SQL PROJECT ON

HEALTH CARE ANALYTICS

BY DEEPTI CHAND

ABOUT PROJECT

PROJECT OVERVIEW

The Healthcare Analytics System is designed to manage and analyze data within a healthcare facility, providing a comprehensive view of patient care, doctor performance, billing, and staff management. This system integrates various aspects of healthcare data to facilitate better decision-making, streamline operations, and improve patient outcomes.

TABLES AND THEIR FUNCTIONALITIES

Patients Table

Stores information about patients including their personal details, contact information, and status.

Doctors Table

Contains details about doctors, their specialties, and contact information.

Appointments Table

Tracks appointments between patients and doctors, including appointment dates and statuses.

Medical_Records Table

Records medical information about patients, including diagnoses, treatments, and the date of the record.

Prescriptions Table

Maintains details about prescriptions issued to patients, including medication details and dosage.

Billing Table

Manages billing information for patients, tracking amounts, billing dates, and payment statuses.

Departments Table

Organizes information about different departments in the healthcare facility and their head doctors.

Hospital_Staff Table

Contains details about the hospital staff, their positions, and the departments they belong to.

FUNCTIONALITY OVERVIEW

PATIENT MANAGEMENT

Track and manage patient details including personal information, contact details, and status.

DOCTOR MANAGEMENT

Maintain records of doctors, their specialties, and contact information.

APPOINTMENTS SCHEDULING AND TRACKING

Schedule, track, and update appointment details.

MEDICAL RECORDS MAINTENANCE

Document diagnoses, treatments, and other medical records for patients.

PRESCRIPTION MANAGEMENT

Manage and track prescriptions issued to patients, including medication details and dosage instructions.

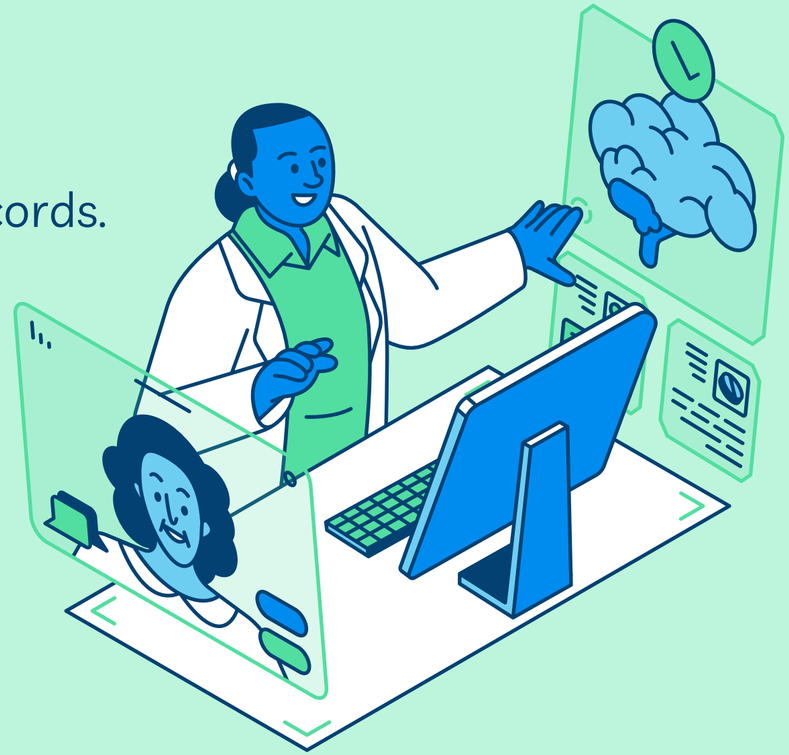
BILLING AND PAYMENT TRACKING

Handle billing processes, track payment statuses, and manage financial records.

DEPARTMENT AND STAFF MANAGEMENT

Organize and manage information about various departments and their head doctors.

Track hospital staff details, their positions, and departmental associations.

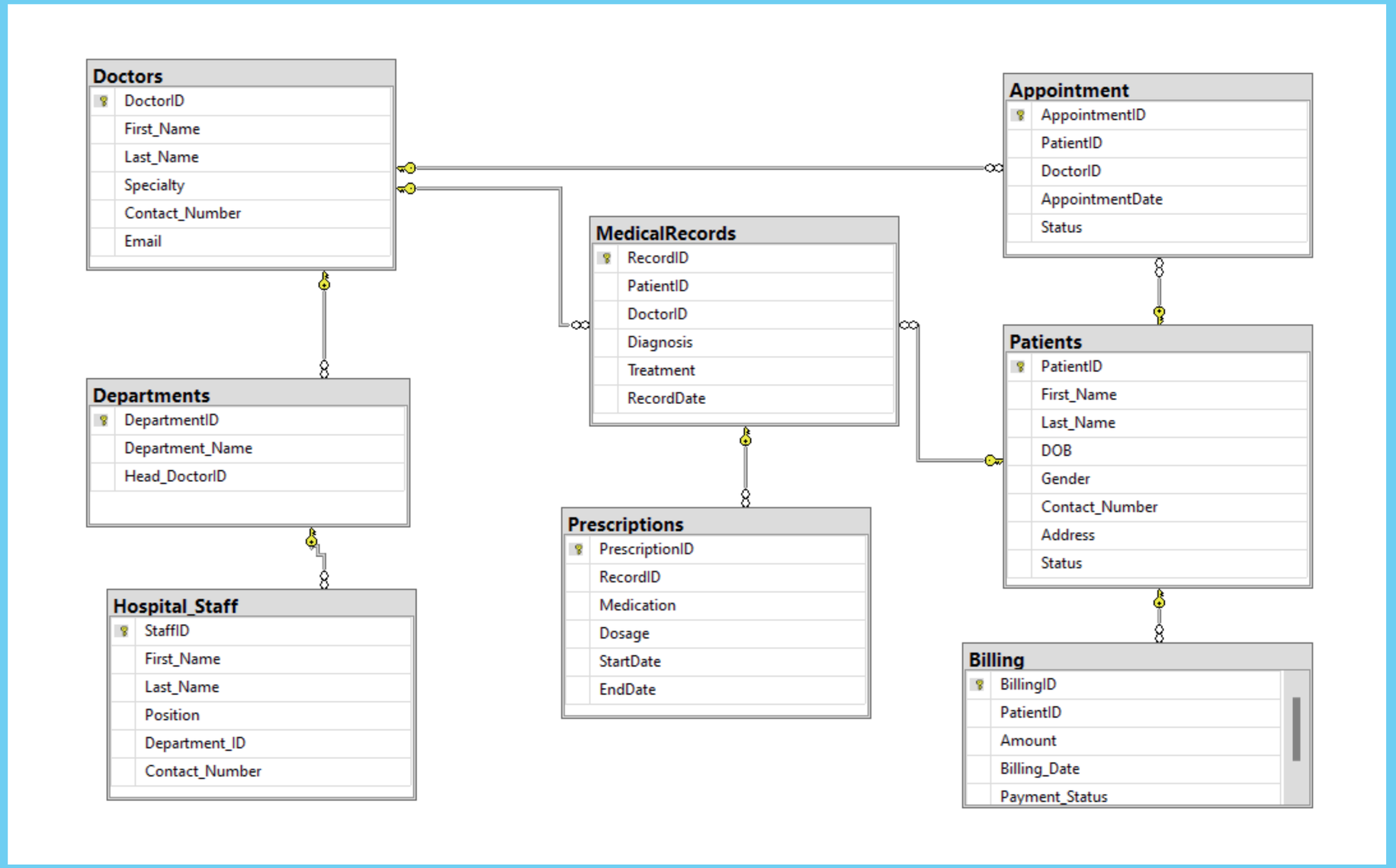


Conclusion

The Healthcare Analytics System provides a robust framework for managing various aspects of healthcare data, enabling healthcare providers to deliver better patient care and streamline their operations.

By integrating patient management, doctor performance tracking, appointment scheduling, medical record maintenance, prescription management, billing, and staff organization, this system offers a comprehensive solution for healthcare analytics.

DATABASE SCHEMA



IMPORTANT COMMANDS IN SQL

With Examples

SELECT Statement

Used to retrieve data from one or more tables.

Query

```
Select * From Doctors
Select Specialty from Doctors

-------------------------------
```

Output:

Results | Messages

	DoctorID	First_Name	Last_Name	Specialty	Contact_Number	Email
1	1	Dr. Ravi	Patel	Cardiology	9876543215	ravi.patel@example.com
2	2	Dr. Deepthi	Desai	Neurology	9876543216	deepthi.desai@example.com
3	3	Dr. Anil	Joshi	Orthopedics	9876543217	anil.joshi@example.com
4	4	Dr. Meera	Rao	Pediatrics	9876543218	meera.rao@example.com
5	5	Dr. Vikram	Kapoor	Dermatology	9876543219	vikram.kapoor@example.com

	Specialty
1	Cardiology
2	Neurology
3	Orthopedics
4	Pediatrics
5	Dermatology

WHERE Clause :

Filters records based on specified conditions.

Query

```
1) SELECT * FROM Patients WHERE Gender= 'Female'
----------------------------------------


2) SELECT First_Name, Last_Name
FROM Doctors
WHERE First_Name = 'Dr. Deepthi';
```

Output

Results | Messages

	PatientID	First_Name	Last_Name	DOB	Gender	Contact_Number	Address	Status
1	2	Neha	Sharma	1990-07-23	Female	9876543211	Delhi, Delhi	Active
2	4	Priya	Mehta	1995-09-15	Female	9876543213	Chennai, Tamil Nadu	InActive
3	6	Sneha	Patel	1991-12-11	Female	9876543215	Ahmedabad, Gujarat	Active
4	8	Pooja	Rao	1993-11-30	Female	9876543217	Hyderabad, Telangana	Active
5	10	Anita	Desai	1992-10-05	Female	9876543219	Jaipur, Rajasthan	Active

	First_Name	Last_Name
1	Dr. Deepthi	Desai

ORDER BY :

Sorts the result set in ascending or descending order.

ASC FOR Ascending Order & **DESC** For Descending Order

Query

```
-----OrderBy-----
SELECT * FROM Patients ORDER BY Gender ASC;
SELECT * FROM Patients ORDER BY Gender Desc;
```

Output

Results | Messages

3	6	Sneha	Patel	1991-12-11	Female	9876543215	Ahmedabad, Gujarat	Active
4	8	Pooja	Rao	1993-11-30	Female	9876543217	Hyderabad, Telangana	Active
5	10	Anita	Desai	1992-10-05	Female	9876543219	Jaipur, Rajasthan	Active
6	1	Arjun	Verma	1985-04-12	Male	9876543210	Mumbai, Maharashtra	Active
7	9	Rahul	Nair	1988-03-22	Male	9876543218	Thiruvananthapuram, Kerala	Active
8	7	Vikram	Joshi	1984-08-25	Male	9876543216	Kolkata, West Bengal	Active
9	5	Amit	Kumar	1987-05-20	Male	9876543214	Pune, Maharashtra	Active
10	3	Rohan	Singh	1982-01-30	Male	9876543212	Bangalore, Karnataka	Active

	PatientID	First_Name	Last_Name	DOB	Gender	Contact_Number	Address	Status
1	1	Arjun	Verma	1985-04-12	Male	9876543210	Mumbai, Maharashtra	Active
2	3	Rohan	Singh	1982-01-30	Male	9876543212	Bangalore, Karnataka	Active
3	5	Amit	Kumar	1987-05-20	Male	9876543214	Pune, Maharashtra	Active
4	7	Vikram	Joshi	1984-08-25	Male	9876543216	Kolkata, West Bengal	Active
5	9	Rahul	Nair	1988-03-22	Male	9876543218	Thiruvananthapuram, Kerala	Active
6	10	Anita	Desai	1992-10-05	Female	9876543219	Jaipur, Rajasthan	Active
7	8	Pooja	Rao	1993-11-30	Female	9876543217	Hyderabad, Telangana	Active
8	6	Sneha	Patel	1991-12-11	Female	9876543215	Ahmedabad, Gujarat	Active

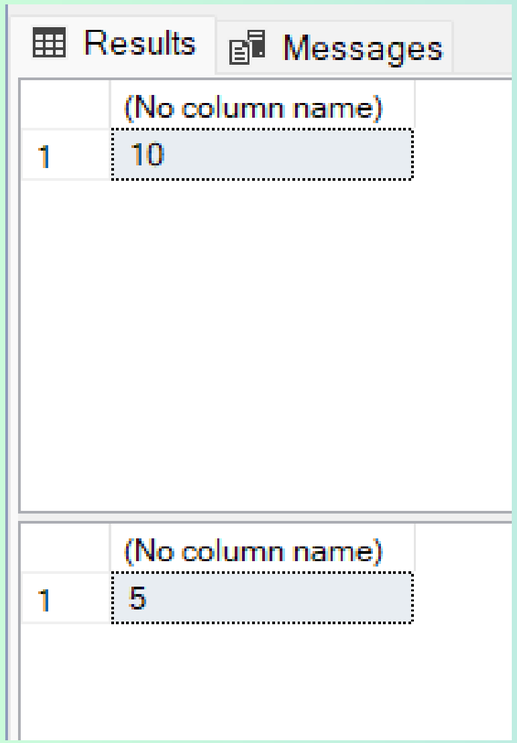
COUNT

Returns the number of rows that match a specified condition

Query

```
-------------Count-----------------
SELECT COUNT(*) FROM Appointment ;
SELECT COUNT(*) FROM Patients where Gender = 'Female';
```

Output



AGGREGATE FUNCTIONS: SUM, AVG, MIN, MAX

Sum(): Returns the sum of a numeric column.

Query

```
SElECT * FROM Billing;
SELECT SUM(Amount) as TotalBilling FROM Billing;
--------------------
```

Output

Results | Messages

	BillingID	PatientID	Amount	Billing_Date	Payment_Status
1	1	1	1500.00	2024-07-02	Paid
2	2	2	2000.00	2024-07-06	Paid
3	3	3	2000.00	2024-07-11	Pending
4	4	4	500.00	2024-07-16	Paid
5	5	5	800.00	2024-07-21	Paid
6	6	6	1800.00	2024-08-02	Pending
7	7	7	2200.00	2024-08-07	Paid

	TotalBilling
1	10800.00

Avg():Returns the average value of a numeric column.

Min():Returns the minimum value of a column.

Max():Returns the maximum value of a column.

Query

```
SElECT * FROM Billing;
SELECT AVG(Amount) as AvgBilling FROM Billing;
SELECT MAX(Amount) as HighestBilling FROM Billing;
SELECT MIN(Amount) as LowestBilling FROM Billing;
-----------------------------
```

Output

Results | Messages

	BillingID	PatientID	Amount	Billing_Date	Payment_
1	1	1	1500.00	2024-07-02	Paid
2	2	2	2000.00	2024-07-06	Paid
3	3	3	2000.00	2024-07-11	Pending
4	4	4	500.00	2024-07-16	Paid
5	5	5	800.00	2024-07-21	Paid
6	6	6	1800.00	2024-08-02	Pending
7	7	7	2200.00	2024-08-07	Paid

	AvgBilling
1	1542.857142

	HighestBilling
1	2200.00

	LowestBilling
1	500.00

Subqueries

A subquery, also known as an inner query or nested query, is a query within another SQL query.

The subquery is executed first, and its result is used by the outer query.

A subquery can be used in **SELECT, INSERT, UPDATE, or DELETE** statements or inside another subquery.

Question : Retrieve doctors who have appointments scheduled after '2023-01-01'

Query

```
Select  First_Name Last_Name From Doctors
WHERE DoctorID IN (SELECT DISTINCT DoctorID FROM Appointment WHERE AppointmentDate > '2023-01-01');
```

Output

Results | Messages

	Last_Name
1	Dr. Ravi
2	Dr. Deepthi
3	Dr. Anil
4	Dr. Meera
5	Dr. Vikram

GROUP BY:

Groups rows that have the same values in specified columns into summary rows.

--Question: Count the number of patients by gender.

Query

```
----------------------
SELECT Gender, COUNT(*) AS NumberOfPatients
FROM Patients
GROUP BY Gender;
--------------------------------
```

Output

Results | Messages

	Gender	NumberOfPatients
1	Female	5
2	Male	5

HAVING :

Filters records that are grouped by the GROUP BY clause.

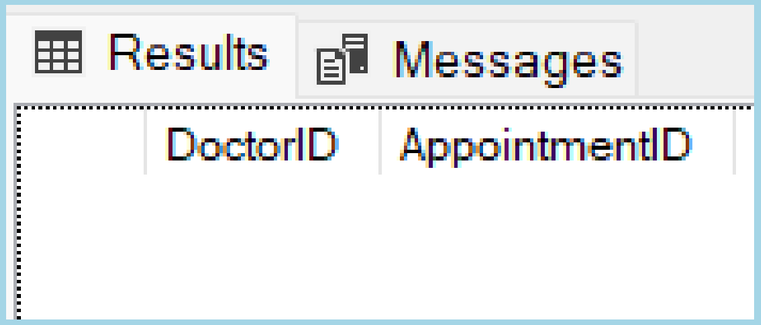
.

--Question: Retrieve doctors who have more than 10 appointments.

Query

```
SELECT DoctorID, Count(*) As AppointmentID
FROM Appointment
GROUP BY DoctorID
HAVING COUNT(*)>10
```

Output



Note :

This means that no Doctor had more than 10 appointments , or the conditions specified in the query did not match any records in the database.

JOINS

Although SQL has many types of joins, we'll focus only on the most commonly used in this project.

INNER JOIN: Combines rows from two or more tables based on a related column between them.

--Question: Retrieve patients' names along with their doctors' names for all appointments--

Query

```
SELECT P.First_Name AS PatientFirstName,
       P.Last_Name As PatientLastName,
       D.First_Name As DoctorFirstName,
       D.Last_Name As DoctorLastName,
       A.AppointmentDate
       From Appointment A
       Inner Join Patients P
       On A.PatientID = P.PatientID
       Inner Join Doctors D
       On A.PatientID = D.DoctorID;
```

Output

Results | Messages

	PatientFirstName	PatientLastName	DoctorFirstName	DoctorLastName	AppointmentDate
1	Arjun	Verma	Dr. Ravi	Patel	2024-07-01 10:00:00.000
2	Neha	Sharma	Dr. Deepthi	Desai	2024-07-05 11:00:00.000
3	Rohan	Singh	Dr. Anil	Joshi	2024-07-10 12:00:00.000
4	Priya	Mehta	Dr. Meera	Rao	2024-07-15 09:00:00.000
5	Amit	Kumar	Dr. Vikram	Kapoor	2024-07-20 14:00:00.000

LEFT JOIN:

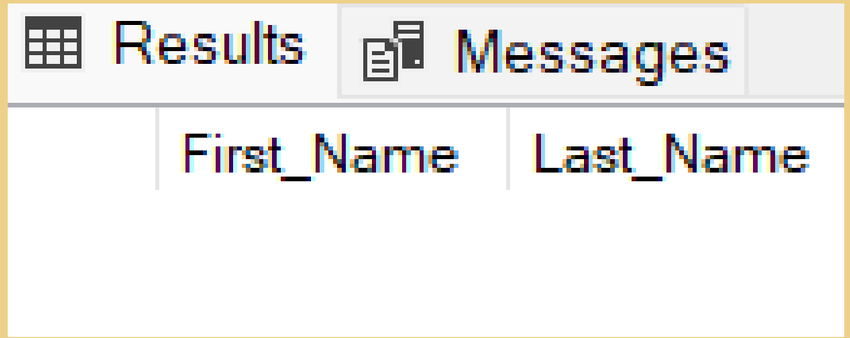
Returns all rows from the left table and the matched rows from the right table.
If no match, **NULL** values are returned for columns from the right table.

--Question: Retrieve patients who have never had an appointment. names for all appointments--.

Query

```
SELECT P.First_Name,P.Last_Name
From Patients P
Left Join Appointment A
ON
P.PatientID = A.PatientID
WHERE A.AppointmentDate is NULL
```

Output



Note : This means that there are no patients who never had an appointments , or the conditions specified in the query did not match any records in the database.

Union / Union All Operator

UNION :

- Removes duplicate rows from the final result set.
- Generally slower due to the duplicate removal process.
- Use when you need a distinct list of results from multiple queries.

UNION ALL :

- Includes all rows, with duplicates preserved.
- Faster because it doesn’t check for duplicates.
- Use when you need all rows from multiple queries, including duplicates, and performance is a concern.

--Question: Retrieve names from both the Doctors and Patients tables----

Query

```
SELECT First_Name, Last_Name FROM Doctors
UNION
SELECT First_Name, Last_Name FROM Patients
```

Output

Results | Messages

	First_Name	Last_Name
1	Amit	Kumar
2	Anita	Desai
3	Arjun	Verma
4	Dr. Anil	Joshi
5	Dr. Deepthi	Desai
6	Dr. Meera	Rao
7	Dr. Ravi	Patel
8	Dr. Vikram	Kapoor
9	Neha	Sharma
10	Pooja	Rao
11	Priya	Mehta
12	Rahul	Nair
13	Rohan	Singh
14	Sneha	Patel
15	Vikram	Joshi

WINDOW FUNCTIONS

ROW_NUMBER():

Assigns a unique sequential integer to each row.

RANK():

Assigns a rank to each row with possible gaps in ranks for tied rows.
Ranking rows in a result set based on a specific column with consideration of ties.

DENSE_RANK():

Assigns a rank to each row without gaps for tied rows.
Ranking rows in a result set with continuous ranks even for ties.

--Question: Assign row numbers to billing records ordered by amount.

Query

```
--------------ROW_NUMBER------------
SELECT Amount, ROW_NUMBER() OVER (ORDER BY Amount Desc) AS "Row_Num" FROM Billing
--------RANK---------
SELECT Amount, RANK() OVER (ORDER BY Amount Desc) AS "Row_Num" FROM Billing
--------------------DENSE_RANK-----------------
SELECT Amount, DENSE_RANK() OVER (ORDER BY Amount Desc) AS "Row_Num" FROM Billing
-------------------------------------------------------------
```

Output

Results | Messages

	Amount	Row_Num
1	2200.00	1
2	2000.00	2
3	2000.00	3
4	1800.00	4
5	1500.00	5
6	800.00	6
7	500.00	7

Results | Messages

	Amount	RANK
1	2200.00	1
2	2000.00	2
3	2000.00	2
4	1800.00	4
5	1500.00	5
6	800.00	6
7	500.00	7

Results | Messages

	Amount	DENSE_RANK
1	2200.00	1
2	2000.00	2
3	2000.00	2
4	1800.00	3
5	1500.00	4
6	800.00	5
7	500.00	6

Common Table Expressions (CTE)

- A basic CTE that defines a temporary result set.
  - Allows for easier handling of complex queries with multiple joins or aggregations.
  - The result set of a CTE can be referenced multiple times within the main query.
- Hierarchical Data Handling
- Improved Readability

--Question: Retrieve doctors and the number of patients they have seen.

Query

```
WITH PatientCounts AS (
    SELECT DoctorID, COUNT(*) AS NumberOfPatients
    FROM
    Appointment
    GROUP BY
    DoctorID
)
SELECT Concat(D.First_Name,' ',D.Last_Name) as DoctorName, P.NumberOfPatients
FROM Doctors D
JOIN PatientCounts P ON D.DoctorID = P.DoctorID;
```

Results | Messages

	DoctorName	NumberOfPatients
1	Dr. Ravi Patel	3
2	Dr. Deepthi Desai	2
3	Dr. Anil Joshi	3
4	Dr. Meera Rao	1
5	Dr. Vikram Kapoor	1

Correlated Subquery

A correlated subquery is a subquery that references columns from the outer query. It is executed once for each row processed by the outer query and uses values from the current row of the outer query to perform its operation.

----Question : Retrieve patients with pending appointments ---

Query

```
SELECT P.First_Name, P.Last_Name
FROM Patients P
WHERE EXISTS (SELECT 1 FROM Appointment A WHERE A.PatientID = P.PatientID AND A.Status = 'Pending');
----------------------------
```

Output

Results | Messages

	First_Name	Last_Name
1	Pooja	Rao
2	Rahul	Nair
3	Anita	Desai

THANK YOU!

For reviewing this SQL Project in Advance.

Your support and insights are greatly appreciated!