Name: **Sachin Mishra**
Email: smzp274@gmail.com
LinkedIn: https://www.linkedin.com/in/sachin-mishra-4169b7213/

DIABETES PREDICTION ASSESSMENT – Psyliq

1. Retrieve the Patient_id and ages of all patients.

   SELECT Patiend_id, age FROM diabetes_prediction;

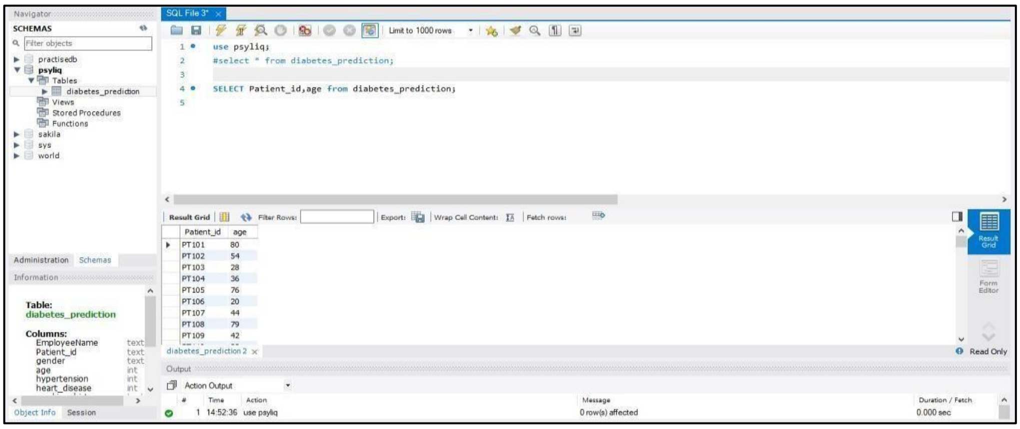


Table: diabetes_prediction

Columns:

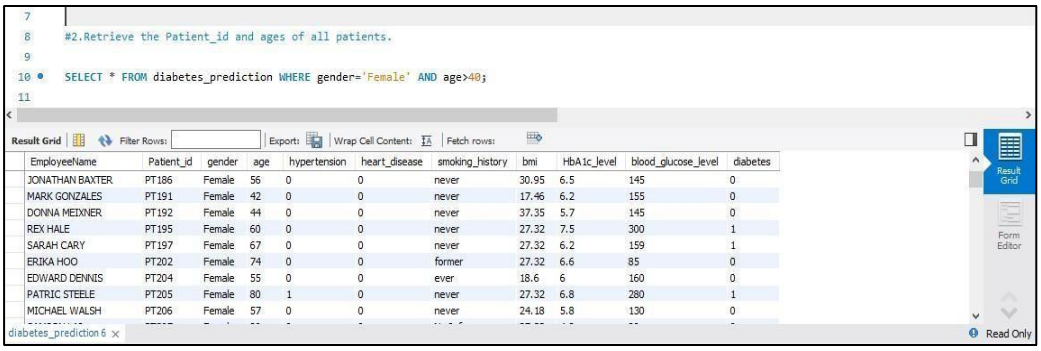
EmployeeName	text
Patient_id	text
gender	text
age	int
hypertension	int
heart_disease	int

```
use psyliq;
#select * from diabetes_prediction;

SELECT Patient_id,age from diabetes_prediction;
```

Patient_id	age
PT101	80
PT102	54
PT103	28
PT104	36
PT105	76
PT106	20
PT107	44
PT108	79
PT109	42

#	Time	Action	Message	Duration / Fetch
1	14:52:36	use psyliq	0 row(s) affected	0.000 sec

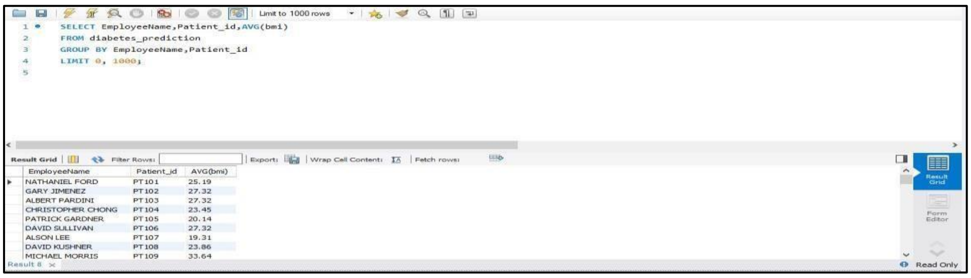
2. Select all female patients who are older than 40.
   SELECT * FROM diabetes_prediction WHERE gender=’Female’ AND age>40;



```

#2.Retrieve the Patient_id and ages of all patients.

SELECT * FROM diabetes_prediction WHERE gender='Female' AND age>40;

```

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
JONATHAN BAXTER	PT186	Female	56	0	0	never	30.95	6.5	145	0
MARK GONZALES	PT191	Female	42	0	0	never	17.46	6.2	155	0
DONNA MEIXNER	PT192	Female	44	0	0	never	37.35	5.7	145	0
REX HALE	PT195	Female	60	0	0	never	27.32	7.5	300	1
SARAH CARY	PT197	Female	67	0	0	never	27.32	6.2	159	1
ERIKA HOO	PT202	Female	74	0	0	former	27.32	6.6	85	0
EDWARD DENNIS	PT204	Female	55	0	0	ever	18.6	6	160	0
PATRIC STEELE	PT205	Female	80	1	0	never	27.32	6.8	280	1
MICHAEL WALSH	PT206	Female	57	0	0	never	24.18	5.8	130	0

3. Calculate the average BMI of patients.
   SELECT
   EmployeeName,Patient_id,AVG(bmi)
   FROM diabetes_prediction
   GROUP BY
   EMPLOYEENAME,Patiend_id
   LIMIT 0,1000;

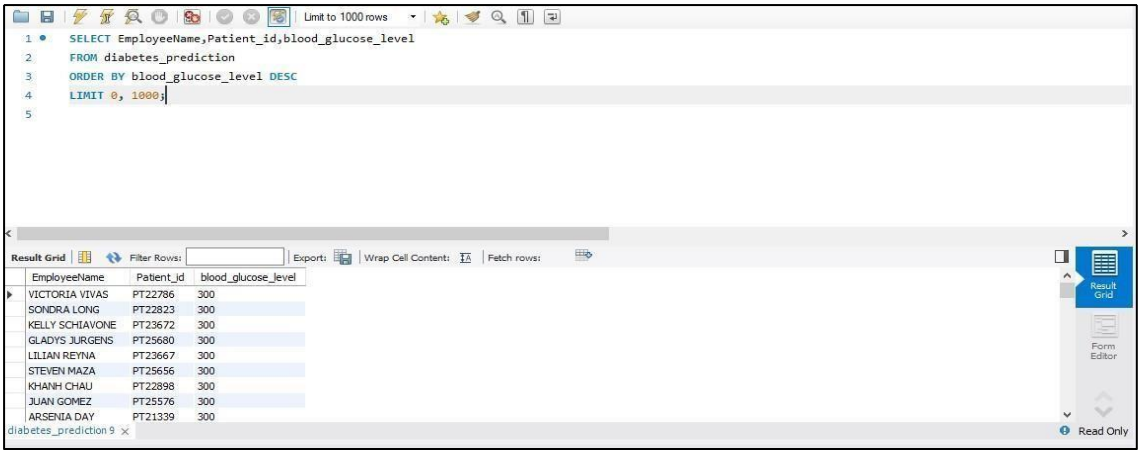


```
SELECT EmployeeName,Patient_id,AVG(bmi)
FROM diabetes_prediction
GROUP BY EmployeeName,Patient_id
LIMIT 0, 1000;
```

EmployeeName	Patient_id	AVG(bmi)
NATHANIEL FORD	PT101	25.19
GARY JIMENEZ	PT102	27.32
ALBERT PARDINI	PT103	27.32
CHRISTOPHER CHONG	PT104	23.45
PATRICK GARDNER	PT105	20.14
DAVID SULLIVAN	PT106	27.32
ALSON LEE	PT107	19.31
DAVID KUSHNER	PT108	23.86
MICHAEL MORRIS	PT109	33.64

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

```
SELECT EmployeeName, Patient_id,
blood_glucose_levelFROM diabetes_prediction
ORDER BY blood_glucose_level
DESC,LIMIT0,1000;
```

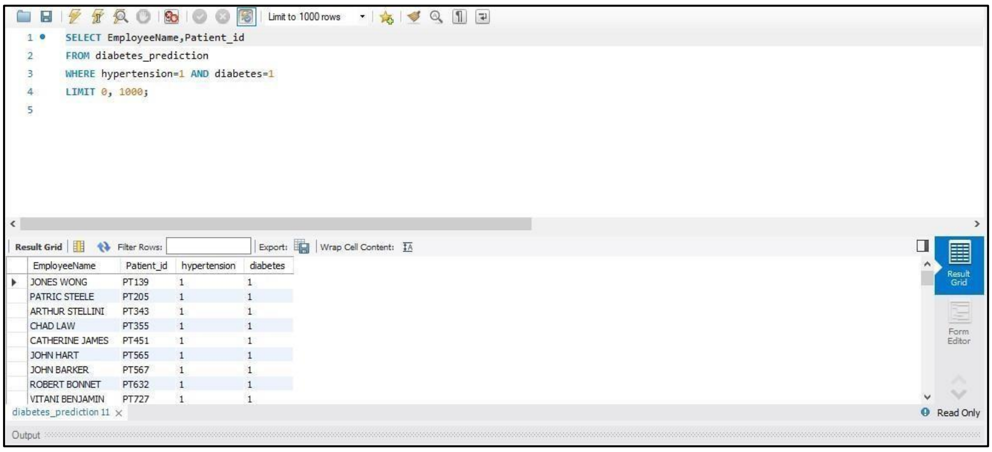


```
SELECT EmployeeName,Patient_id,blood_glucose_level
FROM diabetes_prediction
ORDER BY blood_glucose_level DESC
LIMIT 0, 1000;
```

EmployeeName	Patient_id	blood_glucose_level
VICTORIA VIVAS	PT22786	300
SONDRA LONG	PT22823	300
KELLY SCHIAVONE	PT23672	300
GLADYS JURGENS	PT25680	300
LILIAN REYNA	PT23667	300
STEVEN MAZA	PT25656	300
KHANH CHAU	PT22898	300
JUAN GOMEZ	PT25576	300
ARSENIA DAY	PT21339	300

5. Find patients who have hypertension and diabetes.

```
SELECT
EmployeeName,Patient_id
FROM diabetes_prediction
WHERE hypertension=1 AND
diabetes=1LIMIT0,1000;
```

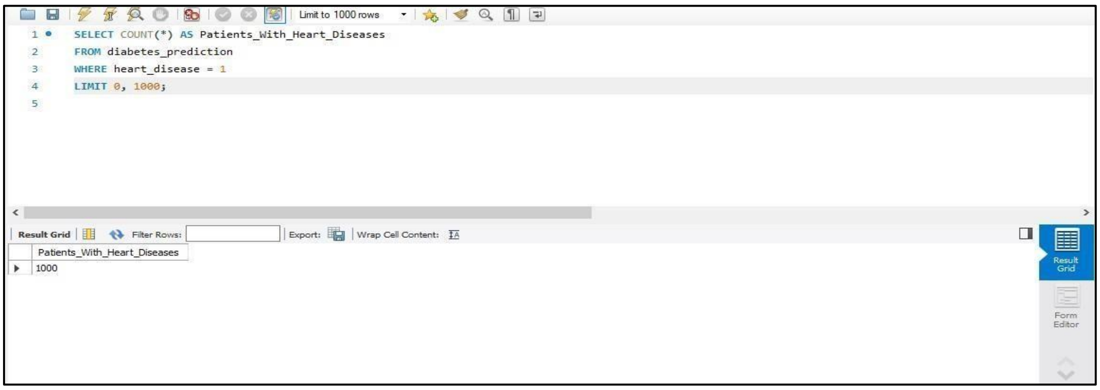


```
SELECT EmployeeName,Patient_id
FROM diabetes_prediction
WHERE hypertension=1 AND diabetes=1
LIMIT 0, 1000;
```

EmployeeName	Patient_id	hypertension	diabetes
JONES WONG	PT139	1	1
PATRIC STEELE	PT205	1	1
ARTHUR STELLINI	PT343	1	1
CHAD LAW	PT355	1	1
CATHERINE JAMES	PT451	1	1
JOHN HART	PT565	1	1
JOHN BARKER	PT567	1	1
ROBERT BONNET	PT632	1	1
VITANI BENJAMIN	PT727	1	1

6. Determine the number of patients with heart disease.

```
SELECT COUNT(*) AS
 Patients_With_Heart_diseasesFROM
 diabetes_prediction
WHERE heart_disease=1
```

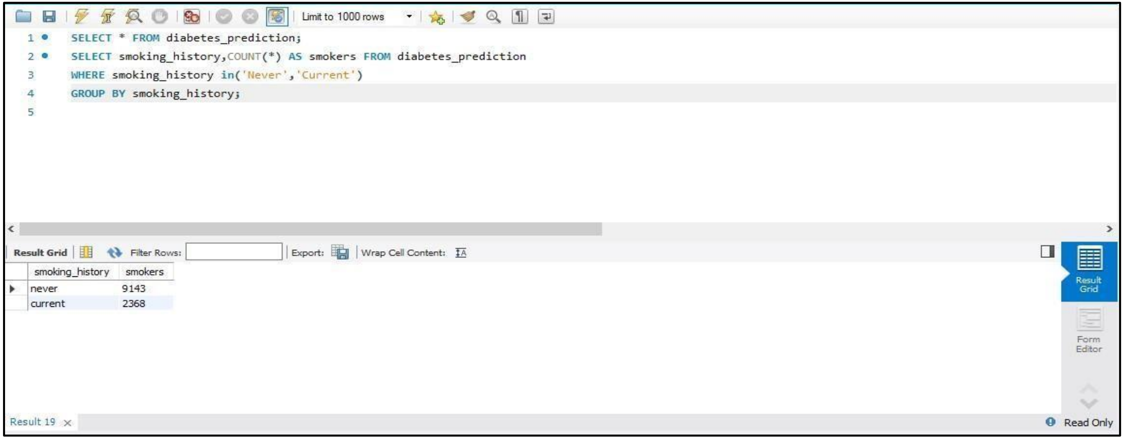


```
SELECT COUNT(*) AS Patients_With_Heart_Diseases
FROM diabetes_prediction
WHERE heart_disease = 1
LIMIT 0, 1000;
```

Patients_With_Heart_Diseases
1000

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

   SELECT * FROM diabetes_prediction;
   SELECT smoking_history, COUNT(*) AS smokers
   FROM diabetes_prediction
   WHERE smoking_history in(“Never”,”Current”)
   GROUP BY smoking_history



```
SELECT * FROM diabetes_prediction;
SELECT smoking_history,COUNT(*) AS smokers FROM diabetes_prediction
WHERE smoking_history in('Never','Current')
GROUP BY smoking_history;
```

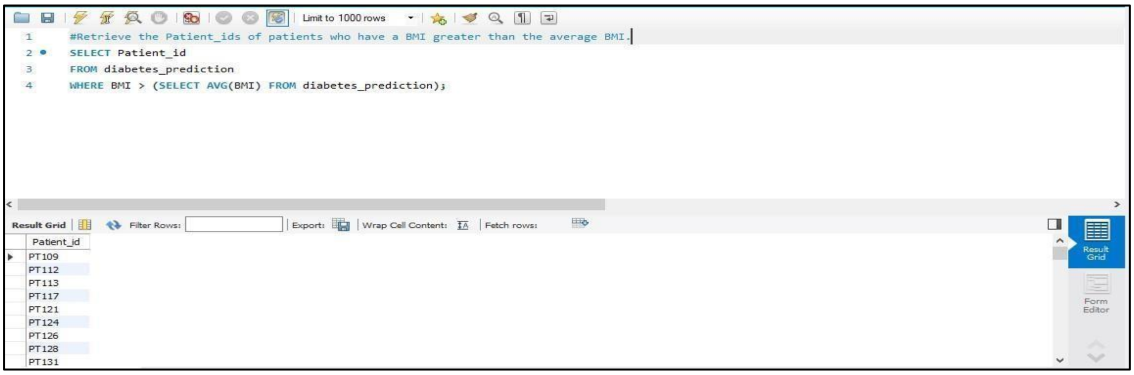
smoking_history	smokers
never	9143
current	2368

8. Retrieve the Patient_ids 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);



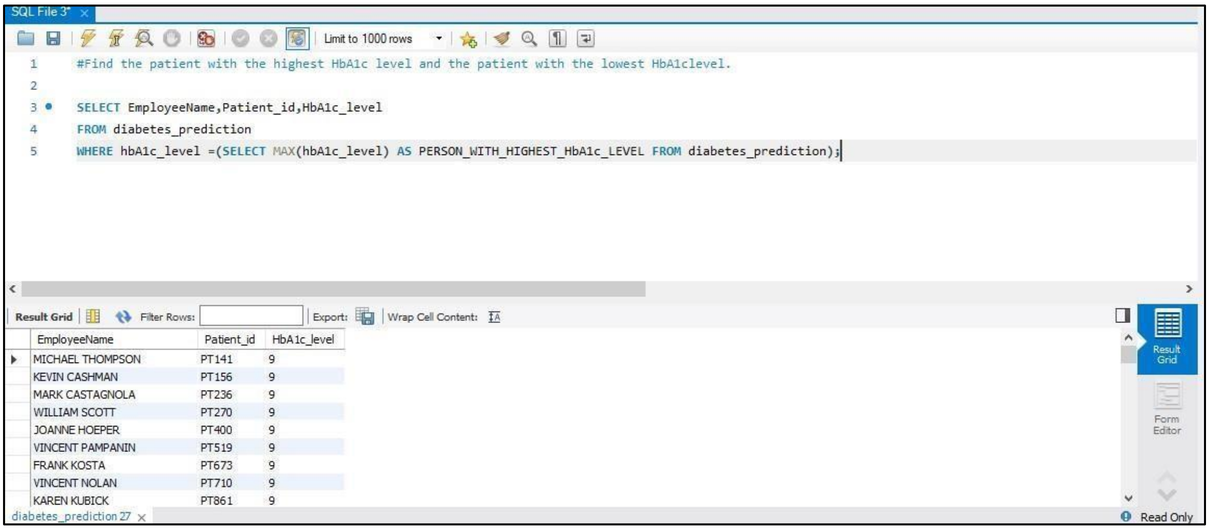
```
#Retrieve the Patient_ids of patients who have a BMI greater than the average BMI.
SELECT Patient_id
FROM diabetes_prediction
WHERE BMI > (SELECT AVG(BMI) FROM diabetes_prediction);
```

Patient_id
PT109
PT112
PT113
PT117
PT121
PT124
PT126
PT128
PT131

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

Highest hbA1c_level

```
SELECT
EmployeeName,Patient_id,HbA1c_levelFROM
diabetes_prediction
WHERE
hbA1c_level=(SELECTMAX(hbA1c_level)AS
PERSON_WITH_HIGHEST_hbA1c_LEVEL
FROM diabetes_prediction;
```

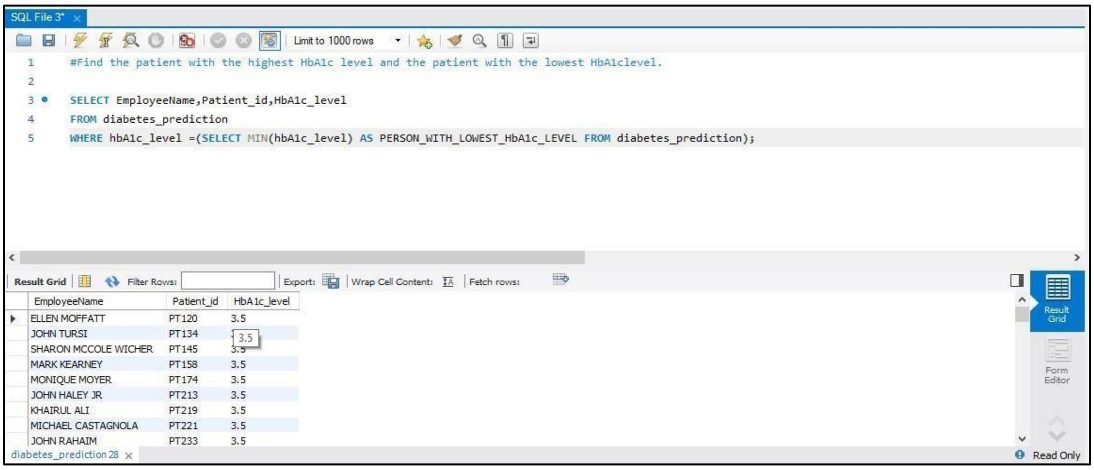


```
#Find the patient with the highest HbA1c level and the patient with the lowest HbA1clevel.

SELECT EmployeeName,Patient_id,HbA1c_level
FROM diabetes_prediction
WHERE hbA1c_level =(SELECT MAX(hbA1c_level) AS PERSON_WITH_HIGHEST_HbA1c_LEVEL FROM diabetes_prediction);
```

EmployeeName	Patient_id	HbA1c_level
MICHAEL THOMPSON	PT141	9
KEVIN CASHMAN	PT156	9
MARK CASTAGNOLA	PT236	9
WILLIAM SCOTT	PT270	9
JOANNE HOEPER	PT400	9
VINCENT PAMPANIN	PT519	9
FRANK KOSTA	PT673	9
VINCENT NOLAN	PT710	9
KAREN KUBICK	PT861	9

Lowest hbA1c_level

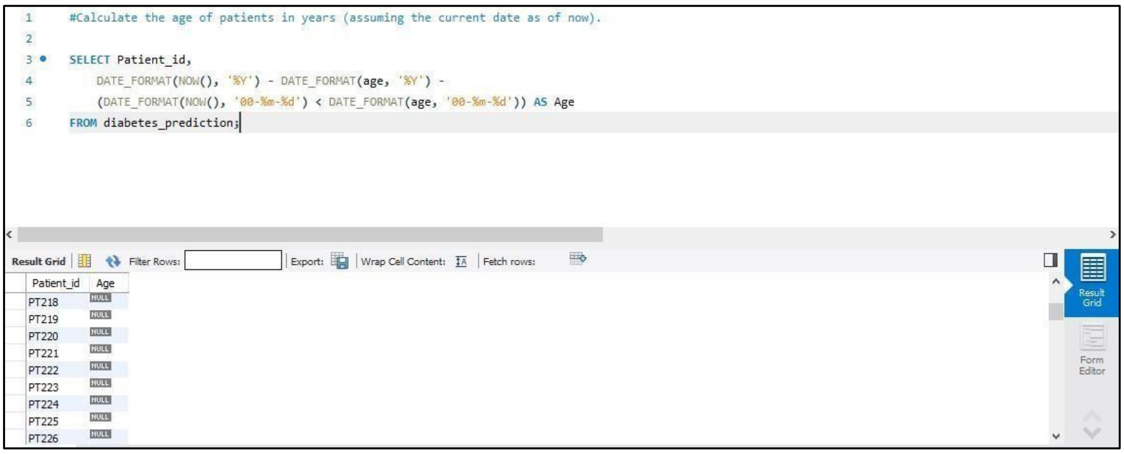
```
SELECT
EmployeeName,Patient_id,HbA1c_level FROM
diabetes_prediction
WHEREhbA1c_level=(SELECTMIN(hbA1c_lev
el)AS
PERSON_WITH_LOWEST_hbA1c_LEVEL
FROM diabetes_prediction;
```



```sql
#Find the patient with the highest HbA1c level and the patient with the lowest HbA1clevel.

SELECT EmployeeName,Patient_id,HbA1c_level
FROM diabetes_prediction
WHERE hbA1c_level =(SELECT MIN(hbA1c_level) AS PERSON_WITH_LOWEST_HbA1c_LEVEL FROM diabetes_prediction);
```

EmployeeName	Patient_id	HbA1c_level
ELLEN MOFFATT	PT120	3.5
JOHN TURSI	PT134	3.5
SHARON MCCOLE WICHER	PT145	3.5
MARK KEARNEY	PT158	3.5
MONIQUE MOYER	PT174	3.5
JOHN HALEY JR	PT213	3.5
KHAIRUL ALI	PT219	3.5
MICHAEL CASTAGNOLA	PT221	3.5
JOHN RAHAIM	PT233	3.5

10. Calculate the age of patients in years (assuming the current date as ofnow).
SELECT Patient_id,
DATE_FORMAT(NOW(),'%Y') – DATE_FORMAT(age,'%Y') –
DATE_FORMAT(NOW(),'00-%m-
%d') < DATE_FORMAT(age, '00-%m-%d'))AS Age
FROM diabetes_prediction

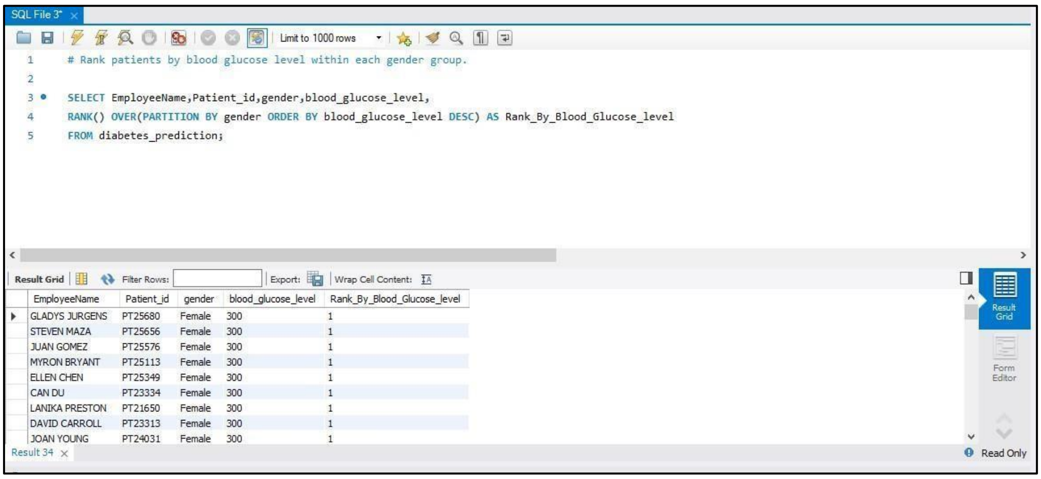


```sql
#Calculate the age of patients in years (assuming the current date as of now).

SELECT Patient_id,
    DATE_FORMAT(NOW(), '%Y') - DATE_FORMAT(age, '%Y') -
    (DATE_FORMAT(NOW(), '00-%m-%d') < DATE_FORMAT(age, '00-%m-%d')) AS Age
FROM diabetes_prediction;
```

Patient_id	Age
PT218	NULL
PT219	NULL
PT220	NULL
PT221	NULL
PT222	NULL
PT223	NULL
PT224	NULL
PT225	NULL
PT226	NULL

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

SELECT EmployeeName,Patient_id,gender,blood_glucose_level,
RANK() OVER(PARTITION BY gender ORDER BY blood_glucose_level
DESC)ASRank_ By_Blood_Glucose_Level;

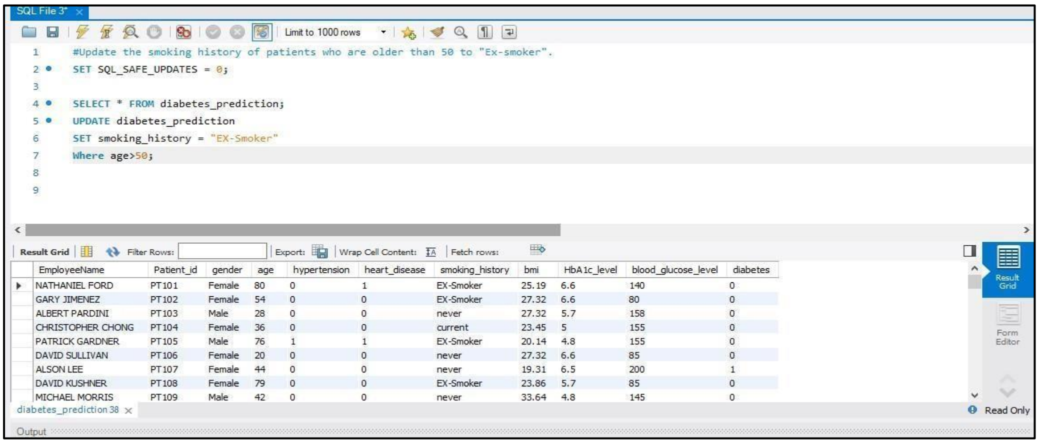


```sql
# Rank patients by blood glucose level within each gender group.

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

EmployeeName	Patient_id	gender	blood_glucose_level	Rank_By_Blood_Glucose_level
GLADYS JURGENS	PT25680	Female	300	1
STEVEN MAZA	PT25656	Female	300	1
JUAN GOMEZ	PT25576	Female	300	1
MYRON BRYANT	PT25113	Female	300	1
ELLEN CHEN	PT25349	Female	300	1
CAN DU	PT23334	Female	300	1
LANIKA PRESTON	PT21650	Female	300	1
DAVID CARROLL	PT23313	Female	300	1
JOAN YOUNG	PT24031	Female	300	1

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

SELECT * FROM diabetes_prediction;UPDATE
diabetes_prediction
SET smoking_history = “EX-
Smoker”WHEREage>50;

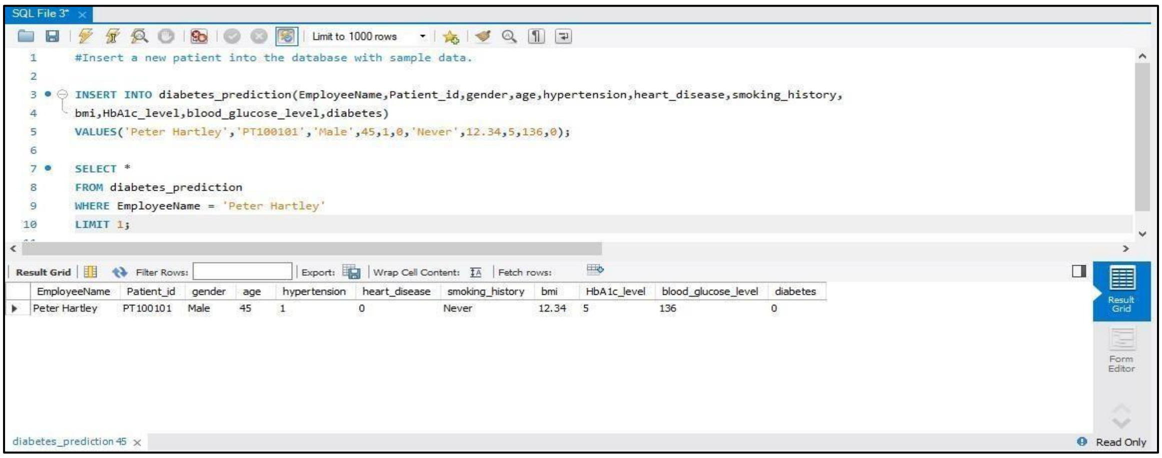


```sql
#Update the smoking history of patients who are older than 50 to "Ex-smoker".
SET SQL_SAFE_UPDATES = 0;

SELECT * FROM diabetes_prediction;
UPDATE diabetes_prediction
SET smoking_history = "EX-Smoker"
Where age>50;
```

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

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

INSERT INTO diabetes_prediction (EmployeeName, Patient_id, gender,age, hypertension,heart_disease, smoking_history, bmi, HbA1c_level, blood_glucose_level, diabetes)
VALUES (‘Peter Hartley’,’PT100101’,’Male’,45,1,0,’Never’,12.34,5,136,0);

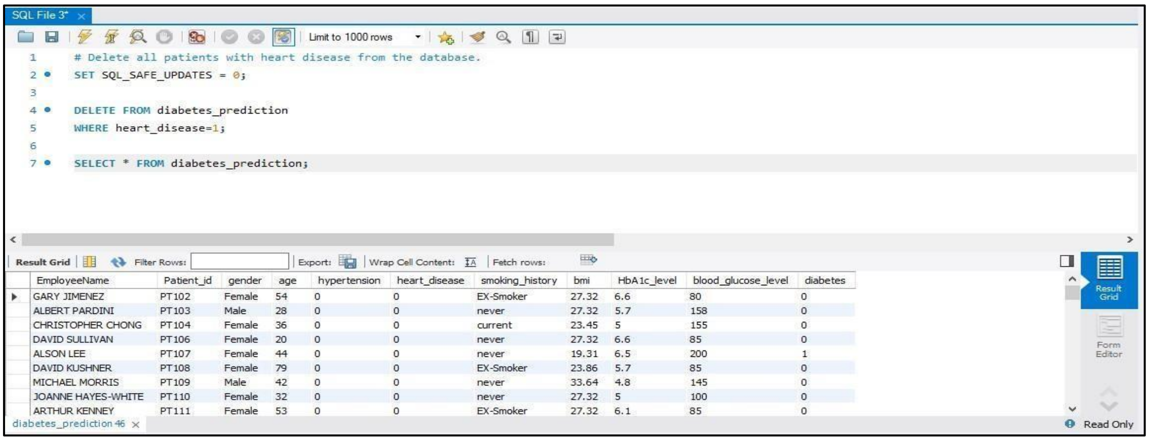


```sql
#Insert a new patient into the database with sample data.

INSERT INTO diabetes_prediction(EmployeeName,Patient_id,gender,age,hypertension,heart_disease,smoking_history,
bmi,HbA1c_level,blood_glucose_level,diabetes)
VALUES('Peter Hartley','PT100101','Male',45,1,0,'Never',12.34,5,136,0);

SELECT *
FROM diabetes_prediction
WHERE EmployeeName = 'Peter Hartley'
LIMIT 1;
```

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
Peter Hartley	PT100101	Male	45	1	0	Never	12.34	5	136	0

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

```
DELETE FROM
diabetes_predictionWHERE
heart_disease=1;
```

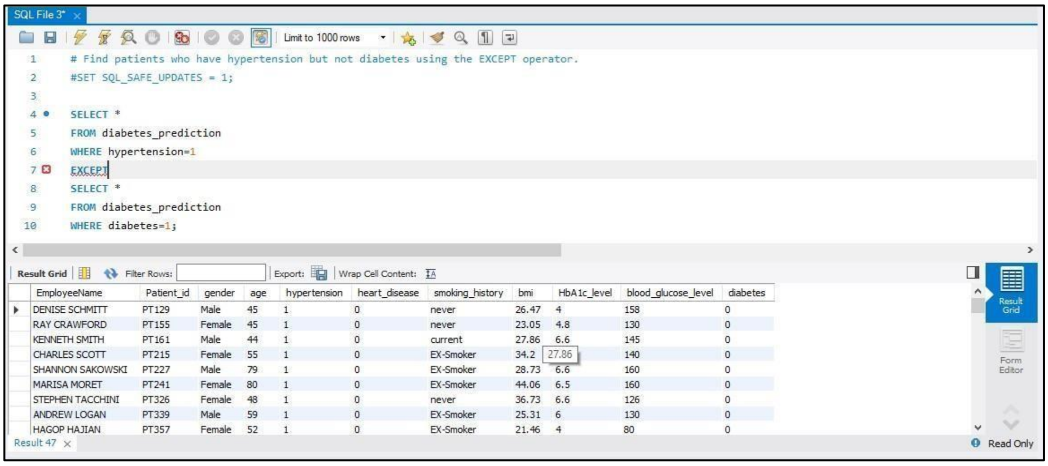


```sql
# Delete all patients with heart disease from the database.
SET SQL_SAFE_UPDATES = 0;

DELETE FROM diabetes_prediction
WHERE heart_disease=1;

SELECT * FROM diabetes_prediction;
```

EmployeeName	Patient_id	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
GARY JIMENEZ	PT102	Female	54	0	0	EX-Smoker	27.32	6.6	80	0
ALBERT PARDINI	PT103	Male	28	0	0	never	27.32	5.7	158	0
CHRISTOPHER CHONG	PT104	Female	36	0	0	current	23.45	5	155	0
DAVID SULLIVAN	PT106	Female	20	0	0	never	27.32	6.6	85	0
ALSON LEE	PT107	Female	44	0	0	never	19.31	6.5	200	1
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
JOANNE HAYES-WHITE	PT110	Female	32	0	0	never	27.32	5	100	0
ARTHUR KENNEY	PT111	Female	53	0	0	EX-Smoker	27.32	6.1	85	0

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

```
SELECT * FROM
diabetes_predictionWHERE
hypertension=1
EXCEPT
SELECT * FROM diabetes_prediction WHERE diabetes=1;
```



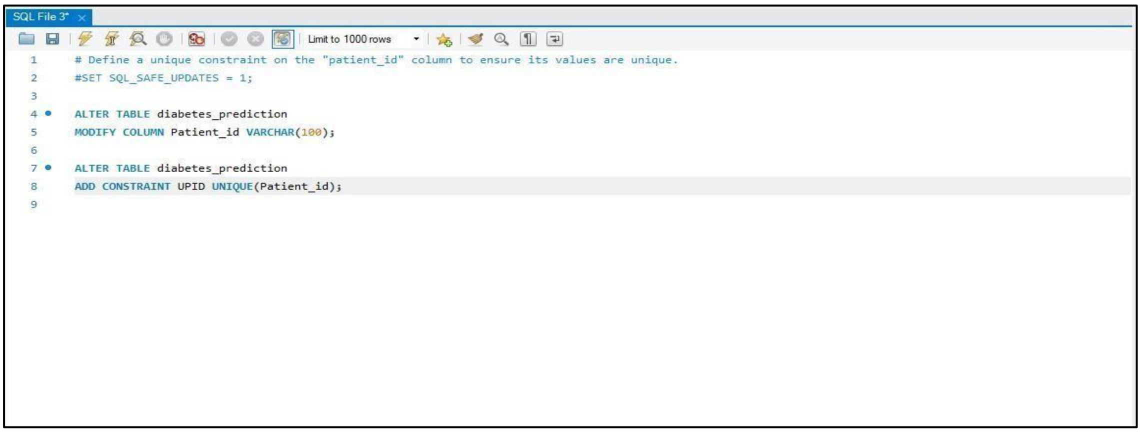
```sql
# Find patients who have hypertension but not diabetes using the EXCEPT operator.
#SET SQL_SAFE_UPDATES = 1;

SELECT *
FROM diabetes_prediction
WHERE hypertension=1
EXCEPT
SELECT *
FROM diabetes_prediction
WHERE diabetes=1;
```

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	27.86	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

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

ALTER TABLE diabetes_prediction
MODIFY COLUMN Patient_id VARCHAR(100);

ALTER TABLE diabetes_prediction
ADD CONSTRAINT UPID UNIQUE(Patient_id);

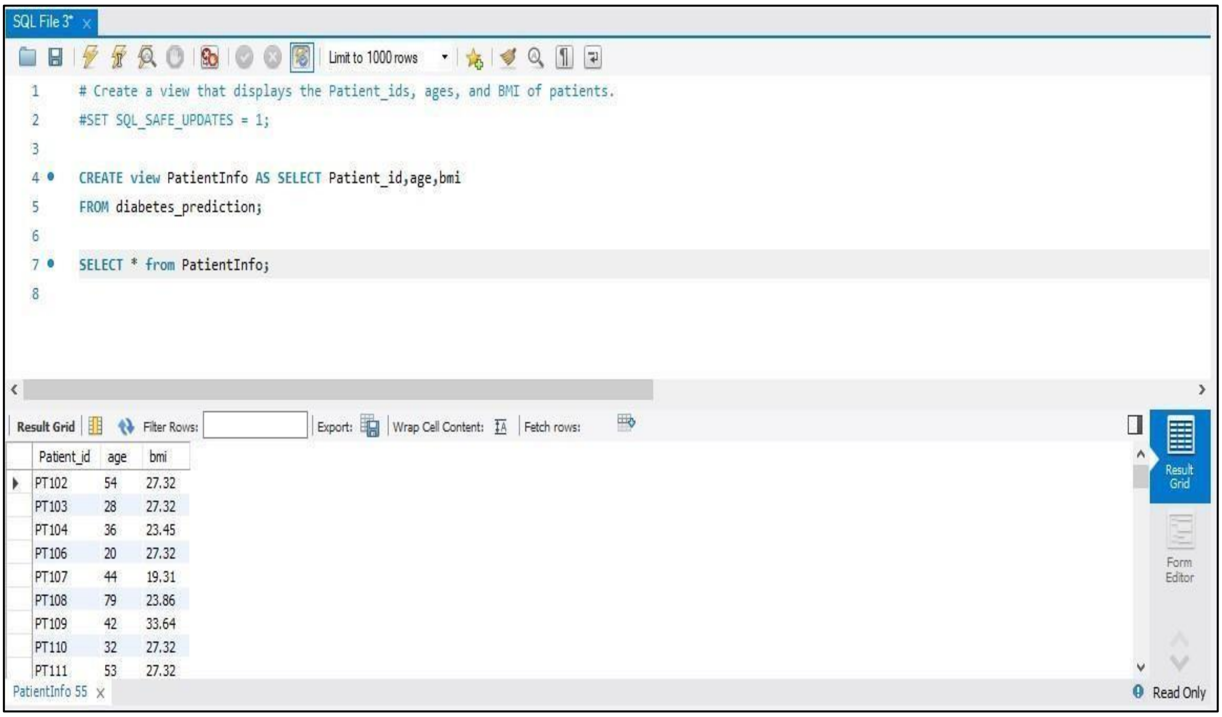


```
# Define a unique constraint on the "patient_id" column to ensure its values are unique.
#SET SQL_SAFE_UPDATES = 1;

ALTER TABLE diabetes_prediction
MODIFY COLUMN Patient_id VARCHAR(100);

ALTER TABLE diabetes_prediction
ADD CONSTRAINT UPID UNIQUE(Patient_id);

```

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

CREATE view PatientInfo AS SELECT
Patient_id,age,bmi FROM diabetes_prediction;

SELECT * from PatientInfo;



```
# Create a view that displays the Patient_ids, ages, and BMI of patients.
#SET SQL_SAFE_UPDATES = 1;

CREATE view PatientInfo AS SELECT Patient_id,age,bmi
FROM diabetes_prediction;

SELECT * from PatientInfo;

```

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

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

Reducing data redundancy and enhancing data integrity in a database schema involves implementing the following improvements:

1. Normalization:
   Apply normalization techniques to organize data efficiently. Break down large tables into smaller ones and establish relationships to minimize redundancy. This helps in maintaining data integrity by ensuring that each piece of information is stored in only one place.

2. Use of Primary and Foreign Keys:
   Define and utilize primary keys to uniquely identify records within a table. Implement foreign keys to establish relationships between tables, ensuring accurate linkages and preventing orphaned records.

3. Elimination of Transitive Dependencies:
   Identify and eliminate transitive dependencies, where data is indirectly dependent on other data. This helps in removing unnecessary redundancy and inconsistencies in the database, leading to improved data integrity.

4. Views:
   Create views to present a consolidated and normalized perspective of the data. Views act as virtual tables, reducing redundancy by providing a unified source of truth for related information and simplifying data access.

5. Referential Integrity:
   Enforce referential integrity by ensuring that foreign keys reference valid primary keys. This prevents inconsistencies, orphaned records, and strengthens the overall integrity of the data.

6. Triggers:
   Implement triggers to automate actions based on specified events, such as inserting or updating records. Triggers help enforce business rules and maintain data integrity by automatically updating related information, reducing redundancy.

7. Indexes:
   Utilize indexes to optimize query performance. Indexes allow for quick data retrieval based on specified criteria, reducing the need to scan through large datasets and minimizing redundancy in data access.

8. Data Validation:
   Implement robust data validation rules to ensure the accuracy and consistency of data entered into the database. By enforcing validation criteria, redundancy is minimized, reducing the likelihood of duplicate records or errors in data input.

9. Documentation:
   Maintain comprehensive documentation of the database schema, including relationships, constraints, and considerations for developers and users. Clear documentation aids in understanding the database structure, reducing the risk of errors and enhancing data integrity.

10. Regular Review and Update:
    Conduct periodic reviews of the database schema to identify opportunities for optimization and improvement. Adapt the schema based on evolving business requirements, ensuring that it continues to support data integrity goals over time.

Incorporating these improvements significantly reduces data redundancy, enhances data integrity, and establishes a more efficient and reliable database schema.

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

Optimizing the performance of SQL queries on a dataset involves several strategies to ensure efficient data retrieval and processing. Here are some key approaches:

1. Indexing:
   - Identify columns frequently used in WHERE clauses, JOIN conditions, or ORDER BY clauses.
   - Create indexes on these columns to speed up data retrieval.
   - Be cautious not to over-index, as it may impact write performance.

2. Query Rewriting:
   - Review and rewrite complex queries to simplify their structure.
   - Use appropriate JOIN types and conditions to minimize the amount of data processed.
   - Consider breaking down complex queries into simpler, well-optimized subqueries.

3. Avoid SELECT *:
   - Instead of selecting all columns using `SELECT *`, explicitly list only the necessary columns.
   - This reduces the amount of data transferred and improves query performance.

4. Use Proper Joins:
   - Choose the most efficient JOIN type (INNER JOIN, LEFT JOIN, etc.) based on the relationships between tables.
   - Ensure that join conditions are properly indexed.

5. Aggregate Functions:
   - Minimize the use of expensive aggregate functions like COUNT DISTINCT or GROUP BY when not essential.
   - Optimize queries that involve aggregation to reduce processing time.

6. Partitioning:
   - If applicable, consider partitioning large tables based on certain criteria (e.g., date ranges).
   - Partition pruning can significantly reduce the amount of data scanned during queries.

7. Update Statistics:
   - Keep database statistics up-to-date to help the query planner make informed decisions.
   - Regularly analyze and update index statistics for accurate query optimization.

8. Limit Data Retrieval:
   - Use the LIMIT or TOP clause to restrict the number of rows returned, especially when displaying data in a user interface.
   - This reduces the load on both the database server and the network.

9. Caching:
   - Implement caching mechanisms for frequently executed and static queries.
   - Cached results can be served quickly without hitting the database, enhancing performance.

10. Materialized Views:
    - Consider using materialized views for precomputed aggregations or complex queries.
    - Materialized views store results physically, reducing the need to recalculate values during each query.

11. Database Sharding:
- If the dataset is massive, consider horizontal partitioning (sharding) to distribute the data across multiple servers.
- This can improve parallel processing and reduce the load on individual nodes.

12. Regular Database Maintenance:
- Perform routine database maintenance tasks such as vacuuming, updating statistics, and reorganizing indexes to keep the database in optimal condition.

13. Use Stored Procedures:
- Store frequently used queries as stored procedures, which can be compiled and optimized by the database engine.
- This reduces the overhead of query parsing and optimization.

14. Network Optimization:
- Optimize the network infrastructure between the application server and the database server to minimize latency.

15. Hardware Scaling:
- Consider scaling the hardware, such as adding more RAM, optimizing disk I/O, or upgrading the CPU, to improve overall system performance.

Employing a combination of these strategies significantly enhances the performance of SQL queries on your dataset. To achieve optimal results, analyze the specific characteristics of your dataset and workload, tailoring these optimizations accordingly.