**Q1. write a query to get the number of patients for each age CATEGORY using case statement**

**Query**

WITH CTE AS(

SELECT PATIENTID,date\_part('year',age(d.dob)) as patient\_age FROM DEMOGRAPHICS D

)

SELECT '20 TO 30' AS AGE\_RANGE, SUM(CASE WHEN C.PATIENT\_AGE>20  AND C.PATIENT\_AGE<30 THEN 1 ELSE 0 END) AS count\_below\_range

FROM DEMOGRAPHICS D, CTE C

WHERE C.PATIENTID=D.PATIENTID AND C.patient\_age BETWEEN 20 AND 30

UNION ALL

SELECT '30 TO 50' AS AGE\_RANGE, SUM(CASE WHEN C.PATIENT\_AGE>30  AND C.PATIENT\_AGE<50 THEN 1 ELSE 0 END) AS count\_below\_range

FROM DEMOGRAPHICS D, CTE C

WHERE C.PATIENTID=D.PATIENTID AND C.patient\_age BETWEEN 30 AND 50

UNION ALL

SELECT '50 TO 60' AS AGE\_RANGE,  SUM(CASE WHEN C.PATIENT\_AGE>50  AND C.PATIENT\_AGE<60 THEN 1 ELSE 0 END) AS count\_below\_range

FROM DEMOGRAPHICS D, CTE C

WHERE C.PATIENTID=D.PATIENTID AND C.patient\_age BETWEEN 50 AND 55

UNION ALL

SELECT '60 TO 70' AS AGE\_RANGE,  SUM(CASE WHEN C.PATIENT\_AGE>60  AND C.PATIENT\_AGE<70 THEN 1 ELSE 0 END) AS count\_below\_range

FROM DEMOGRAPHICS D, CTE C

WHERE C.PATIENTID=D.PATIENTID AND C.patient\_age BETWEEN 60 AND 70

UNION ALL

SELECT '70 TO 80' AS AGE\_RANGE, SUM(CASE WHEN C.PATIENT\_AGE>70  AND C.PATIENT\_AGE<80 THEN 1 ELSE 0 END) AS count\_below\_range

FROM DEMOGRAPHICS D, CTE C

WHERE C.PATIENTID=D.PATIENTID AND C.patient\_age  BETWEEN 70 AND 80

**Output**

**A white rectangular object with a black border

Description automatically generated**

**Q2. Display patients names who have the same last name.**

**Query**

select concat(d.Firstname||' '||d.Lastname) as Patient\_name from demographics d where d.Lastname in (select Lastname from demographics

group by Lastname

having count(\*) > 1)

**OUTPUT**

**A close-up of a computer screen

Description automatically generated**

**Q3. Write a query to the get number of Patient\_IDs who visited between the year of 1970 and 1990**

select count(d.patientid) as number\_of\_patients

from demographics d

where to\_char("dob",'yyyy-mm') between '1970-01' and '1979-01'

A white rectangular object with a white border

Description automatically generated

**Q4. Write a query to get comma-separated values of patient details .(Use a maximum of 6 columns from different tables)**

**Query**

SELECT

    concat(d.patientid, ',',d.firstname,',',d.lastname,',',d.gender,',',d.hba1c,',',d.dob,',',ROUND(CAST(h.mean\_hr AS NUMERIC), 2) ,',',g.glucose\_value\_mgdl) as PATIENT\_DETAILS

   FROM demographics d

LEFT JOIN hr h ON d.patientid = h.patientid

LEFT JOIN dexcom g ON d.patientid = g.patientid

group by patient\_details;

**OUTPUT**

**A computer screen with a green and white rectangle

Description automatically generated**

**Q5. Write a query to determine get the Patient IDs details who are in  prediabetic stage .**

SELECT

    d.patientid,

    d.gender,

    d.hba1c ,

case

  when d.hba1c <5.7 then 'normal'

  when d.hba1c >5.7 and d.hba1c<6.4 then 'prediabetic'

  when  d.hba1c>=6.4 then 'diabetic'

  else 'invalid range'

  end range

FROM

    demographics d

WHERE

   d.hba1c >5.7 and d.hba1c<6.4

**OUTPUT**

A white rectangular object with a black border

Description automatically generated

**Q6. Display any 10 random DM patients.**

**Query:**

Select d.patientid, concat(d.firstname,'',d.lastname) as patient\_name

From demographics d

Order By Random(),patientid asc

LIMIT 10

**OUTPUT**

A white rectangular object with a black border

Description automatically generated

**Q7. Write a query by using common table expressions and case statements to display birthyear ranges for each patient.**

**Query**

WITH CTE AS(

SELECT PATIENTID,date\_part('year',age(d.dob)) as patient\_age FROM DEMOGRAPHICS D

)

Select c. patient\_age,

Case

WHEN c.patient\_age > 30 And  patient\_age <=50  THEN '30-50'

WHEN c.patient\_age > 50 And  patient\_age <=60  THEN '50-60'

WHEN c.patient\_age > 60 And  c.patient\_age <=70  THEN '60-70'

ELSE '70+'

End "birthyear ranges"

From cte c

**OUTPUT**

**A white rectangular object with a white border

Description automatically generated**

**Q8. Display a list of Patient IDs and  glucose level**

**Query**

SELECT patientid,

       hba1c,

CASE WHEN hba1c <=5.7 THEN 'normal'

WHEN hba1c >=5.7 and hba1c<=6.4 THEN 'Pre-diabetic'

WHEN hba1c >=6.5 THEN 'Diabetic'

    End as  glucose\_level

FROM DEMOGRAPHICS D

**OUTPUT**

**A white rectangular object with a black border

Description automatically generated**

**Q9. Count of patients by first letter of firstname.**

**Query**

SELECT SUBSTRING(d.firstname,1,1) AS FIRST\_Letter, COUNT(\*)

FROM demographics d

GROUP BY 1;

**OUTPUT**

**A close-up of a computer screen

Description automatically generated**

**Q10. Calculate the Average HR with Handling NULL values with 0**

 SELECT

    patientid,

    AVG(COALESCE(mean\_hr, 0)) AS average\_heart\_rate

FROM

    hr

GROUP BY

    patientid;

**OUTPUT**

**A close-up of a computer screen

Description automatically generated**

**Q11.  provide a list of patients grouped into three age-based categories with HbA1c levels greater than 5.0? Include their full names, ages, and HbA1c levels in each category."**

**Query**

   with cte as (

select patientid, date\_part('year',age(dob)) as age from demographics group by patientid

)SELECT

  concat(Firstname,' ',Lastname) as Fullname,

  c.age,

  NTILE(3) OVER(

ORDER BY c.age

  ),

  hba1c as glucose\_level

     FROM

demographics, cte c

    WHERE

hba1c >5.0

**OUTPUT**

**A close-up of a computer screen

Description automatically generated**

**Q12. Rank the patients in the dataset based on their average heart rate (mean\_hr) in descending order.**

**Query**

WITH RankedPatients AS (

    SELECT

        d.patientid,

        h.mean\_hr,

        RANK() OVER (ORDER BY h.mean\_hr DESC) AS heart\_rate\_rank

    FROM

        demographics d

    JOIN

        hr h ON d.patientid = h.patientid

)

SELECT

    patientid,

     heart\_rate\_rank

FROM

    RankedPatients

ORDER BY

    heart\_rate\_rank;

**OUTPUT**

A close-up of a computer screen

Description automatically generated

**Q13. Calculate the 25th and 75th percentiles of heart rate (mean\_hr) for each patient:**

**Query**

SELECT

    patientid,

    percentile\_cont(0.25) WITHIN GROUP (ORDER BY mean\_hr) AS pcont\_heart\_rate,

    percentile\_disc(0.75) WITHIN GROUP (ORDER BY mean\_hr) AS pdisc\_heart\_rate

FROM hr

GROUP BY patientid;

**OUTPUT**

**A white rectangular object with a black border

Description automatically generated**

**Q14. Return the patientid without glucose value data in the table and instead of null change it to 0.**

**Query 1 using coalesce function and is null function**

select

d.patientid,

coalesce(dx.glucose\_value\_mgdl,0) as glucose\_value

from dexcom dx

right join demographics d

on d.patientid=dx.patientid

where glucose\_value\_mgdl is null

**OUTPUT**

**A white rectangular object with a black border

Description automatically generated**

**Query 2 updating the data in to the table using null if function**

UPDATE dexcom

SET glucose\_value\_mgdl = NULLIF(glucose\_value\_mgdl, 0);

**OUTPUT**

A white background with a black border

Description automatically generated with medium confidence

**Q15. generate rownumber for demographics table**

**Query**

select patientid, row\_number() over(order by patientid) from demographics

**OUTPUT**

A close-up of a computer screen

Description automatically generated

**Q16. Determine the calorie difference between the current food log entry and the previous food log entry by using LAG function.**

**Query**

WITH CalorieChanges AS (

    SELECT

        f.patientid,

        f.calorie,

        f.datetime::date as log\_date,

    COALESCE(LAG(f.calorie) OVER (PARTITION BY f.patientid ORDER BY f.datetime::date),0) AS previous\_calorie,

        f.logged\_food

    FROM foodlog f

)

SELECT

    patientid,

    logged\_food,

    log\_date,

    calorie,

    previous\_calorie,

    COALESCE(calorie - previous\_calorie, 0) AS calorie\_diff

FROM CalorieChanges

ORDER BY patientid, log\_date;

**OUTPUT**

**A white and grey text box

Description automatically generated with medium confidence**

**Q17 Identifying the food\_logged and the log\_date for the food log entry two entries after each food logged entry using the LEAD function.**

**Query**

WITH FoodLogEntries AS (

    SELECT

        patientid,

        logged\_food,

        datetime::date as log\_date,

       coalesce( LEAD(logged\_food, 2) OVER (PARTITION BY patientid ORDER BY datetime::date),'N/A') AS food\_two\_entries\_after,--skip by 2 entries

       coalesce( LEAD(datetime::date, 2) OVER (PARTITION BY patientid ORDER BY datetime::date),null) AS date\_two\_entries\_after --skip by 2 entries

    FROM foodlog

)

SELECT

    patientid,

    logged\_food,

    log\_date,

    food\_two\_entries\_after,

    date\_two\_entries\_after

FROM FoodLogEntries

ORDER BY patientid, log\_date;

**OUTPUT**

**A white rectangular object with black lines

Description automatically generated**

**Q18. Write a query to get a list of Patients order by age ascending order.**

**Query**

       SELECT 'Firstname'||' '|| 'Lastname' AS full\_name, dob, EXTRACT(year FROM age(current\_date,dob)) :: int as age

       FROM demographics

       ORDER BY age ASC;

**OUTPUT**

**A white rectangular object with a white border

Description automatically generated**

**Q19. Write a query to find the max and min dates for each patient where the glucose entry is done**

**Query**

SELECT

        patientid,

        MIN(datestamp) AS min\_date,

        MAX(datestamp) AS max\_date

FROM

        dexcom

GROUP BY

        Patientid

**OUTPUT**

**A white rectangular object with a black border

Description automatically generated**

**Q20. generate a series of date using (generate\_series)**

**Query**

SELECT ('2020-06-06'::DATE + (generate\_series(0, 10) \* interval '1 day'))::DATE;

**OUTPUT**

**A close-up of a computer screen

Description automatically generated**

**Q21. write query to find out how many hours each patient was under observation**

**Query**

SELECT

    patientid,

    ROUND(EXTRACT(EPOCH FROM (MAX(datestamp) - MIN(datestamp)) / 3600), 2) AS under\_obbservation\_hours

FROM dexcom

GROUP BY patientid

ORDER BY patientid ASC;

**OUTPUT**

A close-up of a computer screen

Description automatically generated

**Q22. format the DOB field in human readable format**

**Query**

SELECT

  patientid,

  TO\_CHAR(dob, 'Mon DD, YYYY') AS formatted\_date

FROM demographics;

**OUTPUT**

A white rectangular object with a black border

Description automatically generated

**Q23. write a query to findout the days each patient was under observation**

**Query**

SELECT

    patientid,

    COUNT(DISTINCT datestamp::DATE) AS number\_of\_tested\_days

FROM dexcom

GROUP BY patientid

ORDER BY patientid;

**OUTPUT**

**A close-up of a white rectangular object

Description automatically generated**

**Q24. write a query to display the patient names as array**

**Query**

SELECT

     ARRAY\_AGG (Firstname|| ' ' ||Lastname|| ' '||PatientID) Fullname

FROM

     Demographics

**OUTPUT**

A screenshot of a computer

Description automatically generated

**Q25. print the patient\_with\_highest\_calorie\_intake and patient\_with\_lowest\_calorie\_intake**

**Query**

SELECT

   -- patientid,

    FIRST\_VALUE(patientid) OVER (ORDER BY calorie DESC) AS patient\_with\_highest\_calorie\_intake,

    LAST\_VALUE(patientid) OVER (ORDER BY calorie) AS patient\_with\_lowest\_calorie\_intake

FROM foodlog

LIMIT 1;

**OUTPUT**

**A white rectangular object with a black border

Description automatically generated**

**Q26. Rank Patients Based on Missing Demographic Information query**

**Query**

SELECT

    patientid,

    COUNT(NULLIF(firstname IS NULL, true)) +

    COUNT(NULLIF(lastname IS NULL, true)) +

    COUNT(NULLIF(dob IS NULL, true)) +

    COUNT(NULLIF(hba1c IS NULL, true)) +

    COUNT(NULLIF(gender IS NULL, true)) AS missing\_demographics\_count

FROM

    demographics

GROUP BY

    patientid

ORDER BY

    missing\_demographics\_count DESC;

**OUTPUT**

A white rectangular object with a white border

Description automatically generated with medium confidence

**Q27. Display all the patients with Hyperglcemia and Hypoglcemia \*\*\*\*\*added nulls last**

**Query**

SELECT d.patientid, d.glucose\_value\_mgdl,

CASE

      WHEN  d.glucose\_value\_mgdl <70 then 'Low level(Hypoglcemia)'

  WHEN d.glucose\_value\_mgdl  >= 126 then 'High level(Hyperglcemia)'

  WHEN d.glucose\_value\_mgdl IS NULL then 'invalid'

  ELSE 'Normal'

  END

 FROM dexcom d

 ORDER By patientid desc NULLS LAST;

**OUTPUT**

**A white rectangular object with a white border

Description automatically generated**

**Q29.Display full name of all the patients with normal HbA1c level.**

**Query**

SELECT CONCAT(firstname,' ',lastname) AS fullname,hba1c

FROM demographics

WHERE  hba1c < 5.7;

**OUTPUT**

A close-up of a computer screen

Description automatically generated

**Q28.List all the patients with high stress level using hr and hrv value . \*\*\*\*\*added hrv value also to find the stress level**

**Query**

SELECT h.patientid,h.mean\_hr,i.hrv FROM hr h,ibi i

WHERE mean\_hr > 100 and hrv<20

ORDER BY patientid ASC;

**OUTPUT**

**A white rectangular object with a white border

Description automatically generated**

**Q28.Calculate the number of calories eaten by patient 4 on date 2020-03-04.**

**Query**

SELECT patientid,SUM(calorie) AS total\_calorie

FROM foodlog

WHERE patientid= '4' AND datetime::DATE = '2020-03-04'

GROUP BY patientid;

**OUTPUT**

A white rectangular object with a white border

Description automatically generated

**Q29.Which patient has maximum fluctuation in blood glucose level.**

**Query**

WITH sd AS (

SELECT patientid,STDDEV(glucose\_value\_mgdl) AS standard\_deviation

FROM dexcom

GROUP BY patientid

)

SELECT patientid,MAX(standard\_deviation) AS max\_flucation

FROM sd

GROUP BY patientid

ORDER BY max\_flucation DESC

LIMIT 1;

**OUTPUT**

**A close-up of a computer screen

Description automatically generated**

**Q30.create a user defined function that returns the patient whose name ends with input**

**Query**

create  or replace function get\_patientrec\_details(inputletter varchar)

returns setof demographics

language plpgsql

as

$$

begin

return query

select patientid,gender,hba1c, dob, firstname || ' ' || lastname as fullname, lastname

from demographics

where lastname like '%' || inputletter;

end;

$$

**OUTPUT**

A white rectangular object with a black border

Description automatically generated

select \* from get\_patientrec\_details('s')

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**Q31. select eldest and youngest patients**

**Query**

SELECT patientid,

       CONCAT(firstname, ' ', lastname) AS full\_name,

       DATE\_PART('YEAR', age(current\_date, dob)) AS age

FROM demographics

WHERE DATE\_PART('YEAR', age(current\_date, dob)) = (

    SELECT MAX(DATE\_PART('YEAR', age(current\_date, dob)))

    FROM demographics

)

OR DATE\_PART('YEAR', age(current\_date, dob)) = (

    SELECT MIN(DATE\_PART('YEAR', age(current\_date, dob)))

    FROM demographics

);

**OUTPUT**

A close-up of a computer screen

Description automatically generated

**Q32.display patient id and highest calorie consumed on each day**

**Query**

SELECT patientid,

       DATE(datetime) AS day,

       MAX(calorie) AS max\_calorie

FROM foodlog

GROUP BY patientid, day order by patientid

**OUTPUT**

A close-up of a computer screen

Description automatically generated

**Q33. select the second youngest patient**

**Query**

SELECT patientid,

       CONCAT(firstname, ' ', lastname) AS full\_name,

       DATE\_PART('YEAR', age(current\_date, dob)) AS age

FROM demographics

order by age ASC

OFFSET 1 LIMIT 1

**OUTPUT**

A close-up of a computer screen

Description automatically generated

**Q34.  SELECT THE MOST UNHEALTY FOOD**

**Query**

select logged\_food  from foodlog order by calorie desc limit 1

**OUTPUT**

A white rectangular object with a black border

Description automatically generated