Healthcare Analytics with SQL

Skills take away From This Project:

- Proficiency in SQL Joins (Inner, Left, Right, Full Outer).
- Advanced usage of Aggregate Functions (e.g., COUNT, SUM).
- Expertise in Window Functions (e.g., RANK, DENSE_RANK).
- Working with Subqueries and Conditional Logic.
- Understanding of Date and Time Functions for analysing temporal data.
- Experience with complex data analysis and reporting in a healthcare context

Problem statement:

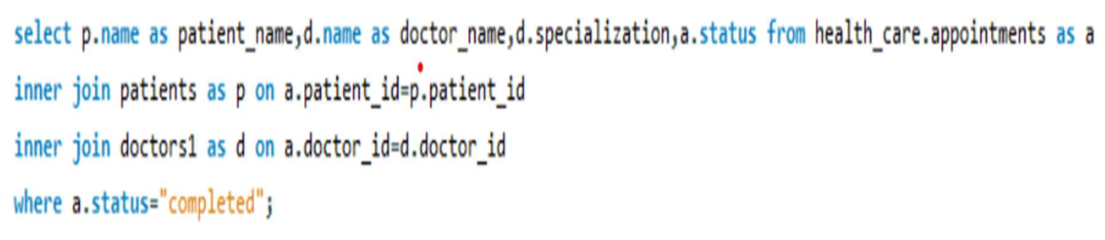
The project aims to analyse healthcare data, focusing on extracting meaningful insights about patients, doctors, appointments, diagnoses, and treatments using advanced SQL techniques. Learners will apply SQL operations like joins, subqueries, window functions, and more to analyse healthcare metrics.

Approach:

Task 1:

Inner and Equi Joins

- To Write a query to fetch details of all completed appointments, including the patient's name, doctor's name, and specialization.

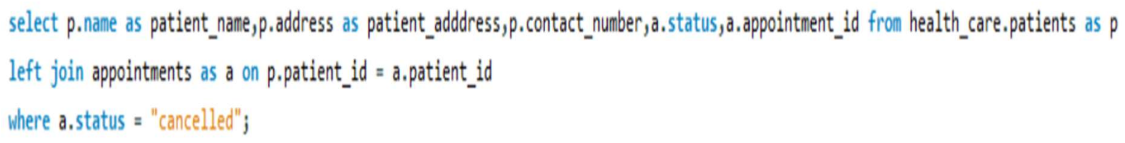


```sql
select p.name as patient_name,d.name as doctor_name,d.specialization,a.status from health_care.appointments as a
inner join patients as p on a.patient_id=p.patient_id
inner join doctors1 as d on a.doctor_id=d.doctor_id
where a.status="completed";
```

Task 2:

<u>Left Join with Null Handling</u>

- Retrieve all patients who have never had an appointment. Include their name, contact details, and address in the output.

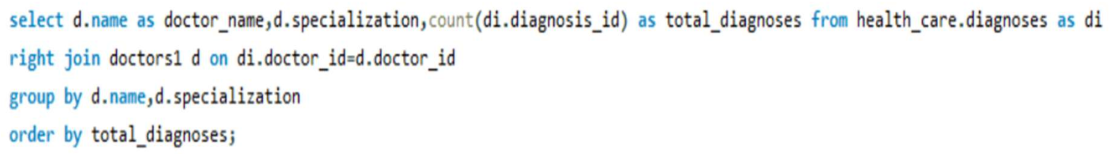


```
select p.name as patient_name,p.address as patient_adddress,p.contact_number,a.status,a.appointment_id from health_care.patients as p
left join appointments as a on p.patient_id = a.patient_id
where a.status = "cancelled";
```

Task 3:

<u>Right Join and Aggregate Functions</u>

- Find the total number of diagnoses for each doctor, including doctors who haven't diagnosed any patients. Display the doctor's name, specialization, and total diagnoses.

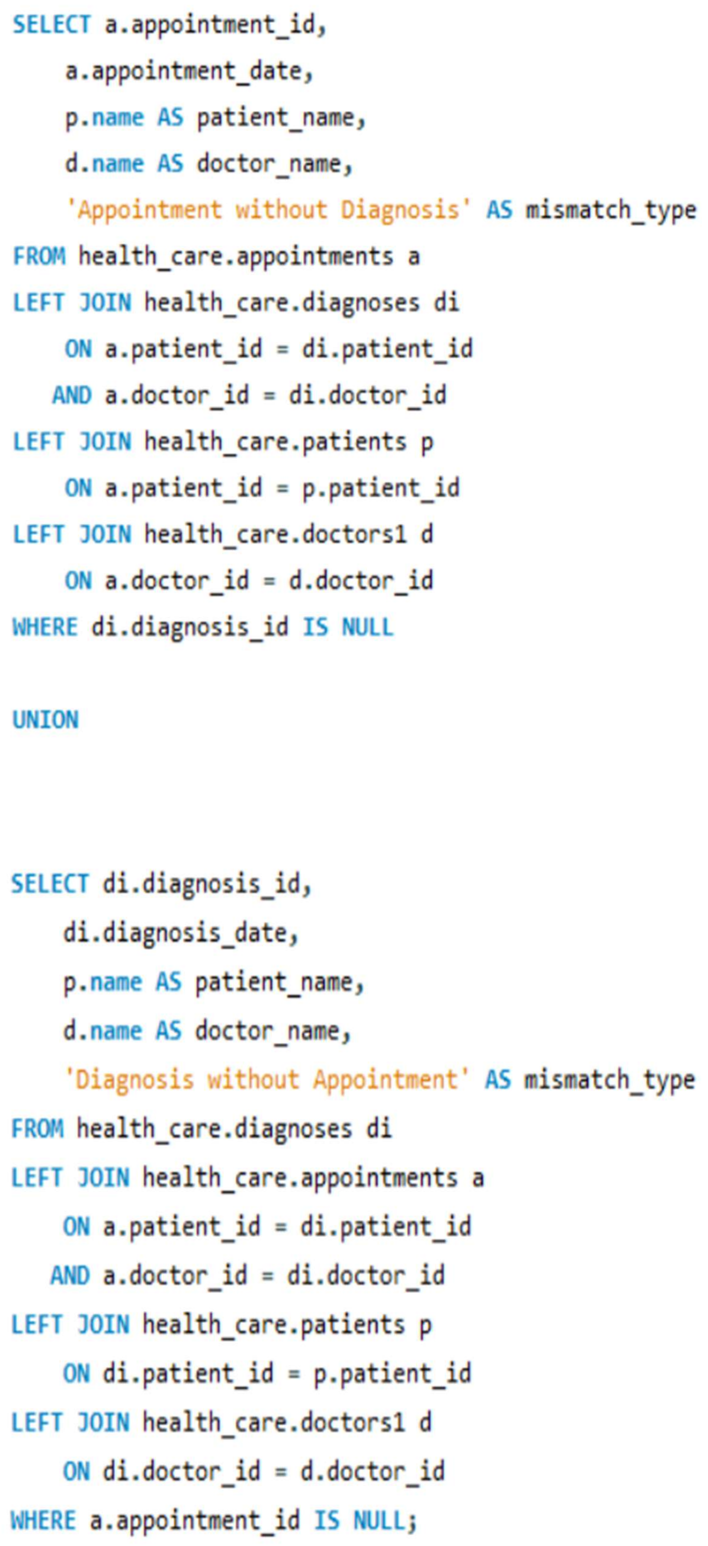


```
select d.name as doctor_name,d.specialization,count(di.diagnosis_id) as total_diagnoses from health_care.diagnoses as di
right join doctors1 d on di.doctor_id=d.doctor_id
group by d.name,d.specialization
order by total_diagnoses;
```

Task 4:

<u>Full Join for Overlapping Data</u>

- Write a query to identify mismatches between the appointments and diagnoses tables. Include all appointments and diagnoses with their corresponding patient and doctor details.

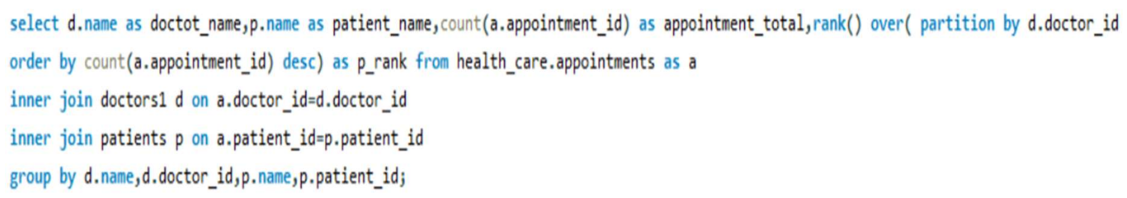


```sql
SELECT a.appointment_id,
    a.appointment_date,
    p.name AS patient_name,
    d.name AS doctor_name,
    'Appointment without Diagnosis' AS mismatch_type
FROM health_care.appointments a
LEFT JOIN health_care.diagnoses di
    ON a.patient_id = di.patient_id
   AND a.doctor_id = di.doctor_id
LEFT JOIN health_care.patients p
    ON a.patient_id = p.patient_id
LEFT JOIN health_care.doctors1 d
    ON a.doctor_id = d.doctor_id
WHERE di.diagnosis_id IS NULL

UNION



SELECT di.diagnosis_id,
    di.diagnosis_date,
    p.name AS patient_name,
    d.name AS doctor_name,
    'Diagnosis without Appointment' AS mismatch_type
FROM health_care.diagnoses di
LEFT JOIN health_care.appointments a
    ON a.patient_id = di.patient_id
   AND a.doctor_id = di.doctor_id
LEFT JOIN health_care.patients p
    ON di.patient_id = p.patient_id
LEFT JOIN health_care.doctors1 d
    ON di.doctor_id = d.doctor_id
WHERE a.appointment_id IS NULL;
```

Task 5:

Window Functions (Ranking and Aggregation)

- For each doctor, rank their patients based on the number of appointments in descending order.

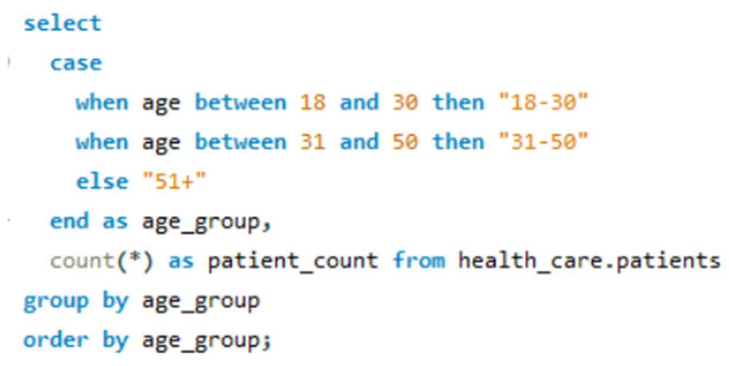


```
select d.name as doctot_name,p.name as patient_name,count(a.appointment_id) as appointment_total,rank() over( partition by d.doctor_id
order by count(a.appointment_id) desc) as p_rank from health_care.appointments as a
inner join doctors1 d on a.doctor_id=d.doctor_id
inner join patients p on a.patient_id=p.patient_id
group by d.name,d.doctor_id,p.name,p.patient_id;
```

Task 6:

Conditional Expressions

- Write a query to categorize patients by age group (e.g., 18-30, 31-50, 51+). Count the number of patients in each age group.



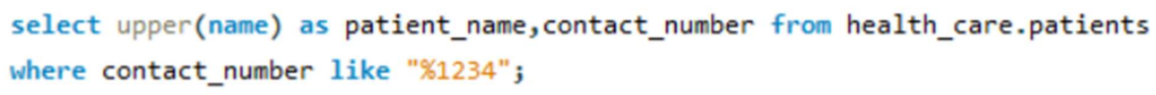
```
select
  case
    when age between 18 and 30 then "18-30"
    when age between 31 and 50 then "31-50"
    else "51+"
  end as age_group,
  count(*) as patient_count from health_care.patients
group by age_group
order by age_group;
```

Task 7:

<u>Numeric and String Functions</u>

- Retrieve a list of patients whose contact numbers end with "1234" and display

their names in uppercase.

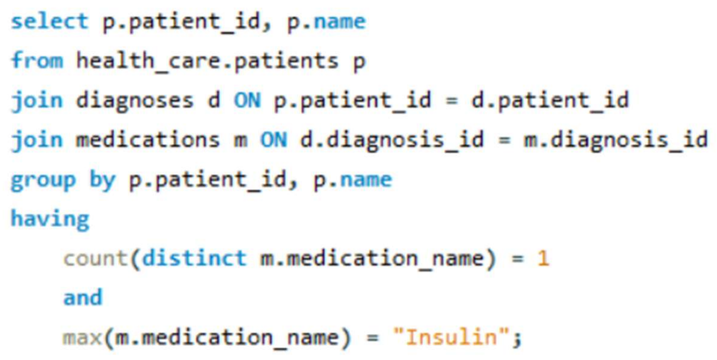


```sql
select upper(name) as patient_name,contact_number from health_care.patients
where contact_number like "%1234";
```

Task 8:

<u>Subqueries for Filtering</u>

- Find patients who have only been prescribed "Insulin" in any of their diagnoses.

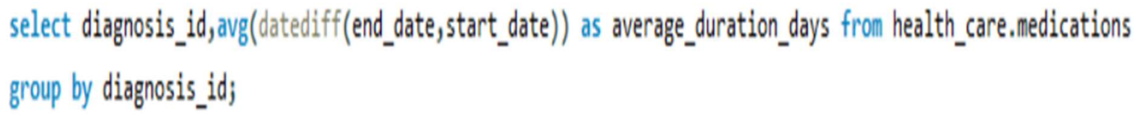


```sql
select p.patient_id, p.name
from health_care.patients p
join diagnoses d ON p.patient_id = d.patient_id
join medications m ON d.diagnosis_id = m.diagnosis_id
group by p.patient_id, p.name
having
    count(distinct m.medication_name) = 1
    and
    max(m.medication_name) = "Insulin";
```

Task 9:

<u>Date and Time Functions</u>

- Calculate the average duration (in days) for which medications are prescribed for each diagnosis.

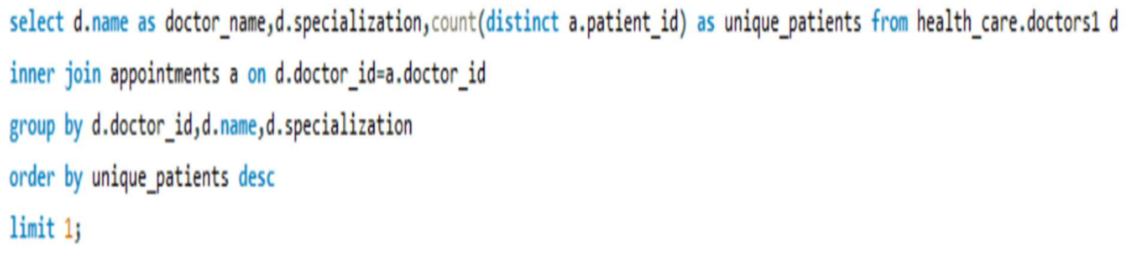


```sql
select diagnosis_id,avg(datediff(end_date,start_date)) as average_duration_days from health_care.medications
group by diagnosis_id;
```

Task 10:

<u>Complex Joins and Aggregation</u>

- Write a query to identify the doctor who has attended the most unique patients, Include the doctor's name, specialization, and the count of unique patients.



```sql
select d.name as doctor_name,d.specialization,count(distinct a.patient_id) as unique_patients from health_care.doctors1 d
inner join appointments a on d.doctor_id=a.doctor_id
group by d.doctor_id,d.name,d.specialization
order by unique_patients desc
limit 1;
```

Conclusion:

This project demonstrates the effective use of advanced SQL techniques to analyse real-world healthcare data and derive meaningful insights. By applying joins, subqueries, window functions, aggregate operations, and date/time functions, the analysis successfully connects patients, doctors, appointments, diagnoses, and treatments into a coherent view. Overall, the project strengthens practical SQL skills while showcasing how structured query logic can support data-driven decision-making and reporting in a healthcare analytics context.