## TASK 3

# SQL-DIABETES PREDICTION

1. Retrieve the Patient_id and ages of all patients.

```
SELECT Patient_id,age
FROM diabetes_prediction;
```

2. Select all female patients who are older than 40.

```
SELECT gender,age,Patient_id
FROM diabetes_prediction
WHERE gender="female" and age>40;
```

3. Calculate the average BMI of patients.

```
SELECT Avg(bmi) AS Average_bmi
FROM  diabetes_prediction;
```

4. List patients in descending order of blood glucose levels.

```
SELECT Patient_id,blood_glucose_level
FROM diabetes_prediction
ORDER BY blood_glucose_level DESC;
```

5. Find patients who have hypertension and diabetes.

```
SELECT Patient_id,hypertension,diabetes
FROM diabetes_prediction
WHERE hypertension="1" and diabetes="1";
```

6. Determine the number of patients with heart disease.

```
SELECT COUNT(*) AS num_heart_disease
FROM diabetes_prediction
WHERE heart_disease="1" ;
```

7. Group patients by smoking history and count how many smokers and non-smokers there are.

```
SELECT smoking_history,COUNT(*) AS num_patients
FROM diabetes_prediction
GROUP BY smoking_history;
```

8. Retrieve the Patient_ids of patients who have a BMI greater than the average BMI.

```
SELECT patient_id
FROM diabetes_prediction
WHERE   bmi > (SELECT AVG(bmi) from
diabetes_prediction);
```

9. Find the patient with the highest HbA1c level and the patient with the lowest HbA1clevel.

```
SELECT * FROM diabetes_prediction
ORDER BY  HbA1c_level DESC
LIMIT 1;

SELECT * FROM diabetes_prediction
ORDER BY  HbA1c_level ASC
LIMIT 1;
```

10. Calculate the age of patients in years (assuming the current date as of now).

```
SELECT Patient_id,age
DATEDIFF(curdate(),str_to_date(age,"%y-
 %m=%d"))/365
AS calculated_age
FROM diabetes_prediction;
```

11. Rank patients by blood glucose level within each gender group.

```
RANK() OVER (PARTITION BY gender ORDER BY
blood_glucose_level DESC)
AS glucose_rank
FROM diabetes_prediction;
```

12. Update the smoking history of patients who are older than 50 to "Ex-smoker."

```
UPDATE diabetes_prediction
SET smoking_history="Ex-smoker"
WHERE age>50;
```

13. Insert a new patient into the database with sample data.

```
    INSERT into
diabetes_prediction(Patient_id,gender,age,hypertension,h
eart_disease,smoking_history,bmi,HbA1c_level,blood_gluc
ose_level,diabetes)
    Values(“Shyam”,”PT100101”,”male”,23,0,0,”No
    info”,30.2,6.1,120,0);
```

14. Delete all patients with heart disease from the database.

```
SELECT *FROM diabetes_prediction
WHERE heart_disease="1";
```

15. Find patients who have hypertension but not diabetes using the EXCEPT operator

```
SELECT Patient_id FROM diabetes_prediction
WHERE hypertension="1"
EXCEPT SELECT Patient_id
FROM diabetes_prediction
WHERE diabetes="0";
```

16. Define a unique constraint on the "patient_id" column to ensure its values are unique.

```
ALTER table diabetes_prediction
ADD CONSTRAINT unique_patient_id
UNIQUE(Patient_id);
```

17. Create a view that displays the Patient_ids, ages, and BMI of patients.

```
CREATE VIEW details  AS
SELECT Patient_id,age,bmi
FROM diabetes_prediction;
```

18. Suggest improvements in the database schema to reduce data redundancy and improve data integrity.

- Consider normalizing the data to reduce redundancy. For instance, create separate tables for patient details, smoking history, and medical conditions, linking them using foreign keys.
- Use appropriate data types for each column. For example, use DATE for birth_date instead of age, and ensure that numerical values like BMI, HbA1c levels, and blood glucose levels use the correct numeric types.
- Ensure that the schema adheres to the principles of database normalization to minimize data anomalies.

19. Explain how you can optimize the performance of SQL queries on this dataset.

- Index columns frequently used in WHERE clauses, especially patient_id, to speed up search operations.
- Regularly analyze and optimize queries using the EXPLAIN statement to understand their execution plans.
- Consider partitioning large tables to improve query performance, especially if certain subsets of data are queried more frequently than others.

These are general suggestions, and the effectiveness may depend on the specific characteristics of your dataset and usage patterns.