



DIABETES PREDICTION -SQL

HIRAN JOSEPH

DIABETES PREDICTION OVERVIEW

Number of Records:

The dataset consists of 100,001 records, providing a substantial volume of data for analysis.

Main Attributes:

The primary attributes include EmployeeName, Patient_id, gender, age, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, and diabetes..

Significance of Attributes:

Each attribute plays a crucial role in understanding diabetes-related factors, offering insights into patient demographics, health conditions, and lifestyle choices.

1. RETRIEVE THE PATIENT_ID AND AGES OF ALL PATIENTS.

The screenshot shows a database interface with the following details:

- Navigator:** Shows the schema structure. Under the **psliq** database, there is a **Tables** folder containing **diabetes_prediction**.
- Query 1:** The query is:

```
1 •   SELECT * FROM psliq.diabetes_prediction_csv;
2 •   select Patient_id,age
3     from psliq.diabetes_prediction_csv;
```
- Result Grid:** Displays the results of the query. The columns are **Patient_id** and **age**. The data is as follows:

Patient_id	age
PT101	80
PT102	54
PT103	28
PT104	36
PT105	76

- Output:** Shows the execution details:
 - Action Output: # 1 Time 01:52:50 Action select Patient_id,age from psliq.diabetes_prediction_csv
 - Message: 23977 row(s) returned
 - Duration / Fetch: 0.000 sec / 0.016

2. SELECT ALL FEMALE PATIENTS WHO ARE OLDER THAN 40.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `psliq`, which contains tables like `automobile`, `netflix`, `project_2`, and `diabetes_prediction_csv`.
- Query Editor:** The query `SELECT * FROM psliq.diabetes_prediction_csv;` has been modified to filter female patients older than 40. The final query is:

```
1 •  SELECT * FROM psliq.diabetes_prediction_csv;
2 •  select Patient_id,age
3   from psliq.diabetes_prediction_csv;
4 •  SELECT *
5   FROM psliq.diabetes_prediction_csv
6 WHERE gender = "female" AND age>40;
```
- Result Grid:** The result grid displays the following data:

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
NATHANIEL FORD	PT101	Female	80	0	1	never	25.19	6.6	140	0
GARY JIMENEZ	PT102	Female	54	0	0	No Info	27.32	6.6	80	0
ALSON LEE	PT107	Female	44	0	0	never	19.31	6.5	200	1
DAVID KUSHNER	PT108	Female	79	0	0	No Info	23.86	5.7	85	0
ARTHUR KENNEY	PT111	Female	53	0	0	never	27.32	6.1	85	0

- Output Log:** The log shows the execution of the query and its results.

#	Time	Action	Message	Duration / Fetch
1	02:20:38	SELECT * FROM psliq.diabetes_prediction_csv WHERE gender = "female" AND age>40	7507 row(s) returned	0.016 sec / 0.015 sec

3. CALCULATE THE AVERAGE BMI OF PATIENTS.

The screenshot shows a database management interface with the following details:

- Navigator:** Shows the schema structure with Schemas like automobile, netflix, project_2, and psliq, and Tables like diabetes_prediction_csv, new_table, Views, Stored Procedures, and Functions.
- Query 1:** The current query window titled "diabetes_prediction_csv" contains the following SQL code:

```
7 •  SELECT avg(bmi) as Average_bmi
8   FROM psliq.diabetes_prediction_csv;
9
10
11
12
```
- Result Grid:** Displays the result of the query:

Average_bmi
27.36402927805647
- Result 7:** The history window shows two actions:

#	Time	Action	Message	Duration / Fetch
1	02:20:38	SELECT * FROM psliq.diabetes_prediction_csv WHERE gender = "female" AND age>40	7507 row(s) returned	0.016 sec / 0.015 sec
2	02:36:36	SELECT avg(bmi) as Average_bmi FROM psliq.diabetes_prediction_csv	1 row(s) returned	0.000 sec / 0.000 sec
- Information:** Details about the "diabetes_prediction" table, including columns: EmployeeName, Patient_id, gender, age, hypertension, heart_disease.
- Object Info:** Tab for viewing object information.
- Session:** Tab for viewing session details.

4. LIST PATIENTS IN DESCENDING ORDER OF BLOOD GLUCOSE LEVELS.

The screenshot shows a database query interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq (selected), tables, views, stored procedures, functions.
- Query 1:** diabetes_prediction_csv
- SQL Query:**

```
7 •    SELECT avg(bmi) as Average_bmi
8     FROM psliq.diabetes_prediction_csv;
9
10 •   SELECT *
11     FROM psliq.diabetes_prediction_csv
12    ORDER BY blood_glucose_level DESC;
```
- Result Grid:** Displays patient data with columns: EmployeeName, Patient_id, gender, age, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes. The data includes:

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
LANIKA PRESTON	PT21650	Female	62	1	0	never	53.02	6.6	300	1
DEE DEE TYSON	PT21704	Female	67	1	0	never	29.2	5.8	300	1
CHARLOTTE CUI	PT21731	Male	79	0	1	No Info	27.32	7	300	1
NELLY TAN	PT21897	Female	29	0	0	ever	40.62	6.6	300	1
MARIO ZEPEDA	PT22372	Female	80	0	0	never	27.32	7	300	1
- Information:** diabetes_prediction_csv 9
- Output:** Action Output, showing the query execution details:

#	Time	Action	Message	Duration / Fetch
1	02:39:43	SELECT * FROM psliq.diabetes_prediction_csv ORDER BY blood_glucose_level DESC	23977 row(s) returned	0.031 sec / 0.047 sec

5. FIND PATIENTS WHO HAVE HYPERTENSION AND DIABETES.

The screenshot shows a database interface with the following components:

- Navigator:** On the left, it displays the schema structure. Under the 'psliq' schema, there is a 'Tables' section containing 'diabetes_prediction' and 'new_table'.
- Query Editor:** The main area shows a query in the 'Query 1' tab:

```
16
17 •  SELECT *
18   FROM psliq.diabetes_prediction_csv
19  WHERE hypertension=1 AND diabetes=1;
20
21
```
- Result Grid:** Below the query editor, the results are displayed in a grid format. The columns are: EmployeeName, Patient_id, gender, age, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, and diabetes. The data includes:JONES WONG PT139 Male 50 1 0 current 27.32 5.7 260 1
PATRIC STEELE PT205 Female 80 1 0 never 27.32 6.8 280 1
ARTHUR STELLINI PT343 Male 57 1 1 not current 27.77 6.6 160 1
CHAD LAW PT355 Male 63 1 0 ever 35.06 5.8 200 1
CATHERINE JAMES PT451 Female 52 1 0 never 50.3 6.6 155 1
- Information Panel:** On the right side, there is an 'Information' panel for the 'diabetes_prediction_csv' table, showing columns: EmployeeName, Patient_id, gender, age, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, and diabetes.
- Output Panel:** At the bottom, the 'Output' panel shows the execution history with two entries:# Time Action Message Duration / Fetch
1 02:39:43 SELECT * FROM psliq.diabetes_prediction_csv ORDER BY blood_glucose_level DESC 23977 row(s) returned 0.031 sec / 0.047 sec
2 02:41:56 SELECT * FROM psliq.diabetes_prediction_csv WHERE hypertension=1 AND diabetes=1 501 row(s) returned 0.016 sec / 0.000 sec

6. DETERMINE THE NUMBER OF PATIENTS WITH HEART DISEASE.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the **SCHEMAS** tree with `automobile`, `netflix`, `project_2`, and `psliq`. Under `psliq`, there are `Tables` (`diabetes_prediction_csv`, `new_table`), `Views`, `Stored Procedures`, and `Functions`.
- Query Editor:** Tab `Query 1` contains the following SQL code:

```
22
23
24 •  SELECT count(*)  as Heart_disease_patients
25   FROM psliq.diabetes_prediction_csv
26  WHERE heart_disease=1;
27
```
- Result Grid:** Shows the result of the query: `Heart_disease_patients` with value `932`.
- Result History:** Tab `Result 12` shows the history of actions taken:| # | Time | Action | Message | Duration / Fetch |
| --- | --- | --- | --- | --- |
| 1 | 02:39:43 | SELECT * FROM psliq.diabetes_prediction_csv ORDER BY blood_glucose_level DESC | 23977 row(s) returned | 0.031 sec / 0.047 sec |
| 2 | 02:41:56 | SELECT * FROM psliq.diabetes_prediction_csv WHERE hypertension=1 AND diabetes=1 | 501 row(s) returned | 0.016 sec / 0.000 sec |
| 3 | 02:44:47 | SELECT count(*) as Heart_disease_patients FROM psliq.diabetes_prediction_csv WHE... | 1 row(s) returned | 0.016 sec / 0.000 sec |
| 4 | 02:45:40 | SELECT count(*) as Heart_disease_patients FROM psliq.new_table WHERE heart_dise... | Error Code: 1054. Unknown column 'heart_disease' in 'where clause' | 0.000 sec |
| 5 | 02:45:53 | SELECT * FROM psliq.new_table | 0 row(s) returned | 0.000 sec / 0.000 sec |
| 6 | 02:46:17 | SELECT count(*) as Heart_disease_patients FROM psliq.diabetes_prediction_csv WHE... | 1 row(s) returned | 0.000 sec / 0.000 sec |

7. GROUP PATIENTS BY SMOKING HISTORY AND COUNT HOW MANY SMOKERS AND NON SMOKERS THERE ARE.

The screenshot shows the MySQL Workbench interface with the following details:

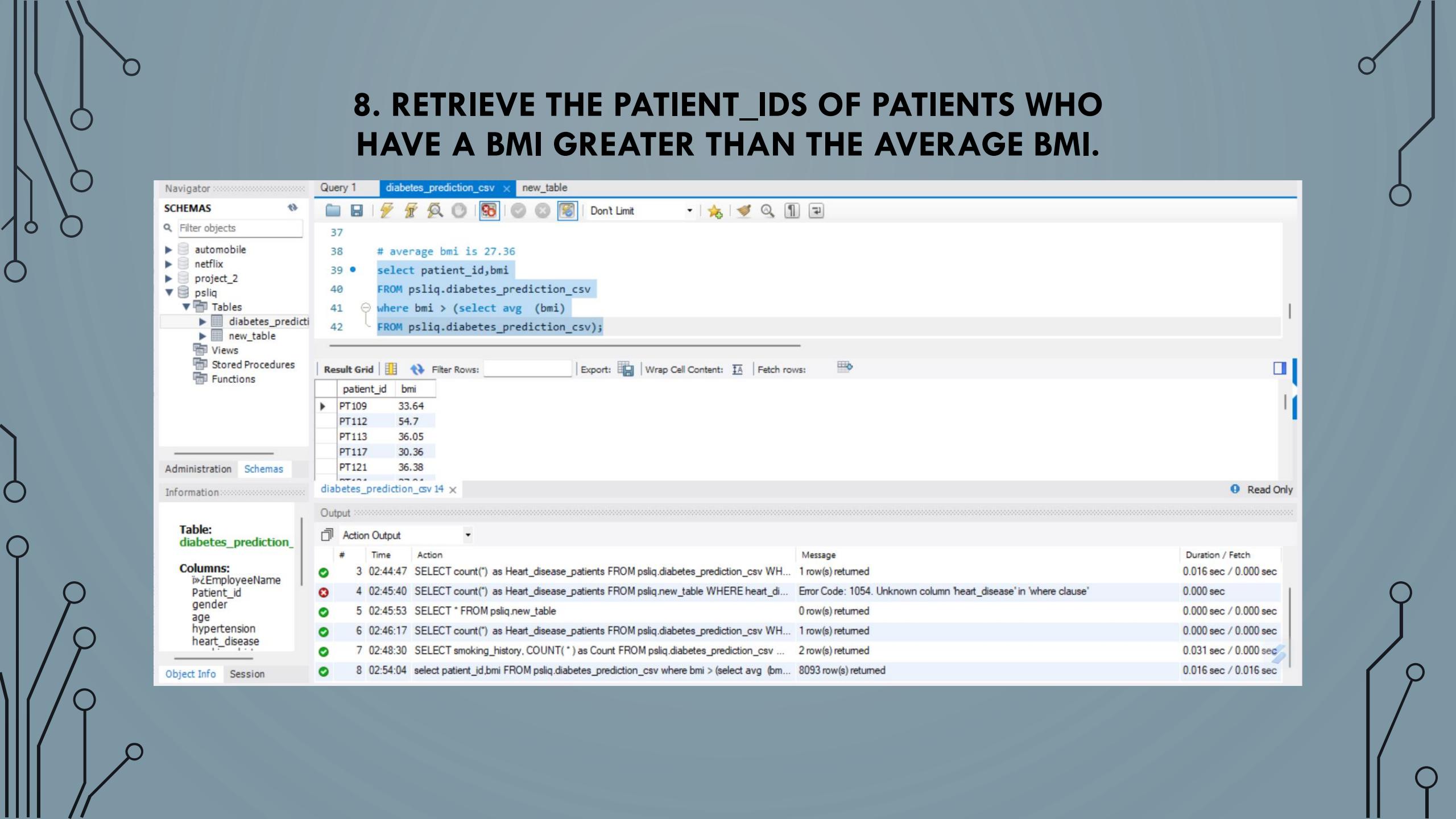
- Schemas:** automobile, netflix, project_2, psliq (selected), diabetes_prediction_csv, new_table, Views, Stored Procedures, Functions.
- Query Editor:** Contains the following SQL code:

```
30
31 • SELECT smoking_history, COUNT( * ) as Count
32   FROM psliq.diabetes_prediction_csv
33   WHERE smoking_history = "current" OR smoking_history="never"
34   GROUP BY smoking_history;
35
```
- Result Grid:** Displays the results of the query:

smoking_history	Count
never	8534
current	2228
- Information Panel:** Shows the **Table: diabetes_prediction_** with columns: EmployeeName, patient_id, gender, age, hypertension, heart_disease.
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
2	02:41:56	SELECT * FROM psliq.diabetes_prediction_csv WHERE hypertension=1 AND diabete...	501 row(s) returned	0.016 sec / 0.000 sec
3	02:44:47	SELECT count(*) as Heart_disease_patients FROM psliq.diabetes_prediction_csv WH...	1 row(s) returned	0.016 sec / 0.000 sec
4	02:45:40	SELECT count(*) as Heart_disease_patients FROM psliq.new_table WHERE heart_di...	Error Code: 1054. Unknown column 'heart_disease' in 'where clause'	0.000 sec
5	02:45:53	SELECT * FROM psliq.new_table	0 row(s) returned	0.000 sec / 0.000 sec
6	02:46:17	SELECT count(*) as Heart_disease_patients FROM psliq.diabetes_prediction_csv WH...	1 row(s) returned	0.000 sec / 0.000 sec
7	02:48:30	SELECT smoking_history,COUNT(*) as Count FROM psliq.diabetes_prediction_csv ...	2 row(s) returned	0.031 sec / 0.000 sec

8. RETRIEVE THE PATIENT_IDS OF PATIENTS WHO HAVE A BMI GREATER THAN THE AVERAGE BMI.



The screenshot shows a database query interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq (selected).
 - Tables:** diabetes_prediction_csv, new_table (selected).
 - Views, Stored Procedures, Functions:** None.
- Query 1:** A SQL query is being run:

```
37
38  # average bmi is 27.36
39 • select patient_id,bmi
40   FROM psliq.diabetes_prediction_csv
41   where bmi > (select avg (bmi)
42     FROM psliq.diabetes_prediction_csv);
```
- Result Grid:** The results show patient IDs and their corresponding BMI values.

patient_id	bmi
PT109	33.64
PT112	54.7
PT113	36.05
PT117	30.36
PT121	36.38
PT124	37.01
- Information:** Shows the table structure for **diabetes_prediction_csv**.

EmployeeName	Patient_id	gender	age	hypertension	heart_disease
PT109	PT112	PT113	PT117	PT121	PT124
- Action Output:** A log of recent actions with their status, time, message, and duration.

#	Time	Action	Message	Duration / Fetch
3	02:44:47	SELECT count(*) as Heart_disease_patients FROM psliq.diabetes_prediction_csv WHERE heart_disease = 1;	1 row(s) returned	0.016 sec / 0.000 sec
4	02:45:40	SELECT count(*) as Heart_disease_patients FROM psliq.new_table WHERE heart_disease = 1;	Error Code: 1054. Unknown column 'heart_disease' in 'where clause'	0.000 sec
5	02:45:53	SELECT * FROM psliq.new_table;	0 row(s) returned	0.000 sec / 0.000 sec
6	02:46:17	SELECT count(*) as Heart_disease_patients FROM psliq.diabetes_prediction_csv WHERE heart_disease = 1;	1 row(s) returned	0.000 sec / 0.000 sec
7	02:48:30	SELECT smoking_history, COUNT(*) as Count FROM psliq.diabetes_prediction_csv WHERE smoking_history = 1;	2 row(s) returned	0.031 sec / 0.000 sec
8	02:54:04	select patient_id,bmi FROM psliq.diabetes_prediction_csv where bmi > (select avg (bmi) FROM psliq.diabetes_prediction_csv);	8093 row(s) returned	0.016 sec / 0.016 sec

9. FIND THE PATIENT WITH THE HIGHEST HBA1C LEVEL AND THE PATIENT WITH THE LOWEST HBA1C LEVEL.

```
46  
47 • select i»;EmployeeName,patient_id,HbA1c_level as Max_HbA1c_level  
48     FROM psliq.diabetes_prediction_csv  
49     order by HbA1c_level Desc  
50     limit 1;  
51
```

Result Grid		
i»;EmployeeName	patient_id	Max_HbA1c_level
IVETTE RODRIGUEZ	PT23275	9

diabetes_prediction_csv 16 x

```
54  
55 • select i»;EmployeeName,patient_id,HbA1c_level as Max_HbA1c_level  
56     FROM psliq.diabetes_prediction_csv  
57     order by HbA1c_level asc  
58     limit 1;  
59
```

Result Grid		
i»;EmployeeName	patient_id	Max_HbA1c_level
PARKMAN LEW	PT21165	3.5

diabetes_prediction_csv 17 x

10. CALCULATE THE AGE OF PATIENTS IN YEARS (ASSUMING THE CURRENT DATE AS OF NOW).

The screenshot shows a database query interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq (selected), new_table, Views, Stored Procedures, Functions.
- Query 1:** A SQL script named "diabetes_prediction_csv" is being run. The script contains:

```
61
62
63 •  select i»_EmployeeName,patient_id,
64      year(now()) - age as birth_year,
65      year(now()) - year(now()) + age As Current_age
66  FROM psliq.diabetes_prediction_csv
```
- Result Grid:** Displays the results of the query in a tabular format. The columns are EmployeeName, patient_id, birth_year, and Current_age. The data is as follows:

EmployeeName	patient_id	birth_year	Current_age
NATHANIEL FORD	PT101	1943	80
GARY JIMENEZ	PT102	1969	54
ALBERT PARDINI	PT103	1995	28
CHRISTOPHER CHONG	PT104	1987	36
PATRICK GARDNER	PT105	1947	76
DAVID CHILTON	PT106	2002	22

Result 18 X Read Only
- Table:** diabetes_prediction_ (highlighted in green).
Columns: EmployeeName, patient_id, gender, age, hypertension, heart_disease.
- Action Output:** Shows the history of actions taken during the session, including errors and successful queries. The last successful query was executed at 03:08:21 and returned 23977 rows.

11. RANK PATIENTS BY BLOOD GLUCOSE LEVEL WITHIN EACH GENDER GROUP.

The screenshot shows a database query interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq (selected)
- Tables:** diabetes_prediction_csv, new_table
- Query 1:** A SQL query is displayed in the editor:

```
71
72 •    select Patient_id,gender,blood_glucose_level,
73         rank () over (partition by gender order by blood_glucose_level) as blood_glucose_level_ranks_by_gender
74     FROM psliq.diabetes_prediction_csv;
75
76
```
- Result Grid:** The results of the query are shown in a grid format:

Patient_id	gender	blood_glucose_level	blood_glucose_level_ranks_by_gender
PT22446	Female	80	1
PT22313	Female	80	1
PT22614	Female	80	1
PT23032	Female	80	1
PT22684	Female	80	1
PT23246	Female	80	1
- Output:** The output panel shows the query execution details:
 - Action Output: 1 03:17:23 select Patient_id,gender,blood_glucose_level, rank () over (partition by gender order by bl... 23977 row(s) returned
 - Message: Duration / Fetch 0.016 sec / 0.016 sec

12. UPDATE THE SMOKING HISTORY OF PATIENTS WHO ARE OLDER THAN 50 TO "EX-SMOKER."

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema tree with **SCHEMAS** (automobile, netflix, project_2, psliq), **Tables** (diabetes_prediction_csv, new_table), and **Views, Stored Procedures, Functions**.
- Query Editor (Query 1):** Contains the following SQL code:

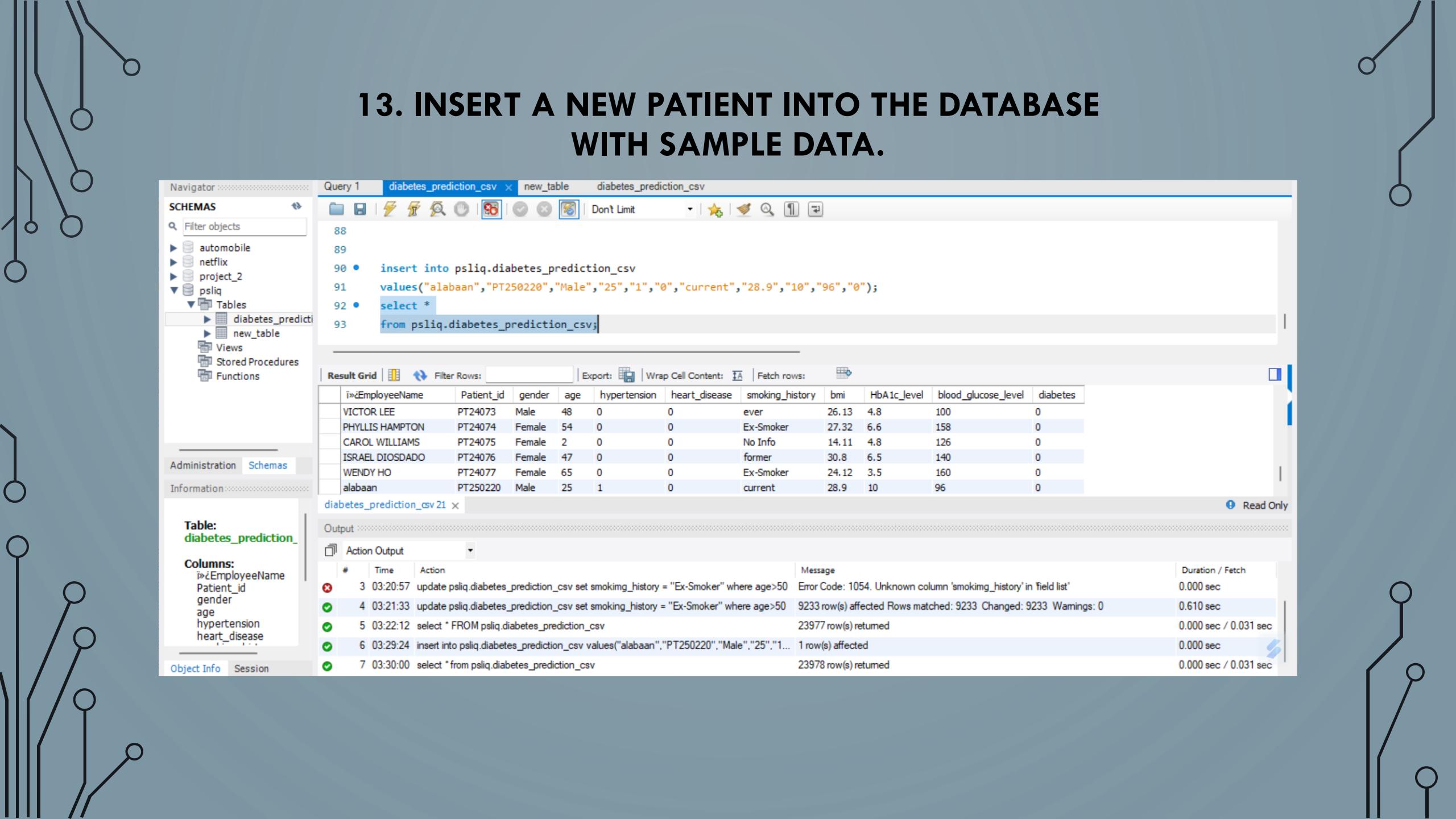
```
80 •  set sql_safe_updates = 0;
81 •  update psliq.diabetes_prediction_csv
82   set smoking_history = "Ex-Smoker"
83   where age>50;
84
85 •  select * FROM psliq.diabetes_prediction_csv;
```
- Result Grid:** Displays the contents of the **diabetes_prediction_csv** table with the following data:

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
NATHANIEL FORD	PT101	Female	80	0	1	Ex-Smoker	25.19	6.6	140	0
GARY JIMENEZ	PT102	Female	54	0	0	Ex-Smoker	27.32	6.6	80	0
ALBERT PARDINI	PT103	Male	28	0	0	never	27.32	5.7	158	0
CHRISTOPHER CHONG	PT104	Female	36	0	0	current	23.45	5	155	0
PATRICK GARDNER	PT105	Male	76	1	1	Ex-Smoker	20.14	4.8	155	0
DAVID SULLIVAN	PT106	Female	20	0	0	never	27.32	6.6	85	0

- Output:** Shows the execution log with the following entries:

#	Action	Message	Duration / Fetch
1	select Patient_id,gender,blood_glucose_level, rank () over (partition by gender order by bl... 23977 row(s) returned	23977 row(s) returned	0.016 sec / 0.016 sec
2	set sql_safe_updates = 0	0 row(s) affected	0.016 sec
3	update psliq.diabetes_prediction_csv set smoking_history = "Ex-Smoker" where age>50	Error Code: 1054. Unknown column 'smoking_history' in field list'	0.000 sec
4	update psliq.diabetes_prediction_csv set smoking_history = "Ex-Smoker" where age>50	9233 row(s) affected Rows matched: 9233 Changed: 9233 Warnings: 0	0.610 sec
5	select * FROM psliq.diabetes_prediction_csv	23977 row(s) returned	0.000 sec / 0.031 sec

13. INSERT A NEW PATIENT INTO THE DATABASE WITH SAMPLE DATA.



The screenshot shows a database management interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq
- Tables:** diabetes_prediction_csv, new_table
- Query Editor:** Contains the following SQL code:

```
88
89
90 • insert into psliq.diabetes_prediction_csv
91   values("alabaan","PT250220","Male","25","1","0","current","28.9","10","96","0");
92 • select *
93   from psliq.diabetes_prediction_csv;
```
- Result Grid:** Displays the contents of the diabetes_prediction_csv table. The columns are EmployeeName, Patient_id, gender, age, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, and diabetes. The data includes rows for VICTOR LEE, PHYLLIS HAMPTON, CAROL WILLIAMS, ISRAEL DIOSDADO, WENDY HO, and a new entry for alabaan.
- Table Information:** Shows the table structure for diabetes_prediction_csv with columns EmployeeName, Patient_id, gender, age, hypertension, heart_disease, and smoking_history.
- Action Output:** Logs the execution of the SQL statements. Statement 3 fails due to an unknown column 'smoking_history'. Statements 4, 5, and 6 succeed. Statement 7 returns 23978 rows.

14. DELETE ALL PATIENTS WITH HEART DISEASE FROM THE DATABASE.



The screenshot shows a database management interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq (selected).
 - Tables:** diabetes_prediction (selected), new_table.
- Query 1:** Contains the following SQL code:

```
132
133
134 • delete from psliq.diabetes_prediction_csv
135   where heart_disease = 1;
136
137 • select *
138   FROM psliq.diabetes_prediction_csv;
139
140
```
- Result Grid:** Shows the structure and data of the diabetes_prediction table.

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
GARY JIMENEZ	PT102	Female	54	0	0	Ex-Smoker	27.32	6.6	80	0
ALBERT PARDINI	PT103	Male	28	0	0	never	27.32	5.7	158	0
CHRISTOPHER CHONG	PT104	Female	36	0	0	current	23.45	5	155	0
DAVID SULLIVAN	PT106	Female	20	0	0	never	27.32	6.6	85	0
ALSON LEE	PT107	Female	44	0	0	never	19.31	6.5	200	1
- Output:** Displays the results of the executed actions.

#	Time	Action	Message	Duration / Fetch
4	04:00:54	delete from psliq.diabetes_prediction_csv where heart_disease = 1	932 row(s) affected	0.031 sec
5	04:01:17	select * FROM psliq.diabetes_prediction_csv	23046 row(s) returned	0.000 sec / 0.031 sec

15. FIND PATIENTS WHO HAVE HYPERTENSION BUT NOT DIABETES USING THE EXCEPT OPERATOR.

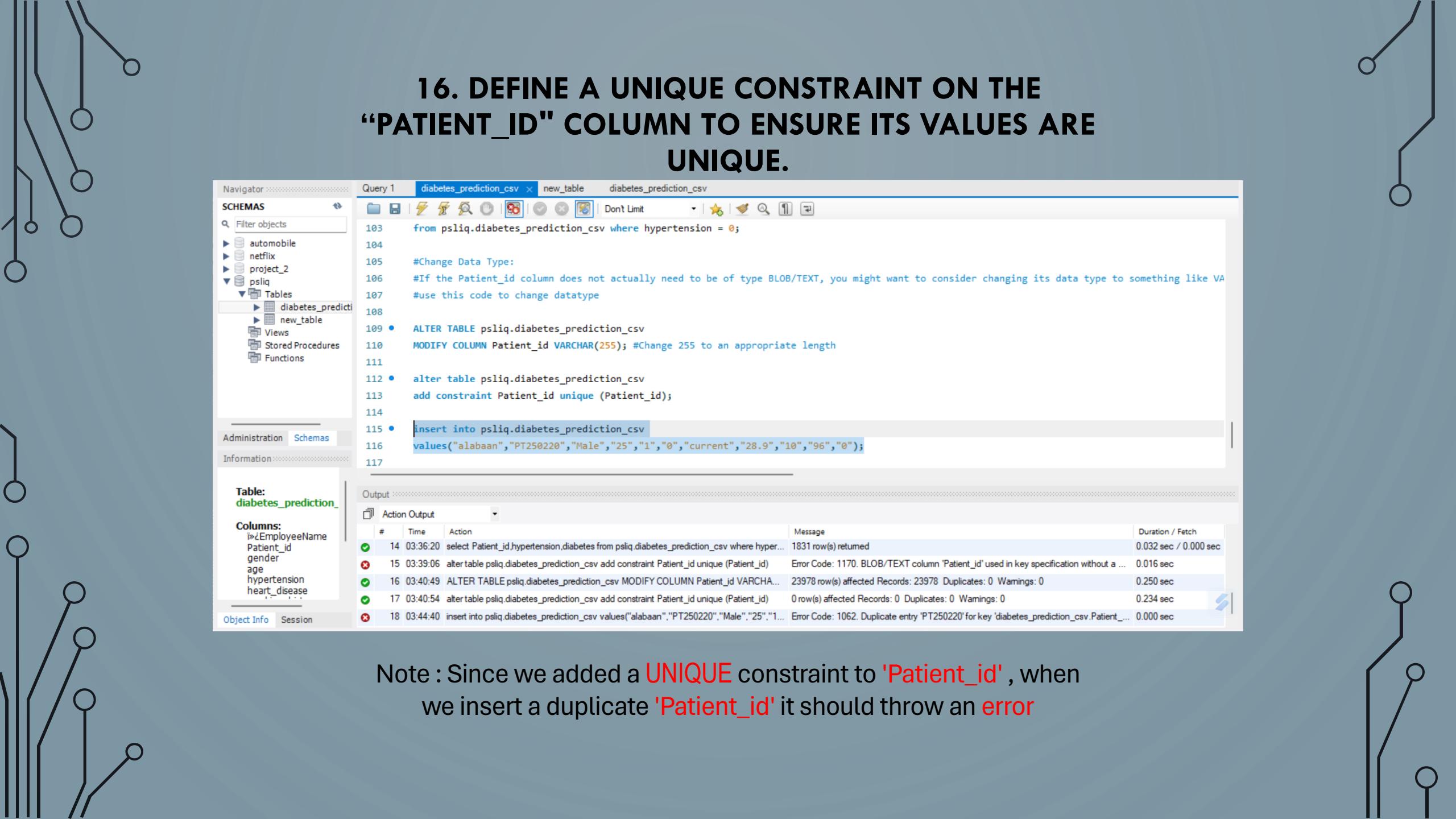
The screenshot shows a database management interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq (selected), tables, views, stored procedures, functions.
- Table Information:** Table: diabetes_prediction, Columns: EmployeeName, Patient_id, gender, age, hypertension, heart_disease.
- Query Editor:** Query 1 (diabetes_prediction_csv) contains the following T-SQL code:

```
98
99 •    select Patient_id,hypertension,diabetes
100   from psliq.diabetes_prediction_csv where hypertension = 1
101 ✘ except
102   select Patient_id,hypertension,diabetes
103   from psliq.diabetes_prediction_csv where hypertension = 0;
```
- Result Grid:** Shows a table with columns Patient_id, hypertension, and diabetes, containing the following data:

Patient_id	hypertension	diabetes
PT105	1	0
PT129	1	0
PT139	1	1
PT143	1	0
PT155	1	0
PT161	1	0
- Action Output:** Shows the history of actions taken on the query, including the execution of each part of the query and the resulting message and duration.

16. DEFINE A UNIQUE CONSTRAINT ON THE "PATIENT_ID" COLUMN TO ENSURE ITS VALUES ARE UNIQUE.



The screenshot shows a database management interface with the following details:

- Schemas:** automobile, netflix, project_2, psliq (selected).
 - Tables:** diabetes_prediction_csv, new_table, Views, Stored Procedures, Functions.
- Query Editor:** Query 1, showing T-SQL code.

```
103 from psliq.diabetes_prediction_csv where hypertension = 0;
104
105 #Change Data Type:
106 #If the Patient_id column does not actually need to be of type BLOB/TEXT, you might want to consider changing its data type to something like VARCHAR
107 #use this code to change datatype
108
109 • ALTER TABLE psliq.diabetes_prediction_csv
110     MODIFY COLUMN Patient_id VARCHAR(255); #Change 255 to an appropriate length
111
112 • alter table psliq.diabetes_prediction_csv
113     add constraint Patient_id unique (Patient_id);
114
115 • insert into psliq.diabetes_prediction_csv
116     values("alabaan","PT250220","Male","25","1","0","current","28.9","10","96","0");
117
```
- Output:** Action Output table showing the execution of the statements.

#	Time	Action	Message	Duration / Fetch
14	03:36:20	select Patient_id,hypertension,diabetes from psliq.diabetes_prediction_csv where hyper...	1831 row(s) returned	0.032 sec / 0.000 sec
15	03:39:06	alter table psliq.diabetes_prediction_csv add constraint Patient_id unique (Patient_id)	Error Code: 1170. BLOB/TEXT column 'Patient_id' used in key specification without a ...	0.016 sec
16	03:40:49	ALTER TABLE psliq.diabetes_prediction_csv MODIFY COLUMN Patient_id VARCHAR...	23978 row(s) affected Records: 23978 Duplicates: 0 Warnings: 0	0.250 sec
17	03:40:54	alter table psliq.diabetes_prediction_csv add constraint Patient_id unique (Patient_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.234 sec
18	03:44:40	insert into psliq.diabetes_prediction_csv values("alabaan","PT250220","Male","25","1..."	Error Code: 1062. Duplicate entry 'PT250220' for key 'diabetes_prediction_csv.Patient_...	0.000 sec
- Table Information:** diabetes_prediction_csv, Columns: EmployeeName, Patient_id, gender, age, hypertension, heart_disease.

Note : Since we added a **UNIQUE** constraint to '**Patient_id**' , when we insert a duplicate '**Patient_id**' it should throw an **error**

17. CREATE A VIEW THAT DISPLAYS THE PATIENT_IDS, AGES, AND BMI OF PATIENTS.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure under the **psliq** database, specifically the **diabetes_prediction** table.
- Query Editor (Query 1):** Contains the SQL code for creating a view:

```
119 • USE psliq;
120
121 • CREATE VIEW Patient_Data AS
122     SELECT Patient_id, age, bmi
123     FROM psliq.diabetes_prediction_csv;
124
125 • select *
126     from Patient_Data;
127
128 • DROP VIEW IF EXISTS Patient_Data;
```
- Result Grid:** Displays the data from the **Patient_Data** view, which is a subset of the **diabetes_prediction** table.

Patient_id	age	bmi
PT101	80	25.19
PT102	54	27.32
PT103	28	27.32
PT104	36	23.45
PT105	76	20.14
- Object Info:** Shows the columns of the **diabetes_prediction** table.
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
2	03:58:09	CREATE VIEW Patient_Data AS SELECT Patient_id, age, bmi FROM psliq.diabetes_p...	0 row(s) affected	0.000 sec
3	03:58:13	select * from Patient_Data	23978 row(s) returned	0.000 sec / 0.016 sec

18. SUGGEST IMPROVEMENTS IN THE DATABASE SCHEMA TO REDUCE DATA REDUNDANCY AND IMPROVE DATA INTEGRITY.

- **Foreign Key Relationships:**
 - Establish and enforce foreign key relationships between tables to maintain referential integrity.
 - This helps prevent orphaned records and ensures consistency in the data.
- Break down large tables into smaller, related tables through normalization to reduce redundancy.
- Ensure that each table has a primary key to uniquely identify records.
 - **Normalization:**
 - **Use of Unique Constraints:**
 - Implement unique constraints on columns where applicable to prevent duplicate values.
 - Utilize unique indexes to enforce uniqueness efficiently.
 - **Data Types and Sizes Optimization:**
 - Optimize data types and sizes for columns to reduce storage requirements.
 - Avoid using excessively large data types if smaller ones are sufficient.
 - **Denormalization (where appropriate):**
 - Evaluate the need for denormalization for performance gains, considering read vs. write operations.
 - Use denormalization selectively, and document reasons for doing so

19. EXPLAIN HOW YOU CAN OPTIMIZE THE PERFORMANCE OF SQL QUERIES ON THIS DATASET.

- **Use Indexing:**
 - Identify and create indexes on columns frequently used in WHERE clauses, JOIN conditions, and ORDER BY clauses.
 - However, be cautious not to over-index, as it can lead to overhead during data modification operations.
- **Optimize SELECT Queries:**
 - Retrieve only the necessary columns rather than using SELECT *.
 - Avoid using SELECT DISTINCT unless absolutely necessary, as it can be resource-intensive.
- **JOIN Optimization:**
 - Use INNER JOINS instead of OUTER JOINS whenever possible, as INNER JOINS are generally more efficient.
 - Consider using appropriate join algorithms (e.g., HASH JOIN, MERGE JOIN) based on the characteristics of your data.
- **Subquery Optimization:**
 - Refactor subqueries to JOINS where feasible, as JOINS are often more performant than subqueries.
 - Use EXISTS or IN clauses judiciously, as they can impact performance.

- **Avoid SELECT * in Production:**
 - Explicitly list the columns you need in the SELECT statement instead of using SELECT *.
 - This reduces the amount of data transferred and improves query performance.
- **Partitioning Tables:**
 - Consider partitioning large tables based on a column that is frequently used in queries.
 - This can significantly enhance query performance, especially in scenarios where only a subset of data is needed.
- **Optimize WHERE Clauses:**
 - Be mindful of the conditions in the WHERE clause and ensure they can leverage existing indexes.
 - Use appropriate comparison operators and avoid unnecessary functions in the WHERE clause.
- **Database Configuration:**
 - Configure database server settings such as memory allocation, buffer pool size, and parallelism to align with the specific requirements of your workload.
- **Query Plan Analysis:**
 - Analyze query execution plans using tools like EXPLAIN (in MySQL) to understand how the database engine is processing the query.
 - Optimize queries based on the information provided by the query plan.

THANK YOU