

DIABETES PREDICTION DATASET

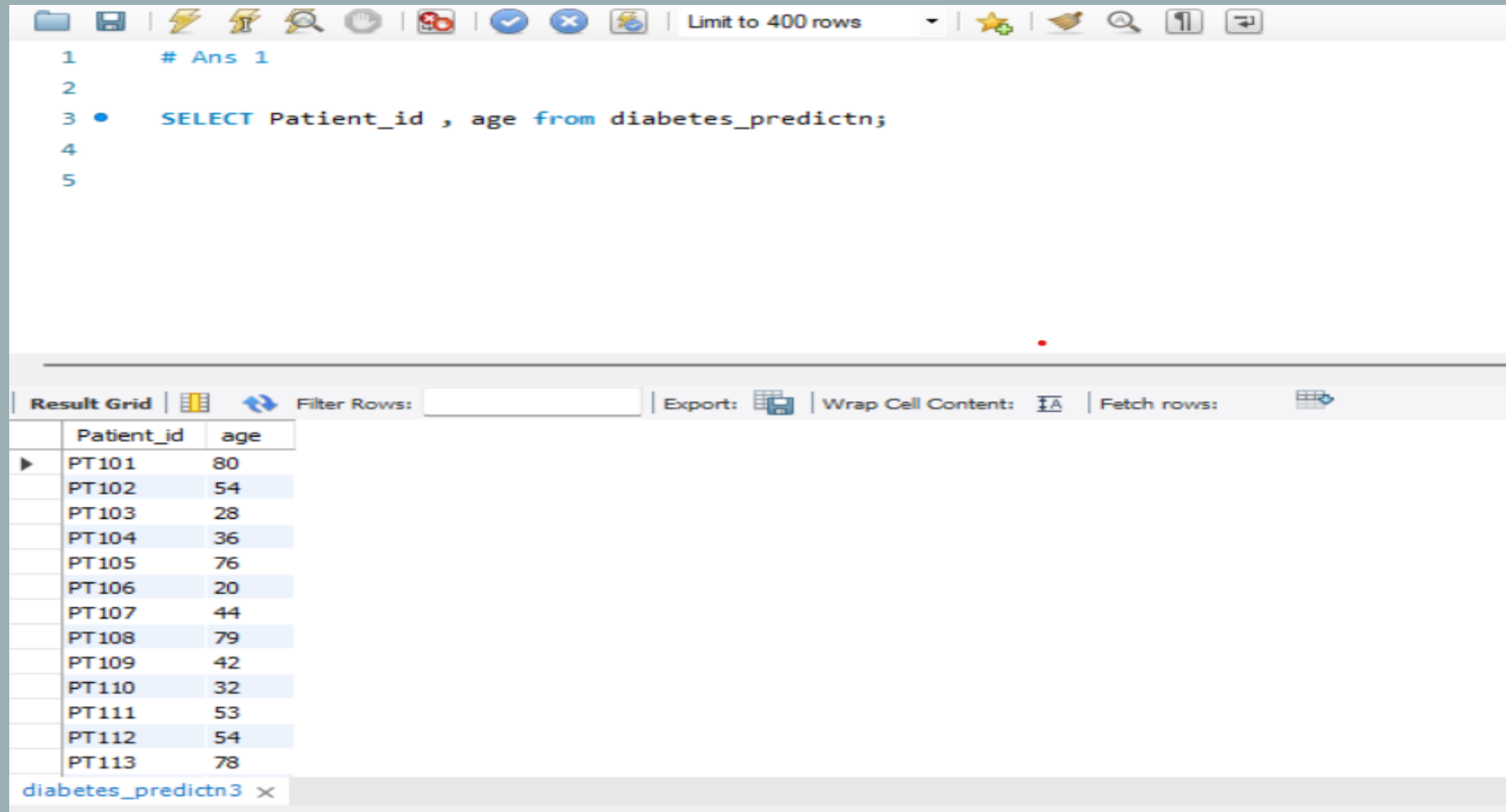
BY
MRINAL DESHMUKH



Basic Introduction

- **Now a day Diabetes continues to be a significant global health challenge affecting millions of people worldwide .**
- **Diabetes is a major cause of blindness, kidney failure, heart attacks, in this dataset we are exploring Diabetes Prediction , Dataset serves as a valuable tool for understanding and predicting the development of diabetes**
- **The Diabetes Prediction Dataset is a collection of data for predicting the likelihood of an individual developing diabetes based on various attributes such as age, BMI (Body Mass Index), glucose levels, blood pressure, and other health-related features.**
- **The dataset consist of 1 Lac record and 11 attribute**

1. Retrieve the Patient_id and ages of all patients.



The screenshot shows a database query editor interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 400 rows' dropdown. The query editor contains the following SQL query:

```
1 # Ans 1
2
3 • SELECT Patient_id , age from diabetes_predictn;
4
5
```

Below the query editor is the 'Result Grid' section, which displays the results of the query in a table format. The table has two columns: 'Patient_id' and 'age'. The results are as follows:

Patient_id	age
PT101	80
PT102	54
PT103	28
PT104	36
PT105	76
PT106	20
PT107	44
PT108	79
PT109	42
PT110	32
PT111	53
PT112	54
PT113	78

The bottom of the interface shows a tab labeled 'diabetes_predictn3' with a close button.

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

Limit to 400 rows

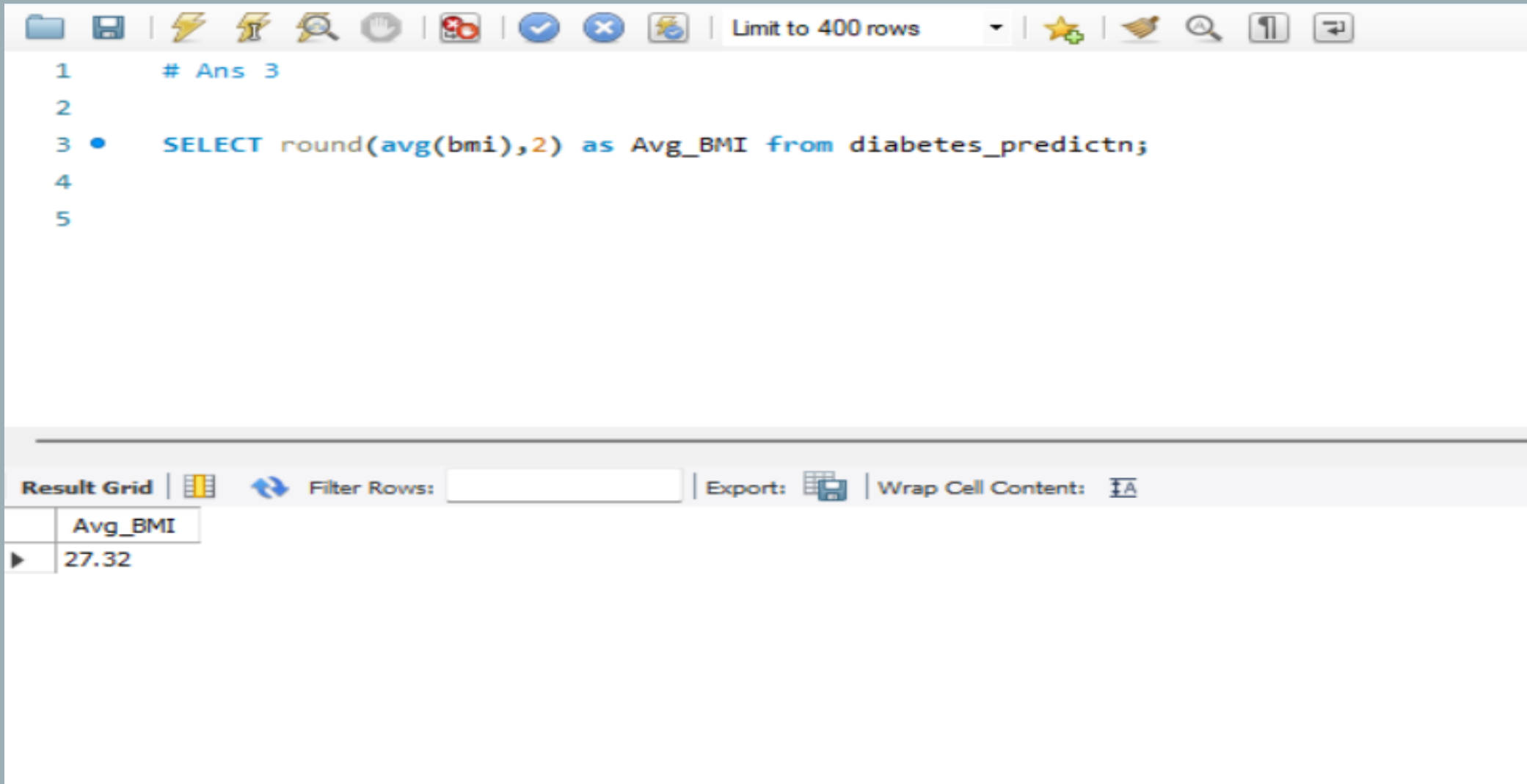
```
1 # 2.Ans
2
3 • select * from diabetes_predictn
4   where gender = "female" and age > 40 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

	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
	PATRICIA JACKSON	PT112	Female	54	0	0	former	54.7	6	100	0
	EDWARD HARRINGTON	PT113	Female	78	0	0	former	36.05	5	130	0
	JOHN MARTIN	PT114	Female	67	0	0	never	25.69	5.8	200	0
	DAVID FRANKLIN	PT115	Female	76	0	0	No Info	27.32	5	160	0
	SEBASTIAN WONG	PT118	Female	42	0	0	never	24.48	5.7	158	0
	MARTY ROSS	PT119	Female	42	0	0	No Info	27.32	5.7	80	0
	GEORGE GARCIA	PT123	Female	69	0	0	never	21.24	4.8	85	0
	VICTOR WYRSCH	PT124	Female	72	0	1	former	27.94	6.5	130	0

diabetes_predictn3 x

3. Calculate the average BMI of patients.



The screenshot shows a SQL query editor interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 400 rows' dropdown. The query text is as follows:

```
1      # Ans 3
2
3      •  SELECT round(avg(bmi),2) as Avg_BMI from diabetes_predictn;
4
5
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays a single row with the column header 'Avg_BMI' and the value '27.32'.

Avg_BMI
27.32

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

Limit to 400 rows

```
1 # Ans 4
2
3 • SELECT EmployeeName , patient_id , blood_glucose_level
4 FROM diabetes_predictn
5 ORDER BY blood_glucose_level DESC ;
6
7
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

	EmployeeName	patient_id	blood_glucose_level
▶	Gilbert J Fragoso	PT99638	300
	Amado A Lumas Jr	PT99663	300
	Shanice M Guidry	PT99672	300
	Angelica J Young	PT99764	300
	Flor D Roman	PT99809	300
	Clyde L Woods	PT99927	300
	Josephine C Cabrera	PT99968	300
	Marquis D Walker	PT100039	300
	Silvia Woo	PT91896	300
	Cliff E Bell	PT89546	300
	Sergey Trofimenko	PT89757	300
	Esther E Velonza	PT91743	300
	Brenda G Velasquez	PT89459	300

diabetes_predictn5 x

Output




5. Find patients who have hypertension and diabetes.

Limit to 400 rows

```
1 # Ans 5
2
3 • SELECT * FROM diabetes_predictn
4 WHERE hypertension = 1 and diabetes = 1 ;
5
6
```

Result Grid

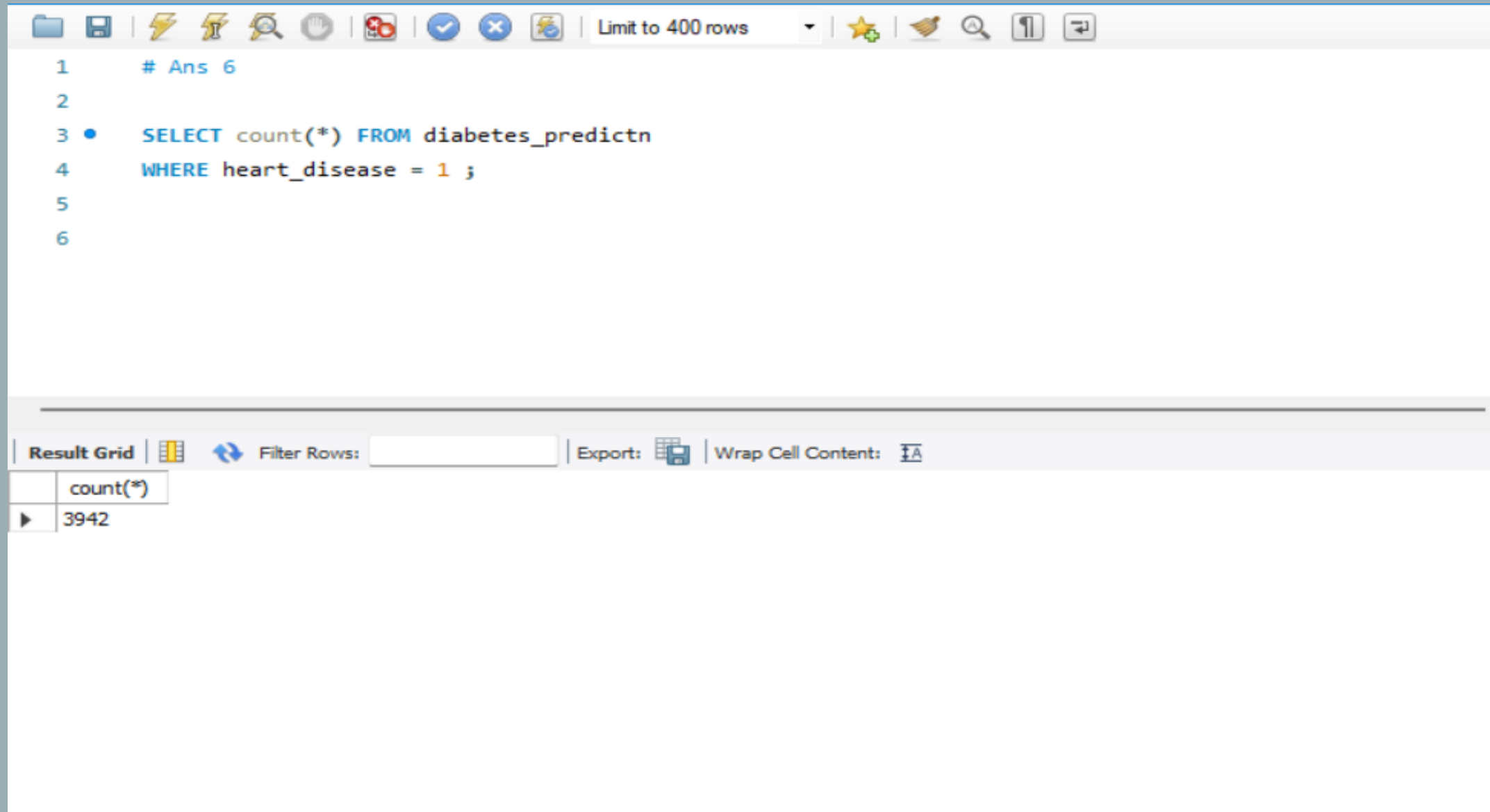
Filter Rows:

Export:  Wrap Cell Content:  Fetch rows: 

	EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
▶	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
	JOHN HART	PT565	Male	48	1	0	current	36.12	6.8	140	1
	JOHN BARKER	PT567	Female	79	1	0	former	27.32	6.5	159	1
	ROBERT BONNET	PT632	Female	49	1	0	not current	36.93	8.8	155	1
	VITANI BENJAMIN	PT727	Male	43	1	0	not current	40.86	6.6	159	1
	LANNIE ADELMAN	PT828	Female	38	1	0	not current	27.32	6.1	160	1
	JOEL DELIZONNA	PT852	Female	28	1	0	never	20.09	6.6	200	1
	KAREN KUBICK	PT861	Male	59	1	0	ever	25.94	9	140	1
	ANA GONZALEZ	PT983	Female	75	1	0	No Info	27.32	6.6	240	1

diabetes_predictn6 x

6. Determine the number of patients with heart disease.



The screenshot shows a SQL query editor interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 400 rows' dropdown. The query text is as follows:

```
1      # Ans 6
2
3      •  SELECT count(*) FROM diabetes_predictn
4         WHERE heart_disease = 1 ;
5
6
```

Below the query editor is a 'Result Grid' section. It contains a table with the following data:

	count(*)
▶	3942

The 'Result Grid' section also includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox.

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

1 # Ans 7.1
2
3 • SELECT smoking_history, count(*) AS COUNT_1
4 FROM diabetes_predictn
5 GROUP BY smoking_history;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

smoking_history	COUNT_1
never	35095
No Info	35816
current	9286
former	9352
ever	4004
not current	6447

Result 9 x

1 # Ans 7.2
2
3 • SELECT
4 CASE
5 WHEN smoking_history IN ('current', 'ever') THEN 'Smoker'
6 WHEN smoking_history = 'No Info' THEN 'Information NA'
7 ELSE 'Non-Smoker'
8 END AS smoker_status,
9 COUNT(*) AS count
10 FROM
11 diabetes_predictn
12 GROUP BY
13 CASE
14 WHEN smoking_history IN ('current', 'ever') THEN 'Smoker'
15 WHEN smoking_history = 'No Info' THEN 'Information NA'
16 ELSE 'Non-Smoker'
17 END;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

smoker_status	count
Non-Smoker	50894
Information NA	35816
Smoker	13290

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

SQL Query Editor Interface:

```
1 # Ans 8
2
3 • SELECT Patient_id
4 FROM diabetes_predictn
5 WHERE bmi > (SELECT AVG(bmi) FROM diabetes_predictn);
6
```

Result Grid:

Patient_id
PT109
PT112
PT113
PT117
PT121
PT124
PT126
PT128
PT131
PT140
PT143

diabetes_predictn11 x

SQL Query Editor Interface:

```
1 # Ans 8.2
2
3 • SELECT
4     dp.Patient_id,
5     dp.bmi,
6     av.avg_bmi
7 FROM
8     diabetes_predictn dp
9 JOIN (
10     SELECT
11         ROUND(AVG(bmi), 2) AS avg_bmi
12     FROM
13         diabetes_predictn
14 ) av ON dp.bmi > av.avg_bmi;
15
16
```

Result Grid:

Patient_id	bmi	avg_bmi
PT109	33.64	27.32
PT112	54.7	27.32
PT113	36.05	27.32
PT117	30.36	27.32
PT121	36.38	27.32

Result 12 x

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

Limit to 400 rows

1 # Ans 9

2

3 • SELECT *

4 FROM diabetes_predictn

5 ORDER BY HbA1c_level DESC

6 LIMIT 1; -- For highest HbA1c

7

8 • SELECT *

9 FROM diabetes_predictn

10 ORDER BY HbA1c_level ASC

11 LIMIT 1; -- For lowest HbA1c

12

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
Franz Brustmeyer	PT98827	Female	51	0	0	current	49.11	3.5	100	0

Limit to 400 rows

1 # Ans 9.2

2

3 • SELECT *

4 FROM diabetes_predictn

5 WHERE HbA1c_level = (

6 SELECT MAX(HbA1c_level)

7 FROM diabetes_predictn)

8 OR

9 HbA1c_level = (

10 SELECT MIN(HbA1c_level)

11 FROM diabetes_predictn

12)

13 ORDER BY HbA1c_level;

Result Grid

Filter Rows:

Export:

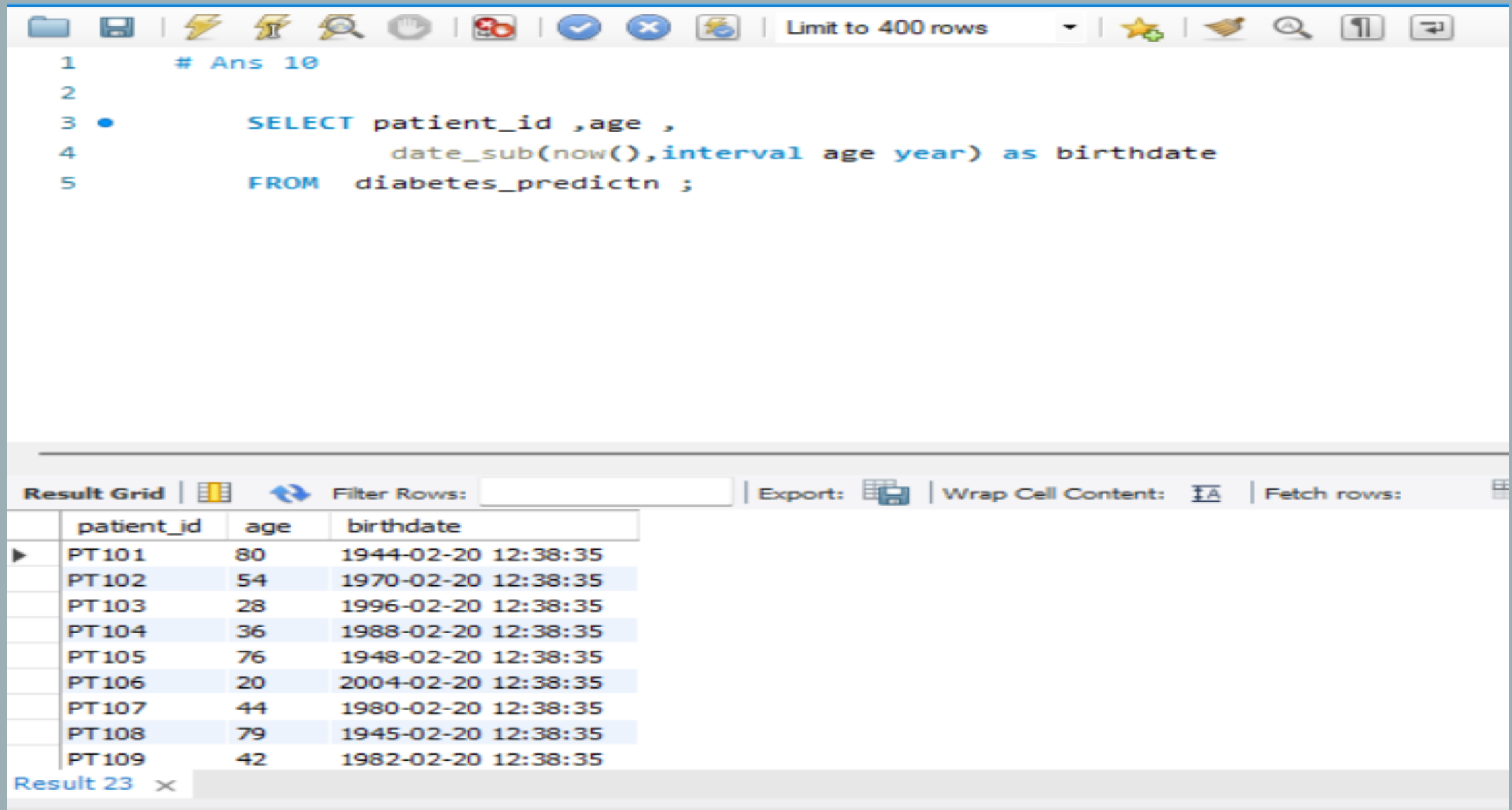
Wrap Cell Content:

Fetch rows:

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
Girma Demissie	PT96157	Male	49	0	0	No Info	27.32	3.5	126	0
Charlotte Coloyan Dela Cruz	PT96162	Female	3	0	0	never	18.54	3.5	155	0
Darnay H McPherson Jr	PT96169	Male	51	0	0	No Info	27.32	3.5	100	0
Sarah E Gieseke	PT96183	Female	5	0	0	No Info	27.32	3.5	200	0
Lily A Lozano	PT96226	Female	80	0	0	No Info	27.32	3.5	145	0
Hao H Nguyen	PT96238	Female	38	0	0	never	33.52	3.5	85	0
Melinda B Oro	PT96241	Female	30	0	0	never	40.31	3.5	90	0
Marilou Lomibao	PT96247	Female	25	0	0	No Info	21.14	3.5	160	0
Don A Alonzo	PT96327	Female	34	0	0	never	27.32	3.5	158	0

diabetes_predictn22 x

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



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 400 rows' dropdown. The SQL editor contains the following query:

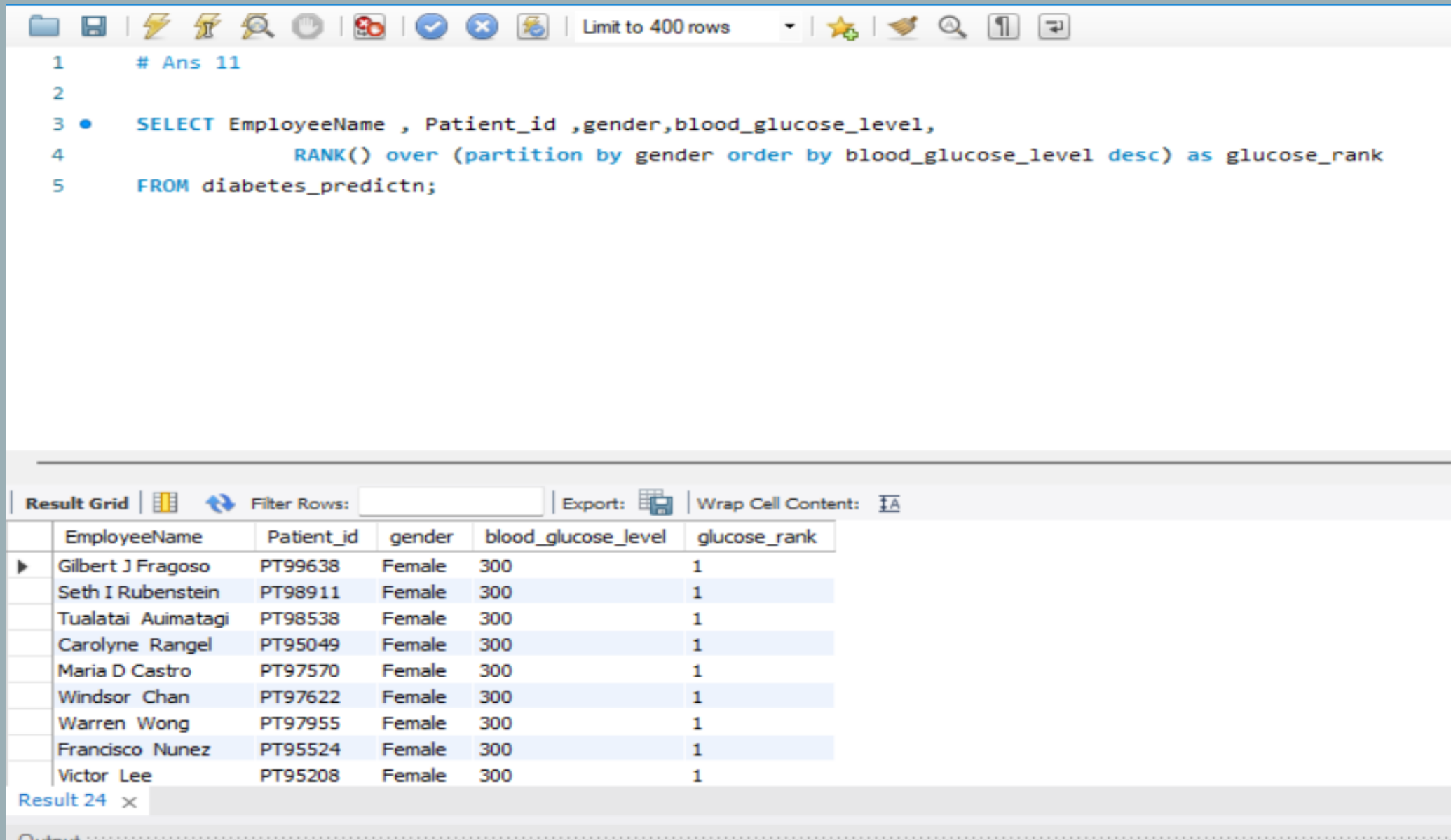
```
1      # Ans 10
2
3      SELECT patient_id ,age ,
4             date_sub(now(),interval age year) as birthdate
5      FROM  diabetes_predictn ;
```

Below the editor is the 'Result Grid' section, which includes a 'Filter Rows' input, an 'Export' button, a 'Wrap Cell Content' toggle, and a 'Fetch rows' dropdown. The result grid displays the following data:

	patient_id	age	birthdate
▶	PT101	80	1944-02-20 12:38:35
	PT102	54	1970-02-20 12:38:35
	PT103	28	1996-02-20 12:38:35
	PT104	36	1988-02-20 12:38:35
	PT105	76	1948-02-20 12:38:35
	PT106	20	2004-02-20 12:38:35
	PT107	44	1980-02-20 12:38:35
	PT108	79	1945-02-20 12:38:35
	PT109	42	1982-02-20 12:38:35

At the bottom left, the text 'Result 23' is visible next to a close button (X).

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



The screenshot shows a SQL query editor window with a toolbar at the top. The query is as follows:

```
1 # Ans 11
2
3 • SELECT EmployeeName , Patient_id ,gender,blood_glucose_level,
4       RANK() over (partition by gender order by blood_glucose_level desc) as glucose_rank
5 FROM diabetes_predictn;
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' toggle. The results are displayed in a table with 6 columns: EmployeeName, Patient_id, gender, blood_glucose_level, and glucose_rank. There are 10 rows of data, all showing a glucose rank of 1 for females with a blood glucose level of 300.

	EmployeeName	Patient_id	gender	blood_glucose_level	glucose_rank
▶	Gilbert J Fragoso	PT99638	Female	300	1
	Seth I Rubenstein	PT98911	Female	300	1
	Tualatai Auimatagi	PT98538	Female	300	1
	Carolyn Rangel	PT95049	Female	300	1
	Maria D Castro	PT97570	Female	300	1
	Windsor Chan	PT97622	Female	300	1
	Warren Wong	PT97955	Female	300	1
	Francisco Nunez	PT95524	Female	300	1
	Victor Lee	PT95208	Female	300	1

At the bottom left, there is a tab labeled 'Result 24' with a close button (X).

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

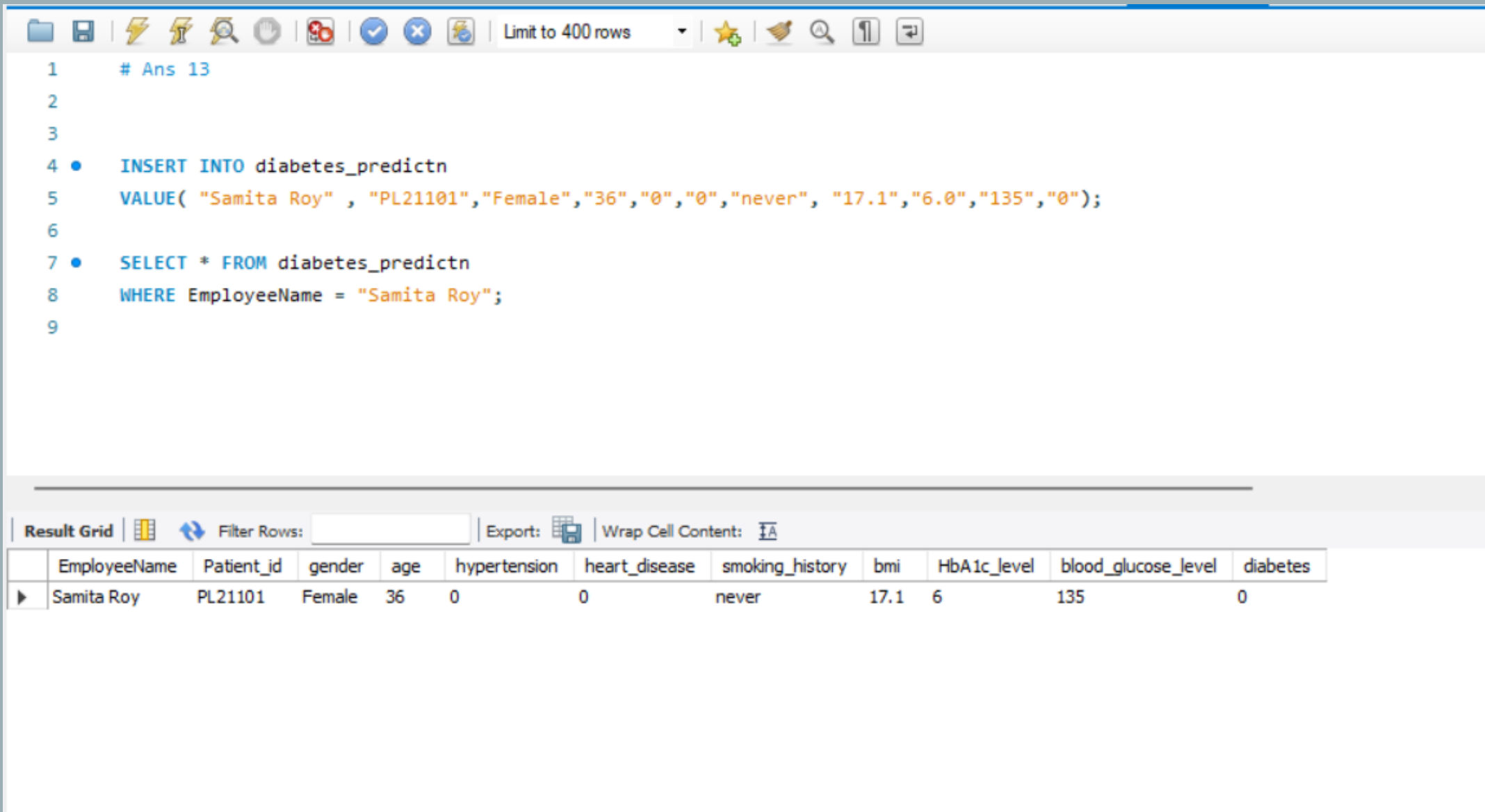
```
1  # Ans 12
2
3  •  set sql_safe_updates = 0 ;
4
5  •  UPDATE diabetes_predictn
6      SET smoking_history = "EX-smoker"
7      WHERE age > 50 ;
8
9  •  SELECT * FROM diabetes_predictn;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

	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
	ALSON LEE	PT107	Female	44	0	0	never	19.31	6.5	200	1
	DAVID KUSHNER	PT108	Female	79	0	0	EX-smoker	23.86	5.7	85	0
	MICHAEL MORRIS	PT109	Male	42	0	0	never	33.64	4.8	145	0

diabetes_predictn25 x

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



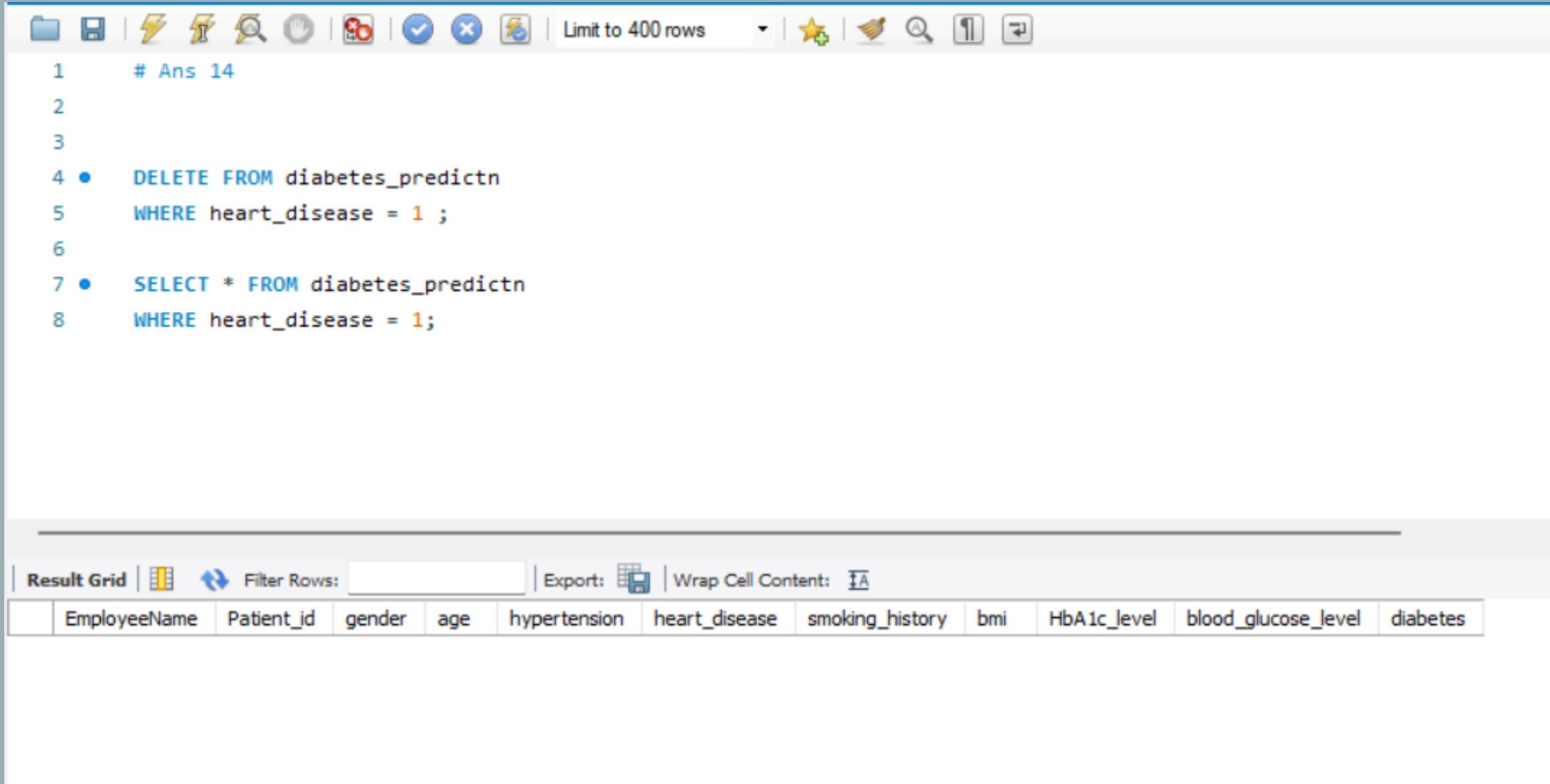
The screenshot shows a database management interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 400 rows' dropdown. The SQL editor displays the following code:

```
1 # Ans 13
2
3
4 • INSERT INTO diabetes_predictn
5   VALUE( "Samita Roy" , "PL21101","Female","36","0","0","never", "17.1","6.0","135","0");
6
7 • SELECT * FROM diabetes_predictn
8   WHERE EmployeeName = "Samita Roy";
9
```

Below the editor is a 'Result Grid' section with a 'Filter Rows' input and an 'Export' button. The grid displays the following data:

	EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
▶	Samita Roy	PL21101	Female	36	0	0	never	17.1	6	135	0

I 4. Delete all patients with heart disease from the database.



The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a "Limit to 400 rows" dropdown. The main area displays a SQL script with line numbers 1 through 8. The script consists of two queries: a DELETE statement and a SELECT statement, both targeting the 'diabetes_predictn' table. The bottom section features a "Result Grid" tab, a "Filter Rows" input field, and an "Export" button. Below these is a table header with 12 columns: EmployeeName, Patient_id, gender, age, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, and diabetes.

```
1 # Ans 14
2
3
4 • DELETE FROM diabetes_predictn
5   WHERE heart_disease = 1 ;
6
7 • SELECT * FROM diabetes_predictn
8   WHERE heart_disease = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: IA

	EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
--	--------------	------------	--------	-----	--------------	---------------	-----------------	-----	-------------	---------------------	----------


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

```
1 # Ans 15
2
3
4 ● SELECT * FROM diabetes_predictn
5 WHERE hypertension = 1
6 ✖ Except
7 SELECT * FROM diabetes_predictn
8 WHERE diabetes = 1 ;
```

Result Grid


 Filter Rows:

 Export:

 Wrap Cell Content:



	EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
▶	DENISE SCHMITT	PT129	Male	45	1	0	never	26.47	4	158	0
	RAY CRAWFORD	PT155	Female	45	1	0	never	23.05	4.8	130	0
	KENNETH SMITH	PT161	Male	44	1	0	current	27.86	6.6	145	0
	CHARLES SCOTT	PT215	Female	55	1	0	EX-smoker	34.2	5.7	140	0
	SHANNON SAKOWSKI	PT227	Male	79	1	0	EX-smoker	28.73	6.6	160	0
	MARISA MORET	PT241	Female	80	1	0	EX-smoker	44.06	6.5	160	0
	STEPHEN TACCHINI	PT326	Female	48	1	0	never	36.73	6.6	126	0
	ANDREW LOGAN	PT339	Male	59	1	0	EX-smoker	25.31	6	130	0
	HAGOP HAJIAN	PT357	Female	52	1	0	EX-smoker	21.46	4	80	0

Result 29 

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



1 # Ans 16

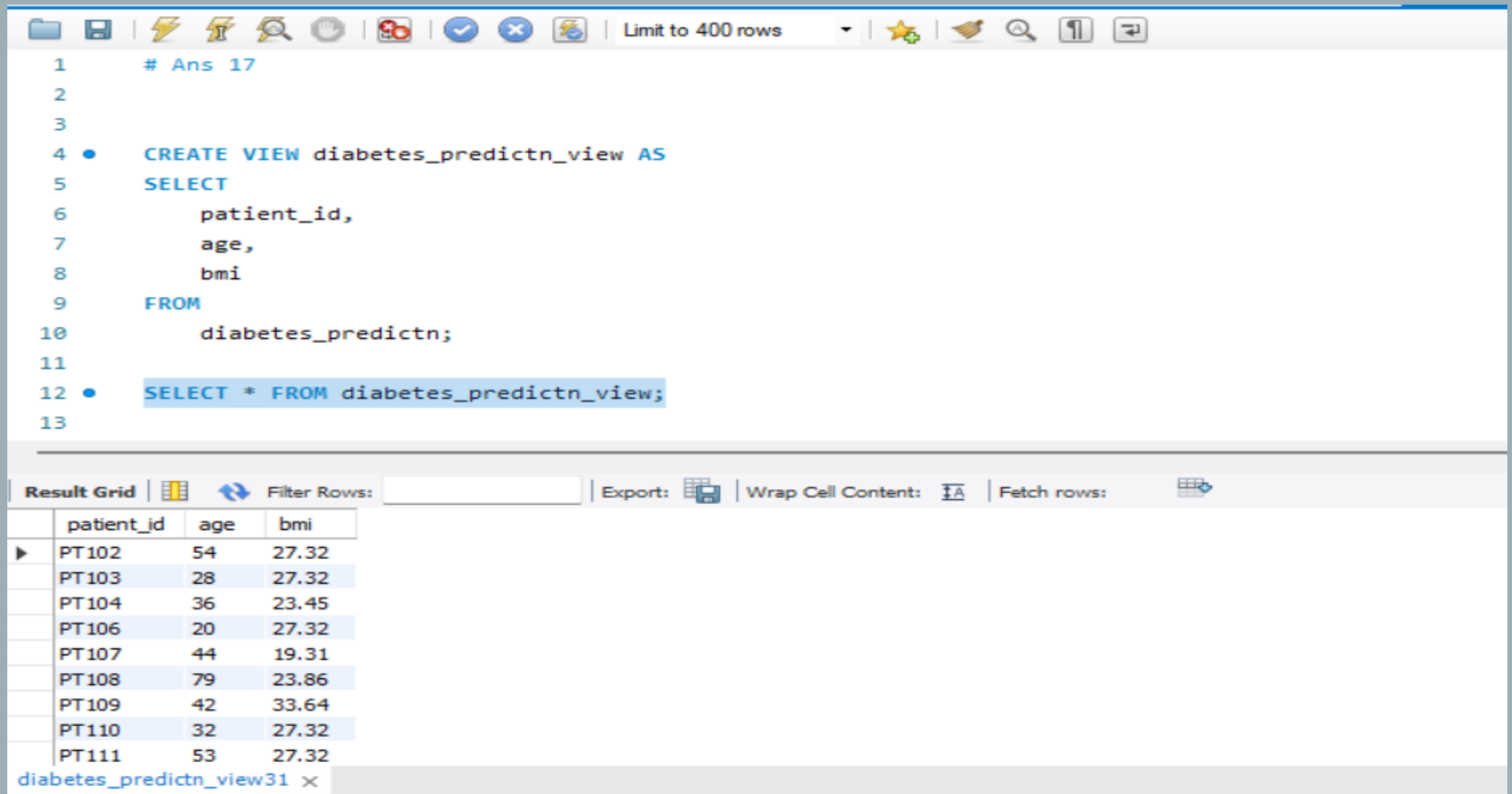
2

3

4 • ALTER TABLE diabetes_predictn

5 ADD CONSTRAINT patient_id UNIQUE (patient_id(255));

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



The screenshot shows a database management interface. The top toolbar includes icons for file operations, execution, and search, along with a "Limit to 400 rows" dropdown. The SQL editor contains the following code:

```
1 # Ans 17
2
3
4 • CREATE VIEW diabetes_predictn_view AS
5 SELECT
6     patient_id,
7     age,
8     bmi
9 FROM
10     diabetes_predictn;
11
12 • SELECT * FROM diabetes_predictn_view;
13
```

Below the editor is the "Result Grid" section, which includes a "Filter Rows:" input field and buttons for "Export:", "Wrap Cell Content:", and "Fetch rows:". The grid displays the following data:

	patient_id	age	bmi
▶	PT102	54	27.32
	PT103	28	27.32
	PT104	36	23.45
	PT106	20	27.32
	PT107	44	19.31
	PT108	79	23.86
	PT109	42	33.64
	PT110	32	27.32
	PT111	53	27.32

At the bottom, a tab labeled "diabetes_predictn_view31" is visible.

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

- 1) **Normalization:** Break down large tables into smaller
- 2) **Use of Primary Keys:** It helps in maintaining data integrity and avoiding duplicate records.
- 3) **Foreign Key Constraints:** Implement foreign key constraints to enforce referential integrity between related tables
- 4) **Unique Constraints:** Apply unique constraints to prevent duplicate values within columns where necessary, ensuring data consistency.
- 5) **Regular Maintenance:** Regularly review and update the database schema based on changing requirements and performance considerations.
- 6) **Review Data Types:** Use appropriate data types for columns to optimize storage and ensure data integrity. Avoid using excessively large data types where not necessary.

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

- 1) **Write efficient queries with proper filtering and minimal calculations.**
- 2) **Duplicate data selectively to reduce complexity and speed up queries**
- 3) **Avoid SELECT *** Instead of selecting all columns using SELECT *, explicitly list the required columns. This reduces the amount of data transferred between the database server and the client application, resulting in faster query execution.
- 4) **Retrieve only the necessary columns and rows from the database by using SELECT**
- 5) **Reduce the use of subqueries .Prefer EXISTS and IN operators over subqueries or JOINS when checking for the existence of records in another table**

Thanking you