Name: Priyam Bhattacharya
Email: itspriyambhattacharya@gmail.com
LinkedIn: https://linkedin.com/in/itspriyambhattacharya

Diabetes Prediction

1. Retrieve the Patient_id and ages of all patients.

Ans : SELECT Patient_id, age FROM dp;

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

Ans : SELECT * FROM dp WHERE gender = 'Female' AND age > 40;

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

Ans : SELECT * FROM dp ORDER BY blood_glucose_level DESC;

5. Find patients who have hypertension and diabetes.

Ans : SELECT * FROM dp WHERE hypentension = 1 AND diabetes = 1;

6. Determine the number of patients with heart disease.

Ans : SELECT COUNT(*) AS number_of_patients_with_heart_disease FROM dp WHERE heart_disease = 1;

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

Ans : SELECT smoking_history, COUNT(*) AS number_of_patients

FROM dp

GROUP BY smoking_history;

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

Ans : SELECT Patient_id

FROM dp

WHERE bmi > (SELECT AVG(bmi) FROM dp);

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

Ans :

-- Patient with the highest HbA1c level

SELECT *

FROM dp

WHERE HbA1_level = (SELECT MAX(HbA1_level) FROM dp);

Name: Priyam Bhattacharya
Email: itspriyambhattacharya@gmail.com
LinkedIn: https://linkedin.com/in/itspriyambhattacharya

-- Patient with the lowest HbA1c level

SELECT *

FROM dp

WHERE HbA1_level = (SELECT MIN(HbA1_level) FROM dp);

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

Ans : SELECT Patient_id, age,

DATEDIFF(CURDATE(), STR_TO_DATE(age, '%Y-%m-%d')) / 365 AS calculated_age

FROM dp;

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

Ans : SELECT Patient_id, gender, blood_glucose_level,

RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level DESC) AS glucose_level_rank

FROM dp;

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

Ans : UPDATE dp

SET smoking_history = 'Ex-smoker'

WHERE age > 50;

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

Ans : INSERT INTO dp

(EmployeeName, Patiend_id, gender, age, hypentension, heart_disease, smoking_history, bmi, HbA1_level, blood_glucose_level, diabetes)

VALUES

('John Doe', 'P123456', 'male', 35, 'no', 'no', 'non-smoker', 25.5, 5.7, 120, 'no');

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

Ans : DELETE FROM dp

WHERE heart_disease = 1;

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

Ans : SELECT Patient_id

Name: Priyam Bhattacharya
Email: itspriyambhattacharya@gmail.com
LinkedIn: https://linkedin.com/in/itspriyambhattacharya

FROM dp

WHERE hypentension = 1

EXCEPT

SELECT Patient_id

FROM dp

WHERE diabetes = 1;

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

Ans : ALTER TABLE dp ADD UNIQUE (patient_id);

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

Ans : CREATE VIEW patient_info AS

SELECT Patient_id, age, bmi

FROM dp;

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

Ans : To reduce data redundancy and improve data integrity in our database schema, we can consider the following suggestions:

1. **Normalization:** Ensure our database follows normalization rules, particularly up to at least the third normal form (3NF). This minimizes redundancy by organizing data efficiently.
2. **Use of Primary Keys:** Ensure each table has a primary key to uniquely identify each record. This helps in avoiding duplicate entries.
3. **Foreign Keys:** Use foreign keys to establish relationships between tables. This maintains referential integrity and prevents inconsistencies.
4. **Data Types and Constraints:** Choose appropriate data types for columns to minimize storage space.
5. **Composite Keys:** In cases where a combination of columns can uniquely identify a record, consider using a composite key instead of a single column as the primary key.

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

Ans : Optimizing the performance of SQL queries involves various strategies, and the specific approach depends on the nature of your dataset, the complexity of your queries, and the underlying database management system. Here are some general tips to optimize SQL queries on our dataset:

Name: Priyam Bhattacharya
Email: itspriyambhattacharya@gmail.com
LinkedIn: https://linkedin.com/in/itspriyambhattacharya

1. **Use Indexing:** Identify columns frequently used in WHERE clauses and JOIN conditions and create indexes on those columns. Be cautious not to over-index, as it can impact write performance.

2. **Optimize JOIN Operations:** Use INNER JOIN instead of OUTER JOIN when possible, as INNER JOIN is generally more efficient. Ensure that columns used in JOIN conditions are indexed.

3. **Limit the Result Set:** Only retrieve the columns you need. Avoid using SELECT * if you don't need all columns. Use the LIMIT clause to restrict the number of rows returned, especially for large datasets.

4. **Avoid SELECT DISTINCT:** Use SELECT DISTINCT sparingly, as it can be resource-intensive. Consider whether it's necessary or if alternative approaches can achieve the same result.