Diabetes Prediction Analysis

BY: SRINITHI JAIKUMAR

TABLE OF CONTENTS

OBJECTIVE

- To retrieve and process data efficiently and accurately.
- To perform calculations and aggregations, and organize data into meaningful groups.
- To extract valuable insights that can guide strategic decision-making and enhance the well-being of the workforce.
- To develop or improve skills in data handling and manipulation.
- To identify areas for improvement that will help maintain data integrity.

DATA OVERVIEW

The dataset under analysis comprises 1,00,000 rows and 11 columns, each representing a distinct aspect of the workforce. Below is a brief overview of the key columns present in the dataset:

- **EmployeeName:** The name of the employee responsible for collecting or managing the patient data.
- **Patient_id:** A unique identifier assigned to each patient. This is used to anonymize patient data
- **Gender:** The gender of the patient. Common values are 'Male', 'Female'.
- **D.O.B:** Date of Birth of the patient. This is used to calculate the patient's age.

DATA OVERVIEW

- **Hypertension:** Indicates whether the patient has hypertension (high blood pressure). This could be a binary variable (e.g., 1 or 0 indicating 'Yes' or 'No')

- **heart_disease:** Indicates whether the patient has a history of heart disease. This could be a binary variable(e.g., 1 or 0 indicating 'Yes' or 'No')

- **Smoking_history:** Records the patient's smoking history. This can include categories such as 'Never smoked', 'Former smoker', 'Current smoker'.

- **Bmi:** Body Mass Index of the patient. This is a numerical value calculated from the patient's height and weight and is used to assess whether a person is underweight, normal weight, overweight, or obese.

DATA OVERVIEW

- **HbA1c_level:** Hemoglobin A1c level, which measures the average blood sugar levels over the past 2-3 months. It is a key indicator for managing diabetes.

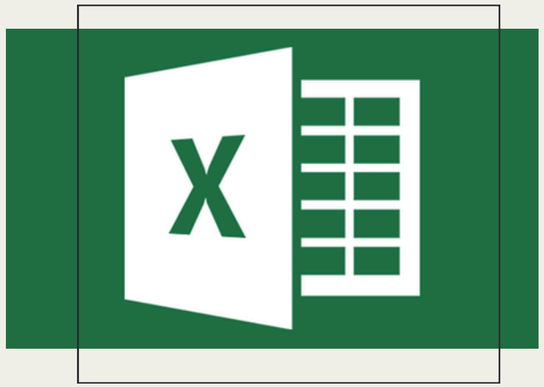
- **Blood_glucose_level:** The current blood glucose (sugar) level of the patient. This is crucial for diagnosing and managing diabetes and other metabolic disorders.

- **Diabetes:** Indicates whether the patient has diabetes. This could be a binary variable (e.g., 'Yes' or 'No').

DATA OVERVIEW

EmployeeName	Patient_id	gender	D.O.B	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
NATHANIEL FORD	PT101	Female	05-11-1992	0	1	never	25.19	6.6	140	0
GARY JIMENEZ	PT102	Female	11-11-1992	0	0	No Info	27.32	6.6	80	0
ALBERT PARDINI	PT103	Male	13-11-1992	0	0	never	27.32	5.7	158	0
CHRISTOPHER CHO	PT104	Female	05-12-1992	0	0	current	23.45	5	155	0
PATRICK GARDNER	PT105	Male	03-01-1989	1	1	current	20.14	4.8	155	0
DAVID SULLIVAN	PT106	Female	05-01-1989	0	0	never	27.32	6.6	85	0
ALSON LEE	PT107	Female	23-01-1989	0	0	never	19.31	6.5	200	1
DAVID KUSHNER	PT108	Female	05-02-1989	0	0	No Info	23.86	5.7	85	0
MICHAEL MORRIS	PT109	Male	21-02-1989	0	0	never	33.64	4.8	145	0
JOANNE HAYES-WH	PT110	Female	09-03-1989	0	0	never	27.32	5	100	0
ARTHUR KENNEY	PT111	Female	19-03-1989	0	0	never	27.32	6.1	85	0
PATRICIA JACKSON	PT112	Female	01-04-1989	0	0	former	54.7	6	100	0
EDWARD HARRING	PT113	Female	14-04-1989	0	0	former	36.05	5	130	0
JOHN MARTIN	PT114	Female	21-04-1989	0	0	never	25.69	5.8	200	0
DAVID FRANKLIN	PT115	Female	26-04-1989	0	0	No Info	27.32	5	160	0
RICHARD CORRIEA	PT116	Male	27-04-1989	0	0	No Info	27.32	6.6	126	0
AMY HART	PT117	Male	29-04-1989	0	0	never	30.36	6.1	200	0
SEBASTIAN WONG	PT118	Female	30-04-1989	0	0	never	24.48	5.7	158	0
MARTY ROSS	PT119	Female	10-05-1989	0	0	No Info	27.32	5.7	80	0

TOOLS USED



MS Excel

Excel will provide a familiar platform for initial data cleaning, exploration, and basic visualizations.

Power BI

Power BI will then come into play, allowing us to create sophisticated dashboards and interactive reports that bring our findings to life

My SQL Workbench

Database Workbench can be used to view, create and edit tables, indexes, stored procedures and other database meta data objects.

Questions

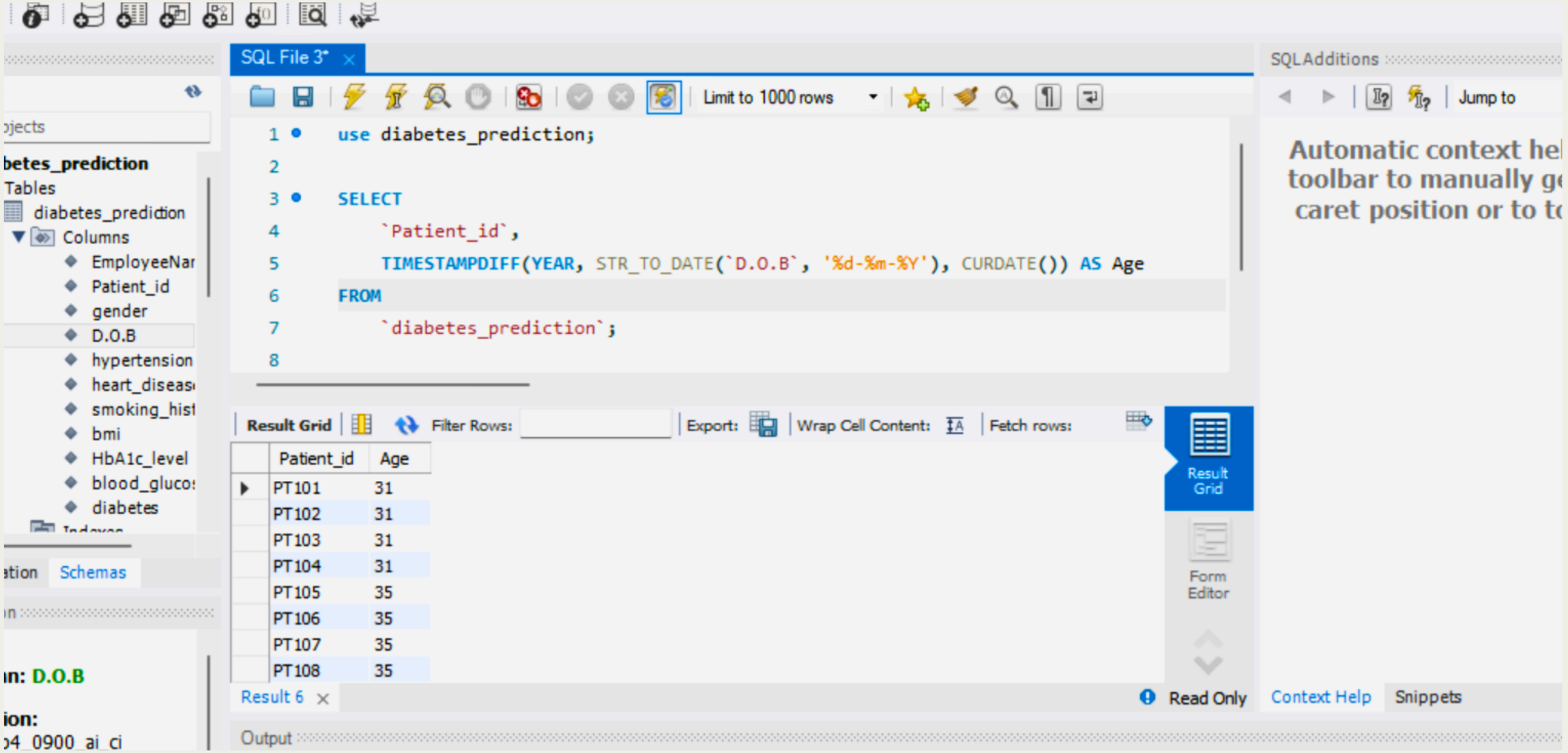
QUESTION #1

Retrieve the Patient_id and ages of all patients.

QUERY USED:

```
SELECT  `Patient_id`,
TIMESTAMPDIFF(YEAR, STR_TO_DATE(`D.O.B`, '%d-%m-
%Y'),CURDATE()) AS Age
FROM  `diabetes_prediction`;
```

ANSWER


SQL File 3*
Limit to 1000 rows
use diabetes_prediction;
SELECT
`Patient_id`,
TIMESTAMPDIFF(YEAR, STR_TO_DATE(`D.O.B`, '%d-%m-%Y'), CURDATE()) AS Age
FROM
`diabetes_prediction`;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:
Patient_id Age
PT101 31
PT102 31
PT103 31
PT104 31
PT105 35
PT106 35
PT107 35
PT108 35
Result 6
Read Only
SQLAdditions
Jump to
Automatic context he
toolbar to manually g
caret position or to t
Context Help
Snippets
Output
Tables
diabetes_prediction
Columns
EmployeeNar
Patient_id
gender
D.O.B
hypertension
heart_diseas
smoking_hist
bmi
HbA1c_level
blood_gluco
diabetes
Schemas

ANSWER

Explanation:

- To calculate age we are using a in-built function calLed “TIMESTAMPDIFF”
- This function in MySQL is used to return a value after subtracting a DateTime expression from another.

Syntax : TIMESTAMPDIFF(unit,expr1,expr2)

Parameters :

- It will accept three parameters.
- unit – It denotes the unit for the result. It can be one of the following-MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR
- expr1 – First date or DateTime expressions.
- expr2 – Second date or DateTime expressions.

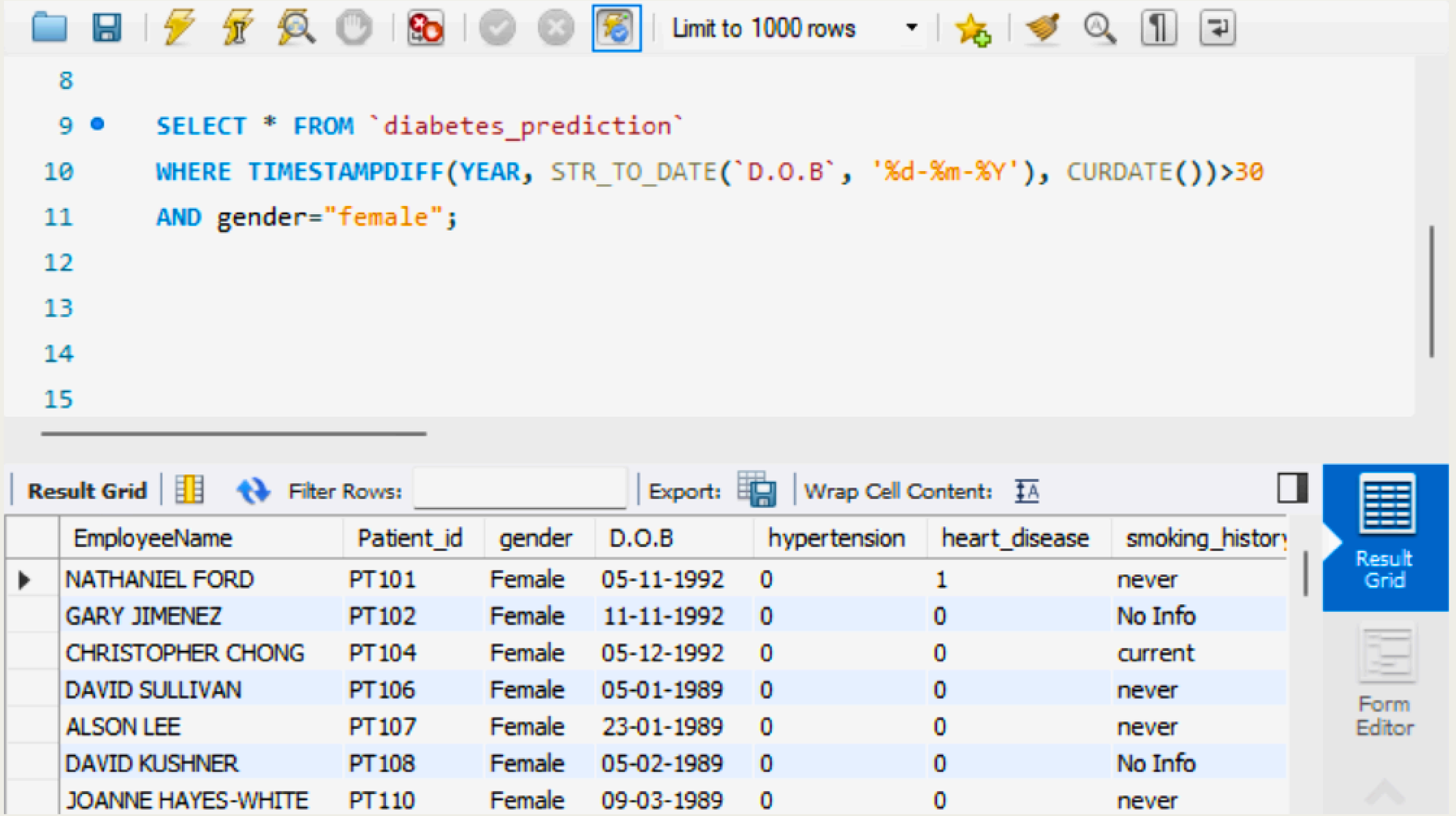
QUESTION #2

Select all female patients who are older than 30.

QUERY USED:

```
SELECT * FROM `diabetes_prediction`
WHERE
    TIMESTAMPDIFF(YEAR, STR_TO_DATE(`D.O.B`, '%d-%m-%Y'), CURDATE())>30
AND
    gender="female";
```

ANSWER



EmployeeName	Patient_id	gender	D.O.B	hypertension	heart_disease	smoking_history
NATHANIEL FORD	PT101	Female	05-11-1992	0	1	never
GARY JIMENEZ	PT102	Female	11-11-1992	0	0	No Info
CHRISTOPHER CHONG	PT104	Female	05-12-1992	0	0	current
DAVID SULLIVAN	PT106	Female	05-01-1989	0	0	never
ALSON LEE	PT107	Female	23-01-1989	0	0	never
DAVID KUSHNER	PT108	Female	05-02-1989	0	0	No Info
JOANNE HAYES-WHITE	PT110	Female	09-03-1989	0	0	never

ANSWER

Explanation:

- Similar to the first question, we use TIMESTAMPDIFF() to calculate the age and apply conditioning to display the data of females who are older than 30

Syntax :

TIMESTAMPDIFF(unit,expr1,expr2)

Parameters :

It will accept three parameters.

- unit – It denotes the unit for the result. It can be one of the following-MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR
- expr1 – First date or DateTime expressions.
- expr2 – Second date or DateTime expressions.

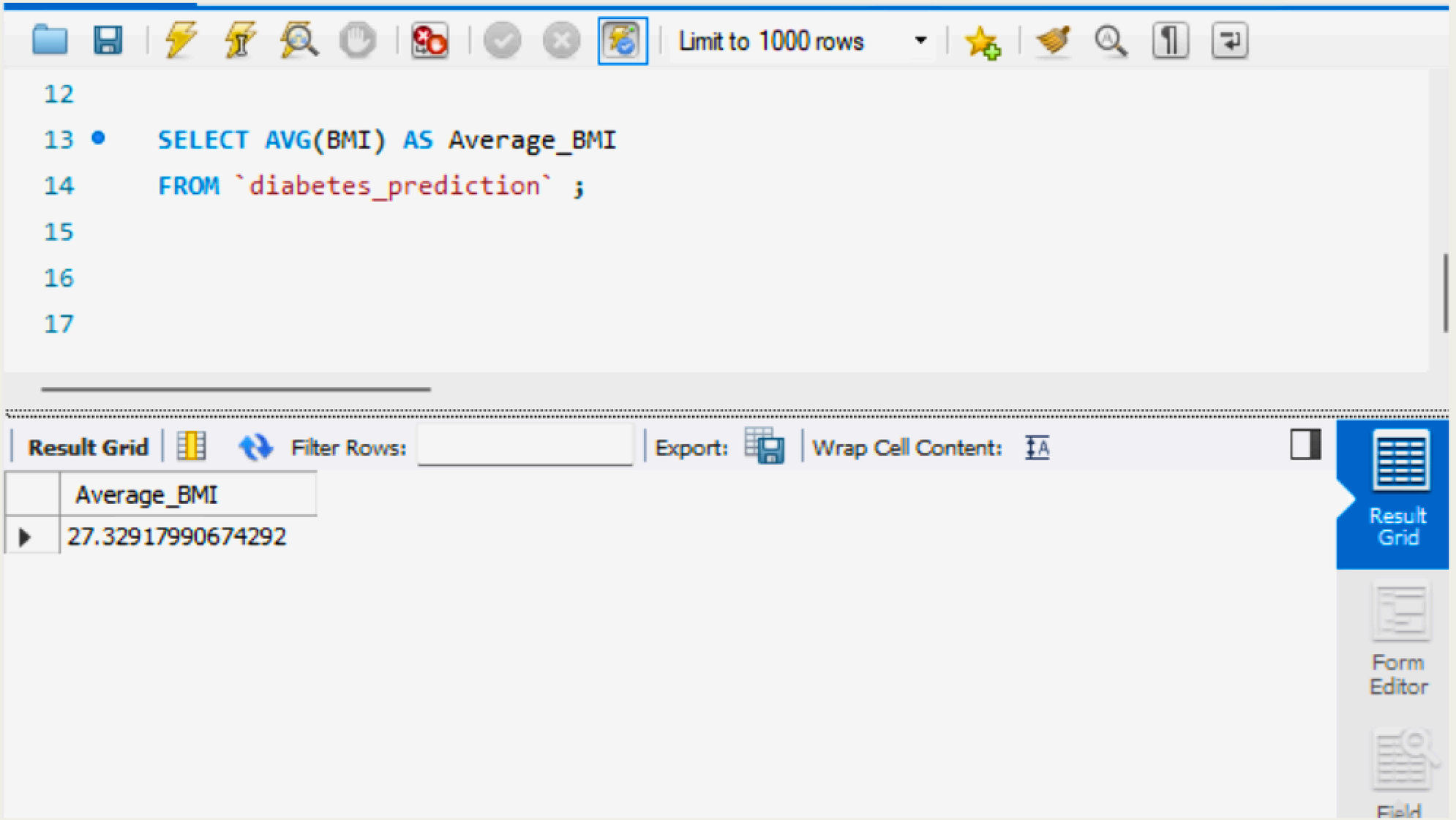
QUESTION #3

Calculate the average BMI of patients.

QUERY USED:

```
SELECT AVG(BMI) AS Average_BMI
FROM `diabetes_prediction` ;
```

ANSWER


Limit to 1000 rows
12
13 SELECT AVG(BMI) AS Average_BMI
14 FROM `diabetes_prediction` ;
15
16
17
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Average_BMI
27.32917990674292
Result Grid
Form Editor

A N S W E R

Explanation:

- Here we use the inbuilt function AVG() to calculate the average BMI of the employees and display the data as `Average_BMI`

Syntax :

AVG(expression)

Parameters :

- expression- Required. A numeric value

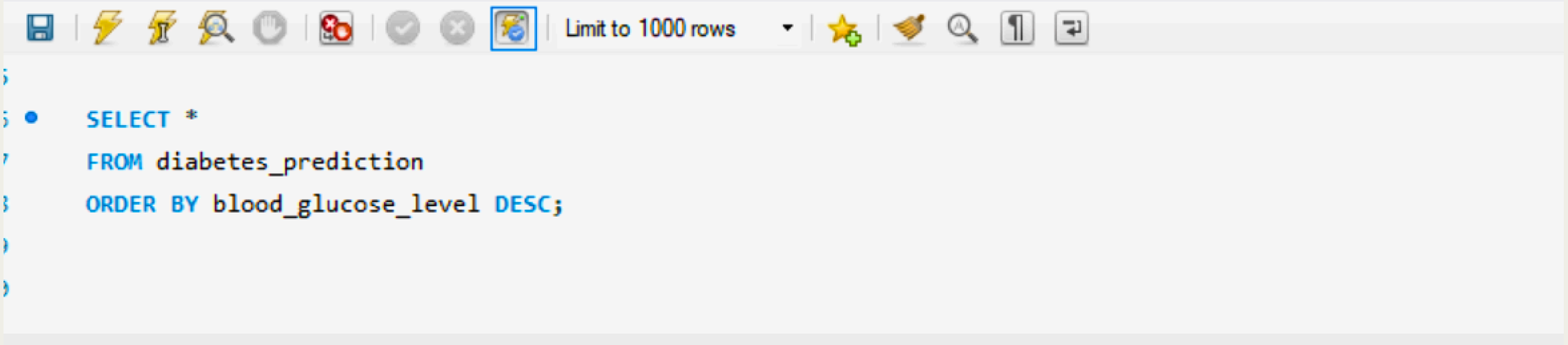
QUESTION #4

List patients in descending order of blood glucose levels.

QUERY USED:

```
SELECT * FROM diabetes_prediction
ORDER BY
blood_glucose_level DESC;
```

ANSWER



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

Patient_id	gender	D.O.B	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabet
PT26831	Female	28-12-1999	0	0	never	29.4	9	300	1
PT27875	Female	02-01-1995	1	0	ever	24.42	6	300	1
PT27314	Female	30-12-1999	1	0	never	43.42	8.2	300	1
PT26826	Male	28-12-1999	0	0	former	36.74	5.7	300	1
PT27698	Male	02-01-1995	0	1	former	24.4	5.8	280	1
PT26500	Female	26-12-1999	0	0	current	28.89	8.8	280	1
PT27616	Male	01-01-1995	0	1	former	30.43	6	280	1
PT27083	Male	29-12-1999	0	0	never	24.62	6.5	280	1

ANSWER

Explanation:

- THE ORDER BY Clause sorts the retrieved records by the blood_glucose_level column. The DESC keyword indicates that the sorting should be in descending order, meaning that records with the highest blood glucose levels will appear first in the result set.

Syntax :

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

QUESTION #5

Find patients who have hypertension and diabetes.

QUERY USED:

```
SELECT Patient_id
FROM  diabetes_prediction
WHERE
hypertension = 1  AND  diabetes = 1;
```

ANSWER

SQL File 3*

Limit to 1000 rows

```
19
20 •    SELECT *
21      FROM diabetes_prediction
22      WHERE hypertension = 1 AND diabetes = 1;
23
24
```

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

EmployeeName	Patient_id	gender	D.O.B	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glu
JONES WONG	PT139	Male	09-08-1989	1	0	current	27.32	5.7	260
PATRIC STEELE	PT205	Female	04-06-1997	1	0	never	27.32	6.8	280
ARTHUR STELLINI	PT343	Male	07-09-1997	1	1	not current	27.77	6.6	160
CHAD LAW	PT355	Male	12-09-1997	1	0	ever	35.06	5.8	200
CATHERINE JAMES	PT451	Female	21-10-1997	1	0	never	50.3	6.6	155
JOHN HART	PT565	Male	10-11-1997	1	0	current	36.12	6.8	140
JOHN BARKER	PT567	Female	11-11-1997	1	0	former	27.32	6.5	159
ROBERT BONNET	PT632	Female	01-12-1997	1	0	not current	36.93	8.8	155

Result Grid

Form Editor

Field

ANSWER

Explanation:

- We apply AND conditioning to display the data of patients who have both hypertension and diabetes.

Syntax :

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

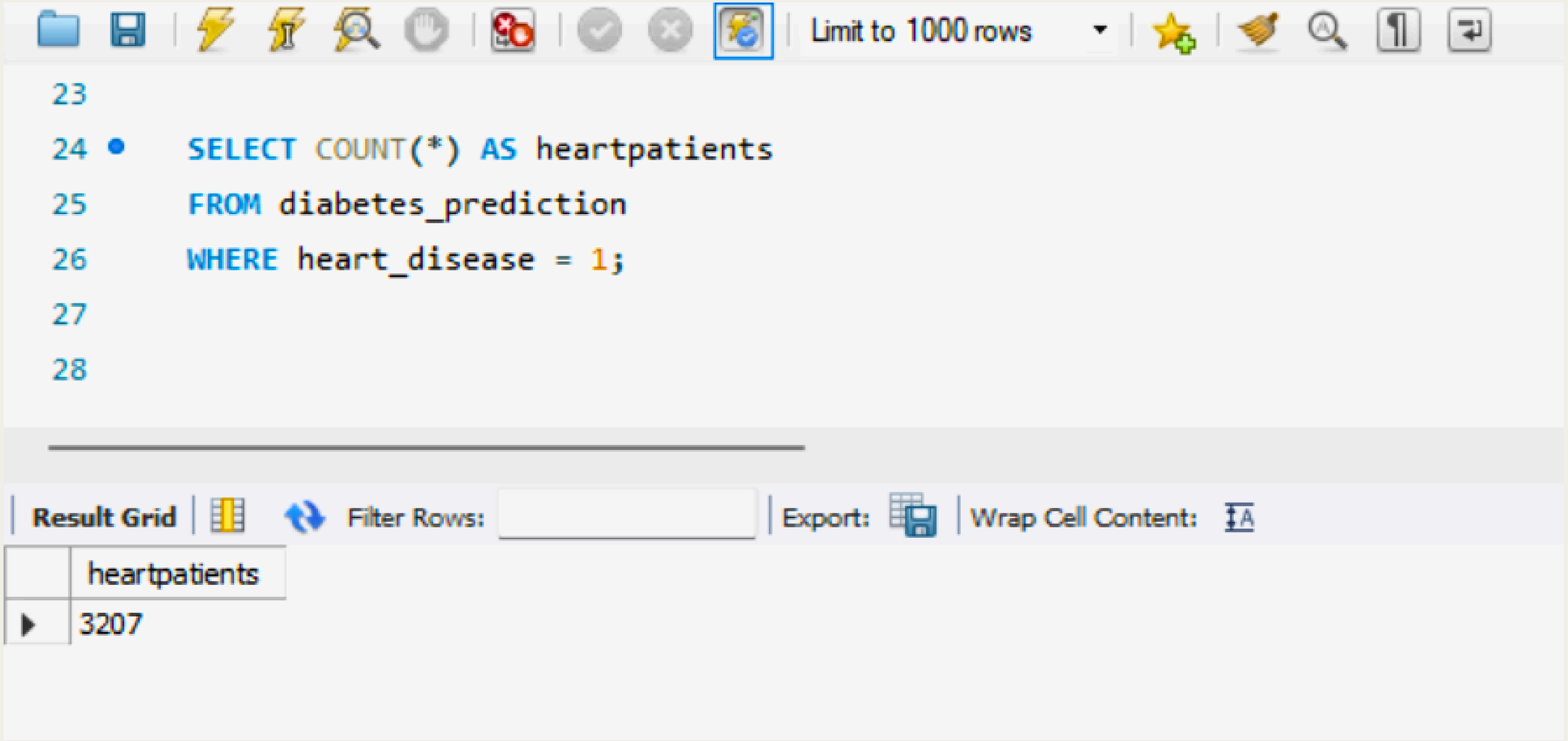
QUESTION #6

Determine the number of patients with heart disease.

QUERY USED:

```
SELECT COUNT(*) AS heartpatients
FROM diabetes_prediction
WHERE heart_disease = 1;
```

ANSWER


Limit to 1000 rows
23
24 • SELECT COUNT(*) AS heartpatients
25 FROM diabetes_prediction
26 WHERE heart_disease = 1;
27
28
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
heartpatients
3207

ANSWER

Explanation:

- The COUNT(*) function returns the total number of records, and the AS clause gives this count a label (alias) for easier reference in the output.
- The WHERE clause filters the records to include only datas where the heart_disease column has a value of 1

Syntax :

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

QUESTION #7

Organize patients based on their smoking history and tally the number of smokers and non-smokers in each group.

QUERY USED:

```
SELECT CASE
    WHEN `smoking_history` = 'current' THEN 'Smoker'
    WHEN `smoking_history` IN ('never', 'No Info', 'former', 'ever', 'not current')
THEN 'Non-Smoker'
    ELSE 'Unknown'
    END AS smoking_status,
    COUNT(*) AS count
FROM diabetes_prediction
GROUP BY smoking_status;
```

ANSWER

SQL File 3

Limit to 1000 rows

```
28  SELECT CASE
29      WHEN `smoking_history` = 'current' THEN 'Smoker'
30      WHEN `smoking_history` IN ('never', 'No Info', 'former', 'ever', 'not current') THEN 'Non-Smoker'
31      ELSE 'Unknown'
32      END AS smoking_status,
33      COUNT(*) AS count
34  FROM diabetes_prediction
35  GROUP BY smoking_status;
36
```

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

smoking_status	count
Non-Smoker	74455
Smoker	7682

Result Grid

Form Editor

QUESTION #8

Retrieve the Patient_id of patients who have a BMI greater than the average BMI

QUERY USED:

```
SELECT patient_id, bmi
FROM diabetes_prediction
WHERE bmi > (SELECT AVG (bmi)
FROM diabetes_prediction);
```

ANSWER

SQL File 3*

Don't Limit

```
35      GROUP BY smoking_status;
36
37  •   SELECT patient_id, bmi
38      FROM diabetes_prediction
39      WHERE bmi > (SELECT AVG (bmi) FROM diabetes_prediction);
40
41
42
43
```

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

patient_id	bmi
PT109	33.64
PT112	54.7
PT113	36.05
PT117	30.36
PT121	36.38
PT124	27.94

Result Grid

Form Editor

QUESTION #9

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

QUERY USED:

```
SELECT patient_id, HbA1c_level
FROM diabetes_prediction
WHERE
    HbA1c_level = (SELECT MAX(HbA1c_level) FROM diabetes_prediction)
OR
    HbA1c_level = (SELECT MIN(HbA1c_level) FROM diabetes_prediction);
```

ANSWER

SQL File 3*

Don't Limit

```
40
41 •  SELECT patient_id, HbA1c_level
42    FROM diabetes_prediction
43    WHERE
44        HbA1c_level = (SELECT MAX(HbA1c_level) FROM diabetes_prediction) OR
45        HbA1c_level = (SELECT MIN(HbA1c_level) FROM diabetes_prediction);
46
47
48
```

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

patient_id	HbA1c_level
PT120	3.5
PT134	3.5
PT141	9
PT145	3.5
PT156	9
PT158	3.5

Result Grid

Form Editor

QUESTION #10

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

QUERY USED:

```
SELECT
    Patient_id,TIMESTAMPDIFF(YEAR, `diabetes_prediction`.`D.O.B`, CURDATE())
AS Age
FROM
    diabetes_prediction;
```

ANSWER

```sql
SELECT
    patient_id,
    ROUND(DATEDIFF(CURRENT_DATE,
    DATE_FORMAT(STR_TO_DATE(`D.O.B`, '%d-%m-%Y'), '%Y-%m-%d')) / 365, 0) AS age_in_years
FROM
    diabetes_prediction;
```

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

patient_id	age_in_years
PT101	32
PT102	32
PT103	32
PT104	32
PT105	35
PT106	35

QUESTION #11

Rank patients by blood glucose level within each gender group

QUERY USED:

```
SELECT
  Patient_id,
  gender,
  blood_glucose_level,
  RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level) AS
Glucose_Rank_Within_Gender
FROM diabetes_prediction;
```

ANSWER

SQL File 3*

Don't Limit

```
54 •  SELECT
55      Patient_id,
56      gender,
57      blood_glucose_level,
58      RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level) AS Glucose_Rank_Within_Gender
59    FROM diabetes_prediction;
60
```

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

Patient_id	gender	blood_glucose_level	Glucose_Rank_Within_Gender
PT48652	Female	80	1
PT47293	Female	80	1
PT46688	Female	80	1
PT45459	Female	80	1
PT45117	Female	85	3443
PT45910	Female	85	3443
PT44887	Female	85	3443

Result Grid

Form Editor

QUESTION #12

Update the smoking history of patients who are older than 40 to "Ex-smoker"

QUERY USED:

```
UPDATE diabetes_prediction
SET smoking_history = 'Ex-smoker'
WHERE TIMESTAMPDIFF(YEAR, STR_TO_DATE(`D.O.B`, '%d-%m-%Y'),
CURDATE()) > 40;
```

ANSWER

```
66 •  UPDATE diabetes_prediction
67    SET smoking_history = 'Ex-smoker'
68    WHERE TIMESTAMPDIFF(YEAR, STR_TO_DATE(`D.O.B`, '%d-%m-%Y'), CURDATE()) > 40;
69
70
71
72
```

Output

Action Output

#	Time	Action	Message
1	22:31:19	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE TIMESTA...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0

QUESTION #13

Insert a new patient into the database with sample data

QUERY USED:

```
INSERT INTO
diabetes_prediction (EmployeeName, patient_id, Gender,`D.O.B`, hypertension,
heart_disease, smoking_history,bmi, HbA1c_level, blood_glucose_level,
diabetes)
VALUES ('Ryan Reynolds','PT077','Male', 18-08-2002,0,0,'never',20.35,5,130,0);
```

ANSWER

```
70  INSERT INTO diabetes_prediction (EmployeeName, patient_id, Gender,
71  `D.O.B`, hypertension, heart_disease, smoking_history,
72  bmi, HbA1c_level, blood_glucose_level, diabetes)
73  VALUES ('Ryan Reynolds','PT077','Male', 18-08-2002,0,0,'never',
74  20.35,5,130,0);
75
76
```

Output

Action Output

#	Time	Action	Message
1	11:43:25	INSERT INTO diabetes_prediction (EmployeeName, patient_id, Gender, `D.O.B`, hype...	1 row(s) affected

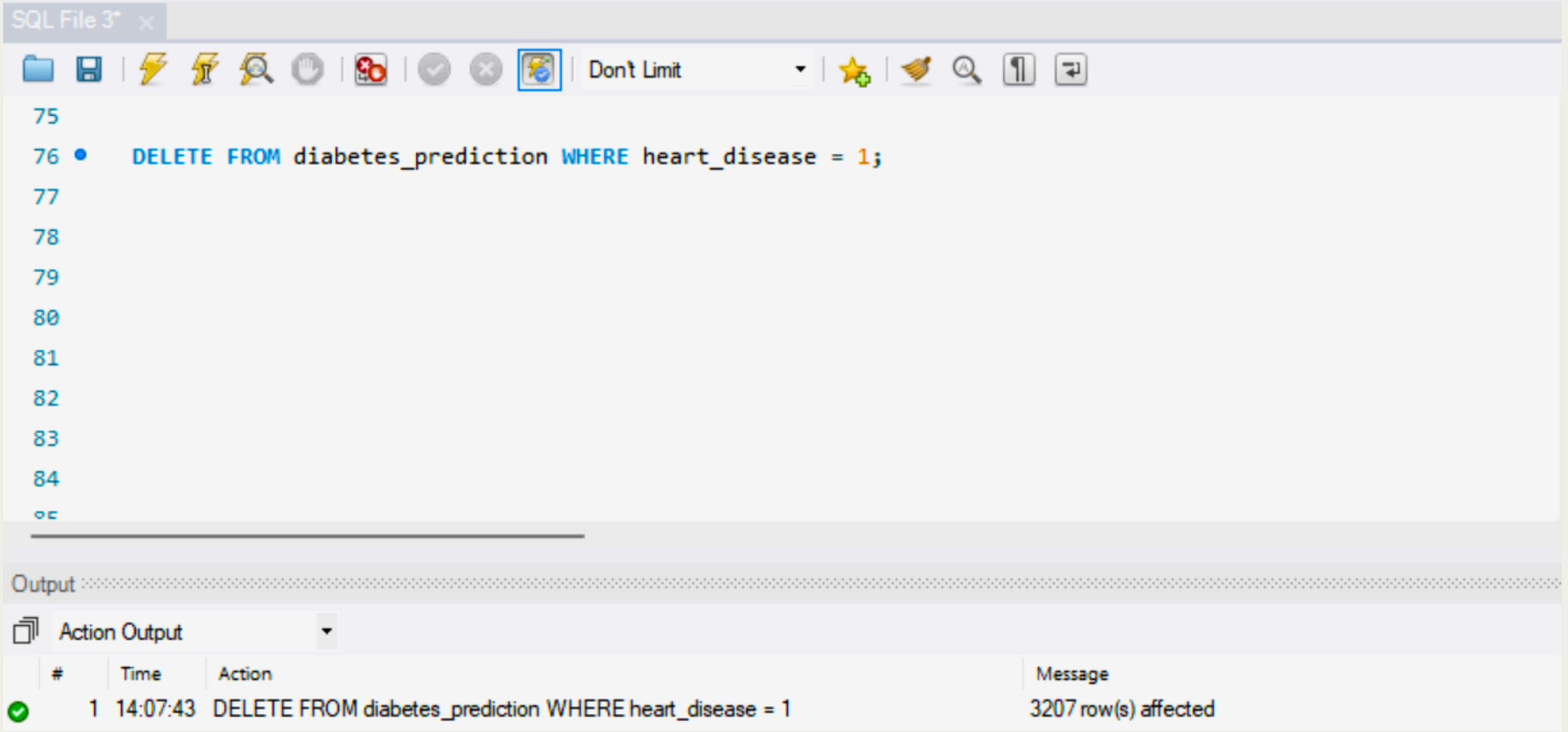
QUESTION #14

Delete all patients with heart disease from the database

QUERY USED:

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

ANSWER


SQL File 3*
Don't Limit
75
76 DELETE FROM diabetes_prediction WHERE heart_disease = 1;
77
78
79
80
81
82
83
84
Output
Action Output
Time Action Message
1 14:07:43 DELETE FROM diabetes_prediction WHERE heart_disease = 1 3207 row(s) affected

ANSWER

Explanation:

- The COUNT(*) function returns the total number of records, and the AS clause gives this count a label (alias) for easier reference in the output.
- The WHERE clause filters the records to include only datas where the heart_disease column has a value of 1

Syntax :

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

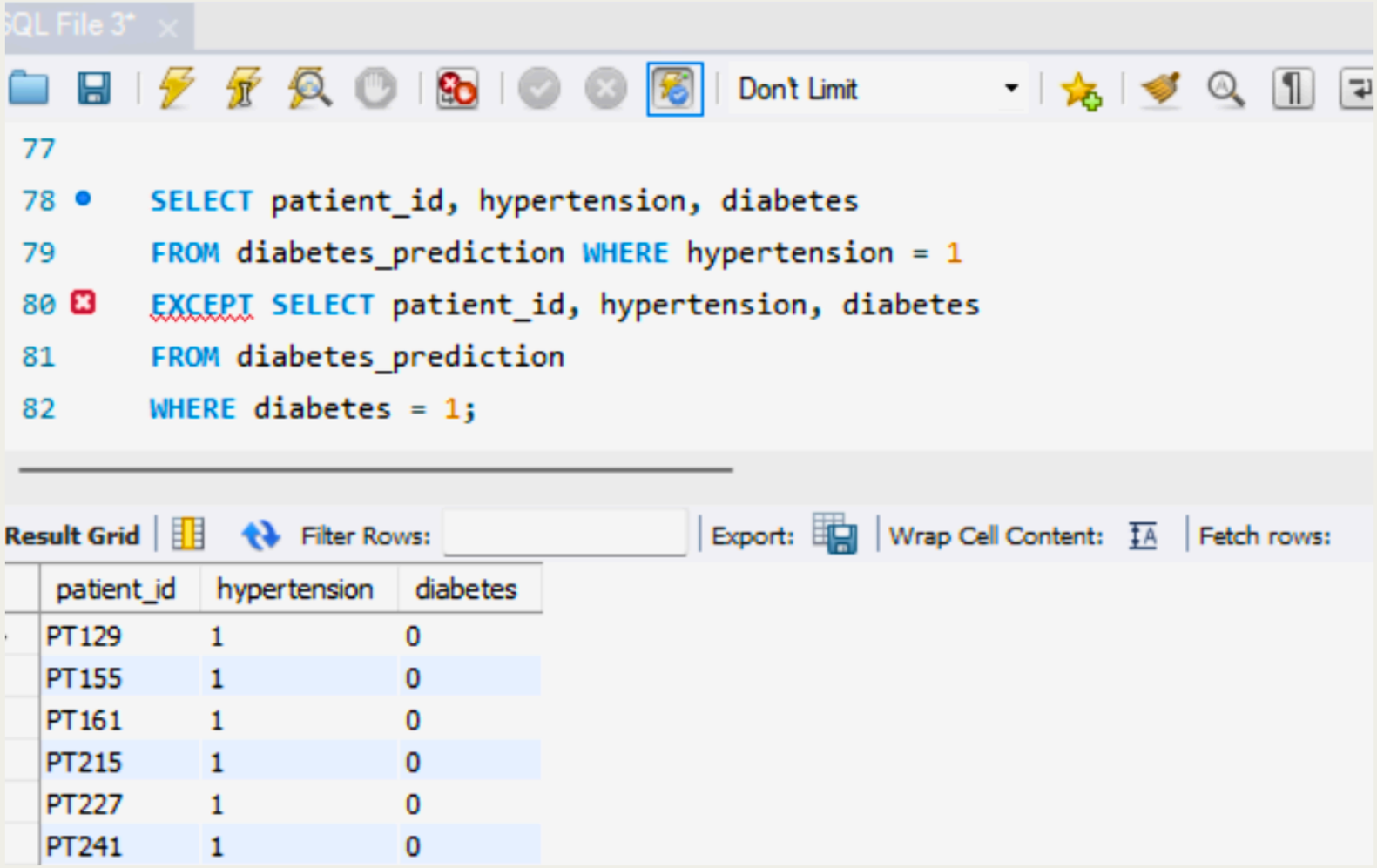
QUESTION #15

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

QUERY USED:

```
SELECT patient_id, hypertension, diabetes
FROM diabetes_prediction WHERE hypertension = 1
EXCEPT SELECT patient_id, hypertension, diabetes
FROM diabetes_prediction
WHERE diabetes = 1;
```

ANSWER


SQL File 3*
Don't Limit
77
78 SELECT patient_id, hypertension, diabetes
79 FROM diabetes_prediction WHERE hypertension = 1
80 EXCEPT SELECT patient_id, hypertension, diabetes
81 FROM diabetes_prediction
82 WHERE diabetes = 1;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:
patient_id hypertension diabetes
PT129 1 0
PT155 1 0
PT161 1 0
PT215 1 0
PT227 1 0
PT241 1 0

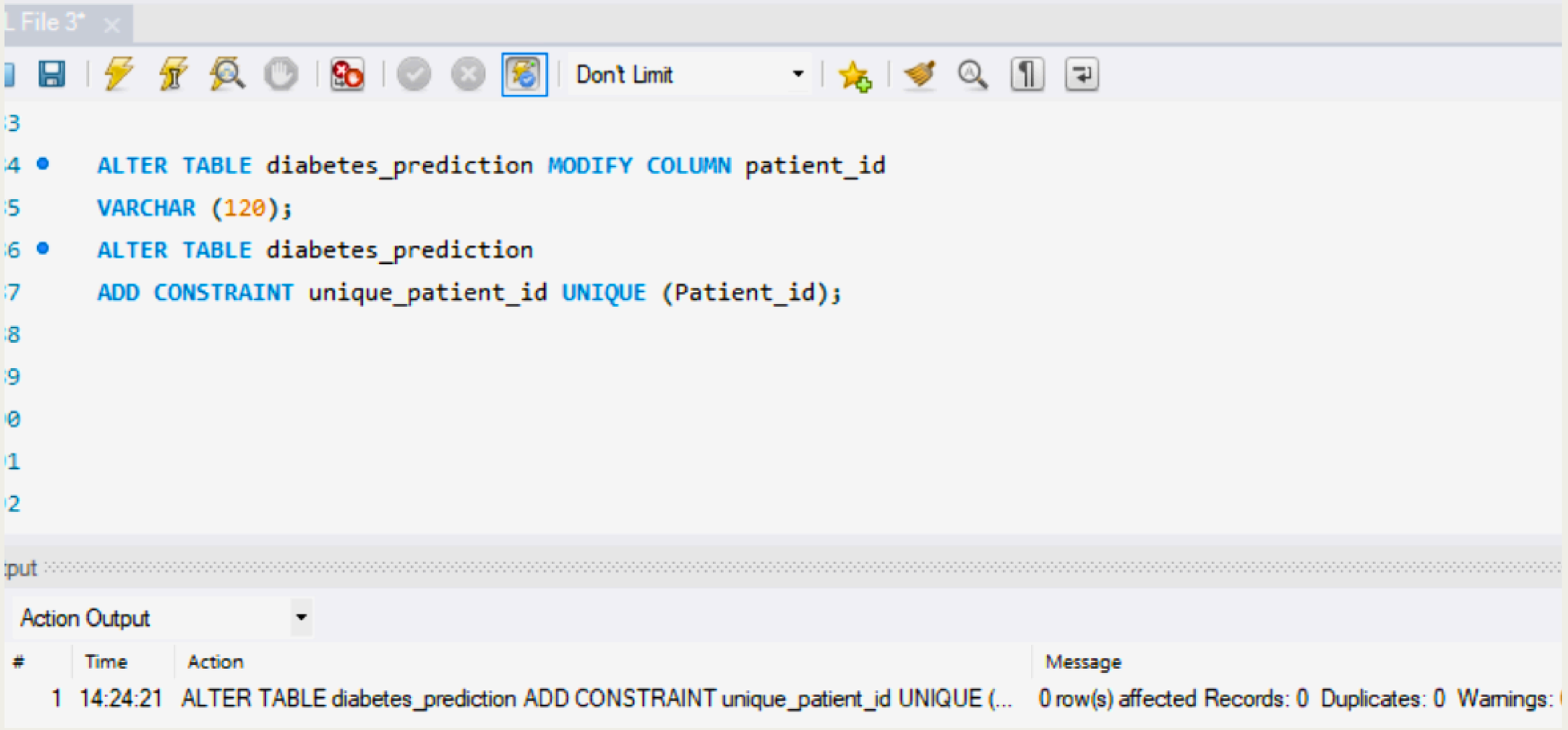
QUESTION #16

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

QUERY USED:

```
ALTER TABLE diabetes_prediction MODIFY COLUMN patient_id
VARCHAR (120);
ALTER TABLE diabetes_prediction
ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id);
```

ANSWER


File 3*
Don't Limit
ALTER TABLE diabetes_prediction MODIFY COLUMN patient_id
VARCHAR (120);
ALTER TABLE diabetes_prediction
ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id);
Action Output
Time Action Message
1 14:24:21 ALTER TABLE diabetes_prediction ADD CONSTRAINT unique_patient_id UNIQUE (... 0 row(s) affected Records: 0 Duplicates: 0 Warnings:

QUESTION #17

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

QUERY USED:

CREATE VIEW PatientInfo AS SELECT `Patient_id`, TIMESTAMPDIFF(YEAR, `D.O.B`, CURDATE()) AS Age, `bmi` FROM diabetes_prediction;

ANSWER

SQL File 3*

Don't Limit

```
89  CREATE VIEW PatientInfo AS
90  SELECT
91   `Patient_id`,
92   TIMESTAMPDIFF(YEAR,
93  `D.O.B`, CURDATE()) AS Age,
94   `bmi`
95  FROM diabetes_prediction;
96
97
```

Output

Action Output

#	Time	Action	Message
1	20:50:53	CREATE VIEW PatientInfo AS SELECT `Patient_id`, TIMESTAMPDIFF(YEAR, `D.O....	0 row(s) affected

QUESTION #18

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

ANSWER

Normalize the Database

Normalization involves decomposing a database into multiple related tables to minimize redundancy and dependency. The process typically follows several normal forms (1NF, 2NF, 3NF, BCNF, etc.).

Implement Constraints

- Primary Key Constraints: Ensure each row has a unique identifier.
- Unique Constraints: Ensure values in specific columns are unique across the table.
- Check Constraints: Ensure that values in a column meet specific conditions.
- Default Constraints: Assign default values to columns if no value is provided.

Use Foreign Keys

Implement foreign keys to create relationships between tables, which enforces referential integrity. Foreign keys ensure that a value in one table must match a value in another table, preventing orphan records and ensuring consistency.

Use Indexes

Proper indexing improves query performance but should be used judiciously to avoid unnecessary overhead. Unique indexes help maintain uniqueness of values in columns.

Avoid Duplicate Data

Identify and eliminate duplicate data entries by restructuring the database. For instance, if the same information is stored in multiple places, consider combining those into a single table and using foreign keys to reference it.

Denormalize for Performance (if necessary)

In some cases, denormalization is necessary to improve performance, particularly in read-heavy databases. Carefully consider and document any denormalization to ensure it doesn't lead to excessive redundancy or integrity issues.

QUESTION #19

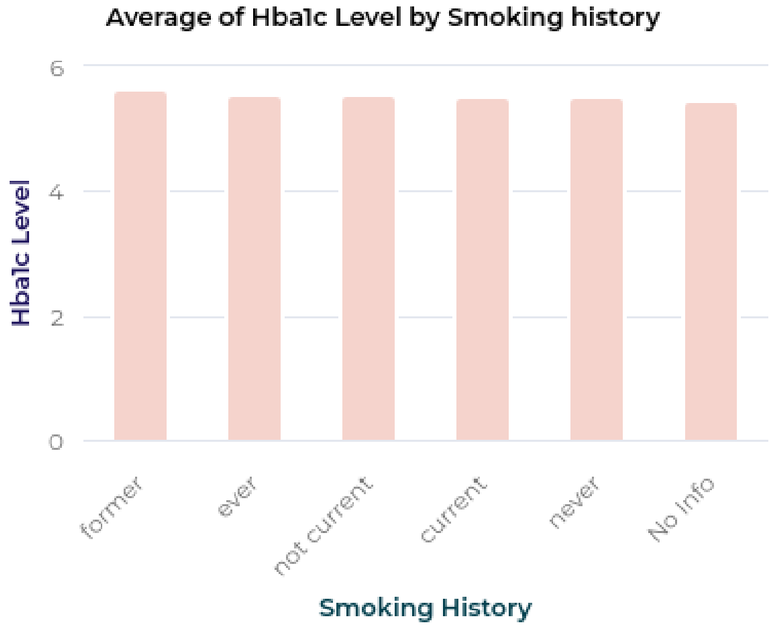
Explain how you can optimize the performance of SQL queries on this dataset.

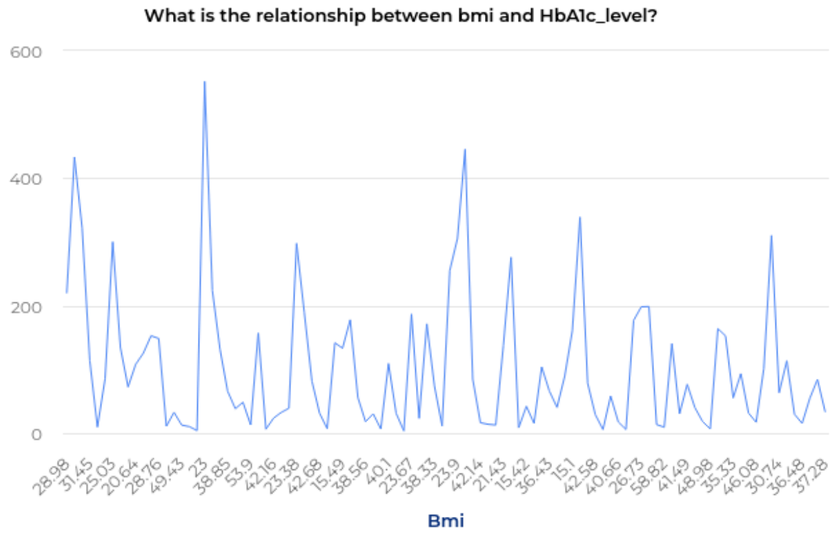
ANSWER

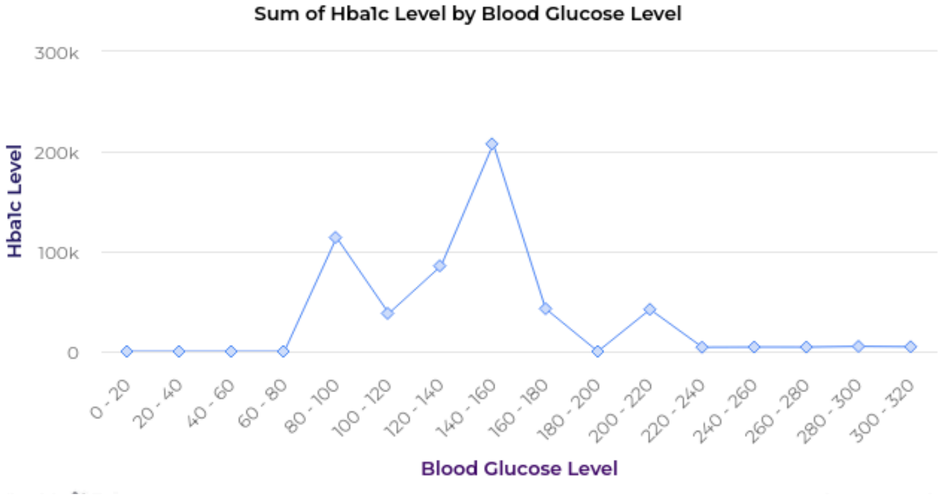
- **Use Indexes:** Create indexes on frequently queried columns.
- **Optimize JOINs:** Use appropriate JOIN types and conditions.
- **Limit Data Retrieval:** Fetch only necessary columns and rows.
- **Avoid Subqueries:** Rewrite queries to minimize subquery usage.
- **Optimize GROUP BY and ORDER BY:** Minimize usage, consider indexing.
- **Use EXISTS and NOT EXISTS:** Prefer over IN and NOT IN for subqueries.
- **Parameterize Queries:** Prevent SQL injection, improve caching.
- **Update Statistics:** Keep table and index statistics up-to-date.
- **Consider Materialized Views:** Precompute and store complex query results.
- **Partition Tables:** Split large tables into smaller partitions.
- **Use Analytical Functions:** Leverage SQL's built-in analytical functions.
- **Optimize Server Configuration:** Tune server settings for workload and resources.

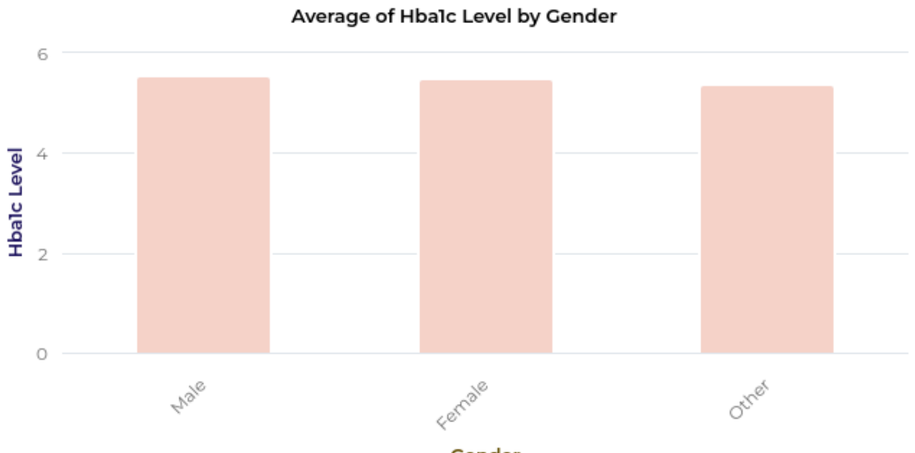
DASHBOARD

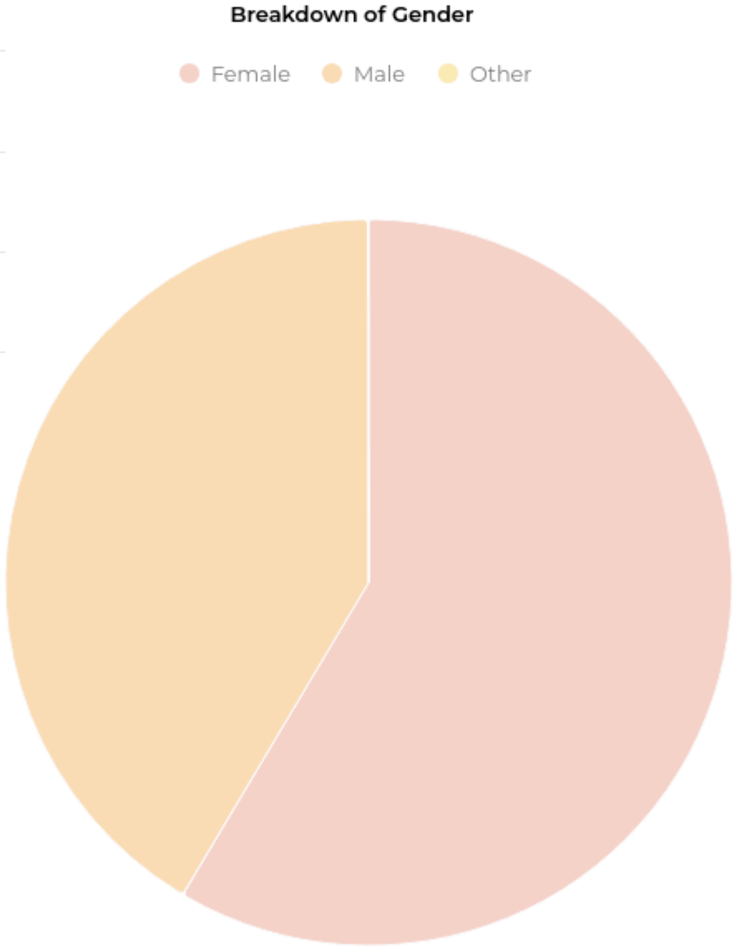
Diabetes and Health Risk Assessment Dashboard











CONCLUSION

The diabetes analysis project provided valuable insights into the factors contributing to diabetes and identified key areas for intervention. The analysis was comprehensive and actionable by leveraging tools like Excel, SQL Workbench, and Power BI. The findings and recommendations from this project can significantly contribute to improving diabetes management and prevention strategies, ultimately enhancing public health outcomes.

I am grateful for the opportunity to work on this project and look forward to applying these analytical skills to future data-driven challenges.

Thank you!

Srinithi Jaikumar
B.Tech Artificial Intelligence and Data Science
Srinithij.aids2022@citchennai.net