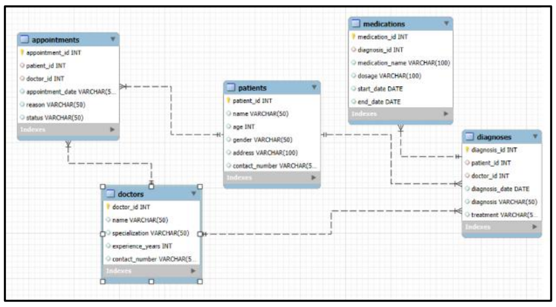
Healthcare Analytics

MYSQL

Problem Statement:

The project aims to analyze 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 analyze healthcare metrics.

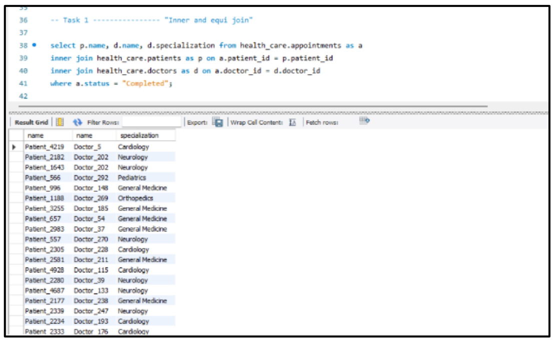
➢ ER Diagram



- The ER diagram illustrates a healthcare database schema with tables for patients, doctors, appointments, diagnoses, and medications, showing how each entity is related through primary and foreign key relationships. It visually organizes the structure and connections needed to track clinical details such as patient visits, doctor assignments, diagnosis events, and prescribed medications.

➢ Inner and Equi Joins

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



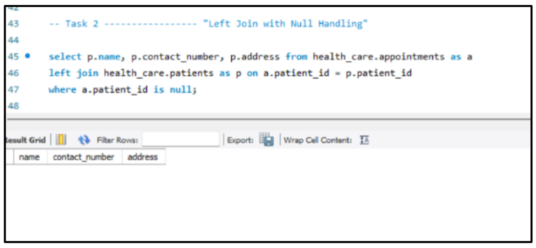
```sql
-- Task 1 --------------- "Inner and equi join"

select p.name, d.name, d.specialization from health_care.appointments as a
inner join health_care.patients as p on a.patient_id = p.patient_id
inner join health_care.doctors as d on a.doctor_id = d.doctor_id
where a.status = "Completed";
```

```
name          name        specialization
Patient_4219  Doctor_5    Cardiology
Patient_2182  Doctor_202  Neurology
Patient_1643  Doctor_202  Neurology
Patient_566   Doctor_292  Pediatrics
Patient_996   Doctor_148  General Medicine
Patient_1188  Doctor_269  Orthopedics
Patient_3255  Doctor_185  General Medicine
Patient_657   Doctor_54   General Medicine
Patient_2983  Doctor_37   General Medicine
Patient_557   Doctor_270  Neurology
Patient_2305  Doctor_228  Cardiology
Patient_2581  Doctor_211  General Medicine
Patient_4928  Doctor_115  Cardiology
Patient_2280  Doctor_39   Neurology
Patient_4687  Doctor_133  Neurology
Patient_2177  Doctor_238  General Medicine
Patient_2339  Doctor_247  Neurology
Patient_2234  Doctor_193  Cardiology
Patient_2333  Doctor_176  Cardiology
```

- This query retrieves details of all completed appointments by joining the **appointments**, **patients**, and **doctors** tables using Inner Joins. It displays each patient's name, doctor's name, and specialization where the appointment status is marked as "Completed".

➢ **Left Join with Null Handling**

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



```
-- Task 2 ----------------- "Left Join with Null Handling"

select p.name, p.contact_number, p.address from health_care.appointments as a
left join health_care.patients as p on a.patient_id = p.patient_id
where a.patient_id is null;
```

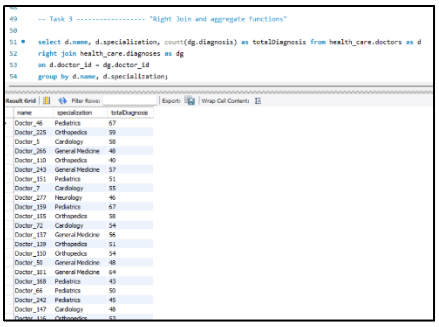
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	contact_number	address

- This query uses a Left Join between the appointments and patients tables to retrieve all patients, including those who have never had an appointment. It checks for NULL in the appointment's patient_id field to filter out patients without any appointments; since the output has no records, it indicates all patients have had at least one appointment.

➢ **Right Join and Aggregate Functions**

Task: 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.



```
-- Task 3 ----------------- "Right Join and aggregate functions"

select d.name, d.specialization, count(dg.diagnosis) as totalDiagnosis from health_care.doctors as d
right join health_care.diagnoses as dg
on d.doctor_id = dg.doctor_id
group by d.name, d.specialization;
```

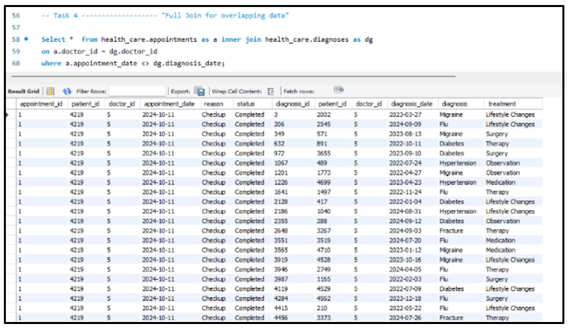
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	specialization	totalDiagnosis
Doctor_46	Pediatrics	67
Doctor_225	Orthopedics	59
Doctor_5	Cardiology	58
Doctor_266	General Medicine	48
Doctor_110	Orthopedics	40
Doctor_243	General Medicine	57
Doctor_151	Pediatrics	51
Doctor_7	Cardiology	55
Doctor_277	Neurology	46
Doctor_159	Pediatrics	67
Doctor_155	Orthopedics	58
Doctor_72	Cardiology	54
Doctor_137	General Medicine	56
Doctor_139	Orthopedics	51
Doctor_150	Orthopedics	54
Doctor_50	General Medicine	48
Doctor_101	General Medicine	64
Doctor_168	Pediatrics	43
Doctor_66	Pediatrics	50
Doctor_242	Pediatrics	45
Doctor_147	Cardiology	48
Doctor_116	Orthopedics	53

- This query uses a Right Join between the doctors and diagnoses tables to find the total number of diagnoses per doctor, including those doctors who have not diagnosed any patients. It groups the results by the doctor's name and specialization, displaying the doctor's details along with the count of diagnoses, which will be zero for doctors without any diagnoses.

➢ **Full Join for Overlapping Data**

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



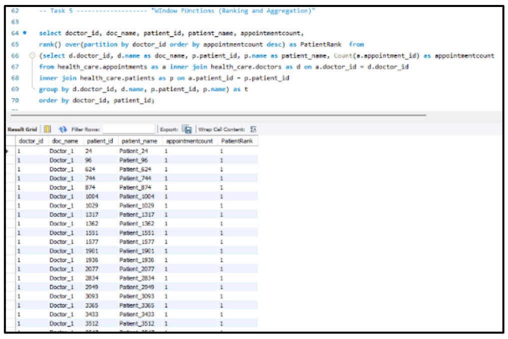
```sql
-- Task 4 ------------------ "Full Join for overlapping data"

Select *  from health_care.appointments as a inner join health_care.diagnoses as dg
on a.doctor_id = dg.doctor_id
where a.appointment_date <> dg.diagnosis_date;
```

appointment_id	patient_id	doctor_id	appointment_date	reason	status	diagnosis_id	patient_id	doctor_id	diagnosis_date	diagnosis	treatment
1	4219	5	2024-10-11	Checkup	Completed	3	2002	5	2023-03-27	Migraine	Lifestyle Changes
1	4219	5	2024-10-11	Checkup	Completed	206	2545	5	2024-09-09	Flu	Lifestyle Changes
1	4219	5	2024-10-11	Checkup	Completed	349	571	5	2023-08-13	Migraine	Surgery
1	4219	5	2024-10-11	Checkup	Completed	632	891	5	2022-10-11	Diabetes	Therapy
1	4219	5	2024-10-11	Checkup	Completed	972	3655	5	2023-09-10	Diabetes	Surgery
1	4219	5	2024-10-11	Checkup	Completed	1067	489	5	2022-07-24	Hypertension	Observation
1	4219	5	2024-10-11	Checkup	Completed	1201	1773	5	2022-04-27	Migraine	Observation
1	4219	5	2024-10-11	Checkup	Completed	1226	4699	5	2023-04-23	Hypertension	Medication
1	4219	5	2024-10-11	Checkup	Completed	1641	1497	5	2022-11-24	Flu	Therapy
1	4219	5	2024-10-11	Checkup	Completed	2128	417	5	2022-01-04	Diabetes	Lifestyle Changes
1	4219	5	2024-10-11	Checkup	Completed	2186	1040	5	2024-08-31	Hypertension	Lifestyle Changes
1	4219	5	2024-10-11	Checkup	Completed	2355	288	5	2024-09-12	Diabetes	Observation
1	4219	5	2024-10-11	Checkup	Completed	2648	3267	5	2024-09-03	Fracture	Therapy
1	4219	5	2024-10-11	Checkup	Completed	3551	3519	5	2024-07-20	Flu	Medication
1	4219	5	2024-10-11	Checkup	Completed	3565	4710	5	2023-01-12	Migraine	Medication
1	4219	5	2024-10-11	Checkup	Completed	3919	4528	5	2023-10-16	Migraine	Lifestyle Changes
1	4219	5	2024-10-11	Checkup	Completed	3946	2749	5	2024-04-05	Flu	Therapy
1	4219	5	2024-10-11	Checkup	Completed	3987	1165	5	2022-02-03	Flu	Surgery
1	4219	5	2024-10-11	Checkup	Completed	4119	4529	5	2022-07-09	Diabetes	Lifestyle Changes
1	4219	5	2024-10-11	Checkup	Completed	4284	4562	5	2023-12-18	Flu	Surgery
1	4219	5	2024-10-11	Checkup	Completed	4415	210	5	2022-05-22	Flu	Lifestyle Changes
1	4219	5	2024-10-11	Checkup	Completed	4456	3373	5	2024-07-26	Fracture	Therapy

- This query attempts to identify mismatches between the appointments and diagnoses tables by performing an Inner Join on doctor_id and filtering on differing dates between appointment_date and diagnosis_date. For a complete comparison including all unmatched records from both tables, a Full Outer Join is typically used, but since MySQL lacks direct support, it requires combining Left and Right Joins with UNION to include all appointments and diagnoses along with their patient and doctor details.

➢ **Window Functions (Ranking and Aggregation)**

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



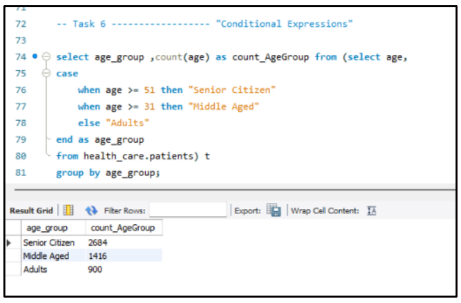
```sql
-- Task 5 ------------------ "WIndow FUnctions (Ranking and Aggregation)"

select doctor_id, doc_name, patient_id, patient_name, appointmentcount,
rank() over(partition by doctor_id order by appointmentcount desc) as PatientRank  from
(select d.doctor_id, d.name as doc_name, p.patient_id, p.name as patient_name, Count(a.appointment_id) as appointmentcount
from health_care.appointments as a inner join health_care.doctors as d on a.doctor_id = d.doctor_id
inner join health_care.patients as p on a.patient_id = p.patient_id
group by d.doctor_id, d.name, p.patient_id, p.name) as t
order by doctor_id, patient_id;
```

doctor_id	doc_name	patient_id	patient_name	appointmentcount	PatientRank
1	Doctor_1	24	Patient_24	1	1
1	Doctor_1	96	Patient_96	1	1
1	Doctor_1	624	Patient_624	1	1
1	Doctor_1	744	Patient_744	1	1
1	Doctor_1	874	Patient_874	1	1
1	Doctor_1	1004	Patient_1004	1	1
1	Doctor_1	1029	Patient_1029	1	1
1	Doctor_1	1317	Patient_1317	1	1
1	Doctor_1	1362	Patient_1362	1	1
1	Doctor_1	1551	Patient_1551	1	1
1	Doctor_1	1577	Patient_1577	1	1
1	Doctor_1	1901	Patient_1901	1	1
1	Doctor_1	1936	Patient_1936	1	1
1	Doctor_1	2077	Patient_2077	1	1
1	Doctor_1	2834	Patient_2834	1	1
1	Doctor_1	2949	Patient_2949	1	1
1	Doctor_1	3093	Patient_3093	1	1
1	Doctor_1	3365	Patient_3365	1	1
1	Doctor_1	3433	Patient_3433	1	1
1	Doctor_1	3512	Patient_3512	1	1

- This query uses SQL window functions to rank each doctor's patients according to the number of appointments, with the highest number ranked first. It leverages a subquery to calculate appointment counts per patient per doctor, then applies the RANK() function partitioned by doctor_id and ordered by appointment count in descending order.

➢ **Conditional Expressions**

Task: 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.



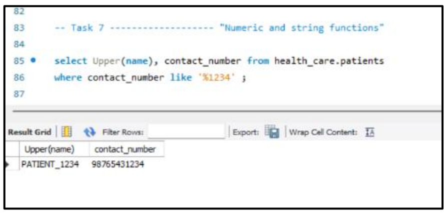
```sql
-- Task 6 ------------------ "Conditional Expressions"

select age_group ,count(age) as count_AgeGroup from (select age,
case
    when age >= 51 then "Senior Citizen"
    when age >= 31 then "Middle Aged"
    else "Adults"
end as age_group
from health_care.patients) t
group by age_group;
```

age_group	count_AgeGroup
Senior Citizen	2684
Middle Aged	1416
Adults	900

- This query uses a CASE statement to categorize patients into age groups such as "Adults," "Middle Aged," and "Senior Citizen" based on their age. It then counts and displays the number of patients in each age group by grouping the results accordingly.

➢ Numeric and String Functions

Task: Retrieve a list of patients whose contact numbers end with "1234" and display their names in uppercase.



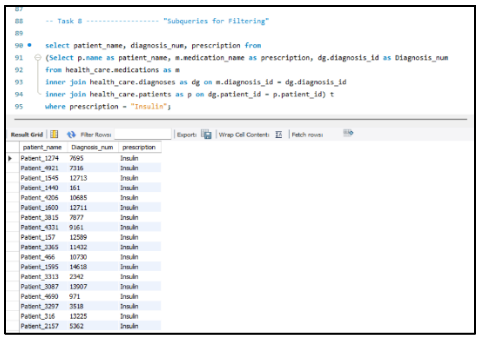
```sql
-- Task 7 ------------------ "Numeric and string functions"

select Upper(name), contact_number from health_care.patients
where contact_number like '%1234' ;
```

Upper(name)	contact_number
PATIENT_1234	98765431234

- This query retrieves the names and contact numbers of patients whose contact numbers end with "1234." It uses the UPPER() function to display each matching patient's name in uppercase for standardized output.

➢ Subqueries for Filtering

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



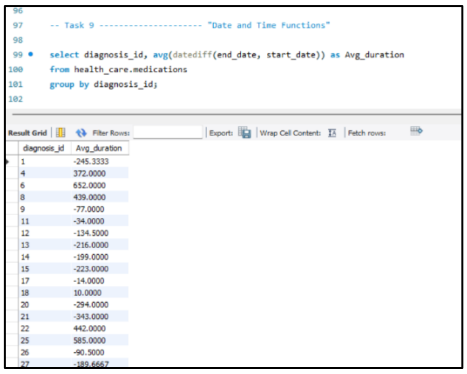
```sql
-- Task 8 ------------------ "Subqueries for Filtering"

select patient_name, diagnosis_num, prescription from
(Select p.name as patient_name, m.medication_name as prescription, dg.diagnosis_id as Diagnosis_num
from health_care.medications as m
inner join health_care.diagnoses as dg on m.diagnosis_id = dg.diagnosis_id
inner join health_care.patients as p on dg.patient_id = p.patient_id) t
where prescription = "Insulin";
```

patient_name	Diagnosis_num	prescription
Patient_1274	7695	Insulin
Patient_4921	7316	Insulin
Patient_1545	12713	Insulin
Patient_1440	161	Insulin
Patient_4206	10685	Insulin
Patient_1600	12711	Insulin
Patient_3815	7877	Insulin
Patient_4331	9161	Insulin
Patient_157	12589	Insulin
Patient_3365	11432	Insulin
Patient_466	10730	Insulin
Patient_1595	14618	Insulin
Patient_3313	2342	Insulin
Patient_3087	13907	Insulin
Patient_4690	971	Insulin
Patient_3297	3518	Insulin
Patient_316	13225	Insulin
Patient_2157	5362	Insulin

- This query uses a subquery to join patients, diagnoses, and medications, assembling a dataset of each patient’s prescribed medications. It then filters the results to show only those patients who have "Insulin" as a prescription in any of their diagnoses.

➢ Date and Time Functions

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

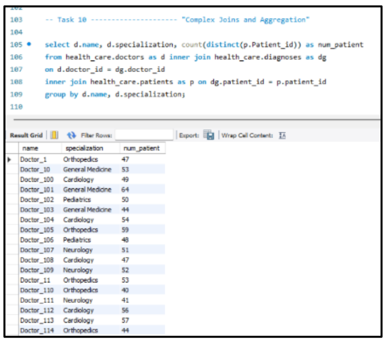


```sql
-- Task 9 -------------------- "Date and Time Functions"

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

diagnosis_id	Avg_duration
1	-245.3333
4	372.0000
6	652.0000
8	439.0000
9	-77.0000
11	-34.0000
12	-134.5000
13	-216.0000
14	-199.0000
15	-223.0000
17	-14.0000
18	10.0000
20	-294.0000
21	-343.0000
22	442.0000
25	585.0000
26	-90.5000
27	-189.6667

- This query calculates the average number of days that medications are prescribed for each diagnosis by using the DATEDIFF function on the medication start and end dates. It groups the results by diagnosis_id and uses the AVG function to determine the average prescription duration per diagnosis.

- **Complex Joins and Aggregation**
  Task: 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
-- Task 10 -------------------- "Complex Joins and Aggregation"

select d.name, d.specialization, count(distinct(p.Patient_id)) as num_patient
from health_care.doctors as d inner join health_care.diagnoses as dg
on d.doctor_id = dg.doctor_id
inner join health_care.patients as p on dg.patient_id = p.patient_id
group by d.name, d.specialization;
```

name	specialization	num_patient
Doctor_1	Orthopedics	47
Doctor_10	General Medicine	53
Doctor_100	Cardiology	49
Doctor_101	General Medicine	64
Doctor_102	Pediatrics	50
Doctor_103	General Medicine	44
Doctor_104	Cardiology	54
Doctor_105	Orthopedics	59
Doctor_106	Pediatrics	48
Doctor_107	Neurology	51
Doctor_108	Cardiology	47
Doctor_109	Neurology	52
Doctor_11	Orthopedics	53
Doctor_110	Orthopedics	40
Doctor_111	Neurology	41
Doctor_112	Cardiology	56
Doctor_113	Cardiology	57
Doctor_114	Orthopedics	44

- This query joins the doctors, diagnoses, and patients tables to count the number of unique patients each doctor has attended, using COUNT(DISTINCT) for accuracy. It displays the doctor's name, specialization, and the total unique patient count, helping identify which doctor has served the most diverse patient base.