Please past below

---

(1) the SQL code you have used to create the schema of your database (only create table and alter table statements (if any), not statements for inserting values)

-- We have written 2 databases (sql_HOSPITAL_OLD, sql_HOSPITAL_NEW) which the new one is the modified schema and it is in the optimization part.

```
DROP DATABASE IF EXISTS `sql_HOSPITAL_OLD`;
CREATE DATABASE `sql_HOSPITAL_OLD`;
USE `sql_HOSPITAL_OLD`;


DROP TABLE IF EXISTS `Physician`;
CREATE TABLE `Physician` (
  `EmployeeID` BIGINT NOT NULL,
  `Name` CHAR(255) NOT NULL,
  `Position` CHAR(255) NOT NULL,
  `SSN` BIGINT NOT NULL,
  CONSTRAINT `pk_physician` PRIMARY KEY(`EmployeeID`)

DROP TABLE IF EXISTS `Department`;
CREATE TABLE `Department` (
  `DepartmentID` BIGINT NOT NULL,
  `Name` CHAR(255) NOT NULL,
  `Head` BIGINT NOT NULL,
  CONSTRAINT `pk_Department` PRIMARY KEY(`DepartmentID`),
  CONSTRAINT `fk_Department_Physician_EmployeeID` FOREIGN KEY(`Head`)
REFERENCES `Physician`(`EmployeeID`)
);

DROP TABLE IF EXISTS `Affiliated_With`;
CREATE TABLE `Affiliated_With` (
  `Physician` BIGINT NOT NULL,
  `Department` BIGINT NOT NULL,
  `PrimaryAffiliation` BOOLEAN NOT NULL,
  CONSTRAINT `fk_Affiliated_With_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Affiliated_With_Department_DepartmentID` FOREIGN KEY(`Department`)
REFERENCES `Department`(`DepartmentID`),
  PRIMARY KEY(`Physician`, `Department`)
);
```

```
DROP TABLE IF EXISTS `Procedures`;
CREATE TABLE `Procedures` (
  `Code` BIGINT PRIMARY KEY NOT NULL,
  `Name` CHAR(255) NOT NULL,
  `Cost` REAL NOT NULL
);

DROP TABLE IF EXISTS `Trained_In`;
CREATE TABLE `Trained_In` (
  `Physician` BIGINT NOT NULL,
  `Treatment` BIGINT NOT NULL,
  `CertificationDate` DATETIME NOT NULL,
  `CertificationExpires` DATETIME NOT NULL,
  CONSTRAINT `fk_Trained_In_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Trained_In_Procedures_Code` FOREIGN KEY(`Treatment`)
REFERENCES `Procedures`(`Code`),
  PRIMARY KEY(`Physician`, `Treatment`)
);

DROP TABLE IF EXISTS `Patient`;
CREATE TABLE `Patient` (
  `SSN` BIGINT PRIMARY KEY NOT NULL,
  `Name` CHAR(255) NOT NULL,
  `Address` CHAR(255) NOT NULL,
  `Phone` CHAR(255) NOT NULL,
  `InsuranceID` CHAR(255) NOT NULL,
  `PCP` BIGINT NOT NULL,
  CONSTRAINT `fk_Patient_Physician_EmployeeID` FOREIGN KEY(`PCP`) REFERENCES
`Physician`(`EmployeeID`)
);

DROP TABLE IF EXISTS `Nurse`;
CREATE TABLE `Nurse` (
  `EmployeeID` BIGINT PRIMARY KEY NOT NULL,
  `Name` CHAR(255) NOT NULL,
  `Position` CHAR(255) NOT NULL,
  `Registered` BOOLEAN NOT NULL,
  `SSN` BIGINT NOT NULL
);

DROP TABLE IF EXISTS `Appointment`;
CREATE TABLE `Appointment` (
```

```
  `AppointmentID` BIGINT PRIMARY KEY NOT NULL,
  `Patient` BIGINT NOT NULL,
  `PrepNurse` BIGINT,
  `Physician` BIGINT NOT NULL,
  `Start` DATETIME NOT NULL,
  `End` DATETIME NOT NULL,
  `ExaminationRoom` LONGTEXT NOT NULL,
  CONSTRAINT `fk_Appointment_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Appointment_Nurse_EmployeeID` FOREIGN KEY(`PrepNurse`)
REFERENCES `Nurse`(`EmployeeID`),
  CONSTRAINT `fk_Appointment_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`)
);

DROP TABLE IF EXISTS `Medication`;
CREATE TABLE `Medication` (
  `Code` BIGINT PRIMARY KEY NOT NULL,
  `Name` CHAR(255) NOT NULL,
  `Brand` CHAR(255) NOT NULL,
  `Description` CHAR(255) NOT NULL
);

DROP TABLE IF EXISTS `Prescribes`;
CREATE TABLE `Prescribes` (
  `Physician` BIGINT NOT NULL,
  `Patient` BIGINT NOT NULL,
  `Medication` BIGINT NOT NULL,
  `Date` DATETIME NOT NULL,
  `Appointment` BIGINT,
  `Dose` CHAR(255) NOT NULL,
  PRIMARY KEY(`Physician`, `Patient`, `Medication`, `Date`),
  CONSTRAINT `fk_Prescribes_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Prescribes_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Prescribes_Medication_Code` FOREIGN KEY(`Medication`)
REFERENCES `Medication`(`Code`),
  CONSTRAINT `fk_Prescribes_Appointment_AppointmentID` FOREIGN KEY(`Appointment`)
REFERENCES `Appointment`(`AppointmentID`)
);

DROP TABLE IF EXISTS `Block`;
```

```
CREATE TABLE `Block` (
  `BlockFloor` BIGINT NOT NULL,
  `BlockCode` BIGINT NOT NULL,
  PRIMARY KEY(`BlockFloor`, `BlockCode`)
);

DROP TABLE IF EXISTS `Room`;
CREATE TABLE `Room` (
  `RoomNumber` BIGINT PRIMARY KEY NOT NULL,
  `RoomType` CHAR(255) NOT NULL,
  `BlockFloor` BIGINT NOT NULL,
  `BlockCode` BIGINT NOT NULL,
  `Unavailable` BOOLEAN NOT NULL,
  CONSTRAINT `fk_Room_Block_PK` FOREIGN KEY(`BlockFloor`, `BlockCode`)
REFERENCES `Block`(`BlockFloor`, `BlockCode`)
);

DROP TABLE IF EXISTS `On_Call`;
CREATE TABLE `On_Call` (
  `Nurse` BIGINT NOT NULL,
  `BlockFloor` BIGINT NOT NULL,
  `BlockCode` BIGINT NOT NULL,
  `OnCallStart` DATETIME NOT NULL,
  `OnCallEnd` DATETIME NOT NULL,
  PRIMARY KEY(`Nurse`, `BlockFloor`, `BlockCode`, `OnCallStart`, `OnCallEnd`),
  CONSTRAINT `fk_OnCall_Nurse_EmployeeID` FOREIGN KEY(`Nurse`) REFERENCES
`Nurse`(`EmployeeID`),
  CONSTRAINT `fk_OnCall_Block_Floor` FOREIGN KEY(`BlockFloor`, `BlockCode`)
REFERENCES `Block`(`BlockFloor`, `BlockCode`)
);

DROP TABLE IF EXISTS `Stay`;
CREATE TABLE `Stay` (
  `StayID` BIGINT PRIMARY KEY NOT NULL,
  `Patient` BIGINT NOT NULL,
  `Room` BIGINT NOT NULL,
  `StayStart` DATETIME NOT NULL,
  `StayEnd` DATETIME NOT NULL,
  CONSTRAINT `fk_Stay_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Stay_Room_Number` FOREIGN KEY(`Room`) REFERENCES
`Room`(`RoomNumber`)
);
```

```
DROP TABLE IF EXISTS `Undergoes`;
CREATE TABLE `Undergoes` (
  `Patient` BIGINT NOT NULL,
  `Procedures` BIGINT NOT NULL,
  `Stay` BIGINT NOT NULL,
  `DateUndergoes` DATETIME NOT NULL,
  `Physician` BIGINT NOT NULL,
  `AssistingNurse` BIGINT,
  PRIMARY KEY(`Patient`, `Procedures`, `Stay`, `DateUndergoes`),
  CONSTRAINT `fk_Undergoes_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Undergoes_Procedures_Code` FOREIGN KEY(`Procedures`)
REFERENCES `Procedures`(`Code`),
  CONSTRAINT `fk_Undergoes_Stay_StayID` FOREIGN KEY(`Stay`) REFERENCES
`Stay`(`StayID`),
  CONSTRAINT `fk_Undergoes_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Undergoes_Nurse_EmployeeID` FOREIGN KEY(`AssistingNurse`)
REFERENCES `Nurse`(`EmployeeID`)
);

-- For the optimization part-1 (materialized view) which is described further.

CREATE TABLE table_patient_procedure AS
        SELECT patient.SSN AS PatientId,
    patient.Name AS Patient_Name,
    patient.Address AS Patient_Address,
        patient.Phone AS Patient_Contact,
    patient.InsuranceID AS Patient_Insurance,
    undergoes.DateUndergoes AS Procedure_Date,
    procedures.Name AS Procedure_Name,
        physician.Name AS Physician_Name,
    physician.Position AS Physician_Position
FROM patient, undergoes, procedures, physician
WHERE patient.SSN = undergoes.Patient
        AND procedures.Code = undergoes.Procedures
    AND undergoes.Physician = physician.EmployeeID
    AND undergoes.DateUndergoes > '2020-04-01'
        AND undergoes.DateUndergoes < '2020-04-30';

CREATE TABLE table_patient_doctors_nurse AS
 SELECT patient.SSN AS PatientId,
```

```
 patient.Name AS Patient_Name,
 patient.Address AS Patient_Address,
 patient.Phone AS Patient_Contact,
 patient.InsuranceID AS Patient_Insurance,
 appointment.Start AS Appointment_Date,
 physician.Name AS Physician_Name,
 physician.Position AS Physician_Position,
 nurse.Name AS Nurse_Name,
 nurse.Position AS Nurse_Position
 FROM patient, appointment, physician, nurse
 WHERE patient.SSN = appointment.Patient
            AND appointment.Physician =physician.EmployeeID
       AND appointment.PrepNurse = nurse.EmployeeID
       AND appointment.Start >= '2020-04-01'
       AND appointment.End   <= '2020-04-30';

CREATE TABLE table_patient_prescribed_medicines AS
SELECT patient.SSN AS PatientId,
          patient.Name AS Patient_Name,
          patient.Address AS Patient_Address,
          patient.Phone AS Patient_Contact,
          patient.InsuranceID AS Patient_Insurance,
          appointment.Start AS Appointment_Date,
          medication.Name AS Medication_Name,
          prescribes.Dose AS Dosage
FROM patient,appointment,prescribes,medication
WHERE patient.SSN = appointment.Patient
           AND appointment.AppointmentID = prescribes.Appointment
      AND prescribes.Medication = medication.code
      AND appointment.Start >= '2020-04-01'
      AND appointment.End <= '2020-04-30';

CREATE TABLE table_patient_appointment_procedure AS
          SELECT patient.Name AS Patient_Name,
                    patient.Address AS Patient_Address,
                    patient.Phone AS Patient_Contact,
                    appointment.Start AS Appointment_Date,
                    procedures.Name AS Procedure_Name
    FROM patient,
                  appointment,
                  undergoes,
                  procedures
    WHERE appointment.Patient = undergoes.Patient
```

```
                AND procedures.Code = undergoes.Procedures
                AND patient.SSN = appointment.Patient
                AND patient.SSN = undergoes.Patient ;

-- For the optimization part 2 (changing query and Indexing)

ALTER TABLE Appointment
add key covered(Patient, PrepNurse, Physician);
ALTER TABLE Nurse
ADD KEY Nurse_Name_Registered(Name, Registered);
```

---

(2) the SQL code of the queries (possibly with an explanation)

```
USE sql_HOSPITAL;
-- q1
-- Availability of rooms in each block in each floor
SELECT blockfloor AS "Floor",
    blockcode AS "Block",
    count(*) "Number of available rooms"
FROM Room
WHERE unavailable='false'
GROUP BY blockfloor,
     blockcode
ORDER BY blockfloor,
     blockcode;

-- Q-2
-- Find out details of prescribed medicines for patients between the 1st to 5th of April.
-- Medication name
-- Dosage
-- Patient details
SELECT Medication.Name AS Medication_Name,
    Prescribes.Dose  AS Dosage,
          Patient.SSN AS PatientId,
          Patient.Name AS Patient_Name,
    Appointment.Start AS Appointment_Date,
          Patient.Address AS Patient_Address,
    patient.Phone AS Patient_Contact,
    patient.InsuranceID AS Patient_Insurance
FROM patient,appointment,prescribes,medication
WHERE patient.SSN = appointment.Patient
          AND appointment.AppointmentID = prescribes.Appointment
    AND prescribes.Medication = medication.code
```

```
    AND appointment.Start >= '2020-04-01'
    AND appointment.End <= '2020-04-05';

-- q3
-- Physicians who did the undergoe but their certification expiration had been passed
-- Patient, Physician, UndergoesDate, Certification_End_Date

SELECT
P.Name AS PatientName,
U.physician as PhysicianID,
U.DateUndergoes AS UdergoesDate,
IFNULL(T.CertificationExpires,'VERIFIED') AS CERTIFICATE_EXPIRES
FROM undergoes U
LEFT JOIN patient P
        ON U.Patient=P.SSN
LEFT JOIN trained_in T
        USING(physician)
ORDER BY DateUndergoes;

-- q4
-- Names of all the physicians,
-- Their procedures which were NOT certified for THAT procedure.
-- Date when the procedure was carried out and
-- Name of the patient on which procedure have been carried out

SELECT p.name AS "Physician",
      pr.name AS "Procedure",
      u.DateUndergoes,
      pt.name AS "Patient"
FROM Physician p,
    Undergoes u,
    Patient pt,
    Procedures pr
WHERE u.patient = pt.SSN
  AND u.Procedures = pr.Code
  AND u.physician = p.EmployeeID
  AND NOT EXISTS
   ( SELECT *
    FROM Trained_In t
    WHERE t.treatment = u.Procedures
      AND t.physician = u.physician );

-- q5
```

```
-- Patients who had at least two appointment
-- Nurse who prepared the appointment which was a registered nurse
-- And the physician who has carried out primary care.

SELECT pt.name AS "Patient",
       p.name AS "Primary Physician",
       n.name AS "Nurse"
FROM Appointment a
JOIN Patient pt ON a.patient=pt.ssn
JOIN Nurse n ON a.prepnurse=n.employeeid
JOIN Physician p ON pt.pcp=p.employeeid
WHERE a.patient IN
    (SELECT Patient
     FROM Appointment a
     GROUP BY a.patient
     HAVING count(*)>=2)
  AND n.registered='true'
ORDER BY pt.name;

-- q6
-- All the nurses who have ever been on call for room 102.
SELECT n.name
FROM nurse n
WHERE employeeid IN
    ( SELECT oc.Nurse
     FROM on_call oc,
          room r
     WHERE oc.blockfloor = r.blockfloor
       AND oc.blockcode = r.blockcode
       AND r.roomnumber = 102);

-- Q-7
-- Finding all On call nurses,
-- Patients in different blocks
-- And their rooms
-- Between dates 01-04-2020 and 05-04-2020.

 SELECT nurse.Name as NurseName,
                    nurse.Position as NursePosition,
                    on_call.OnCallStart,
        on_call.OnCallEnd,
        patient.Name AS Patient_Name,
                    patient.Address AS Patient_Address,
```

```
        patient.phone AS Patient_Contact,
                    patient.InsuranceID AS Patient_Insuranceid
FROM on_call, nurse, patient
WHERE patient.SSN != nurse.SSN
            AND on_call.Nurse = nurse.EmployeeID
      AND on_call.OnCallStart >= '2020-04-01'
            AND on_call.OnCallEnd <= '2020-04-30'
      AND nurse.Registered=1
      ORDER BY BlockFloor ASC;

-- Q-8
-- Physicians and department relations with certification start and end dates.
-- Physician name
-- Position
-- Department
-- Start and End of their certification
SELECT physician.Name AS Physician_Name,
     physician.Position AS Physician_Position,
     department.Name AS Department_Name,
       trained_in.CertificationDate,
       trained_in.CertificationExpires
FROM affiliated_with, department,physician, trained_in
WHERE affiliated_with.Physician = physician.EmployeeID
    AND affiliated_with.Department = department.DepartmentID
limit 100;

-- Q-9
-- Patients who STAYED but did NOT UNDERGO on any procedure between two dates (using
left join)
-- Patients details with Start and End date of stay

SELECT patient.SSN AS PatientId,
             patient.Name AS Patient_Name,
             patient.Address AS Patient_Address,
             patient.Phone AS Patient_Contact,
             stay.StayStart,
       stay.StayEnd
FROM stay
     left JOIN undergoes ON stay.Patient != undergoes.Patient
     left JOIN patient ON stay.Patient = patient.SSN
WHERE stay.StayStart >='2020-04-20'
      AND stay.StayEnd<='2020-04-23';
```

```
-- Q-10
-- Patients who got APPOINTMENTS, but had NOT been PRESCRIBED.
-- PATIENT details
SELECT distinct patient.Name as Patient_Name,
                                patient.Address as Patient_Address,
                                patient.Phone as Patient_Contact,
                                physician.Name as Physician_Name,
                                physician.Position as Physician_Position,
                                appointment.Start as Appointment_Date
FROM patient,
                appointment, prescribes,
                physician
WHERE       appointment.AppointmentID <> prescribes.Appointment
                AND physician.EmployeeID=appointment.Physician
      AND appointment.patient=patient.SSN;

-- Q-11
-- Details of patients
-- Undergoing particular procedure on "2020-04-25"
-- With doctor's name and procedure name.

SELECT patient.SSN as PatientId,
            patient.Name as Patient_Name,
      patient.Address as Patient_Address,
      patient.Phone as Patient_Contact,
      patient.InsuranceID as Patient_Insurance,
      undergoes.DateUndergoes as Procedure_Date,
      procedures.Name as Procedure_Name,
      physician.Name as Physician_Name,
      physician.Position as Physician_Position
FROM patient, undergoes, procedures, physician
WHERE patient.SSN = undergoes.Patient
          AND procedures.Code = undergoes.Procedures
          AND undergoes.Physician=physician.EmployeeID
          AND undergoes.DateUndergoes >= '2020-04-25'
          AND undergoes.DateUndergoes < '2020-04-26';

-- Q-12
-- Find out the appointment details
-- Of patients between 01-04-2020 to 05-04-2020
-- With different doctors with their attending nurse name.

 SELECT patient.SSN as PatientId,
```

```
                patient.Name as Patient_Name,
        patient.Address as Patient_Address,
                patient.Phone as Patient_Contact,
                patient.InsuranceID as Patient_Insurance,
                appointment.Start Appointment_Date,
                physician.Name as Physician_Name,
        physician.Position as Physician_Position,
                nurse.Name as Nurse_Name,
        nurse.Position as Nurse_Position
FROM patient, appointment, physician, nurse
WHERE patient.SSN = appointment.Patient
            AND appointment.Physician =physician.EmployeeID
            AND appointment.PrepNurse = nurse.EmployeeID
      AND appointment.Start >= '2020-04-01'
      AND appointment.End <= '2020-04-05';

 -- Q-13
-- PATIENTS THAT HAD APPOINTMENTS WHICH PROCEEDED TO AN UNDERGOE

SELECT patient.Name AS Patient_Name,
             appointment.Start AS Appointment_Date,
             procedures.Name AS Procedure_Name,
             patient.Phone AS Patient_Contact
FROM patient, appointment, undergoes, procedures
WHERE appointment.Patient = undergoes.Patient
            AND procedures.Code=undergoes.Procedures
            AND patient.SSN=appointment.Patient
            AND patient.SSN=undergoes.Patient;
-- Patient, Physician, UndergoesDate, Certification_End_Date
```

---

(3) the SQL code used for query optimization for HW2. For each query, indicate the un-optimized version and the optimized one. In case the optimization has been realized through indexes, insert the SQL code for the index creation; in case you have modified the schema (e.g. changed the domain of a field, or constructed a new materialized table, etc.), insert the code you have used for this modification.

```
-- OPTIMIZATION SECTION
-- Part-1 USING Materialized Views
-- In this Part we USE materialized views for optimizing performance.
-- For materialized views a seperate table needs to be created.
```

```
-- OPTIMIZATION I (query 11)

CREATE TABLE table_patient_procedure AS
            SELECT patient.SSN AS PatientId,
      patient.Name AS Patient_Name,
      patient.Address AS Patient_Address,
            patient.Phone AS Patient_Contact,
      patient.InsuranceID AS Patient_Insurance,
      undergoes.DateUndergoes AS Procedure_Date,
      procedures.Name AS Procedure_Name,
            physician.Name AS Physician_Name,
      physician.Position AS Physician_Position
FROM patient, undergoes, procedures, physician
WHERE patient.SSN = undergoes.Patient
            AND procedures.Code = undergoes.Procedures
      AND undergoes.Physician = physician.EmployeeID
      AND undergoes.DateUndergoes > '2020-04-01'
            AND undergoes.DateUndergoes < '2020-04-30';

-- Optionally we can add index(es) for the queries we want to speed up.

CREATE INDEX index_procedure_date
              ON table_patient_procedure (Procedure_Date);

CREATE VIEW patient_and_procedure AS
       SELECT * FROM table_patient_procedure;

/* On large database we can see
   query executes 10x faster than the first query */

-- SLOW QUERY

SELECT patient.SSN as PatientId,
              patient.Name as Patient_Name,
       patient.Address as Patient_Address,
       patient.Phone as Patient_Contact,
       patient.InsuranceID as Patient_Insurance,
       undergoes.DateUndergoes as Procedure_Date,
       procedures.Name as Procedure_Name,
       physician.Name as Physician_Name,
       physician.Position as Physician_Position
FROM patient, undergoes, procedures, physician
WHERE patient.SSN = undergoes.Patient
```

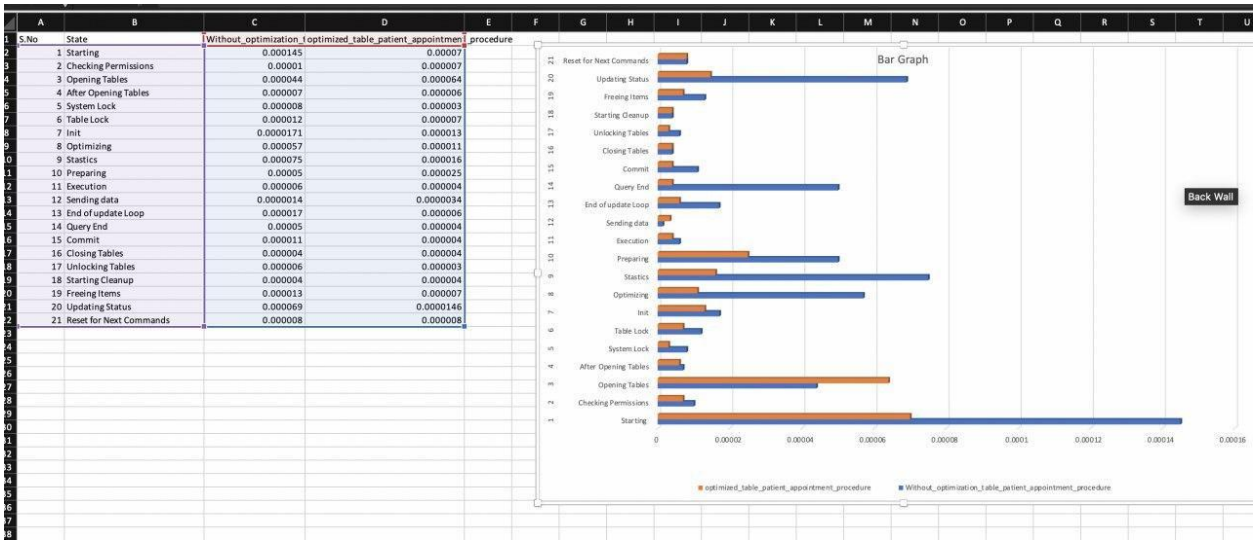
```
 AND procedures.Code = undergoes.Procedures
 AND undergoes.Physician=physician.EmployeeID
 AND undergoes.DateUndergoes >= '2020-04-25'
 AND undergoes.DateUndergoes < '2020-04-26';

 SHOW STATUS like "Last_Query_Cost";

-- Fast query

EXPLAIN SELECT *
FROM patient_and_procedure
WHERE Procedure_Date >= '2020-04-25'
            AND Procedure_Date < '2020-04-26';

SHOW STATUS like "Last_Query_Cost";

-- OPTIMIZATION II (query 12)

CREATE TABLE table_patient_doctors_nurse AS
 SELECT patient.SSN AS PatientId,
 patient.Name AS Patient_Name,
 patient.Address AS Patient_Address,
 patient.Phone AS Patient_Contact,
 patient.InsuranceID AS Patient_Insurance,
 appointment.Start AS Appointment_Date,
 physician.Name AS Physician_Name,
 physician.Position AS Physician_Position,
 nurse.Name AS Nurse_Name,
 nurse.Position AS Nurse_Position
 FROM patient, appointment, physician, nurse
 WHERE patient.SSN = appointment.Patient
             AND appointment.Physician =physician.EmployeeID
       AND appointment.PrepNurse = nurse.EmployeeID
       AND appointment.Start >= '2020-04-01'
       AND appointment.End   <= '2020-04-30';

CREATE INDEX index_appointment_date
          ON table_patient_doctors_nurse (Appointment_Date);

CREATE VIEW patient_and_doctors_nurse AS
   SELECT * FROM table_patient_doctors_nurse;
```

```
/*-- Slow Query --*/
 SELECT patient.SSN as PatientId,
                patient.Name as Patient_Name,
       patient.Address as Patient_Address,
                patient.Phone as Patient_Contact,
                patient.InsuranceID as Patient_Insurance,
                appointment.Start Appointment_Date,
                physician.Name as Physician_Name,
       physician.Position as Physician_Position,
                nurse.Name as Nurse_Name,
       nurse.Position as Nurse_Position
FROM patient, appointment, physician, nurse
WHERE patient.SSN = appointment.Patient
            AND appointment.Physician =physician.EmployeeID
            AND appointment.PrepNurse = nurse.EmployeeID
     AND appointment.Start >= '2020-04-01'
     AND appointment.End <= '2020-04-05';

SHOW STATUS like "Last_Query_Cost";
-- Slow query Cost : 21.913

-- Fast Query
Explain
Select *
FROM patient_and_doctors_nurse
WHERE Appointment_Date >= "2020-04-01"
            AND Appointment_Date<="2020-04-05";

SHOW STATUS like "Last_Query_Cost";
-- Fast query Cost : 10.749
```

```
-- OPTIMIZATION - III - query 2

CREATE TABLE table_patient_prescribed_medicines AS
SELECT patient.SSN AS PatientId,
          patient.Name AS Patient_Name,
          patient.Address AS Patient_Address,
          patient.Phone AS Patient_Contact,
          patient.InsuranceID AS Patient_Insurance,
          appointment.Start AS Appointment_Date,
          medication.Name AS Medication_Name,
          prescribes.Dose AS Dosage
FROM patient,appointment,prescribes,medication
WHERE patient.SSN = appointment.Patient
            AND appointment.AppointmentID = prescribes.Appointment
     AND prescribes.Medication = medication.code
     AND appointment.Start >= '2020-04-01'
     AND appointment.End <= '2020-04-30';

CREATE INDEX index_appointment_date
          ON table_patient_prescribed_medicines (Appointment_Date);

CREATE VIEW patient_prescribed_medicines AS
  SELECT * FROM table_patient_prescribed_medicines;

-- Slow Query

SELECT medication.Name AS Medication_Name,
       prescribes.Dose  AS Dosage,
              patient.SSN AS PatientId,
             patient.Name AS Patient_Name,
       appointment.Start AS Appointment_Date,
             patient.Address AS Patient_Address,
       patient.Phone AS Patient_Contact,
       patient.InsuranceID AS Patient_Insurance
FROM patient,appointment,prescribes,medication
WHERE patient.SSN = appointment.Patient
            AND appointment.AppointmentID = prescribes.Appointment
     AND prescribes.Medication = medication.code
     AND appointment.Start >= '2020-04-01'
     AND appointment.End <= '2020-04-05';

SHOW STATUS like "Last_Query_Cost";
-- Slow query Cost : '27.544'
```

```
-- Fast Query
Explain
SELECT *
FROM patient_prescribed_medicines
WHERE Appointment_Date >= "2020-04-01"
AND Appointment_Date<="2020-04-05";

SHOW STATUS like "Last_Query_Cost";
-- Fast query Cost : 10.249


-- OPTIMIZATION - IV - query 13

CREATE TABLE table_patient_appointment_procedure AS
        SELECT patient.Name AS Patient_Name,
                  patient.Address AS Patient_Address,
                  patient.Phone AS Patient_Contact,
                  appointment.Start AS Appointment_Date,
                  procedures.Name AS Procedure_Name
   FROM patient,
                appointment,
                undergoes,
                procedures
   WHERE appointment.Patient = undergoes.Patient
                 AND procedures.Code = undergoes.Procedures
                 AND patient.SSN = appointment.Patient
                 AND patient.SSN = undergoes.Patient ;

CREATE INDEX index_appointment_date
        ON table_patient_appointment_procedure (Appointment_Date);

CREATE VIEW patient_appointment_procedure AS
  SELECT * FROM table_patient_appointment_procedure;

-- Slow Query
SELECT patient.Name AS Patient_Name,
           patient.Address AS Patient_Address,
           patient.Phone AS Patient_Contact,
           appointment.Start AS Appointment_Date,
           procedures.Name AS Procedure_Name
FROM patient, appointment, undergoes, procedures
WHERE appointment.Patient = undergoes.Patient
```

```
        AND procedures.Code=undergoes.Procedures
        AND patient.SSN=appointment.Patient
        AND patient.SSN=undergoes.Patient;

SHOW STATUS like "Last_Query_Cost";
-- Slow query Cost : '67.013'

-- Fast Query
Explain
SELECT *
FROM patient_appointment_procedure;
        -- WHERE Appointment_Date >= "2020-04-01"
        -- AND Appointment_Date<="2020-04-05";

SHOW STATUS like "Last_Query_Cost";
-- Fast query Cost : 0.949

-- Analytics:
-- This is the query analysis through profiling in MySQL which we have visualized.

-- The blue lines are without optimization cost of query 13
-- The orange lines are optimized cost of query 13
```



S.No	State	Without_optimization_t	optimized_table_patient_appointmen	procedure
1	Starting	0.000145	0.00007	
2	Checking Permissions	0.00001	0.000007	
3	Opening Tables	0.000044	0.000064	
4	After Opening Tables	0.000007	0.000006	
5	System Lock	0.000008	0.000003	
6	Table Lock	0.000012	0.000007	
7	Init	0.0000171	0.000013	
8	Optimizing	0.000057	0.000011	
9	Stastics	0.000075	0.000016	
10	Preparing	0.00005	0.000025	
11	Execution	0.000006	0.000004	
12	Sending data	0.0000014	0.0000034	
13	End of update Loop	0.000017	0.000006	
14	Query End	0.00005	0.000004	
15	Commit	0.000011	0.000004	
16	Closing Tables	0.000004	0.000004	
17	Unlocking Tables	0.000006	0.000003	
18	Starting Cleanup	0.000004	0.000004	
19	Freeing Items	0.000013	0.000007	
20	Updating Status	0.000069	0.0000146	
21	Reset for Next Commands	0.000008	0.000008	


```
/* -- Part 2 Changing Query Statement and Indexing */

-- OPTIMIZATION - 5 - query 5
```

```
 /*-- Slow Query --*/

 EXPLAIN SELECT pt.name AS "Patient",
        p.name AS "Primary Physician",
        n.name AS "Nurse"
FROM Appointment a
JOIN Patient pt ON a.patient=pt.ssn
JOIN Nurse n ON a.prepnurse=n.employeeid
JOIN Physician p ON pt.pcp=p.employeeid
WHERE a.patient IN
    (SELECT Patient
     FROM Appointment a
     GROUP BY a.patient
     HAVING count(*)>=2)
  AND n.registered='true'
ORDER BY pt.name;

show status like "last_query_cost";
-- Slow query Cost : '72.498000'

-- Fast query:
-- 1

ALTER TABLE Appointment
add key covered(Patient, PrepNurse, Physician);
ALTER TABLE Nurse
ADD KEY Nurse_Name_Registered(Name, Registered);

EXPLAIN SELECT pt.name AS "Patient",
       p.name AS "Primary Physician",
       n.name AS "Nurse"
FROM Appointment a
JOIN (select SSN,Name,PCP from Patient order by Name) pt ON a.patient=pt.ssn
JOIN (select EmployeeID,Name from Nurse where Registered = "true") n ON
a.prepnurse=n.employeeid
JOIN (select EmployeeID,Name from Physician) p ON pt.pcp=p.employeeid
WHERE a.patient IN
    (SELECT Patient
     FROM Appointment a
     GROUP BY a.patient
     HAVING count(*)>=2);

show status like "last_query_cost";
```

```
-- Fast Query cost = '62.498000'
-- 2
EXPLAIN SELECT pt.name AS "Patient",
       p.name AS "Primary Physician",
       n.name AS "Nurse"
FROM  (SELECT Patient,PrepNurse,Physician
     FROM Appointment a
     WHERE Patient in (select Patient from Appointment a group by a.patient having
count(*)>=2)) a
JOIN (select SSN,Name,PCP from Patient order by Name) pt ON a.patient=pt.ssn
JOIN (select EmployeeID,Name from Nurse where Registered = "true") n ON
a.prepnurse=n.employeeid
JOIN (select EmployeeID,Name from Physician) p ON pt.pcp=p.employeeid;

show status like "last_query_cost";

-- Fast Query cost = '62.498000'


/*  Part 3 Changing Schema */

-- For this part we have modified the schema known as sql_HOSPITAL_NEW

DROP DATABASE IF EXISTS `sql_HOSPITAL_NEW`;
CREATE DATABASE `sql_HOSPITAL_NEW`;
USE `sql_HOSPITAL_NEW`;


DROP TABLE IF EXISTS `Physician`;
CREATE TABLE `Physician` (
  `EmployeeID` SMALLINT(4) NOT NULL,
  `Name` VARCHAR(30) NOT NULL,
  `Position` VARCHAR(30) NOT NULL,
  `SSN` INT(9) NOT NULL,
  CONSTRAINT `pk_physician` PRIMARY KEY(`EmployeeID`)
);

DROP TABLE IF EXISTS `Department`;
CREATE TABLE `Department` (
  `DepartmentID` TINYINT(2) NOT NULL,
  `Name` VARCHAR(30) NOT NULL,
  `Head` SMALLINT(4) NOT NULL,
```

```
  CONSTRAINT `pk_Department` PRIMARY KEY(`DepartmentID`),
  CONSTRAINT `fk_Department_Physician_EmployeeID` FOREIGN KEY(`Head`)
REFERENCES `Physician`(`EmployeeID`)
);

DROP TABLE IF EXISTS `Affiliated_With`;
CREATE TABLE `Affiliated_With` (
  `Physician` SMALLINT(4) NOT NULL,
  `Department` TINYINT(2) NOT NULL,
  `PrimaryAffiliation` BOOLEAN NOT NULL,
  CONSTRAINT `fk_Affiliated_With_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Affiliated_With_Department_DepartmentID` FOREIGN KEY(`Department`)
REFERENCES `Department`(`DepartmentID`),
  PRIMARY KEY(`Physician`, `Department`)
);

DROP TABLE IF EXISTS `Procedures`;
CREATE TABLE `Procedures` (
  `Code` TINYINT(2) PRIMARY KEY NOT NULL,
  `Name` VARCHAR(30) NOT NULL,
  `Cost` REAL NOT NULL
);

DROP TABLE IF EXISTS `Trained_In`;
CREATE TABLE `Trained_In` (
  `Physician` SMALLINT(4) NOT NULL,
  `Treatment` TINYINT(2) NOT NULL,
  `CertificationDate` TIMESTAMP NOT NULL,
  `CertificationExpires` TIMESTAMP NOT NULL,
  CONSTRAINT `fk_Trained_In_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Trained_In_Procedures_Code` FOREIGN KEY(`Treatment`)
REFERENCES `Procedures`(`Code`),
  PRIMARY KEY(`Physician`, `Treatment`)
);

DROP TABLE IF EXISTS `Patient`;
CREATE TABLE `Patient` (
  `SSN` INT(9) PRIMARY KEY NOT NULL,
  `Name` VARCHAR(30) NOT NULL,
  `Address` VARCHAR(60) NOT NULL,
  `Phone` VARCHAR(16) NOT NULL,
```

```
  `InsuranceID` VARCHAR(15) NOT NULL,
  `PCP` SMALLINT(4) NOT NULL,
  CONSTRAINT `fk_Patient_Physician_EmployeeID` FOREIGN KEY(`PCP`) REFERENCES
`Physician`(`EmployeeID`)
);

DROP TABLE IF EXISTS `Nurse`;
CREATE TABLE `Nurse` (
  `EmployeeID` SMALLINT(4) PRIMARY KEY NOT NULL,
  `Name` VARCHAR(30) NOT NULL,
  `Position` VARCHAR(30) NOT NULL,
  `Registered` BOOLEAN NOT NULL,
  `SSN` INT(9) NOT NULL
);

DROP TABLE IF EXISTS `Appointment`;
CREATE TABLE `Appointment` (
  `AppointmentID` INT(8) PRIMARY KEY NOT NULL,
  `Patient` INT(9) NOT NULL,
  `PrepNurse` SMALLINT(4),
  `Physician` SMALLINT(4) NOT NULL,
  `Start` TIMESTAMP NOT NULL,
  `End` TIMESTAMP NOT NULL,
  `ExaminationRoom` TINYTEXT NOT NULL,
  CONSTRAINT `fk_Appointment_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Appointment_Nurse_EmployeeID` FOREIGN KEY(`PrepNurse`)
REFERENCES `Nurse`(`EmployeeID`),
  CONSTRAINT `fk_Appointment_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`)
);

DROP TABLE IF EXISTS `Medication`;
CREATE TABLE `Medication` (
  `Code` TINYINT PRIMARY KEY NOT NULL,
  `Name` VARCHAR(30) NOT NULL,
  `Brand` VARCHAR(30) NOT NULL,
  `Description` VARCHAR(30) NOT NULL
);

DROP TABLE IF EXISTS `Prescribes`;
CREATE TABLE `Prescribes` (
  `Physician` SMALLINT(4) NOT NULL,
```

```
  `Patient` INT(9) NOT NULL,
  `Medication` TINYINT NOT NULL,
  `Date` TIMESTAMP NOT NULL,
  `Appointment` INT(8),
  `Dose` VARCHAR(15) NOT NULL,
  PRIMARY KEY(`Physician`, `Patient`, `Medication`, `Date`),
  CONSTRAINT `fk_Prescribes_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Prescribes_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Prescribes_Medication_Code` FOREIGN KEY(`Medication`)
REFERENCES `Medication`(`Code`),
  CONSTRAINT `fk_Prescribes_Appointment_AppointmentID` FOREIGN KEY(`Appointment`)
REFERENCES `Appointment`(`AppointmentID`)
);

DROP TABLE IF EXISTS `Block`;
CREATE TABLE `Block` (
  `BlockFloor` TINYINT(1) NOT NULL,
  `BlockCode` TINYINT(2) NOT NULL,
  PRIMARY KEY(`BlockFloor`, `BlockCode`)
);

DROP TABLE IF EXISTS `Room`;
CREATE TABLE `Room` (
  `RoomNumber` SMALLINT(4) PRIMARY KEY NOT NULL,
  `RoomType` VARCHAR(30) NOT NULL,
  `BlockFloor` TINYINT(1) NOT NULL,
  `BlockCode` TINYINT(2) NOT NULL,
  `Unavailable` BOOLEAN NOT NULL,
  CONSTRAINT `fk_Room_Block_PK` FOREIGN KEY(`BlockFloor`, `BlockCode`)
REFERENCES `Block`(`BlockFloor`, `BlockCode`)
);

DROP TABLE IF EXISTS `On_Call`;
CREATE TABLE `On_Call` (
  `Nurse` SMALLINT(3) NOT NULL,
  `BlockFloor` TINYINT(1) NOT NULL,
  `BlockCode` TINYINT(2) NOT NULL,
  `OnCallStart` TIMESTAMP NOT NULL,
  `OnCallEnd` TIMESTAMP NOT NULL,
  PRIMARY KEY(`Nurse`, `BlockFloor`, `BlockCode`, `OnCallStart`, `OnCallEnd`),
```

```
  CONSTRAINT `fk_OnCall_Nurse_EmployeeID` FOREIGN KEY(`Nurse`) REFERENCES
`Nurse`(`EmployeeID`),
  CONSTRAINT `fk_OnCall_Block_Floor` FOREIGN KEY(`BlockFloor`, `BlockCode`)
REFERENCES `Block`(`BlockFloor`, `BlockCode`)
);

DROP TABLE IF EXISTS `Stay`;
CREATE TABLE `Stay` (
  `StayID` SMALLINT(4) PRIMARY KEY NOT NULL,
  `Patient` INT(9) NOT NULL,
  `Room` SMALLINT(4) NOT NULL,
  `StayStart` TIMESTAMP NOT NULL,
  `StayEnd` TIMESTAMP NOT NULL,
  CONSTRAINT `fk_Stay_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Stay_Room_Number` FOREIGN KEY(`Room`) REFERENCES
`Room`(`RoomNumber`)
);

DROP TABLE IF EXISTS `Undergoes`;
CREATE TABLE `Undergoes` (
  `Patient` INT(9) NOT NULL,
  `Procedures` TINYINT(2) NOT NULL,
  `Stay` SMALLINT(4) NOT NULL,
  `DateUndergoes` TIMESTAMP NOT NULL,
  `Physician` SMALLINT(4) NOT NULL,
  `AssistingNurse` SMALLINT(4),
  PRIMARY KEY(`Patient`, `Procedures`, `Stay`, `DateUndergoes`),
  CONSTRAINT `fk_Undergoes_Patient_SSN` FOREIGN KEY(`Patient`) REFERENCES
`Patient`(`SSN`),
  CONSTRAINT `fk_Undergoes_Procedures_Code` FOREIGN KEY(`Procedures`)
REFERENCES `Procedures`(`Code`),
  CONSTRAINT `fk_Undergoes_Stay_StayID` FOREIGN KEY(`Stay`) REFERENCES
`Stay`(`StayID`),
  CONSTRAINT `fk_Undergoes_Physician_EmployeeID` FOREIGN KEY(`Physician`)
REFERENCES `Physician`(`EmployeeID`),
  CONSTRAINT `fk_Undergoes_Nurse_EmployeeID` FOREIGN KEY(`AssistingNurse`)
REFERENCES `Nurse`(`EmployeeID`)
);
-----------------------------------
/* Slow Query */
-- Optimization for q10
```

```
USE sql_HOSPITAL_OLD;
SELECT distinct patient.Name as Patient_Name,
                                patient.Address as Patient_Address,
                                patient.Phone as Patient_Contact,
                                physician.Name as Physician_Name,
                                physician.Position as Physician_Position,
                                appointment.Start as Appointment_Date
FROM patient,
                appointment, prescribes,
                physician
WHERE       appointment.AppointmentID <> prescribes.Appointment
                AND physician.EmployeeID=appointment.Physician
      AND appointment.patient=patient.SSN;

SHOW STATUS LIKE "last_query_cost";
-- cost 1082.47

/* Faster Query */

USE sql_HOSPITAL_NEW;
SELECT distinct patient.Name as Patient_Name,
                                patient.Address as Patient_Address,
                                patient.Phone as Patient_Contact,
                                physician.Name as Physician_Name,
                                physician.Position as Physician_Position,
                                appointment.Start as Appointment_Date
FROM patient,
                appointment, prescribes,
                physician
WHERE       appointment.AppointmentID <> prescribes.Appointment
                AND physician.EmployeeID=appointment.Physician
      AND appointment.patient=patient.SSN;

SHOW STATUS LIKE "last_query_cost";
-- cost 1080.56
```

/* Additional Information */

-- We have also put the database on the Heroku platform which response to our queries using RESTful API which we have developed using python and Flask micro framework.

-- Sample API response in Browser and also in Postman: