Queries

```
Create database Project ;

set sql_safe_updates= 0;
UPDATE diagnoses
SET appointment_id = FLOOR(1 + RAND() * 10000);


-- Primary key --

-- Patients table
ALTER TABLE patients
ADD PRIMARY KEY (patient_id);

-- Doctors table
ALTER TABLE doctors
ADD PRIMARY KEY (doctor_id);

-- Appointments table
ALTER TABLE appointments
ADD PRIMARY KEY (appointment_id);

-- Diagnoses table
ALTER TABLE diagnoses
ADD PRIMARY KEY (diagnosis_id);

-- Medications table
ALTER TABLE medications
ADD PRIMARY KEY (medication_id);

-- foreign key --

-- Link Appointments with Patients
ALTER TABLE appointments
ADD CONSTRAINT fk_appointments_patients
FOREIGN KEY (patient_id) REFERENCES patients(patient_id);

-- Link Appointments with  Doctors
ALTER TABLE appointments
ADD CONSTRAINT fk_appointments_doctors
FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id);

-- Link Diagnoses with Appointments
ALTER TABLE diagnoses
ADD CONSTRAINT fk_diagnoses_appointments
FOREIGN KEY (appointment_id) REFERENCES appointments(appointment_id);
```

```
     -- Link Medications with Diagnoses
     ALTER TABLE medications
     ADD CONSTRAINT fk_medications_diagnoses
     FOREIGN KEY (diagnosis_id) REFERENCES diagnoses(diagnosis_id);


                              --   Queries

  -- Task -1 Inner Joins ( All completed appointments )

  select patients.patient_name, doctors.doctor_name,doctors.specialization,  appointments.status
  from appointments
  Inner join  patients ON patients.patient_id=appointments.patient_id     -- joins appointments with
  patients --
  Inner join  doctors ON doctors.doctor_id=appointments.doctor_id       -- joins appointments with
  doctors --
  WHERE appointments.status='completed'                        -- filtering conditions --
  order by  doctors.specialization ;                         -- sorting the data --

  -- Task -2 Left Joins with Null Handling ( Patients had no appointments )

  Select patients.patient_name , patients.contact_number , patients.address , appointments.status
  from patients
  Left join appointments ON patients.patient_id= appointments.patient_id        -- Left joins patients
  with appointments -
  where appointments.status is null ;                                    -- Handling Null values –

  -- Task - 3 Right Joins and aggregate functions ( Total number of diagnosis for each doctors )

  SELECT doctors.doctor_name , doctors.specialization , COUNT(diagnoses.diagnosis_id) AS
  Total_Diagnosis   -- Aggregate functions count --
  FROM diagnoses
  RIGHT JOIN doctors ON doctors.doctor_id=diagnoses.doctor_id                                   --
  Rightjoins --
  GROUP BY doctors.doctor_name , doctors.specialization                                  --
  Aggregate functions group by –
  order by total_diagnosis desc ;                                   -- order by for finding max diagnosis
  count --

-- Task -4 Full Join ( Mismatches between appointments and diagnoses )

select appointments.appointment_id ,appointments.patient_id ,appointments.doctor_id ,
          diagnoses.diagnosis_id, diagnoses.diagnosis , diagnoses.treatment         -- relevant
columns from appointments & diagnoses --
     from appointments
     left join diagnoses on
     appointments.appointment_id = diagnoses.appointment_id
```

```
       union                                        -- combining 2 tables using union , left and right join
 select   appointments.appointment_id ,appointments.patient_id ,appointments.doctor_id ,
            diagnoses.diagnosis_id, diagnoses.diagnosis , diagnoses.treatment          --  relevant
 columns from appointments & diagnoses --
       from diagnoses
       right join appointments on
       appointments.appointment_id=diagnoses.appointment_id ;              -- matching records from
 both tables --




        --    Task -5 Window functions ( Ranking patients per doctor )

       select  doctor_id , patient_id , count(appointment_id) as total_appointments ,   -- counting total
.no.of appointments
       dense_rank() over ( partition by doctor_id order by count(appointment_id) desc ) as
patients_rank -- applying dense rank to ranking patients and partition by creates separate ranking
       from appointments
       group by doctor_id , patient_id                  -- calculating total appointments for each doctor-
patients
       order by  doctor_id, patients_rank  asc ;         -- sorting the result

             --  Task -6 Conditional Expressions ( Number of patients in age group)

       select
       case                                              -- using case statement
       when age between 18 and 30  then '18-30 ( Young age)'        -- categorizing patients into
different age groups
       when age between 31 and 50 then ' 31-50 (Middle age )'
       when age between 51 and 59 then '51-59 ( Late middle aged )'
       when age >= 60 then '60+ ( Senior citizen )'
       Else 'minor '
       end as Age_group,
       count(*)  as 'Total_patients'                              -- counting total patients
       from patients
       group by age_group                                        -- grouping rows according to age category
       order by age_group ;                                      -- sorting the results




       -- Task -7  Numeric and string functions( for finding Patients contact number )

         SELECT upper(patient_name ) AS PATIENT_NAME , Contact_number    -- Converting all
patients in upper case
          FROM patients                                          -- from patients table
          WHERE contact_number LIKE '%1234' ;                         -- filtering where contactnumber
ends with 1234
```

```
        -- Task -8 Subqueries( retrieving Patients diagnosed insulin )

        select patients.patient_id , patients.patient_name  from patients where patients.patient_id in  --
selecting patients table containing patients details
        (select diagnoses.patient_id from diagnoses
        join medications on diagnoses.diagnosis_id= medications.diagnosis_id  -- join medications
table
        where medications.medication_name='insulin');     -- subquery finds all patients who
diagnosed with insulin using where clause

        -- Task - 9 Datediff (for computing average duration of days for each diagnosis)

        select medication_name ,                                    -- Avg find the average duration  , rounds
the average to 0 decimal points
        round(avg(datediff(end_date , start_date )),0) as Avg_prescribed_days  -- datediff calculates
difference between start date and end date
        from medications
        group by medication_name ;                                  -- grouping results by each medication
name


        -- Task -10 Complex joins with aggregate functions (Doctor attended most unique patients )


        SELECT
           doctors.doctor_name,                                    -- doctor name
           doctors.specialization,                                 -- doctor specialization
           COUNT(DISTINCT appointments.patient_id) AS unique_patients_attended     -- count to
identify unique patients
        FROM
           doctors
        JOIN
           appointments  ON doctors.doctor_id = appointments.doctor_id          -- Joins doctors with
appointments
        GROUP BY
           doctors.doctor_name, doctors.specialization                          -- group results by
doctorname and specialization
        ORDER BY
           unique_patients_attended DESC ;                                 -- ordering highest uniquepatient
id
```