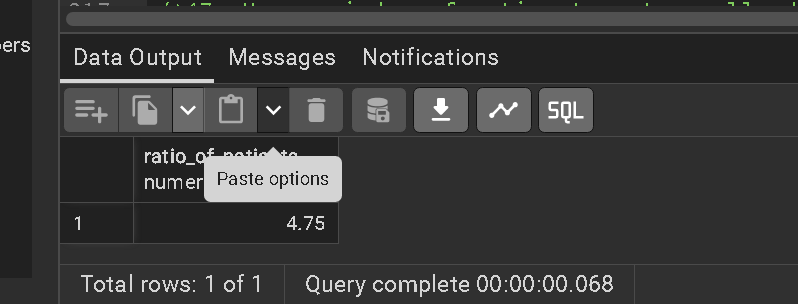
select count(inpatient\_number)

from responsivenes

where consciousness='ResponsiveToPain'

group by consciousness),2) as ratio\_of\_patients

Output:



1. Use a windows function to return all admission ways along with occupation which is related to the highest MAP value.

with admission\_rank as

(select

hd.admission\_way,

d.occupation,

l.map\_value,

rank()over(partition by hd.admission\_way order by l.map\_value desc) as rank\_map from hospitalization\_discharge hd

join demography d on hd.inpatient\_number = d.inpatient\_number

join labs l on hd.inpatient\_number= l.inpatient\_number

order by 1,2,3 )

select distinct admission\_way,occupation, map\_value

from admission\_rank

where rank\_map =1

Output:

A screenshot of a computer

Description automatically generated

1. Display the patients with the highest BMI.

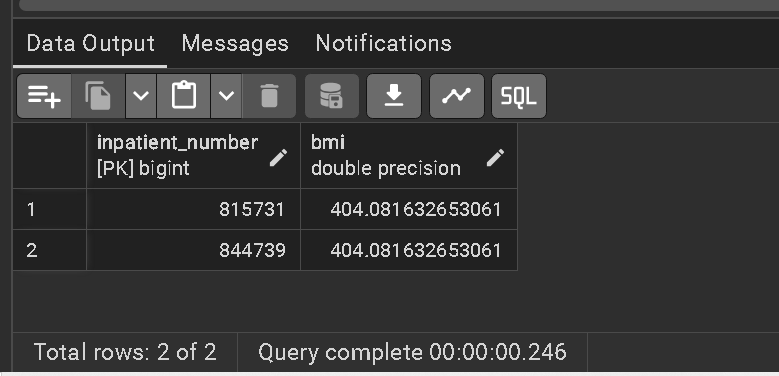
select inpatient\_number,

bmi

from demography

where bmi=(select max(bmi) from demography)

Output:



1. Find the list of Patients who has leukopenia.

Select inpatient\_number, white\_blood\_cell

From labs where white\_blood\_cell <= 3.0e9;

Output:

A screenshot of a graph

Description automatically generated

1. What is the most frequent weekday of admission?

SELECT to\_char(admission\_date, 'Day') AS Day\_of\_week,

Count (\*)

FROM hospitalization\_discharge

group by Day\_of\_week;

Output:

A screenshot of a graph

Description automatically generated

1. Create a console bar chart using the '▰' symbol for count of patients in any age category where theres more than 100 patients"

SELECT agecat,

COUNT(inpatient\_number) as patient\_count,

RPAD('' ,(COUNT(inpatient\_number)/50)::int,'▰'::varchar) AS bar\_chart

FROM demography

GROUP by agecat

HAVING COUNT(inpatient\_number)>100

ORDER BY patient\_count ASC;

Output:

A screenshot of a computer

Description automatically generated

1. Find the variance of the patients' D\_dimer value and display it along with the correlation to CCI score and display them together.

WITH cte\_varaince AS

(

SELECT labs.inpatient\_number,

VARIANCE(labs.d\_dimer) over() AS var\_dimer

FROM labs

)

SELECT var\_dimer,

corr(var\_dimer,ph.cci\_score)

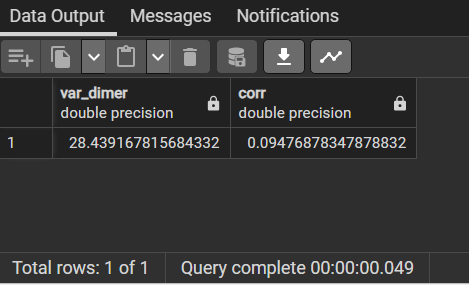
FROM cte\_varaince AS var

JOIN patienthistory as ph

ON var.inpatient\_number = ph.inpatient\_number

GROUP BY var\_dimer;

Output:



1. Which adm ward had the lowest rate of Outcome Death?

select admission\_ward, COUNT(outcome\_during\_hospitalization) AS death\_count

from hospitalization\_discharge

where outcome\_during\_hospitalization = 'Dead'

group by admission\_ward,outcome\_during\_hospitalization

ORDER BY death\_count Asc;

Output:

A screenshot of a computer

Description automatically generated

1. What % of those in a coma also have diabetes. Use the GCS scale to evaluate.

WITH coma\_patients AS (

SELECT r.inpatient\_number

FROM responsivenes r

WHERE r.gcs <= 8

),

coma\_with\_diabetes AS (

SELECT

cp.inpatient\_number

FROM

coma\_patients cp JOIN patienthistory ph

ON cp.inpatient\_number = ph.inpatient\_number

where ph.diabetes = 1)

SELECT

case

when COUNT(coma\_patients.inpatient\_number) = 0 THEN 0

else (COUNT(coma\_with\_diabetes.inpatient\_number) \* 100) / COUNT(coma\_patients.inpatient\_number)

end AS coma\_patients\_with\_diabetes\_percentage

FROM coma\_patients

LEFT JOIN coma\_with\_diabetes

ON coma\_patients.inpatient\_number = coma\_with\_diabetes.inpatient\_number;

Output:

A screenshot of a computer

Description automatically generated

1. Display the drugs prescribed by the youngest patient

select \* from patient\_precriptions

where Inpatient\_number =

(select Inpatient\_number from demography

order by agecat asc

limit 1);

Output:

A screenshot of a computer

Description automatically generated

1. Create a view on the public.responsivenes table using the check constraint

create or replace view public.responsivenes\_view as

select inpatient\_number,

eye\_opening,

verbal\_response,

movement,

consciousness,

gcs

from responsivenes

where

gcs between 3 and 15

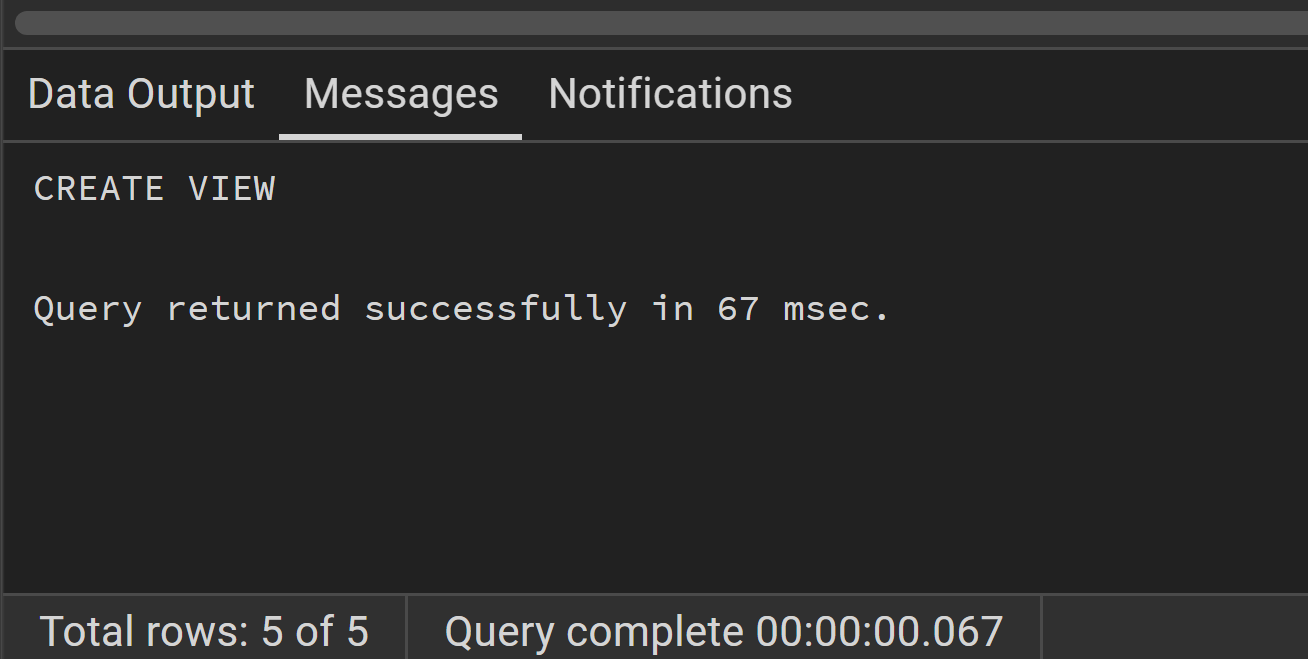
and eye\_opening in (0,1,2,3,4)

and verbal\_response in (0,1,2,3,4,5)

and movement in (0,1,2,3,4,5,6)

with check option;

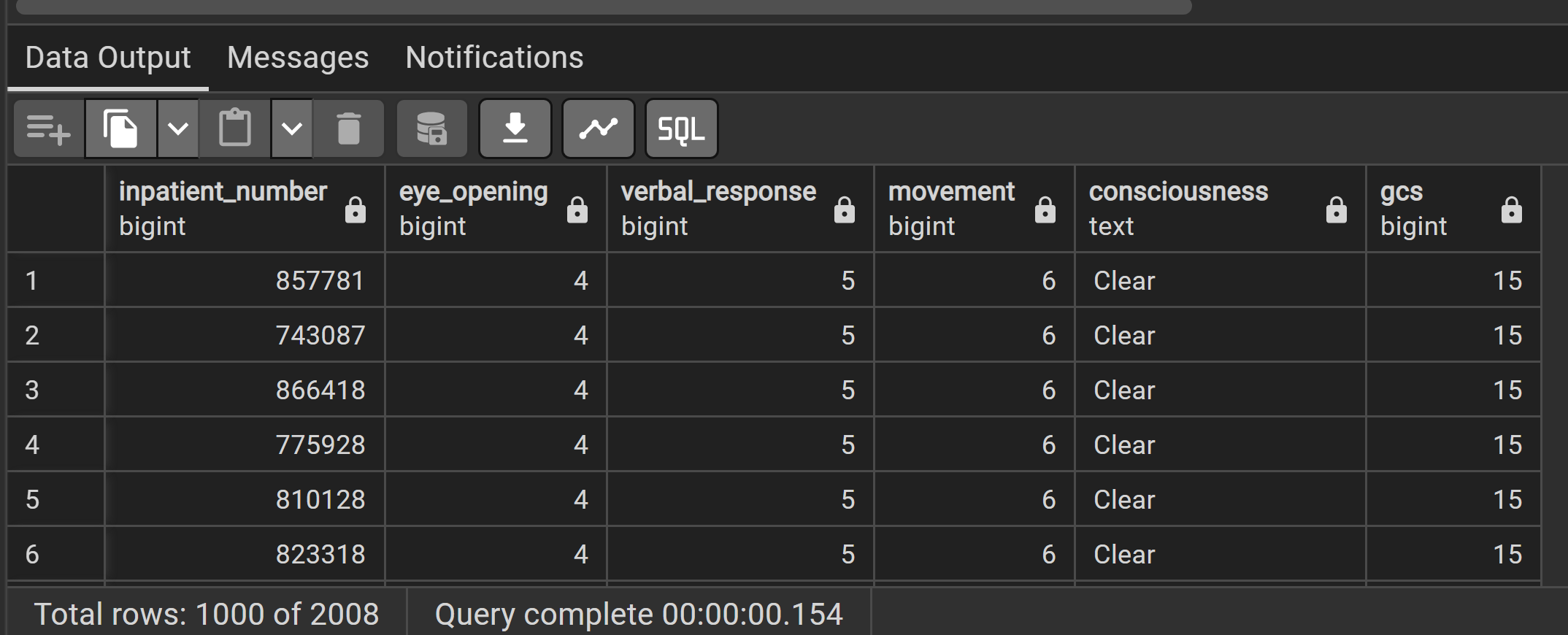
Output:



**selecting all the records in the view**

select \* from responsivenes\_view;

Output:



1. Determine if a word is a palindrome and display true or false. Create a temporary table and store any words of your choice for this question

create Temporary table temporary\_words (

word varchar(100)

);

insert into temporary\_words (word) values

('favor'),

('level'),

('flex'),

('rotor'),

('public'),

('madam'),

('fixed'),

('better'),

('radar'),

('expect');

Select \* from temporary\_words;

select temporary\_words ,

case

when word = reverse(word) then 'true'

else 'false'

end as palindrome

from temporary\_words;

Output:

A screenshot of a computer

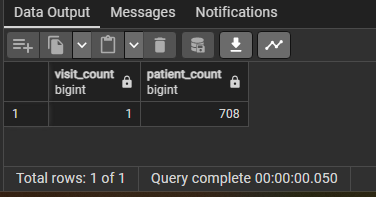
Description automatically generated

1. How many visits were common among those with a readmission in 6 months

select visit\_times, count(inpatient\_number) as patient\_count from hospitalization\_discharge where re\_admission\_within\_6\_months = 1 group by visit\_times order by visit\_times LIMIT 1

Output:

1 visit was common among those with a readmission in 6 months



1. What is the size of the database Cardiac\_Failure

Select pg\_Size\_pretty(pg\_database\_size('Cardiac\_Failure')) as Cardiac\_Failure\_Database\_Size;

Output:

A screenshot of a computer

Description automatically generated

1. Find the greatest common denominator and the lowest common multiple of the numbers 365 and 300. show it in one query

SELECT GCD(365,300) AS Graetest\_common\_denominator,

LCM(365,300) AS Least\_common\_multiple;

Output:

A screenshot of a computer

Description automatically generated

1. Group patients by destination of discharge and show what % of all patients in each group was re-admitted within 28 days.Partition these groups as 2: high rate of readmission, low rate of re-admission. Use windows functions.

with readmission\_count as

(select destinationdischarge, count(inpatient\_number) as count\_patients,

sum(case when re\_admission\_within\_28\_days = 1 then 1 else 0 end) as readmitted\_patients

from hospitalization\_discharge

group by destinationdischarge),

readmission\_percentage AS

(select destinationdischarge,

count\_patients,

readmitted\_patients,

(readmitted\_patients\*100.0/count\_patients) as readmission\_percentage

from readmission\_count

)

select destinationdischarge,

readmission\_percentage,

case when readmission\_percentage > (select percentile\_cont(0.5) within group(order by readmission\_percentage) from readmission\_percentage)

then 'High'

ELSE 'Low'

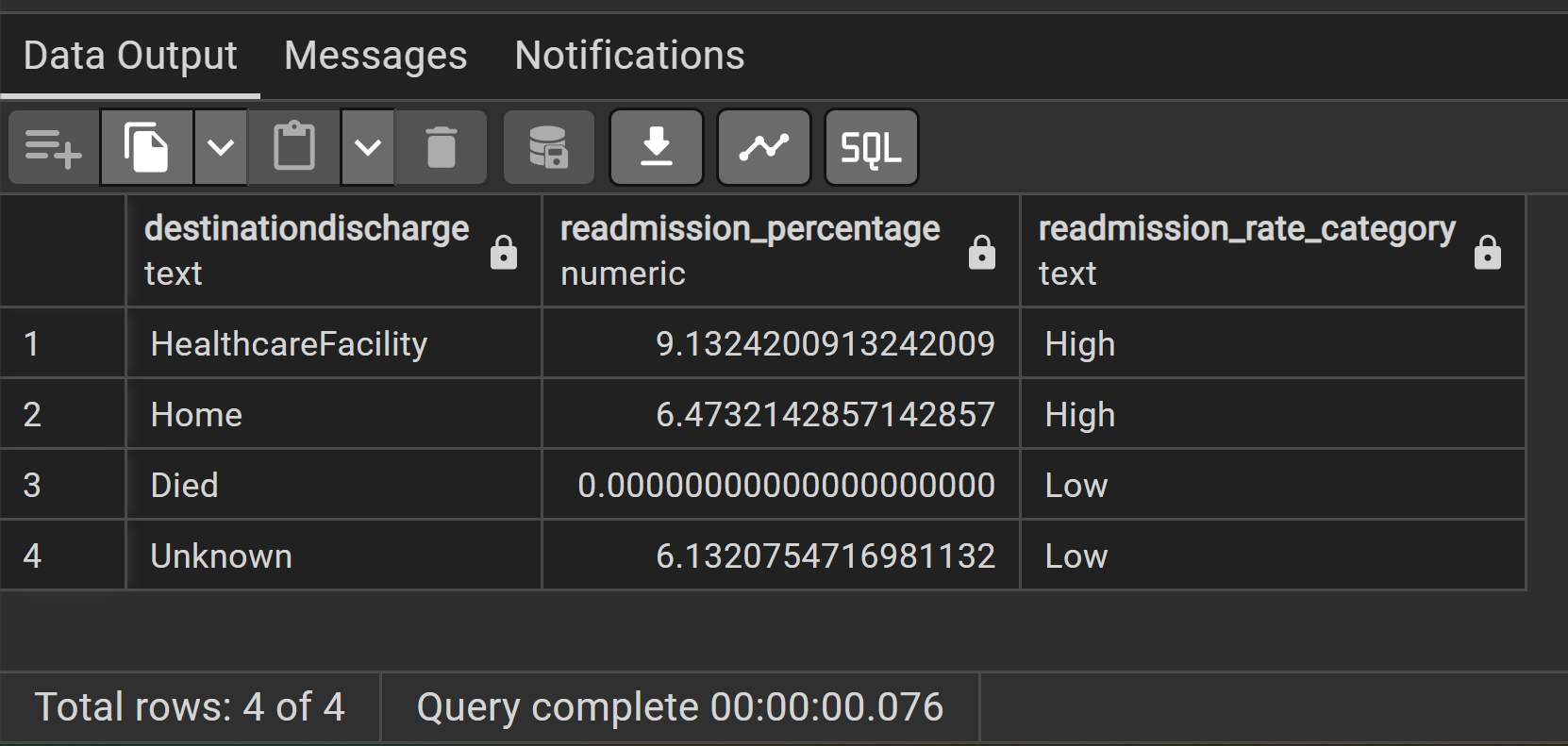
END AS readmission\_rate\_category

from readmission\_percentage

group by destinationdischarge,readmission\_percentage,readmission\_rate\_category

order by readmission\_rate\_category

Output:



1. What is the size of the table labs in KB without the indexes or additional objects

Select pg\_Size\_pretty(pg\_relation\_size('labs'));

Output:

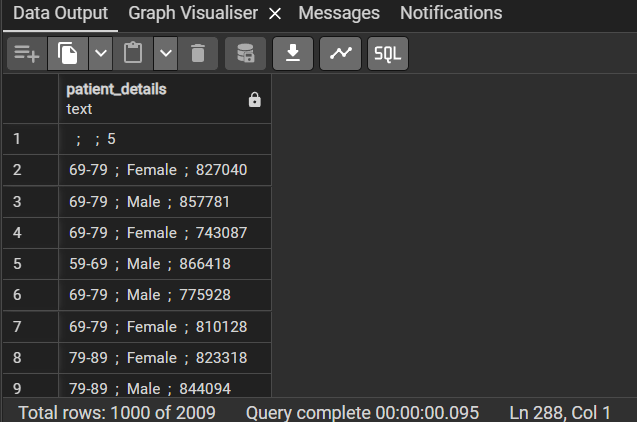
A screenshot of a computer

Description automatically generated

1. concatenate age, gender and patient ID with a ';' in between without using the || operator

Select Concat(agecat,' ; ',gender, ' ; ', inpatient\_number) as patient\_details from demography;

Output:



1. Display a reverse of any 5 drug names

Select drug\_name,

Reverse(drug\_name)

from patient\_precriptions

order by drug\_name asc

limit 5;

Output:

A screenshot of a graph visualizer

Description automatically generated

1. What is the variance from mean for all patients GCS score

SELECT inpatient\_number,

gcs,

round(avg(gcs) over (),2) AS mean\_gcs,

variance(gcs) over () AS variance\_gcs,

round(( gcs - avg(gcs) over () )/ ( variance(gcs) over ()),5 )as gcs\_variance\_from\_mean

from responsivenes

GROUP BY inpatient\_number,gcs;

Output:

A screenshot of a computer

Description automatically generated

1. Using a while loop and a raise notice command, print the 7 times table as the result

do $$

declare

counter integer := 0;

begin

while counter < 20 loop

raise notice '7 \* % = %',counter,counter \* 7;

counter := counter + 1;

end loop;

end;

$$;

Output:

A screenshot of a computer

Description automatically generated

1. Show month number and month name next to each other(admission\_date), ensure that month number is always 2 digits. eg, 5 should be 05".

SELECT inpatient\_number,

admission\_date,

(CONCAT(TO\_CHAR(DATE\_PART('month',admission\_date), 'fm00'),

',' ,

TO\_CHAR(admission\_date,'Mon') )) AS Month

FROM hospitalization\_discharge;

Output:

A screenshot of a computer

Description automatically generated

1. How many patients with both heart failures had kidney disease or cancer.

SELECT SUM(ph.moderate\_to\_severe\_chronic\_kidney\_disease) AS Kidney\_disease,

SUM(ph.leukemia + ph.malignant\_lymphoma) AS cancer,

count(ph.inpatient\_number) AS Total\_patients\_with\_kidneyorcancer

FROM cardiaccomplications AS comp

JOIN patienthistory AS ph

ON comp.inpatient\_number = ph.inpatient\_number

WHERE type\_of\_heart\_failure = 'Both'

AND( ph.moderate\_to\_severe\_chronic\_kidney\_disease =1

OR ph.leukemia = 1 OR ph.malignant\_lymphoma = 1);

**Did not consider solid\_tumor as cancer because, not all solid\_tumor is cancer.**

A screenshot of a computer

Description automatically generated

1. Return the number of bits and the number of characters for every value in the column: Occupation

SELECT DISTINCT occupation,

bit\_length(occupation) as number\_of\_bits,

length(occupation) as number\_of\_characters

FROM demography

WHERE occupation is not null

GROUP BY occupation;

Output:

A screenshot of a computer

Description automatically generated

1. Create a stored procedure that adds a column to table cardiaccomplications. The column should just be the todays date.

-- Create a Procedure

CREATE OR REPLACE PROCEDURE pr\_date\_today ()

language plpgsql

as

$$

begin

-- create and update the date\_today column to cardiaccomplications table

ALTER TABLE cardiaccomplications

ADD COLUMN date\_today date default current\_date;

commit;

end;

$$;

Output:

A screenshot of a computer message

Description automatically generated

-- Call the procedure

call pr\_date\_today();

Output:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

SELECT inpatient\_number,date\_today FROM cardiaccomplications;

Output:

A screenshot of a computer

Description automatically generated

1. What is the 2nd highest BMI of the patients with 5 highest myoglobin values. Use windows functions in solution.

WITH temp\_myoglobin AS

(

SELECT labs.inpatient\_number,

labs.myoglobin,

demo.bmi,

rank() over (ORDER BY labs.myoglobin DESC) AS myoglobin\_rank

FROM labs

JOIN demography as demo

ON labs.inpatient\_number = demo.inpatient\_number

WHERE myoglobin is not null

),temp\_bmi AS (

SELECT inpatient\_number,

myoglobin,

round(bmi::numeric, 2),

rank () over (order by bmi DESC) AS bmi\_rank,

myoglobin\_rank from temp\_myoglobin

WHERE myoglobin\_rank <=5

ORDER BY bmi DESC

)

SELECT \* FROM temp\_bmi WHERE bmi\_rank = 2;

Output:

A screenshot of a computer

Description automatically generated

1. What is the standard deviation from mean for all patients pulse.

SELECT inpatient\_number,

pulse,

round(avg(pulse) over (),2) AS mean\_pulse,

round(stddev(pulse) over (),2) AS stddev\_pulse,

round(( pulse - avg(pulse) over () )/ ( stddev(pulse) over ()),5 )as pulse\_stdev\_from\_mean

from labs

GROUP BY inpatient\_number,pulse;

Output:

A screenshot of a computer

Description automatically generated

1. Create a procedure to drop the age column from demography.

CREATE OR REPLACE PROCEDURE pr\_age\_del ()

language plpgsql

as

$$

begin

-- Query to delete the age column from demography

ALTER TABLE demography DROP COLUMN age;

commit;

end;

$$;

Output:

A screenshot of a message

Description automatically generated

SELECT \* FROM demography;

Output:

A screenshot of a computer

Description automatically generated



-- call the procedure pr\_age\_del

call pr\_age\_del();

Output:

A screenshot of a computer

Description automatically generated

SELECT \* FROM demography;

Output:

A screenshot of a computer

Description automatically generated

1. What was the average CCI score for those with a BMI>30 vs for those <30.

SELECT --COUNT(demo.bmi),

CASE

WHEN demo.bmi>30 THEN 'BMI>30'

WHEN demo.bmi<30 THEN 'BMI<30'

END AS bmi\_range,

AVG(ph.cci\_score) AS avg\_cci

FROM demography as demo

JOIN patienthistory AS ph

ON demo.inpatient\_number = ph.inpatient\_number

GROUP BY bmi\_range;

Output:

A screenshot of a computer

Description automatically generated

1. Write a trigger after insert on the Patient Demography table. if the BMI >40, warn for high risk of heart risks.

CREATE FUNCTION bmi\_trigg\_fn() -- Create function

RETURNS TRIGGER AS $$

BEGIN

if NEW.bmi > 40 THEN

RAISE NOTICE 'High risk of heart risks for patient id % with BMI (% )',NEW.inpatient\_number,NEW.bmi;

end if;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

Output:

A screenshot of a computer message

Description automatically generated

CREATE TRIGGER bmi\_trigg --- Create trigger

AFTER INSERT ON "demography"

FOR EACH ROW

EXECUTE FUNCTION bmi\_trigg\_fn();

Output:

A screenshot of a computer

Description automatically generated

A black and white text

Description automatically generated

--Insert records to demography

INSERT INTO demography (inpatient\_number,gender,weight,height,bmi,occupation,agecat)

VALUES(987665,'Female',120,1.5,41,'UrbanResident','79-89');

Output:

A screenshot of a computer

Description automatically generated

1. Most obese patients belong to which age group and gender. You may make an assumption for what qualifies as obese based on your research.

SELECT agecat,

gender,

COUNT(inpatient\_number) AS number\_obese

FROM demography

WHERE bmi>=30 -- considering 30 as obese

AND gender is not null

GROUP BY agecat,gender

ORDER BY number\_obese DESC LIMIT 1;

* **This is the output from the original dataset, did not include inserted values from Q75**

Output:

A screenshot of a computer

Description automatically generated

1. Show all response details of a patient in a JSON array.

SELECT array\_to\_json(array\_agg(row\_to\_json(responsivenes))) AS json\_resp

FROM responsivenes;

Output:

A screenshot of a computer

Description automatically generated

1. Update the table public.patienthistory. Set type\_ii\_respiratory\_failure to be upper case, query the results of the updated table without writing a second query.

UPDATE

patienthistory

SET

type\_ii\_respiratory\_failure =UPPER(type\_ii\_respiratory\_failure)

RETURNING type\_ii\_respiratory\_failure;

Output:

A screenshot of a computer

Description automatically generated

1. Find all patients using Digoxin or Furosemide using regex.

SELECT DISTINCT inpatient\_number

FROM patient\_precriptions

WHERE drug\_name ~\* '(Digoxin|Furosemide)';

Output:

A screenshot of a computer

Description automatically generated

1. Using a recursive query, show any 10 patients linked to the drug: "Furosemide injection".

WITH RECURSIVE cte\_rec\_query

AS (

SELECT

inpatient\_number,

drug\_name

FROM

patient\_precriptions

WHERE

drug\_name = 'Furosemide injection'

UNION

SELECT

p.inpatient\_number,

p.drug\_name

FROM patient\_precriptions p

INNER JOIN cte\_rec\_query r1

ON r1.inpatient\_number = p.inpatient\_number

)

SELECT \* FROM cte\_rec\_query LIMIT 10 ;

Output:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated