PSYLIQ

About The Work

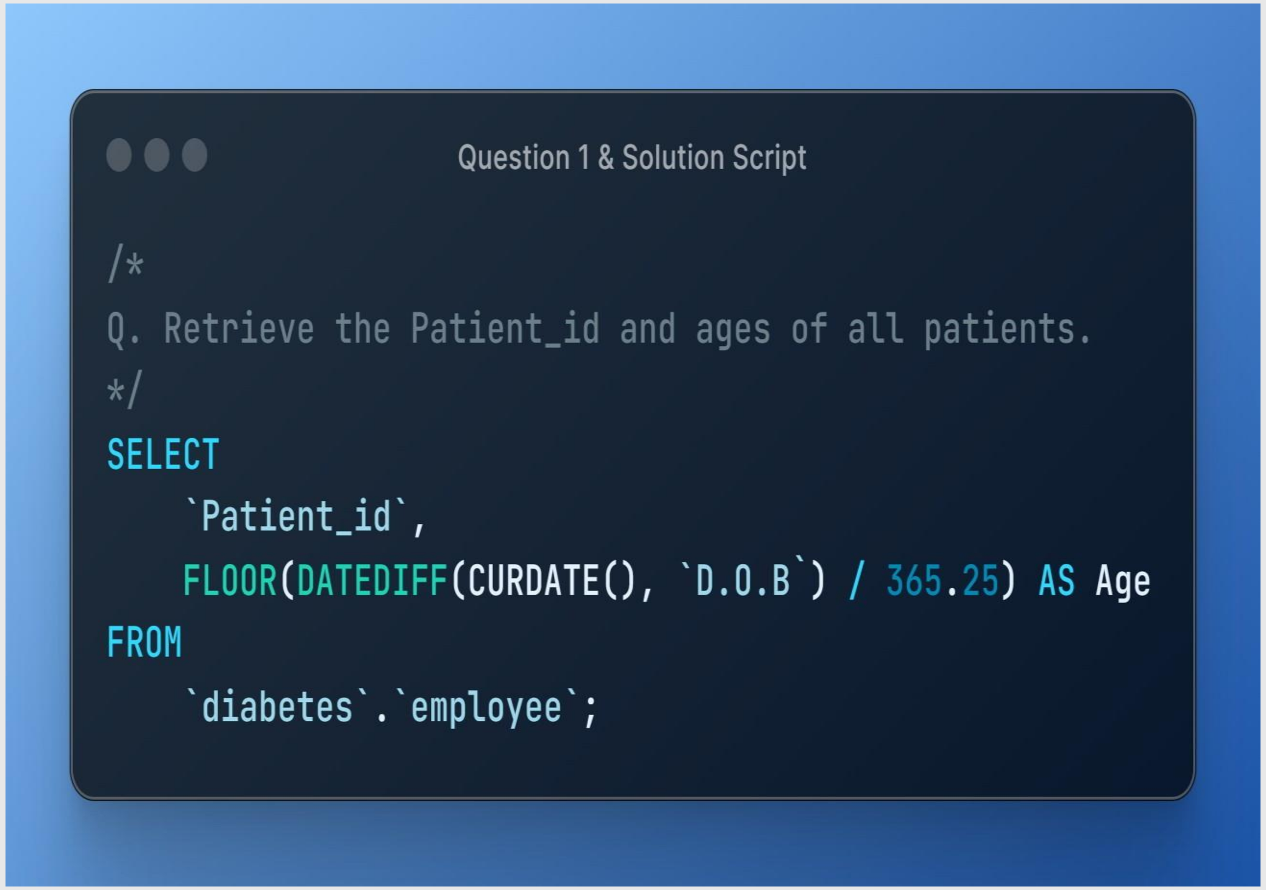
Title: Finding Diabetic Patients Database Insights Using MySQL

Tools Used

In this project, I was entrusted with analyzing a substantial dataset containing diabetic patient information. This data encompassed a range of crucial metrics, including patient identifiers, smoking status, HbA1c readings, blood pressure measurements, body mass index (BMI), and blood glucose levels (mg/dL). My responsibility involved meticulously examining this data to address specific inquiries and implementing targeted modifications within the MySQL database.

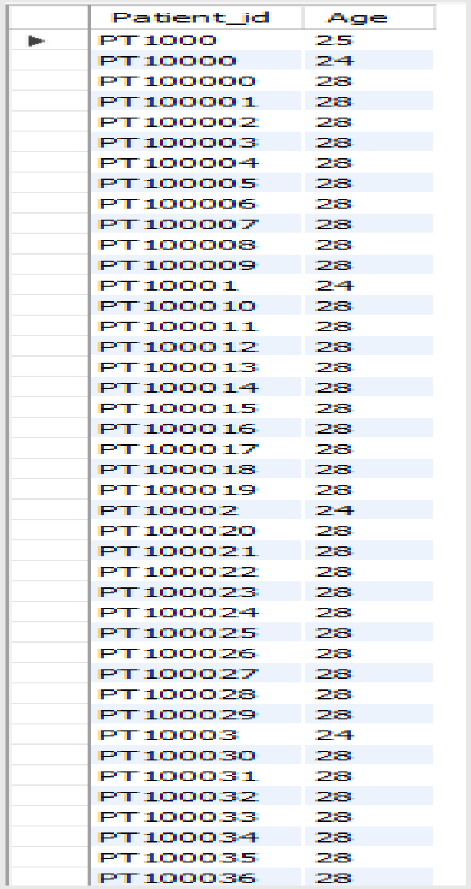
**Please note: Due to limitations in image size, the SQL output snippets included in this report may not display the entire dataset. However, relevant visualizations of the data have been included to effectively communicate the key findings.*

Question |SQL Script |Database Visual



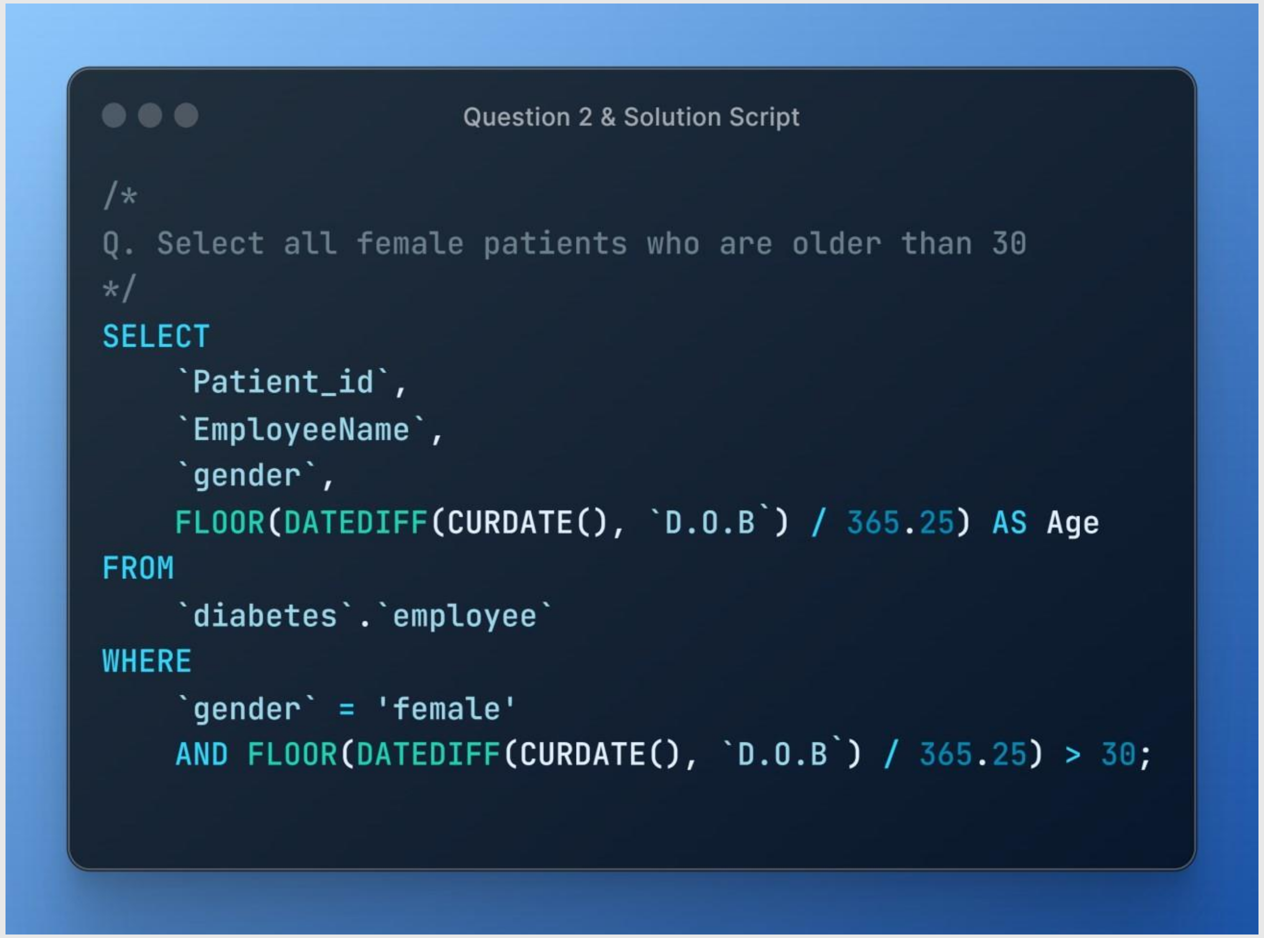
Question 1 & Solution Script

```sql
/*
Q. Retrieve the Patient_id and ages of all patients.
*/
SELECT
    `Patient_id`,
    FLOOR(DATEDIFF(CURDATE(), `D.O.B`) / 365.25) AS Age
FROM
    `diabetes`.`employee`;
```

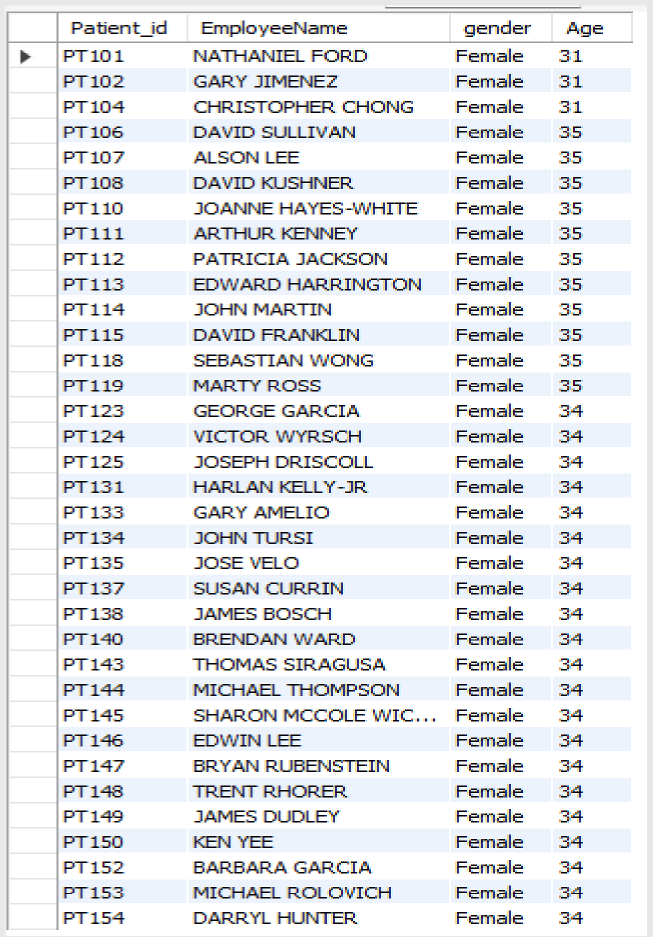


Patient_id	Age
PT1000	25
PT10000	24
PT100000	28
PT100001	28
PT100002	28
PT100003	28
PT100004	28
PT100005	28
PT100006	28
PT100007	28
PT100008	28
PT100009	28
PT10001	24
PT100010	28
PT100011	28
PT100012	28
PT100013	28
PT100014	28
PT100015	28
PT100016	28
PT100017	28
PT100018	28
PT100019	28
PT10002	24
PT100020	28
PT100021	28
PT100022	28
PT100023	28
PT100024	28
PT100025	28
PT100026	28
PT100027	28
PT100028	28
PT100029	28
PT10003	24
PT100030	28
PT100031	28
PT100032	28
PT100033	28
PT100034	28
PT100035	28
PT100036	28

Question |SQL Script |Database Visual

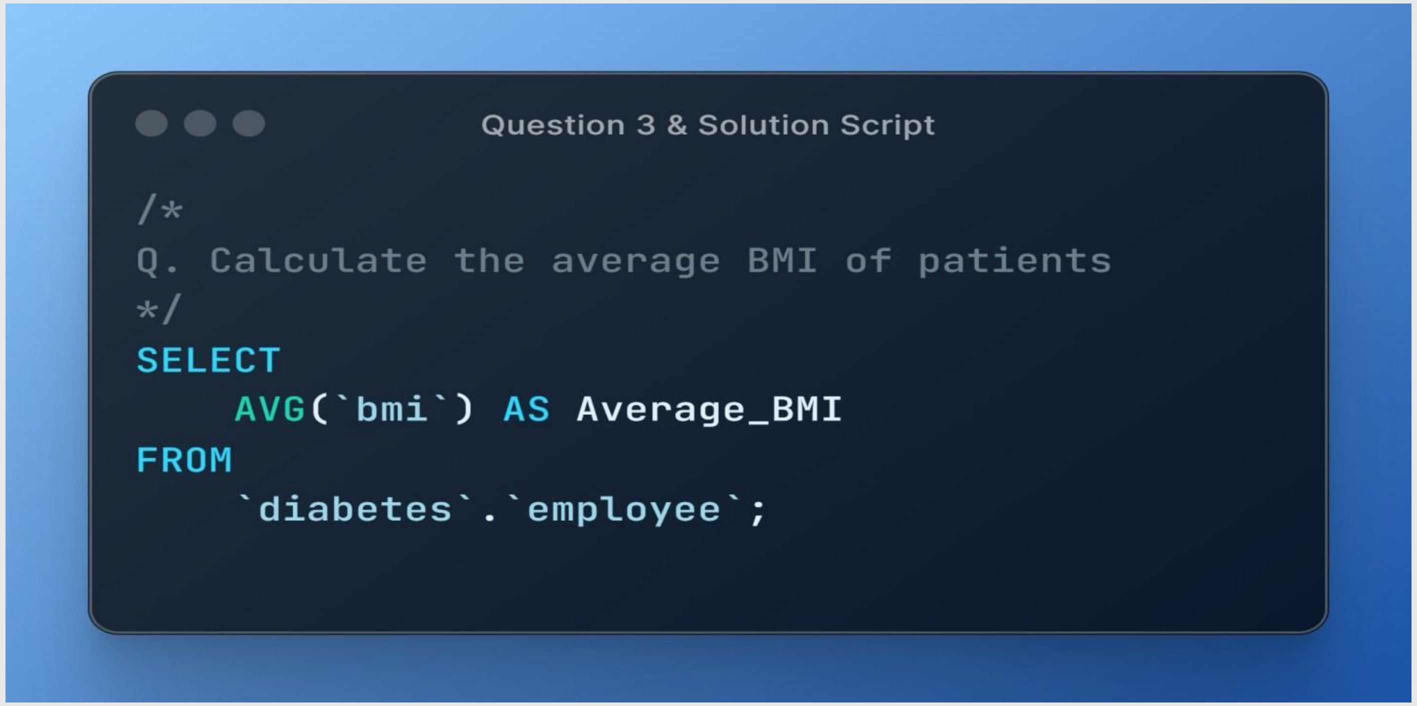

Question 2 & Solution Script

```
/*
Q. Select all female patients who are older than 30
*/
SELECT
    `Patient_id`,
    `EmployeeName`,
    `gender`,
    FLOOR(DATEDIFF(CURDATE(), `D.O.B`) / 365.25) AS Age
FROM
    `diabetes`.`employee`
WHERE
    `gender` = 'female'
    AND FLOOR(DATEDIFF(CURDATE(), `D.O.B`) / 365.25) > 30;
```



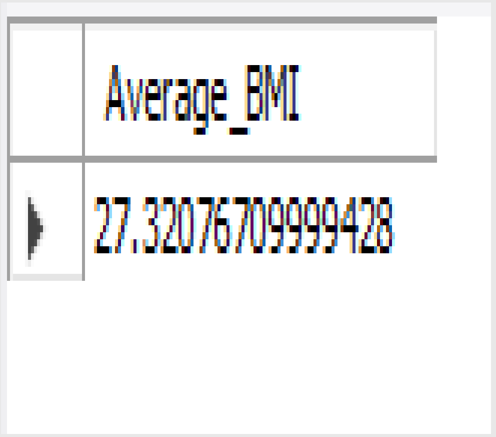
	Patient_id	EmployeeName	gender	Age
▶	PT101	NATHANIEL FORD	Female	31
	PT102	GARY JIMENEZ	Female	31
	PT104	CHRISTOPHER CHONG	Female	31
	PT106	DAVID SULLIVAN	Female	35
	PT107	ALSON LEE	Female	35
	PT108	DAVID KUSHNER	Female	35
	PT110	JOANNE HAYES-WHITE	Female	35
	PT111	ARTHUR KENNEY	Female	35
	PT112	PATRICIA JACKSON	Female	35
	PT113	EDWARD HARRINGTON	Female	35
	PT114	JOHN MARTIN	Female	35
	PT115	DAVID FRANKLIN	Female	35
	PT118	SEBASTIAN WONG	Female	35
	PT119	MARTY ROSS	Female	35
	PT123	GEORGE GARCIA	Female	34
	PT124	VICTOR WYRSCH	Female	34
	PT125	JOSEPH DRISCOLL	Female	34
	PT131	HARLAN KELLY-JR	Female	34
	PT133	GARY AMELIO	Female	34
	PT134	JOHN TURSI	Female	34
	PT135	JOSE VELO	Female	34
	PT137	SUSAN CURRIN	Female	34
	PT138	JAMES BOSCH	Female	34
	PT140	BRENDAN WARD	Female	34
	PT143	THOMAS SIRAGUSA	Female	34
	PT144	MICHAEL THOMPSON	Female	34
	PT145	SHARON MCCOLE WIC...	Female	34
	PT146	EDWIN LEE	Female	34
	PT147	BRYAN RUBENSTEIN	Female	34
	PT148	TRENT RHORER	Female	34
	PT149	JAMES DUDLEY	Female	34
	PT150	KEN YEE	Female	34
	PT152	BARBARA GARCIA	Female	34
	PT153	MICHAEL ROLOVICH	Female	34
	PT154	DARRYL HUNTER	Female	34

Question |SQL Script |Database Visual



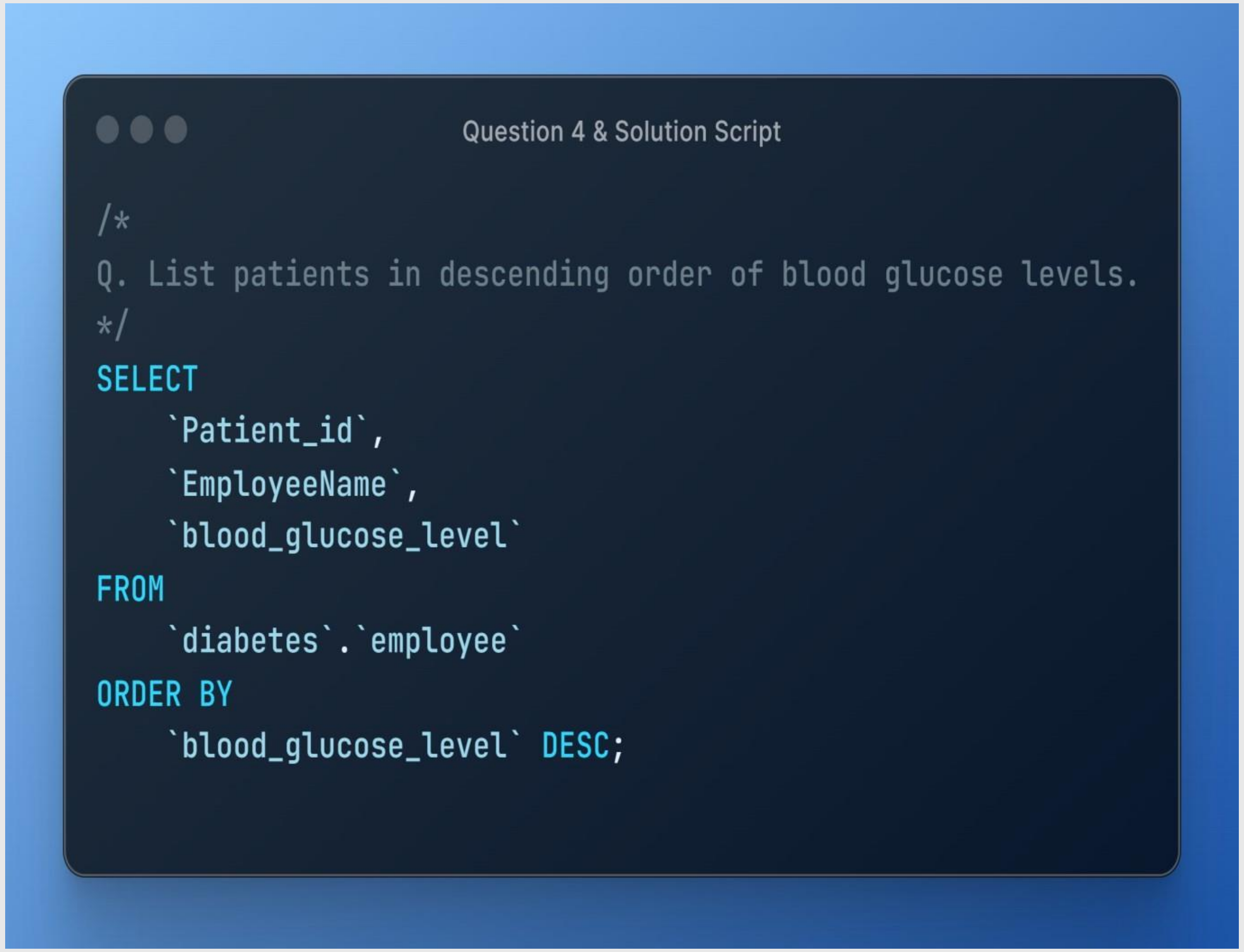
Question 3 & Solution Script

```
/*
Q. Calculate the average BMI of patients
*/
SELECT
    AVG(`bmi`) AS Average_BMI
FROM
    `diabetes`.`employee`;
```



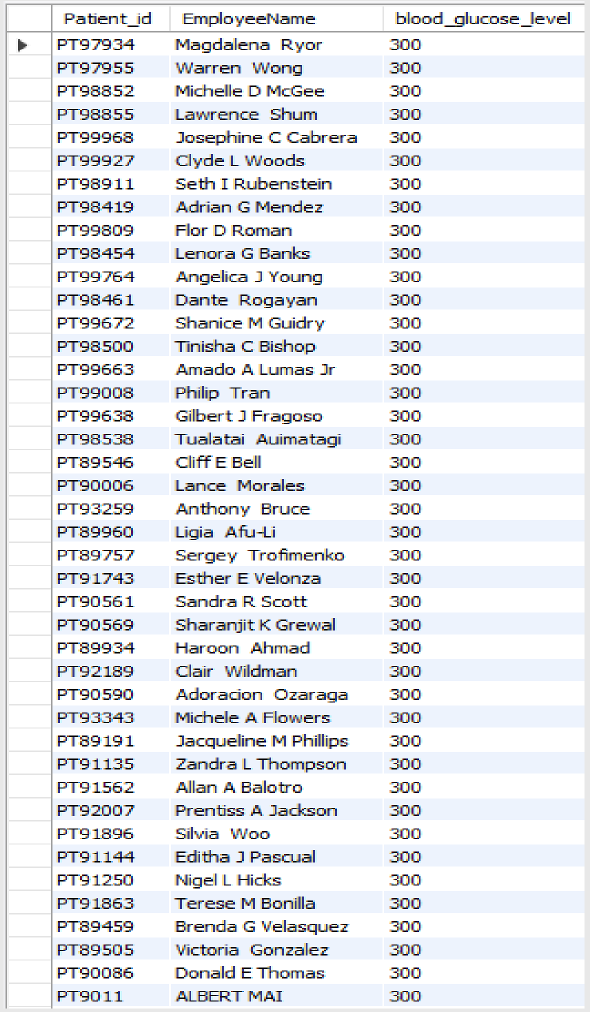
	Average_BMI
▶	27.3207670999428

Question |SQL Script |Database Visual



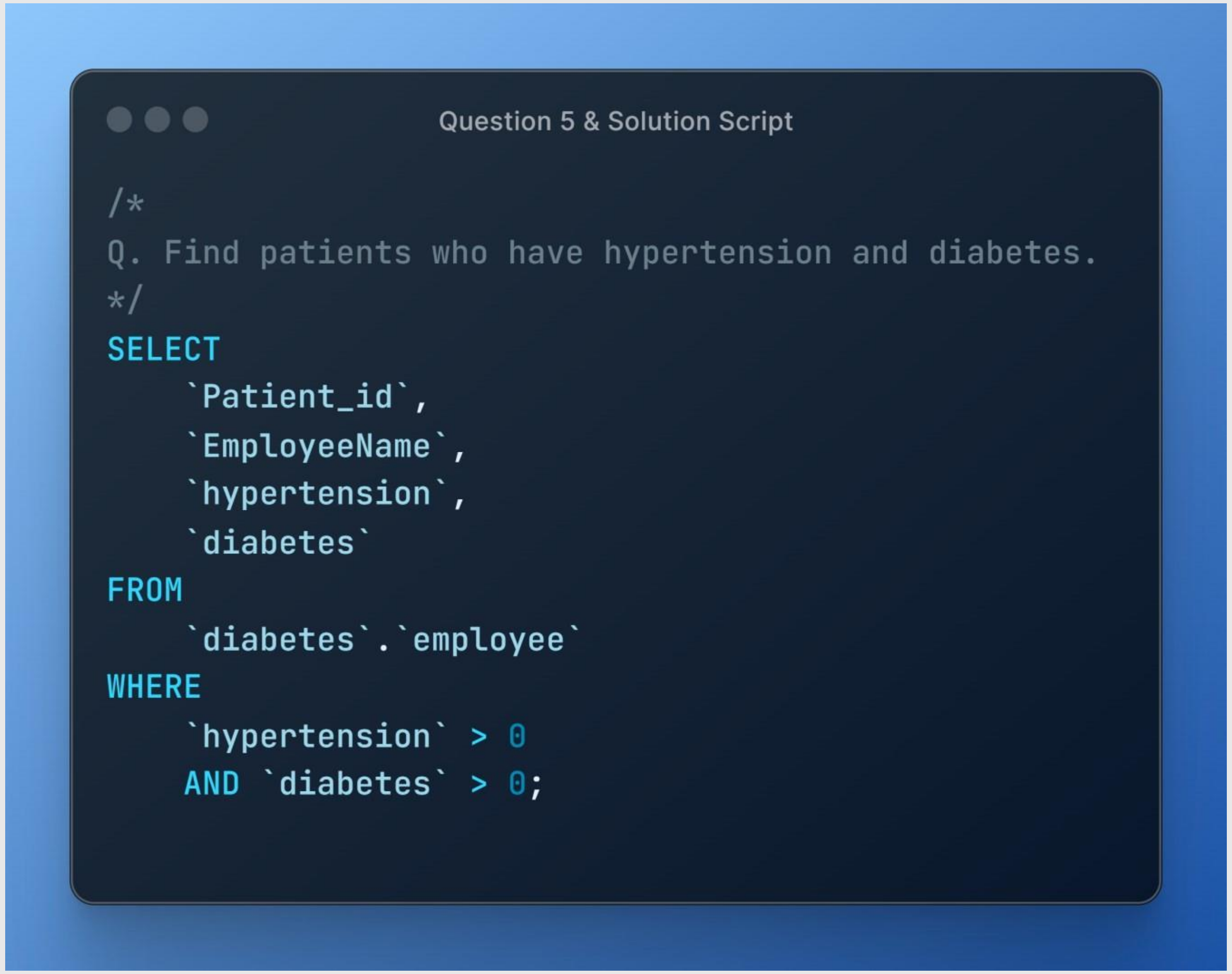
Question 4 & Solution Script

```sql
/*
Q. List patients in descending order of blood glucose levels.
*/
SELECT
    `Patient_id`,
    `EmployeeName`,
    `blood_glucose_level`
FROM
    `diabetes`.`employee`
ORDER BY
    `blood_glucose_level` DESC;
```



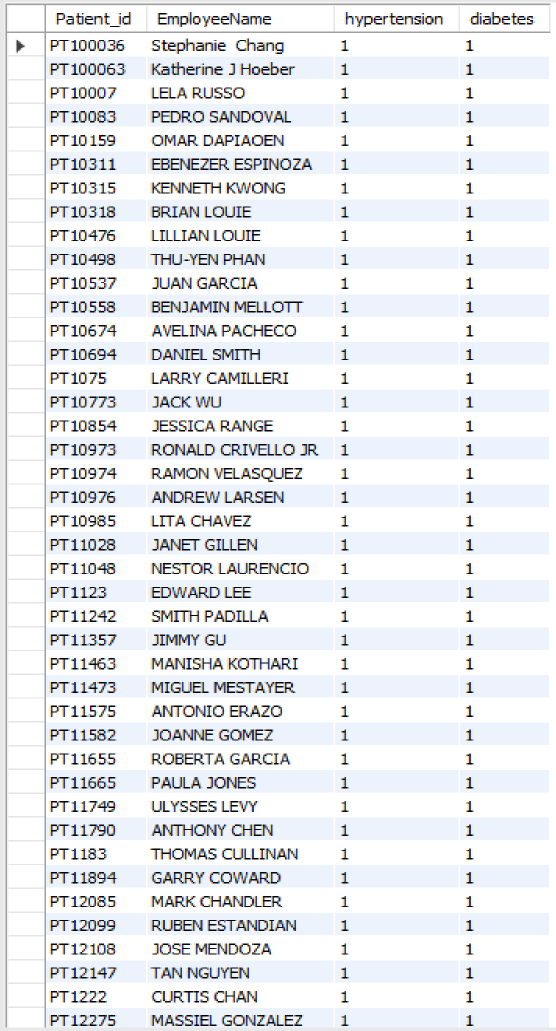
Patient_id	EmployeeName	blood_glucose_level
PT97934	Magdalena Ryor	300
PT97955	Warren Wong	300
PT98852	Michelle D McGee	300
PT98855	Lawrence Shum	300
PT99968	Josephine C Cabrera	300
PT99927	Clyde L Woods	300
PT98911	Seth I Rubenstein	300
PT98419	Adrian G Mendez	300
PT99809	Flor D Roman	300
PT98454	Lenora G Banks	300
PT99764	Angelica J Young	300
PT98461	Dante Rogayan	300
PT99672	Shanice M Guidry	300
PT98500	Tinisha C Bishop	300
PT99663	Amado A Lumas Jr	300
PT99008	Philip Tran	300
PT99638	Gilbert J Fragoso	300
PT98538	Tualatai Auimatagi	300
PT89546	Cliff E Bell	300
PT90006	Lance Morales	300
PT93259	Anthony Bruce	300
PT89960	Ligia Afu-Li	300
PT89757	Sergey Trofimenko	300
PT91743	Esther E Velonza	300
PT90561	Sandra R Scott	300
PT90569	Sharanjit K Grewal	300
PT89934	Haroon Ahmad	300
PT92189	Clair Wildman	300
PT90590	Adoracion Ozaraga	300
PT93343	Michele A Flowers	300
PT89191	Jacqueline M Phillips	300
PT91135	Zandra L Thompson	300
PT91562	Allan A Balotro	300
PT92007	Prentiss A Jackson	300
PT91896	Silvia Woo	300
PT91144	Editha J Pascual	300
PT91250	Nigel L Hicks	300
PT91863	Terese M Bonilla	300
PT89459	Brenda G Velasquez	300
PT89505	Victoria Gonzalez	300
PT90086	Donald E Thomas	300
PT9011	ALBERT MAI	300

Question |SQL Script |Database Visual



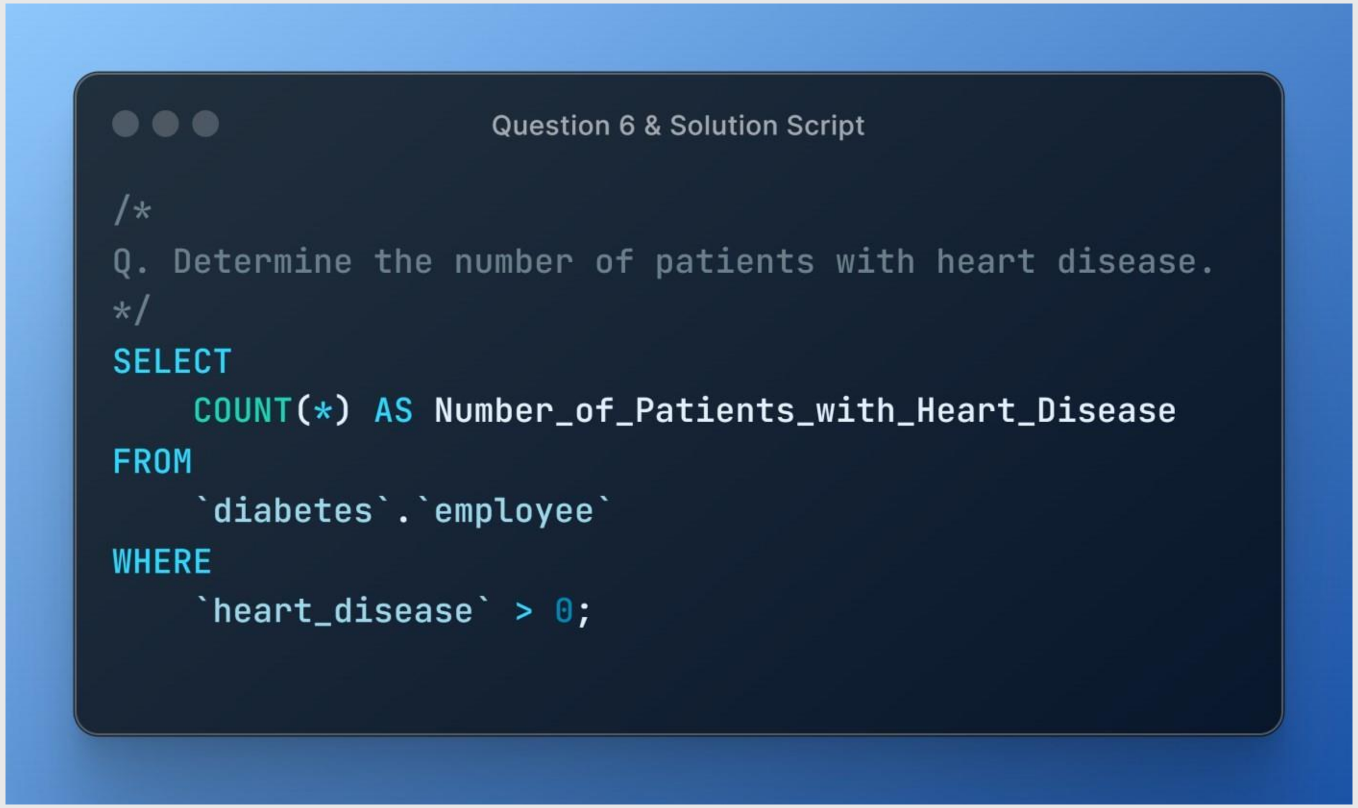
Question 5 & Solution Script

```sql
/*
Q. Find patients who have hypertension and diabetes.
*/
SELECT
    `Patient_id`,
    `EmployeeName`,
    `hypertension`,
    `diabetes`
FROM
    `diabetes`.`employee`
WHERE
    `hypertension` > 0
    AND `diabetes` > 0;
```

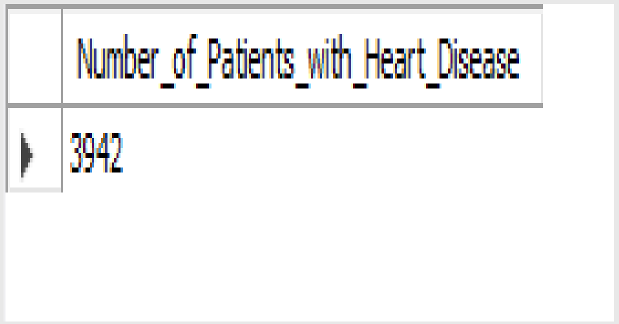


Patient_id	EmployeeName	hypertension	diabetes
PT100036	Stephanie Chang	1	1
PT100063	Katherine J Hoeber	1	1
PT10007	LELA RUSSO	1	1
PT10083	PEDRO SANDOVAL	1	1
PT10159	OMAR DAPIAOEN	1	1
PT10311	EBENEZER ESPINOZA	1	1
PT10315	KENNETH KWONG	1	1
PT10318	BRIAN LOUIE	1	1
PT10476	LILLIAN LOUIE	1	1
PT10498	THU-YEN PHAN	1	1
PT10537	JUAN GARCIA	1	1
PT10558	BENJAMIN MELLOTT	1	1
PT10674	AVELINA PACHECO	1	1
PT10694	DANIEL SMITH	1	1
PT1075	LARRY CAMILLERI	1	1
PT10773	JACK WU	1	1
PT10854	JESSICA RANGE	1	1
PT10973	RONALD CRIVELLO JR	1	1
PT10974	RAMON VELASQUEZ	1	1
PT10976	ANDREW LARSEN	1	1
PT10985	LITA CHAVEZ	1	1
PT11028	JANET GILLEN	1	1
PT11048	NESTOR LAURENCIO	1	1
PT1123	EDWARD LEE	1	1
PT11242	SMITH PADILLA	1	1
PT11357	JIMMY GU	1	1
PT11463	MANISHA KOTHARI	1	1
PT11473	MIGUEL MESTAYER	1	1
PT11575	ANTONIO ERAZO	1	1
PT11582	JOANNE GOMEZ	1	1
PT11655	ROBERTA GARCIA	1	1
PT11665	PAULA JONES	1	1
PT11749	ULYSSES LEVY	1	1
PT11790	ANTHONY CHEN	1	1
PT1183	THOMAS CULLINAN	1	1
PT11894	GARRY COWARD	1	1
PT12085	MARK CHANDLER	1	1
PT12099	RUBEN ESTANDIAN	1	1
PT12108	JOSE MENDOZA	1	1
PT12147	TAN NGUYEN	1	1
PT1222	CURTIS CHAN	1	1
PT12275	MASSIEL GONZALEZ	1	1

Question |SQL Script |Database Visual

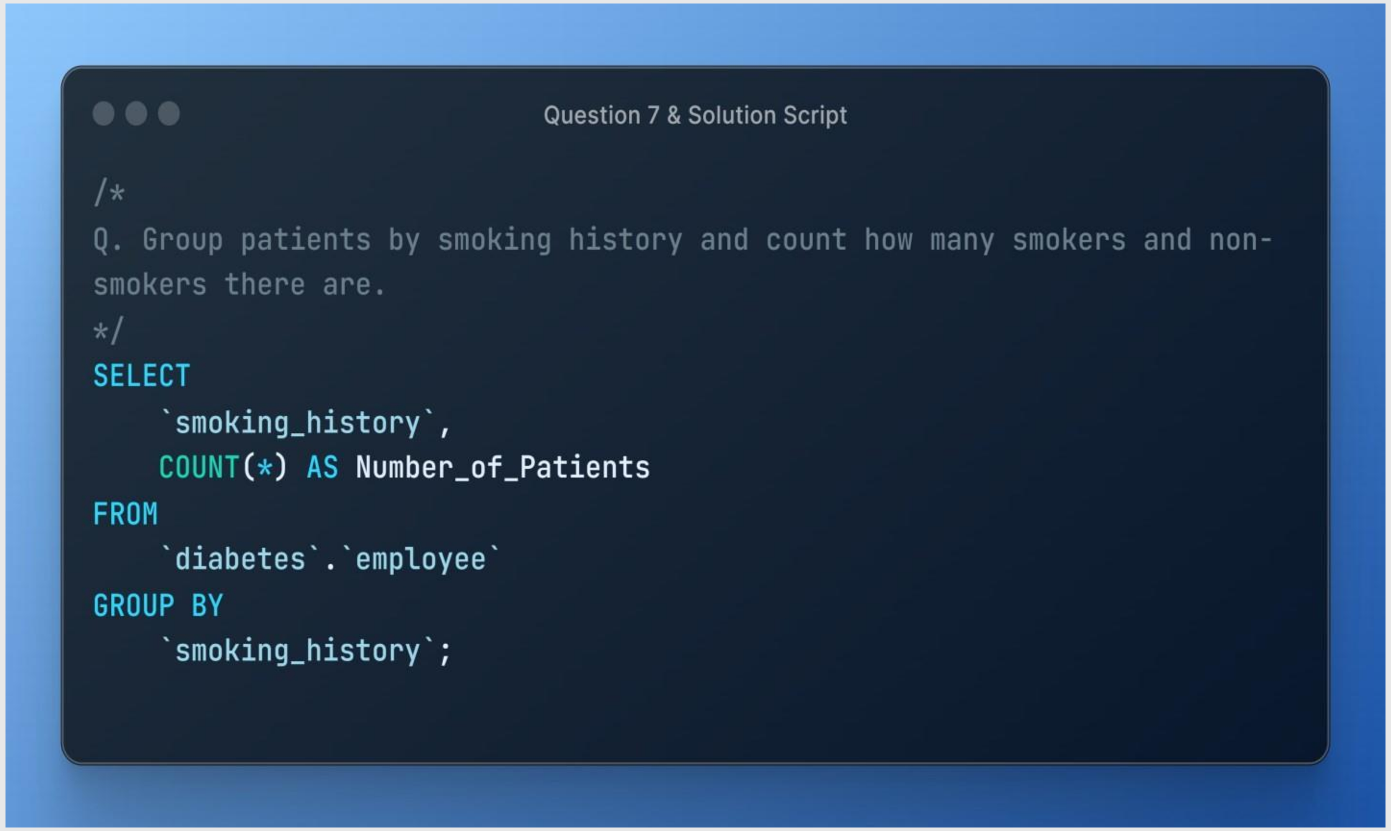

Question 6 & Solution Script

```sql
/*
Q. Determine the number of patients with heart disease.
*/
SELECT
    COUNT(*) AS Number_of_Patients_with_Heart_Disease
FROM
    `diabetes`.`employee`
WHERE
    `heart_disease` > 0;
```

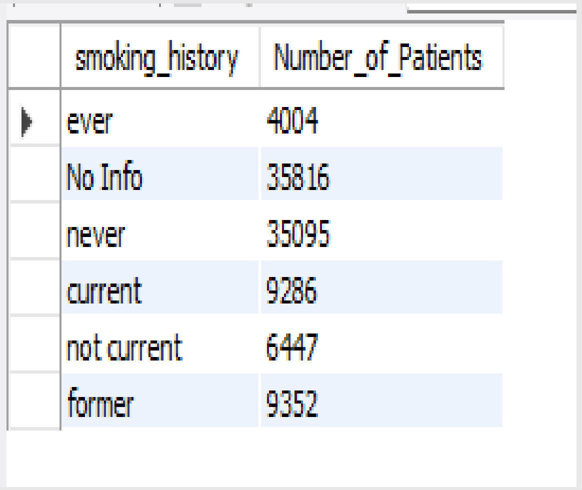


	Number_of_Patients_with_Heart_Disease
▶	3942

Question |SQL Script |Database Visual

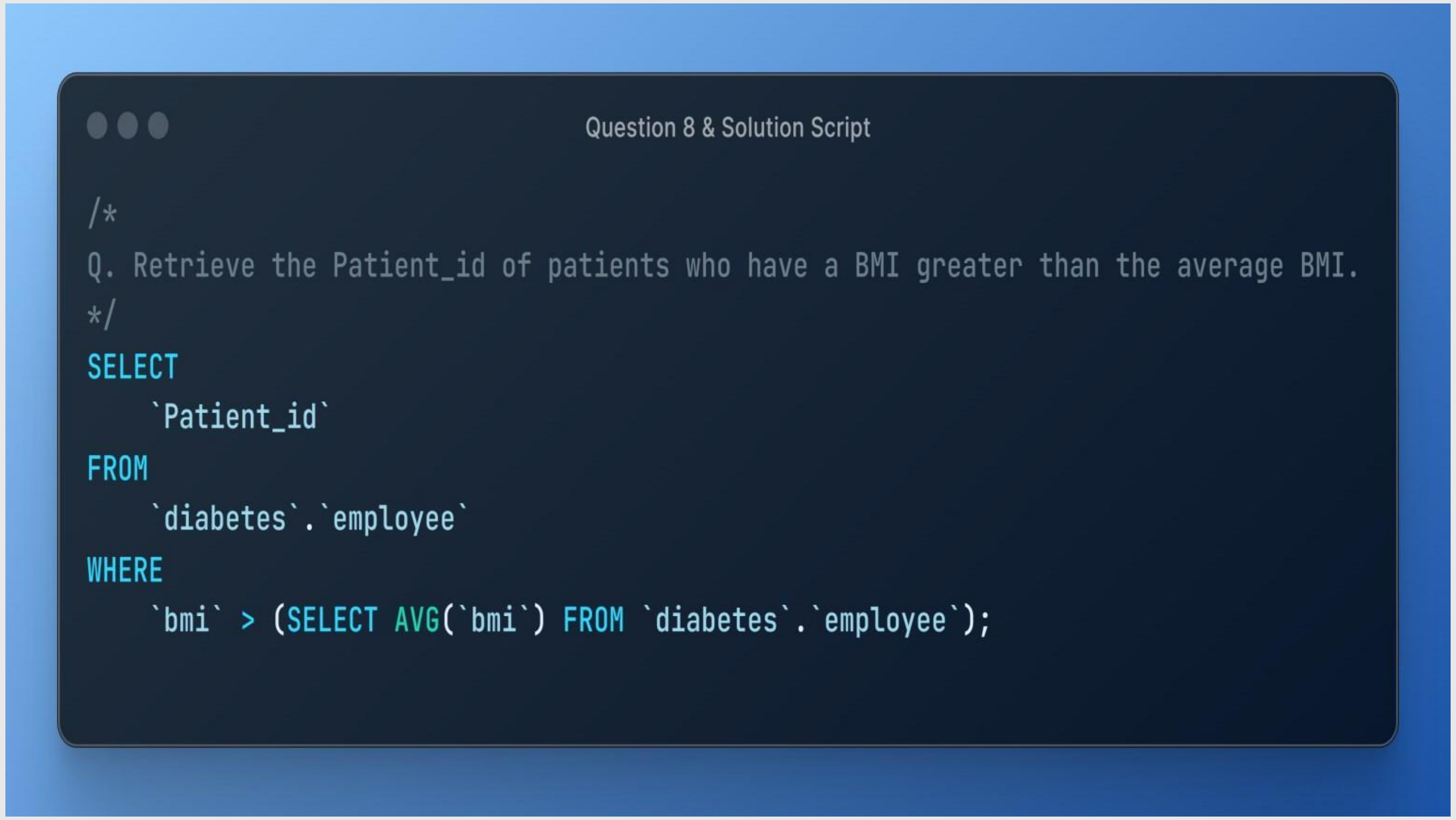

Question 7 & Solution Script

```
/*
Q. Group patients by smoking history and count how many smokers and non-
smokers there are.
*/
SELECT
    `smoking_history`,
    COUNT(*) AS Number_of_Patients
FROM
    `diabetes`.`employee`
GROUP BY
    `smoking_history`;
```



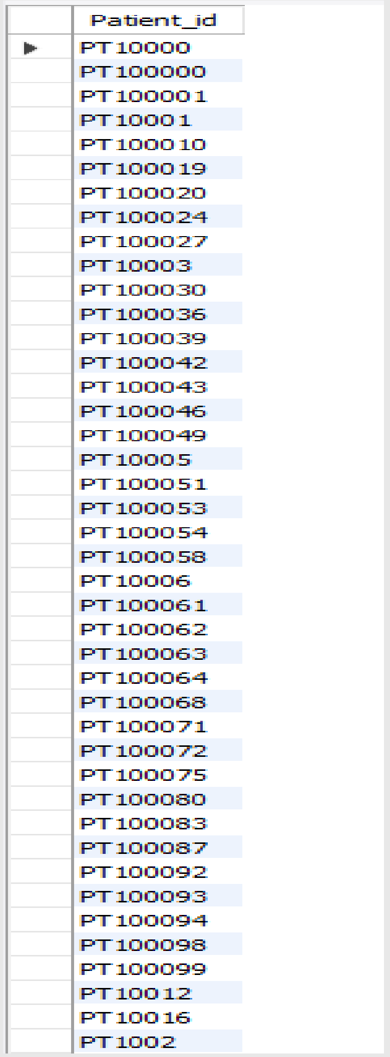
smoking_history	Number_of_Patients
ever	4004
No Info	35816
never	35095
current	9286
not current	6447
former	9352

Question |SQL Script |Database Visual



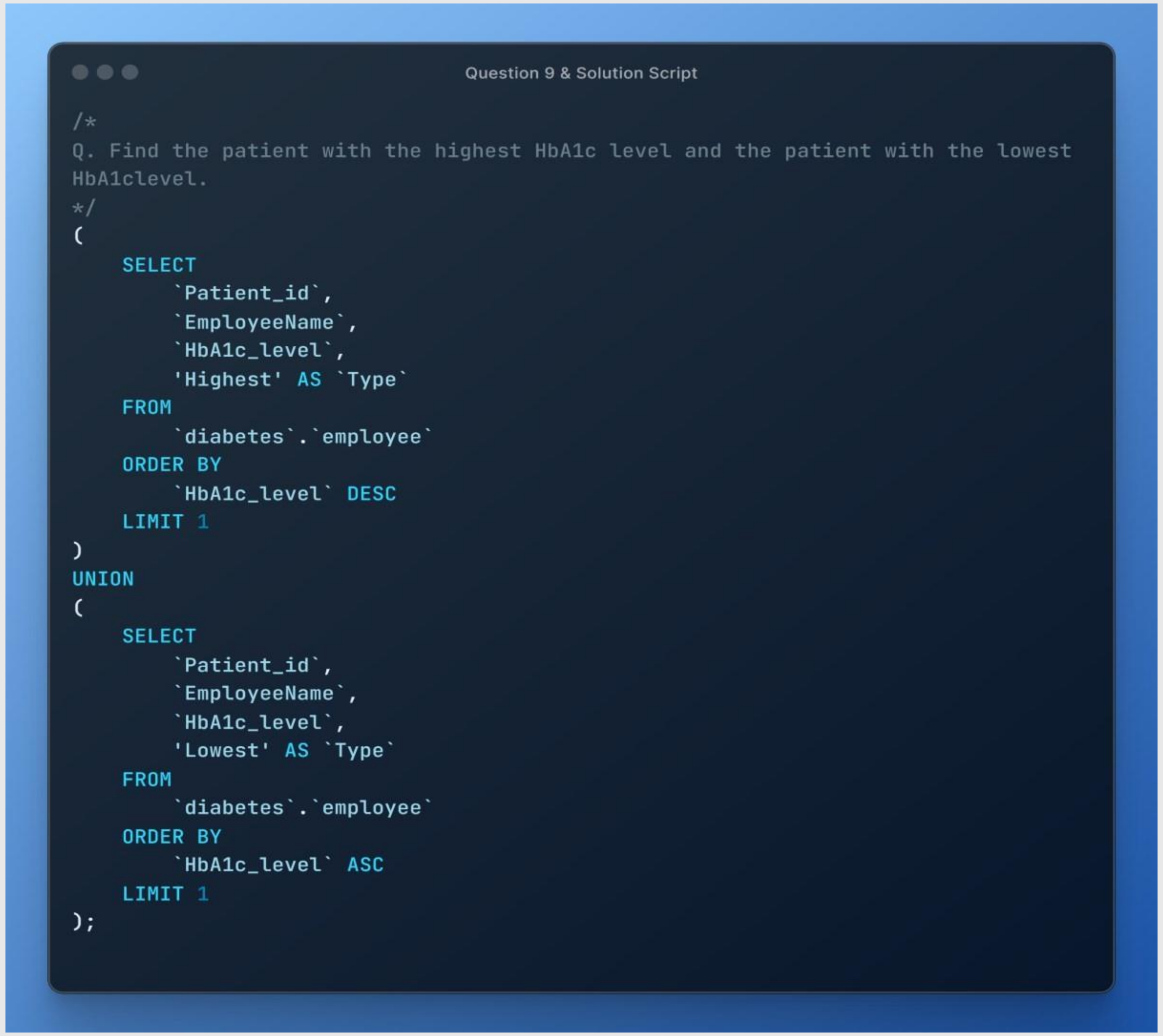
Question 8 & Solution Script

```sql
/*
Q. Retrieve the Patient_id of patients who have a BMI greater than the average BMI.
*/
SELECT
    `Patient_id`
FROM
    `diabetes`.`employee`
WHERE
    `bmi` > (SELECT AVG(`bmi`) FROM `diabetes`.`employee`);
```



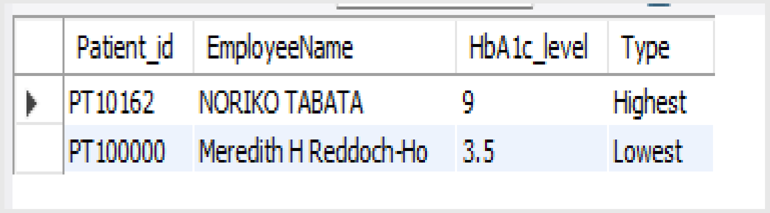
	Patient_id
▶	PT10000
	PT100000
	PT100001
	PT10001
	PT100010
	PT100019
	PT100020
	PT100024
	PT100027
	PT10003
	PT100030
	PT100036
	PT100039
	PT100042
	PT100043
	PT100046
	PT100049
	PT10005
	PT100051
	PT100053
	PT100054
	PT100058
	PT10006
	PT100061
	PT100062
	PT100063
	PT100064
	PT100068
	PT100071
	PT100072
	PT100075
	PT100080
	PT100083
	PT100087
	PT100092
	PT100093
	PT100094
	PT100098
	PT100099
	PT10012
	PT10016
	PT1002

Question |SQL Script |Database Visual



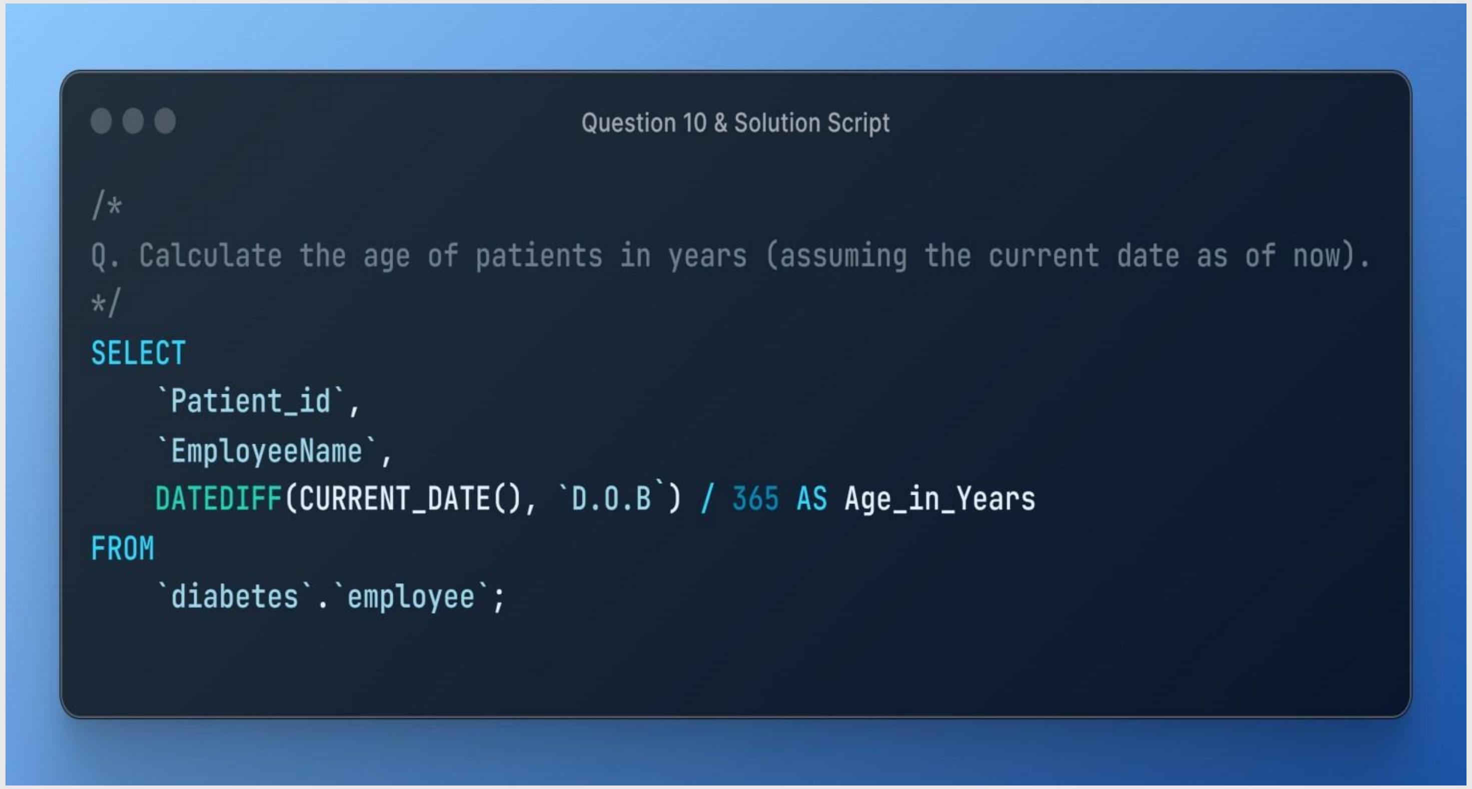
Question 9 & Solution Script

```
/*
Q. Find the patient with the highest HbA1c level and the patient with the lowest
HbA1clevel.
*/
(
    SELECT
        `Patient_id`,
        `EmployeeName`,
        `HbA1c_level`,
        'Highest' AS `Type`
    FROM
        `diabetes`.`employee`
    ORDER BY
        `HbA1c_level` DESC
    LIMIT 1
)
UNION
(
    SELECT
        `Patient_id`,
        `EmployeeName`,
        `HbA1c_level`,
        'Lowest' AS `Type`
    FROM
        `diabetes`.`employee`
    ORDER BY
        `HbA1c_level` ASC
    LIMIT 1
);
```



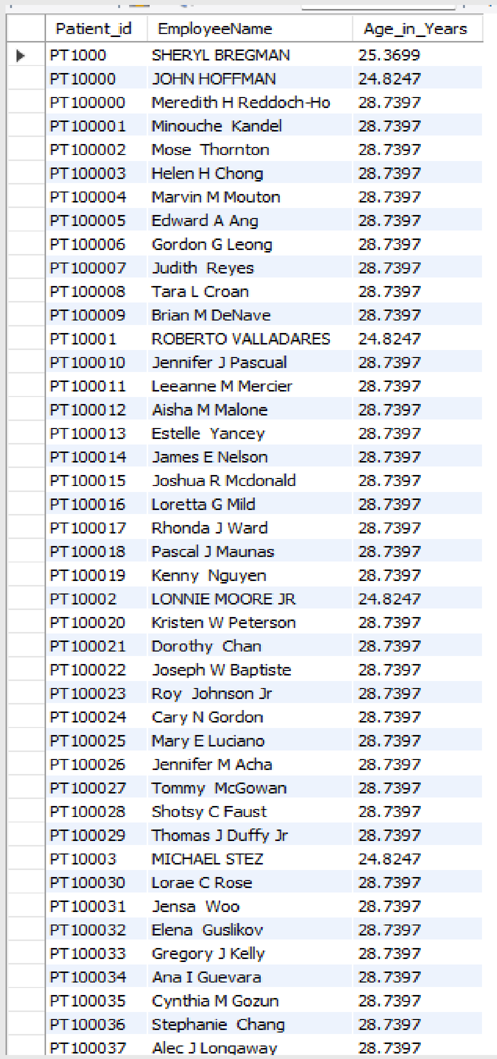
Patient_id	EmployeeName	HbA1c_level	Type
PT10162	NORIKO TABATA	9	Highest
PT100000	Meredith H Reddoch-Ho	3.5	Lowest

Question |SQL Script |Database Visual



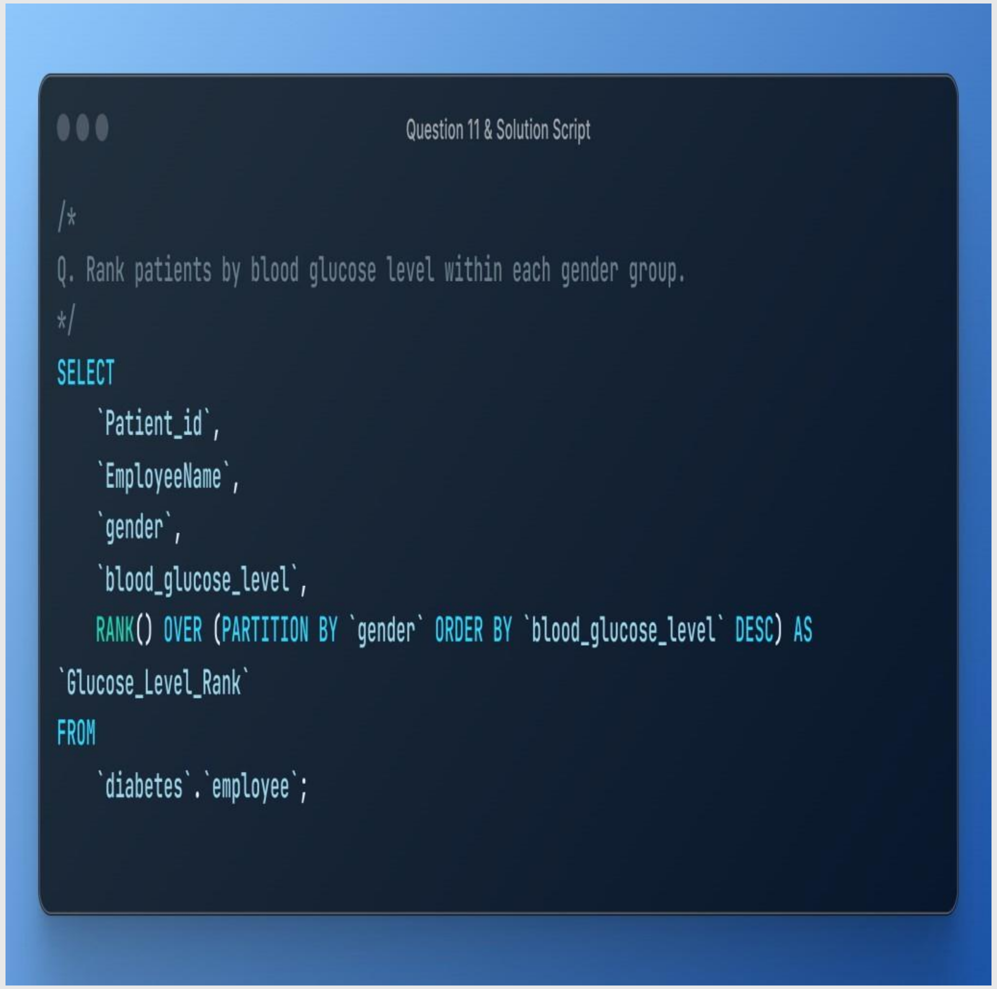
Question 10 & Solution Script

```
/*
Q. Calculate the age of patients in years (assuming the current date as of now).
*/
SELECT
    `Patient_id`,
    `EmployeeName`,
    DATEDIFF(CURRENT_DATE(), `D.O.B`) / 365 AS Age_in_Years
FROM
    `diabetes`.`employee`;
```



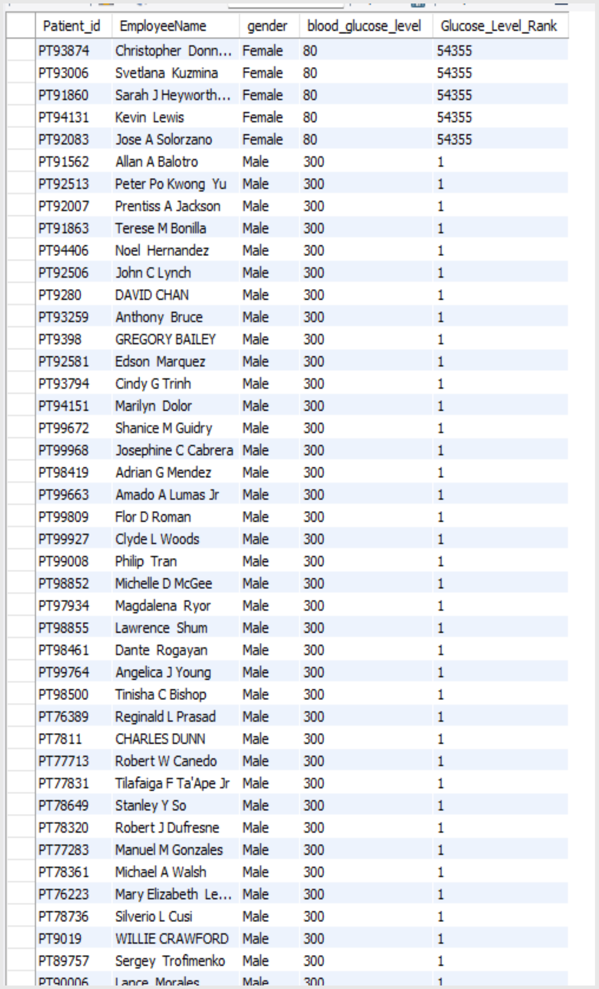
Patient_id	EmployeeName	Age_in_Years
PT1000	SHERYL BREGMAN	25.3699
PT10000	JOHN HOFFMAN	24.8247
PT100000	Meredith H Reddoch-Ho	28.7397
PT100001	Minouche Kandel	28.7397
PT100002	Mose Thornton	28.7397
PT100003	Helen H Chong	28.7397
PT100004	Marvin M Mouton	28.7397
PT100005	Edward A Ang	28.7397
PT100006	Gordon G Leong	28.7397
PT100007	Judith Reyes	28.7397
PT100008	Tara L Croan	28.7397
PT100009	Brian M DeNave	28.7397
PT10001	ROBERTO VALLADARES	24.8247
PT100010	Jennifer J Pascual	28.7397
PT100011	Leeanne M Mercier	28.7397
PT100012	Aisha M Malone	28.7397
PT100013	Estelle Yancey	28.7397
PT100014	James E Nelson	28.7397
PT100015	Joshua R Mcdonald	28.7397
PT100016	Loretta G Mild	28.7397
PT100017	Rhonda J Ward	28.7397
PT100018	Pascal J Maunas	28.7397
PT100019	Kenny Nguyen	28.7397
PT10002	LONNIE MOORE JR	24.8247
PT100020	Kristen W Peterson	28.7397
PT100021	Dorothy Chan	28.7397
PT100022	Joseph W Baptiste	28.7397
PT100023	Roy Johnson Jr	28.7397
PT100024	Cary N Gordon	28.7397
PT100025	Mary E Luciano	28.7397
PT100026	Jennifer M Acha	28.7397
PT100027	Tommy McGowan	28.7397
PT100028	Shotsy C Faust	28.7397
PT100029	Thomas J Duffy Jr	28.7397
PT10003	MICHAEL STEZ	24.8247
PT100030	Lorae C Rose	28.7397
PT100031	Jensa Woo	28.7397
PT100032	Elena Guslikov	28.7397
PT100033	Gregory J Kelly	28.7397
PT100034	Ana I Guevara	28.7397
PT100035	Cynthia M Gozun	28.7397
PT100036	Stephanie Chang	28.7397
PT100037	Alec J Longaway	28.7397

Question |SQL Script |Database Visual

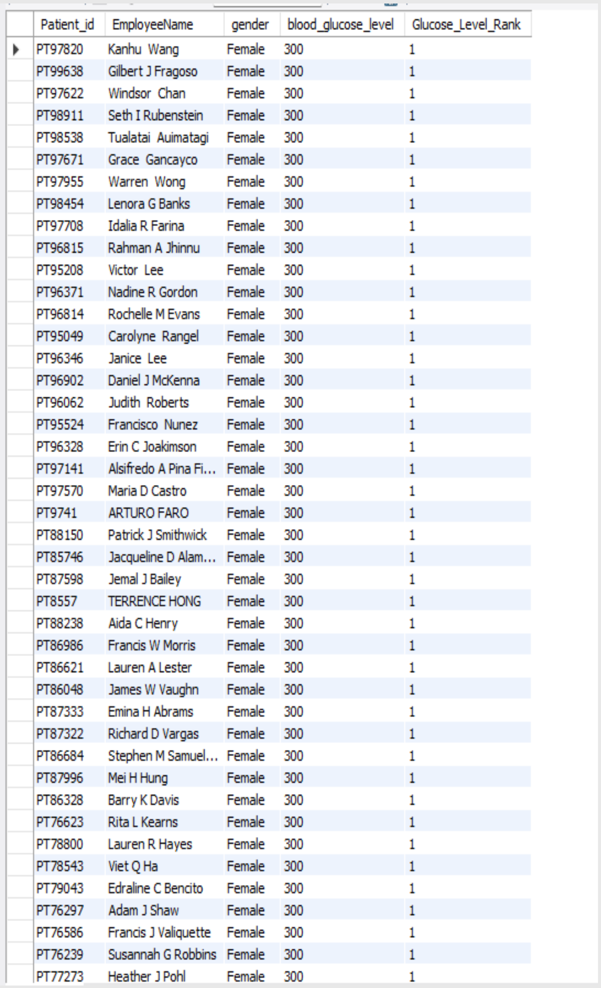


Question 11 & Solution Script

```
/*
Q. Rank patients by blood glucose level within each gender group.
*/
SELECT
    `Patient_id`,
    `EmployeeName`,
    `gender`,
    `blood_glucose_level`,
    RANK() OVER (PARTITION BY `gender` ORDER BY `blood_glucose_level` DESC) AS
`Glucose_Level_Rank`
FROM
    `diabetes`.`employee`;
```

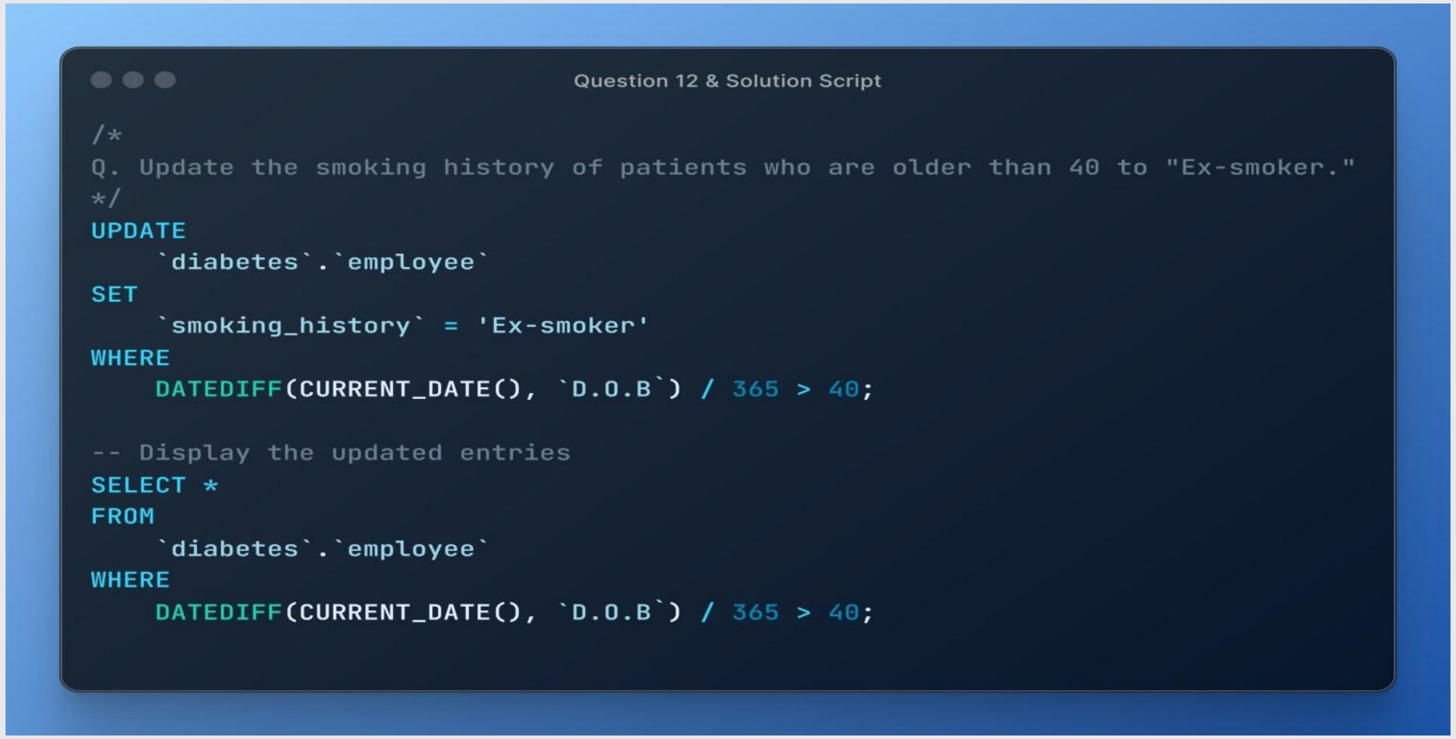


Patient_id	EmployeeName	gender	blood_glucose_level	Glucose_Level_Rank
PT93874	Christopher Donn...	Female	80	54355
PT93006	Svetlana Kuzmina	Female	80	54355
PT91860	Sarah J Heyworth...	Female	80	54355
PT94131	Kevin Lewis	Female	80	54355
PT92083	Jose A Solorzano	Female	80	54355
PT91562	Allan A Balotro	Male	300	1
PT92513	Peter Po Kwong Yu	Male	300	1
PT92007	Prentiss A Jackson	Male	300	1
PT91863	Terese M Bonilla	Male	300	1
PT94406	Noel Hernandez	Male	300	1
PT92506	John C Lynch	Male	300	1
PT9280	DAVID CHAN	Male	300	1
PT93259	Anthony Bruce	Male	300	1
PT9398	GREGORY BAILEY	Male	300	1
PT92581	Edson Marquez	Male	300	1
PT93794	Cindy G Trinh	Male	300	1
PT94151	Marilyn Dolor	Male	300	1
PT99672	Shanice M Guidry	Male	300	1
PT99968	Josephine C Cabrera	Male	300	1
PT98419	Adrian G Mendez	Male	300	1
PT99663	Amado A Lumas Jr	Male	300	1
PT99809	Flor D Roman	Male	300	1
PT99927	Clyde L Woods	Male	300	1
PT99008	Philip Tran	Male	300	1
PT98852	Michelle D McGee	Male	300	1
PT97934	Magdalena Ryor	Male	300	1
PT98855	Lawrence Shum	Male	300	1
PT98461	Dante Rogayan	Male	300	1
PT99764	Angelica J Young	Male	300	1
PT98500	Tinisha C Bishop	Male	300	1
PT76389	Reginald L Prasad	Male	300	1
PT7811	CHARLES DUNN	Male	300	1
PT77713	Robert W Canedo	Male	300	1
PT77831	Tilafaiga F Ta'Ape Jr	Male	300	1
PT78649	Stanley Y So	Male	300	1
PT78320	Robert J Dufresne	Male	300	1
PT77283	Manuel M Gonzales	Male	300	1
PT78361	Michael A Walsh	Male	300	1
PT76223	Mary Elizabeth Le...	Male	300	1
PT78736	Silverio L Cusi	Male	300	1
PT9019	WILLIE CRAWFORD	Male	300	1
PT89757	Sergey Trofimenko	Male	300	1
PT90006	Lance Morales	Male	300	1



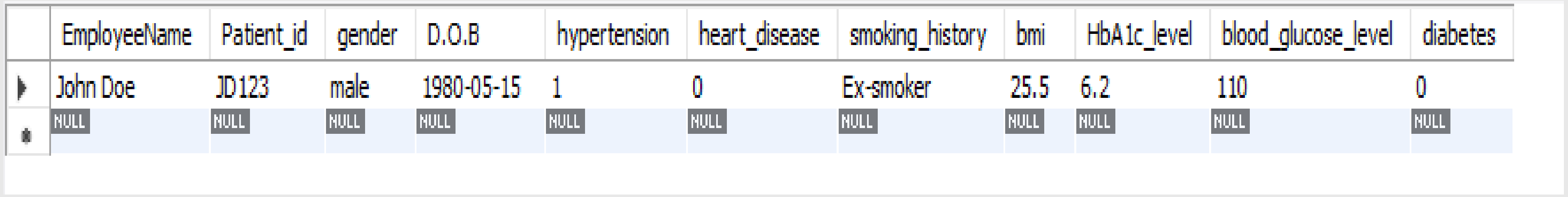
Patient_id	EmployeeName	gender	blood_glucose_level	Glucose_Level_Rank
PT97820	Kanhu Wang	Female	300	1
PT99638	Gilbert J Fragoso	Female	300	1
PT97622	Windsor Chan	Female	300	1
PT98911	Seth I Rubenstein	Female	300	1
PT98538	Tualatai Auimatagi	Female	300	1
PT97671	Grace Gancayco	Female	300	1
PT97955	Warren Wong	Female	300	1
PT98454	Lenora G Banks	Female	300	1
PT97708	Idalia R Farina	Female	300	1
PT96815	Rahman A Jhinnu	Female	300	1
PT95208	Victor Lee	Female	300	1
PT96371	Nadine R Gordon	Female	300	1
PT96814	Rochelle M Evans	Female	300	1
PT95049	Carolyne Rangel	Female	300	1
PT96346	Janice Lee	Female	300	1
PT96902	Daniel J McKenna	Female	300	1
PT96062	Judith Roberts	Female	300	1
PT95524	Francisco Nunez	Female	300	1
PT96328	Erin C Joakimson	Female	300	1
PT97141	Alsifredo A Pina Fi...	Female	300	1
PT97570	Maria D Castro	Female	300	1
PT9741	ARTURO FARO	Female	300	1
PT88150	Patrick J Smithwick	Female	300	1
PT85746	Jacqueline D Alam...	Female	300	1
PT87598	Jemal J Bailey	Female	300	1
PT8557	TERRENCE HONG	Female	300	1
PT88238	Aida C Henry	Female	300	1
PT86986	Francis W Morris	Female	300	1
PT86621	Lauren A Lester	Female	300	1
PT86048	James W Vaughn	Female	300	1
PT87333	Emina H Abrams	Female	300	1
PT87322	Richard D Vargas	Female	300	1
PT86684	Stephen M Samuel...	Female	300	1
PT87996	Mei H Hung	Female	300	1
PT86328	Barry K Davis	Female	300	1
PT76623	Rita L Kearns	Female	300	1
PT78800	Lauren R Hayes	Female	300	1
PT78543	Viet Q Ha	Female	300	1
PT79043	Edraline C Bencito	Female	300	1
PT76297	Adam J Shaw	Female	300	1
PT76586	Francis J Valiquette	Female	300	1
PT76239	Susannah G Robbins	Female	300	1
PT77273	Heather J Pohl	Female	300	1

Question |SQL Script |Database Visual


Question 12 & Solution Script

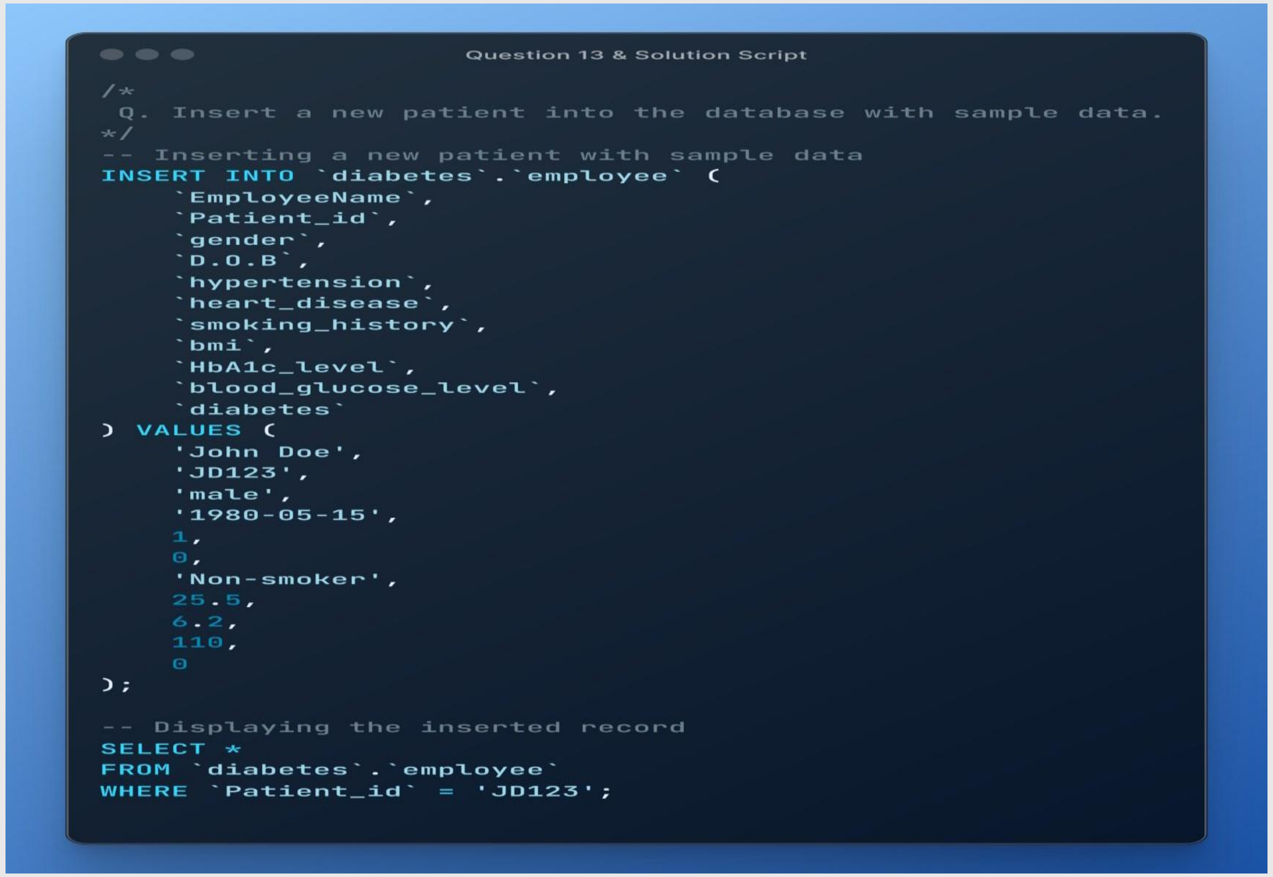
```
/*
Q. Update the smoking history of patients who are older than 40 to "Ex-smoker."
*/
UPDATE
    `diabetes`.`employee`
SET
    `smoking_history` = 'Ex-smoker'
WHERE
    DATEDIFF(CURRENT_DATE(), `D.O.B`) / 365 > 40;

-- Display the updated entries
SELECT *
FROM
    `diabetes`.`employee`
WHERE
    DATEDIFF(CURRENT_DATE(), `D.O.B`) / 365 > 40;
```



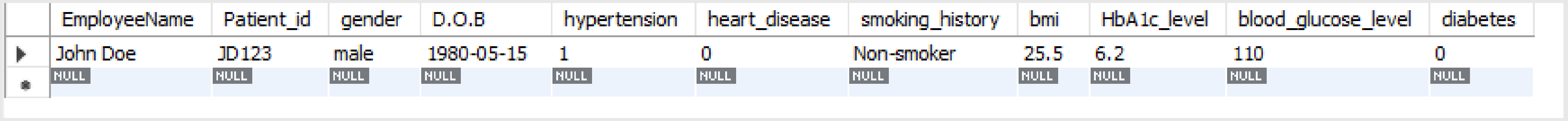
EmployeeName	Patient_id	gender	D.O.B	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
John Doe	JD123	male	1980-05-15	1	0	Ex-smoker	25.5	6.2	110	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Question |SQL Script |Database Visual


Question 13 & Solution Script

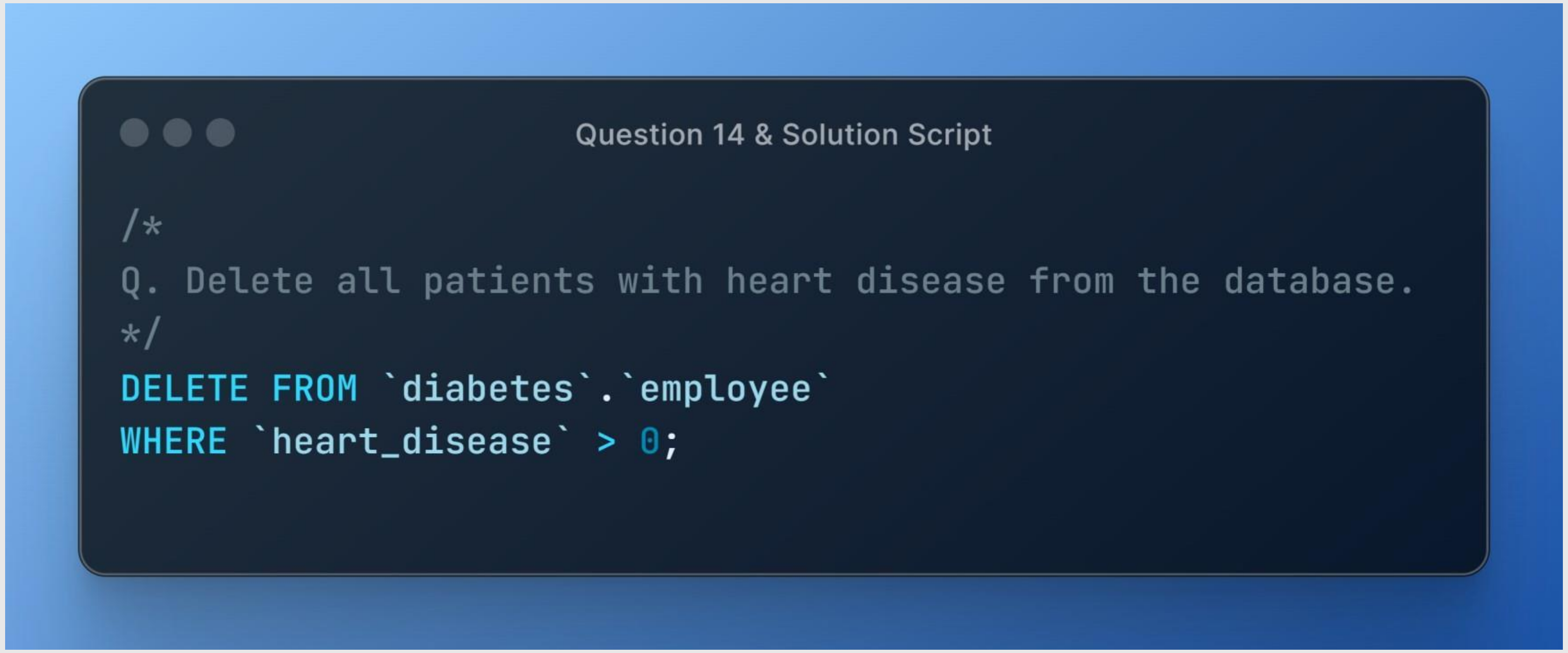
```sql
/*
 Q. Insert a new patient into the database with sample data.
*/
-- Inserting a new patient with sample data
INSERT INTO `diabetes`.`employee` (
    `EmployeeName`,
    `Patient_id`,
    `gender`,
    `D.O.B`,
    `hypertension`,
    `heart_disease`,
    `smoking_history`,
    `bmi`,
    `HbA1c_level`,
    `blood_glucose_level`,
    `diabetes`
) VALUES (
    'John Doe',
    'JD123',
    'male',
    '1980-05-15',
    1,
    0,
    'Non-smoker',
    25.5,
    6.2,
    110,
    0
);

-- Displaying the inserted record
SELECT *
FROM `diabetes`.`employee`
WHERE `Patient_id` = 'JD123';
```



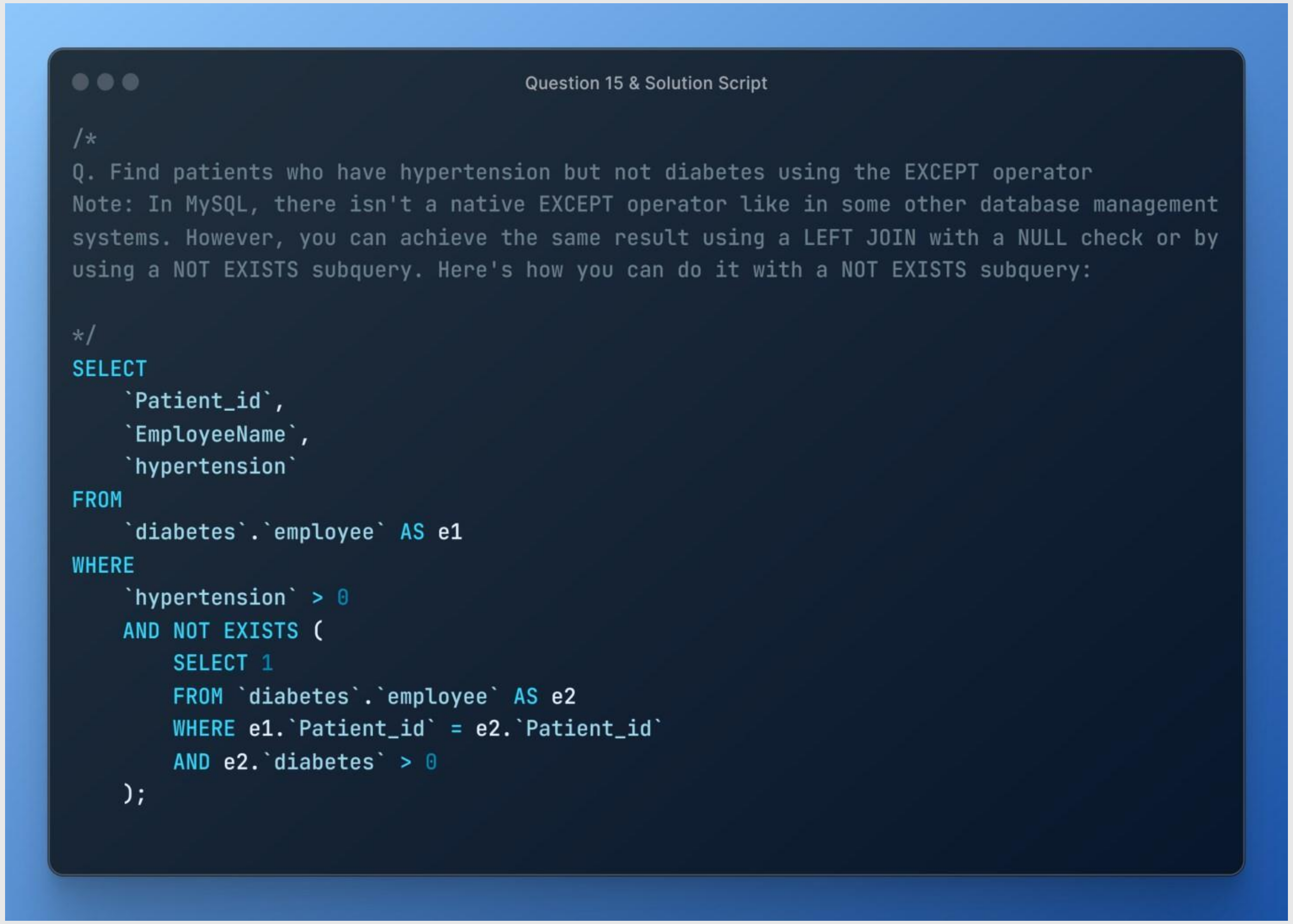
	EmployeeName	Patient_id	gender	D.O.B	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
▶	John Doe	JD123	male	1980-05-15	1	0	Non-smoker	25.5	6.2	110	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Question |SQL Script |Database Visual


Question 14 & Solution Script

```
/*
Q. Delete all patients with heart disease from the database.
*/
DELETE FROM `diabetes`.`employee`
WHERE `heart_disease` > 0;
```

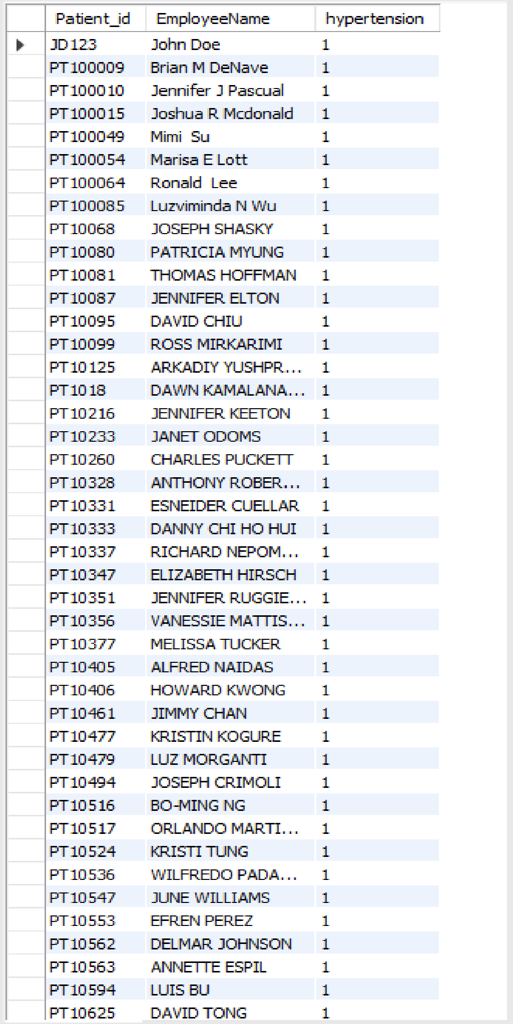
Question |SQL Script |Database Visual



Question 15 & Solution Script

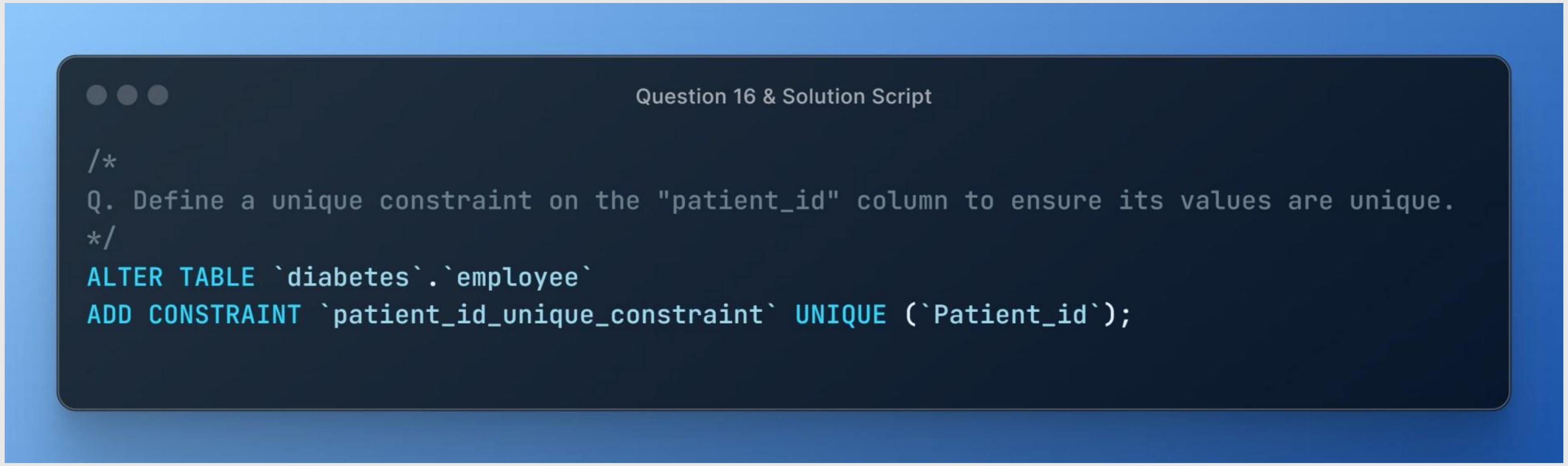
```
/*
Q. Find patients who have hypertension but not diabetes using the EXCEPT operator
Note: In MySQL, there isn't a native EXCEPT operator like in some other database management
systems. However, you can achieve the same result using a LEFT JOIN with a NULL check or by
using a NOT EXISTS subquery. Here's how you can do it with a NOT EXISTS subquery:

*/
SELECT
    `Patient_id`,
    `EmployeeName`,
    `hypertension`
FROM
    `diabetes`.`employee` AS e1
WHERE
    `hypertension` > 0
    AND NOT EXISTS (
        SELECT 1
        FROM `diabetes`.`employee` AS e2
        WHERE e1.`Patient_id` = e2.`Patient_id`
        AND e2.`diabetes` > 0
    );
```



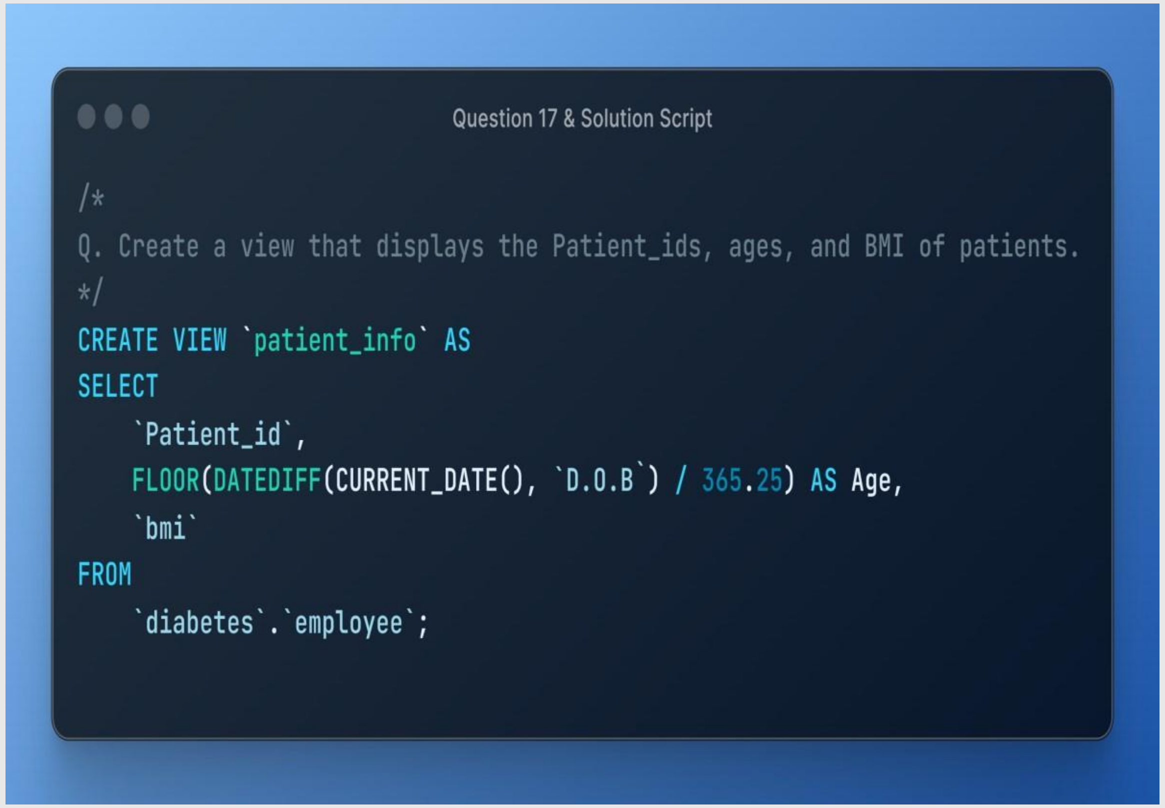
Patient_id	EmployeeName	hypertension
JD123	John Doe	1
PT100009	Brian M DeNave	1
PT100010	Jennifer J Pascual	1
PT100015	Joshua R Mcdonald	1
PT100049	Mimi Su	1
PT100054	Marisa E Lott	1
PT100064	Ronald Lee	1
PT100085	Luzviminda N Wu	1
PT10068	JOSEPH SHASKY	1
PT10080	PATRICIA MYUNG	1
PT10081	THOMAS HOFFMAN	1
PT10087	JENNIFER ELTON	1
PT10095	DAVID CHIU	1
PT10099	ROSS MIRKARIMI	1
PT10125	ARKADIY YUSHPR...	1
PT1018	DAWN KAMALANA...	1
PT10216	JENNIFER KEETON	1
PT10233	JANET ODOMS	1
PT10260	CHARLES PUCKETT	1
PT10328	ANTHONY ROBER...	1
PT10331	ESNEIDER CUELLAR	1
PT10333	DANNY CHI HO HUI	1
PT10337	RICHARD NEPOM...	1
PT10347	ELIZABETH HIRSCH	1
PT10351	JENNIFER RUGGIE...	1
PT10356	VANESSIE MATTIS...	1
PT10377	MELISSA TUCKER	1
PT10405	ALFRED NAIDAS	1
PT10406	HOWARD KWONG	1
PT10461	JIMMY CHAN	1
PT10477	KRISTIN KOGURE	1
PT10479	LUZ MORGANTI	1
PT10494	JOSEPH CRIMOLI	1
PT10516	BO-MING NG	1
PT10517	ORLANDO MARTI...	1
PT10524	KRISTI TUNG	1
PT10536	WILFREDO PADA...	1
PT10547	JUNE WILLIAMS	1
PT10553	EFREN PEREZ	1
PT10562	DELMAR JOHNSON	1
PT10563	ANNETTE ESPIL	1
PT10594	LUIS BU	1
PT10625	DAVID TONG	1

Question |SQL Script |Database Visual

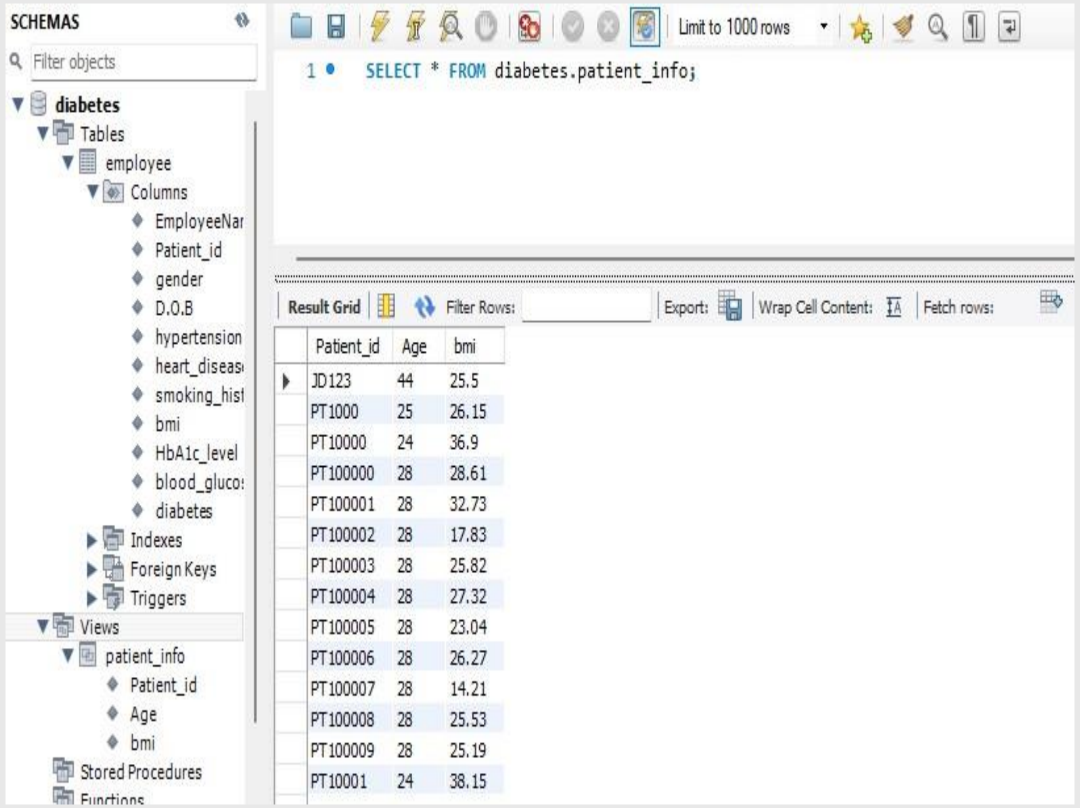

Question 16 & Solution Script

```
/*
Q. Define a unique constraint on the "patient_id" column to ensure its values are unique.
*/
ALTER TABLE `diabetes`.`employee`
ADD CONSTRAINT `patient_id_unique_constraint` UNIQUE (`Patient_id`);
```

Question |SQL Script |Database Visual


Question 17 & Solution Script

```
/*
Q. Create a view that displays the Patient_ids, ages, and BMI of patients.
*/
CREATE VIEW `patient_info` AS
SELECT
    `Patient_id`,
    FLOOR(DATEDIFF(CURRENT_DATE(), `D.O.B`) / 365.25) AS Age,
    `bmi`
FROM
    `diabetes`.`employee`;
```



```
SELECT * FROM diabetes.patient_info;
```

Patient_id	Age	bmi
JD123	44	25.5
PT1000	25	26.15
PT10000	24	36.9
PT100000	28	28.61
PT100001	28	32.73
PT100002	28	17.83
PT100003	28	25.82
PT100004	28	27.32
PT100005	28	23.04
PT100006	28	26.27
PT100007	28	14.21
PT100008	28	25.53
PT100009	28	25.19
PT10001	24	38.15

Suggestions

Question 18: Database Schema Improvement Recommendations

To enhance data integrity and minimize redundancy in the database schema, implementing normalization is crucial. By breaking down tables into smaller, related ones, redundant data storage can be eliminated. For instance, separating patient demographics from health data allows for better organization and reduces duplication. Utilizing foreign keys to establish relationships between tables further ensures data consistency. Consistent data types and constraints, such as NOT NULL and UNIQUE, enforce data integrity. Avoiding denormalization unless necessary preserves the benefits of normalization and reduces the risk of anomalies. Views provide a unified interface for complex queries without duplicating data, enhancing schema simplicity. Regular maintenance tasks, like index optimization and data cleanup, are essential for long-term schema effectiveness.

Question 1G: Optimization Strategies for SQL Queries

Optimizing SQL queries involves various techniques to enhance performance. Indexing key columns improves data retrieval speed, while query optimization, including efficient JOINs and WHERE clauses, enhances query efficiency. Normalization minimizes redundancy, improving data integrity and query performance. Denormalization selectively improves performance for read-heavy data while maintaining benefits of normalization. Partitioning large tables reduces scan time by organizing data into manageable chunks. Query caching stores frequently accessed results, reducing query execution time. Optimal data types and regular maintenance tasks like index rebuilding ensure efficient storage and operation. Connection pooling reduces overhead by reusing database connections, improving query response time. Scaling options, both horizontally and vertically, cater to increasing data demands. Batch processing efficiently handles bulk data operations, enhancing overall throughput and query performance. These strategies collectively optimize SQL query performance on datasets, ensuring efficient database operation.