TEAM13-GROUPING LEGENDS-SQL HACKATHON

Category 1 Questions: SIMPLE QUERIES

1. List the total count of patients from the demography table.

QUERY:

```
select count(DISTINCT patientid) from gv_demography
```

OUTPUT:

Insights:

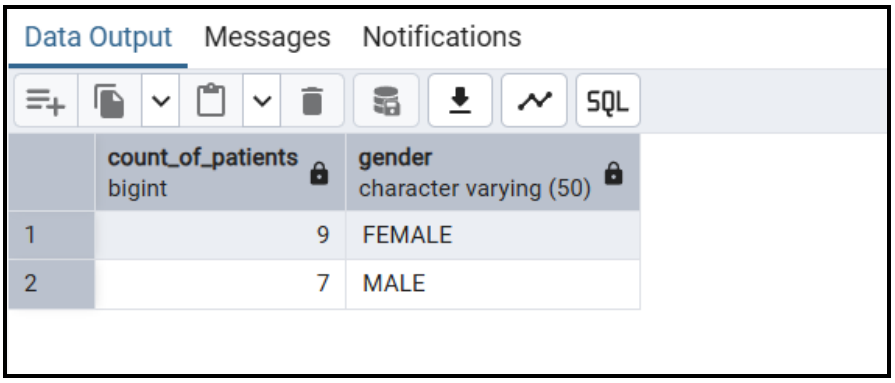
This above result gives the total number of patients in this dataset.

2. List the total count of male/female from the demography table.

QUERY:

```
select count(DISTINCT patientid) count_of_patients,gender
from gv_demography
group by gender
```

OUTPUT:



	count_of_patients bigint	gender character varying (50)
1	9	FEMALE
2	7	MALE

Insights:

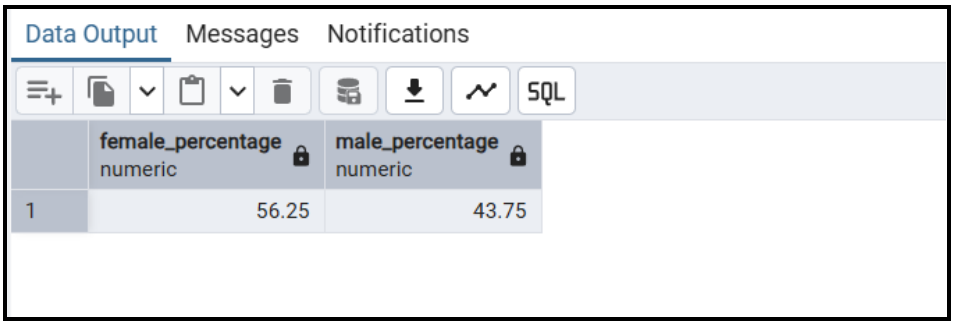
The above result gives the number of male and female patients in this dataset.

3. What percentage of dataset is maleVsfemale?

QUERY:

```
14
15  SELECT ROUND(COUNT(CASE WHEN gender = 'FEMALE' THEN 1 END) * 100.0 / COUNT(*),2)
16  AS Female_percentage,
17  ROUND(COUNT(CASE WHEN gender = 'MALE' THEN 1 END) * 100.0 / COUNT(*),2)
18  AS Male_percentage FROM gv_demography;
19
20
```

OUTPUT:



	female_percentage numeric	male_percentage numeric
1	56.25	43.75

Insights:

This result gives the percentage of male and female patients in the dataset.

4. **Which patient has the maximum hba1c?**

QUERY:

```
22
23  select patientid,gender,hba1c
24  from gv_demography
25  where hba1c=(select max(hba1c) from gv_demography)
```

OUTPUT:

Data Output | Messages | Notifications

	patientid [PK] integer	gender character varying (50)	hba1c numeric (4,2)
1	4	FEMALE	6.40

Insights:

The above result gives the patient who has maximum hba1c value measured.

5. **Find the overall Average glucose level of patients with their gender?**

QUERY:

```
select g.patientid ,d.gender ,
ROUND(avg(g.glucosevaluemgdl),2) as avg_glucose
from gv_dexcom g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,d.gender
ORDER BY g.patientid
```

OUTPUT:

Data Output | Messages | Notifications

	patientid integer	gender character varying (50)	avg_glucose numeric
1	1	FEMALE	106.09
2	2	MALE	130.43
3	3	FEMALE	107.89
4	4	FEMALE	112.65
5	5	FEMALE	104.73
6	6	FEMALE	124.59
7	7	FEMALE	93.07

Total rows: 16 Query complete 00:00:00.058

Insights:

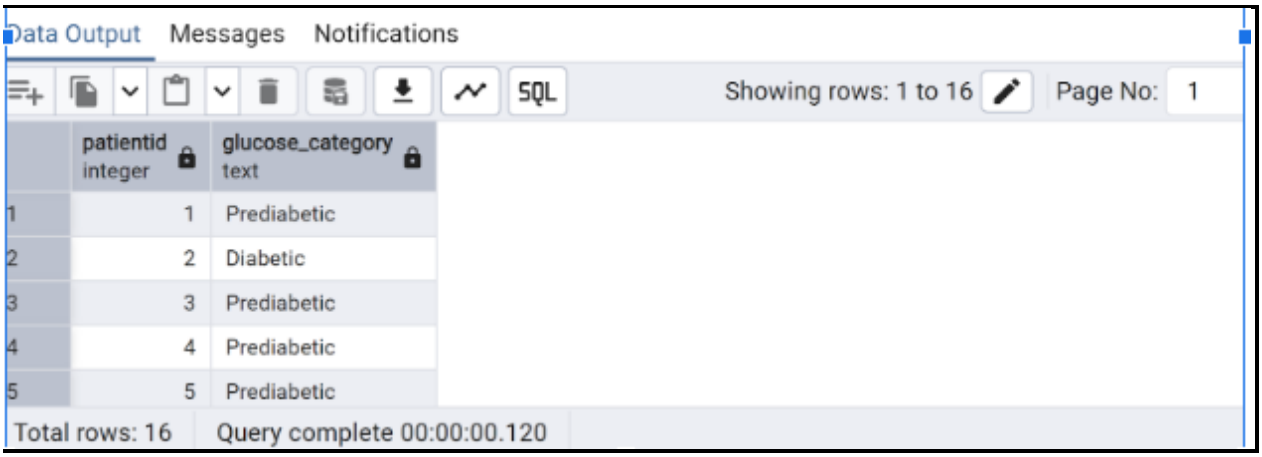
The above result gives the overall average glucose levels of patients.

6. **Categorize patients based on their glucose category**

QUERY:

```
-- 3.Group patients by the glucose range as normal, prediabetic or diabetic
SELECT
    patientid,
    CASE
        WHEN avg(glucosevaluemgdl) < 100 THEN 'Normal'
        WHEN avg(glucosevaluemgdl) BETWEEN 100 AND 125 THEN 'Prediabetic'
        WHEN avg(glucosevaluemgdl) >= 125 THEN 'Diabetic'
    END AS glucose_category
FROM public.gv_dexcom
group by patientid
order by patientid ;
```

OUTPUT:



Insights: By categorizing patients based on glucose values,we understand risk distribution and could guide 'PreDiabetics' to prevent progression to Diabetes.

7. **Display the count of patients by glucose category (‘normal’, ‘prediabetic’ or ‘diabetic’)**

QUERY:

```
98  ---4. patients count by category (normal, prediabetic or diabetic)
99  select glucose_category, count(*)  from
100 (SELECT
101     patientid,
102     CASE
103         WHEN avg(glucosevaluemgdl) < 100 THEN 'Normal'
104         WHEN avg(glucosevaluemgdl) BETWEEN 100 AND 125 THEN 'Prediabetic'
105         WHEN avg(glucosevaluemgdl) >= 125 THEN 'Diabetic'
106     END AS glucose_category
107 FROM public.gv_dexcom
108 group by patientid
109 order by patientid)
110 group by glucose_category;
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 3 | Page No: 1

	glucose_category text	count bigint
1	Diabetic	3
2	Normal	1
3	Prediabetic	12

Total rows: 3 | Query complete 00:00:00.078

Insights: In the given dataset, the majority fall under Pre-Diabetics. Counts help in resource allocation

8. **Show total calorie intake of patients in a day**

QUERY:

```
---- 5. show total calorie intake of patients in a day
select gv_food.patientid, sum(gv_food.calorie)as Total_Calorie_intake,gv_food_log.dateof from
gv_food
join gv_food_log on gv_food.food_id = gv_food_log.food_id
group by gv_food_log.dateof,gv_food.patientid
order by gv_food.patientid
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 139 | Page No: 1 of 1

	patientid integer	total_calorie_intake numeric	dateof timestamp without time zone
1	1	944	2020-02-13 00:00:00
2	1	1760	2020-02-14 00:00:00
3	1	1715	2020-02-15 00:00:00
4	1	1081	2020-02-16 00:00:00
5	1	860.7	2020-02-17 00:00:00
6	1	1802.5	2020-02-18 00:00:00
7	1	1521	2020-02-19 00:00:00
8	1	1050	2020-02-20 00:00:00

Insights : Understand which patients consume more or fewer calories and identify trends or outliers like unusually high or low intake days.

9. **Follow-up to the above Query: Compare calorie intake of patient per day with gender**

QUERY:

```
select d.patientid,d.gender, f.Total_Calorie_intake, f.dateof from gv_demography d
join
(select gv_food.patientid,
    sum(gv_food.calorie) as Total_Calorie_intake,gv_food_log.dateof from gv_food
    join gv_food_log on gv_food.food_id = gv_food_log.food_id
    group by gv_food_log.dateof,gv_food.patientid
    order by gv_food.patientid )as f on d.patientid = f.patientid
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 139 | Page No: 1

	patientid integer	gender character varying (50)	total_calorie_intake numeric	dateof timestamp without time zone
1	1	FEMALE	944	2020-02-13 00:00:00
2	1	FEMALE	1760	2020-02-14 00:00:00
3	1	FEMALE	1715	2020-02-15 00:00:00
4	1	FEMALE	1081	2020-02-16 00:00:00
5	1	FEMALE	860.7	2020-02-17 00:00:00
6	1	FEMALE	1802.5	2020-02-18 00:00:00
7	1	FEMALE	1521	2020-02-19 00:00:00

Total rows: 139 Query complete 00:00:00.133

Insights: This query provides gender-based dietary analysis showing calorie intake

of each patient and helps in extending the analysis about glucose control, weight or other health metrics.

10. **Average glucose within 3 hours after food logged?**

QUERY:

```
----- 7.  Average glucose within 3 hours after food logged
SELECT f.patientid, f.logged_food,
       ROUND(AVG(d.glucosevaluemgdl), 2) AS avg_postmeal_glucose
FROM gv_food_log f
JOIN gv_dexcom d
  ON f.patientid = d.patientid
 AND d.timeof BETWEEN f.time_begin AND f.time_begin + interval '3 hour'
GROUP BY f.patientid, f.logged_food
ORDER BY avg_postmeal_glucose DESC;
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 546 | Page No: 1 of 1

	patientid integer	logged_food character varying (200)	avg_postmeal_glucose numeric
1	6	M&Ms	191.03
2	10	Medium French Fries	185.75
3	10	Diet Pepsi	185.75
4	10	Chicken Gyro	185.75
5	6	3/4 large blueberry pancake	184.64
6	10	Lamb Gyro	179.00

Total rows: 546 | Query complete 00:00:00.825 | CRLF

Insights: This query tells the average glucose reading of each patient 3 hrs

after food intake. In Prescriptive analysis, this is useful in dietary recommendations for Diabetes management.

11. List all the foods with their calories from highest to lowest?

QUERY:

```
/* List all the foods with their calories from highest to lowest*/

select max(calorie) as maximum_calorie_food,logged_food as food
from gv_food
group by food_id,logged_food
order by maximum_calorie_food desc
```

OUTPUT:

Data Output | Messages | Notifications

	maximum_calorie_food numeric	food character varying (200)
1	2245	(Pillsbury) Cinammon Rolls
2	1808	Plain cheese pizza
3	1360	(Red Baron) Brick Oven Pepperoni Pizza
4	1320	Totinos pizza
5	1235	Sausage Biscuit
6	1210	(Outback Steakhouse) Chicken Tacos
7	1181.3	cookie
8	1145	Cheddar Cheese

Total rows: 1364 Query complete 00:00:00.190

Insights:

The above result gives the count of calories of foods in descending order from highest to the least. This helps to determine which foods tend cause spike in glucose levels in patients.

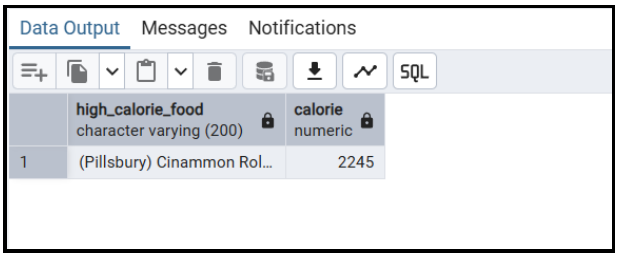
12. **List the food with the maximum calorific value and sugar value?**

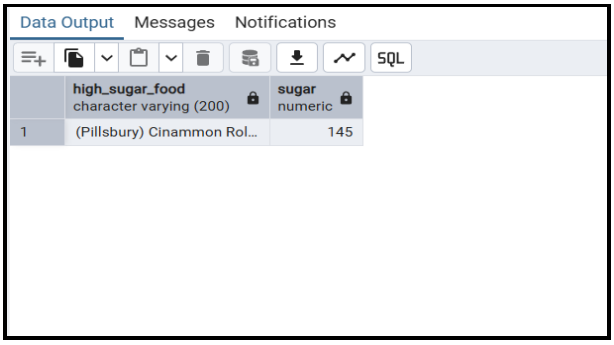
QUERY:

```
/* Food with the maximum calorific value*/

select distinct logged_food as high_calorie_food,calorie
from gv_food
where calorie=(select max(calorie) as high_calorie_food
from gv_food )
order by calorie desc
```

```
/* Food with the maximum sugar content*/

select distinct logged_food as high_sugar_food,sugar
from gv_food
where sugar=(select max(sugar) as high_sugar_food
from gv_food )
```

OUTPUT:





Insights:

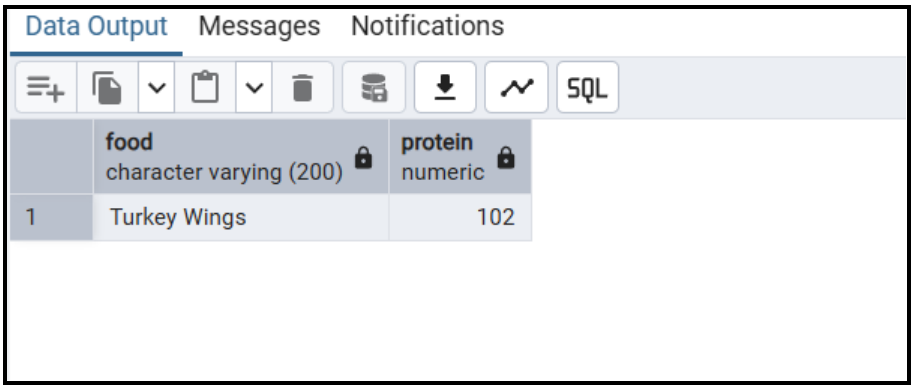
The above result gives the food which has maximum calorie content and sugar content. This also helps in determining whether they are responsible for glucose spike in patients.

13.List the food which has maximum protein content?

QUERY:

```
/* List the foods which have maximum protein content*/


select distinct logged_food as food,protein
from gv_food
where protein=(select max(protein) as high_protein_food
from gv_food )
```

OUTPUT:



Insights:

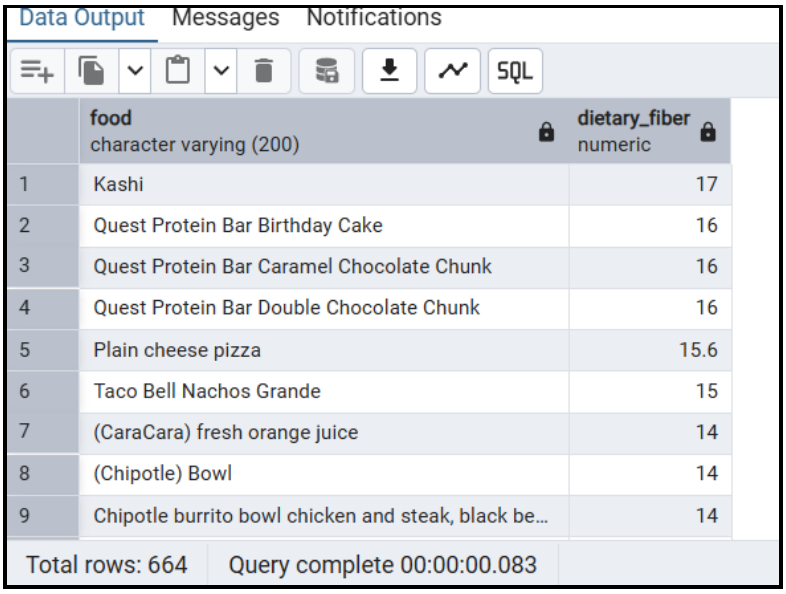
The above result gives the food which is high on protein compared to all other foods. High protein diet does not cause glucose to spike in the body and is good for controlling diabetes.

14.List all the foods with their fiber content highest to lowest?

QUERY:

```
97
98  /* List all the foods with their fiber content highest to lowest*/
99
00  select distinct logged_food as food, max(dietary_fiber) as dietary_fiber
01  from gv_food
02  where dietary_fiber is not null
03  group by food_id,logged_food,dietary_fiber
04  order by dietary_fiber desc
05
```

OUTPUT:



Data Output | Messages | Notifications

	food character varying (200)	dietary_fiber numeric
1	Kashi	17
2	Quest Protein Bar Birthday Cake	16
3	Quest Protein Bar Caramel Chocolate Chunk	16
4	Quest Protein Bar Double Chocolate Chunk	16
5	Plain cheese pizza	15.6
6	Taco Bell Nachos Grande	15
7	(CaraCara) fresh orange juice	14
8	(Chipotle) Bowl	14
9	Chipotle burrito bowl chicken and steak, black be...	14

Total rows: 664 Query complete 00:00:00.083

Insights:

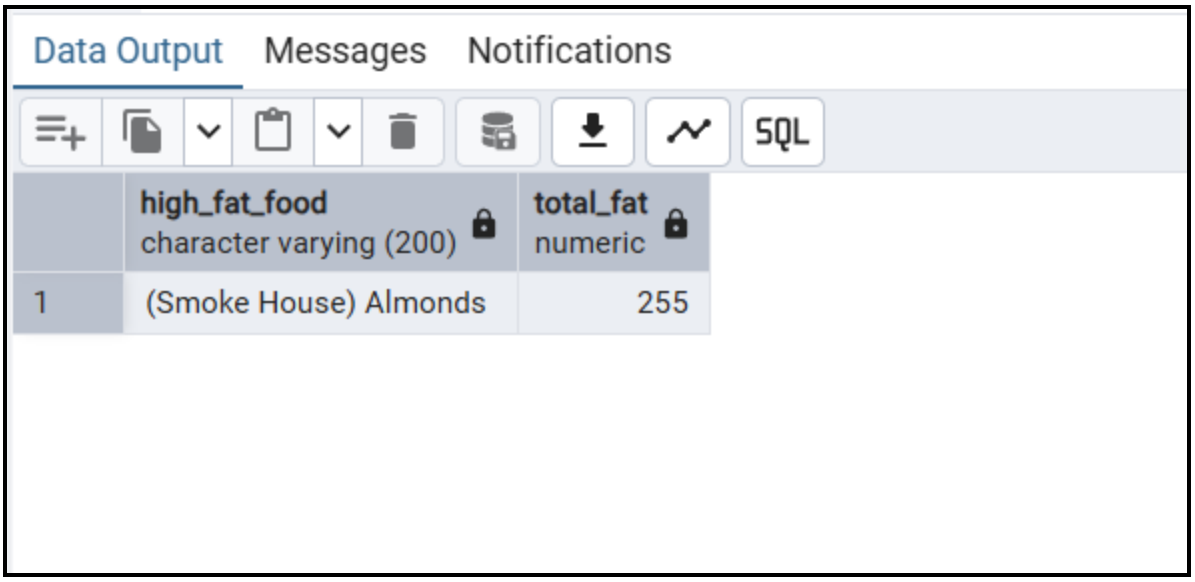
The above result lists all the foods with their dietary fiber content from highest to lowest. High fiber diet is also very good in controlling diabetes and rise in glucose levels.

15.List the foods which are high on fat?

QUERY:

```
/* List the food which is high on fat*/

select distinct logged_food as high_fat_food,total_fat
from gv_food
where total_fat=(select max(total_fat) as high_fat_food
from gv_food )
```

OUTPUT:



Insights:

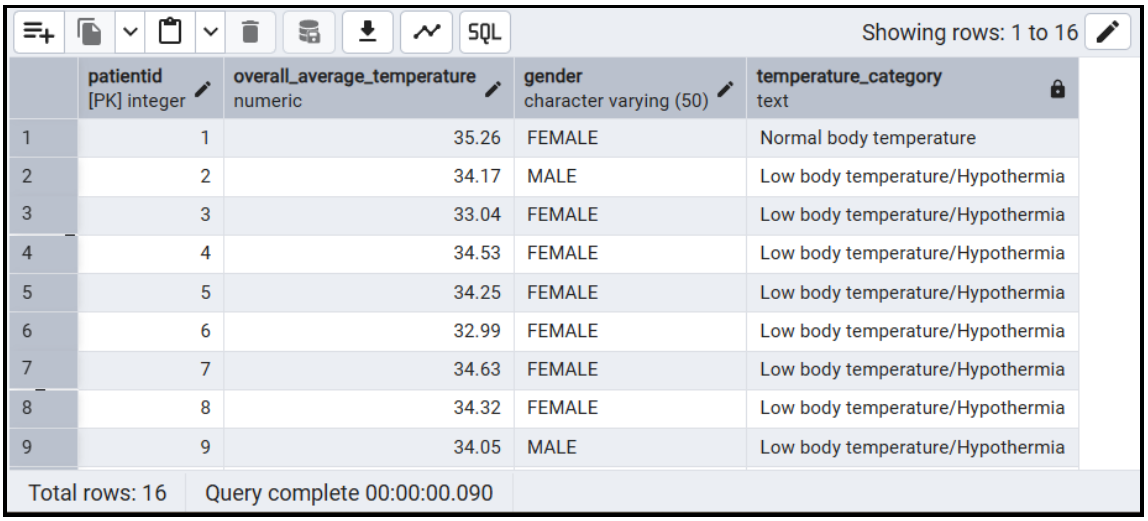
The above query lists the food which is high on fat content. They do play a role in the glucose levels of patients.

16.Categorize patients based on their temperature levels?

QUERY:

```
select patientid,
temperature,
timeof,
CASE
when temperature <35 then 'Low body temperature/Hypothermia'
when temperature BETWEEN 35 and 37 then 'Normal body temperature'
when temperature>37.5 then 'Fever'
END as Temperature_category
FROM gv_temperature
order by patientid
```

OUTPUT:



Showing rows: 1 to 16

	patientid [PK] integer	overall_average_temperature numeric	gender character varying (50)	temperature_category text
1	1	35.26	FEMALE	Normal body temperature
2	2	34.17	MALE	Low body temperature/Hypothermia
3	3	33.04	FEMALE	Low body temperature/Hypothermia
4	4	34.53	FEMALE	Low body temperature/Hypothermia
5	5	34.25	FEMALE	Low body temperature/Hypothermia
6	6	32.99	FEMALE	Low body temperature/Hypothermia
7	7	34.63	FEMALE	Low body temperature/Hypothermia
8	8	34.32	FEMALE	Low body temperature/Hypothermia
9	9	34.05	MALE	Low body temperature/Hypothermia

Total rows: 16 Query complete 00:00:00.090

Insights:

The above query categorizes the patients with their temperature category. High Temperature is categorized as fever and low temperature is categorized as Hypothermia

17.Calculate the average heart rate per day for each patient

QUERY:

```
select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.hr),2) as avg_hr_day_wise
from gv_hr g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid
```

F5

OUTPUT:

Data Output | Messages | Notifications

Showing rows:

	patientid integer	gender character varying (50)	day timestamp without time zone	avg_hr_day_wise numeric
1	1	FEMALE	2020-02-13 00:00:00	74.91
2	1	FEMALE	2020-02-14 00:00:00	74.46
3	1	FEMALE	2020-02-15 00:00:00	72.55
4	1	FEMALE	2020-02-16 00:00:00	76.09
5	1	FEMALE	2020-02-17 00:00:00	77.87
6	1	FEMALE	2020-02-18 00:00:00	71.01
7	1	FEMALE	2020-02-19 00:00:00	68.29
8	1	FEMALE	2020-02-20 00:00:00	73.20
9	1	FEMALE	2020-02-21 00:00:00	77.05

Total rows: 145 Query complete 00:00:00.327

Insights:

The above results gives the day wise average heartrate for each patient. This shows whether the heart rate remains stable or fluctuates each day.

18.To calculate day wise ibi (Interbeat Interval) values for all the patients

QUERY:

```
/*To calculate day wise ibi values for all the patients*/

select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.ibi),2) as avg_ibi_day_wise
from gv_ibi g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid
```

OUTPUT:

Data Output | Messages | Notifications

Showing r

	patientid integer	gender character varying (50)	day timestamp without time zone	avg_ibi_day_wise numeric
1	1	FEMALE	2020-02-13 00:00:00	0.88
2	1	FEMALE	2020-02-14 00:00:00	0.98
3	1	FEMALE	2020-02-15 00:00:00	1.02
4	1	FEMALE	2020-02-16 00:00:00	0.85
5	1	FEMALE	2020-02-17 00:00:00	0.99
6	1	FEMALE	2020-02-18 00:00:00	0.88
7	1	FEMALE	2020-02-19 00:00:00	0.88
8	1	FEMALE	2020-02-20 00:00:00	1.04
9	1	FEMALE	2020-02-21 00:00:00	0.98

Total rows: 145 Query complete 00:00:00.115

Insights:

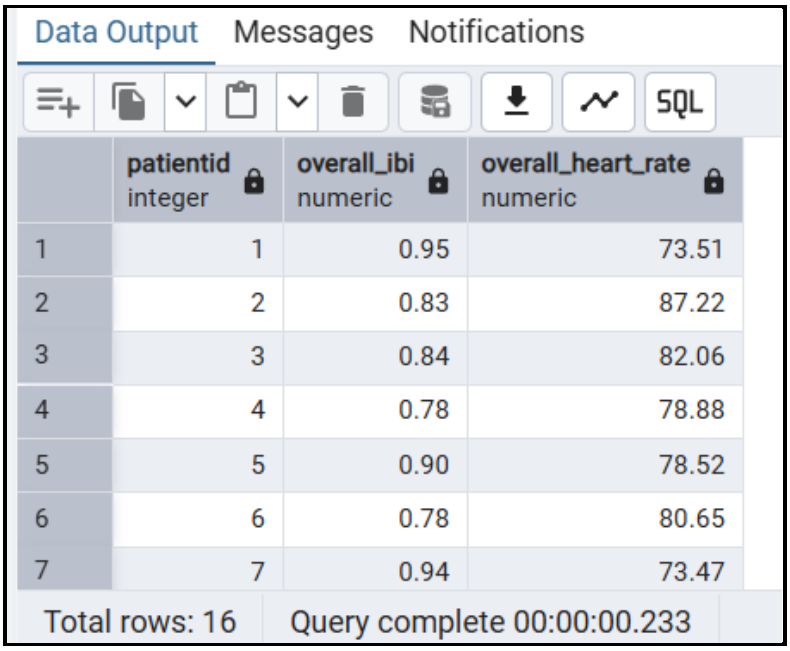
The above result gives the interbeat interval (IBI values). An interbeat interval (IBI) is the time duration between two successive heartbeats. Also known as the heart period or RR Interval. it is a fundamental measure of heart rate and is a primary input for heart rate variability (HRV) analysis. The higher the IBI,lower the heart rate. Higher IBI is actually good for the heart and the heart rate tends to remain low which is a very healthy sign for a patient.

19.How are IBI and HR inversely linked to each other?

QUERY:

```
/* Relationship between IBI and HR*/

select i.patientid,i.overall_ibi,h.overall_heart_rate
from gv_ibi_values i
INNER JOIN gv_heartrate h
ON i.patientid=h.patientid
```

OUTPUT:



Data Output | Messages | Notifications

	patientid integer	overall_ibi numeric	overall_heart_rate numeric
1	1	0.95	73.51
2	2	0.83	87.22
3	3	0.84	82.06
4	4	0.78	78.88
5	5	0.90	78.52
6	6	0.78	80.65
7	7	0.94	73.47

Total rows: 16 Query complete 00:00:00.233

Insights:

The above results give a strong inverse relationship between IBI and HR. Whenever the IBI increases, heart rate decreases, which is good for the heart.

20.Calculate the average EDA per day for each patient?

QUERY:

```
select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.eda),2) as avg_eda_day_wise
from gv_eda g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 14

	patientid integer	gender character varying (50)	day timestamp without time zone	avg_eda_day_wise numeric
1	1	FEMALE	2020-02-13 00:00:00	0.26
2	1	FEMALE	2020-02-14 00:00:00	6.89
3	1	FEMALE	2020-02-15 00:00:00	1.75
4	1	FEMALE	2020-02-16 00:00:00	0.27
5	1	FEMALE	2020-02-17 00:00:00	0.84
6	1	FEMALE	2020-02-18 00:00:00	0.82
7	1	FEMALE	2020-02-19 00:00:00	1.21
8	1	FEMALE	2020-02-20 00:00:00	3.46
9	1	FEMALE	2020-02-21 00:00:00	0.97

Total rows: 145 Query complete 00:00:00.097

Insights:

Electrodermal activity (EDA), also known as galvanic skin response (GSR), is a physiological measure that reflects changes in the electrical conductivity of the skin. Low EDA is actually for the body and it means the body is effectively managing stress where as high EDA means the person tends to have more stress. Lower the EDA, the better it is. Here we can see that patientid 1 has a very high EDA score ot 6.89. This means that on that particular day, the person must have been under severe stress.

21.Calculate the average temperature measured per day for each patient

QUERY:

```
select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.temperature),2) as avg_temperature_day_wise
from gv_temperature g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 14

	patientid integer	gender character varying (50)	day timestamp without time zone	avg_temperature_day_wise numeric
1	1	FEMALE	2020-02-13 00:00:00	32.37
2	1	FEMALE	2020-02-14 00:00:00	36.35
3	1	FEMALE	2020-02-15 00:00:00	35.67
4	1	FEMALE	2020-02-16 00:00:00	32.54
5	1	FEMALE	2020-02-17 00:00:00	36.05
6	1	FEMALE	2020-02-18 00:00:00	36.14
7	1	FEMALE	2020-02-19 00:00:00	36.31
8	1	FEMALE	2020-02-20 00:00:00	35.42
9	1	FEMALE	2020-02-21 00:00:00	35.71

Total rows: 145 Query complete 00:00:00.072

Insights:

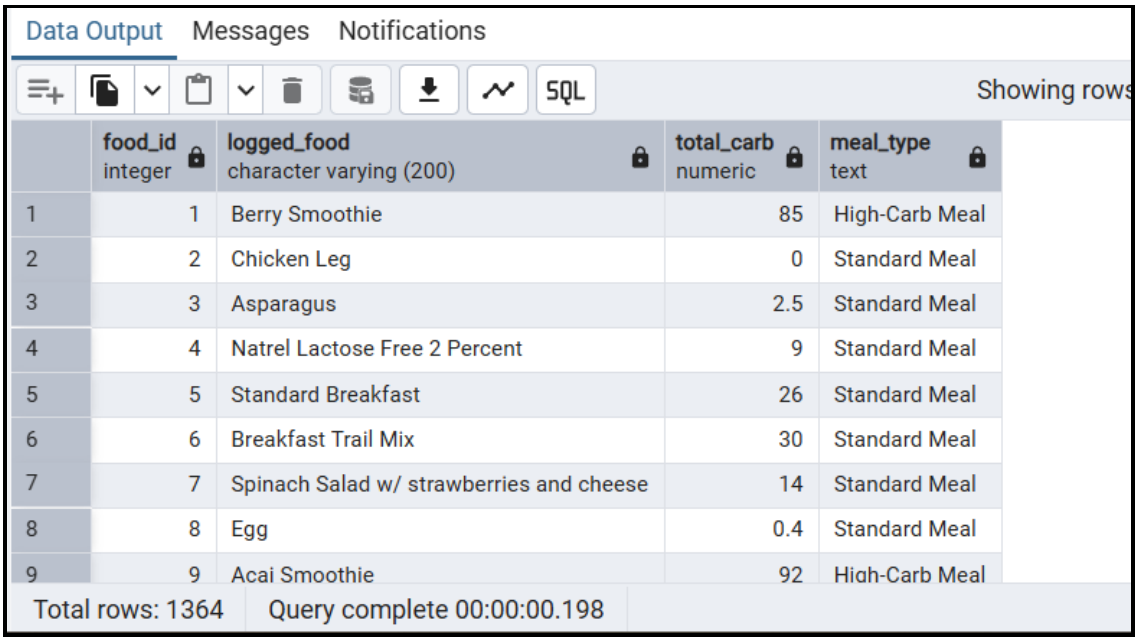
The above query gives the average body temperature measured each day. Usually the normal body temperature is 37.5 degrees Celsius. Any temperature less than this is hypothermia and any temperature higher than this causes fever.

22.Categorize the meal type based on their carbohydrate levels

QUERY:

```
/*21: Use a CASE statement on the foodlog table to create a new column, Meal_Type:
If total_carb >50, label as 'High-Carb Meal'. Otherwise, label as 'Standard Meal'.
List the patientid, total_carb, and the Meal_Type*/
SELECT
    food_id,
    logged_food,
    total_carb,
    CASE
        WHEN total_carb > 50 THEN 'High-Carb Meal'
        ELSE 'Standard Meal'
    END AS Meal_Type
FROM
    gv_food;
```

OUTPUT:



Data Output | Messages | Notifications

Showing rows

	food_id integer	logged_food character varying (200)	total_carb numeric	meal_type text
1	1	Berry Smoothie	85	High-Carb Meal
2	2	Chicken Leg	0	Standard Meal
3	3	Asparagus	2.5	Standard Meal
4	4	Natrel Lactose Free 2 Percent	9	Standard Meal
5	5	Standard Breakfast	26	Standard Meal
6	6	Breakfast Trail Mix	30	Standard Meal
7	7	Spinach Salad w/ strawberries and cheese	14	Standard Meal
8	8	Egg	0.4	Standard Meal
9	9	Acai Smoothie	92	High-Carb Meal

Total rows: 1364 | Query complete 00:00:00.198

Insights:

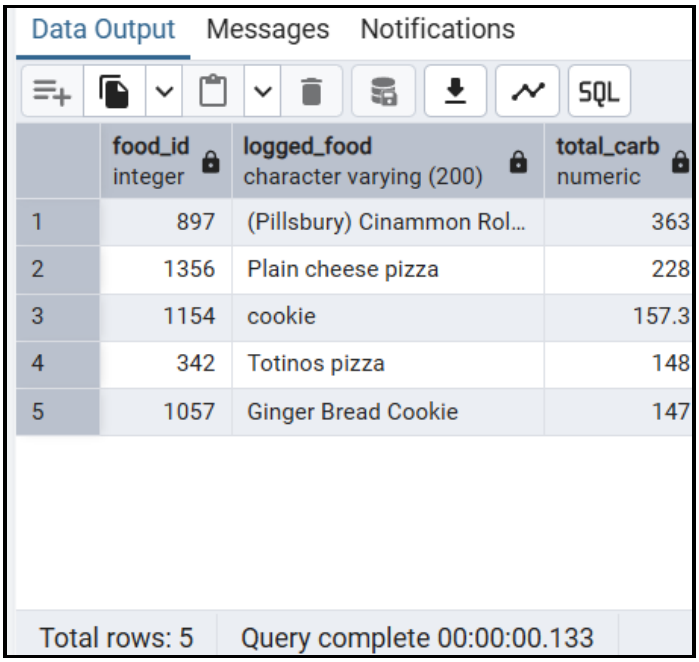
The above query lists the meals based on their carbohydrate content. High carb meals are a major cause for glucose spike in patients

23.List the top 5 high carbohydrate foods.

QUERY:

```
/*Q5: List the top 5 high carb foods*/

SELECT
    food_id,
    logged_food,
    total_carb
FROM
    gv_food
ORDER BY
    total_carb DESC
LIMIT 5;
```

OUTPUT:



Data Output | Messages | Notifications

	food_id integer	logged_food character varying (200)	total_carb numeric
1	897	(Pillsbury) Cinammon Rol...	363
2	1356	Plain cheese pizza	228
3	1154	cookie	157.3
4	342	Totinos pizza	148
5	1057	Ginger Bread Cookie	147

Total rows: 5 | Query complete 00:00:00.133

Insights:

The above output gives the top 5 foods that have the maximum carbohydrate content.

24.Display the maximum glucose level of each patient and the food they ate on that day and the food's calorie content and sugar content.

QUERY:

```
/* Display the maximum glucose level of each patient and also the food they ate
with its calorie content and sugar content*/

SELECT
    d.patientid,
    d.gender,
    MAX(x.glucosevaluemgdl) AS max_glucose_reading,
    f.logged_food,
    f.calorie,
    f.sugar,
    x.timeof
FROM
    gv_demography d
INNER JOIN
    gv_dexcom x ON d.patientid = x.patientid
INNER JOIN
    gv_food_log l ON l.timeof=x.timeof
INNER JOIN
    gv_food f ON f.food_id=l.food_id
GROUP BY
    d.patientid, d.gender,x.timeof,f.sugar,f.logged_food,f.calorie
order by
    max_glucose_reading desc
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 196 | Page No: 1 of 1

	patientid integer	gender character varying (50)	max_glucose_reading numeric	logged_food character varying (200)	calorie numeric	sugar numeric	timeof timestamp without time zone
1	10	FEMALE	213.00	Boneless Skinless Chicken Thigh	186	0	2020-03-27 12:04:00
2	12	MALE	168.00	Cheese and Crackers	179	2.2	2020-05-06 19:42:00
3	12	MALE	163.00	Dark Chocolate Chip	458	40	2020-05-06 19:22:00
4	10	FEMALE	155.00	Hard Boiled Egg	156	1.1	2020-03-25 06:54:00
5	12	MALE	150.00	Stevia	0	0	2020-05-04 06:32:00
6	12	MALE	150.00	Coffee	2.4	0	2020-05-04 06:32:00
7	12	MALE	150.00	Cream	57	1.1	2020-05-04 06:32:00
8	5	FEMALE	149.00	Milk	60	4	2020-03-01 08:45:00
9	5	FEMALE	149.00	Frosted Flakes	220	20	2020-03-01 08:45:00
10	5	FEMALE	144.00	Cheese	113	0.1	2020-02-28 22:00:00

Total rows: 196 | Query complete 00:00:00.138 | CRLF | Ln 235, Col 16

Insights:

The above query displays the maximum glucose level of a patient and on which day he had the maximum level and what food he ate. The calorie content and sugar content is also displayed. This helps to determine whether a person has high glucose level due to high calories or high sugar

Category 2-INTERMEDIATE LEVEL QUERIES

1. **Get the patients, number of days their ibi values were measured, along with their average ibi values**

QUERY:

```
/* 1.Get the patients, number of days their ibi values were measured, along with their
average ibi values*/

select patientid, count(distinct DATE_TRUNC('day', timeof)) as number_of_days_measured
round(avg(ibi),2) as avg_ibi,
max(ibi) as max_ibi,
min(ibi) as min_ibi
from gv_ibi
group by patientid
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows

	patientid integer	number_of_days_measured bigint	avg_ibi numeric	max_ibi numeric	min_ibi numeric
1	1	10	0.96	1.17	0.50
2	2	9	0.83	1.33	0.48
3	3	8	0.86	1.23	0.47
4	4	9	0.80	1.22	0.45
5	5	10	0.91	1.19	0.31
6	6	9	0.78	1.20	0.36
7	7	9	0.94	1.20	0.54

Total rows: 16 | Query complete 00:00:00.556

Insights:

The above result gives us the number of days each patient's IBI value was measured and their maximum IBI and the minimum IBI value. The higher the IBI, lower is the heart rate. This is actually very good for the body as the body is in a very calm and composed state and this prevents the onset of diabetes and even heart-related diseases.

2. Rank the average ibi values using rank()

QUERY:

```
/* 2.Rank the average ibi values using rank()*/

with ibi_summary as
(select patientid, count(distinct DATE_TRUNC('day', timeof))
as number_of_days_measured,
round(avg(ibi),2) as avg_ibi
from gv_ibi
group by patientid)
select a.patientid,a.number_of_days_measured,
a.avg_ibi,
rank() over (order by a.avg_ibi) as ibi_rank
from ibi_summary a
order by ibi_rank
```

OUTPUT:

Data Output | Messages | Notifications

	patientid integer	number_of_days_measured bigint	avg_ibi numeric	ibi_rank bigint
1	15	8	0.66	1
2	16	8	0.77	2
3	6	9	0.78	3
4	4	9	0.80	4
5	2	9	0.83	5
6	3	8	0.86	6
7	14	8	0.88	7

Total rows: 16 | Query complete 00:00:00.152

Insights:

This result ranks the average ibi values from least to the highest. Patient id 15 has the least IBI which is very worrying as he has a very high average heart rate value 90. Lower the IBI, higher the heart rate will be. Patient id15 has a very high chance of developing heart related diseases.

3. Calculate the moving average of ibi values?

QUERY:

```
/* 3.Calculate the moving average of ibi values*/

select patientid,timeof,ibi,
avg(ibi) over (partition by patientid order by timeof
rows between 2 preceding and current row) as moving_avg_ibi
from gv_ibi
order by patientid
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 1000

	patientid integer	timeof timestamp without time zone	ibi numeric	moving_avg_ibi numeric
1	1	2020-02-13 15:33:00	0.85	0.85000000000000000000
2	1	2020-02-13 15:34:00	0.94	0.89500000000000000000
3	1	2020-02-13 15:35:00	1.00	0.93000000000000000000
4	1	2020-02-13 15:36:00	0.71	0.88333333333333333333
5	1	2020-02-13 15:37:00	0.95	0.88666666666666666667
6	1	2020-02-13 15:39:00	1.04	0.90000000000000000000
7	1	2020-02-13 15:40:00	1.00	0.99666666666666666667

Total rows: 14813 Query complete 00:00:00.214

Insights:

By averaging a set of consecutive IBI values, the moving average highlights slower, more gradual changes in heart rhythm. These longer-term trends can be more indicative of the body's physiological state, such as its response to stress, exercise, or rest

4. Calculate the average, maximum,minimum eda values?

QUERY:

```sql
/* 4.Calculate the average, maximum,minimum eda values*/

select patientid, count(distinct DATE_TRUNC('day', timeof)) as number_of_days_measured
round(avg(eda),2) as avg_eda,
max(eda) as max_eda,
min(eda) as min_eda
from gv_eda
group by patientid
```

OUTPUT:

Data Output | Messages | Notifications

	patientid integer	number_of_days_measured bigint	avg_eda numeric	max_eda numeric	min_eda numeric
1	1	10	2.18	17.1400000	0.0400000
2	2	9	0.82	3.6700000	0.0600000
3	3	8	0.71	18.1900000	0.0000000
4	4	9	0.67	10.4600000	0.0400000
5	5	10	1.11	16.6400000	0.0400000
6	6	9	0.20	4.9500000	0.0200000
7	7	9	2.66	30.6900000	0.0000000

Total rows: 16 Query complete 00:00:00.159

Insights:

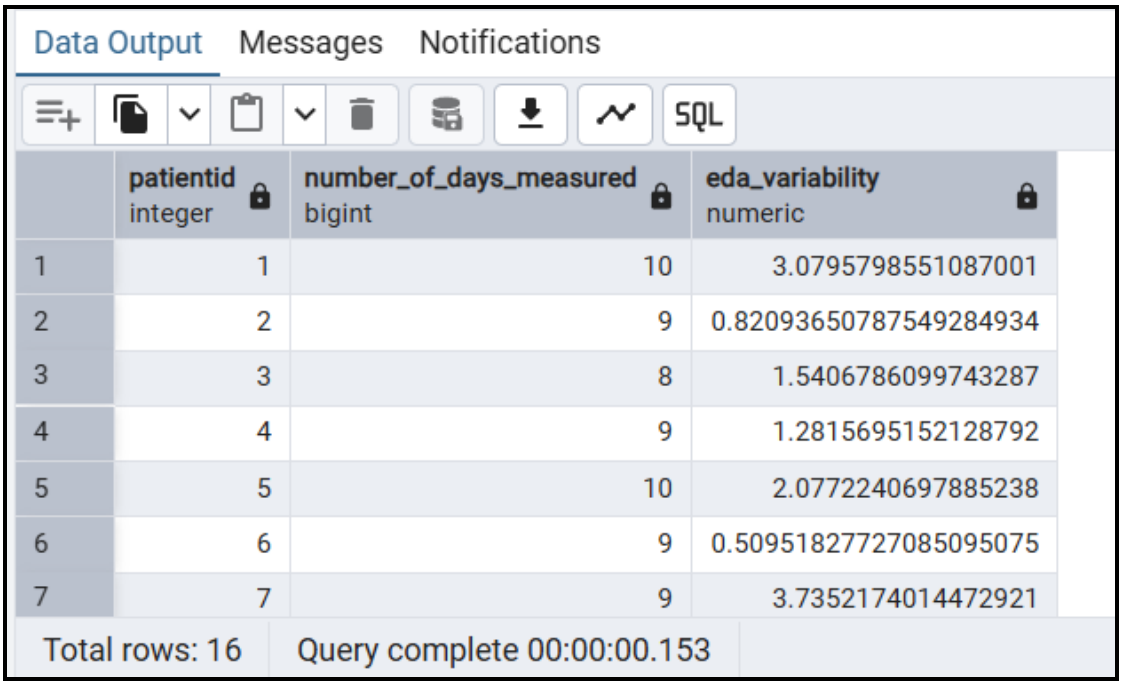
The above output gives the average EDA, maximum EDA, and minimum EDA of each patient. Higher the EDA, it means that the person is under severe stress and he needs to work on reducing his stress levels to maintain a healthy body.

5. How much does the EDA fluctuate?

QUERY:

```sql
/* 5.How much can EDA fluctuate*/

select patientid,
count(distinct DATE_TRUNC('day', timeof)) as number_of_days_measured,
stddev(eda) as eda_variability
from gv_eda
group by patientid
```

OUTPUT:



Data Output | Messages | Notifications

	patientid integer	number_of_days_measured bigint	eda_variability numeric
1	1	10	3.0795798551087001
2	2	9	0.82093650787549284934
3	3	8	1.5406786099743287
4	4	9	1.2815695152128792
5	5	10	2.0772240697885238
6	6	9	0.50951827727085095075
7	7	9	3.7352174014472921

Total rows: 16 | Query complete 00:00:00.153

Insights:

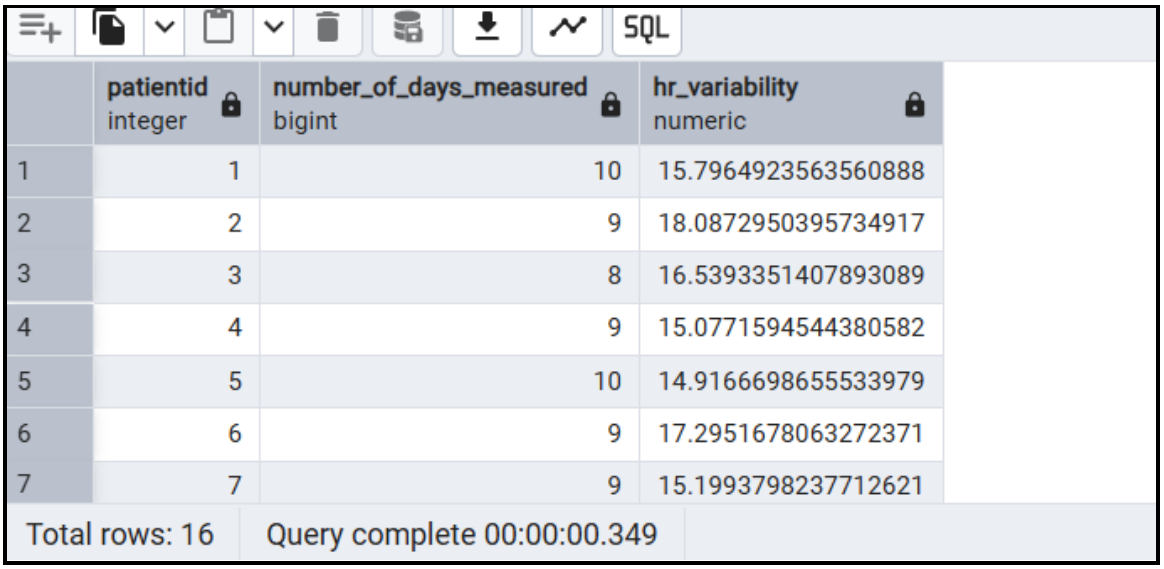
The above result gives EDA variability for each patient. If the EDA variability >3, it indicates strong swings in autonomic activity. The patient may have large shifts between stress and relaxation, which sometimes seen in people with high emotional reactivity, anxiety, or stress dysregulation. If the EDA <3, indicates blunted or reduced sympathetic responsiveness. This could occur in fatigue, depression, or autonomic hypoactivity (reduced physiological flexibility). It can also indicate stable calmness if the person is consistently relaxed. For example, here patient id 1 and 7 has a very high EDA variability which means they might be stressed or in some kind of trauma.

6. How much does the HR (Heart Rate)(HRV-Heart Rate Variability)fluctuate?

QUERY:

```
/*Heart Rate variability*/

select patientid,
count(distinct DATE_TRUNC('day', timeof)) as number_of_days_measured,
stddev(hr) as hr_variability
from gv_hr
group by patientid
```

OUTPUT:



	patientid integer	number_of_days_measured bigint	hr_variability numeric
1	1	10	15.7964923563560888
2	2	9	18.0872950395734917
3	3	8	16.5393351407893089
4	4	9	15.0771594544380582
5	5	10	14.9166698655533979
6	6	9	17.2951678063272371
7	7	9	15.1993798237712621

Total rows: 16 Query complete 00:00:00.349

Insights:

High HR (heart rates) are generally associated with low HR variability. So if a person has high HR Variability (17-18), it means that that person has good cardiovascular fitness, resilience to stress and better recovery. Moderate HRV (15-17) indicates normal and healthy body where the Body adapts to stress. Low HRV (<15) indicates chronic stress, fatigue, inflammation, poor sleep, depression or cardiac risk.

7. Rank the patients with high stress levels

QUERY:

```
/*6.Rank the patients with high stress levels*/

with eda_values as
(select patientid,
count(distinct DATE_TRUNC('day', timeof)) as number_of_days_measured,
stddev(eda) as eda_variability
from gv_eda
group by patientid)
select a.patientid,a.number_of_days_measured,a.eda_variability,
rank() over (order by a.eda_variability) as eda_variability_ranking
from eda_values a
order by eda_variability_ranking desc
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows

	patientid integer	number_of_days_measured bigint	eda_variability numeric	eda_variability_ranking bigint
1	15	8	4.2867057813800322	16
2	7	9	3.7352174014472921	15
3	1	10	3.0795798551087001	14
4	13	10	2.1997230868469491	13
5	11	10	2.1033276354356267	12
6	5	10	2.0772240697885238	11
7	10	9	1.6479009467827164	10

Total rows: 16 Query complete 00:00:00.154

Insights:

High EDA variability (>4) indicates that a patient could reflect anxiety, emotional instability, pain responses, or periods of high stress or agitation (common in sepsis, delirium, or ICU patients). Moderate EDA variability indicates that a person is healthy and is recovering from any health problems. Low EDA variability occurs in cases of fatigue, sedation, severe illness or reduced stress response.

8. **Calculate the daily EDA variability of patients.**

QUERY:

```
/*7.Calculate the daily EDA variability of patients*/

select patientid,
date_trunc('day',timeof) as day,
stddev(eda) as daily_eda_variability
from gv_eda
group by patientid,date_trunc('day',timeof)
order by patientid,day
```

OUTPUT:

Data Output | Messages | Notifications

Showing

	patientid integer	day timestamp without time zone	daily_eda_variability numeric
1	1	2020-02-13 00:00:00	0.26692084133163914977
2	1	2020-02-14 00:00:00	5.0408412656287945
3	1	2020-02-15 00:00:00	1.02709653047469947439
4	1	2020-02-16 00:00:00	0.21244957384888167520
5	1	2020-02-17 00:00:00	1.04591447061025633996
6	1	2020-02-18 00:00:00	1.2587548842182966
7	1	2020-02-19 00:00:00	1.7651858917669258

Total rows: 145 Query complete 00:00:00.139

Insights:

The above results show the daily EDA variability of patients. This helps to determine the everyday stress levels of patients and helps them to mitigate stress related illnesses. Moderate EDA variability indicates a healthy body.

9. Calculate the cumulative eda values for each patient day wise

QUERY:

```sql
/* 8.Calculate the cumulative eda values for each patient day wise*/

with eda_values as
(select patientid,
DATE_TRUNC('day', timeof) as DAY,
ROUND(avg(eda),2) as avg_eda
from gv_eda
group by patientid,TIMEOF
ORDER BY patientid),
overall_avg_eda_each_day as
(SELECT a.patientid,a.DAY,avg(a.avg_eda) as avg_eda
from eda_values a
group by a.DAY,a.patientid
order by a.patientid,a.DAY)
select b.patientid,
b.DAY,
b.avg_eda,
avg(b.avg_eda) over (partition by patientid order by b.DAY
rows between unbounded preceding and current row) as cumulative_avg_eda
from overall_avg_eda_each_day b
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 145

	patientid integer	day timestamp without time zone	avg_eda numeric	cumulative_avg_eda numeric
1	1	2020-02-13 00:00:00	0.26462809917355371901	0.26462809917355371901
2	1	2020-02-14 00:00:00	6.8875833333333333	3.57610571625344350951
3	1	2020-02-15 00:00:00	1.7476666666666667	2.96662603305785123967
4	1	2020-02-16 00:00:00	0.27218750000000000000	2.29301639979338842975
5	1	2020-02-17 00:00:00	0.84125000000000000000	2.00266311983471074380
6	1	2020-02-18 00:00:00	0.81508333333333333333	1.80473315541781450872
7	1	2020-02-19 00:00:00	1.2133333333333333	1.72024746654860290748
8	1	2020-02-20 00:00:00	3.4636666666666667	1.93817486656336088154
9	1	2020-02-21 00:00:00	0.96878787878787878788	1.83046520125497398225
10	1	2020-02-22 00:00:00	3.5530000000000000	2.00271868112947658402
11	2	2020-02-21 00:00:00	0.55404058677685950413	0.55404058677685950413

Total rows: 145 Query complete 00:00:00.137

Insights:

cumulative_avg_eda (overall trend)

This smooths the daily fluctuations and helps to see if the patient is:

Improving (Moderate to low EDA) → decreasing sympathetic activity → stabilizing or recovering.

Worsening (High EDA) → increasing sympathetic activation → possible ongoing stress, pain, infection, or worsening clinical condition.

For example if we take patient id 1, on 13th the EDA value is low. But next day, there is a sudden spike in EDA value which indicates that the patient underwent acute stress or strong sympathetic response (eg. Infection or pain). Then next day again it drops drastically which indicates recovery from previous stress episodes.

10. Display EDA variability with their gender

QUERY:

```
/*9.Display EDA variability with their gender*/

select d.gender,
d.patientid,
count(distinct DATE_TRUNC('day', e.timeof)) as number_of_days_measured,
round(avg(e.eda),2) as avg_eda,
stddev(e.eda) as eda_variability
from gv_eda e
INNER JOIN gv_demography d
ON e.patientid=d.patientid
group by d.patientid,d.gender
```

OUTPUT:

Data Output | Messages | Notifications

Showing rows: 1 to 16

	gender character varying (50)	patientid [PK] integer	number_of_days_measured bigint	avg_eda numeric	eda_variability numeric
1	FEMALE	1	10	2.18	3.0795798551087001
2	MALE	2	9	0.82	0.82093650787549284934
3	FEMALE	3	8	0.71	1.5406786099743287
4	FEMALE	4	9	0.67	1.2815695152128792
5	FEMALE	5	10	1.11	2.0772240697885238
6	FEMALE	6	9	0.20	0.50951827727085095075
7	FEMALE	7	9	2.66	3.7352174014472921
8	FEMALE	8	10	0.50	0.65578921590098286588

Total rows: 16 | Query complete 00:00:00.143

Insights:

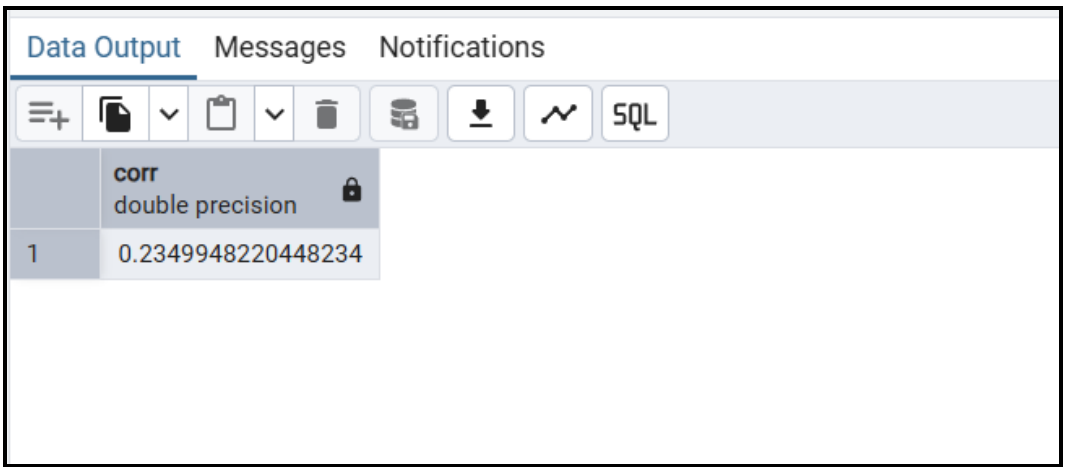
The above result shows the average EDA values with their gender. Here we can see that Female patients have high EDA values compared to male patients. This indicates that female patients have undergone lot of stress episodes compared to male patients.

11. Determine the Correlation between EDA and HR?

QUERY:

```
/* 10.Correlation between EDA and HR*/

with avg_eda_hr_values as
(select h.patientid,avg(h.hr) as avg_heart_rate,AVG(e.eda) as avg_eda
from gv_hr h
INNER JOIN gv_eda e
ON h.patientid=e.patientid
group by h.patientid)
select
corr(a.avg_eda,a.avg_heart_rate)
from avg_eda_hr_values a
```

OUTPUT:



Insights:

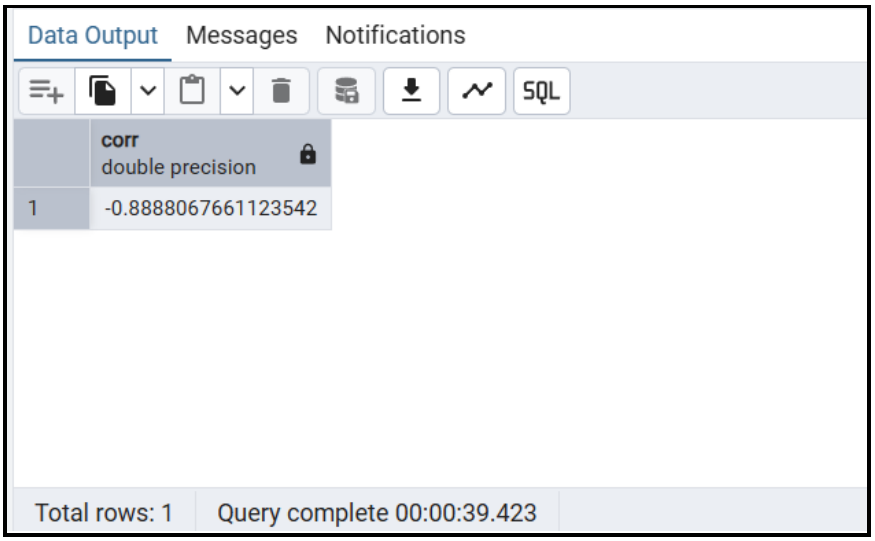
The above results show the correlation between EDA and Heart Rate values of patients. The result indicates that there is very minimal to no positive correlation between EDA and HR as the correlation value is 0.23 which is more or less close to zero.

12. Determine the Correlation between IBI and HR?

QUERY:

```sql
/* 10.Correlation between IBI and HR*/

with avg_ibi_hr_values as
(select h.patientid,avg(h.hr) as avg_heart_rate,AVG(e.ibi) as avg_ibi
from gv_hr h
INNER JOIN gv_ibi e
ON h.patientid=e.patientid
group by h.patientid)
select
corr(a.avg_ibi,a.avg_heart_rate)
from avg_ibi_hr_values a
```

OUTPUT:



Insights:

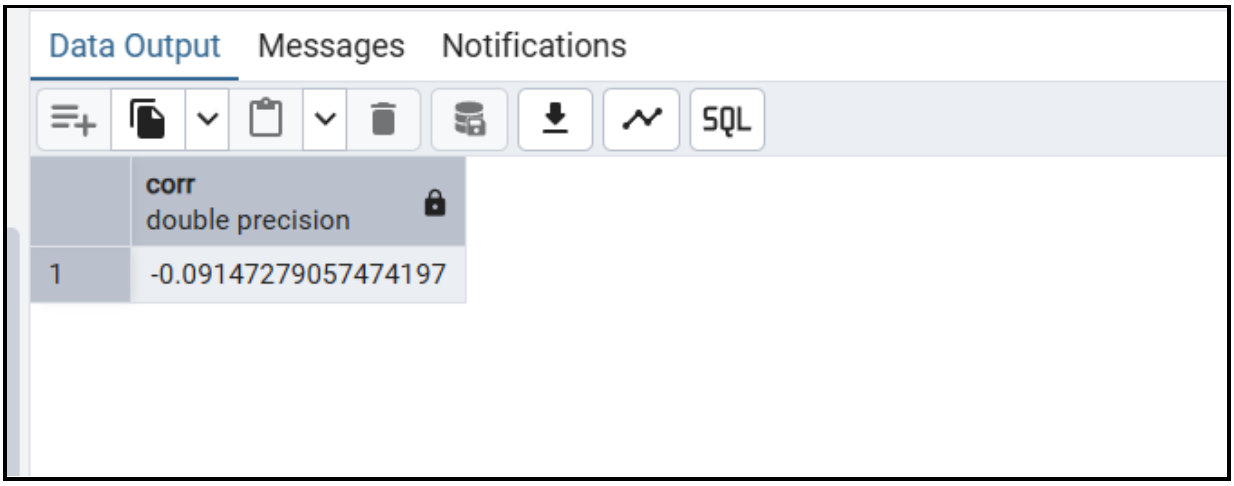
The above result indicates that there is a strong negative correlation between IBI and HR. As previously discussed, when the IBI increases, the HR decreases and also vice versa.

13. **Determine the Correlation between glucose and EDA?**

QUERY:

```sql
/* 11.To find out the correlation between glucose and EDA*/

select corr(e.eda,g.glucosevaluemgdl)
from gv_eda e
INNER JOIN
gv_dexcom g
On e.patientid=g.patientid
```

OUTPUT:



Insights:

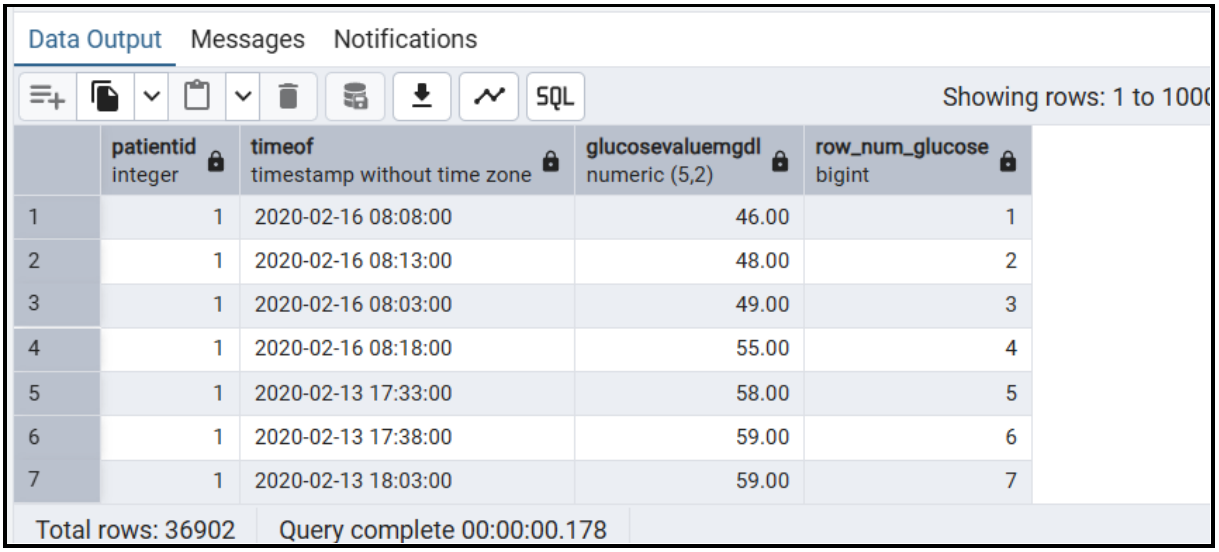
Here we observe that there is absolutely no correlation between EDA and glucose levels.

14. Add Row number based on the glucose levels

QUERY:

```sql
/* 12.Row number added based on the glucose levels*/

select patientid,
timeof,
glucosevaluemgdl,
row_number() over (partition by patientid order by glucosevaluemgdl)
as row_num_glucose
from gv_dexcom
```

OUTPUT:



Data Output | Messages | Notifications

Showing rows: 1 to 1000

	patientid integer	timeof timestamp without time zone	glucosevaluemgdl numeric (5,2)	row_num_glucose bigint
1	1	2020-02-16 08:08:00	46.00	1
2	1	2020-02-16 08:13:00	48.00	2
3	1	2020-02-16 08:03:00	49.00	3
4	1	2020-02-16 08:18:00	55.00	4
5	1	2020-02-13 17:33:00	58.00	5
6	1	2020-02-13 17:38:00	59.00	6
7	1	2020-02-13 18:03:00	59.00	7

Total rows: 36902 Query complete 00:00:00.178

Insights:

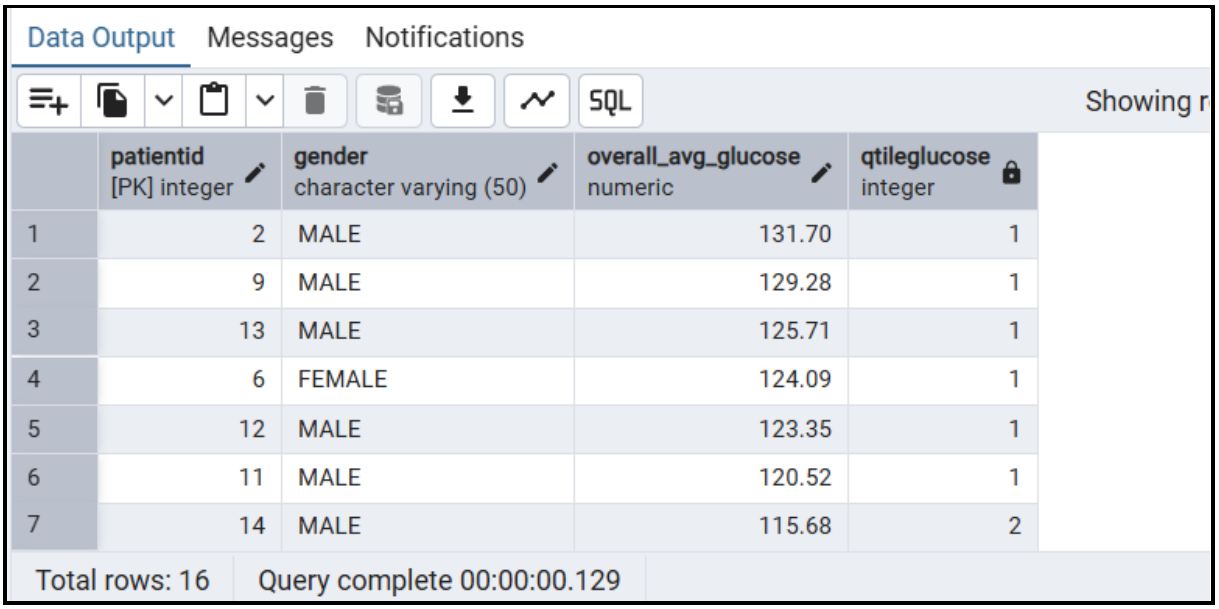
Here we use row_num() to add rows based on glucose levels lowest to highest for each patient for every reading so that we can easily identify on which days the glucose levels was low and gradually how it is increasing during the day.

15. Use NTILE() to determine the ranking of patients based on their glucose levels

QUERY:

```
/* 13.Using NTILE() to determine the ranking of patients based on their glucose levels

select patientid,gender,overall_avg_glucose,
ntile(3) over (order by overall_avg_glucose desc) as Qtileglucose
from gv_glucose
```

OUTPUT:



Data Output | Messages | Notifications

Showing r

	patientid [PK] integer	gender character varying (50)	overall_avg_glucose numeric	qtileglucose integer
1	2	MALE	131.70	1
2	9	MALE	129.28	1
3	13	MALE	125.71	1
4	6	FEMALE	124.09	1
5	12	MALE	123.35	1
6	11	MALE	120.52	1
7	14	MALE	115.68	2

Total rows: 16 Query complete 00:00:00.129

Insights:

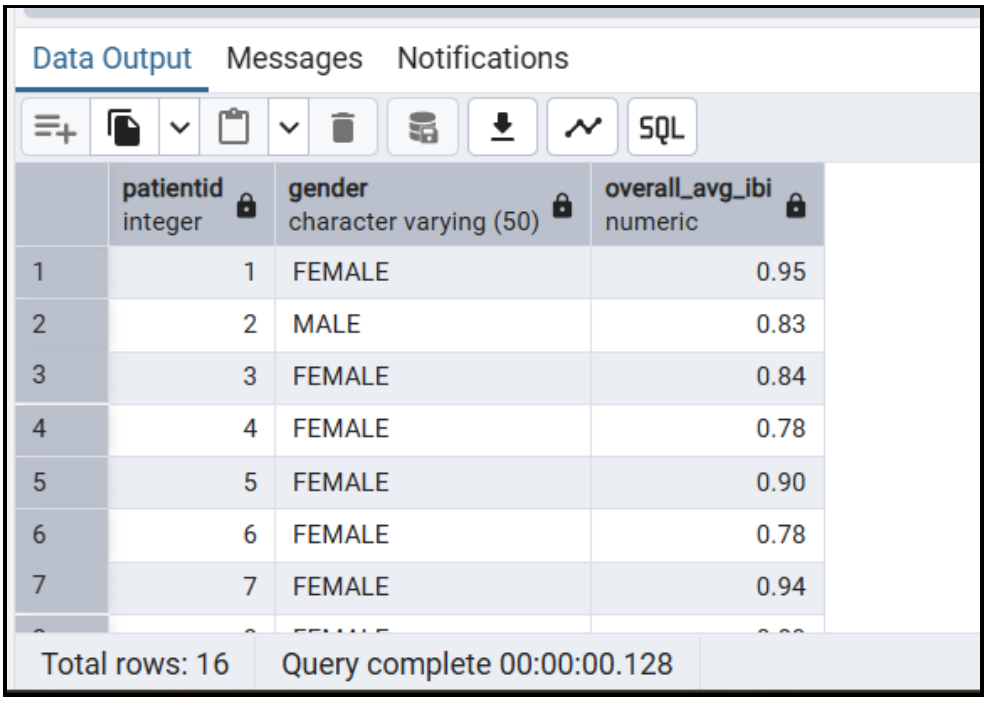
Here, the glucose values of each patient is categorized into three categories using Ntile(). It has categorized the glucose readings from 120-131 as 1, from 109-115 as 2, from 93-109 as 3.

16. Calculate overall avg ibi values across all days for all the patients?

QUERY:

```
/*15. Calculate overall avg ibi values for all the patients*/

with avg_ibi as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.ibi),2) as avg_ibi_day_wise
from gv_ibi g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid)
select a.patientid,a.gender,
round(avg(a.avg_ibi_day_wise),2) as overall_avg_ibi
from avg_ibi a
group by a.patientid,a.gender
order by a.patientid
```

OUTPUT:



Data Output | Messages | Notifications

	patientid integer	gender character varying (50)	overall_avg_ibi numeric
1	1	FEMALE	0.95
2	2	MALE	0.83
3	3	FEMALE	0.84
4	4	FEMALE	0.78
5	5	FEMALE	0.90
6	6	FEMALE	0.78
7	7	FEMALE	0.94

Total rows: 16 | Query complete 00:00:00.128

Insights:

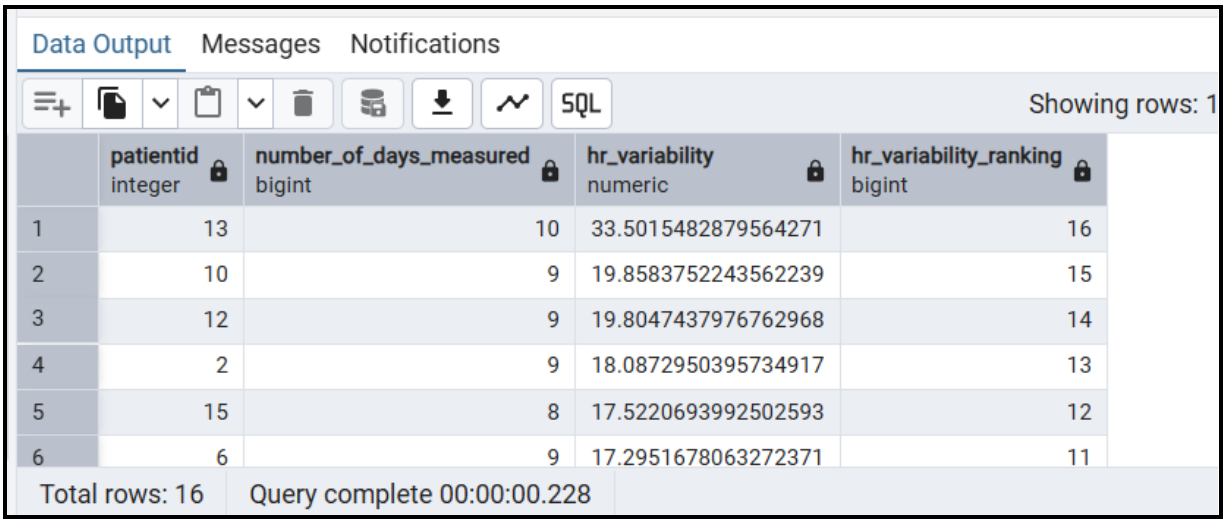
The above result displays the overall average IBI values of all patients.

17. Determine the ranking for the heart rate variability using rank()

QUERY:

```
/* 17.Determine the ranking for the heart rate variability using rank()*/

with hr_values as
(select patientid,
count(distinct DATE_TRUNC('day', timeof)) as number_of_days_measured,
stddev(hr) as hr_variability
from gv_hr
group by patientid)
select a.patientid,a.number_of_days_measured,a.hr_variability,
rank() over (order by a.hr_variability) as hr_variability_ranking
from hr_values a
order by hr_variability_ranking desc
```

OUTPUT:



Data Output | Messages | Notifications

Showing rows: 1

	patientid integer	number_of_days_measured bigint	hr_variability numeric	hr_variability_ranking bigint
1	13	10	33.5015482879564271	16
2	10	9	19.8583752243562239	15
3	12	9	19.8047437976762968	14
4	2	9	18.0872950395734917	13
5	15	8	17.5220693992502593	12
6	6	9	17.2951678063272371	11

Total rows: 16 | Query complete 00:00:00.228

Insights:

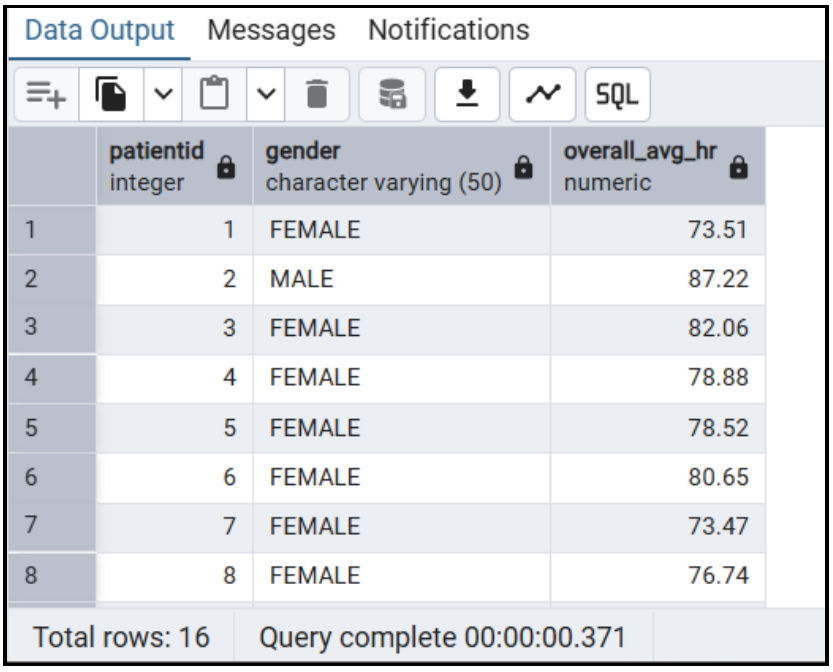
Patients with high HRV indicate healthy vagal tone, good adaptability and strong recovery. They usually recover fast from illnesses. But patients with low HRV have reduced vagal activity and also high stress. They might have critical illnesses or heart-related ailments. High HRV indicates low Heart Rate and low HRV indicates high heart rate. So here we use rank () to rank the patients based on their HRV from high to low. The highest rank is assigned for patients with high HRV and gradually the rank decreases with decreasing HRV values.

18. Calculate overall Average heart Rate for all the patients?

QUERY:

```
/*20. Calculate overall hr  for all the patients*/

with avg_hr as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.hr),2) as avg_hr_day_wise
from gv_hr g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid)
select a.patientid,a.gender,round(avg(a.avg_hr_day_wise),2) as overall_avg_hr
from avg_hr a
group by a.patientid,gender
order by a.patientid
```

OUTPUT



Data Output | Messages | Notifications

	patientid integer	gender character varying (50)	overall_avg_hr numeric
1	1	FEMALE	73.51
2	2	MALE	87.22
3	3	FEMALE	82.06
4	4	FEMALE	78.88
5	5	FEMALE	78.52
6	6	FEMALE	80.65
7	7	FEMALE	73.47
8	8	FEMALE	76.74

Total rows: 16 Query complete 00:00:00.371

Insights:

The above result gives the overall average heart rate for all the patients over all the days.

19. Calculate the overall average temperature of patients

QUERY:

```
/*21. Calculate the overall average temperature of patients*/

with avg_temperature as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.temperature),2) as avg_temperature_day_wise
from gv_temperature g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid)
select a.patientid,a.gender,
round(avg(a.avg_temperature_day_wise),2) as overall_average_temperature
from avg_temperature a
group by a.patientid,a.gender
order by a.patientid
```

OUTPUT

Data Output | Messages | Notifications

	patientid integer	gender character varying (50)	overall_average_temperature numeric
1	1	FEMALE	35.26
2	2	MALE	34.17
3	3	FEMALE	33.04
4	4	FEMALE	34.53
5	5	FEMALE	34.25
6	6	FEMALE	32.99
7	7	FEMALE	34.63
8	8	FEMALE	34.32

Total rows: 16 Query complete 00:00:00.107

Insights:

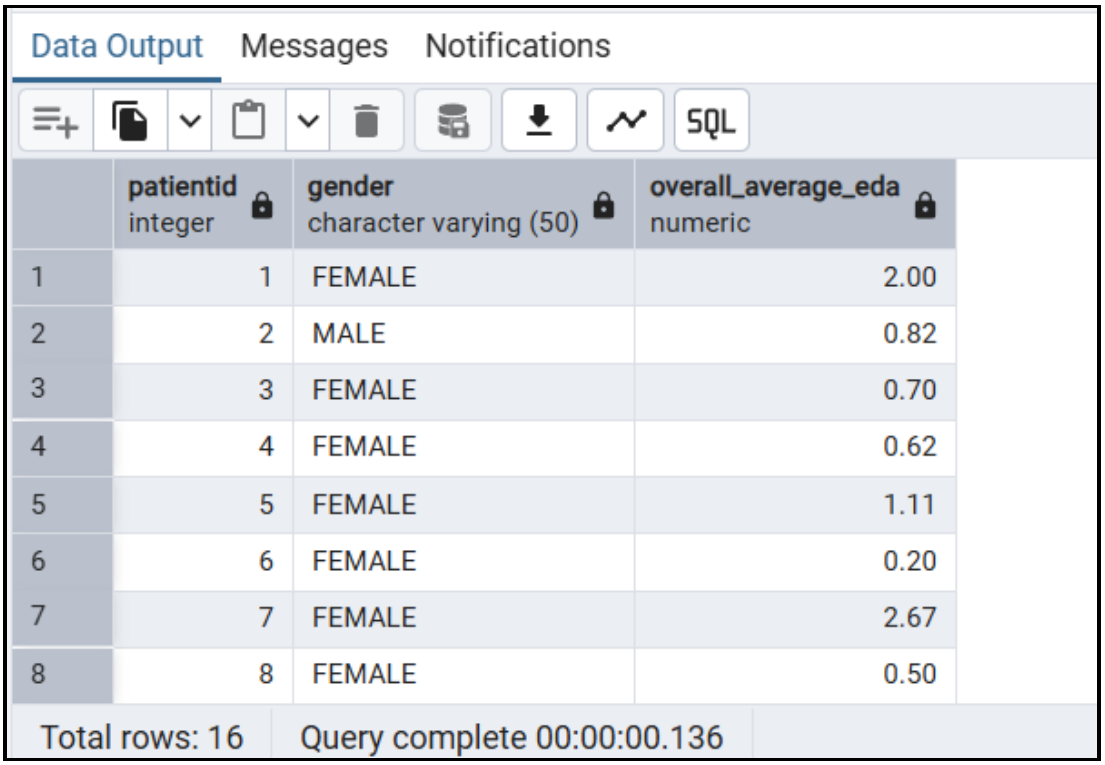
The above results give the overall average temperature of patients across all days.

20. Calculate the overall average EDA values of all patients?

QUERY:

```
/* 22.Calculate the overall average eda values of all patients*/

with avg_eda as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.eda),2) as avg_eda_day_wise
from gv_eda g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid)
select a.patientid,a.gender,
round(avg(a.avg_eda_day_wise),2) as overall_average_eda
from avg_eda a
group by a.patientid,a.gender
order by a.patientid
```

OUTPUT



Data Output | Messages | Notifications

	patientid integer	gender character varying (50)	overall_average_eda numeric
1	1	FEMALE	2.00
2	2	MALE	0.82
3	3	FEMALE	0.70
4	4	FEMALE	0.62
5	5	FEMALE	1.11
6	6	FEMALE	0.20
7	7	FEMALE	2.67
8	8	FEMALE	0.50

Total rows: 16 Query complete 00:00:00.136

Insights:

The above results display the overall average EDA values across all days

21. To find the average ibi per patient,avg ibi overall and their difference

QUERY:

```
/*23. To find the average ibi per patient,avg ibi overall and their difference*/

SELECT
    patientid,
    AVG(ibi) AS patient_average_ibi,
    AVG(ibi) OVER () AS overall_average_ibi,
    AVG(ibi) - AVG(ibi) OVER () AS difference_from_overall
FROM
    gv_ibi
GROUP BY
    patientid,ibi
```

OUTPUT

Data Output | Messages | Notifications

Showing rows: 1 to 959

	patientid integer	patient_average_ibi numeric	overall_average_ibi numeric	difference_from_overall numeric
1	12	0.52000000000000000000	0.82215849843587069864	-0.30215849843587069864
2	6	1.07000000000000000000	0.82215849843587069864	0.24784150156412930136
3	1	0.76000000000000000000	0.82215849843587069864	-0.06215849843587069864
4	14	0.85000000000000000000	0.82215849843587069864	0.02784150156412930136
5	6	0.40000000000000000000	0.82215849843587069864	-0.42215849843587069864
6	13	1.2100000000000000	0.82215849843587069864	0.38784150156412930136
7	10	0.52000000000000000000	0.82215849843587069864	-0.30215849843587069864

Total rows: 959 Query complete 00:00:00.237

Insights:

The above results show the individual patient's average IBI values, the overall average and the difference. If the individual average IBI is more than the overall average, then it is actually a very good sign of good heart health. As mentioned earlier, greater the IBI, lower the heart rate will be, which is actually very good.

22. Find the Total weekly calorie intake of patients

QUERY:

```sql
/* 24.Total weekly calorie intake of patients*/


    SELECT
        fl.patientid,
        EXTRACT(YEAR FROM fl.timeof) AS year,
        EXTRACT(WEEK FROM fl.timeof) AS week_num,
        SUM(f.calorie) AS total_weekly_calories
    FROM
        gv_food_log fl
        INNER JOIN gv_food f
        ON fl.food_id=f.food_id
    GROUP BY
        fl.patientid,
        EXTRACT(YEAR FROM fl.timeof),
        EXTRACT(WEEK FROM fl.timeof)
        ORDER BY fl.patientid
```

OUTPUT

Data Output | Messages | Notifications

	patientid integer	year numeric	week_num numeric	total_weekly_calories numeric
1	1	2020	8	7930.2
2	1	2020	7	5500
3	2	2020	8	6480.6
4	2	2020	9	10539.2
5	4	2020	10	6868.2
6	4	2020	9	5364.2
7	5	2020	10	9323.4
8	5	2020	9	5908.0
9	6	2020	10	14445

Total rows: 36 | Query complete 00:00:00.148

Insights:

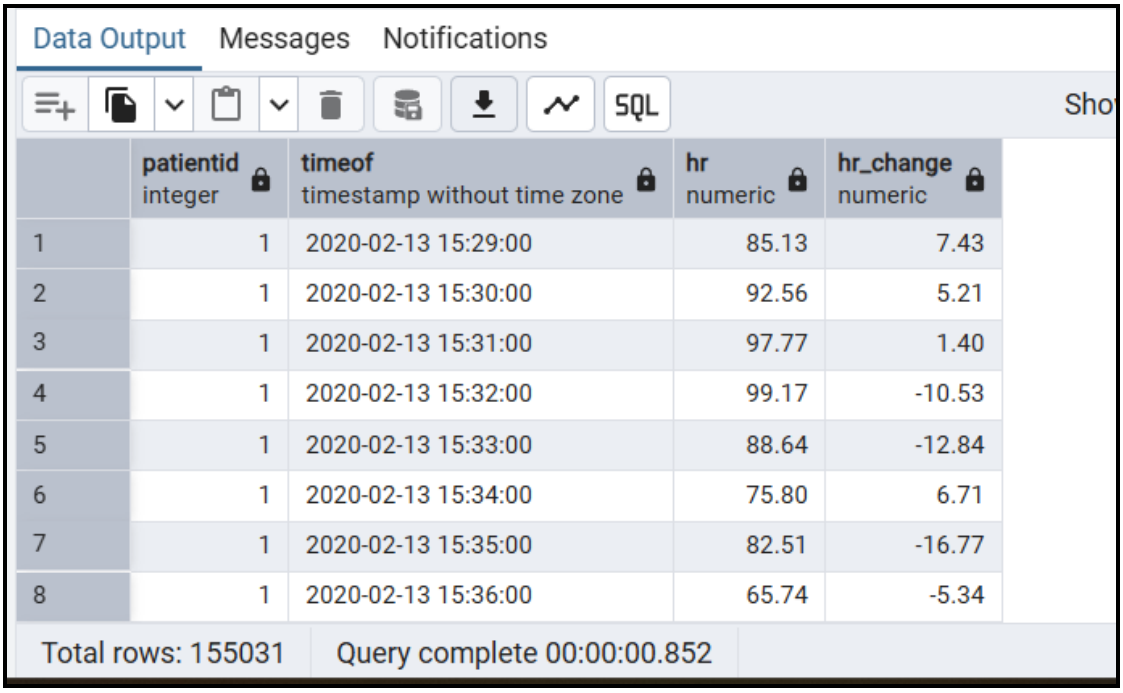
These results give the weekly calorie intake of each patient and on which week of the year.

23. Find the Heart Rate change using lead()

QUERY:

```
/* To find the Heart Rate change using lead()*/

SELECT
  patientid,
  timeof,
  hr,
  LEAD(hr) OVER (PARTITION BY patientid ORDER BY timeof) - hr AS hr_change
FROM gv_hr;
```

OUTPUT



Data Output | Messages | Notifications

SQL

	patientid integer	timeof timestamp without time zone	hr numeric	hr_change numeric
1	1	2020-02-13 15:29:00	85.13	7.43
2	1	2020-02-13 15:30:00	92.56	5.21
3	1	2020-02-13 15:31:00	97.77	1.40
4	1	2020-02-13 15:32:00	99.17	-10.53
5	1	2020-02-13 15:33:00	88.64	-12.84
6	1	2020-02-13 15:34:00	75.80	6.71
7	1	2020-02-13 15:35:00	82.51	-16.77
8	1	2020-02-13 15:36:00	65.74	-5.34

Total rows: 155031 Query complete 00:00:00.852

Insights:

These results show the change in the heart rate each day. It helps to determine how much the heart rate fluctuates every minute and if there is a drastic change, then it might be a worrying sign and attention is required.

24. Find the lowest heart rate and highest heart rate for each patient using first_value() and last_value()

QUERY:

```sql
/* To find the lowest heart rate and highest heart rate for each patient using
first_value() and last_value*/

with avg_hr_day as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.hr),2) as avg_hr_day_wise
from gv_hr g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid)
SELECT
a.patientid,
a.DAY,
a.avg_hr_day_wise,
first_VALUE(a.avg_hr_day_wise) OVER (PARTITION BY patientid ORDER BY a.avg_hr_day_wise
  AS lowest_heart_rate,
last_VALUE(a.avg_hr_day_wise) OVER (PARTITION BY patientid ORDER BY a.avg_hr_day_wise
between unbounded preceding and unbounded following)
  AS highest_heart_rate
 FROM avg_hr_day a;
```

OUTPUT

Data Output | Messages | Notifications

Showing rows: 1 to 145

	patientid integer	day timestamp without time zone	avg_hr_day_wise numeric	lowest_heart_rate numeric	highest_heart_rate numeric
1	1	2020-02-19 00:00:00	68.29	68.29	77.87
2	1	2020-02-22 00:00:00	69.62	68.29	77.87
3	1	2020-02-18 00:00:00	71.01	68.29	77.87
4	1	2020-02-15 00:00:00	72.55	68.29	77.87
5	1	2020-02-20 00:00:00	73.20	68.29	77.87
6	1	2020-02-14 00:00:00	74.46	68.29	77.87
7	1	2020-02-13 00:00:00	74.91	68.29	77.87
8	1	2020-02-16 00:00:00	76.09	68.29	77.87

Total rows: 145 Query complete 00:00:00.287

Insights:

These results capture the lowest heart rate measured and the highest heart rate measured on the first day and the last date.

25. Calculating the median value of glucose values for each patient using percentile_cont() and percentile_disc()?

QUERY:

```sql
/* Calculating the median value of glucose values for each patient using
percentile_cont() and percentile_disc()*/

with avg_glucose_per_day as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.glucosevaluemgdl),2) as avg_glucose_day_wise
from gv_dexcom g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid)
select a.patientid,
percentile_cont(0.5) within group (order by a.avg_glucose_day_wise)
as median_50_glucose,
percentile_disc(0.5) within group (order by a.avg_glucose_day_wise)
as median_50_glucose_disc
from avg_glucose_per_day a
group by a.patientid
order by a.patientid
```

OUTPUT

Data Output | Messages | Notifications

	patientid integer	median_50_glucose double precision	median_50_glucose_disc numeric
1	1	104.065	103.99
2	2	131.43	131.43
3	3	108.79	108.79
4	4	113.1	113.10
5	5	104.695	104.68
6	6	122.22	122.22
7	7	92.76	92.76
8	8	113.84	113.66
9	9	129.75	129.75

Total rows: 16 | Query complete 00:00:00.241

Insights:

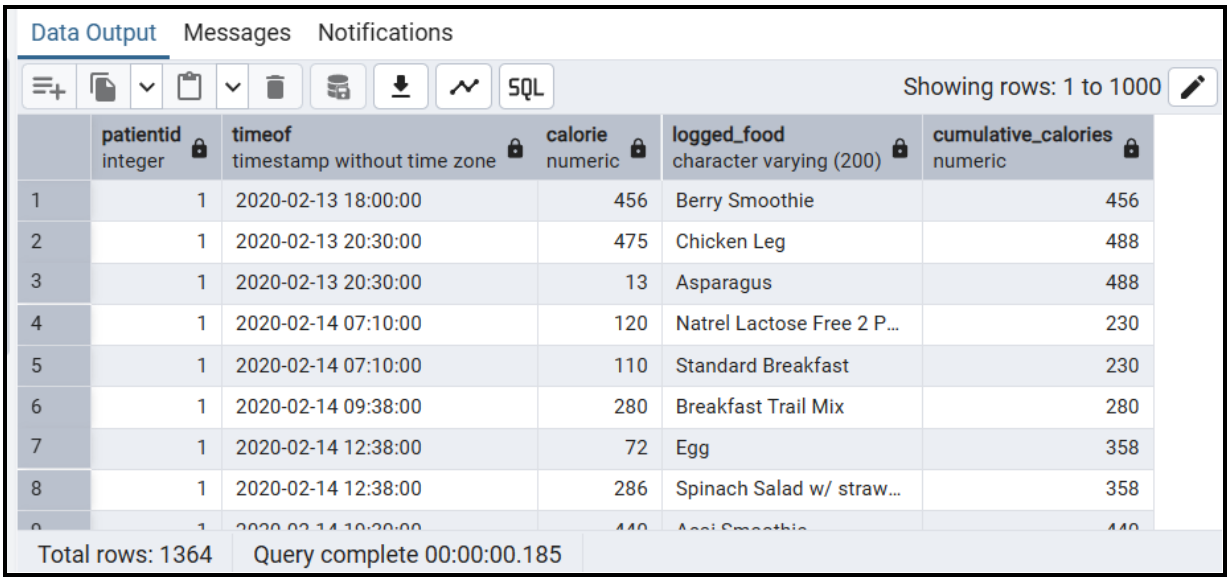
These results capture the median glucose values using percentile_cont() and percentile_disc() functions.

26. To find the cumulative calories based on food taken by patients partioned over timeof

QUERY:

```
/* To find the cumulative calories based on food taken by patients partioned over timeof*/

SELECT
g.patientid,
g.timeof,
f.calorie,
g.logged_food,
SUM(f.calorie) OVER (PARTITION BY g.timeof ORDER BY g.timeof) AS cumulative_calories
FROM
gv_food f
INNER JOIN gv_food_log g
ON f.food_id=g.food_id
ORDER BY
g.patientid, g.timeof
```

OUTPUT



Data Output | Messages | Notifications

Showing rows: 1 to 1000

	patientid integer	timeof timestamp without time zone	calorie numeric	logged_food character varying (200)	cumulative_calories numeric
1	1	2020-02-13 18:00:00	456	Berry Smoothie	456
2	1	2020-02-13 20:30:00	475	Chicken Leg	488
3	1	2020-02-13 20:30:00	13	Asparagus	488
4	1	2020-02-14 07:10:00	120	Natrel Lactose Free 2 P...	230
5	1	2020-02-14 07:10:00	110	Standard Breakfast	230
6	1	2020-02-14 09:38:00	280	Breakfast Trail Mix	280
7	1	2020-02-14 12:38:00	72	Egg	358
8	1	2020-02-14 12:38:00	286	Spinach Salad w/ straw...	358

Total rows: 1364 | Query complete 00:00:00.185

Insights:

These results display the cumulative calories taken by the patient on the same time of a day.

27. **Show Coefficient of variation for glucose values for each patient**

QUERY:

```sql
-- 2. Show Coefficient of variation for each patient.
select patientid, round(avg(glucosevaluemgdl),2) as
avg_gl,ROUND(stddev_pop(glucosevaluemgdl)/avg(glucosevaluemgdl),2)*100 as Coeff_Var
from gv_dexcom
group by patientid
order by Coeff_Var desc
```

OUTPUT

Data Output | Messages | Notifications

Showing rows: 1 to 16 | Page No: 1 of 1

	patientid integer	avg_gl numeric	coeff_var numeric
1	10	111.90	26.00
2	6	124.59	23.00
3	11	119.48	21.00
4	7	93.07	20.00
5	9	128.29	19.00
6	13	127.46	18.00
7	14	116.15	18.00
8	3	107.89	17.00
9	4	112.65	16.00

Total rows: 16 | Query complete 00:00:00.131

Insights:

Identify patients who need closer monitoring.

For example, patients with CV>25% when avg glucose = 250 mgdl are considered to be having

uncontrolled hyperglycemia compared to patients with CV>25% when avg glucose = 110 mgdl.

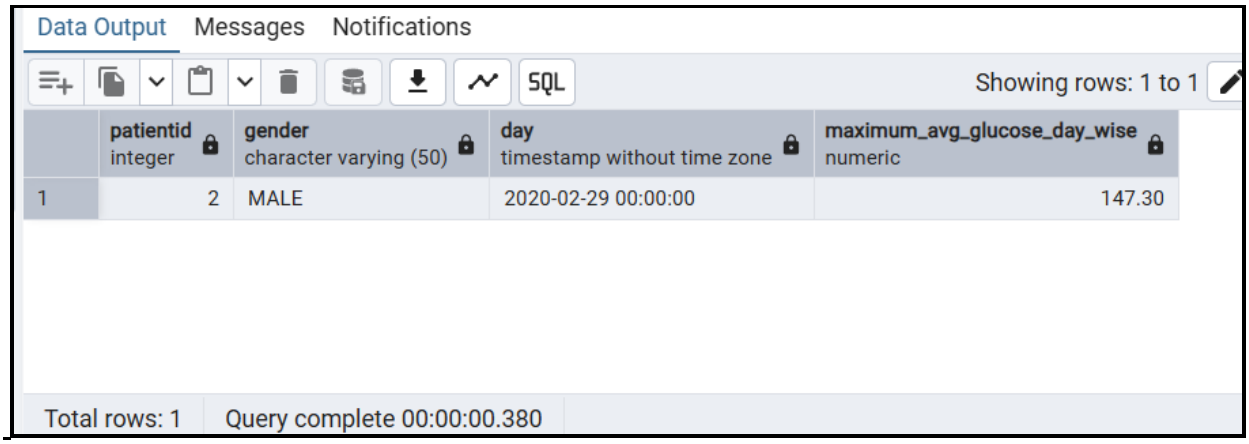
By combining this result with demographic or medication data (e.g., age, insulin use, diet), clinicians can explore why some patients have higher variability — such as diet inconsistencies, poor adherence, or comorbidities.

28. Which patient has the maximum glucose level compared to all other patients and on which day?

QUERY:

```sql
/*which patient has maximum glucose level compared to all the other patients and
on which day*/

with avg_glucose_levels as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.glucosevaluemgdl),2) as avg_glucose_day_wise
from gv_dexcom g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid)
select a.patientid,a.gender,a.DAY,a.avg_glucose_day_wise as maximum_avg_glucose_day_wise
from avg_glucose_levels a
where a.avg_glucose_day_wise=(select max(a.avg_glucose_day_wise)
from avg_glucose_levels a)
```

OUTPUT



Insights:

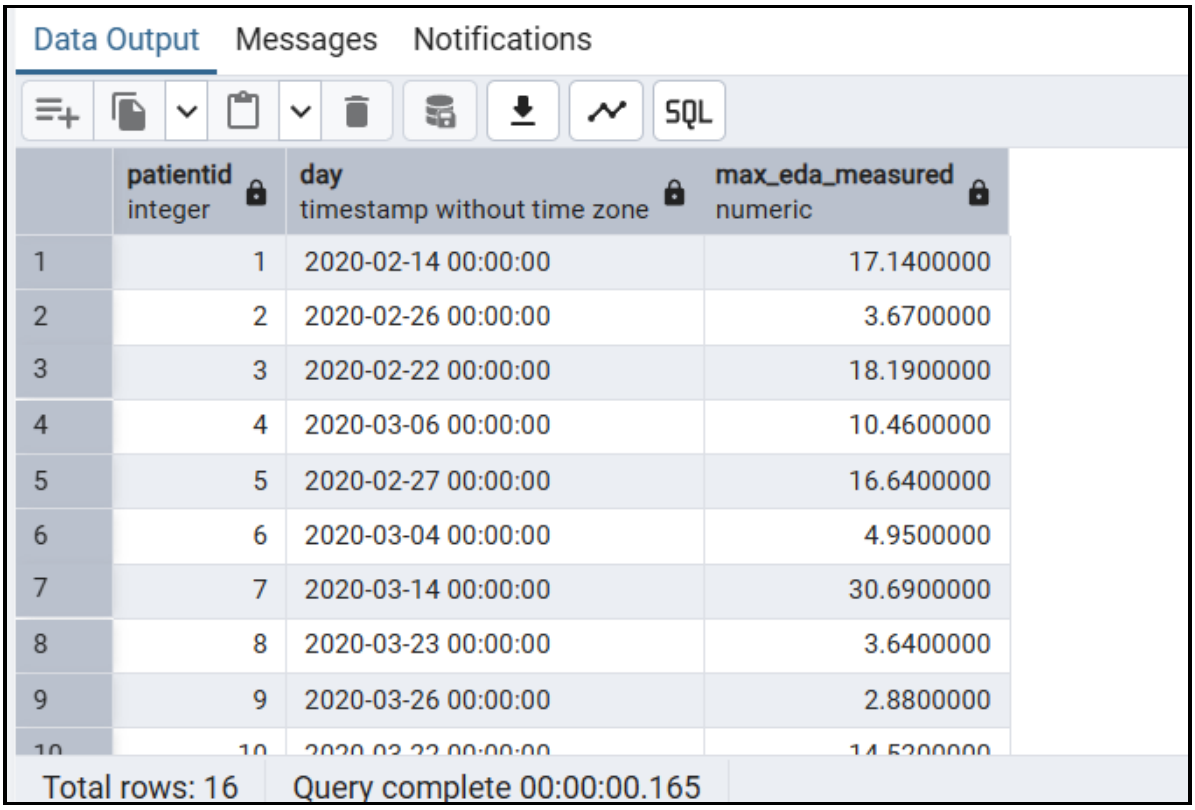
These results show which patient has the maximum glucose levels measured compared to all other patients and on which day, the maximum level was measured.

29. Calculate the maximum EDA measured for each patient and on which day based on their EDA ranking

QUERY:

```sql
/* Calculate the maximum EDA measured for each patient and on which
day based on their EDA ranking*/
WITH DailyEDA AS (
SELECT
patientid,
DATE_TRUNC('day', timeof) as day,
MAX(eda) as MAX_eda_measured
FROM gv_eda
GROUP BY patientid, day
),
RankedEDADays AS (
 SELECT
 patientid,
 day,
 MAX_eda_measured,
 RANK() OVER (
 PARTITION BY patientid
 ORDER BY MAX_eda_measured DESC
    ) AS rank_by_eda
    FROM Dailyeda
)

SELECT
patientid,
day,
MAX_eda_measured
FROM RankedEDADays
WHERE rank_by_eda = 1
ORDER BY patientid;
```

OUTPUT



Data Output | Messages | Notifications

	patientid integer	day timestamp without time zone	max_eda_measured numeric
1	1	2020-02-14 00:00:00	17.1400000
2	2	2020-02-26 00:00:00	3.6700000
3	3	2020-02-22 00:00:00	18.1900000
4	4	2020-03-06 00:00:00	10.4600000
5	5	2020-02-27 00:00:00	16.6400000
6	6	2020-03-04 00:00:00	4.9500000
7	7	2020-03-14 00:00:00	30.6900000
8	8	2020-03-23 00:00:00	3.6400000
9	9	2020-03-26 00:00:00	2.8800000
10	10	2020-03-22 00:00:00	14.5200000

Total rows: 16 Query complete 00:00:00.165

Insights:

The above results show the maximum EDA measured and on which day. This shows that the person on that day was having high stress levels compared to other days.

30. **Calculate the average Heart Rate (HR) for all patients, grouped by the hour of the day (0-23)**

QUERY:

```
SELECT
  EXTRACT(HOUR FROM timeof) AS hour_of_day,
  ROUND(AVG(hr), 2) AS average_hr
FROM gv_hr
GROUP BY hour_of_day
ORDER BY hour_of_day;
```

OUTPUT

Data Output | Messages | Notifications

	hour_of_day numeric	average_hr numeric
1	0	70.48
2	1	68.48
3	2	65.81
4	3	65.34
5	4	64.88
6	5	67.36
7	6	73.70
8	7	79.35
9	8	80.17
10	9	81.94
11	10	84.14

Total rows: 24 Query complete 00:00:00.347

Insights:

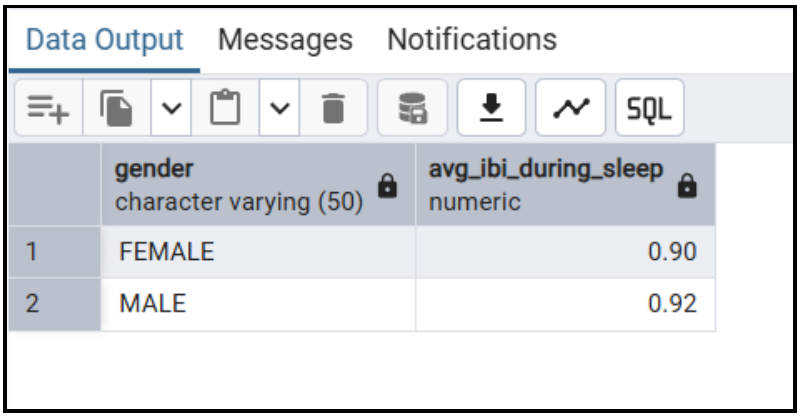
Identifies the circadian rhythm of the group's heart rate. High HR during late afternoon might indicate post-work stress or evening exercise. The lowest HR should typically be observed during deep sleep hours (early morning).

31. **Identify the average IBI for male patients versus female patients specifically during the assumed sleep window (10 PM to 6 AM).**

QUERY:

```sql
--Identify the average IBI for male patients versus female patients specifically
--during the assumed sleep window (10 PM to 6 AM).
SELECT
    d.gender,
    ROUND(AVG(i.ibi), 2) AS avg_ibi_during_sleep
FROM
    gv_demography d
JOIN
    gv_ibi i ON d.patientid = i.patientid
WHERE
    EXTRACT(HOUR FROM i.timeof) >= 22 OR EXTRACT(HOUR FROM i.timeof) <= 6 -- 10 PM to 6 AM
GROUP BY
    d.gender
ORDER BY
    d.gender;
```

OUTPUT



	gender character varying (50)	avg_ibi_during_sleep numeric
1	FEMALE	0.90
2	MALE	0.92

Insights:

Sleep is a crucial recovery period. This comparison checks for gender differences in autonomic nervous system recovery (measured via IBI/HRV) during key rest hours.

32. **For each patient and day, find the difference between the maximum and minimum recorded temperature (daily temperature fluctuation).**

QUERY:

```
--For each patient and day, find the difference between the maximum
--and minimum recorded temperature (daily temperature fluctuation).

SELECT
  patientid,
  DATE_TRUNC('day', timeof) AS observation_day,
  MAX(temperature) AS max_temp,
  MIN(temperature) AS min_temp,
  ROUND(MAX(temperature) - MIN(temperature), 2) AS daily_temp_fluctuation
FROM gv_temperature
GROUP BY patientid, observation_day
ORDER BY patientid, observation_day;
```

OUTPUT

Data Output | Messages | Notifications

Showing rows: 1 to 145

	patientid integer	observation_day timestamp without time zone	max_temp numeric	min_temp numeric	daily_temp_fluctuation numeric
1	1	2020-02-13 00:00:00	34.57	27.46	7.11
2	1	2020-02-14 00:00:00	36.52	35.76	0.76
3	1	2020-02-15 00:00:00	36.23	34.74	1.49
4	1	2020-02-16 00:00:00	35.49	28.14	7.35
5	1	2020-02-17 00:00:00	36.36	35.64	0.72
6	1	2020-02-18 00:00:00	36.47	35.49	0.98
7	1	2020-02-19 00:00:00	36.61	35.80	0.81

Total rows: 145 Query complete 00:00:00.141

Insights:

Daily temperature variability can be an indicator of metabolic state, illness, or fever. A large fluctuation might signal infection or, in the context of sleep studies, poor thermal regulation.

Category3-ADVANCED QUERIES

1. Write a procedure to get all the patients who have glucose levels>110

QUERY:

```
CREATE OR REPLACE PROCEDURE public.high_glucose_patients_proc(
    INOUT ref refcursor)
LANGUAGE 'plpgsql'
AS $$
begin
open ref for
select * from gv_glucose where overall_avg_glucose>110
order by patientid;
END;
$$;

begin;
call high_glucose_patients_proc('refcursor');
FETCH ALL from refcursor
```

OUTPUT

Data Output | Messages | Notifications

	patientid [PK] integer	gender character varying (50)	overall_avg_glucose numeric
1	2	MALE	131.70
2	4	FEMALE	112.29
3	6	FEMALE	124.09
4	8	FEMALE	111.02
5	9	MALE	129.28
6	10	FEMALE	111.91
7	11	MALE	120.52

Total rows: 10 | Query complete 00:00:00.462

Insights:

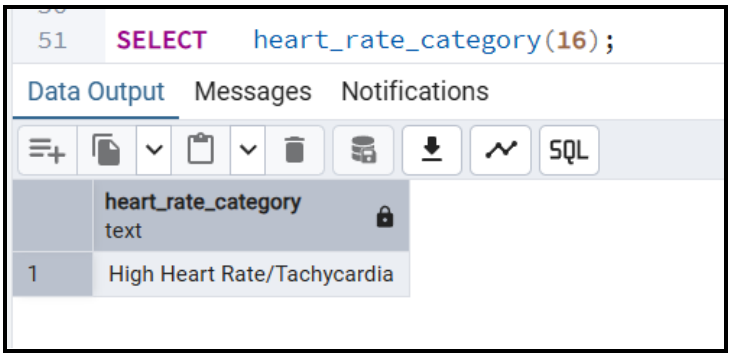
This procedure lists all the patients who have glucose levels>110.

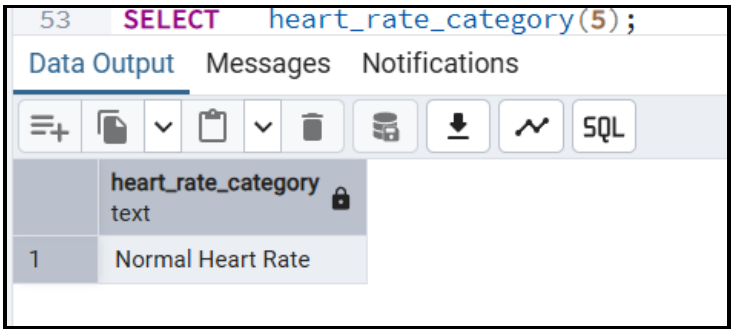
2. **Write a Function to categorize patients into three categories based on heartrate?**

QUERY:

```
/* Function to categorize patients into three categories based on heartrate*/

create or replace function  heart_rate_category(p_patientid INT)
returns text as
$$
declare
patient_heart_rate_level numeric;
heart_rate_category TEXT;
begin
select overall_heart_rate into patient_heart_rate_level
from gv_heartrate
where patientid=p_patientid;
IF patient_heart_rate_level IS NULL THEN
  return 'No heart_rate measured';
END IF;
CASE
   WHEN patient_heart_rate_level<70 then
   heart_rate_category :='Low Heart Rate/Bradycardia';
      WHEN patient_heart_rate_level>70 and patient_heart_rate_level<80  then
   heart_rate_category :='Normal Heart Rate';
   WHEN patient_heart_rate_level>80 THEN
    heart_rate_category :='High Heart Rate/Tachycardia';
   ELSE
     heart_rate_category:='Invalid';
 END CASE;
 return  heart_rate_category;
 END;
 $$
 LANGUAGE PLPGSQL;
```

OUTPUT





Insights:

This function categorizes all the patients based on their heart rate category. Normal heart rate would be 72 beats/min. Any HR value less than 70 is low and HR >80 is high Heart Rate.

3. Create a stored procedure to get all the critical parameters of patient with patientid as input

QUERY:

```
/* Create a stored procedure to get all the critical parameters of patient
with patientid as input*/

create or replace procedure get_patient_info_procedure(IN p_patientid INT,
out avg_glucose NUMERIC,
out avg_heart_rate numeric,
out avg_ibi numeric,
out avg_eda numeric,
out avg_temp numeric)
as
$$
BEGIN
select overall_avg_glucose,overall_average_heart_rate,
overall_average_eda,overall_average_ibi,overall_average_temperature
into avg_glucose,
avg_heart_rate,
avg_ibi,
avg_eda,
avg_temp
from critical_health_parameters where patientid=p_patientid;
IF NOT FOUND THEN
RAISE EXCEPTION 'The patientid with id % does not exist',p_patientid;
END IF;
END;
$$
LANGUAGE plpgsql;
```

OUTPUT

```
80
81  call get_patient_info_procedure(16,null,null,null,null,null)
```

Data Output | Messages | Notifications

Showing rows

	avg_glucose numeric	avg_heart_rate numeric	avg_ibi numeric	avg_eda numeric	avg_temp numeric
1	106.17	82.87	0.86	0.76	32.79

Insights:

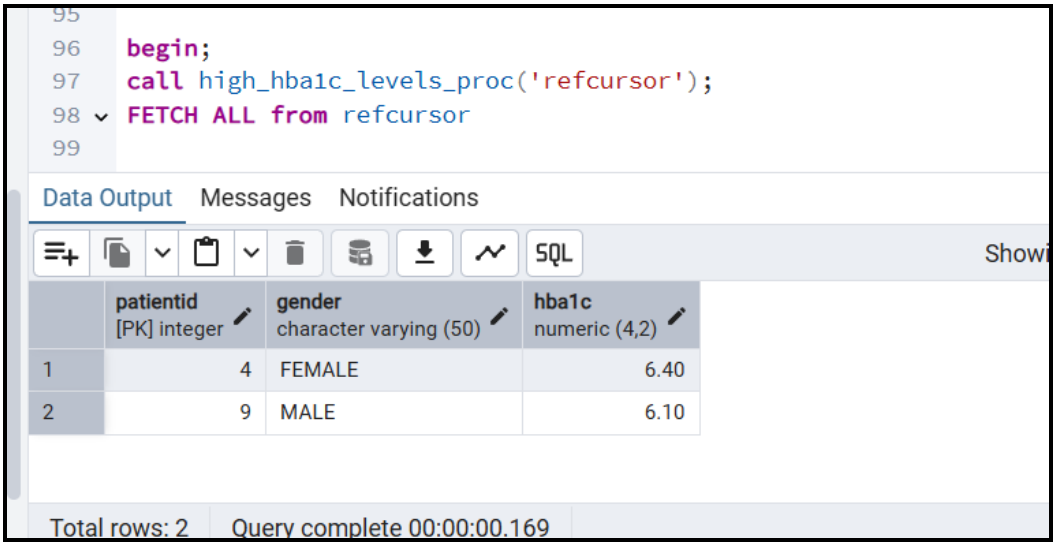
This procedure gives all the vital parameters of a patient at one go just by taking patientid as the input

4. Write a stored procedure to get all the patients who have hba1c>6

QUERY:

```
/* To get the patients who have hba1c>6*/

CREATE OR REPLACE PROCEDURE high_hba1c_levels_proc(
    INOUT ref refcursor)
LANGUAGE 'plpgsql'
AS $$
begin
open ref for
select * from gv_demography where hba1c>6;
END;
$$;

begin;
call high_hba1c_levels_proc('refcursor');
FETCH ALL from refcursor
```

OUTPUT



Insights:

This procedure lists the patients who have hba1c values >6. Hba1c is a critical parameter for glucose analysis. High hba1c (>6) is a major sign of diabetes.

5. Write a stored procedure to check if a patientid exists in the demography table.

QUERY:

```
CREATE OR REPLACE PROCEDURE check_patient_exists(
    IN p_patientid integer,
    OUT exists_flag boolean)
LANGUAGE 'plpgsql'
AS $$
BEGIN
select exists (select 1 from gv_demography where patientid=p_patientid)
into exists_flag;
RAISE NOTICE 'Exists %', exists_flag;
END
$$;
```

OUTPUT

```
118  DO
119  $$
120  DECLARE
121   flag BOOLEAN;
122  BEGIN
123  call check_patient_exists(16,flag);
124  END;
125  $$
```

Data Output | Messages | Notifications

```
NOTICE:  Exists t
DO

Query returned successfully in 139 msec.
```

```
DO
$$
DECLARE
 flag BOOLEAN;
BEGIN
call check_patient_exists(100,flag);
END;
$$
```

Data Output | Messages | Notifications

```
NOTICE:  Exists f
DO

Query returned successfully in 111 msec.
```

6. Write a function to get the abnormal vital parameters

QUERY:

```sql
create or replace function get_vital_parameters_abnormals(
min_glucose NUMERIC DEFAULT 110.0,
min_hr numeric DEFAULT 80.0
)
RETURNS TABLE (
patientid INT,
glucose numeric,
hr numeric
) AS $$
begin
RETURN QUERY
select
p.patientid,
p.overall_avg_glucose,
p.overall_average_heart_rate
from
critical_health_parameters p
where p.overall_avg_glucose>min_glucose
and p.overall_average_heart_rate>min_hr;
end;
$$
language plpgsql;
```

OUTPUT

```sql
197  select * from get_vital_parameters_abnormals();
```

Data Output | Messages | Notifications

	patientid integer	glucose numeric	hr numeric
1	2	131.70	87.22
2	6	124.09	80.65
3	13	125.71	87.03

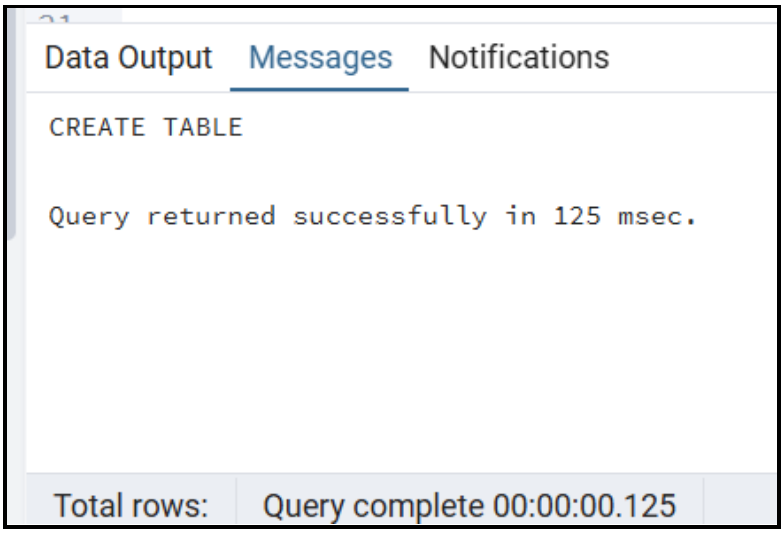
Insights:

This function retrieves all the patients who have abnormal health parameters (mainly high glucose and high HR)

7. **Create a trigger function to log entry of new patient into new_patient_entry table whenever a patient is inserted into the gv_demography table**

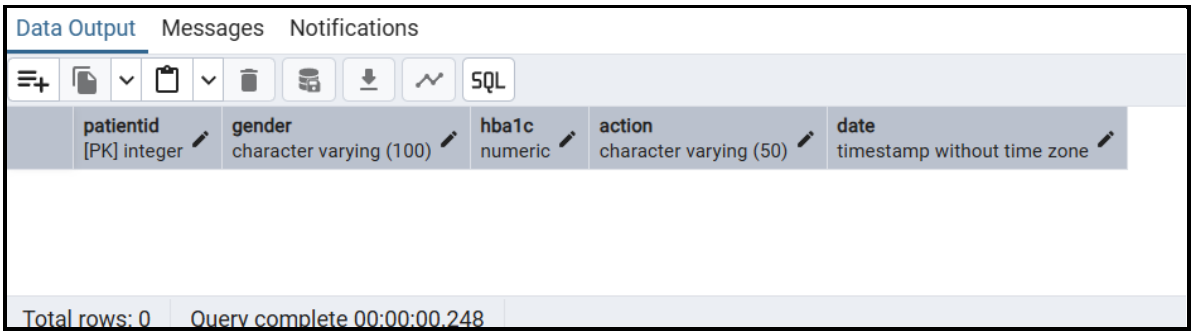
QUERY:

```
/* Create a trigger function to log entry of new patient into new_patient_entry table
whenever a patient is inserted into the gv_demography table*/

create table new_patient_entry
(patientid INT PRIMARY KEY,
gender VARCHAR(100),
hba1c numeric,
action varchar(50),
date TIMESTAMP default CURRENT_TIMESTAMP)
```

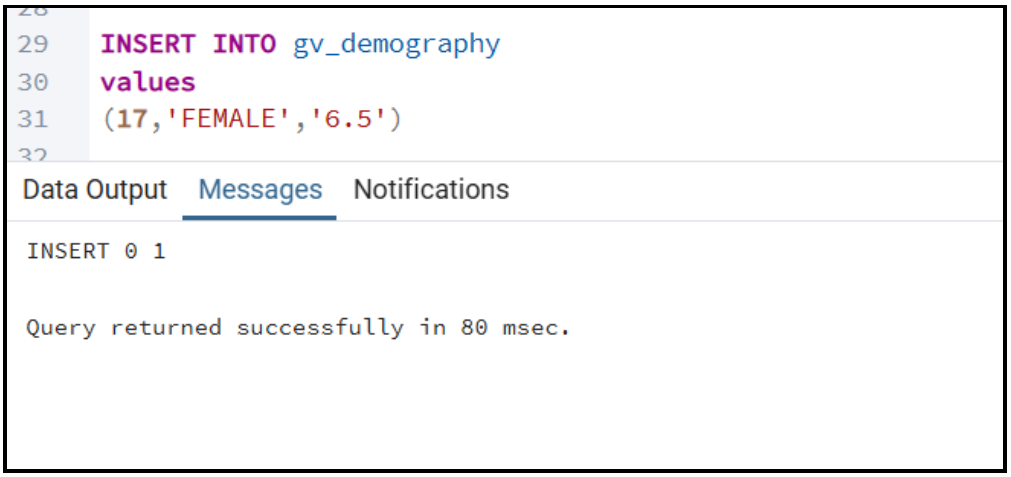



```
CREATE OR REPLACE function trg_after_insert_demography()
RETURNS trigger AS $$
BEGIN
INSERT INTO new_patient_entry (patientid, gender, hba1c, action, date)
VALUES(new.patientid, new.gender, new.hba1c, 'INSERT', DEFAULT);
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_insert_demography
AFTER INSERT ON gv_demography
FOR EACH ROW
EXECUTE FUNCTION trg_after_insert_demography();
```

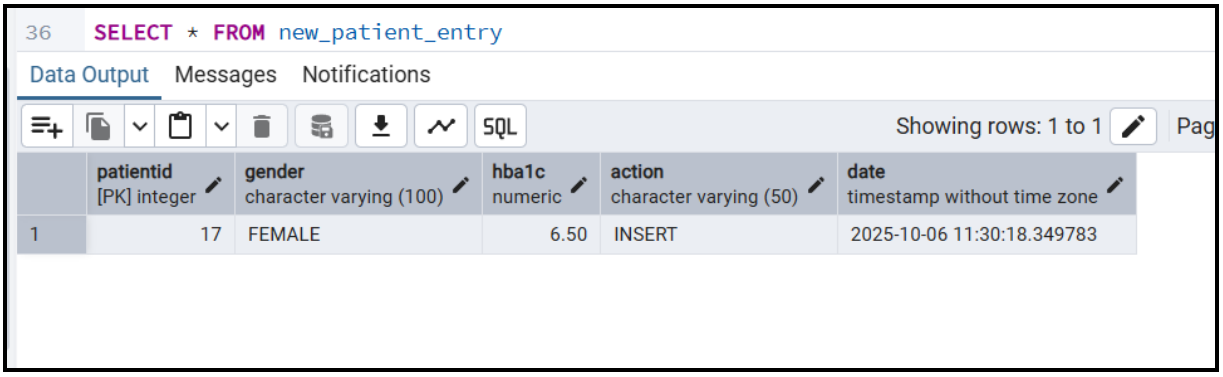
The table new_patient_entry is empty with no records now.



Now inserting a new record in the gv_demography table.



Now when we select from the new_patient_entry table, we can see the new inserted record



8. Write a trigger to log the old and new entry of hba1c patient into the hba1c change table

QUERY:

```
/* Write a trigger to log the old and new entry of hba1c patient into the hba1c change
table*/


create table hba1c_change_table
(patientid INT PRIMARY KEY,
OLD_hba1c numeric,
new_hba1c numeric)
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 70 msec.
```

```
create or replace function trg_after_update_hba1c()
returns trigger AS $$
BEGIN
IF OLD.hba1c<>new.hba1c THEN
insert into hba1c_change_table
values (NEW.patientid,OLD.hba1c,NEW.hba1c);
END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER after_update_hba1c
AFTER UPDATE on gv_demography
FOR EACH ROW
EXECUTE FUNCTION trg_after_update_hba1c();
```

Now we can see that the hba1c_change_table is empty with no records

```
68  select * from hba1c_change_table
69
```

Data Output | Messages | Notifications

	patientid [PK] integer	old_hba1c numeric	new_hba1c numeric

Now we update a record in the gv_demography table with the new hba1c value

```
65  update  gv_demography set  hba1c=6.2
66  where patientid=17
67
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 88 msec.
```

Now when we check the table hba1c_change_table:

Select * from hba1c_change_table

Data Output | Messages | Notifications

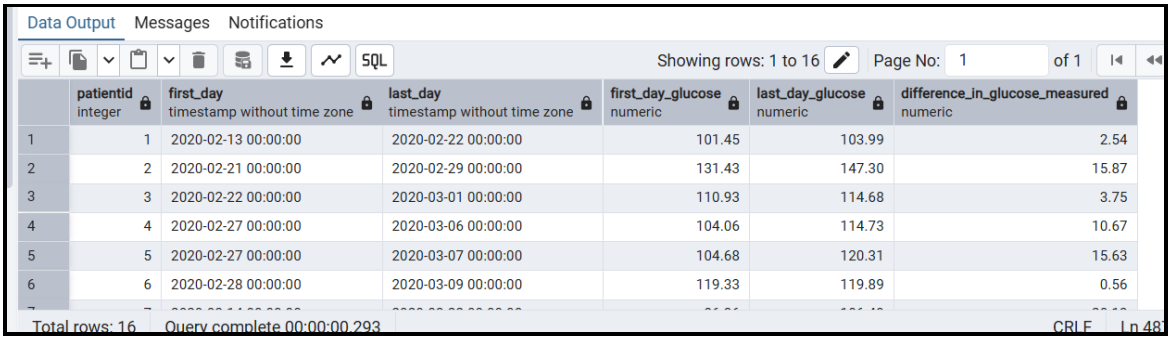
	patientid [PK] integer	old_hba1c numeric	new_hba1c numeric
1	17	6.50	6.20

9. Get the glucose value measured on first day and last day and taking their difference

QUERY:

```
/* To get the glucose value measured on first day and last day and taking their
difference*/

with avg_glucose_level as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.glucosevaluemgdl),2) as avg_glucose_day_wise
from gv_dexcom g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid),
first_day_last_day_glucose as
(select a.patientid,min(a.DAY) as first_day,max(a.DAY) as last_day,
min(a.avg_glucose_day_wise)
from avg_glucose_level a
group by a.patientid)
select a.patientid,b.first_day,b.last_day,
min(a.avg_glucose_day_wise) as first_day_glucose,
max(a.avg_glucose_day_wise) as last_day_glucose,
(max(a.avg_glucose_day_wise)-min(a.avg_glucose_day_wise)) as difference_in_glucose_measure
from first_day_last_day_glucose b
INNER JOIN avg_glucose_level a
ON a.patientid=b.patientid
where a.DAY in (b.first_day,b.last_day)
group by a.patientid,b.first_day,b.last_day
```

OUTPUT



Data Output | Messages | Notifications

Showing rows: 1 to 16 | Page No: 1 of 1

	patientid integer	first_day timestamp without time zone	last_day timestamp without time zone	first_day_glucose numeric	last_day_glucose numeric	difference_in_glucose_measured numeric
1	1	2020-02-13 00:00:00	2020-02-22 00:00:00	101.45	103.99	2.54
2	2	2020-02-21 00:00:00	2020-02-29 00:00:00	131.43	147.30	15.87
3	3	2020-02-22 00:00:00	2020-03-01 00:00:00	110.93	114.68	3.75
4	4	2020-02-27 00:00:00	2020-03-06 00:00:00	104.06	114.73	10.67
5	5	2020-02-27 00:00:00	2020-03-07 00:00:00	104.68	120.31	15.63
6	6	2020-02-28 00:00:00	2020-03-09 00:00:00	119.33	119.89	0.56

Total rows: 16 | Query complete 00:00:00.293 | CRLF | Ln 48

Insights:

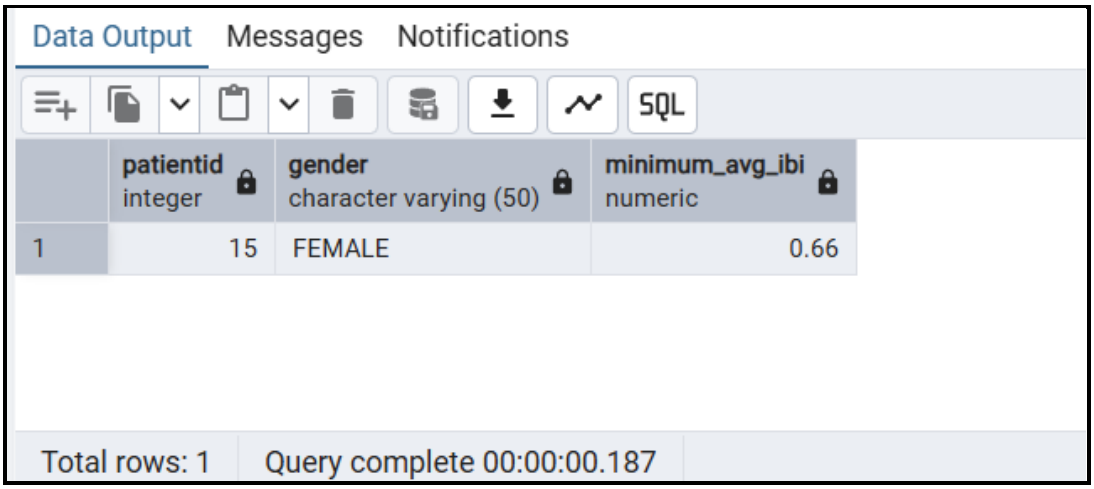
These results give the glucose measurement on the first day and the last day and gives the difference. This easily tells us how the patient has maintained their glucose levels over the days. If it has decreased, it is a good sign.

10. Find the patient who has minimum IBI across all the days and among all the patients

QUERY:

```
/* To find the patient who has minimum avg_ibi across all days*/

with avg_ibi_levels as
(select g.patientid ,d.gender ,
DATE_TRUNC('day',g.timeof) as DAY,
ROUND(avg(g.ibi),2) as avg_ibi_day_wise
from gv_ibi g
INNER JOIN gv_demography d
ON g.patientid=d.patientid
group by g.patientid,DAY,d.gender
ORDER BY g.patientid),

avg_ibi_levels_all_days as
(select a.patientid,a.gender,round(avg(a.avg_ibi_day_wise),2)as avg_ibi
from avg_ibi_levels a
group by a.patientid,a.gender
order by a.patientid)

select b.patientid,b.gender,b.avg_ibi as minimum_avg_ibi
from avg_ibi_levels_all_days b
where b.avg_ibi=(select min(b.avg_ibi)
from avg_ibi_levels_all_days b)
```

OUTPUT



Insights:

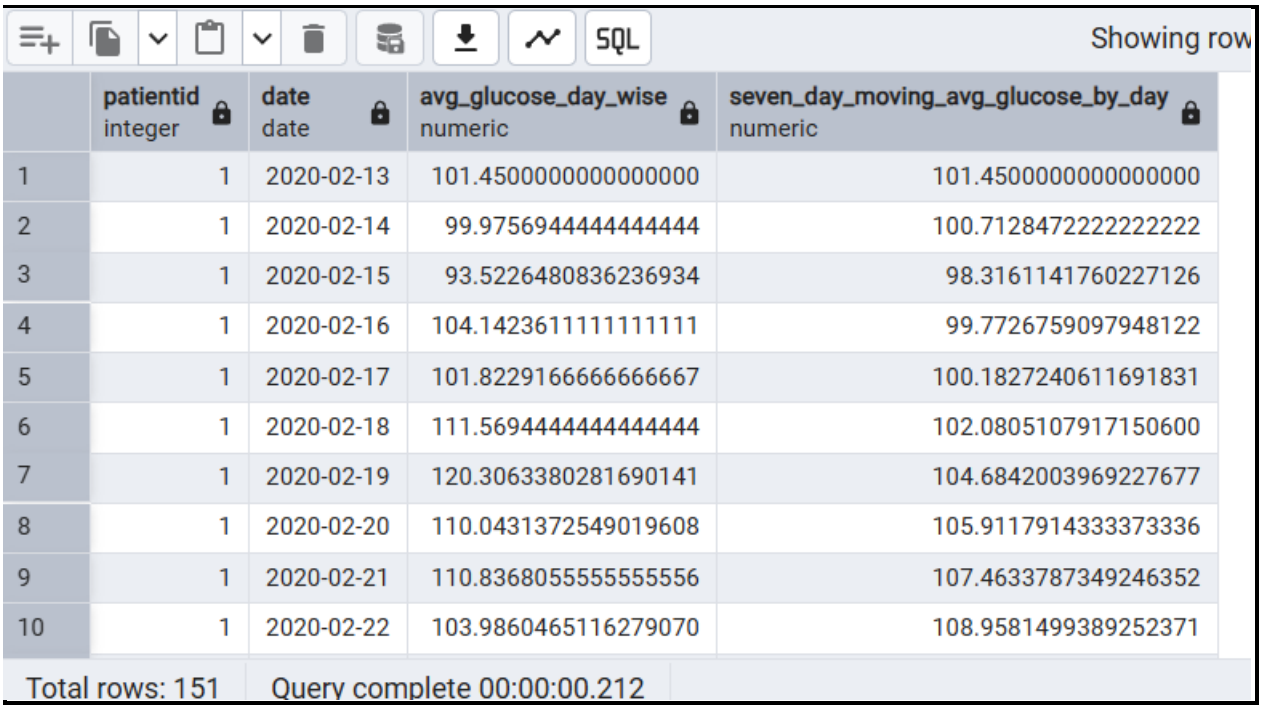
This patient has the minimum IBI value compared to all other patients. This means that she has a very high Heart rate which is an important sign for proper hear functioning.

11.Find the seven day moving average of glucose levels of all the patients

QUERY:

```
/* To find the seven day moving average of glucose levels*/
WITH dailyglucoselevel AS (
 SELECT
patientid,
CAST(timeof AS DATE) AS date,
AVG(glucosevaluemgdl) AS avg_glucose_day_wise
FROM
gv_dexcom
GROUP BY patientid,
CAST(timeof AS DATE)
)
SELECT
d.patientid,
d.date,
d.avg_glucose_day_wise,
AVG(d.avg_glucose_day_wise) OVER (
PARTITION BY d.patientid
ORDER BY d.date
RANGE BETWEEN INTERVAL '6 day' PRECEDING AND CURRENT ROW
    ) AS seven_day_moving_avg_glucose_by_day
FROM
dailyglucoselevel d
ORDER BY
d.patientid, d.date;
```

OUTPUT



Showing row

	patientid integer	date date	avg_glucose_day_wise numeric	seven_day_moving_avg_glucose_by_day numeric
1	1	2020-02-13	101.4500000000000000	101.4500000000000000
2	1	2020-02-14	99.9756944444444444	100.7128472222222222
3	1	2020-02-15	93.5226480836236934	98.3161141760227126
4	1	2020-02-16	104.1423611111111111	99.7726759097948122
5	1	2020-02-17	101.8229166666666667	100.1827240611691831
6	1	2020-02-18	111.5694444444444444	102.0805107917150600
7	1	2020-02-19	120.3063380281690141	104.6842003969227677
8	1	2020-02-20	110.0431372549019608	105.9117914333373336
9	1	2020-02-21	110.8368055555555556	107.4633787349246352
10	1	2020-02-22	103.9860465116279070	108.9581499389252371

Total rows: 151 Query complete 00:00:00.212

Insights:

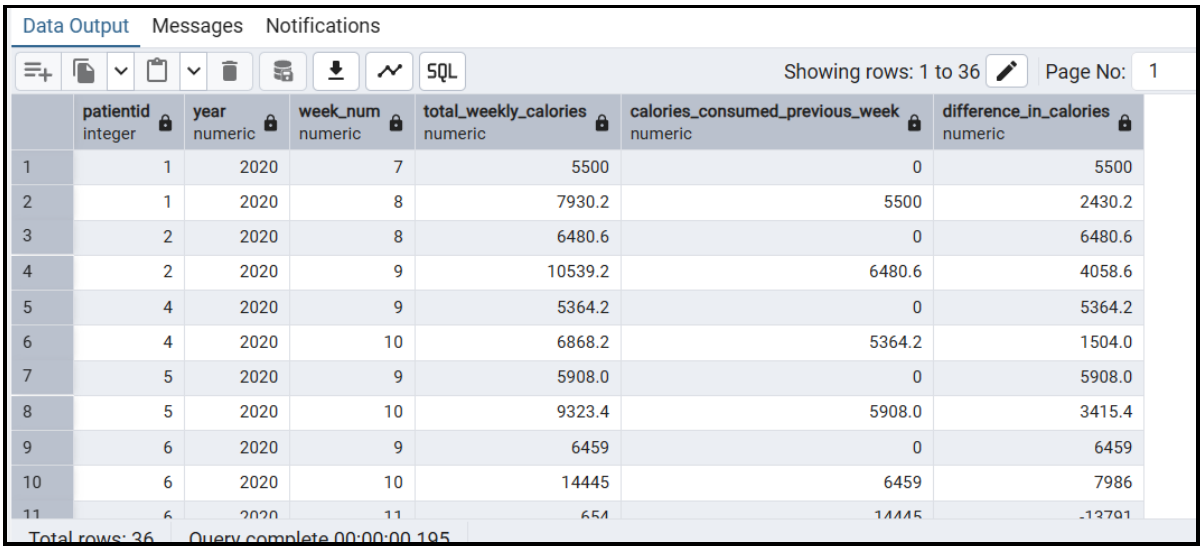
These results gives the 7-day moving average of glucose levels. This helps in easier glucose monitoring in diabetic patients.

12. Find the difference in calories intake by patients between current week and previous week

QUERY:

```sql
/* to find the difference in calories intake by
patients between current week and previous week*/

WITH calories_week_wise AS (
SELECT
fl.patientid,
 EXTRACT(YEAR FROM fl.timeof) AS year,
 EXTRACT(WEEK FROM fl.timeof) AS week_num,
 SUM(f.calorie) AS total_weekly_calories
 FROM
 gv_food_log fl
 INNER JOIN gv_food f
 ON fl.food_id=f.food_id
 GROUP BY
 fl.patientid,
 EXTRACT(YEAR FROM fl.timeof),
 EXTRACT(WEEK FROM fl.timeof)
 ORDER BY fl.patientid
)

select c.patientid,
c.year,
c.week_num,
c.total_weekly_calories,
lag(c.total_weekly_calories,1,0)OVER
(partition by c.patientid
order by c.year,c.week_num )AS calories_consumed_previous_week,
c.total_weekly_calories-lag(c.total_weekly_calories,1,0)
over (partition by c.patientid order by c.year,c.week_num) AS difference_in_calories
from
calories_week_wise c
order by c.patientid, c.year,c.week_num;
```

OUTPUT



Data Output | Messages | Notifications

Showing rows: 1 to 36 | Page No: 1

	patientid integer	year numeric	week_num numeric	total_weekly_calories numeric	calories_consumed_previous_week numeric	difference_in_calories numeric
1	1	2020	7	5500	0	5500
2	1	2020	8	7930.2	5500	2430.2
3	2	2020	8	6480.6	0	6480.6
4	2	2020	9	10539.2	6480.6	4058.6
5	4	2020	9	5364.2	0	5364.2
6	4	2020	10	6868.2	5364.2	1504.0
7	5	2020	9	5908.0	0	5908.0
8	5	2020	10	9323.4	5908.0	3415.4
9	6	2020	9	6459	0	6459
10	6	2020	10	14445	6459	7986
11	6	2020	11	654	14445	-13791

Total rows: 36 Query complete 00:00:00.195

Insights:

These results show the difference in calorie intake between current week and the previous week. This explains about the patients dietary habits across 2 weeks of data

.

13. Display the maximum calories intake by all the patients and on which day based on the calorie ranking

QUERY:

```sql
/* Maximum calorie intake by patient and on which day based on calorie ranking*/

WITH DailyCalorieIntake AS (
SELECT
g.patientid,
g.logged_food,
DATE_TRUNC('day', g.timeof) as day,
MAX(f.calorie) as MAX_calories
FROM gv_food_log g
INNER JOIN gv_food f ON g.food_id = f.food_id
GROUP BY g.patientid, day,g.logged_food
),
RankedCalorieDays AS (
SELECT
 patientid,
 logged_food,
 day,
MAX_calories,
RANK() OVER (
PARTITION BY patientid
ORDER BY MAX_calories DESC
) AS rank_by_calories
FROM DailyCalorieIntake

)
SELECT
patientid,
day,
logged_food,
MAX_calories
FROM RankedCalorieDays
WHERE rank_by_calories = 1
ORDER BY patientid;
```

OUTPUT

Data Output | Messages | Notifications

Showing rows: 1 to 15

	patientid integer	day timestamp without time zone	logged_food character varying (200)	max_calories numeric
1	1	2020-02-18 00:00:00	Ranch Wings	776
2	2	2020-02-21 00:00:00	(Red Baron) Brick Oven Pepperoni Pizza	1360
3	4	2020-03-02 00:00:00	(Chipotle) Bowl	1008
4	5	2020-03-04 00:00:00	(Outback Steakhouse) Chicken Tacos	1210
5	6	2020-03-07 00:00:00	Totinos pizza	1320
6	7	2019-10-24 00:00:00	Boost	654
7	8	2020-03-16 00:00:00	Pizza Triangles	1138
8	9	2020-03-28 00:00:00	Salmon with Butter and Asparagus	660
9	10	2020-03-23 00:00:00	Cheddar Cheese	1145
10	11	2020-04-14 00:00:00	(Pillsbury) Cinammon Rolls	2245
11	12	2020-05-10 00:00:00	Ginger Bread Cookie	806
12	13	2020-01-05 00:00:00	cookie	1181.3

Total rows: 15 Query complete 00:00:00.146

Insights:

The above results display the maximum calories consumed by the patient on which day and what food he had on that day. This helps the patient to streamline their dietary choices for beter glucose control.

14. On which day, maximum glucose levels were measured based on the glucose ranking.

QUERY:

```sql
/* Maximum glucose levels measured and on which day based on glucose level ranking*/

WITH Dailyglucose AS (
SELECT
patientid,
DATE_TRUNC('day', timeof) as day,
MAX(GLUCOSEVALUEMGDL) as MAX_glucose_measured
FROM gv_dexcom
GROUP BY patientid, day
),
RankedglucoseDays AS (
 SELECT
 patientid,
 day,
 MAX_glucose_measured,
 RANK() OVER (
 PARTITION BY patientid
 ORDER BY MAX_glucose_measured DESC
    ) AS rank_by_glucose
    FROM Dailyglucose
)

SELECT
patientid,
day,
MAX_glucose_measured
FROM RankedglucoseDays
WHERE rank_by_glucose = 1
ORDER BY patientid;
```

OUTPUT

Data Output | Messages | Notifications

	patientid integer	day timestamp without time zone	max_glucose_measured numeric
1	1	2020-02-19 00:00:00	155.00
2	2	2020-02-25 00:00:00	200.00
3	3	2020-02-22 00:00:00	197.00
4	4	2020-03-03 00:00:00	189.00
5	5	2020-02-27 00:00:00	181.00
6	6	2020-03-08 00:00:00	222.00
7	7	2020-03-21 00:00:00	192.00
8	8	2020-03-20 00:00:00	180.00
9	9	2020-03-26 00:00:00	237.00
10	10	2020-03-29 00:00:00	261.00

Total rows: 16 Query complete 00:00:00.180

Insights:

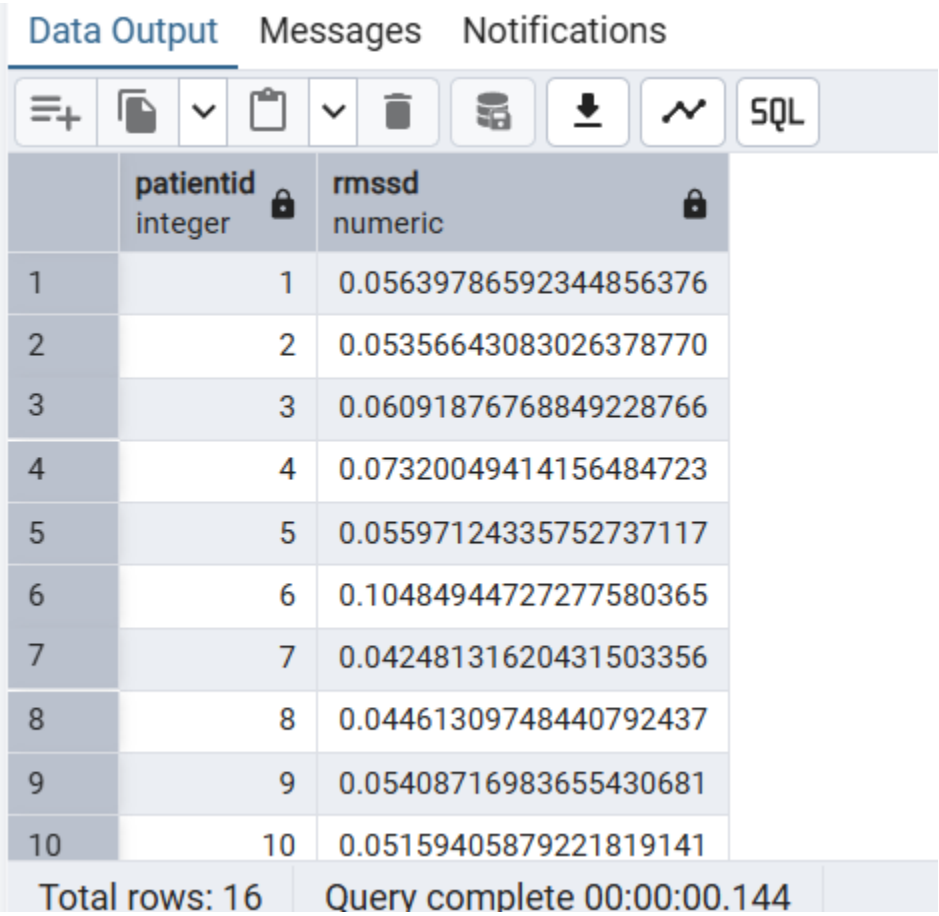
These results explain on which day a patient had maximum glucose level compared to all other days. We can relate this with the previous query result. For example, let's consider patientid 1. He had maximum calorie food on 18thand now in the above result, the same patient id 1 has measured maximum glucose level on the next day. This shows that high calorie food is definitely one of the main reasons for glucose levels spike.

15. Calculate the RMSSD (Root mean square of successive differences) values using ibi values of patients

QUERY:

```sql
/*To calculate the RMSSD*/

WITH lagged_ibi AS (
    SELECT
        patientid,
        ibi,
        LAG(ibi, 1) OVER (PARTITION BY patientid ORDER BY timeof) AS previous_ibi
    FROM gv_ibi
    WHERE ibi IS NOT NULL
)
SELECT
    patientid,
    SQRT(AVG(POWER(ibi - previous_ibi, 2))) AS RMSSD
FROM lagged_ibi
WHERE previous_ibi IS NOT NULL
GROUP BY patientid
ORDER BY patientid;
```

OUTPUT



Data Output | Messages | Notifications

	patientid integer	rmssd numeric
1	1	0.05639786592344856376
2	2	0.05356643083026378770
3	3	0.06091876768849228766
4	4	0.07320049414156484723
5	5	0.05597124335752737117
6	6	0.10484944727277580365
7	7	0.04248131620431503356
8	8	0.04461309748440792437
9	9	0.05408716983655430681
10	10	0.05159405879221819141

Total rows: 16 | Query complete 00:00:00.144

Insights:

RMSSD (Root Mean Square of Successive Differences) is a crirtical parameter and highly regarded time-domain measure of Heart Rate Variability (HRV). It specifically reflects the short-term, beat-to-beat variability in heart rate and is considered a strong indicator of parasympathetic nervous system (PNS) or vagal tone activity.

Here's what we can infer from RMSSD values: Higher RMSSD values indicate higher vagal tone and more robust parasympathetic nervous system activity. This suggests that the body is in a more relaxed state, capable of efficient "rest and digest" functions and recovery.

Lower RMSSD values suggest reduced vagal tone and diminished parasympathetic influence. This often points towards a state of stress, fatigue, or less efficient recovery mechanisms.

Typical ranges for healthy adults at rest can vary significantly, often extending from below 20 to over 70 milliseconds (ms), with young athletes potentially exhibiting values exceeding 200ms.

Here we can see that most of the patients have RMSSD values between 40-105 milliseconds.