DIABETES PREDICTION

TASK 1

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 * FROM diabetes_prediction
WHERE gender='Female' AND age>40;
```

3. Calculate the average BMI of patients.

```
SELECT employeename,patient_id,ROUND(AVG(bmi::numeric), 2) AS bmi_avg FROM
diabetes_prediction
GROUP BY employeename,patient_id
ORDER BY bmi_avg DESC ;
```

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

```sql
SELECT employeename,patient_id,blood_glucose_level
FROM diabetes_prediction
ORDER BY blood_glucose_level DESC;
```

5. Find patients who have hypertension and diabetes.

```sql
SELECT employeename,patient_id
FROM diabetes_prediction
WHERE hypertension=1 AND diabetes=1;
```

6. Determine the number of patients with heart disease.

```sql
SELECT COUNT(patient_id) AS num_of_patients
FROM diabetes_prediction
WHERE heart_disease=1;
```

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

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

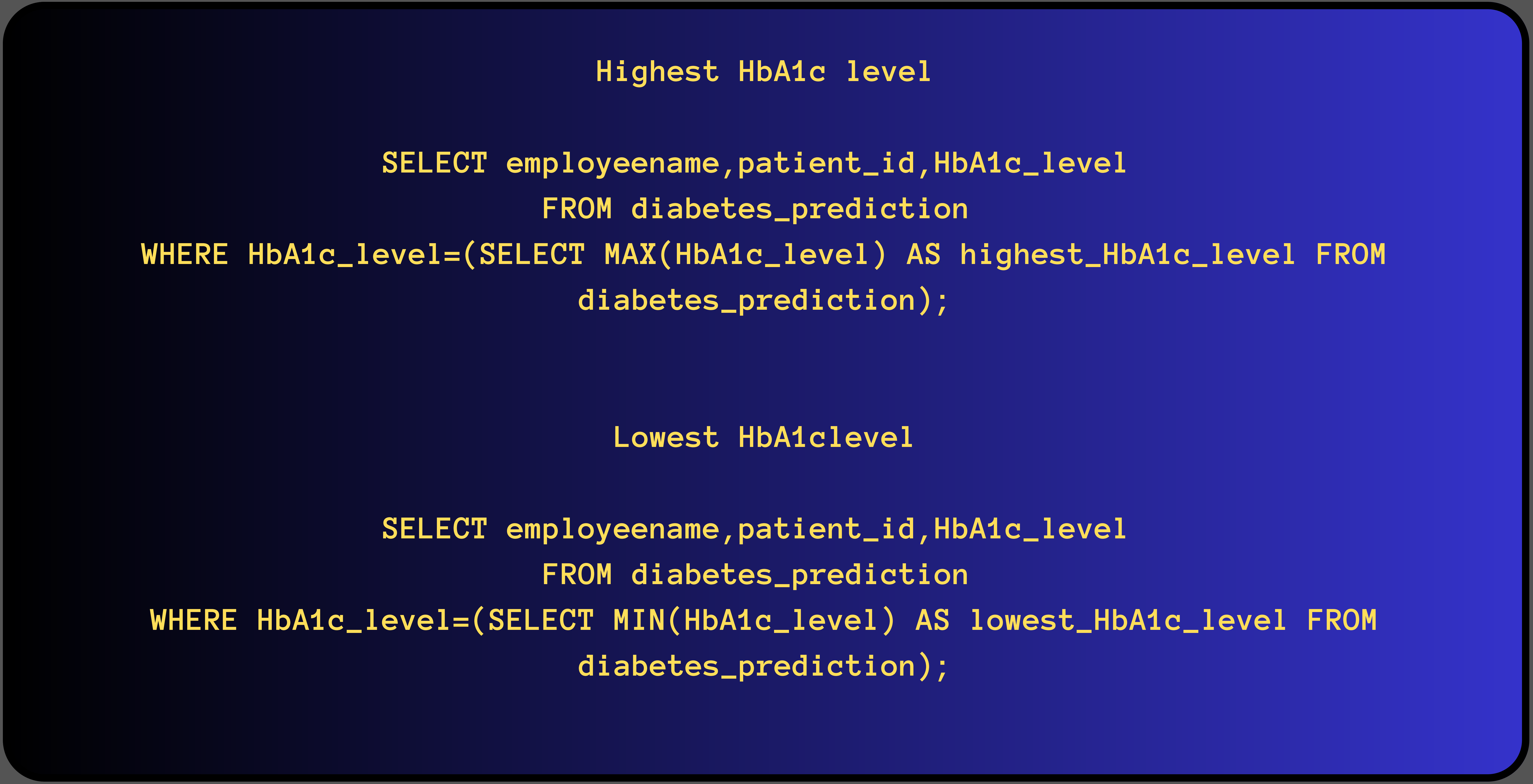
8. Retrieve the Patient_ids of patients who have a BMI greater than the average BMI.

```
SELECT patient_id
FROM diabetes_prediction
WHERE bmi>(SELECT ROUND(AVG(bmi::numeric), 2) FROM diabetes_prediction);
```

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

```
SELECT Patient_id,age,
DATEDIFF(CURRENT_DATE, STR_TO_DATE(age, '%Y-%m-%d')) / 365 AS age
FROM diabetes_prediction;
```

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



```
Highest HbA1c level

SELECT employeename,patient_id,HbA1c_level
FROM diabetes_prediction
WHERE HbA1c_level=(SELECT MAX(HbA1c_level) AS highest_HbA1c_level FROM
diabetes_prediction);


Lowest HbA1clevel

SELECT employeename,patient_id,HbA1c_level
FROM diabetes_prediction
WHERE HbA1c_level=(SELECT MIN(HbA1c_level) AS lowest_HbA1c_level FROM
diabetes_prediction);
```

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

```
SELECT employeename,patient_id,gender,blood_glucose_level,
RANK() OVER(PARTITION BY gender ORDER BY blood_glucose_level DESC) AS
glucose_level_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;
SELECT * FROM diabetes_prediction;
```

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

```
DELETE FROM diabetes_prediction
WHERE heart_disease = 1;
```

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

```
INSERT INTO diabetes_prediction
(EmployeeName, Patient_id, gender, age, hypertension, heart_disease, smoking_history,
bmi,
HbA1c_level, blood_glucose_level, diabetes)
VALUES
('PETER', 'PR134', 'Female', 42, 0, 0, 'non-smoker', 39.5, 5.9, 129, 0);

SELECT * FROM diabetes_prediction;
```

15. 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);
```

16. 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=1;
```

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

```
CREATE VIEW patient_info AS
SELECT patient_id,age,bmi
FROM diabetes_prediction;

SELECT * FROM diabetes_prediction;
```

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

Ensure the database is normalized tables to minimize data redundancy and

Break down tables to eliminate repeating groups of data.

Establish proper primary keys for the uniquely identify records and foreign key is for relationships between tables.

Choose appropriate data types.

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

It involves various strategies to ensure efficient and fast data retrieval,and ensure the keys

Choose normalization appropiate data types.

Use appropriate join types (depending on the relationship between tables.