



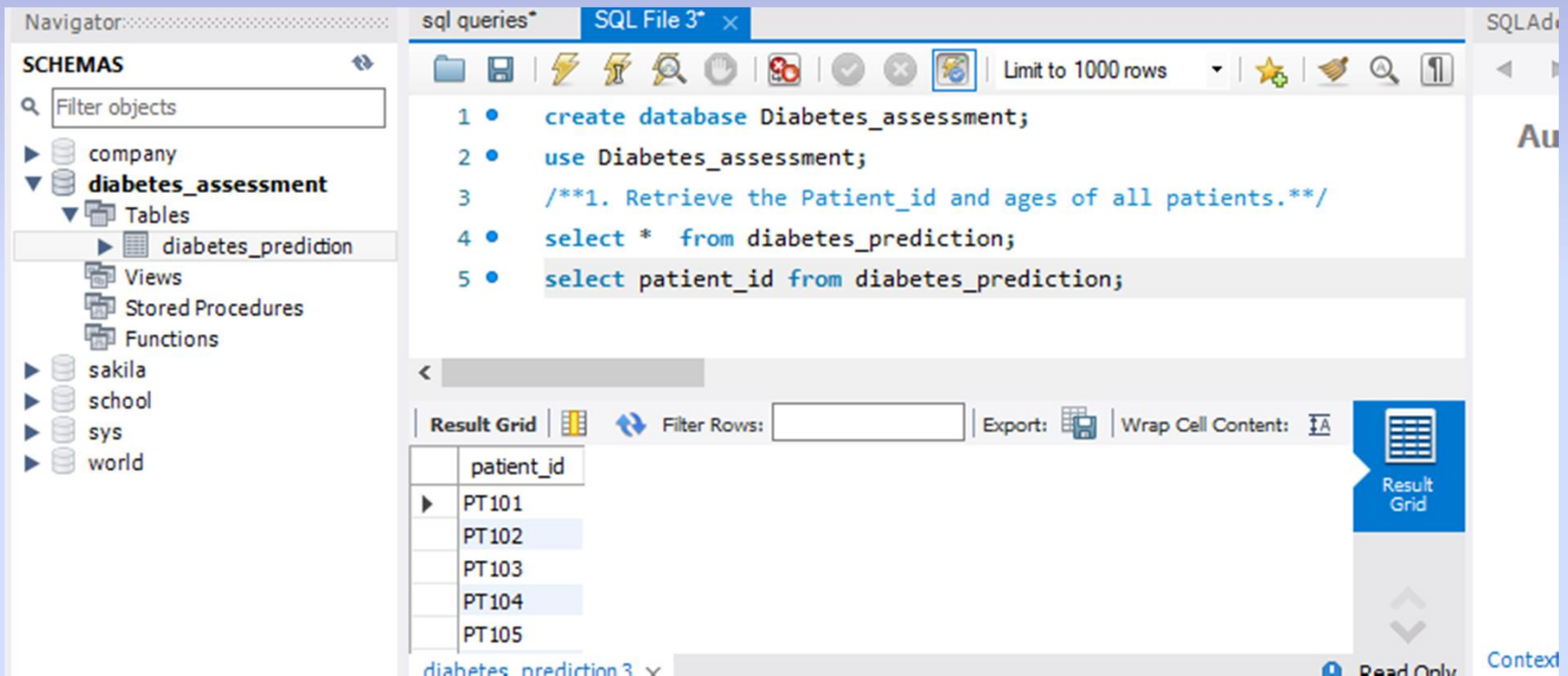
Diabetes Prediction

PSYLIQ DATA ANALYST INTERNSHIP

NEHA DANDEKAR

TASK 2

1. Retrieve the Patient_id of all patients.



The screenshot shows the SQL Developer interface. On the left, the 'Navigator' pane displays the 'SCHEMAS' tree with 'diabetes_assessment' expanded, showing 'Tables' and 'diabetes_prediction'. The main window, titled 'sql queries*' and 'SQL File 3*', contains the following SQL code:

```
1 • create database Diabetes_assessment;  
2 • use Diabetes_assessment;  
3 • /**1. Retrieve the Patient_id and ages of all patients.**/  
4 • select * from diabetes_prediction;  
5 • select patient_id from diabetes_prediction;
```

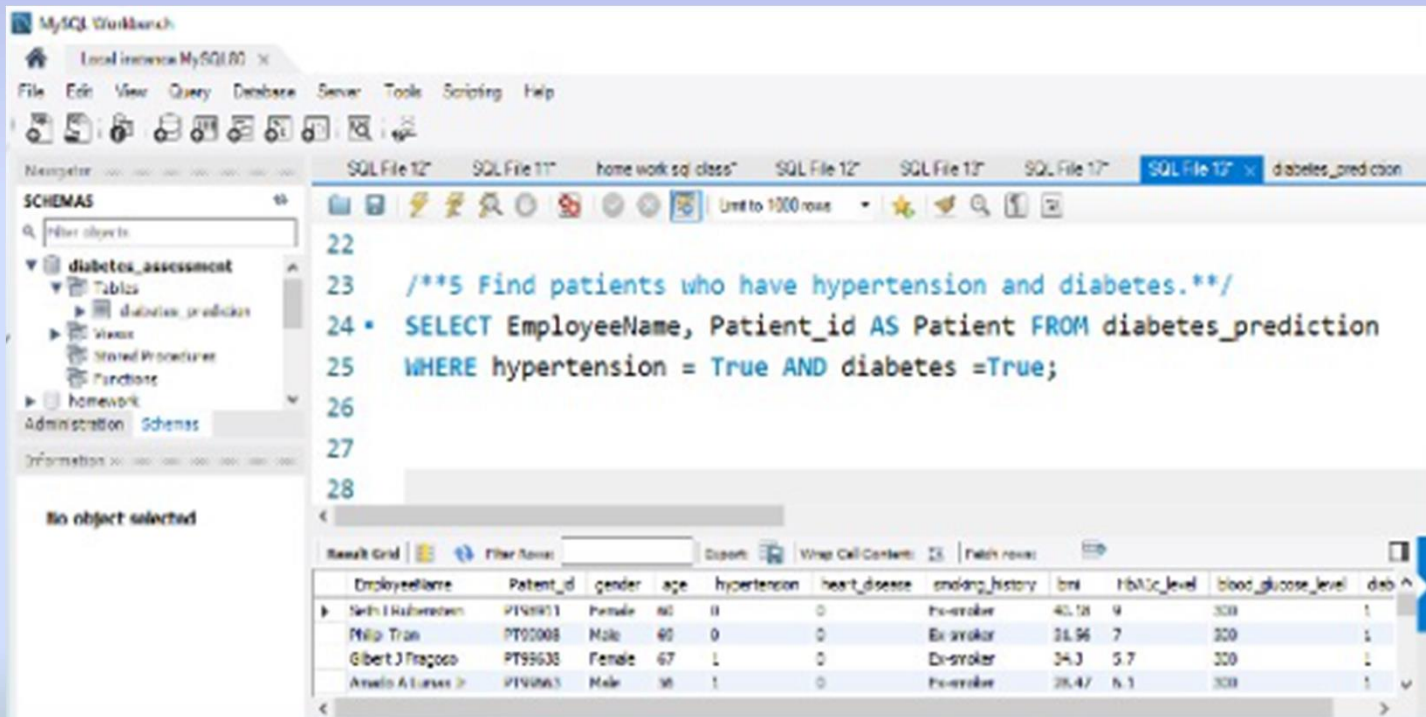
Below the code editor, the 'Result Grid' tab is active, displaying the results of the query. The grid has a single column labeled 'patient_id' and five rows of data:

| patient_id |
|------------|
| PT101 |
| PT102 |
| PT103 |
| PT104 |
| PT105 |

The interface also includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field.

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

- select * from diabetes_prediction WHERE gender='Female' AND age > 40;



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'diabetes_prediction' selected. The main editor window shows the following SQL query:

```

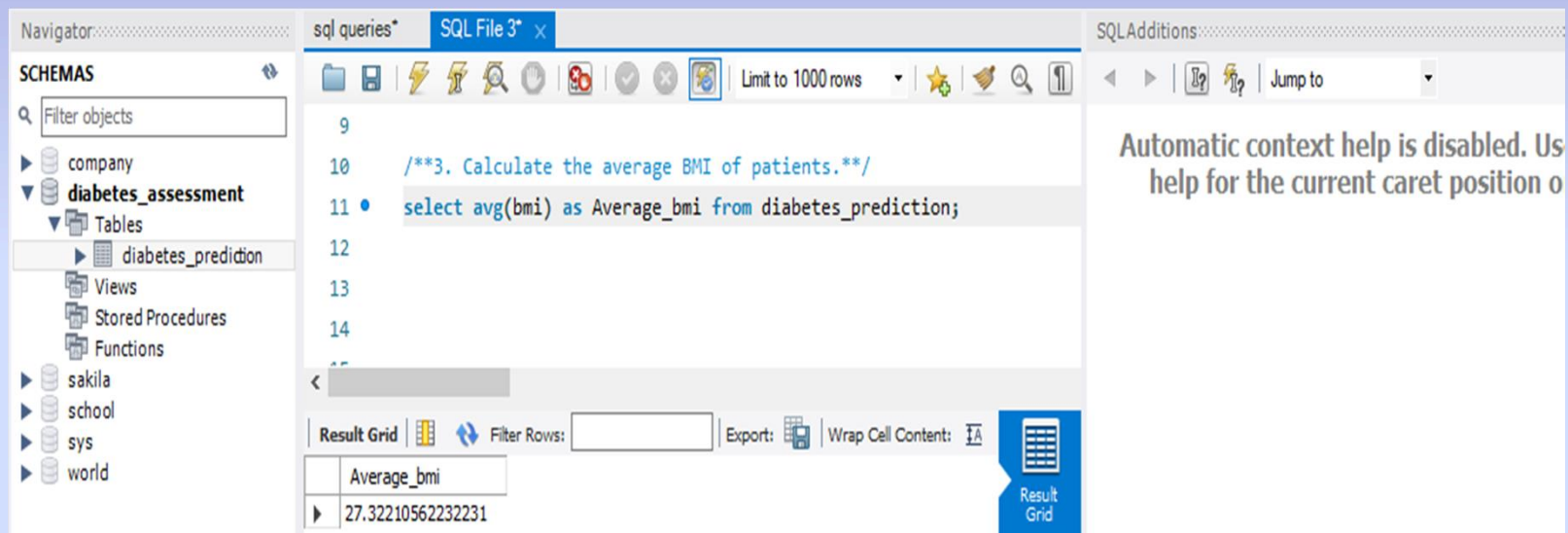
22
23  /**5 Find patients who have hypertension and diabetes.**/
24  SELECT EmployeeName, Patient_id AS Patient FROM diabetes_prediction
25  WHERE hypertension = True AND diabetes =True;
26
27
28

```

The results are displayed in a table below the query:

| EmployeeName | Patient_id | gender | age | hypertension | heart_disease | smoking_history | bmi | HbA1c_level | blood_glucose_level | diabetes |
|-------------------|------------|--------|-----|--------------|---------------|-----------------|-------|-------------|---------------------|----------|
| Seth I Rubenstein | PT98931 | Female | 40 | 0 | 0 | Ex-smoker | 40.18 | 9 | 300 | 1 |
| Milo Tran | PT02008 | Male | 40 | 0 | 0 | Ex-smoker | 31.96 | 7 | 300 | 1 |
| Gilbert J Fragoso | PT99638 | Female | 67 | 1 | 0 | Ex-smoker | 34.3 | 5.7 | 300 | 1 |
| Arnold A Lussac | PT90863 | Male | 38 | 1 | 0 | Ex-smoker | 28.47 | 6.1 | 300 | 1 |

3. Calculate the average BMI of patients.



The screenshot shows the SQL Developer interface with the following components:

- Navigator:** Displays the database schema structure. The **diabetes_assessment** schema is expanded, showing the **diabetes_prediction** table.
- SQL File 3* x:** The main editor window containing the following SQL query:

```
9
10  /**3. Calculate the average BMI of patients.**/
11  •  select avg(bmi) as Average_bmi from diabetes_prediction;
12
13
14
```
- SQLAdditions:** A panel on the right displaying the message: "Automatic context help is disabled. Use help for the current caret position on".
- Result Grid:** A table at the bottom showing the query results:

| Average_bmi |
|-------------------|
| 27.32210562232231 |

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

Navigator: Schemas

- Filter objects
- company
- diabetes_assessment
 - Tables
 - diabetes_prediction
 - Views
 - Stored Procedures
 - Functions
- sakila
- school
- sys
- world

sql queries* SQL File 3* x

Limit to 1000 rows

```

12
13  /**4. List patients in descending order of blood glucose levels.**/
14  • select * from diabetes_prediction order by blood_glucose_level desc;
15
16
17
18
19

```

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_glucose_level | diabe |
|---|--------------------|------------|--------|-----------|--------------|---------------|-----------------|-------|-------------|---------------------|-------|
| ▶ | Adrian G Mendez | PT98419 | Male | 9/29/1995 | 0 | 0 | not current | 27.32 | 6.5 | 300 | 1 |
| | Lenora G Banks | PT98454 | Female | 9/29/1995 | 1 | 0 | never | 38.59 | 6.6 | 300 | 1 |
| | Dante Rogayan | PT98461 | Male | 9/29/1995 | 0 | 0 | No Info | 27.72 | 6.6 | 300 | 1 |
| | Tinisha C Bishop | PT98500 | Male | 9/29/1995 | 0 | 0 | No Info | 27.32 | 8.8 | 300 | 1 |
| | Tualatai Auimatagi | PT98538 | Female | 9/30/1995 | 0 | 0 | never | 26.52 | 8.2 | 300 | 1 |

Administration Schemas

Information: diabetes_prediction 6 x

Output:

5. Find patients who have hypertension and diabetes.

Navigator: sql queries* SQL File 3* x

SCHEMAS

Filter objects

- company
- diabetes_assessment
 - Tables
 - diabetes_prediction
 - Views
 - Stored Procedures
 - Functions
- sakila
- school
- sys
- world

Administration Schemas Information

Limit to 1000 rows

```

15
16 /**5. Find patients who have hypertension and diabetes.**/
17 • select EmployeeName, patient_ hypertension, diabetes from diabetes_prediction where hypertension= True and diabetes= T
18
19
20
21
22
  
```

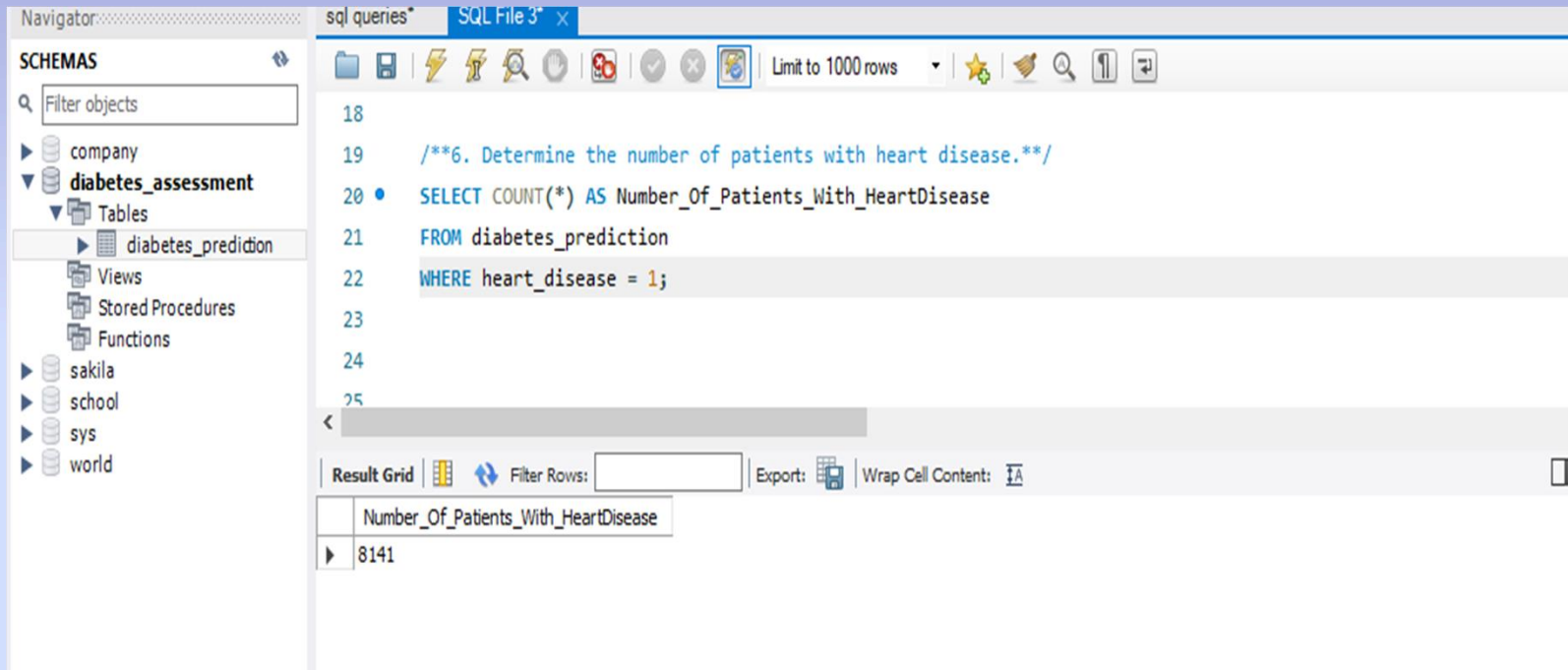
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| EmployeeName | patient_id | hypertension | diabetes |
|-------------------|------------|--------------|----------|
| Eli Thomas | PT40819 | 1 | 1 |
| Kim Truong-Nguyen | PT40821 | 1 | 1 |
| Bevan Dufty | PT40829 | 1 | 1 |
| Nancy Brewer | PT40901 | 1 | 1 |
| Christopher Spear | PT40906 | 1 | 1 |
| Peter Li | PT40970 | 1 | 1 |

diabetes_prediction 10 x

Result Grid Form Editor Read Only

6. Determine the number of patients with heart disease.



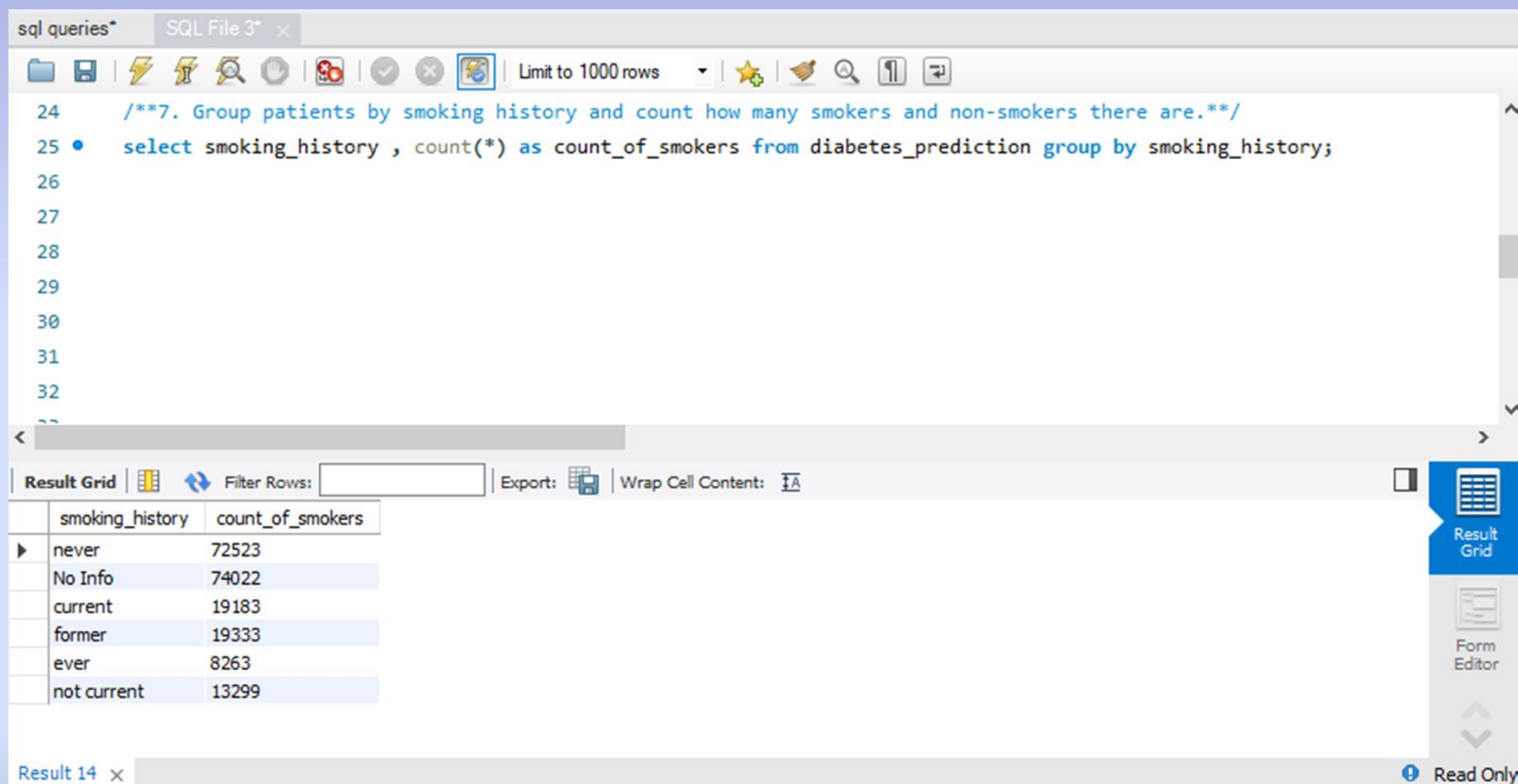
The screenshot shows a SQL IDE interface with a Navigator pane on the left and a main query editor on the right. The Navigator pane displays a tree of schemas, including 'company', 'diabetes_assessment', 'sakila', 'school', 'sys', and 'world'. The 'diabetes_assessment' schema is expanded, showing 'Tables' and 'Views'. The 'diabetes_prediction' table is selected. The main query editor shows a SQL query to count the number of patients with heart disease. The query is as follows:

```
18
19  /**6. Determine the number of patients with heart disease.**/
20  SELECT COUNT(*) AS Number_Of_Patients_With_HeartDisease
21  FROM diabetes_prediction
22  WHERE heart_disease = 1;
23
24
25
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The grid has one column, 'Number_Of_Patients_With_HeartDisease', and one row with the value 8141.

| Number_Of_Patients_With_HeartDisease |
|--------------------------------------|
| 8141 |

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



The screenshot shows a SQL IDE window with a query editor and a result grid. The query editor contains the following SQL code:

```
24  /**7. Group patients by smoking history and count how many smokers and non-smokers there are.**/  
25  •  select smoking_history , count(*) as count_of_smokers from diabetes_prediction group by smoking_history;  
26  
27  
28  
29  
30  
31  
32  
33
```

The result grid displays the following data:

| smoking_history | count_of_smokers |
|-----------------|------------------|
| never | 72523 |
| No Info | 74022 |
| current | 19183 |
| former | 19333 |
| ever | 8263 |
| not current | 13299 |

The IDE interface includes a toolbar with various icons, a "Limit to 1000 rows" dropdown, and a "Result Grid" button on the right. The status bar at the bottom indicates "Result 14" and "Read Only".

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

Navigator: SCHEMAS

Filter objects

- company
- diabetes_assessment
 - Tables
 - diabetes_prediction
 - Views
 - Stored Procedures
 - Functions
- sakila
- school
- sys
- world

Administration Schemas

Information

Table: diabetes_prediction

sql queries* SQL File 3* x

Limit to 1000 rows

```

26
27  /**8. Retrieve the Patient_ids of patients who have a BMI greater than the average BMI.**/
28  • select patient_id , bmi from diabetes_prediction
29      where bmi > (select avg(bmi) from diabetes_prediction);
30
31
32
33
34
35
  
```

Result Grid

| patient_id | bmi |
|------------|-------|
| PT109 | 33.64 |
| PT112 | 54.7 |
| PT113 | 36.05 |
| PT117 | 30.36 |
| PT121 | 36.38 |
| PT124 | 27.94 |
| PT126 | 33.76 |
| PT128 | 27.85 |

diabetes_prediction 15 x

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

sql queries* SQL File 3* x

Limit to 1000 rows

```

30  /**9. Find the patient with the highest HbA1c level and the patient with the lowest HbA1c level.**/
31  select EmployeeName, Patient_id, HbA1c_level,
32  case
33  when HbA1c_level=(select max(HbA1c_level) from diabetes_prediction)
34  then 'Highest'
35  when HbA1c_level=(select min(HbA1c_level) from diabetes_prediction)
36  then 'Highest'
37  end as HbA1c_level_stat
38  from diabetes_prediction
39  where HbA1c_level=(select max(HbA1c_level) from diabetes_prediction) or
40  HbA1c_level=(select min(HbA1c_level) from diabetes_prediction);

```

Result Grid

| EmployeeName | Patient_id | HbA1c_level | HbA1c_level_stat |
|----------------------|------------|-------------|------------------|
| ELLEN MOFFATT | PT120 | 3.5 | Highest |
| JOHN TURSI | PT134 | 3.5 | Highest |
| MICHAEL THOMPSON | PT141 | 9 | Highest |
| SHARON MCCOLE WICHER | PT145 | 3.5 | Highest |
| KEVIN CASHMAN | PT156 | 9 | Highest |
| MARK KEARNEY | PT158 | 3.5 | Highest |
| MONIQUE MOYER | PT174 | 3.5 | Highest |

Table: diabetes_prediction

Columns: EmployeeName text, Patient id text

Result 16 x

Output

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

Navigator: sql queries* SQL File 3* x

SCHEMAS

Filter objects

- company
- diabetes_assessment
 - Tables
 - diabetes_prediction
 - Views
 - Stored Procedures
 - Functions
- sakila
- school
- sys
- world

Limit to 1000 rows

```

42
43 /**11. Rank patients by blood glucose level within each gender group.**/
44 select Patient_id, gender, blood_glucose_level, Rank()
45 over(partition by gender order by blood_glucose_level) as
46 Glucose_Level_Rank from diabetes_prediction;
47
48
49
50
51
52

```

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

| Patient_id | gender | blood_glucose_level | Glucose_Level_Rank |
|------------|--------|---------------------|--------------------|
| PT78487 | Female | 80 | 1 |
| PT80920 | Female | 80 | 1 |
| PT78492 | Female | 80 | 1 |
| PT79735 | Female | 80 | 1 |
| PT77410 | Female | 80 | 1 |
| PT79191 | Female | 80 | 1 |
| PT79182 | Female | 80 | 1 |

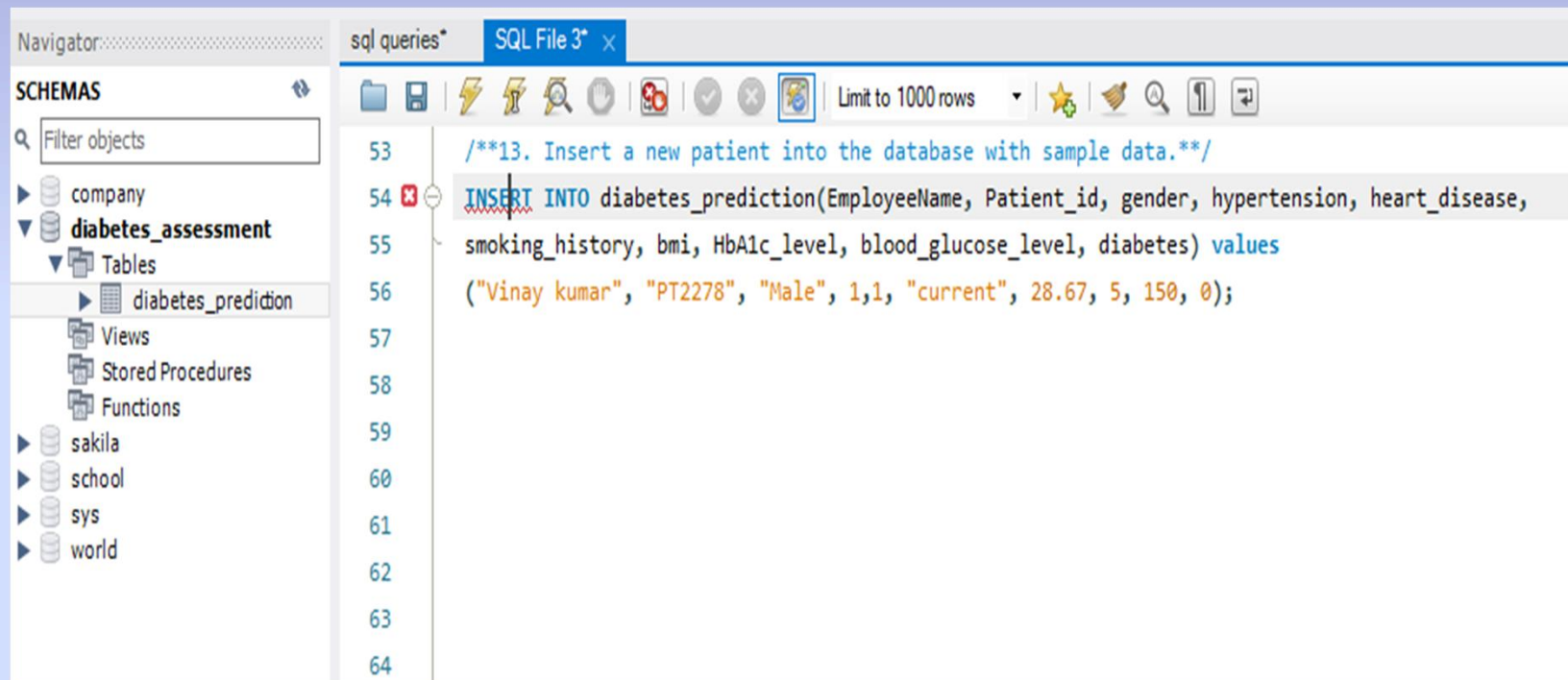
Table: diabetes_prediction

Columns: EmployeeName text

Result 17 x

Read Only

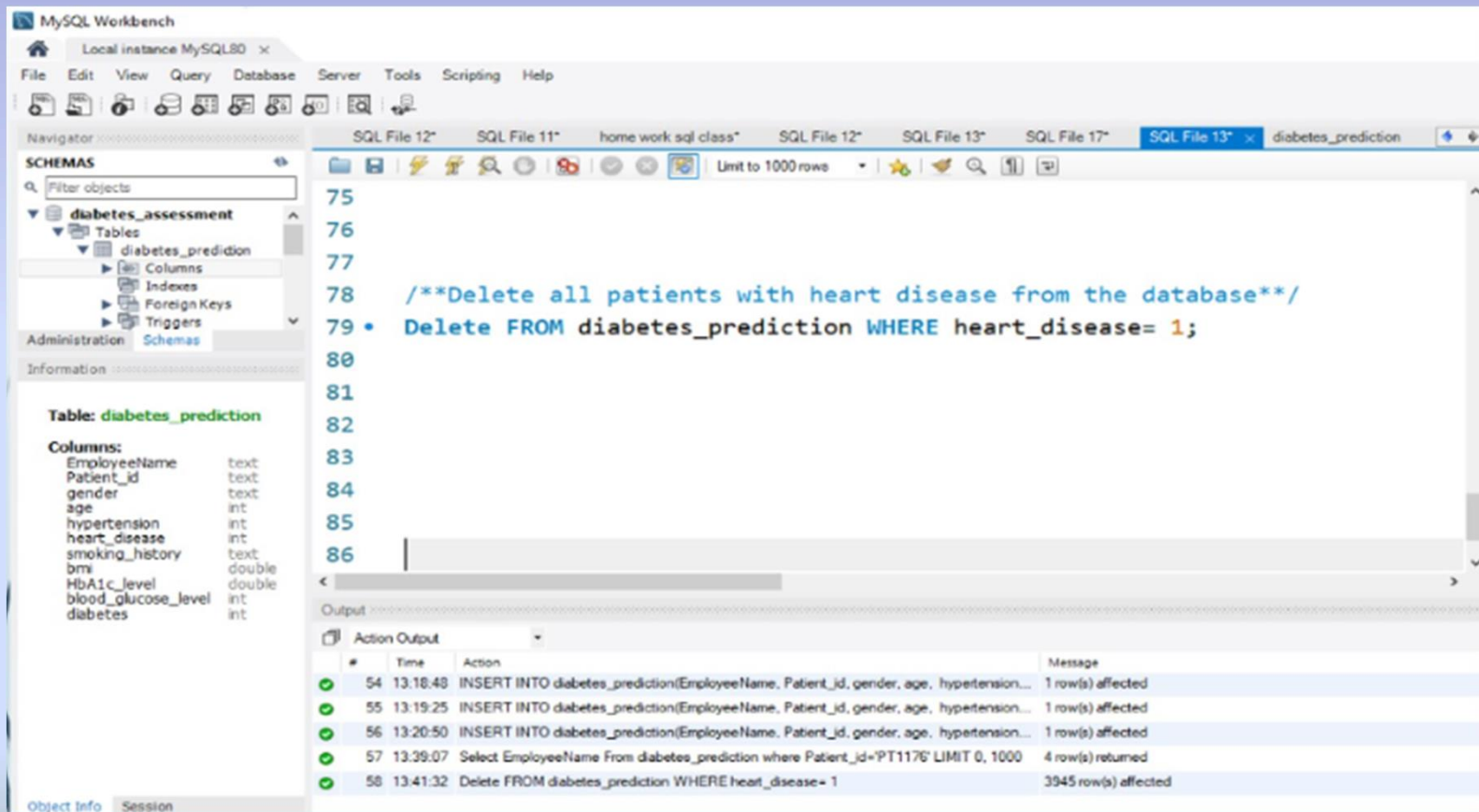
13. Insert a new patient into the database with sample data.



The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view of database objects. The 'diabetes_assessment' schema is expanded, showing a table named 'diabetes_prediction'. The main editor pane shows a SQL query file with the following content:

```
53  /**13. Insert a new patient into the database with sample data.**/  
54  INSERT INTO diabetes_prediction(EmployeeName, Patient_id, gender, hypertension, heart_disease,  
55  smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes) values  
56  ("Vinay kumar", "PT2278", "Male", 1,1, "current", 28.67, 5, 150, 0);  
57  
58  
59  
60  
61  
62  
63  
64
```

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'diabetes_assessment' expanded, showing 'diabetes_prediction' under 'Tables'. The 'Information' tab for 'diabetes_prediction' lists columns: EmployeeName (text), Patient_id (text), gender (text), age (int), hypertension (int), heart_disease (int), smoking_history (text), HbA1c_level (double), blood_glucose_level (double), and diabetes (int). The main editor window shows a SQL query in 'SQL File 13*' named 'diabetes_prediction'.

```

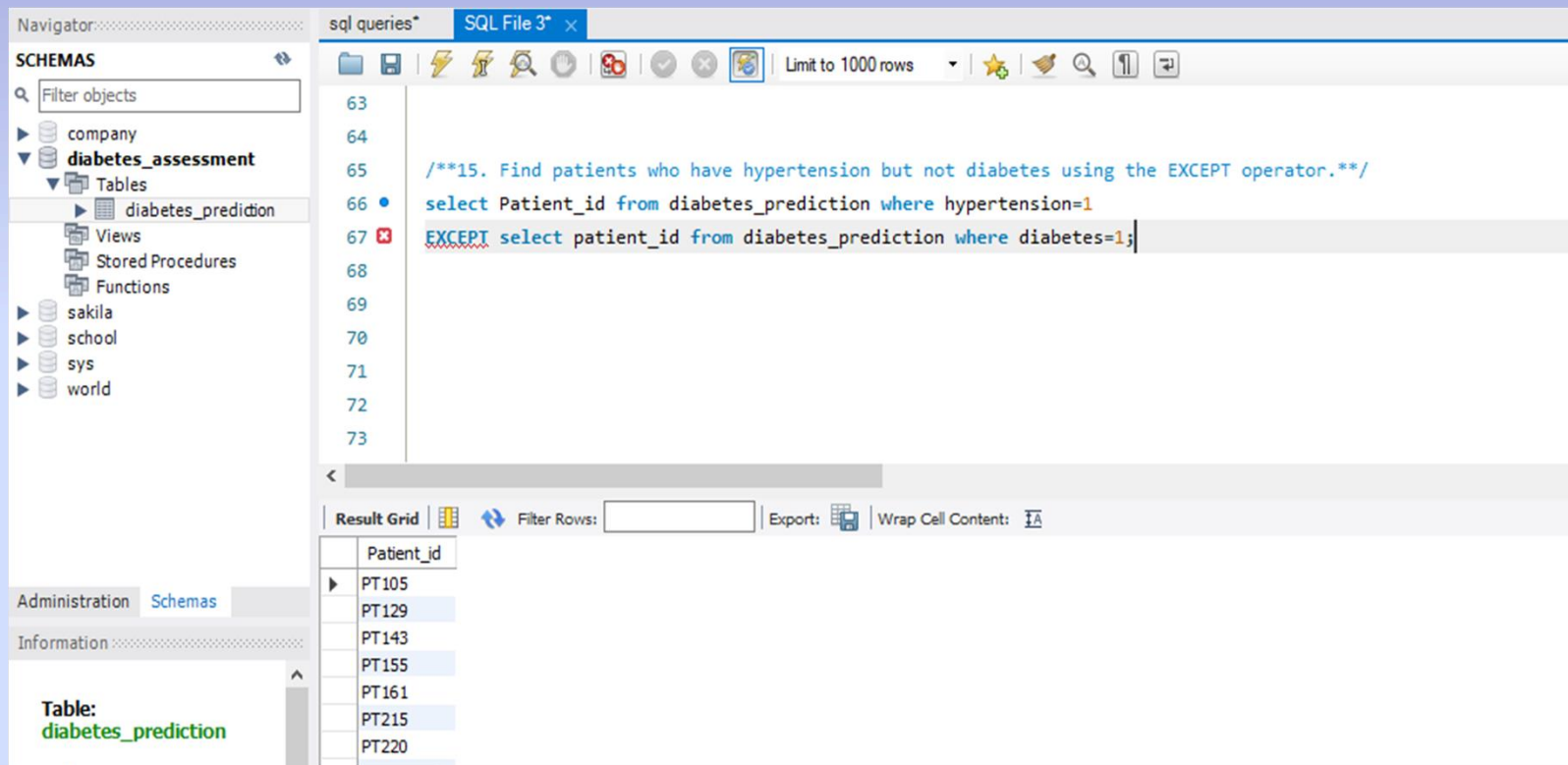
75
76
77
78  /**Delete all patients with heart disease from the database**/
79  • Delete FROM diabetes_prediction WHERE heart_disease= 1;
80
81
82
83
84
85
86

```

The 'Output' tab at the bottom shows the execution results:

| # | Time | Action | Message |
|----|----------|--|----------------------|
| 54 | 13:18:48 | INSERT INTO diabetes_prediction(EmployeeName, Patient_id, gender, age, hypertension... | 1 row(s) affected |
| 55 | 13:19:25 | INSERT INTO diabetes_prediction(EmployeeName, Patient_id, gender, age, hypertension... | 1 row(s) affected |
| 56 | 13:20:50 | INSERT INTO diabetes_prediction(EmployeeName, Patient_id, gender, age, hypertension... | 1 row(s) affected |
| 57 | 13:39:07 | Select EmployeeName From diabetes_prediction where Patient_id='PT1176' LIMIT 0, 1000 | 4 row(s) returned |
| 58 | 13:41:32 | Delete FROM diabetes_prediction WHERE heart_disease= 1 | 3945 row(s) affected |

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

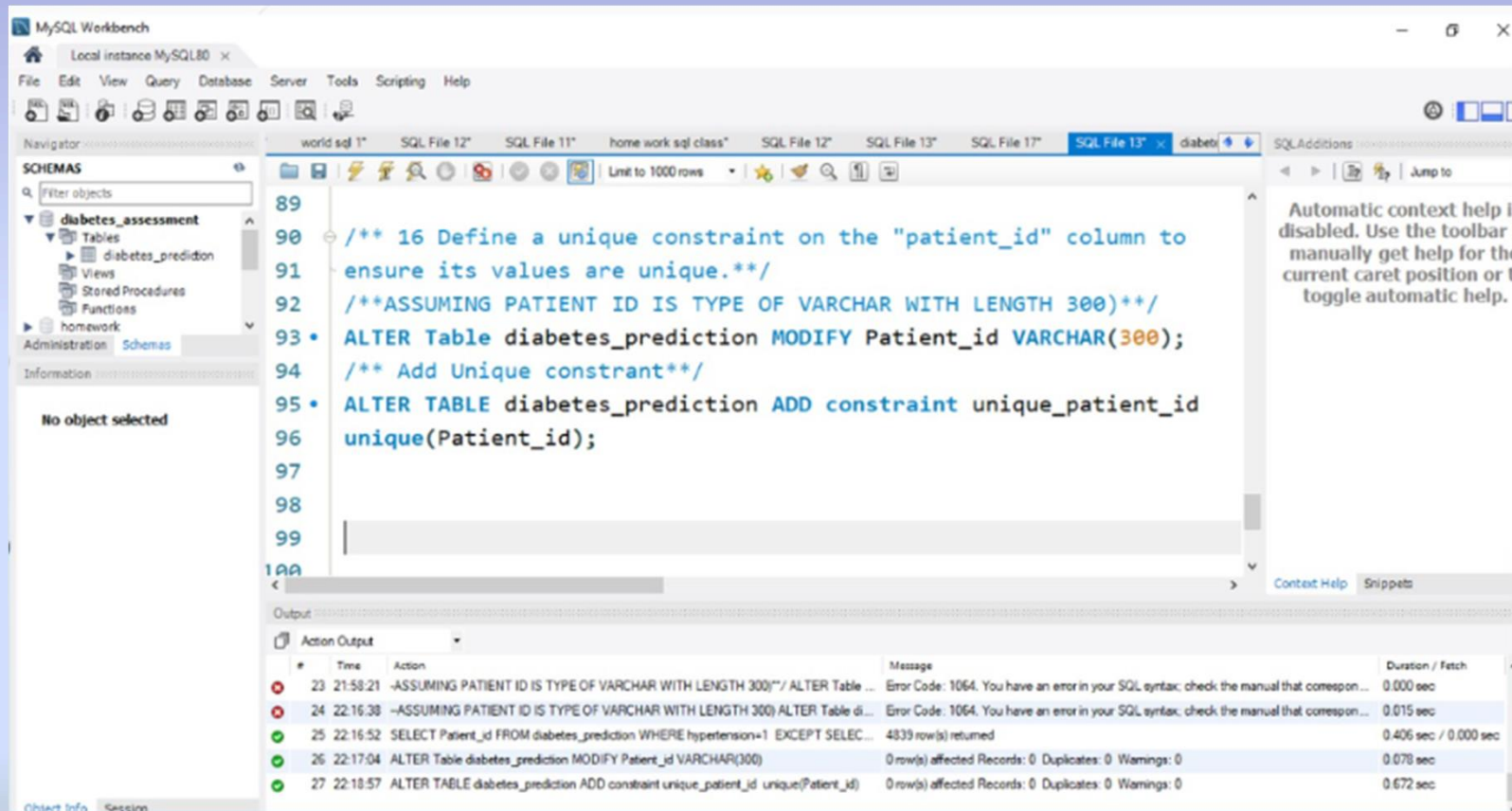


The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view with 'diabetes_prediction' selected under the 'diabetes_assessment' schema. The main editor displays a SQL query using the EXCEPT operator. Below the editor, the 'Result Grid' shows the output of the query, listing patient IDs.

```
/**15. Find patients who have hypertension but not diabetes using the EXCEPT operator.**/  
select Patient_id from diabetes_prediction where hypertension=1  
EXCEPT select patient_id from diabetes_prediction where diabetes=1;
```

| Patient_id |
|------------|
| PT105 |
| PT129 |
| PT143 |
| PT155 |
| PT161 |
| PT215 |
| PT220 |

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'diabetes_assessment' selected. The main editor window shows the following SQL code:

```

89
90 /** 16 Define a unique constraint on the "patient_id" column to
91 ensure its values are unique.**/
92 /**ASSUMING PATIENT ID IS TYPE OF VARCHAR WITH LENGTH 300**/
93 ALTER Table diabetes_prediction MODIFY Patient_id VARCHAR(300);
94 /** Add Unique constraint**/
95 ALTER TABLE diabetes_prediction ADD constraint unique_patient_id
96 unique(Patient_id);
97
98
99
100

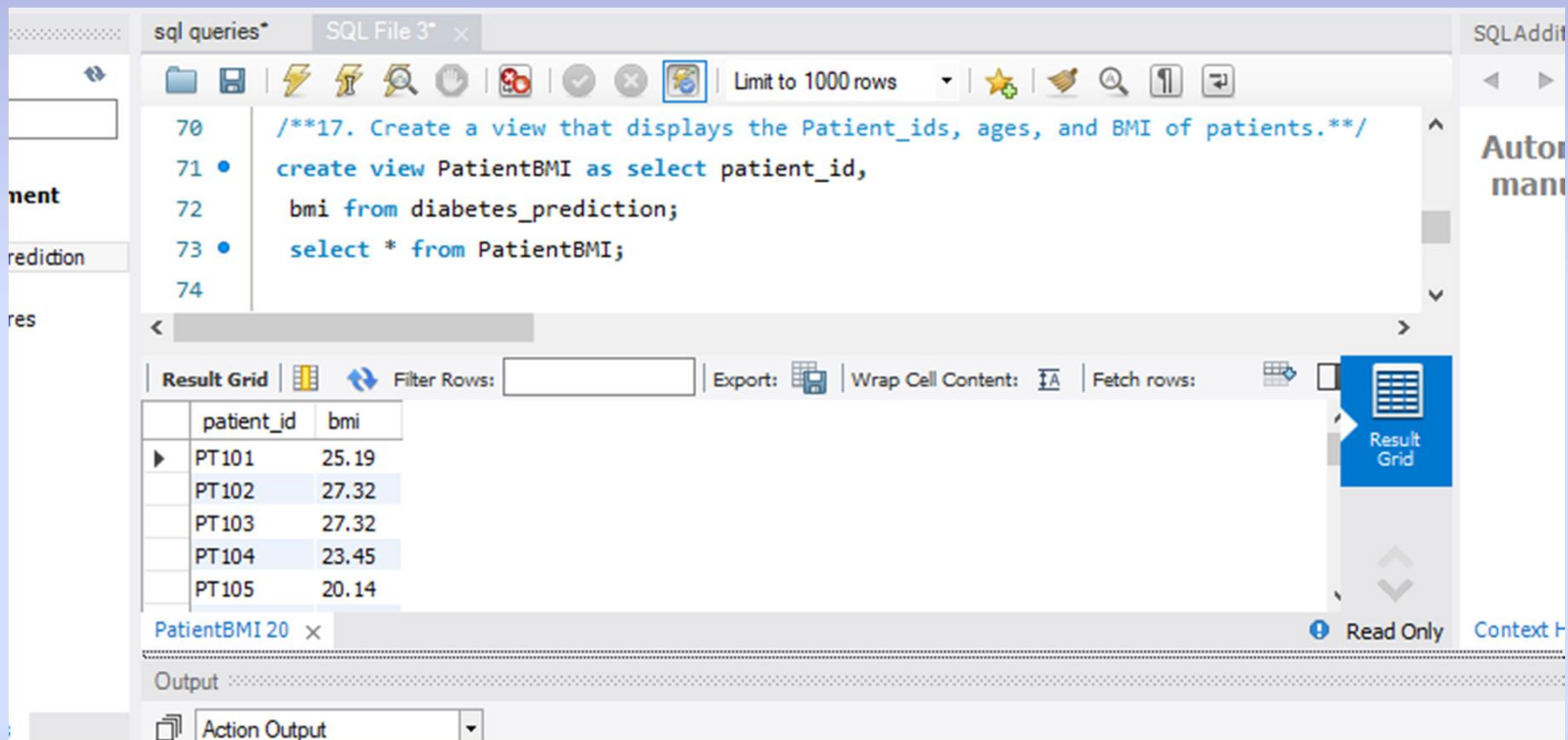
```

The right sidebar shows a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The bottom panel shows the 'Output' window with the following table:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|---|--|-----------------------|
| 23 | 21:58:21 | -ASSUMING PATIENT ID IS TYPE OF VARCHAR WITH LENGTH 300**/ ALTER Table ... | Error Code: 1064. You have an error in your SQL syntax; check the manual that correspon... | 0.000 sec |
| 24 | 22:16:38 | -ASSUMING PATIENT ID IS TYPE OF VARCHAR WITH LENGTH 300) ALTER Table di ... | Error Code: 1064. You have an error in your SQL syntax; check the manual that correspon... | 0.015 sec |
| 25 | 22:16:52 | SELECT Patient_id FROM diabetes_prediction WHERE hypertension=1 EXCEPT SELEC... | 4839 row(s) returned | 0.406 sec / 0.000 sec |
| 26 | 22:17:04 | ALTER Table diabetes_prediction MODIFY Patient_id VARCHAR(300) | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 | 0.078 sec |
| 27 | 22:18:57 | ALTER TABLE diabetes_prediction ADD constraint unique_patient_id unique(Patient_id) | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 | 0.672 sec |

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



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
70  /**17. Create a view that displays the Patient_ids, ages, and BMI of patients.**/  
71  create view PatientBMI as select patient_id,  
72      bmi from diabetes_prediction;  
73  select * from PatientBMI;  
74
```

The result grid displays the following data:

| patient_id | bmi |
|------------|-------|
| PT101 | 25.19 |
| PT102 | 27.32 |
| PT103 | 27.32 |
| PT104 | 23.45 |
| PT105 | 20.14 |

The interface also shows a toolbar with various icons, a "Limit to 1000 rows" dropdown, and a "Result Grid" button. The bottom of the window shows an "Output" section with an "Action Output" dropdown.

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

- To reduce data redundancy and improve data integrity in a database schema, you can consider the following improvements:
- **Normalization:** Normalize the database schema to reduce data redundancy and ensure data integrity. This involves organizing the data into separate tables based on functional dependencies and avoiding repeating groups in a single table.
- **Use of Foreign Keys:** Implement foreign keys to establish relationships between tables. This ensures referential integrity, where data in one table (child table) that references data in another table (parent table) is consistent and accurate.
- **Regular Data Quality Checks:** Implement processes for regular data quality checks and data cleansing.
- **Data Validation Rules:** Define data validation rules to ensure that data entered into the database meets specified criteria.

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

- Enhance query efficiency by abstaining from the use of 'SELECT' and choose columns that really need for better performance.
- Use efficiently the WHERE clause to work with only the necessary records.
- Prefer the use of joins instead of subqueries whenever it's possible.
- use windows to work with the data of interest.
- Use Aggregate functions like Count, Average, to reduce processing time.