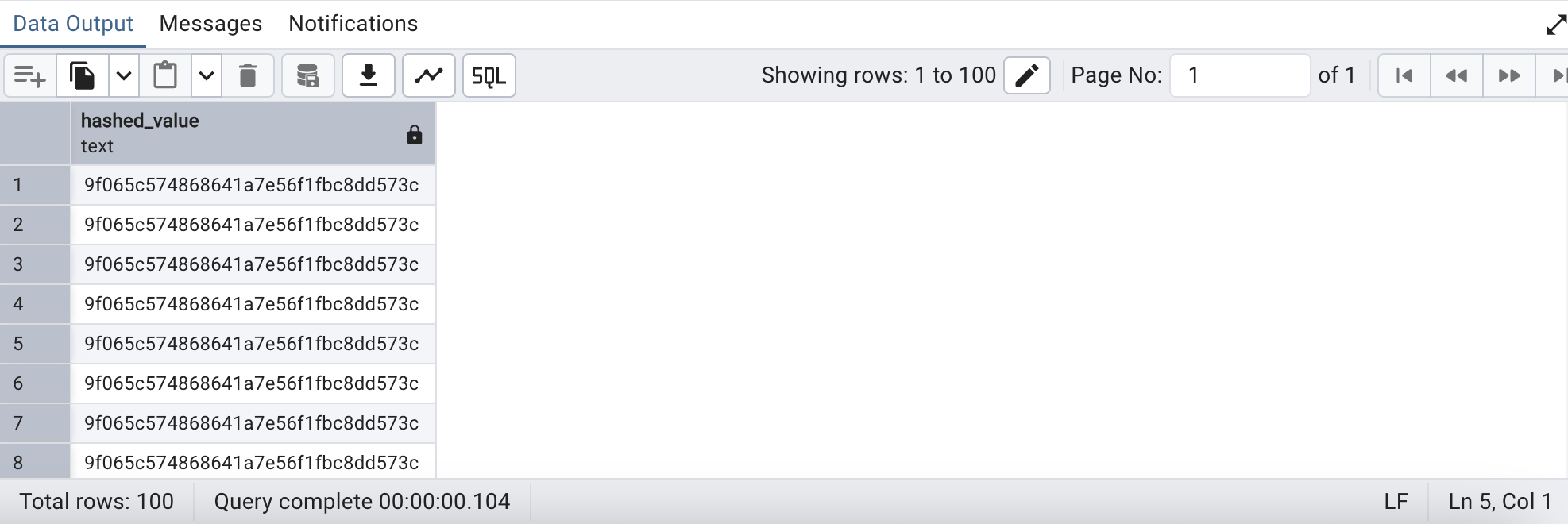
**CriticareDataAnalysis\_Extra**

**1)How can you use a hash function to generate a hash value for a patient id?**

**Query:**

select md5('patient\_id') as hashed\_value

from baseline;

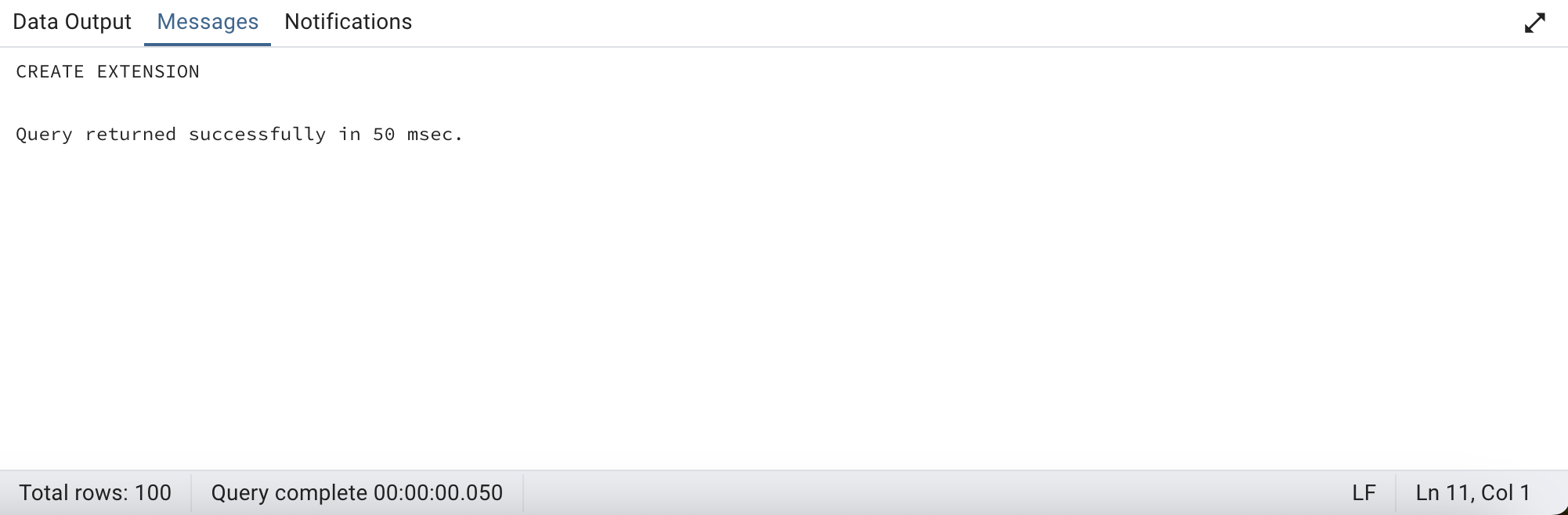
****

------------------------------------------------------------------------------------------------------------------------

**2)Encrypt, encode, decode, and decrypt patient\_id from baseline table for patient details security**

**Query:**

create extension if not exists pgcrypto;



with encrypted\_data as (

--encrypt patient\_id & encode it as base64 for easy storage

select encode(pgp\_sym\_encrypt(cast(patient\_id as text), 'my\_secret\_key'), 'base64') as encrypted\_value

from baseline

limit 100

)

**--decode and decrypt the encrypted data**

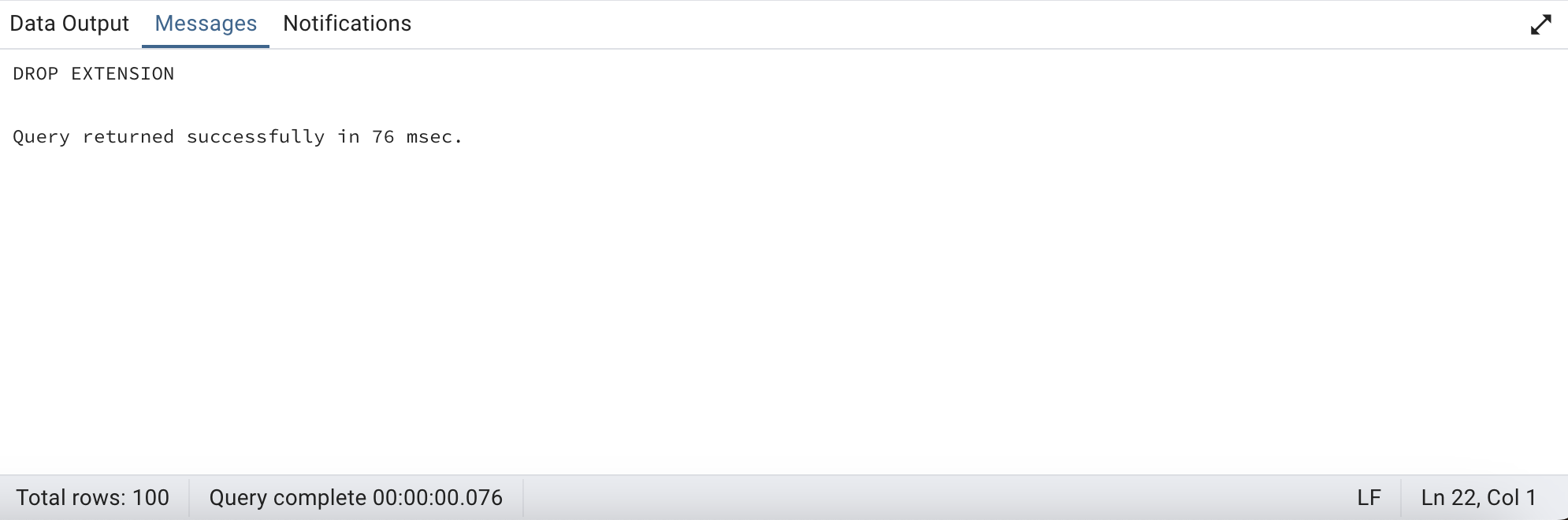
select pgp\_sym\_decrypt(decode(encrypted\_value, 'base64'), 'my\_secret\_key') as decrypted\_value,encrypted\_value

from encrypted\_data;

****

**–drop extension:**

drop extension if exists pgcrypto cascade;



-----------------------------------------------------------------------------------------------------------------

**3) Categorize and count the number of patient transfers based on whether their transfer start time falls in the first half (midnight to noon) or the second half (noon to midnight) of the day using a single bit function?**

**Query:**

with transfers\_category as (

select

case

when extract(hour from starttime) < 12 then 0

else 1

end as start\_time\_bit,

count(\*) as patient\_cnt

from

transfer

where

date(starttime) = date(stoptime)

group by

case

when extract(hour from starttime) < 12 then 0

else 1

end

)

select

start\_time\_bit,

patient\_cnt,

case

when start\_time\_bit = 0 then 'AM'

else 'PM'

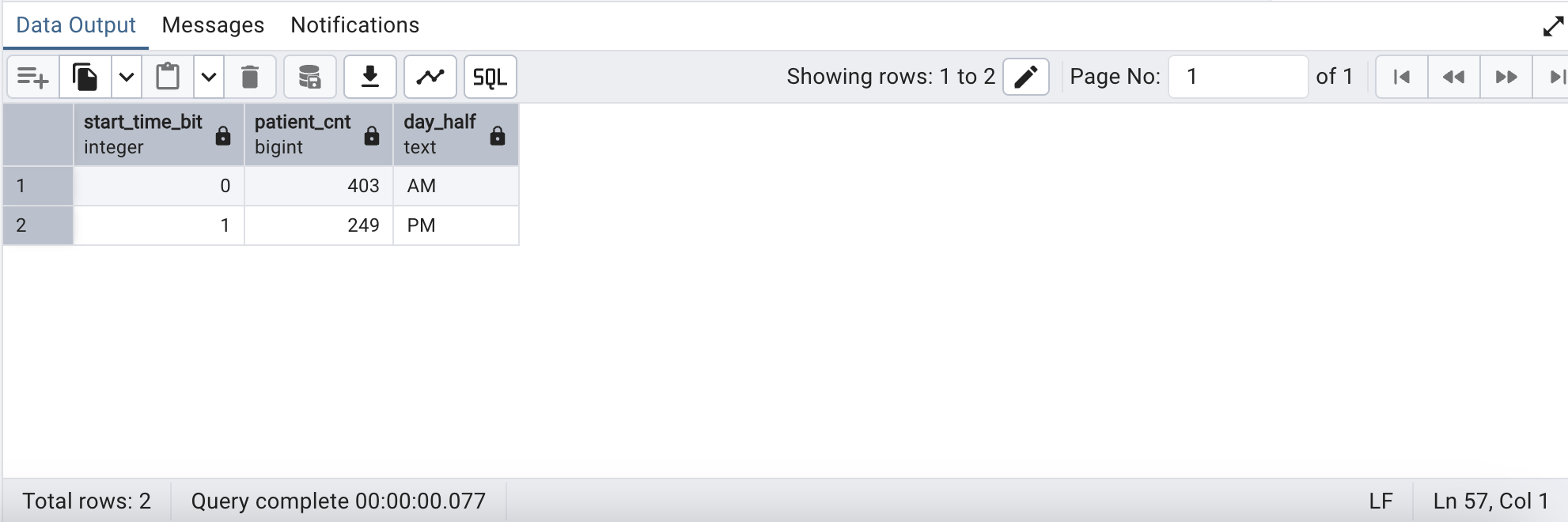
end as day\_half

from

transfers\_category

order by

start\_time\_bit;

****

----------------------------------------------------------------------------------------------------------------------

**4)Use XML functions to retrieve the gender of patients along with their admitdept and count & filter the patients based on gender in an XML format**

**Query:**

select

xmlforest(

sex as patient\_gender,

admitdept as department\_name,

case

when sex = 'Male' then count(\*) filter (where sex = 'Male')

else count(\*) filter (where sex = 'Female')

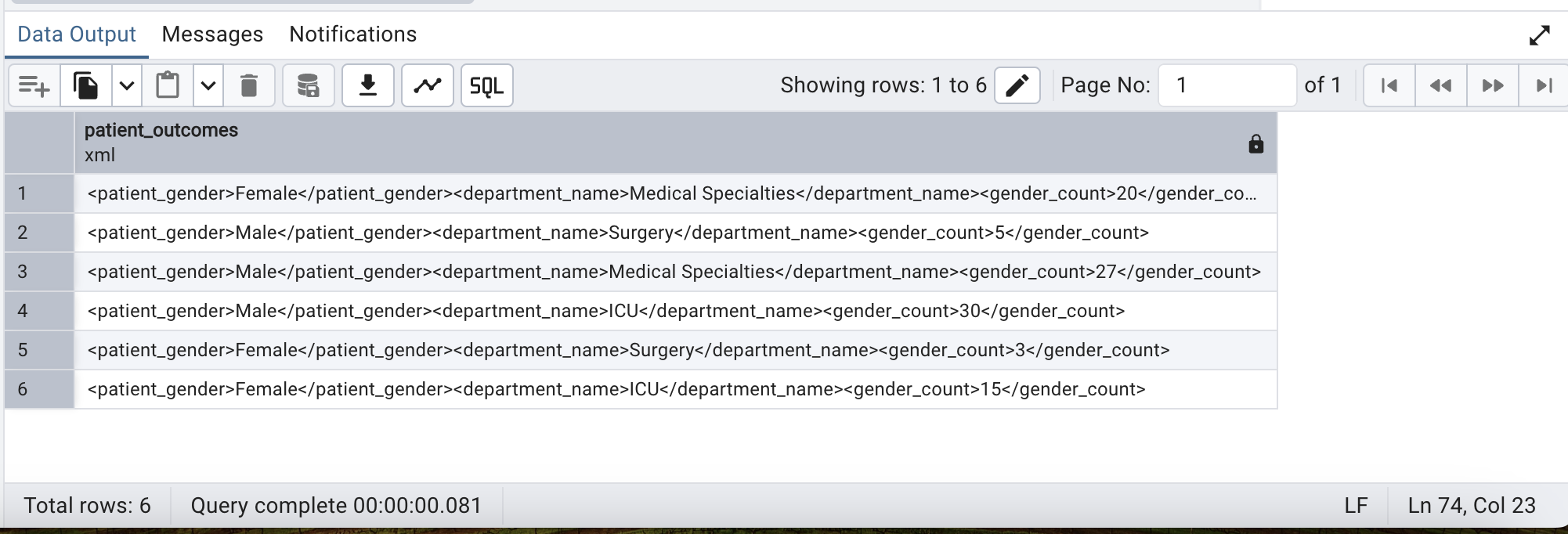
end as gender\_count

) as patient\_outcomes

from

baseline

group by sex,admitdept

****

----------------------------------------------------------------------------------------------------------------

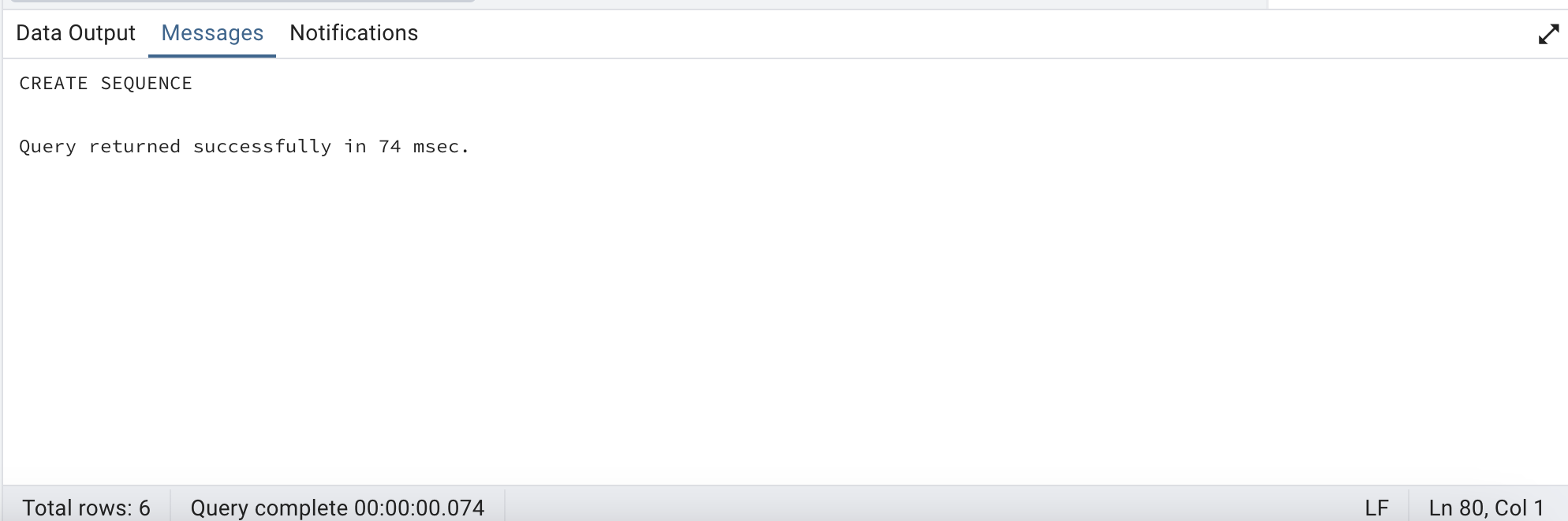
**5)create a sequence to generate unique patient IDs, and then create a table that uses this sequence to store patient records? Also set the sequence to a specific value, insert patient data, and retrieve values from the sequence?**

**Query:**

create sequence patient\_id\_Seq

start with 1000

increment 1;



**-- Set the sequence to a specific value**

select setval('patient\_id\_Seq', 2000 , true) as set\_sequence\_value;

create table patient\_records (

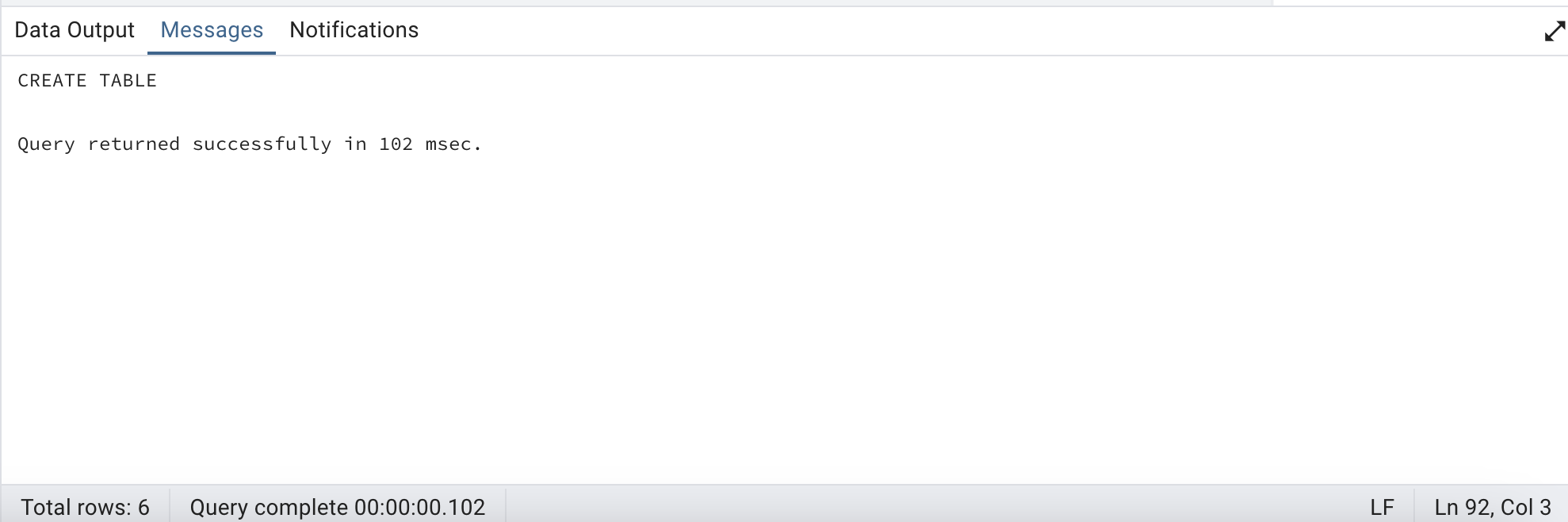
patient\_id int default nextval('patient\_id\_Seq') primary key,

patient\_name varchar(100),

admitdept text,

admitdate date

);

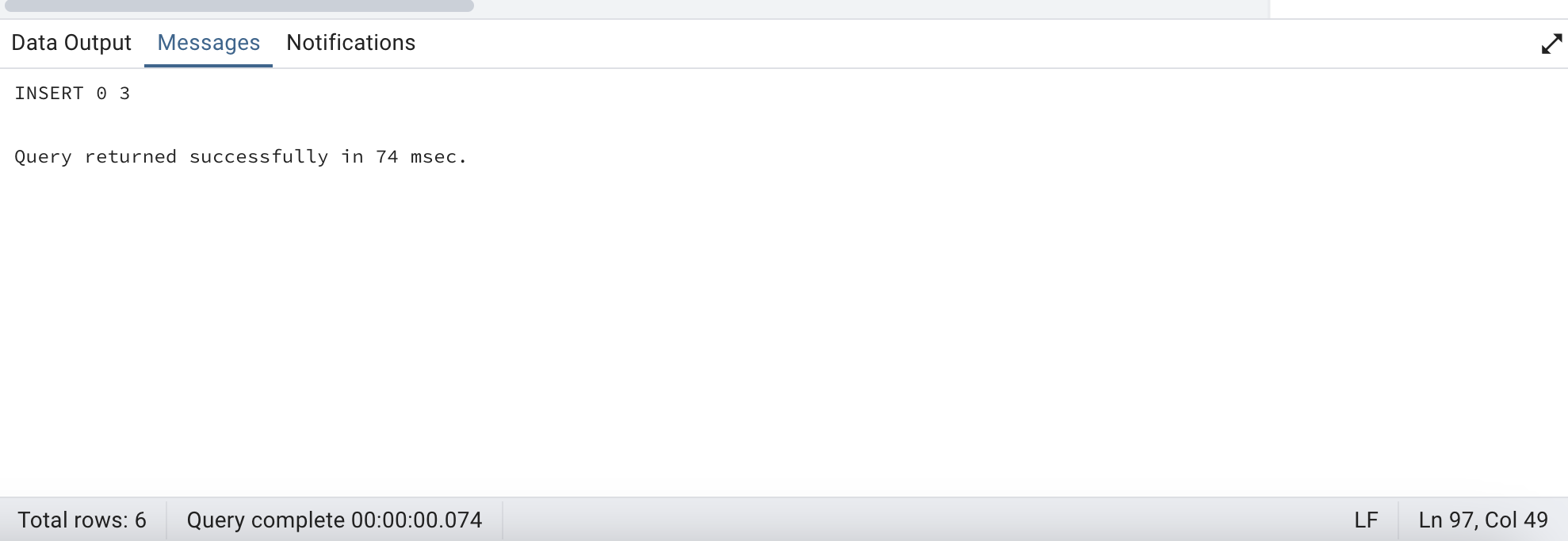


insert into patient\_records (patient\_name,admitdept,admitdate)

values ('John Doe', 'Laproscopic Surgery', '2019-01-22'),

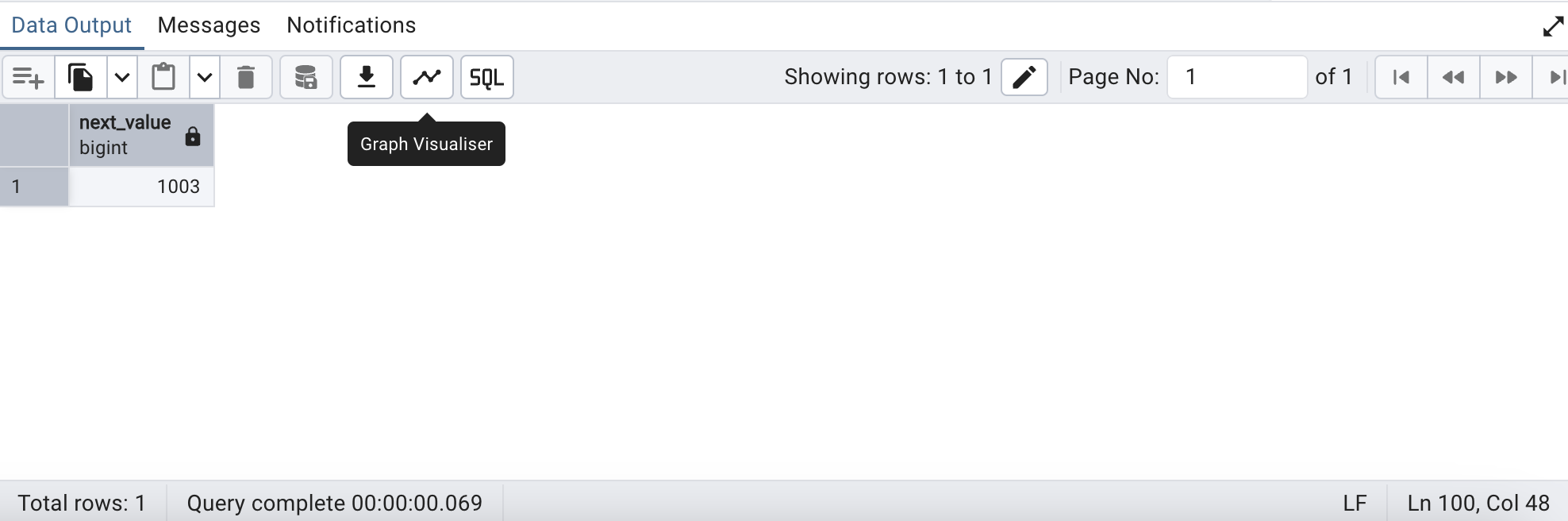
('Jane Smith', 'Heart Surgery', '2015-08-12'),

('Alice Bob', 'Vascular Surgery', '2017-02-06')



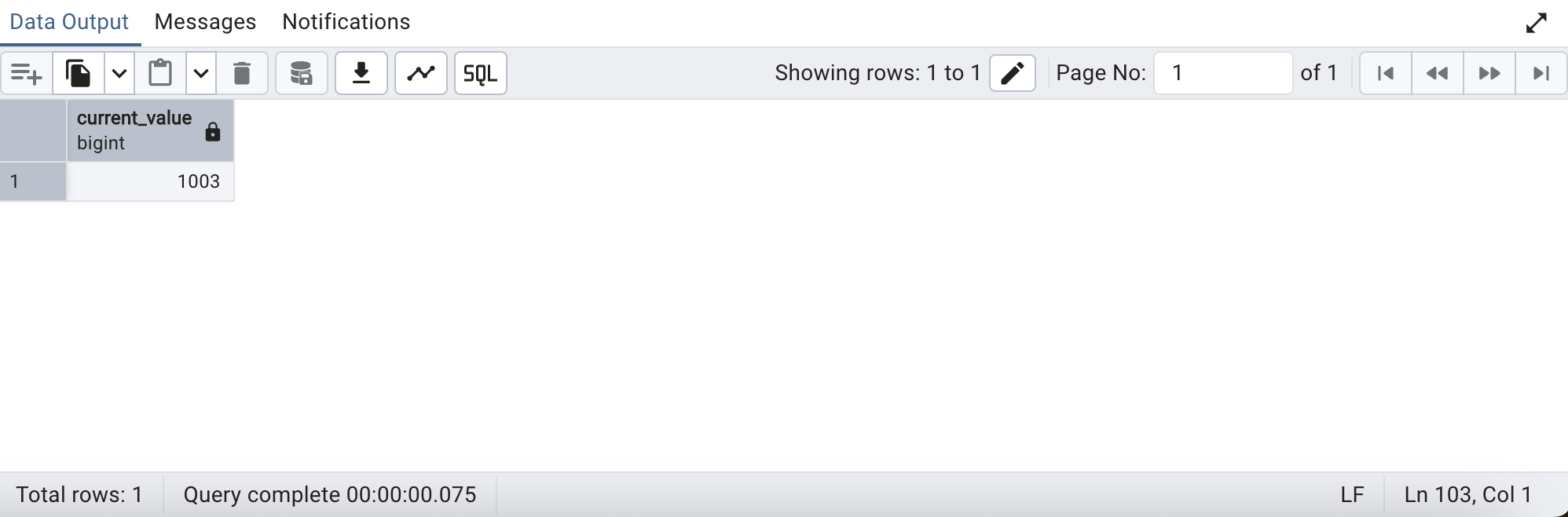
**-- Get the next value in the sequence**

select nextval('patient\_id\_Seq') as next\_value;



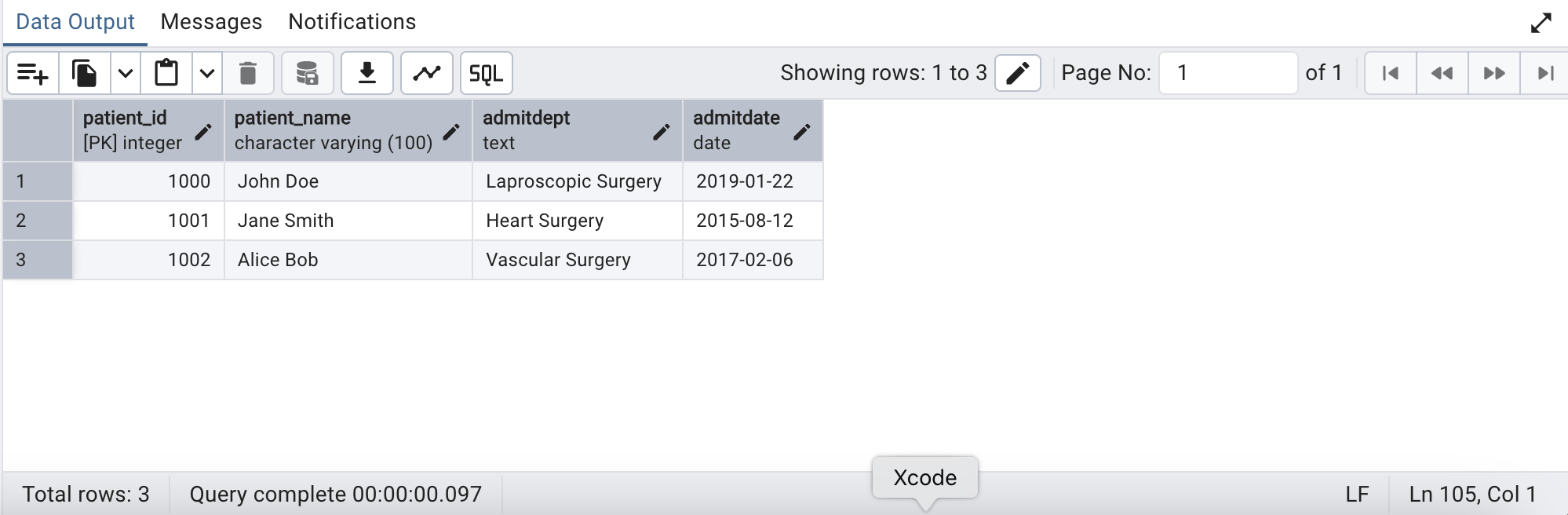
**-- Retrieve the current value of the sequence**

select currval('patient\_id\_Seq') as current\_value;



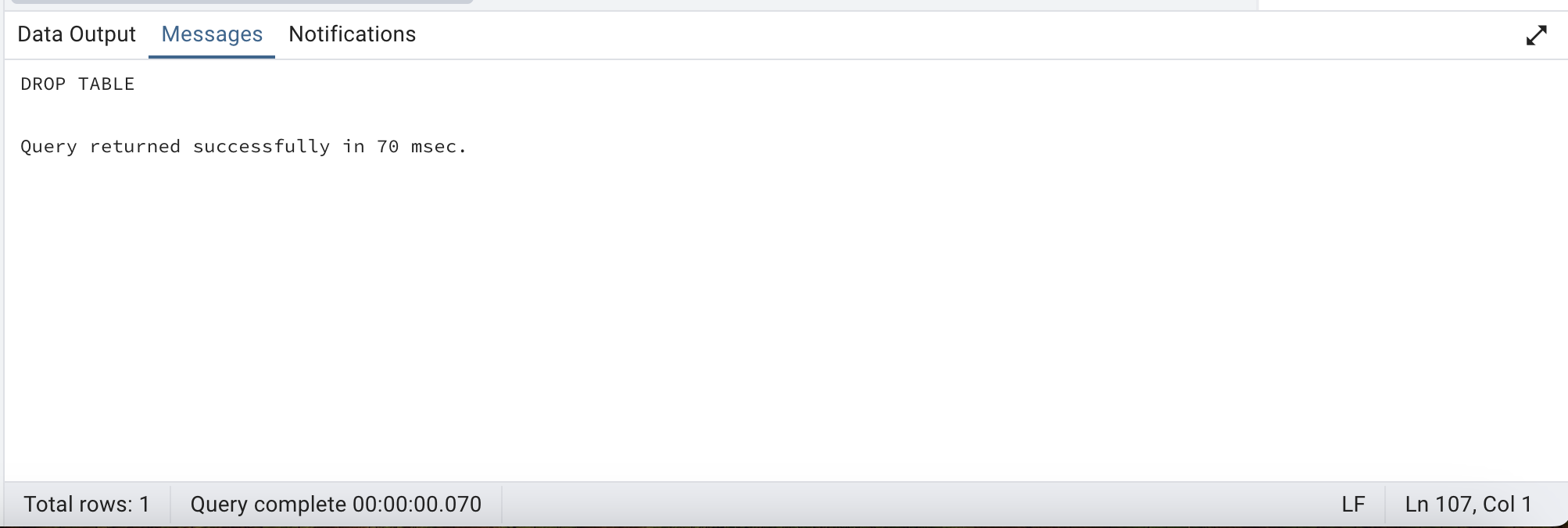
**-- Select all records:**

select \* from patient\_records;



**-- drop records:**

drop table patient\_records;



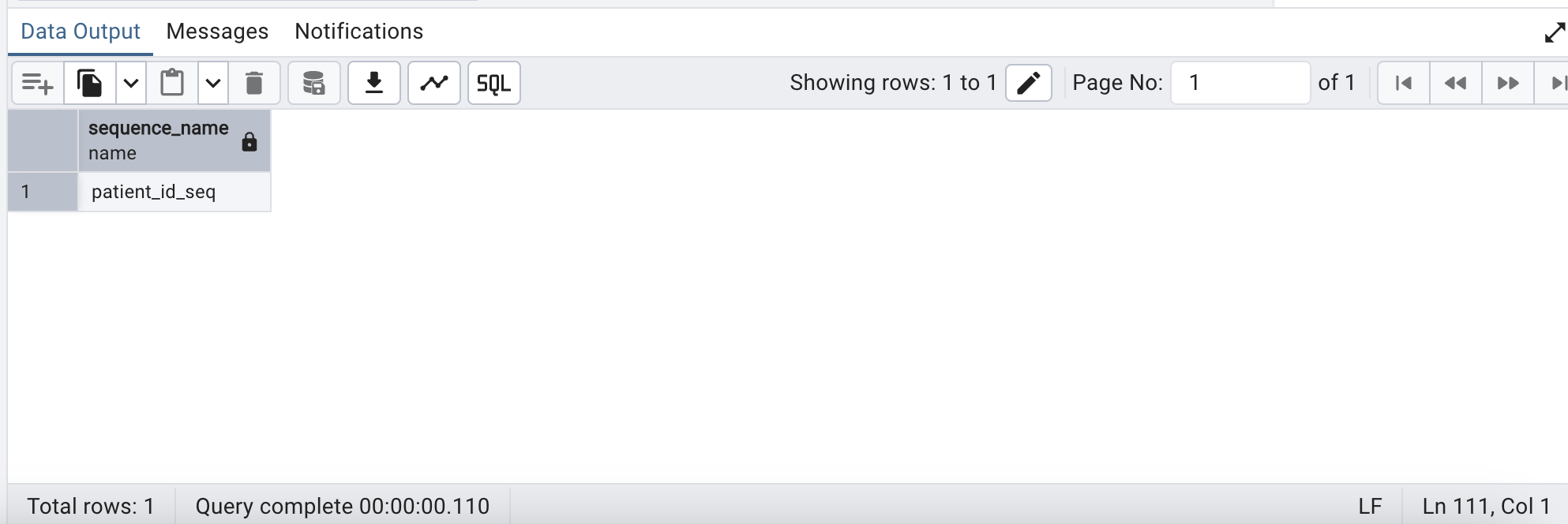
**--optional check**

**-- To Check all sequences in the current schema**

select sequence\_name

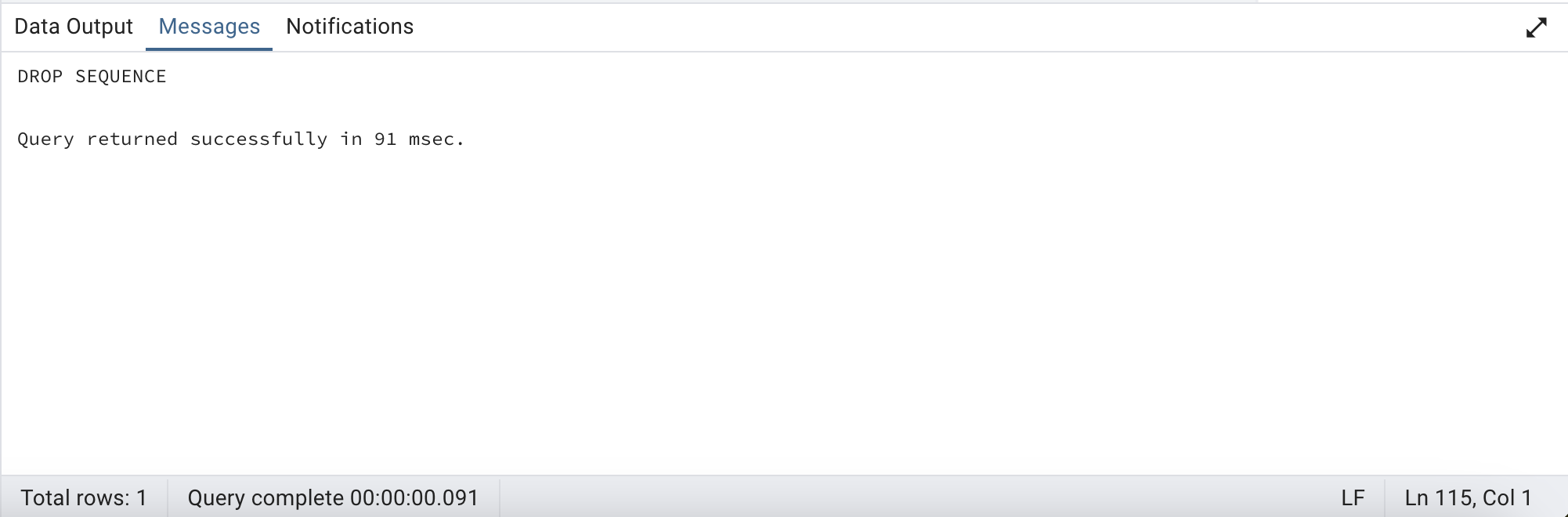
from information\_schema.sequences

where sequence\_schema = 'public';



**-- drop sequence:**

drop sequence patient\_id\_Seq;



---------------------------------------------------------------------------------------------------------

**6)Create an index on the baseline table and use explain analyze to show differences if any.**

**Query:**

explain analyze

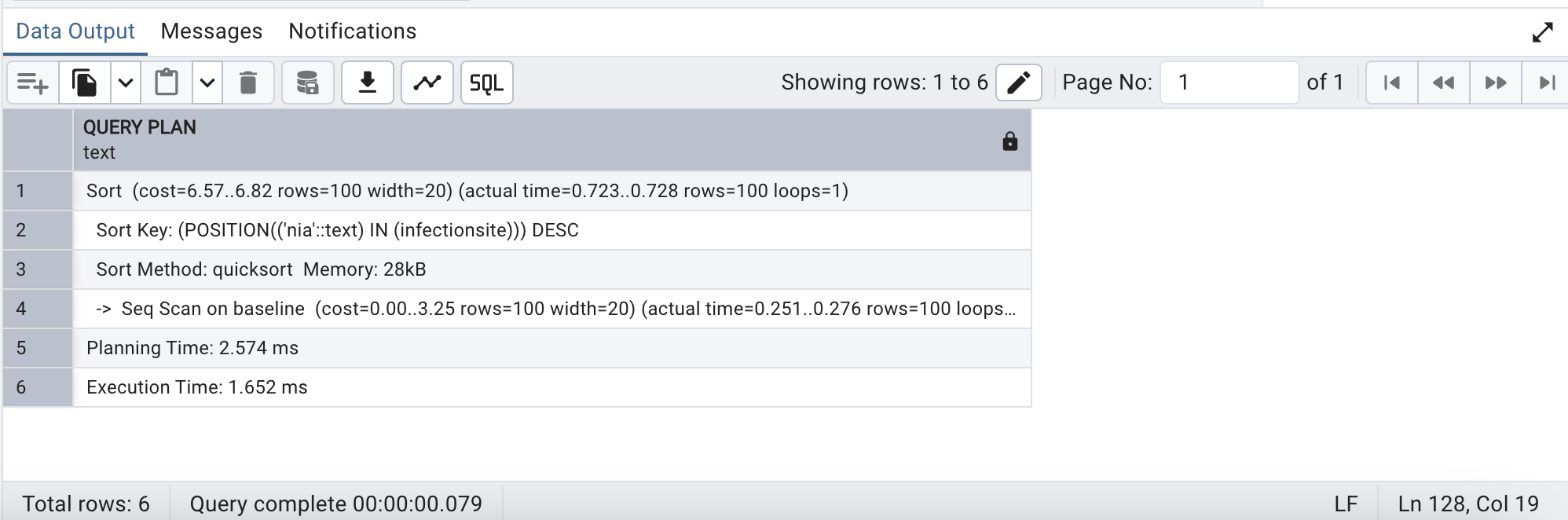
select "patient\_id",

"infectionsite",

position('nia' in "infectionsite")

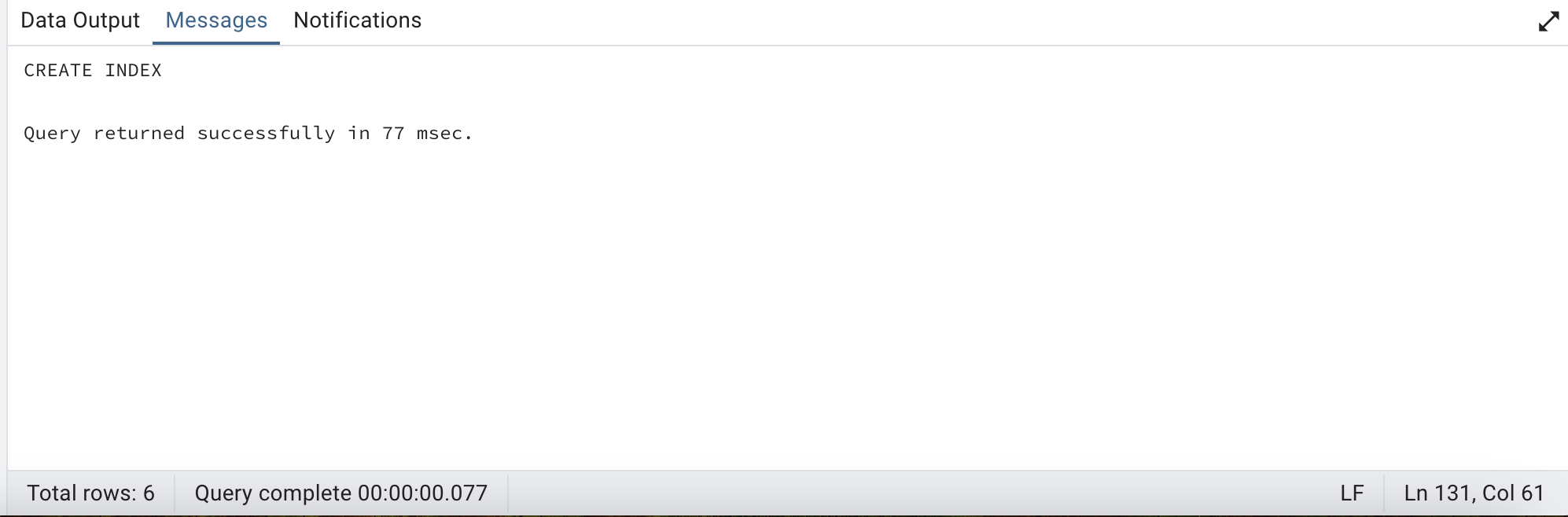
from baseline

order by 3 desc;



**--Create Index**

create index patient on baseline using btree("patient\_id")



**--After creating Index again Run Explain Analyze**

explain analyze

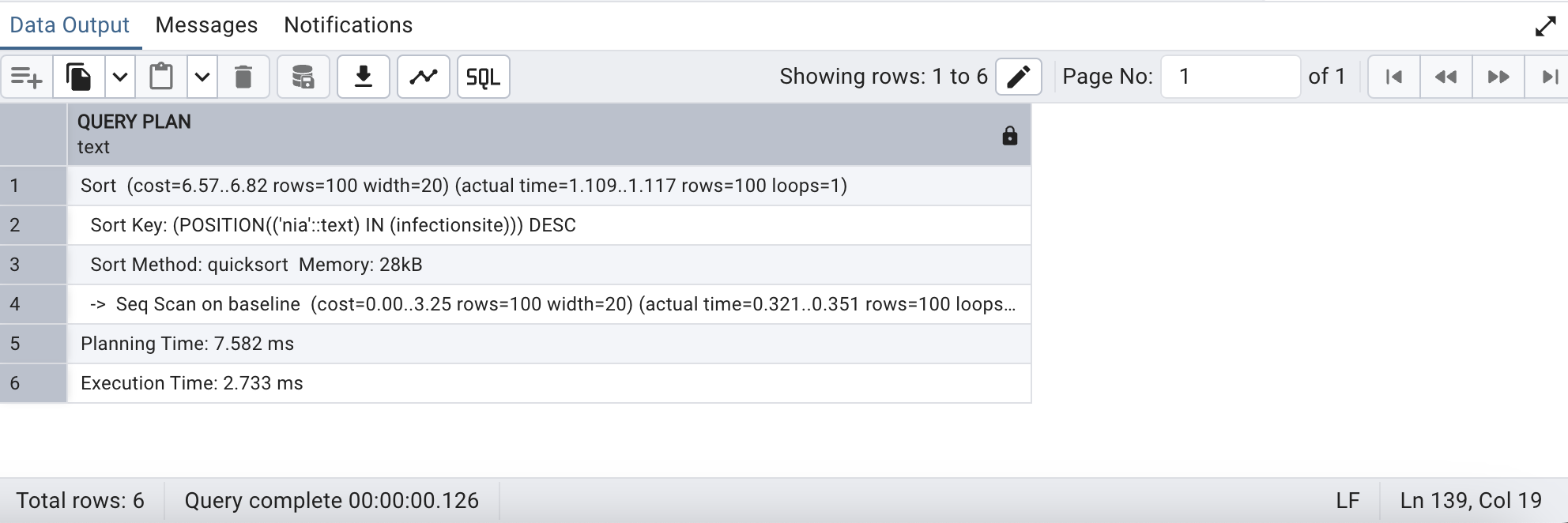
select "patient\_id",

"infectionsite",

position('nia' in "infectionsite")

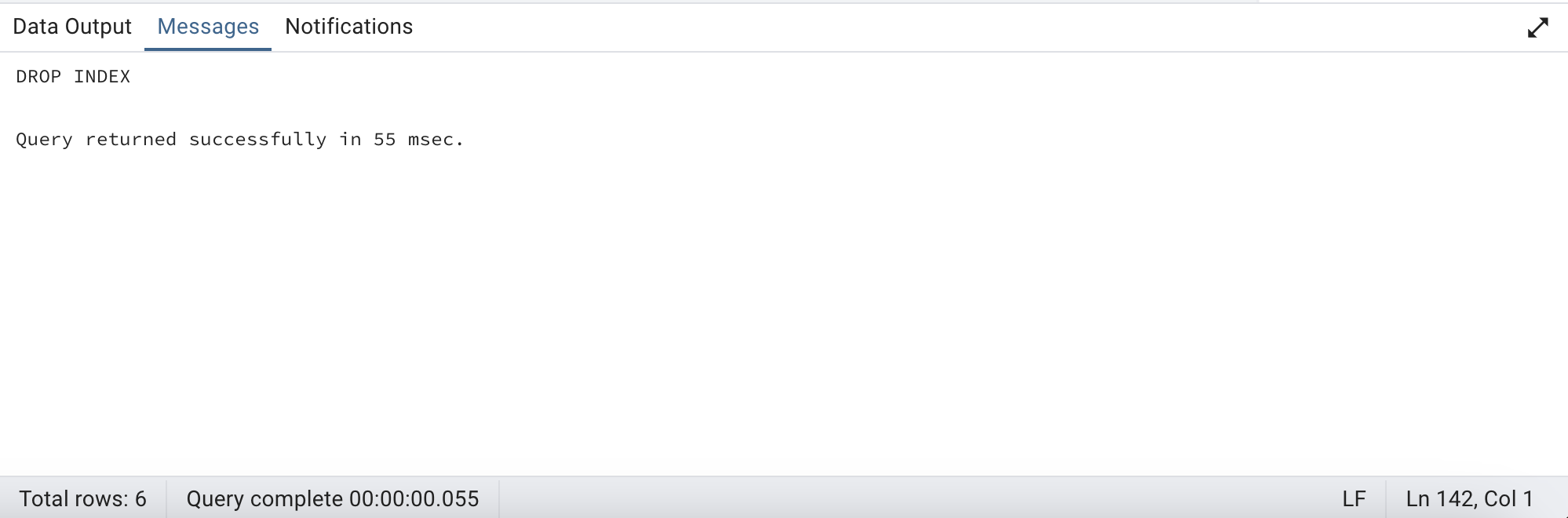
from baseline

order by 3 desc;



**--drop index patient**

drop index patient



------------------------------------------------------------------------------------------------------------------------

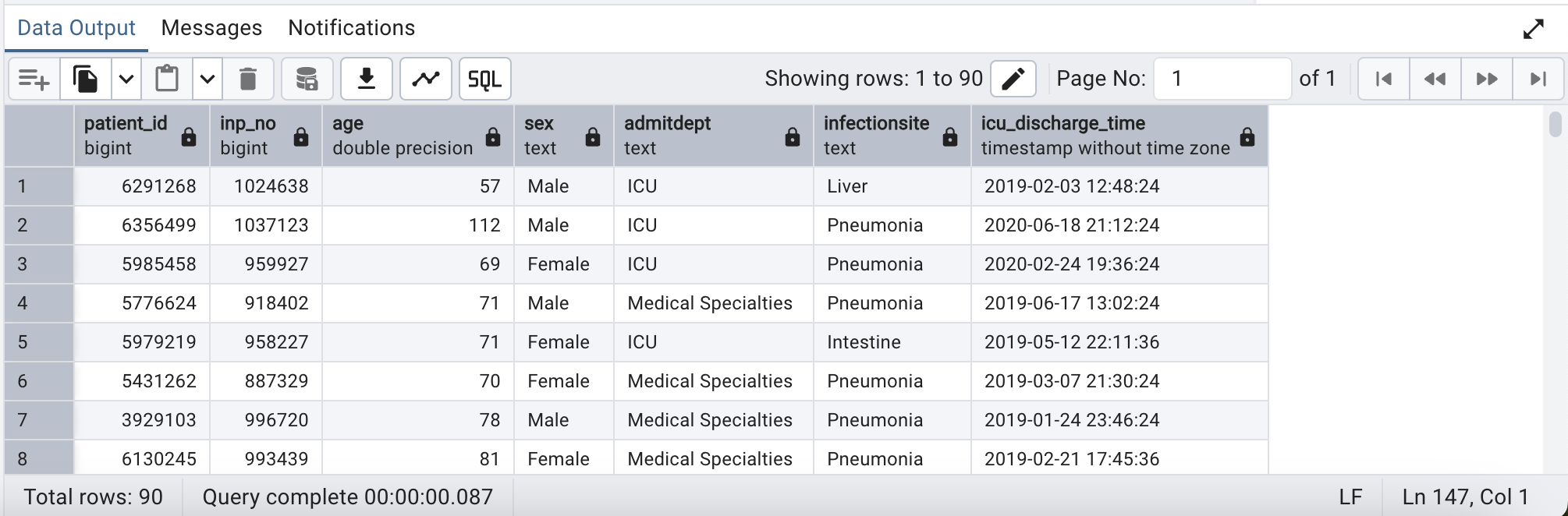
**7)Find patients in the baseline table whose ages are greater than any of the ages listed in a specified array?**

**Query:**

select \*

from baseline

where age > any (array[40,50]);



------------------------------------------------------------------------------------------------------------------

**8)Find the cumulative distribution of patients' ages in the baseline table, and identify what percentage of patients are younger than a specific age threshold?**

**Query:**

select

patient\_id,

age,

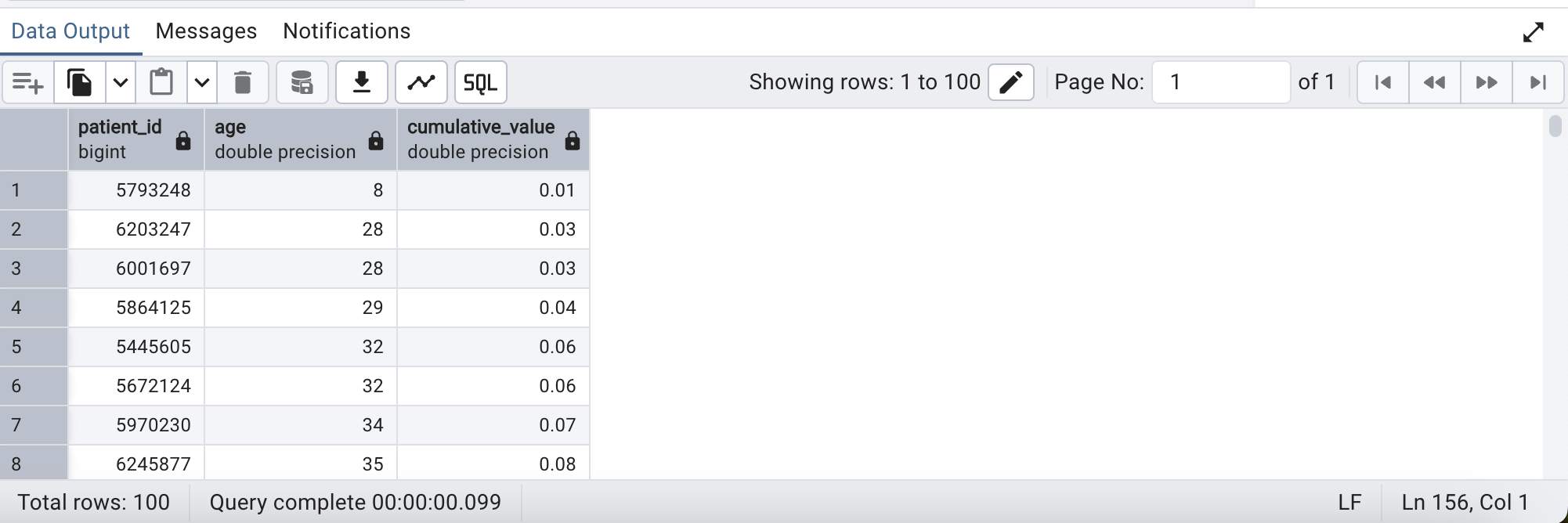
cume\_dist() over (order by age) as cumulative\_value

from

baseline

where

age is not NULL;



----------------------------------------------------------------------------------------------------------------

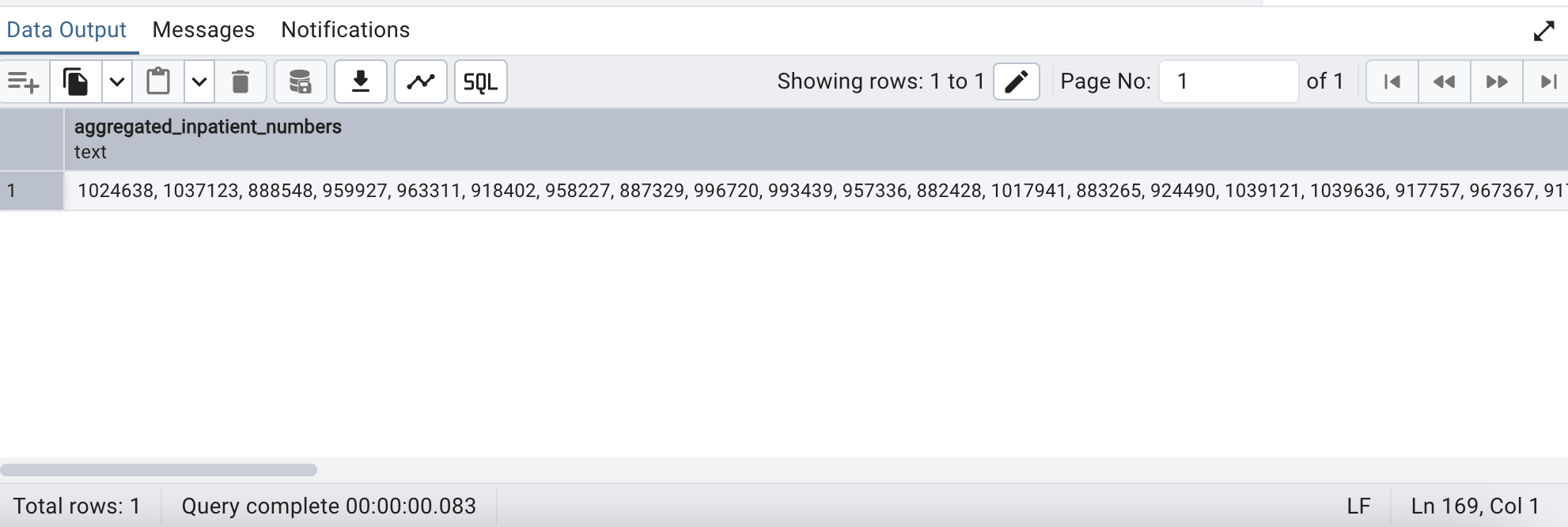
**9)Aggregate all inpatient\_number values in the baseline table into a comma-separated string format**

**Query:**

select

string\_agg(inp\_no::text, ', ') AS aggregated\_inpatient\_numbers

from baseline;



----------------------------------------------------------------------------------------------------------------------------

**10)Categorize patient ages into specific age groups using the width\_bucket function in the baseline table**

**Query:**

select width\_bucket(age, 0, 100, 5) as age\_width\_bucket,

count(\*) as count\_pat

from

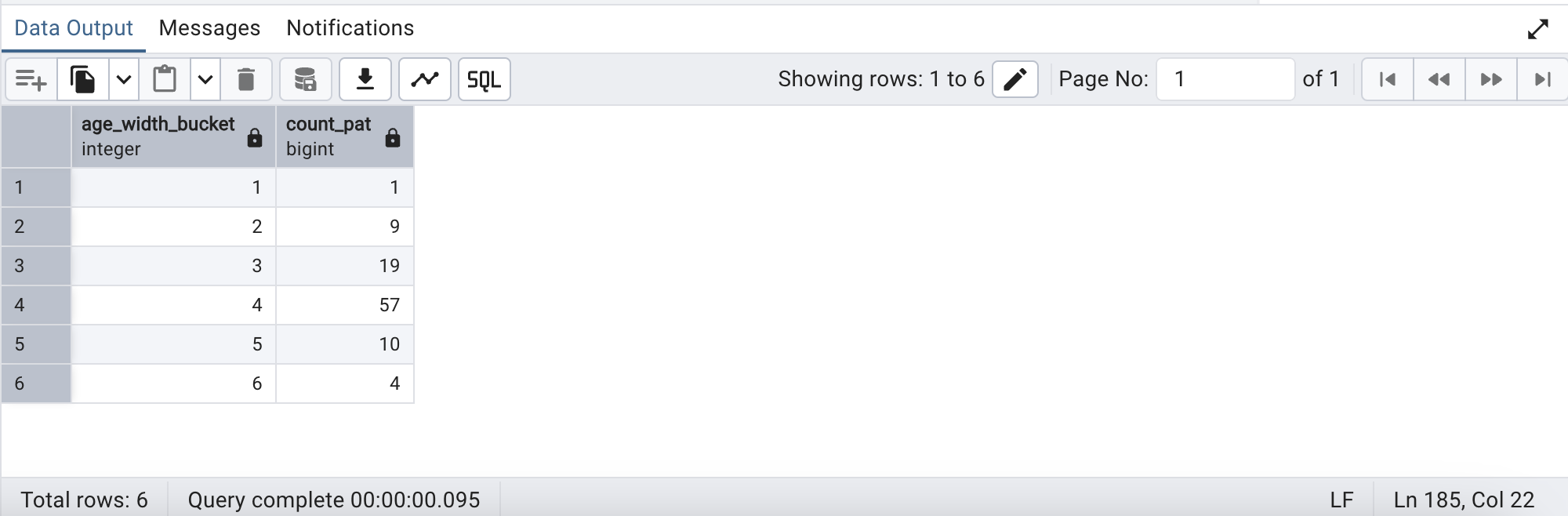
baseline

group by

age\_width\_bucket

order by

age\_width\_bucket;



----------------------------------------------------------------------------------------------------------------------------

**11) Retrieve a specific number of records from the baseline using the FETCH clause in SQL**

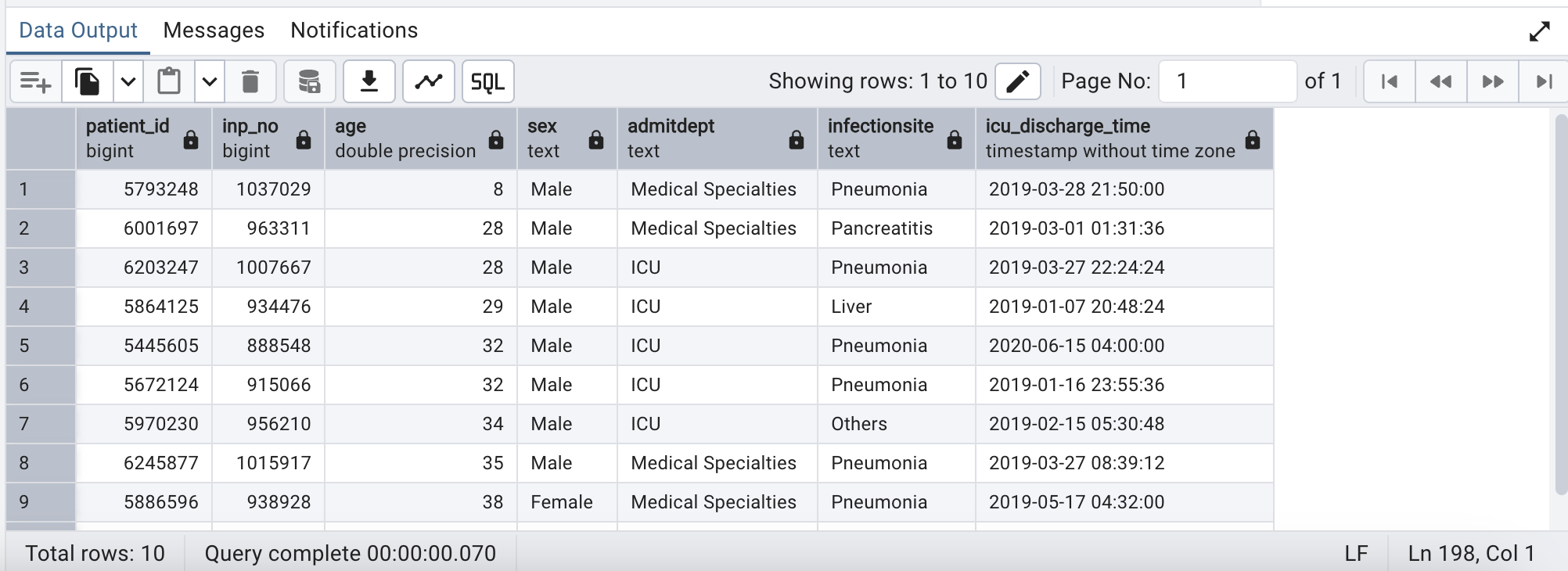
**Query:**

select \*

from baseline

order by age

fetch first 10 rows only;



----------------------------------------------------------------------------------------------------------------------------

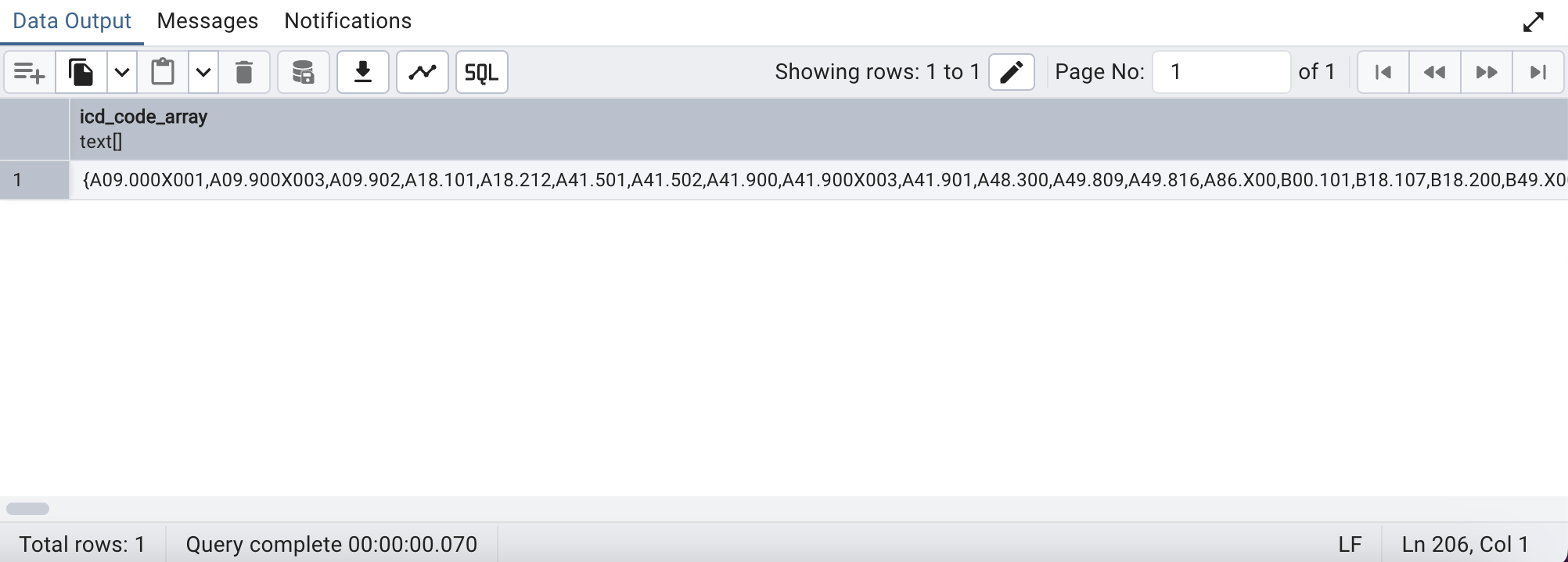
**12) Retrieve all unique types of icd\_code and present them as an array?**

**Query:**

select

array\_agg(distinct icd\_code) as icd\_code\_Array

from icd;



----------------------------------------------------------------------------------------------------------------------------

**13)Write a query using the PERCENT\_RANK() window function to find patients in the top 10% of heart rate measurements. Display their patient ID, heart rate, and percentile rank.**

**Query:**

with ranked\_heart\_rate as (

select

b.patient\_id,

n.heart\_rate,

percent\_rank() over (partition by b.patient\_id order by n.heart\_rate desc) as heart\_rate\_percentile

from

baseline b

join

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

where

n.heart\_rate is not null

)

select

patient\_id,

heart\_rate,

heart\_rate\_percentile

from

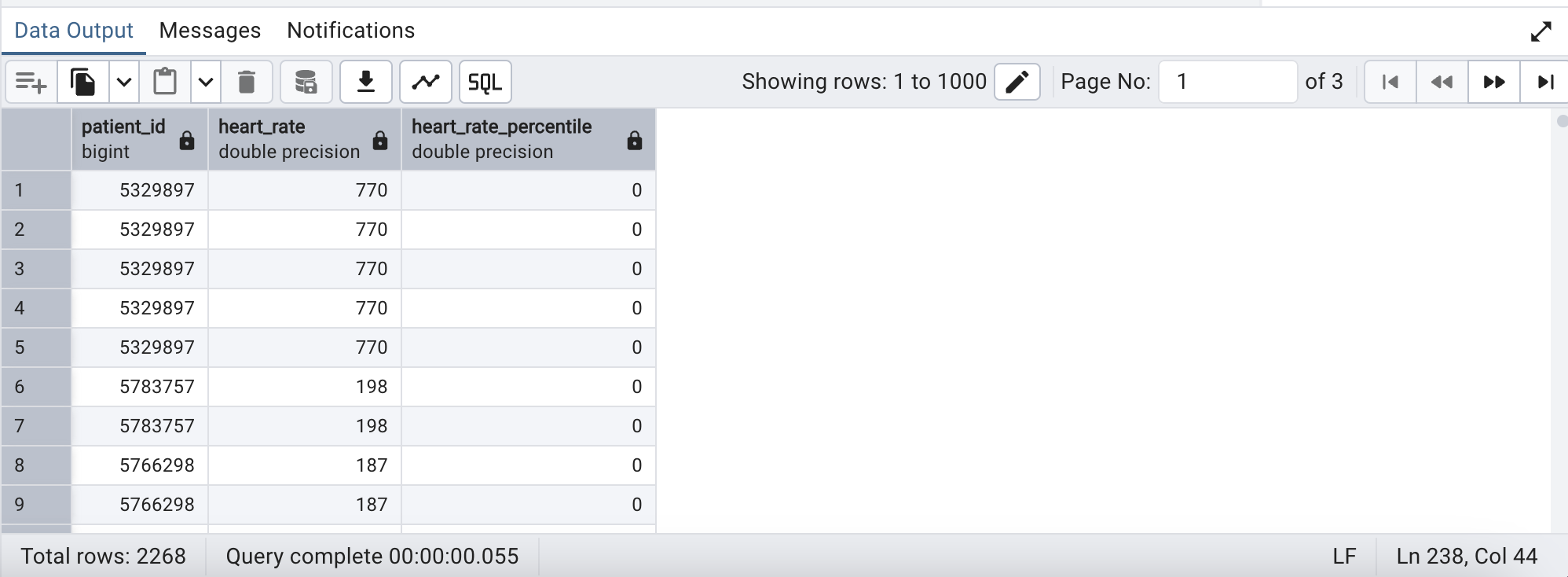
ranked\_heart\_rate

where

heart\_rate\_percentile <= 0.1

order by

heart\_rate\_percentile, heart\_rate desc;



----------------------------------------------------------------------------------------------------------------------------

**14)Find patients whose breathing was in the top 5% during their stay.**

**Query:**

with breathing\_ranked as (

select

inp\_no,

breathing,

ntile(100) over (order by breathing desc) as percentile\_rank

from

nursingchart

where

breathing is not null

)

select

inp\_no,

breathing

from

breathing\_ranked

where

percentile\_rank <= 5

order by

breathing desc;



----------------------------------------------------------------------------------------------------------------------------

**15)Using the nursingchart table, identify instances where a patient's heart rate significantly increases compared to the subsequent chart time. Display the patient ID, current heart rate, subsequent heart rate, and the respective chart times.**

**Query:**

with HeartRateComparison as(

select

inp\_no,

heart\_rate as current\_heart\_rate,

lead(heart\_rate) over (partition by inp\_no order by charttime) as subsequent\_heart\_rate,

charttime as current\_chart\_time,

lead(charttime) over (partition by inp\_no order by charttime) as subsequent\_chart\_time

from

nursingchart

)

select

inp\_no,

current\_heart\_rate,

subsequent\_heart\_rate,

current\_chart\_time,

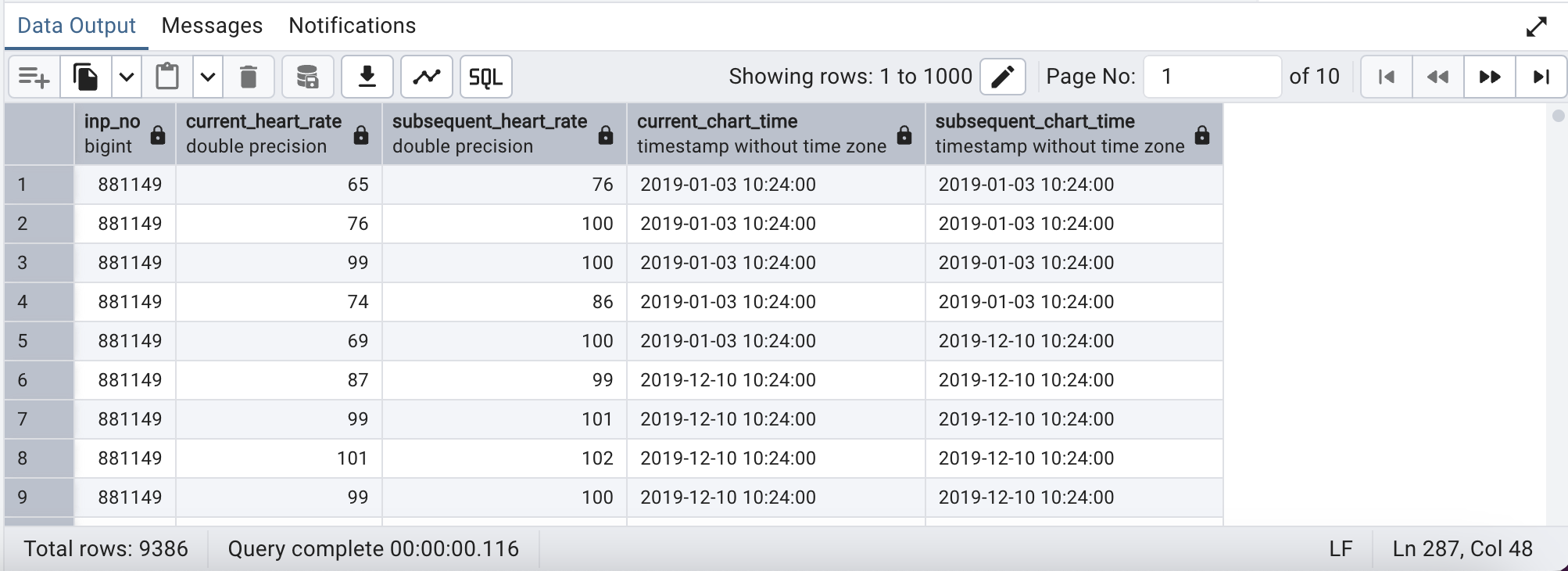
subsequent\_chart\_time

from

HeartRateComparison

where

current\_heart\_rate < subsequent\_heart\_rate;



----------------------------------------------------------------------------------------------------------------------------

**16) Select Rows for baseline in a Range of ctid Values between 11 to 20**

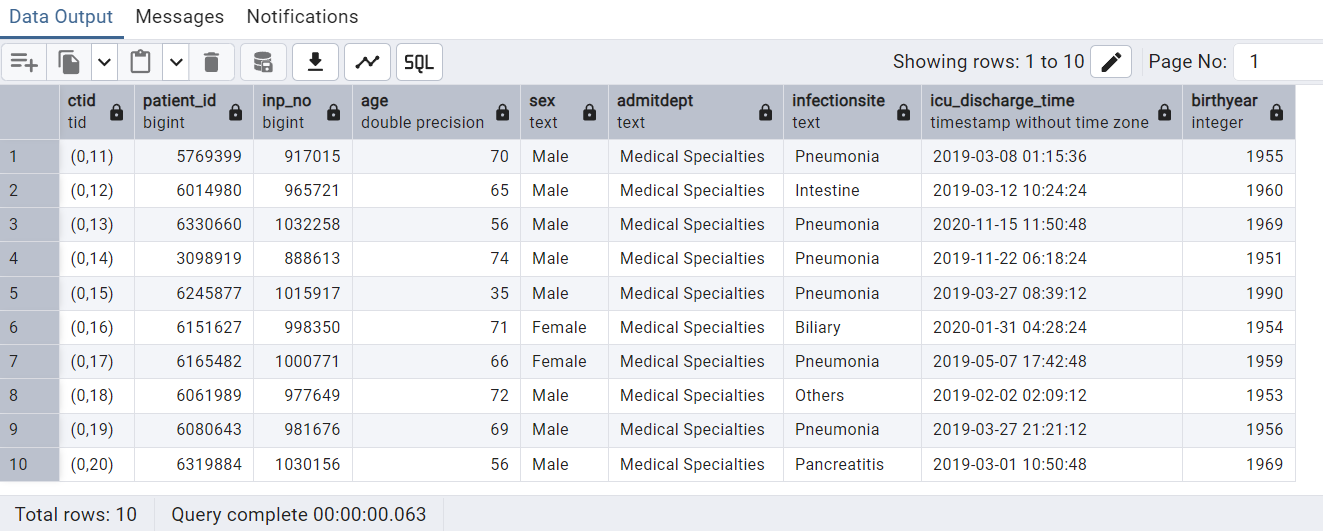
**Query:**

SELECT ctid, \*

FROM baseline

WHERE ctid BETWEEN '(0,10)' AND '(0,20)';

**Output:**

****

**—----------------------------------------------------------------------------------------------------------------------------**

**Q.17 )Display NTH\_VALUE() age function to return all patients with the second most age from table.**

**Query:**

select patient\_id,

age,

NTH\_VALUE(age, 2)

OVER(

ORDER BY age DESC

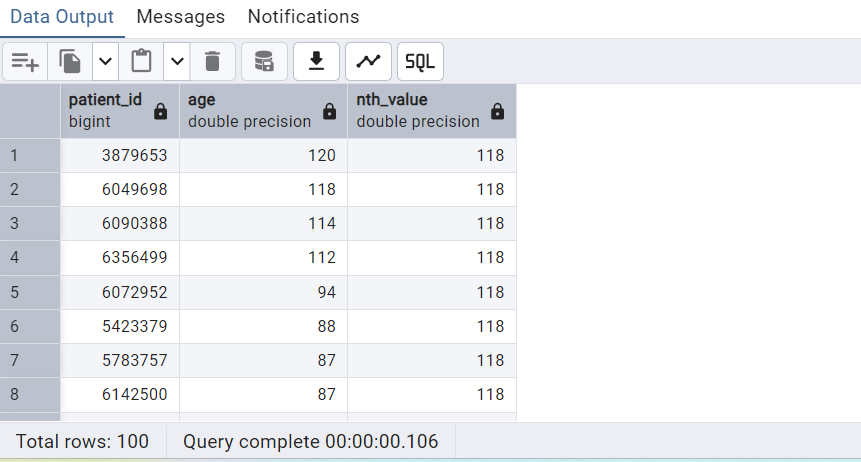
RANGE BETWEEN

UNBOUNDED PRECEDING AND

UNBOUNDED FOLLOWING

) FROM baseline;

**Output:**

****

—----------------------------------------------------------------------------------------------------------------------------

**Q.18) Create materialized view with no data, to display no of male and female patients.**

Query:

CREATE MATERIALIZED VIEW patient\_gender\_count

AS

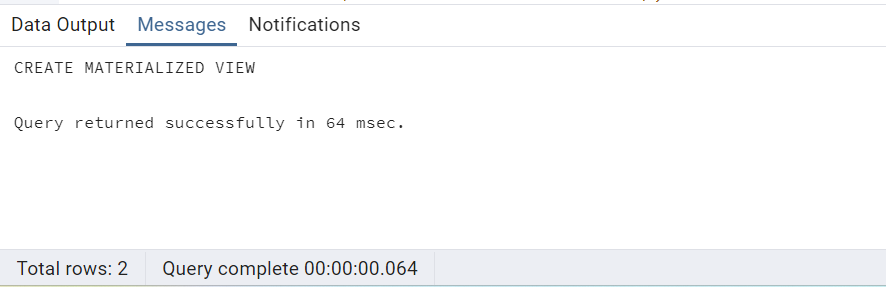
SELECT sex, COUNT(sex) AS patient\_count

FROM baseline

GROUP BY sex

WITH NO DATA;

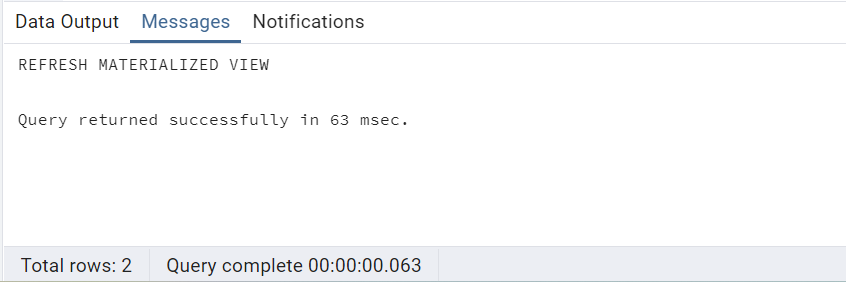
Output:



–to refresh materialized view

REFRESH MATERIALIZED VIEW patient\_gender\_count;

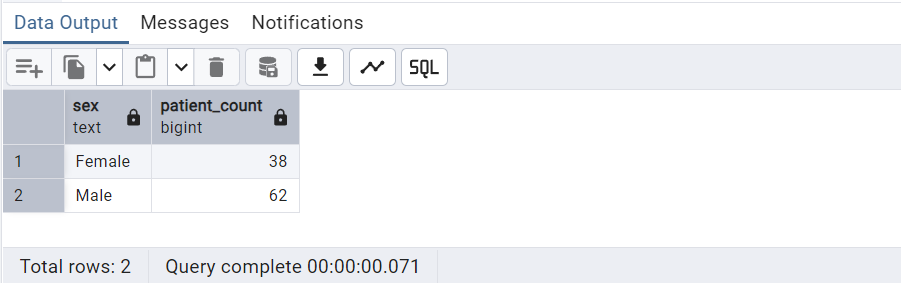
Output:



–query to select materialized view

select \* from patient\_gender\_count;

Output:



—----------------------------------------------------------------------------------------------------------------------------

**Q. 19) Using implicit cursor write a stored procedure to select the given patient details from the procedure call (patient\_id,sex,admit department and drug name).**

**Query:**

CREATE OR REPLACE PROCEDURE patient\_information(patient\_no int)

LANGUAGE plpgsql

AS

$$

DECLARE

patient\_id int;

sex text;

admitdept text;

drug\_name text;

is\_records Boolean := false;

BEGIN

FOR patient\_id , sex,admitdept,drug\_name IN

SELECT b.inp\_no, b.sex, b.admitdept, dr.drugname AS gender

FROM baseline b JOIN drugs dr ON b.patient\_id = dr.patient\_id

WHERE b.patient\_id = patient\_no

LOOP

is\_records := true ;

RAISE NOTICE 'Patient ID: %, Patient Gender: %, Admit Department: %, Drug Name: %', patient\_no, sex, admitdept,drug\_name;

END LOOP;

IF NOT is\_records THEN

RAISE NOTICE 'No patients found for Patient ID: %', patient\_no;

END IF;

END

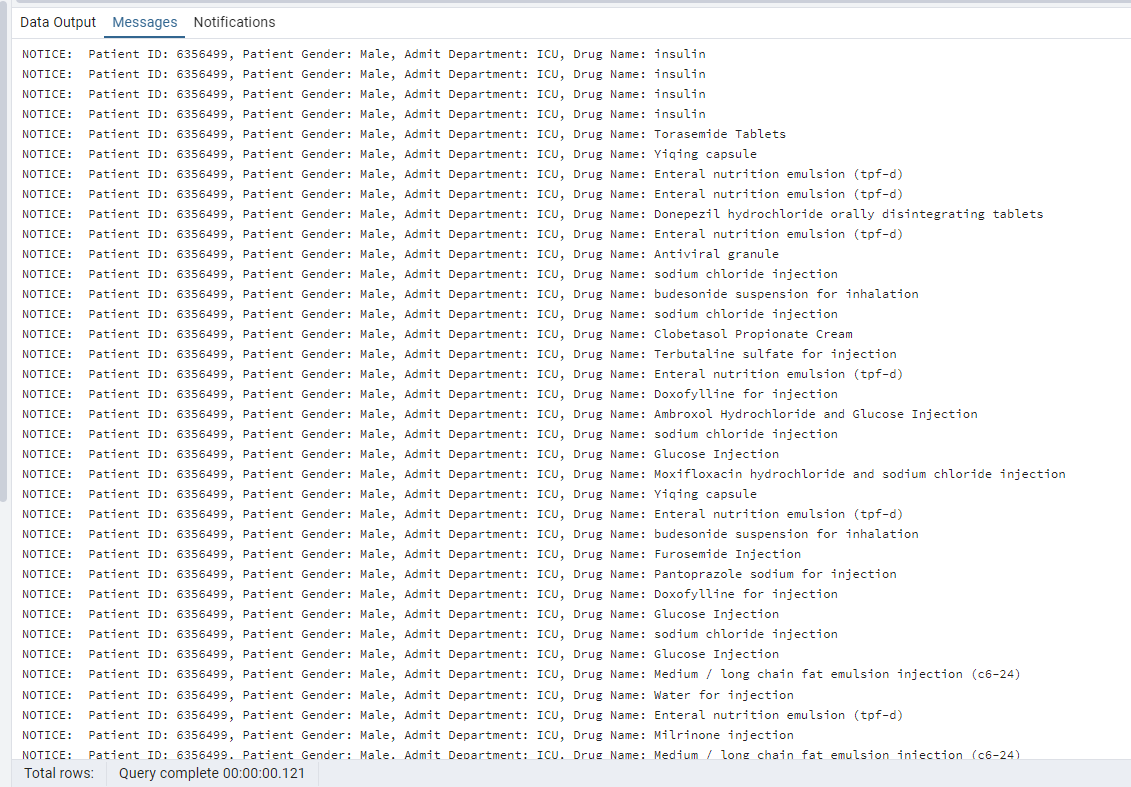
$$;

Call the procedure:

--------------------------

CALL patient\_information(6356499);

**Output:**

****

**20) Create a stored procedure to calculate the average heart rate of a given patient. Use IN and OUT parameters**

**Query:**

**Creating the procedure**

**—-----------------------------**

CREATE OR REPLACE PROCEDURE calculate\_average\_heart\_rate(

IN patient\_id\_input INT, -- IN parameter to filter by patient\_id

OUT avg\_heart\_rate NUMERIC -- OUT parameter to return the average

)

LANGUAGE plpgsql

AS $$

BEGIN

SELECT round(AVG(heart\_rate)::numeric,2)

INTO avg\_heart\_rate

FROM nursingchart

WHERE inp\_no = patient\_id\_input;

IF avg\_heart\_rate IS NULL THEN

avg\_heart\_rate := 0;

END IF;

END;

$$;

**Procedure Call for output**

—-------------------------------

DO $$

DECLARE

avg\_rate NUMERIC;

BEGIN

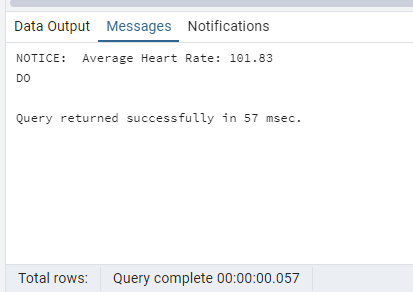
CALL calculate\_average\_heart\_rate(974750, avg\_rate);

RAISE NOTICE 'Average Heart Rate: %', avg\_rate;

END;

$$;

**Output:**

****

**21) Create a table patients and generate transaction control and error handling with BEGIN, COMMIT, ROLLBACK, and EXCEPTION.**

**Query:**

**—------**

CREATE OR REPLACE PROCEDURE table\_manipulation\_transaction(

IN p\_id INT,

IN p\_date VARCHAR,

IN p\_name TEXT,

IN inpno REAL

)

LANGUAGE plpgsql

AS $$

BEGIN

INSERT INTO patients(patient\_id, entry\_date, patient\_name, inp\_no)

VALUES (p\_id, p\_date, p\_name, inpno);

DELETE FROM patients

WHERE inp\_no < inpno;

UPDATE patients

SET inp\_no = 4500

WHERE patient\_id = p\_id;

IF NOT FOUND THEN

RAISE EXCEPTION 'Patient ID % not found for updating inp\_no.', p\_id;

END IF;

IF p\_name IS NULL THEN

RAISE EXCEPTION 'Patient name must be provided.';

END IF;

EXCEPTION

WHEN OTHERS THEN

-- Handle exceptions by reporting an error

RAISE NOTICE 'Transaction rolled back due to an error: %', SQLERRM;

RAISE EXCEPTION 'Error: %', SQLERRM;

END;

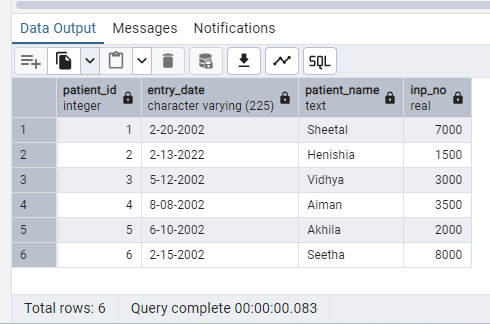
$$;

**Procedure Call**

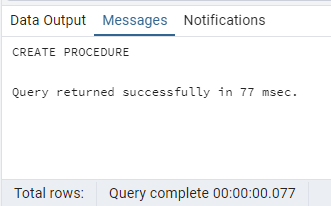
**—-------------------**

CALL table\_manipulation\_transaction(7, '12-12-2012', 'Rani', 5000);

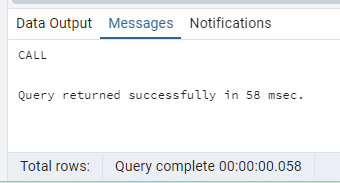
**Output: 1 (created table)**

****

**Output: 2 (code)**

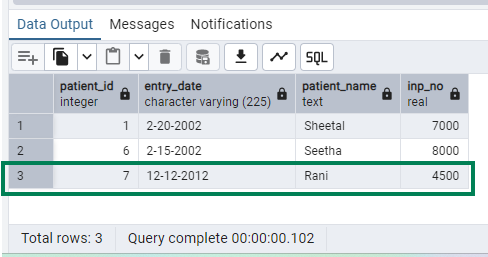


**Output: 3 (Call)**

****

**Output:4 (Updated and deleted rows)**

**(Deleted rows will not be shown as they are already deleted)**

****

**—----------------------------------------------------------------------------------------------------------------------------**

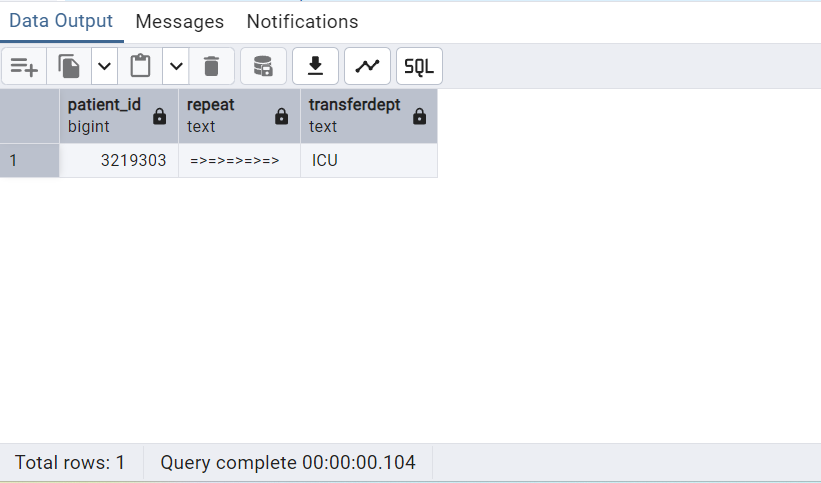
**22)write a query to repeat transfer department as per patient\_id id 5 times whose dept name ='ICU'**

**Query:**

select patient\_id, repeat('=>',5),transferdept from transfer

where transferdept = 'ICU'

limit 1;

****

**—----------------------------------------------------------------------------------------------------------------------------**

**23)write a query to find the position of letter a in firstname of patients USING strpos function**

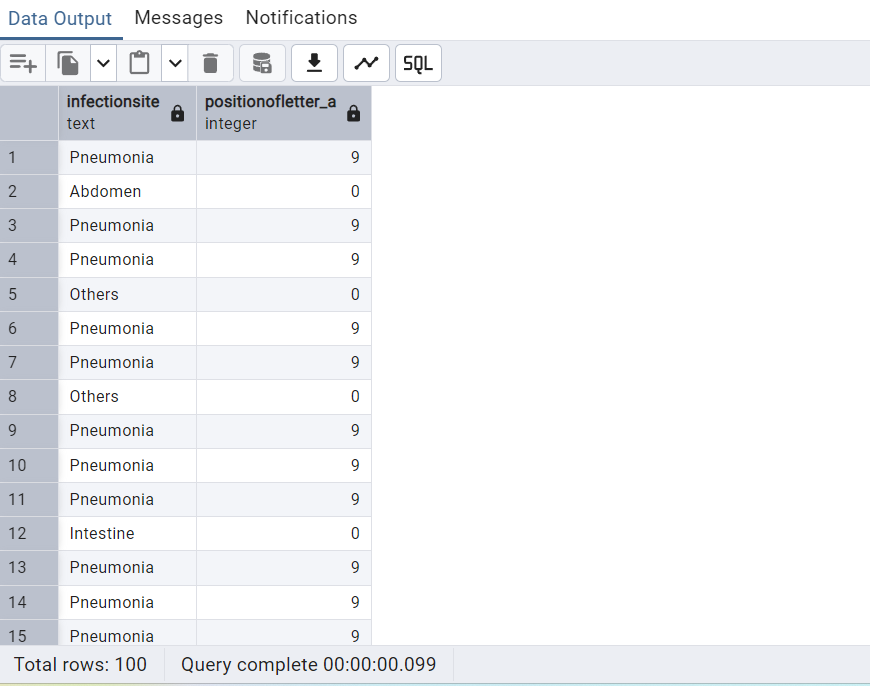
**Query:**

select infectionsite,strpos(infectionsite,'a') as positionofletter\_a from baseline

**–selecting table**

select \* from baseline;

**Output:**

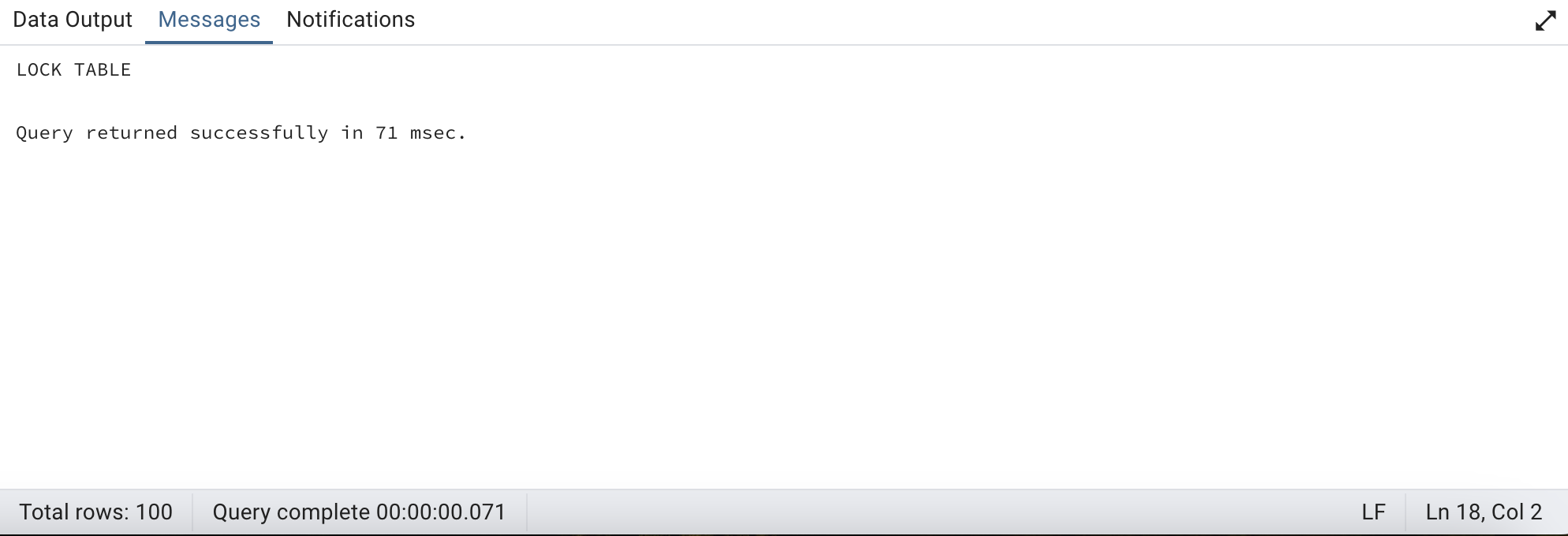


**24)Write a query to lock the table**

**--creates lock**

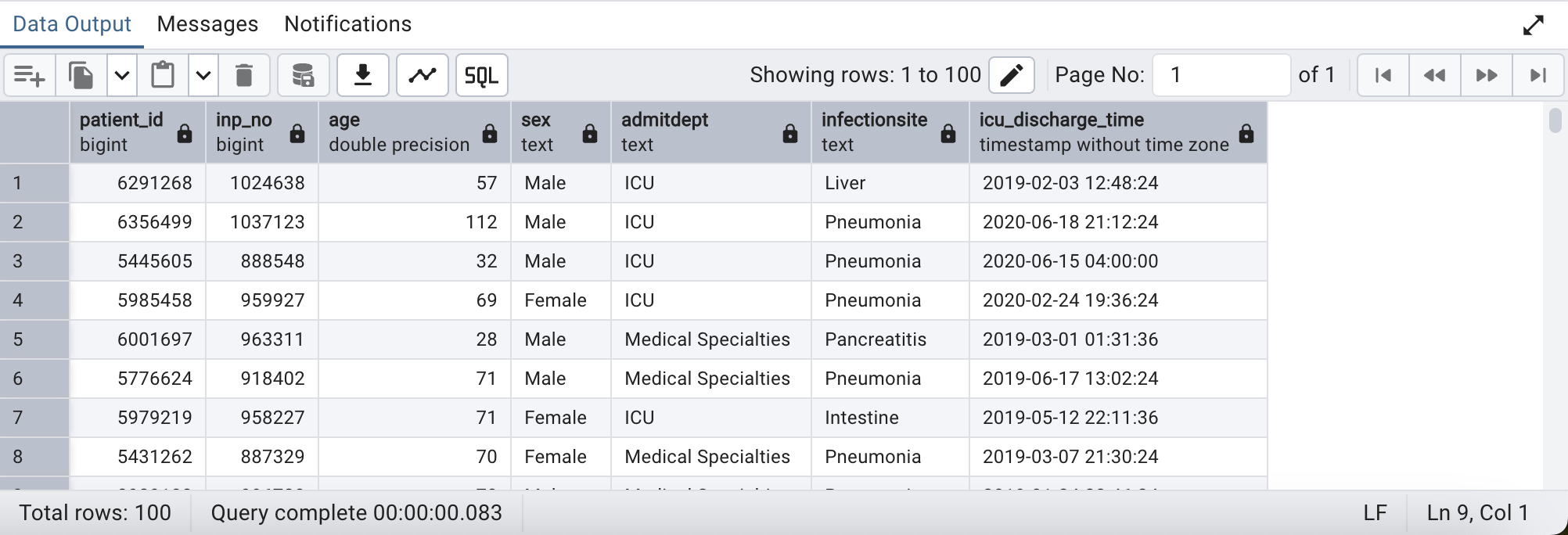
begin;

lock table baseline IN ACCESS SHARE MODE;



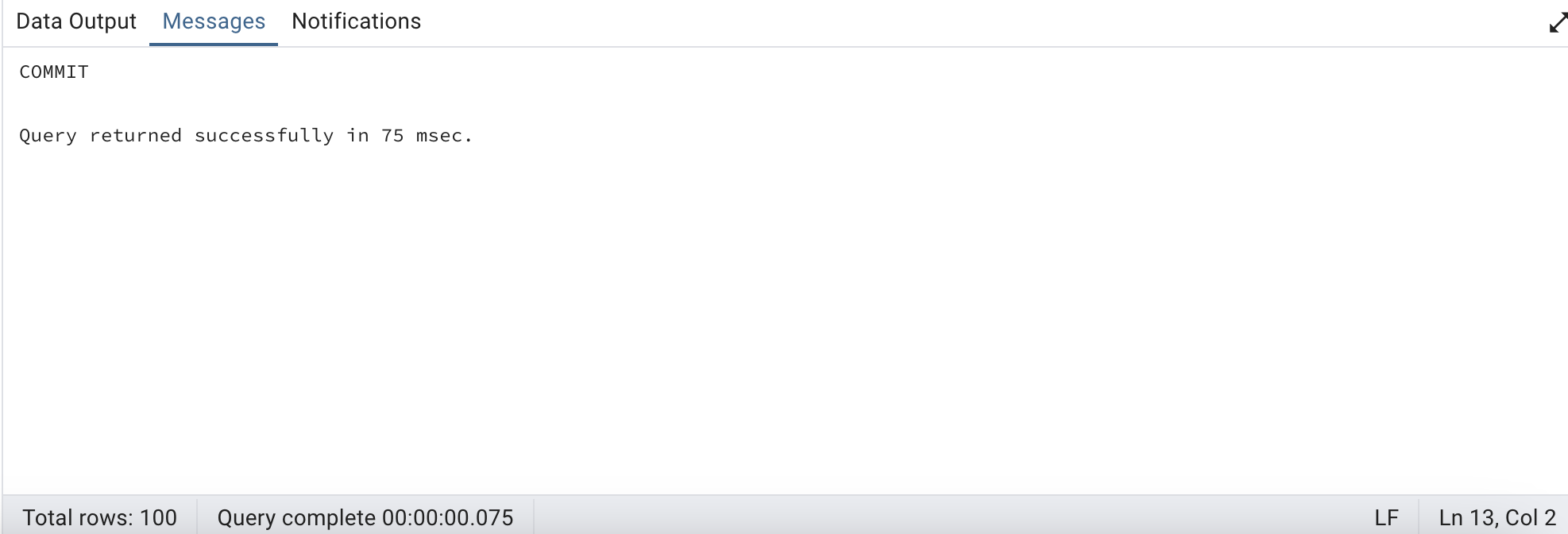
**--validation**

select \* from baseline



**--To Save**

Commit;



**--To Rollback**

Rollback;

