HEALTHCARE ANALYTICS SQL PROJECT REPORT

PROJECT OVERVIEW

This SQL project focuses on analyzing healthcare datasets to understand patient activity, doctor performance, appointment completion, diagnoses, and medication usage. Using SQL joins, window functions, subqueries, and aggregations, I explored how hospitals can derive insights from structured data to improve operations and patient care.

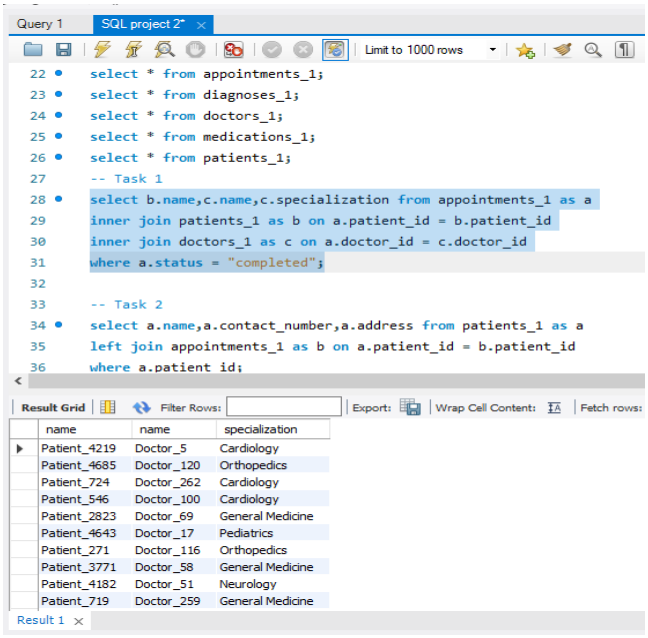
PROBLEM STATEMENT

The objective of this project is to use SQL to analyze core healthcare operations. The tasks include tracking appointment status, identifying inactive patients, measuring doctor workload, detecting mismatched medical records, categorizing patients, evaluating medication duration, and identifying diagnosis patterns. The project aims to enhance understanding of healthcare data through SQL-based exploration

TASK EXPLAINATION

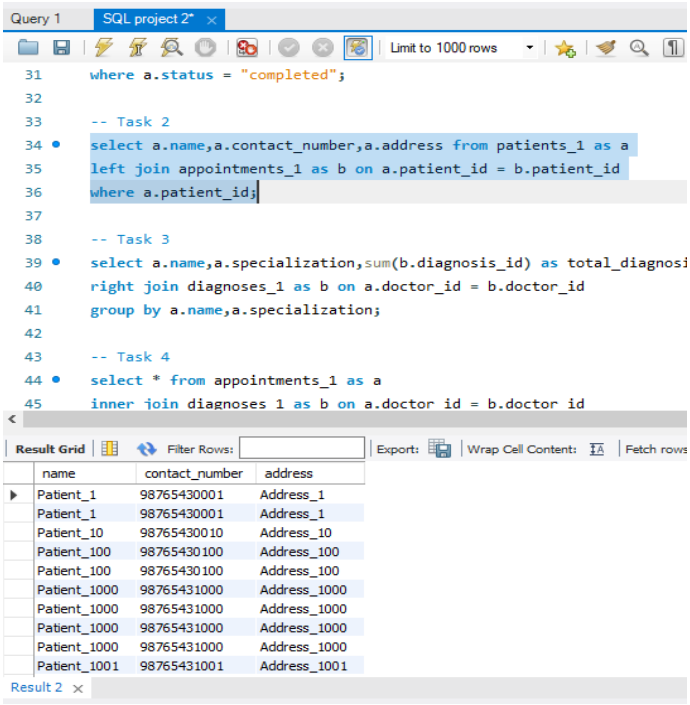
1. Completed Appointments:

Using INNER JOIN, I retrieved the details of all completed appointments with patient and doctor information. This helps understand successful appointment patterns and doctor involvement.



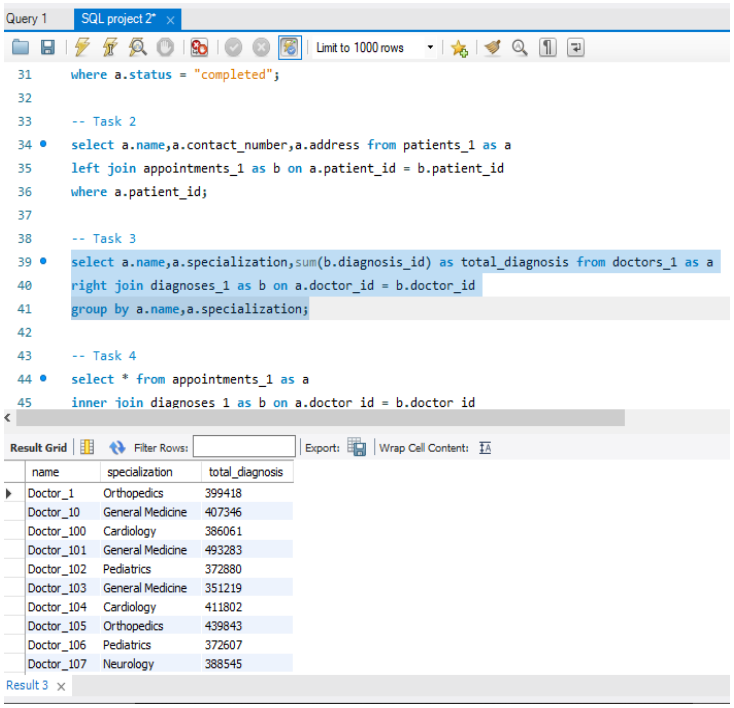
2. Patients Without Appointments:

Using LEFT JOIN, I extracted patients who never booked any appointment. This helps identify inactive patients and improves patient outreach strategies.



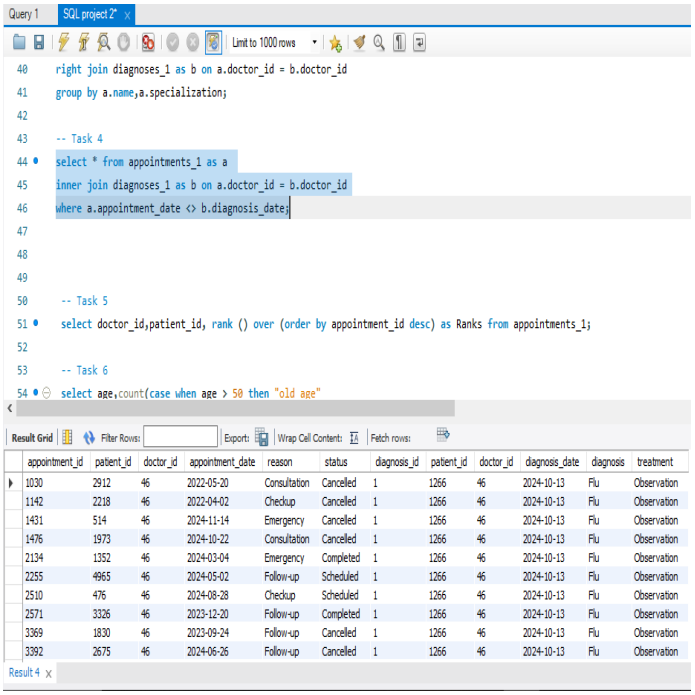
3. Diagnoses Count per Doctor:

Using RIGHT JOIN and COUNT(), I calculated the total diagnoses for each doctor. This shows doctor efficiency and workload.



4. Appointment–Diagnosis Mismatch:

I compared appointment and diagnosis dates to identify mismatches. This helps detect data inconsistency and improves record accuracy.

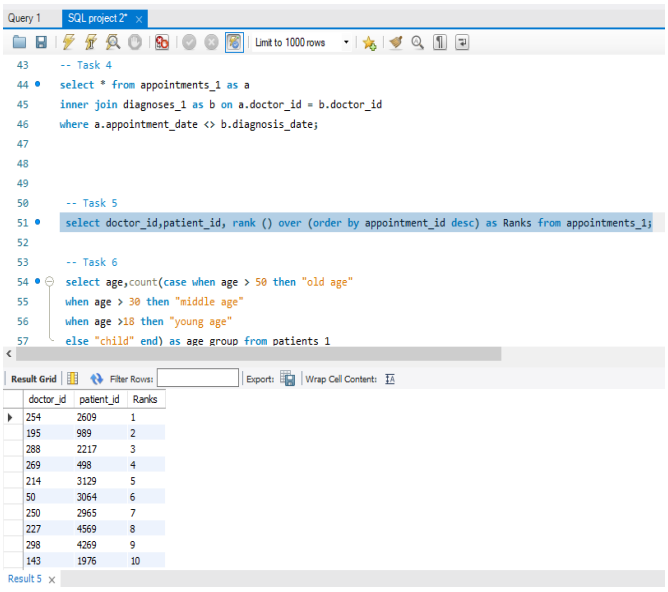


```
right join diagnoses_1 as b on a.doctor_id = b.doctor_id
group by a.name,a.specialization;

-- Task 4
select * from appointments_1 as a
inner join diagnoses_1 as b on a.doctor_id = b.doctor_id
where a.appointment_date <> b.diagnosis_date;


-- Task 5
select doctor_id,patient_id, rank () over (order by appointment_id desc) as Ranks from appointments_1;

-- Task 6
select age,count(case when age > 50 then "old age"
```

appointment_id	patient_id	doctor_id	appointment_date	reason	status	diagnosis_id	patient_id	doctor_id	diagnosis_date	diagnosis	treatment
1030	2912	46	2022-05-20	Consultation	Cancelled	1	1266	46	2024-10-13	Flu	Observation
1142	2218	46	2022-04-02	Checkup	Cancelled	1	1266	46	2024-10-13	Flu	Observation
1431	514	46	2024-11-14	Emergency	Cancelled	1	1266	46	2024-10-13	Flu	Observation
1476	1973	46	2024-10-22	Consultation	Cancelled	1	1266	46	2024-10-13	Flu	Observation
2134	1352	46	2024-03-04	Emergency	Completed	1	1266	46	2024-10-13	Flu	Observation
2255	4965	46	2024-05-02	Follow-up	Scheduled	1	1266	46	2024-10-13	Flu	Observation
2510	476	46	2024-08-28	Checkup	Scheduled	1	1266	46	2024-10-13	Flu	Observation
2571	3326	46	2023-12-20	Follow-up	Completed	1	1266	46	2024-10-13	Flu	Observation
3369	1830	46	2023-09-24	Follow-up	Cancelled	1	1266	46	2024-10-13	Flu	Observation
3392	2675	46	2024-06-26	Follow-up	Cancelled	1	1266	46	2024-10-13	Flu	Observation

5. Patient Ranking by Doctor:

Using RANK() window function, I ranked patients based on appointment frequency. This helps identify highly active patients.

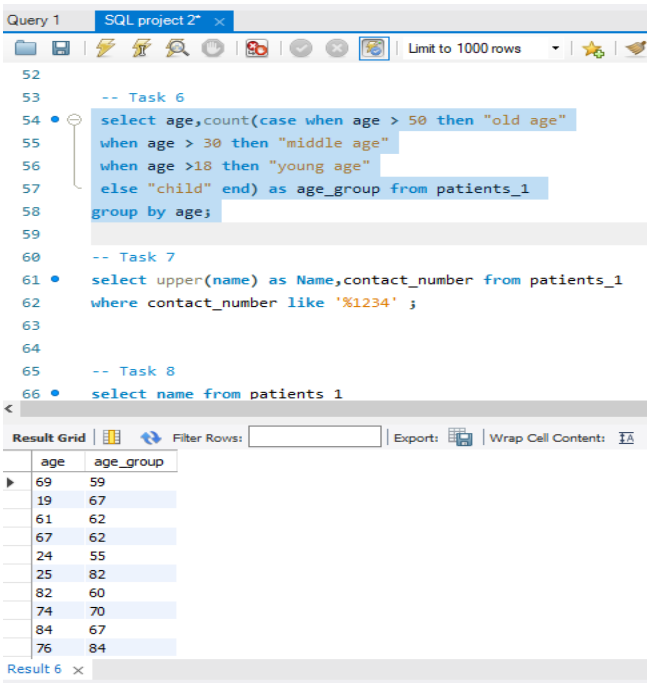


```
-- Task 4
select * from appointments_1 as a
inner join diagnoses_1 as b on a.doctor_id = b.doctor_id
where a.appointment_date <> b.diagnosis_date;


-- Task 5
select doctor_id,patient_id, rank () over (order by appointment_id desc) as Ranks from appointments_1;

-- Task 6
select age,count(case when age > 50 then "old age"
when age > 30 then "middle age"
when age >18 then "young age"
else "child" end) as age_group from patients 1
```

doctor_id	patient_id	Ranks
254	2609	1
195	989	2
288	2217	3
269	498	4
214	3129	5
50	3064	6
250	2965	7
227	4569	8
298	4269	9
143	1976	10

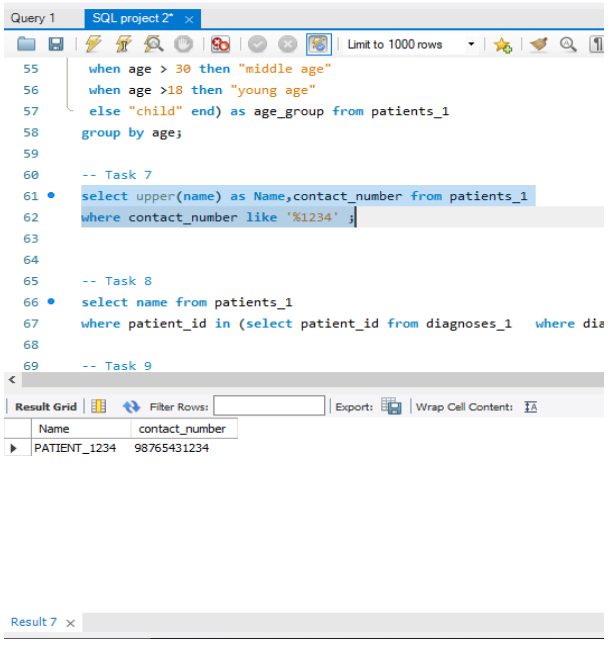
6. Patient Age Group Categorization:

Using CASE, I categorized patients into age groups such as child, young, middle-age, and old-age. This supports demographic analysis.



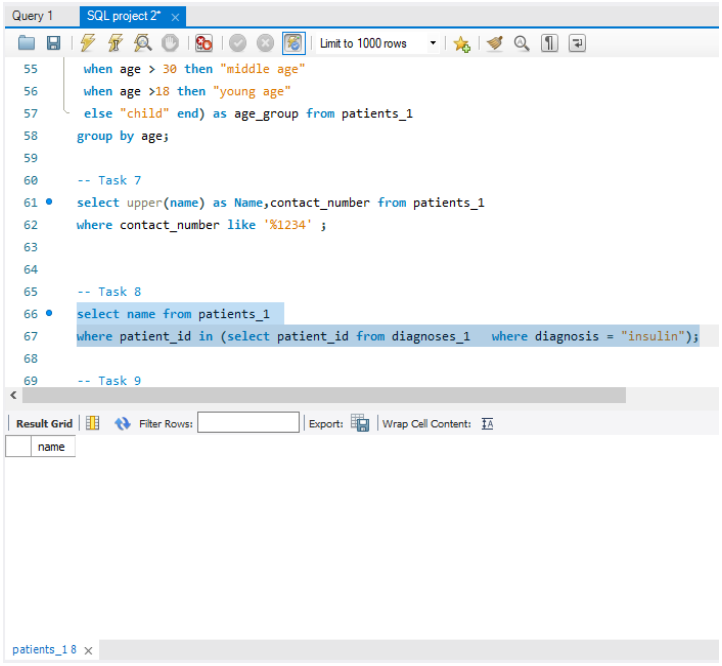
7. Contact Number Matching:

Using LIKE and UPPER(), I retrieved patients whose contact numbers end with '1234'. This shows usage of string filters.



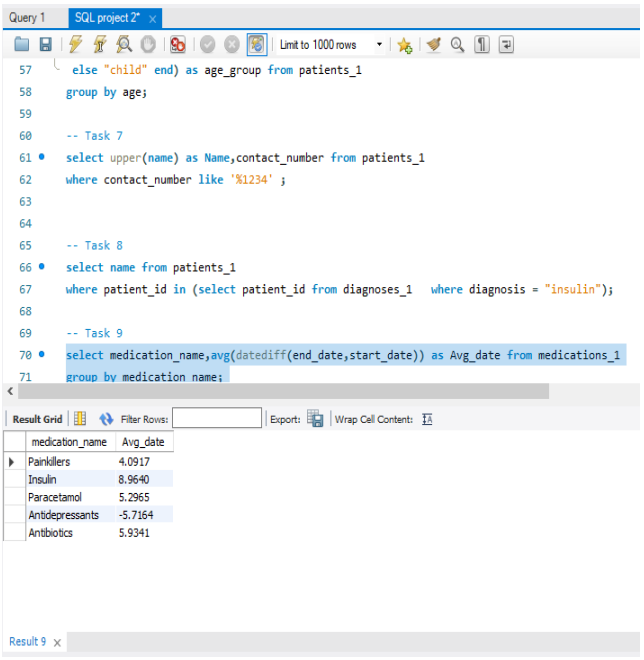
8. Patients Prescribed Insulin:

Using a subquery, I identified patients diagnosed with Insulin-only prescriptions. This helps identify diabetic patients.



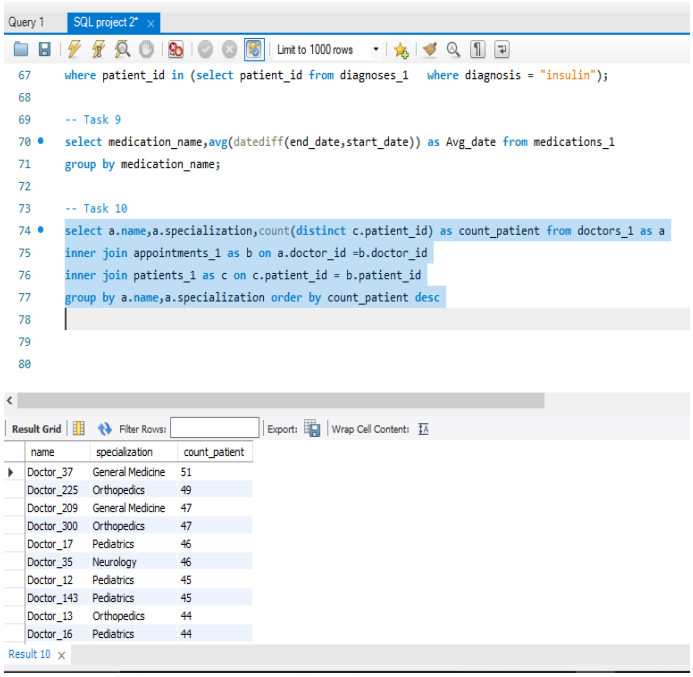
9. Average Medication Duration:

I used DATEDIFF() to calculate the average days each medication was prescribed. This helps analyze treatment patterns.



10. Doctor With Most Unique Patients:

Using COUNT(DISTINCT) with joins, I found the doctor who attended the highest number of unique patients. This identifies top-performing doctors.



CONCLUSION

This healthcare SQL project provided a strong understanding of data relationships in hospital systems. The tasks helped identify doctor performance, appointment patterns, patient demographics, medication usage, and data inaccuracies. Through this project, I strengthened my SQL skills and gained the ability to perform real-world operational analysis using structured healthcare data.