Muskan Kashyap

Diabetes Prediction Project - SQL

PSYLIQ Internship Project

Content

Introduction

In this presentation, I'll walk you through my SQL analysis of a 100,000-row diabetes dataset. Diabetes, a widespread health condition affecting millions, is characterized by abnormal blood sugar levels. I’ve explored key insights, relation between diabetes and heart conditions, alongside the impact of factors like age, BMI, and glucose levels.

Project Questions

And SQL Queries

Q1: Retrieve the patient_id and ages of all patients

```sql
select patient_id, age
from diabetes;
```

Q2: Select all female patients who are older than 40

```sql
select * from diabetes
where gender = 'Female' and age > 40;
```

Q3: Calculate the average BMI of patients

```
select avg(bmi) as Average_BMI
from diabetes;
```

Q4: List patients in descending order of blood glucose levels.

```
select * from diabetes
order by blood_glucose_level desc;
```

Q5: Find patients who have hypertension and diabetes.

```
select * from diabetes
where hypertension = 1 and diabetes = 1;
```

Q6: Determine the number of patients with heart disease.

```
select count(*) as patients_with_heart_disease from diabetes
where heart_disease = 1;
```

Q7: Group patients by smoking history and count how many smokers and nonsmokers there are.

```
select smoking_history, count(*) from diabetes
group by smoking_history;
```

Q8: Retrieve the Patient_ids of patients who have a BMI greater than the average BMI.

```
select patient_id from diabetes
where bmi > (select avg(bmi)
             from diabetes);
```

Q9: Find the patient with the highest HbA1c level and the patient with the lowest HbA1clevel.

```
select patient_id from diabetes
where bmi > (select avg(bmi)
             from diabetes);
```

Q10: Calculate the age of patients in years (assuming the current date as of now).

```
select max(hba1c_level) as Highest_hba1c_level,
       min(hba1c_level) as lowest_hba1c_level
from diabetes;
```

Q11: Rank patients by blood glucose level within each gender group.

```sql
select
  patient_id,
  gender,
  blood_glucose_level,
  RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level DESC) AS glucose_rank
from
  diabetes;
```

Q12: Update the smoking history of patients who are older than 50 to "Ex-smoker."

```sql
UPDATE diabetes
SET smoking_history = 'Ex-smoker'
WHERE age > 50;
```

Q13: Delete all patients with heart disease from the database.

```sql
DELETE FROM diabetes
WHERE heart_disease = 1;
```

Q14: Insert a new patient into the database with sample data.

```
INSERT INTO diabetes (employeename, patient_id,
                      gender, age,
                      hypertension,
                      heart_disease,
                      smoking_history,
                      bmi, hba1c_level,
                      blood_glucose_level,
                      diabetes)
VALUES ('EXTRA NAME',
        'P0',
        'Female',
        23,
        0,
        0,
        'never',
        25,
        5,
        155,
        0);
```

Q15: Find patients who have hypertension but not diabetes using the EXCEPT operator.

```sql
SELECT patient_id
FROM diabetes
WHERE hypertension = 1

EXCEPT

SELECT patient_id
FROM diabetes
WHERE diabetes = 1;
```

Q16: Define a unique constraint on the "patient_id" column to ensure its values are unique.

```
ALTER TABLE diabetes
ADD CONSTRAINT unique_patient_id UNIQUE(patient_id);
```

Q17: 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;

SELECT * FROM patient_info;
```

Q18: Suggest improvements in the database schema to reduce data redundancy and improve data integrity.

- Identify and eliminate redundancy by normalizing the database. Break down the data into smaller, related tables to avoid storing the same information in multiple places.
- Create a separate table for patient information, containing columns such as patient_id, employeename, gender, age, and smoking_history.
- Use the patient_id as a primary key in this table and reference it as a foreign key in other relevant tables.

Q19: Explain how you can optimize the performance of SQL queries on this dataset.

- Identify columns used in WHERE clauses, JOIN conditions, and ORDER BY clauses and create indexes on those columns. This can significantly speed up data retrieval.

- Only select the columns that are necessary for your query. Avoid using SELECT * if you don't need all columns, as fetching unnecessary data can impact performance.

Summary

In this Internship project, I've provided insightful answers to diverse problem questions. The application of SQL functionalities such as JOINs, subqueries, and data modification statements has enabled a comprehensive exploration of patient data, showcasing proficiency in handling real-world analytical challenges. Through this project, I've not only demonstrated technical prowess but also a strategic approach to database design and optimization, emphasizing the value of effective SQL usage in extracting meaningful insights from complex healthcare datasets.

Thank you

Muskan Kashyap

- Data Analyst Intern at Psyliq