

Assignment 2 (DIABETES PREDICATION ASSESSMENT PSYLIQ)

SQL Query

```
select * from diabetes_prediction;
```

1. Retrieve the Patient_id and ages of all patients.

1. Add the age column

```
ALTER TABLE diabetes_prediction
```

```
ADD COLUMN age INT;
```

-- 2. Disable safe update mode

```
SET SQL_SAFE_UPDATES = 0;
```

-- 3. Calculate and update the age column

```
UPDATE diabetes_prediction
```

```
SET age = YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(CONCAT(Year, '-12-31'), '%m%d'));
```

-- 4. Re-enable safe update mode

```
SET SQL_SAFE_UPDATES = 1;
```

```
SELECT Patient_id, Year, age
```

```
FROM diabetes_prediction
```

```
LIMIT 10;
```

```
SELECT Patient_id,
```

```
    YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(CONCAT(Year, '-12-31'), '%m%d')) AS age
```

```
FROM diabetes_prediction;
```

2. Select all female patients who are older than 30

```
SELECT Patient_id, gender,
```

```
    YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(CONCAT(Year))) AS age
```

```
FROM diabetes_prediction
```

```
WHERE gender = 'female' AND
```

```
    YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(CONCAT(Year))) > 30;
```

3. Calculate the average BMI of patients

```
SELECT AVG(bmi) AS average_bmi FROM diabetes_prediction;
```

4. List patients in descending order of blood glucose levels

```
SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC;
```

5. Find patients who have hypertension and diabetes

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

6. Determine the number of patients with heart disease

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

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

```
SELECT smoking_history, COUNT(*) AS count FROM diabetes_prediction GROUP BY smoking_history;
```

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

```
SELECT Patient_id FROM diabetes_prediction WHERE bmi > (SELECT AVG(bmi) FROM diabetes_prediction);
```

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

-- Highest HbA1c level

```
SELECT Patient_id, HbA1c_level  
FROM diabetes_prediction ORDER BY HbA1c_level DESC LIMIT 1;
```

-- Lowest HbA1c level

```
SELECT Patient_id, HbA1c_level FROM diabetes_prediction ORDER BY HbA1c_level ASC LIMIT 1;
```

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

```
SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age FROM diabetes_prediction;
```

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

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

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

```
UPDATE diabetes_prediction
SET smoking_history = 'Ex-smoker'
WHERE (YEAR(CURRENT_DATE) - Year) > 40;
```

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

```
INSERT INTO diabetes_prediction (Patient_id, gender, Year, hypertension, heart_disease, smoking_history, bmi, HbA1c_level,
blood_glucose_level, diabetes)
VALUES ('PT999', 'Male', 1985, 0, 0, 'never', 22.5, 5.5, 90, 0);
```

14. Delete all patients with heart disease from the database

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

#15. Find patients who have hypertension but not diabetes using the EXCEPT operator MySQL does not support the EXCEPT operator directly. Instead, we can use a subquery.

```
SELECT Patient_id
FROM diabetes_prediction
WHERE hypertension = 1
AND Patient_id NOT IN (
    SELECT Patient_id
    FROM diabetes_prediction
    WHERE diabetes = 1
);
```

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

-- Step 1: Check for Duplicate Patient_ids

```
SELECT Patient_id, COUNT(*)
FROM diabetes_prediction
GROUP BY Patient_id
HAVING COUNT(*) > 1;
```

-- If duplicates are found, proceed to Step 2

-- Step 2: Remove or Handle Duplicates

-- Option A: Delete duplicates, keeping only the first occurrence

```

DELETE dp1

FROM diabetes_prediction dp1

INNER JOIN diabetes_prediction dp2

WHERE dp1.Patient_id = dp2.Patient_id

AND dp1.rowid > dp2.rowid;

-- Option B: Update duplicates to make them unique
-- (Uncomment if you prefer updating instead of deleting)
-- UPDATE diabetes_prediction dp1
-- INNER JOIN (
--     SELECT Patient_id, rowid
--     FROM diabetes_prediction
--     WHERE Patient_id IN (
--         SELECT Patient_id
--         FROM diabetes_prediction
--         GROUP BY Patient_id
--         HAVING COUNT(*) > 1
--     )
-- ) dp2 ON dp1.rowid = dp2.rowid
-- SET dp1.Patient_id = CONCAT(dp1.Patient_id, '_', dp1.rowid);

-- Step 3: Add the Unique Constraint
ALTER TABLE diabetes_prediction
ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id);

```

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

```

CREATE VIEW patient_info AS

SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age, bmi

FROM diabetes_prediction;

SET SQL_SAFE_UPDATES = 1;

```

0) Display Data

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'internship' and 'rtkdatabase' selected. The 'rtkdatabase' schema is expanded, showing tables like 'diabetes_prediction'. The main editor contains a SQL query:

```

1 select * from diabetes_prediction;
2
3 # 1. Retrieve the Patient_id and ages of all patients.
4 # SELECT Patient_id, YEAR(CURDATE()) - YEAR("Year") AS age FROM diabetes_prediction;
5
6 # 2. Select all female patients who are older than 30
7 # SELECT Patient_id, gender, YEAR(CURDATE()) - YEAR("D.O.B") AS age FROM diabetes_prediction WHERE gender = 'female' AND YEAR(CURDATE()) - YEAR("D.O.B") > 30;

```

The 'Result Grid' at the bottom displays the query results for the first query. The columns are: InEmployment, Patient_id, gender, hypertension, heart_disease, smoking_history, bmi, fbsA1c_level, blood_glucose_level, diabetes, and Year. The data shows 24 rows of patient information.

1) Retrieve the Patient_id and ages of all patients.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'rtkdatabase' selected. The main editor contains a SQL query:

```

22 SET age = YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m-%d') < DATE_FORMAT(CONCAT(Year, '-12-31'), '%m-%d'));
23
24 -- 4. Re-enable safe update mode
25 SET SQL_SAFE_UPDATES = 0;
26
27 SELECT Patient_id, Year, age
28 FROM diabetes_prediction
29 LIMIT 10;
30
31 SELECT Patient_id,
32       YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m-%d') < DATE_FORMAT(CONCAT(Year, '-12-31'), '%m-%d')) AS age
33 FROM diabetes_prediction;
34

```

The 'Result Grid' at the bottom displays the query results for the first query. The columns are: Patient_id, Year, and age. The data shows 10 rows of patient information.

The 'Action Output' pane at the bottom shows the execution results of the queries:

Time	Action	Message	Duration / Patch
24 23:08:30	UPDATE diabetes_prediction SET age = YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m-%d') < DATE_FORMAT(CONCAT(Year, '-12-31'), '%m-%d'))	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that use...	0.016 sec
25 23:09:26	SET SQL_SAFE_UPDATES = 0	0 rows affected	0.000 sec
26 23:10:03	UPDATE diabetes_prediction SET age = YEAR(CURDATE()) - Year - (DATE_FORMAT(CURDATE(), '%m-%d') < DATE_FORMAT(CONCAT(Year, '-12-31'), '%m-%d'))	0 rows affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
27 23:10:12	SELECT Patient_id, Year, age FROM diabetes_prediction LIMIT 10	0 rows returned	0.000 sec / 0.000 sec
28 23:11:36	SELECT Patient_id, Year, age FROM diabetes_prediction LIMIT 10	0 rows returned	0.000 sec / 0.000 sec
29 23:12:28	SELECT Patient_id, Year, age FROM diabetes_prediction LIMIT 10	0 rows returned	0.000 sec / 0.000 sec

3) Calculate the average BMI of patients.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema for 'mydatabase', including tables like 'diabetes_prediction'. The main editor contains the following SQL queries:

```
27 * SELECT Patient_id, gender,
28 * YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) AS age
29 * FROM diabetes_prediction
30 * WHERE gender = 'female' AND
31 * YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) > 30;
32
33 * 3. Calculate the average BMI of patients
34 * SELECT AVG(bmi) AS average_bmi
35 * FROM diabetes_prediction;
36
37 * 4. List patients in descending order of blood glucose levels
38 * SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC;
```

The 'Result Grid' shows the output of the third query, displaying a single row with the average BMI value: 28.88.

The 'Action Output' pane shows the execution log with the following messages:

- 30 23:13:15 SELECT Patient_id, YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) AS age FROM diabetes_prediction WHERE gender = 'female' AND YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) > 30; 0 rows(s) returned
- 31 23:13:24 SELECT Patient_id, gender, YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) AS age FROM diabetes_prediction WHERE gender = 'female' AND YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) > 30; 0 rows(s) returned
- 32 23:14:13 SELECT Patient_id, gender, YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) AS age FROM diabetes_prediction WHERE gender = 'female' AND YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) > 30; 0 rows(s) returned
- 33 23:16:21 SELECT Patient_id, gender, YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) AS age FROM diabetes_prediction WHERE gender = 'female' AND YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) > 30; 0 rows(s) returned
- 34 23:16:25 SELECT AVG(bmi) AS average_bmi FROM diabetes_prediction LIMIT 0, 1000; 1 row(s) returned
- 35 23:16:59 SELECT AVG(bmi) AS average_bmi FROM diabetes_prediction LIMIT 0, 1000; 1 row(s) returned

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema for 'mydatabase', including tables like 'diabetes_prediction'. The main editor contains the following SQL queries:

```
35 * SELECT AVG(bmi) AS average_bmi FROM diabetes_prediction;
36
37 * 4. List patients in descending order of blood glucose levels
38 * SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC;
39
40 * 5. Find patients who have hypertension and diabetes
41 * SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND diabetes = 1;
42
43 * 6. Determine the number of patients with heart disease
44 * SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1;
45
46 * 7. Group patients by smoking history and count how many smokers and non-smokers there are
47 * SELECT smoking_history, COUNT(*) AS count FROM diabetes_prediction GROUP BY smoking_history;
```

The 'Result Grid' shows the output of the fourth query, displaying a single row with the patient ID and blood glucose level: 28.88.

The 'Action Output' pane shows the execution log with the following messages:

- 42 23:30:40 SELECT Patient_id, (YEAR(CURDATE()) - Year) AS Age FROM diabetes_prediction LIMIT 0, 1000; 0 rows(s) returned
- 43 23:33:41 SELECT Patient_id, (YEAR(CURDATE()) - Year) AS Age FROM diabetes_prediction; 0 rows(s) returned
- 44 23:34:04 SELECT Patient_id, (YEAR(CURDATE()) - Year) AS Age FROM diabetes_prediction; 0 rows(s) returned
- 45 23:34:43 select age from diabetes_prediction; 0 rows(s) returned
- 46 23:35:13 SELECT Patient_id, gender, YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) AS age FROM diabetes_prediction WHERE gender = 'female' AND YEAR(CURDATE()) - Year = (DATE_FORMAT(CURDATE(), '%u%Y')) < DATE_FORMAT(CONCAT(Year)) > 30; 0 rows(s) returned
- 47 23:35:27 SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC; 0 rows(s) returned

5) Find patients who have hypertension and diabetes.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema, including the `diabetes_prediction` table. The main editor contains a series of SQL queries. Query 41, `SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND diabetes = 1;`, is highlighted. The 'Result Grid' shows the results of query 40, displaying `Patient_id` and `blood_glucose_level`. The 'Action Output' pane at the bottom shows the execution progress of the queries, with query 41 successfully returning 0 rows.

```
25 * SELECT AVG(bmi) AS average_bmi FROM diabetes_prediction;
26
27 * 4. List patients in descending order of blood glucose levels
28 * SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC;
29
30
31 * 5. Find patients who have hypertension and diabetes
32 * SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND diabetes = 1;
33
34
35 * 6. Determine the number of patients with heart disease
36 * SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1;
37
38
39 * 7. Group patients by smoking history and count how many smokers and non-smokers there are
40 * SELECT smoking_history, COUNT(*) AS count FROM diabetes_prediction GROUP BY smoking_history;
```

Result Grid:

Patient_id	blood_glucose_level
------------	---------------------

Action Output:

#	Time	Action	Message	Duration / Fetch
40	23:34:43	select avg from diabetes_prediction	0 rows returned	0.000 sec / 0.000 sec
41	23:35:13	SELECT Patient_id, gender, YEAR(CURDATE()) - YEAR(DATE_FORMAT(CURDATE(), '%m/%d')) AS DAT...	Error Code: 1552. Incorrect parameter count in the call to native function 'DATE_FORMAT'	0.000 sec
42	23:35:27	SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC	0 rows returned	0.000 sec / 0.000 sec
43	23:36:03	SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND diabetes = 1	0 rows returned	0.000 sec / 0.000 sec
44	23:37:50	SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1	1 rows returned	0.016 sec / 0.000 sec
45	23:38:40	SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC	0 rows returned	0.000 sec / 0.000 sec

6) Determine the number of patients with heart disease.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema, including the `diabetes_prediction` table. The main editor contains a series of SQL queries. Query 44, `SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1;`, is highlighted. The 'Result Grid' shows the results of query 44, displaying `heart_disease_count` with a value of 1. The 'Action Output' pane at the bottom shows the execution progress of the queries, with query 44 successfully returning 1 row.

```
35 * SELECT AVG(bmi) AS average_bmi FROM diabetes_prediction;
36
37 * 4. List patients in descending order of blood glucose levels
38 * SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC;
39
40
41 * 5. Find patients who have hypertension and diabetes
42 * SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND diabetes = 1;
43
44
45 * 6. Determine the number of patients with heart disease
46 * SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1;
47
48
49 * 7. Group patients by smoking history and count how many smokers and non-smokers there are
50 * SELECT smoking_history, COUNT(*) AS count FROM diabetes_prediction GROUP BY smoking_history;
```

Result Grid:

heart_disease_count
1

Action Output:

#	Time	Action	Message	Duration / Fetch
44	23:34:04	SELECT Patient_id, (2024 - Year) AS Age FROM diabetes_prediction	0 rows returned	0.000 sec / 0.000 sec
45	23:34:43	select avg from diabetes_prediction	0 rows returned	0.000 sec / 0.000 sec
46	23:35:13	SELECT Patient_id, gender, YEAR(CURDATE()) - YEAR(DATE_FORMAT(CURDATE(), '%m/%d')) AS DAT...	Error Code: 1552. Incorrect parameter count in the call to native function 'DATE_FORMAT'	0.000 sec
47	23:35:27	SELECT Patient_id, blood_glucose_level FROM diabetes_prediction ORDER BY blood_glucose_level DESC	0 rows returned	0.000 sec / 0.000 sec
48	23:36:03	SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND diabetes = 1	0 rows returned	0.000 sec / 0.000 sec
49	23:37:50	SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1	1 rows returned	0.016 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. The 'Schemas' pane on the left displays the 'internship' database, with the 'diabetes_prediction' table selected. The 'Table: diabetes_prediction' pane shows columns: Patient_id, gender, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes, and year. The 'Query' editor contains the following SQL code:

```

43 # 6. Determine the number of patients with heart disease
44 SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1;
45
46 #7. Group patients by smoking history and count how many smokers and non-smokers there are
47 SELECT smoking_history, COUNT(*) AS count FROM diabetes_prediction GROUP BY smoking_history;
48
49 # 8. Retrieve the Patient_id of patients who have a BMI greater than the average BMI
50 SELECT Patient_id FROM diabetes_prediction WHERE bmi > (SELECT AVG(bmi) FROM diabetes_prediction);
51
52 #9. Find the patient with the highest HbA1c level and the patient with the lowest HbA1c level.
53 -- Highest HbA1c level
54 SELECT Patient_id, HbA1c_level
55 FROM diabetes_prediction ORDER BY HbA1c_level DESC LIMIT 1;

```

The 'Result Grid' shows the output of the query, with columns 'smoking_history' and 'count'. The 'Action Output' pane shows the execution progress of the queries.

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

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left displays the 'internship' database, with the 'diabetes_prediction' table selected. The 'Table: diabetes_prediction' pane shows columns: Patient_id, gender, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes, and year. The 'Query' editor contains the following SQL code:

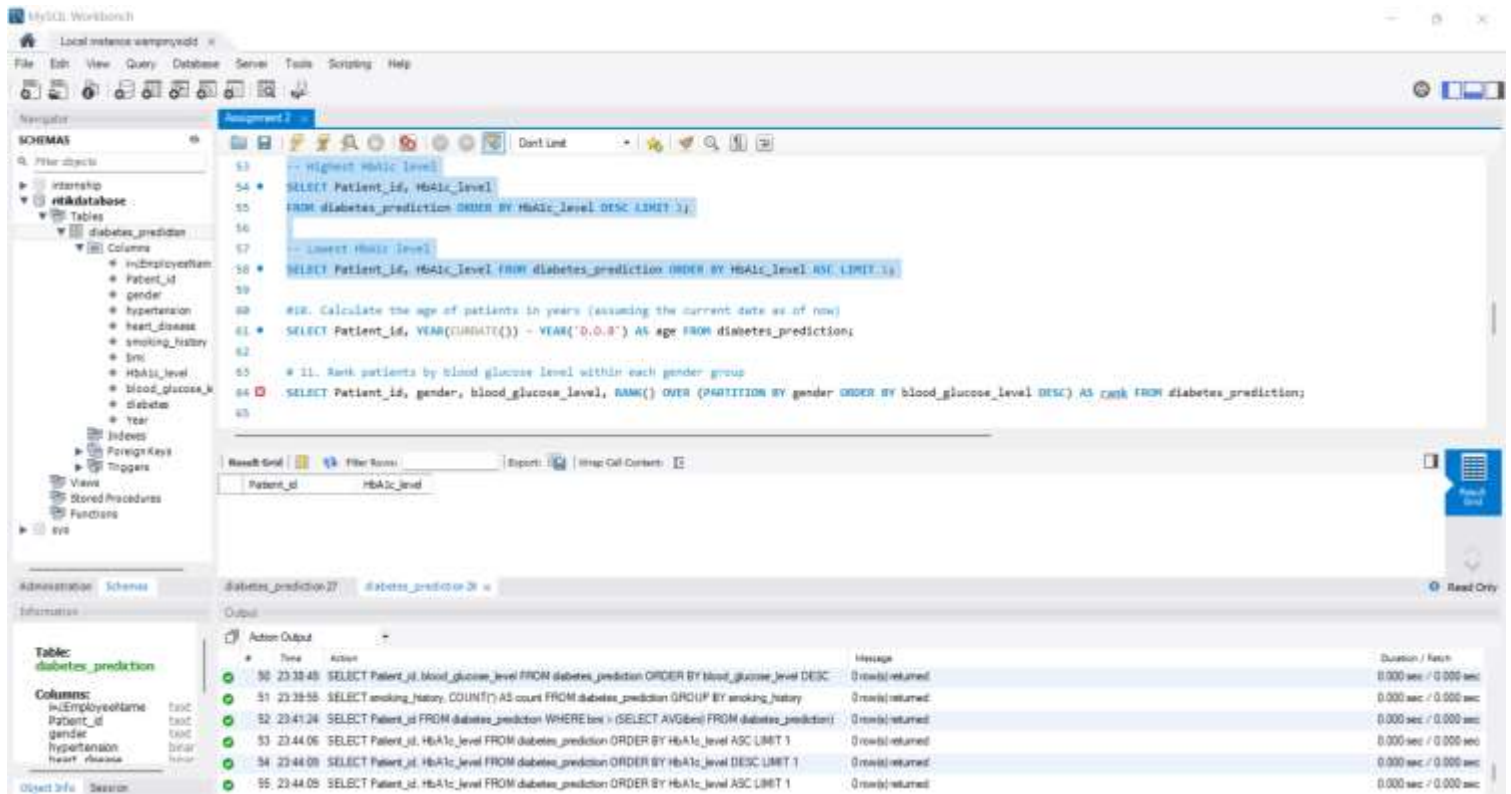
```

43 # 6. Determine the number of patients with heart disease
44 SELECT COUNT(*) AS heart_disease_count FROM diabetes_prediction WHERE heart_disease = 1;
45
46 #7. Group patients by smoking history and count how many smokers and non-smokers there are
47 SELECT smoking_history, COUNT(*) AS count FROM diabetes_prediction GROUP BY smoking_history;
48
49 # 8. Retrieve the Patient_id of patients who have a BMI greater than the average BMI
50 SELECT Patient_id FROM diabetes_prediction WHERE bmi > (SELECT AVG(bmi) FROM diabetes_prediction);
51
52 #9. Find the patient with the highest HbA1c level and the patient with the lowest HbA1c level.
53 -- Highest HbA1c level
54 SELECT Patient_id, HbA1c_level
55 FROM diabetes_prediction ORDER BY HbA1c_level DESC LIMIT 1;

```

The 'Result Grid' shows the output of the query, with columns 'Patient_id'. The 'Action Output' pane shows the execution progress of the queries.

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema for 'mydatabase', including tables like 'diabetes_prediction'. The main editor window contains the following SQL queries:

```

-- Highest HbA1c level
SELECT Patient_id, HbA1c_level
FROM diabetes_prediction ORDER BY HbA1c_level DESC LIMIT 1;

-- Lowest HbA1c level
SELECT Patient_id, HbA1c_level FROM diabetes_prediction ORDER BY HbA1c_level ASC LIMIT 1;

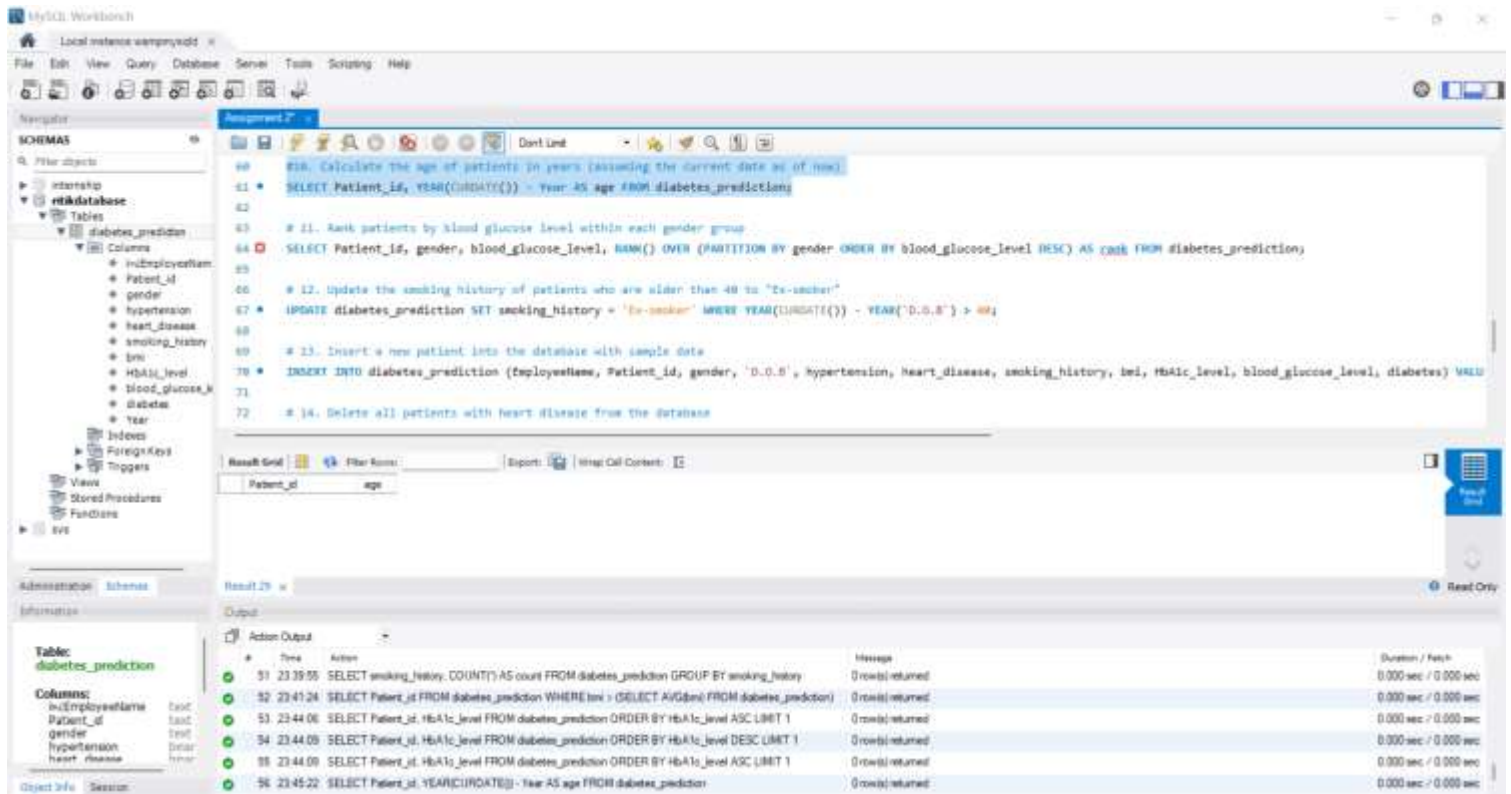
-- 10. Calculate the age of patients in years (assuming the current date as of now)
SELECT Patient_id, YEAR(CURRENT_TIMESTAMP()) - YEAR('D.O.B') AS age FROM diabetes_prediction;

-- 11. Rank patients by blood glucose level within each gender group
SELECT Patient_id, gender, blood_glucose_level, RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level DESC) AS rank FROM diabetes_prediction;

```

The bottom panel shows the 'Action Output' table with columns: Time, Action, Message, and Duration / Fetch. It lists the execution of the above queries, all of which completed successfully.

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema for 'mydatabase'. The main editor window contains the following SQL queries:

```

-- 10. Calculate the age of patients in years (assuming the current date as of now)
SELECT Patient_id, YEAR(CURRENT_TIMESTAMP()) - YEAR('D.O.B') AS age FROM diabetes_prediction;

-- 11. Rank patients by blood glucose level within each gender group
SELECT Patient_id, gender, blood_glucose_level, RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level DESC) AS rank FROM diabetes_prediction;

-- 12. Update the smoking history of patients who are older than 40 to "ex-smoker"
UPDATE diabetes_prediction SET smoking_history = 'ex-smoker' WHERE YEAR(CURRENT_TIMESTAMP()) - YEAR('D.O.B') > 40;

-- 13. Insert a new patient into the database with sample data
INSERT INTO diabetes_prediction (employee_id, Patient_id, gender, 'D.O.B', hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes) VALUES (...);

-- 14. Delete all patients with heart disease from the database
DELETE FROM diabetes_prediction WHERE heart_disease = 1;

```

The bottom panel shows the 'Action Output' table with columns: Time, Action, Message, and Duration / Fetch. It lists the execution of the above queries, all of which completed successfully.

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema for 'interkorp' and 'retadatabase', with the 'diabetes_prediction' table selected. The main editor shows a series of SQL queries. The 'Result Grid' at the bottom displays the output of the queries, showing columns: Patient_id, gender, blood_glucose_level, and RankOfPatient.

```

-- #18. Calculate the age of patients in years (assuming the current date as of now)
-- SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age FROM diabetes_prediction;

-- #21. Rank patients by blood glucose level within each gender group
-- SELECT Patient_id, gender, blood_glucose_level,
-- RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level DESC) AS RankOfPatient
-- FROM diabetes_prediction;

-- #12. Update the smoking history of patients who are older than 40 to "Ex-smoker"
-- UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE YEAR(CURRENT_DATE) - YEAR('D.O.B') > 40;

-- #13. Insert a new patient into the database with sample data
-- INSERT INTO diabetes_prediction (EmployeeName, Patient_id, gender, 'D.O.B', hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes) VALUES

```

The 'Result Grid' shows the following data:

Patient_id	gender	blood_glucose_level	RankOfPatient
53	23-44-06	SELECT Patient_id, HbA1c_level FROM diabetes_prediction ORDER BY HbA1c_level ASC LIMIT 1	0 rows returned
54	23-44-06	SELECT Patient_id, HbA1c_level FROM diabetes_prediction ORDER BY HbA1c_level DESC LIMIT 1	0 rows returned
55	23-44-06	SELECT Patient_id, HbA1c_level FROM diabetes_prediction ORDER BY HbA1c_level ASC LIMIT 1	0 rows returned
56	23-45-22	SELECT Patient_id, YEAR(CURRENT_DATE) - Year AS age FROM diabetes_prediction	0 rows returned
57	23-46-41	SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age FROM diabetes_prediction	0 rows returned
58	23-50-24	SELECT Patient_id, gender, blood_glucose_level, RANK() OVER (PARTITION BY gender ORDER BY blood_...	0 rows returned

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema for 'interkorp' and 'retadatabase', with the 'diabetes_prediction' table selected. The main editor shows a series of SQL queries. The 'Result Grid' at the bottom displays the output of the queries, showing columns: Time, Action, Message, and Duration / Fetch.

```

-- #18. Calculate the age of patients in years (assuming the current date as of now)
-- SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age FROM diabetes_prediction;

-- #21. Rank patients by blood glucose level within each gender group
-- SELECT Patient_id, gender, blood_glucose_level,
-- RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level DESC) AS RankOfPatient
-- FROM diabetes_prediction;

-- #12. Update the smoking history of patients who are older than 40 to "Ex-smoker"
-- UPDATE diabetes_prediction
-- SET smoking_history = 'Ex-smoker'
-- WHERE (YEAR(CURRENT_DATE) - Year) > 40;

-- #13. Insert a new patient into the database with sample data
-- INSERT INTO diabetes_prediction (EmployeeName, Patient_id, gender, 'D.O.B', hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes) VALUES

-- #14. Delete all patients with heart disease from the database
-- DELETE FROM diabetes_prediction WHERE heart_disease = 1;

-- #15. Find patients who have hypertension but not diabetes using the EXCEPT operator
-- MySQL does not support the EXCEPT operator directly. Instead, we can use a subquery.
-- SELECT Patient_id

```

The 'Result Grid' shows the following data:

Time	Action	Message	Duration / Fetch
56	23-45-22	SELECT Patient_id, YEAR(CURRENT_DATE) - Year AS age FROM diabetes_prediction	0 rows returned
57	23-46-41	SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age FROM diabetes_prediction	0 rows returned
58	23-50-24	SELECT Patient_id, gender, blood_glucose_level, RANK() OVER (PARTITION BY gender ORDER BY blood_...	0 rows returned
59	23-51-20	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE YEAR(CURRENT_DATE) - YEAR('D.O.B') > 40	Error Code: 1054 Unknown column 'D.O.B' in 'where clause'
60	23-51-42	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) > 40	0 rows affected Rows matched: 0 Changed: 0 Warnings: 0
61	23-51-45	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) > 40	0 rows affected Rows matched: 0 Changed: 0 Warnings: 0

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

The screenshot shows the MySQL Workbench interface with the following SQL queries in the editor:

```

60 #18. Calculate the age of patients in years (assuming the current date as of now)
61 * SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age FROM diabetes_prediction;
62
63 # 21. Rank patients by blood glucose level within each gender group
64 * SELECT Patient_id, gender, blood_glucose_level,
65 RANK() OVER (PARTITION BY gender ORDER BY blood_glucose_level DESC) AS RankOfPatient
66 FROM diabetes_prediction;
67
68 # 12. Update the smoking history of patients who are older than 40 to "Ex-smoker"
69 * UPDATE diabetes_prediction
70 SET smoking_history = "Ex-smoker"
71 WHERE (YEAR(CURRENT_DATE) - Year) > 40;
72
73 # 13. Insert a new patient into the database with sample data
74 * INSERT INTO diabetes_prediction (Patient_id, gender, Year, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes)
75 VALUES ('P1000', 'Male', 1980, 0, 0, 'never', 22.5, 5.7, 90, 0);
76
77 # 14. Delete all patients with heart disease from the database
78 * DELETE FROM diabetes_prediction WHERE heart_disease = 1;
79
80 #15. Find patients who have hypertension but not diabetes using the EXCEPT operator. MySQL does not support the EXCEPT operator directly. Instead, we can use a subquery.

```

The Output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
57	23:45:41	SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age FROM diabetes_prediction	0 rows returned	0.000 sec / 0.000 sec
58	23:50:24	SELECT Patient_id, gender, blood_glucose_level, RANK() OVER (PARTITION BY gender ORDER BY blood_...	0 rows returned	0.000 sec / 0.000 sec
59	23:51:20	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE YEAR(CURRENT_DATE) - YEAR(D.O.B.	Error Code: 1054 Unknown column 'D.O.B' in 'where clause'	0.000 sec
60	23:51:42	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) ...	0 rows affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
61	23:51:45	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) ...	0 rows affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
62	23:52:47	INSERT INTO diabetes_prediction (Patient_id, gender, Year, hypertension, heart_disease, smoking_history, b...	1 row(s) affected	0.000 sec

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

The screenshot shows the MySQL Workbench interface with the following SQL queries in the editor:

```

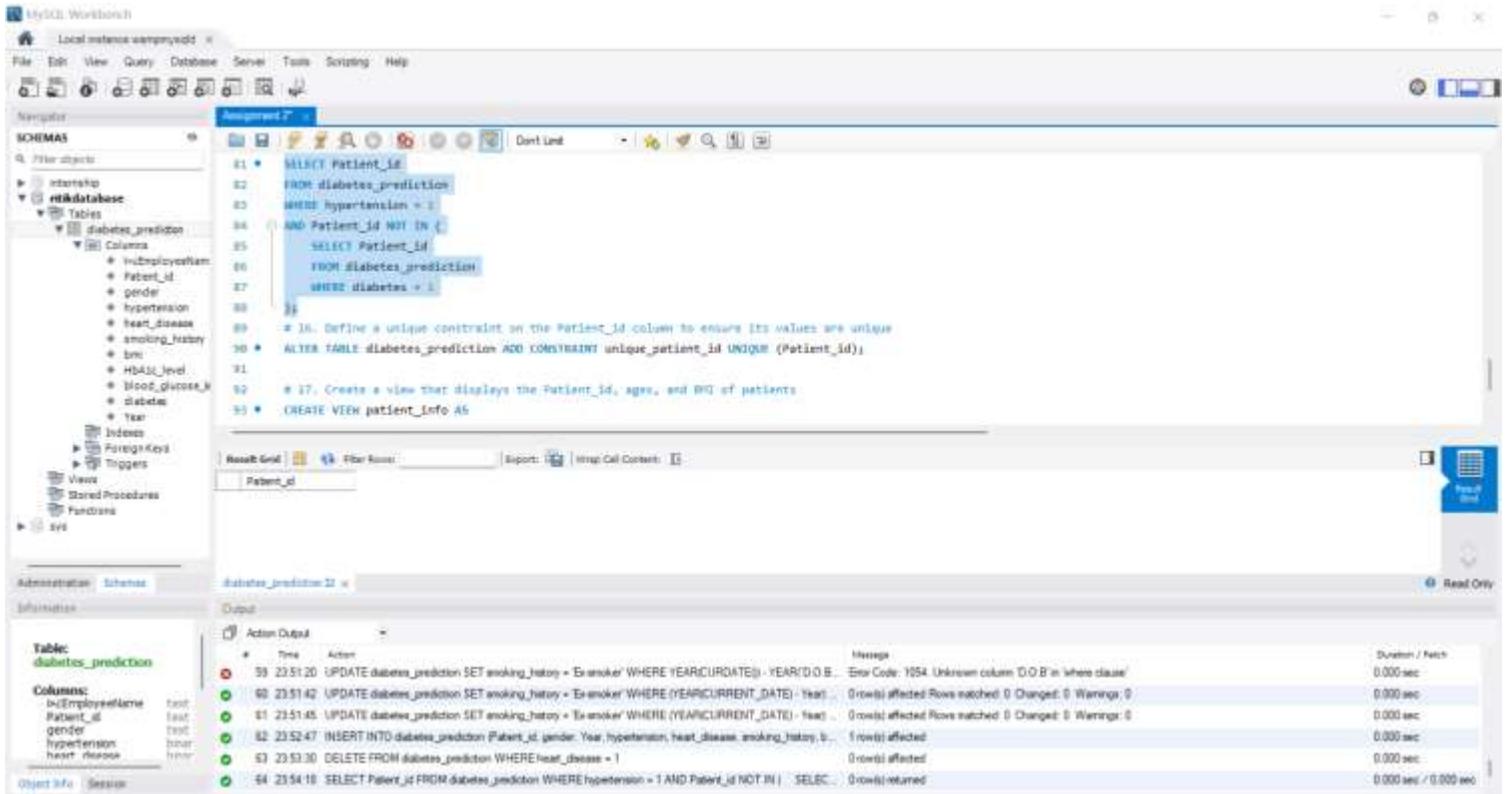
67
68 # 12. Update the smoking history of patients who are older than 40 to "Ex-smoker"
69 * UPDATE diabetes_prediction
70 SET smoking_history = "Ex-smoker"
71 WHERE (YEAR(CURRENT_DATE) - Year) > 40;
72
73 # 13. Insert a new patient into the database with sample data
74 * INSERT INTO diabetes_prediction (Patient_id, gender, Year, hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes)
75 VALUES ('P1000', 'Male', 1980, 0, 0, 'never', 22.5, 5.7, 90, 0);
76
77 # 14. Delete all patients with heart disease from the database
78 * DELETE FROM diabetes_prediction WHERE heart_disease = 1;
79
80 #15. Find patients who have hypertension but not diabetes using the EXCEPT operator. MySQL does not support the EXCEPT operator directly. Instead, we can use a subquery.
81 * SELECT Patient_id
82 FROM diabetes_prediction
83 WHERE hypertension = 1
84 AND Patient_id NOT IN (
85 SELECT Patient_id
86 FROM diabetes_prediction
87 WHERE diabetes = 1

```

The Output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
59	23:50:24	SELECT Patient_id, gender, blood_glucose_level, RANK() OVER (PARTITION BY gender ORDER BY blood_...	0 rows returned	0.000 sec / 0.000 sec
59	23:51:20	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE YEAR(CURRENT_DATE) - YEAR(D.O.B.	Error Code: 1054 Unknown column 'D.O.B' in 'where clause'	0.000 sec
60	23:51:42	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) ...	0 rows affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
61	23:51:45	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) ...	0 rows affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
62	23:52:47	INSERT INTO diabetes_prediction (Patient_id, gender, Year, hypertension, heart_disease, smoking_history, b...	1 row(s) affected	0.000 sec
63	23:53:30	DELETE FROM diabetes_prediction WHERE heart_disease = 1	0 rows affected	0.000 sec

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema, including the 'diabetes_prediction' table. The main editor window contains the following SQL script:

```

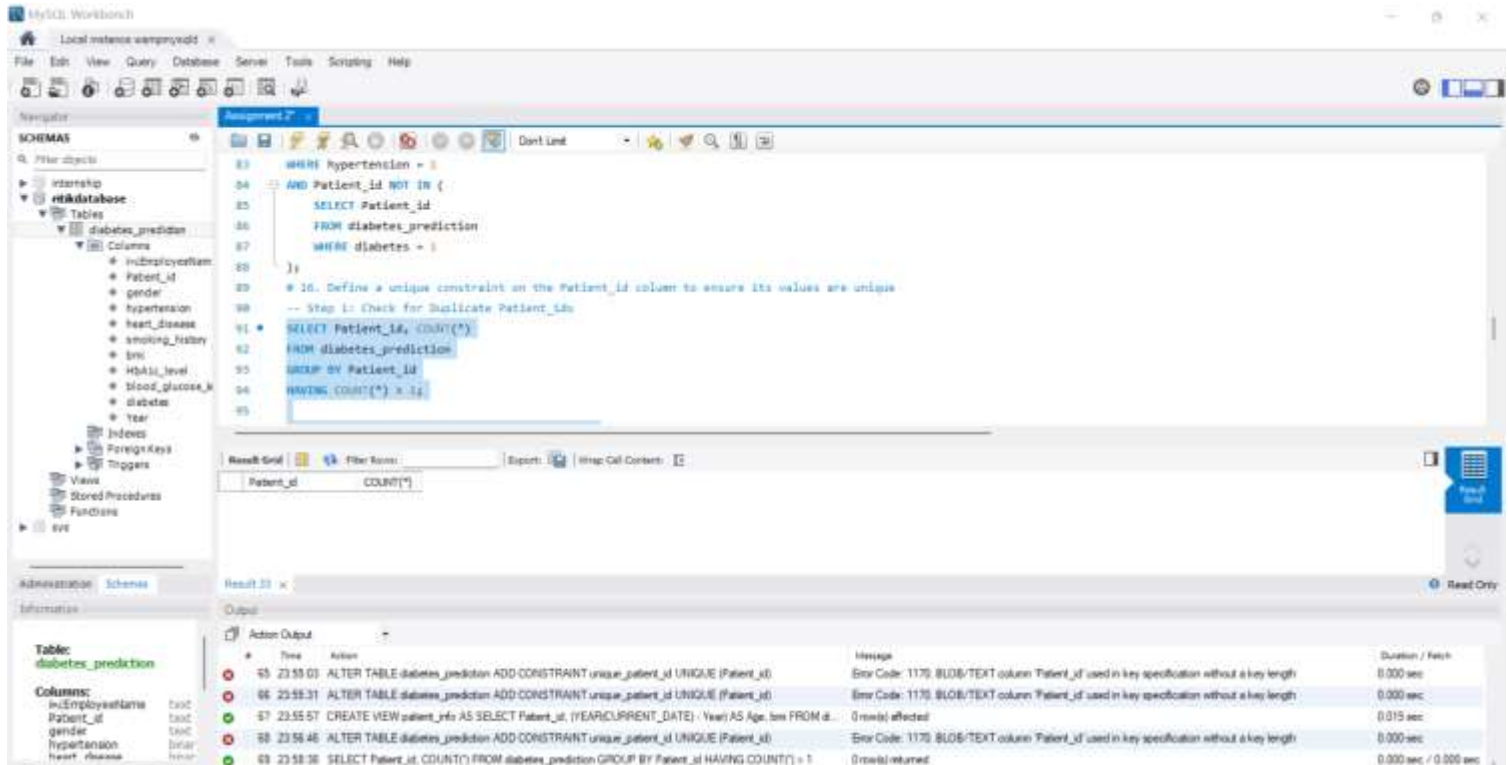
11 SELECT Patient_id
12 FROM diabetes_prediction
13 WHERE hypertension = 1
14 AND Patient_id NOT IN (
15     SELECT Patient_id
16     FROM diabetes_prediction
17     WHERE diabetes = 1
18 );
19 -- 16. Define a unique constraint on the Patient_id column to ensure its values are unique
20 ALTER TABLE diabetes_prediction ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id);
21
22 -- 17. Create a view that displays the Patient_id, ages, and BMI of patients
23 CREATE VIEW patient_info AS

```

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

#	Time	Action	Message	Duration / Patch
59	23:51:20	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE YEAR(CURRENT_DATE) - YEAR(D.O.B) ...	Error Code: 1054. Unknown column 'D.O.B' in 'where clause'	0.000 sec
60	23:51:42	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) ...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
61	23:51:45	UPDATE diabetes_prediction SET smoking_history = 'Ex-smoker' WHERE (YEAR(CURRENT_DATE) - Year) ...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
62	23:52:47	INSERT INTO diabetes_prediction (Patient_id, gender, Year, hypertension, heart_disease, smoking_history, b...	1 row(s) affected	0.000 sec
63	23:53:30	DELETE FROM diabetes_prediction WHERE heart_disease = 1	0 row(s) affected	0.000 sec
64	23:54:18	SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND Patient_id NOT IN (SELECT ...	0 row(s) returned	0.000 sec / 0.000 sec

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 database schema, including the 'diabetes_prediction' table. The main editor window contains the following SQL script:

```

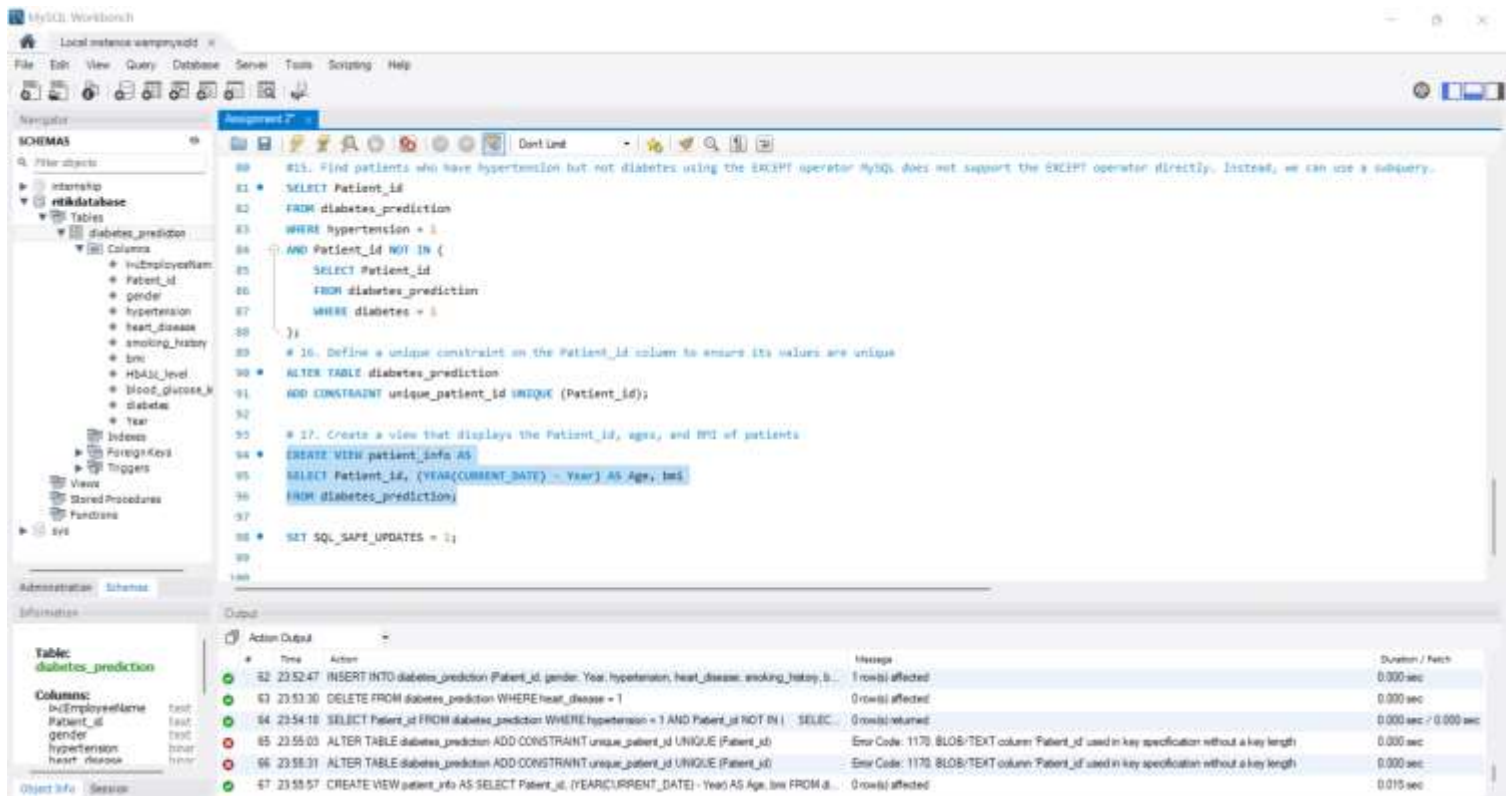
83 WHERE hypertension = 1
84 AND Patient_id NOT IN (
85     SELECT Patient_id
86     FROM diabetes_prediction
87     WHERE diabetes = 1
88 );
89 -- 16. Define a unique constraint on the Patient_id column to ensure its values are unique
90 -- Step 1: Check for Duplicate Patient_ids
91 SELECT Patient_id, COUNT(*)
92 FROM diabetes_prediction
93 GROUP BY Patient_id
94 HAVING COUNT(*) > 1;
95

```

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

#	Time	Action	Message	Duration / Patch
95	23:55:03	ALTER TABLE diabetes_prediction ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id)	Error Code: 1170. BLOB/TEXT column 'Patient_id' used in key specification without a key length	0.000 sec
96	23:55:31	ALTER TABLE diabetes_prediction ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id)	Error Code: 1170. BLOB/TEXT column 'Patient_id' used in key specification without a key length	0.000 sec
97	23:55:57	CREATE VIEW patient_info AS SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age, bmi FROM di...	0 row(s) affected	0.019 sec
98	23:56:46	ALTER TABLE diabetes_prediction ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id)	Error Code: 1170. BLOB/TEXT column 'Patient_id' used in key specification without a key length	0.000 sec
99	23:58:36	SELECT Patient_id, COUNT(*) FROM diabetes_prediction GROUP BY Patient_id HAVING COUNT(*) > 1	0 row(s) returned	0.000 sec / 0.000 sec

17) 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 'interview' and 'interviewdatabase' expanded. The 'interviewdatabase' table is selected, showing its columns: 'interview_id', 'patient_id', 'gender', 'hypertension', 'heart_disease', 'smoking_history', 'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes', and 'Year'. The main editor shows a series of SQL queries. The first query is a SELECT statement to find patients with hypertension but not diabetes. The second query is an ALTER TABLE statement to add a unique constraint on the 'patient_id' column. The third query is a CREATE VIEW statement to create a view named 'patient_info' that displays patient details. The bottom panel shows the 'Action Output' window with the results of the executed queries, including the number of rows affected and any error messages.

```
80 #15. Find patients who have hypertension but not diabetes using the EXCEPT operator MySQL does not support the EXCEPT operator directly. Instead, we can use a subquery.
81 SELECT Patient_id
82 FROM diabetes_prediction
83 WHERE hypertension = 1
84 AND Patient_id NOT IN (
85     SELECT Patient_id
86     FROM diabetes_prediction
87     WHERE diabetes = 1
88 );
89
90 # 16. Define a unique constraint on the Patient_id column to ensure its values are unique
91 ALTER TABLE diabetes_prediction
92 ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id);
93
94 # 17. Create a view that displays the Patient_id, age, and BMI of patients
95 CREATE VIEW patient_info AS
96 SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age, bmi
97 FROM diabetes_prediction;
98
99 SET SQL_SAFE_UPDATES = 0;
```

#	Time	Action	Message	Duration / Fetch
82	23:52:47	INSERT INTO diabetes_prediction (Patient_id, gender, Year, hypertension, heart_disease, smoking_history, b...	1 row(s) affected	0.000 sec
83	23:53:30	DELETE FROM diabetes_prediction WHERE heart_disease = 1	0 row(s) affected	0.000 sec
84	23:54:18	SELECT Patient_id FROM diabetes_prediction WHERE hypertension = 1 AND Patient_id NOT IN (SELECT	0 row(s) returned	0.000 sec / 0.000 sec
85	23:55:03	ALTER TABLE diabetes_prediction ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id)	Error Code: 1170. BLOB/TEXT column 'Patient_id' used in key specification without a key length	0.000 sec
86	23:56:31	ALTER TABLE diabetes_prediction ADD CONSTRAINT unique_patient_id UNIQUE (Patient_id)	Error Code: 1170. BLOB/TEXT column 'Patient_id' used in key specification without a key length	0.000 sec
87	23:58:57	CREATE VIEW patient_info AS SELECT Patient_id, (YEAR(CURRENT_DATE) - Year) AS Age, bmi FROM d...	0 row(s) affected	0.015 sec

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

To reduce data redundancy and improve data integrity, consider the following improvements:

- Normalize the database by splitting it into multiple related tables. For instance:
 - Create a patients table to store personal information (e.g., Patient_id, gender, Year).
 - Create a medical_records table to store medical information (e.g., hypertension, heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes).
- Use foreign keys to link these tables.
- Ensure that columns such as Patient_id are indexed for faster lookups.
- Use appropriate data types for each column (e.g., DATE for D.O.B).

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

To optimize the performance of SQL queries on this dataset:

- Index columns that are frequently used in WHERE clauses (e.g., Patient_id, Year).
- Use appropriate data types to minimize storage space and improve query performance.
- Avoid using SELECT * in production queries; instead, specify the needed columns.
- Use JOINS efficiently and ensure that tables are properly normalized.
- Regularly update statistics and optimize the database to ensure efficient query plans.