HOSPITAL MANAGEMENT SYSTEM

ISM 6218

Group:

NITIKA MISHRA

RAHUL BANKEY

AKASH MISHRA

Table of Contents

Executive Summary

Hospitals are teeming with people like Doctors, Nurses, Staff, Chemists, and Patients, and they have a variety of jobs like Patient Admission, Doctor's Investigation, Operation, Diagnosis, Bill Payment, Room Services, Medicine Issuance, and so on. The hospital management system is a database system that stores information about healthcare and assists healthcare providers in effectively completing their jobs.

Hospitals, like any other industry, now require a DBMS to manage their data. Whether the patient requires efficient health care in a hospital, private health center, clinic, or office management. All patient medical histories, including information about hospital departments, physicians, and other staff members, must be included in medical records. From a DBA perspective, the main entity sets include the Doctor, Nurse, Patient, and Staff. Another core entity set has Room, Medicine, Bill, Payment, and Diagnosis. All these entity sets are created as tables on top of the database with multiple joins in between them.

Project Requirements:

1. Every hospital should have a patient registration system
2. Every patient should book an appointment to see a doctor.
3. Each hospital will have doctors from different specialties.
4. Every patient should have a valid appointment Id to meet a doctor.
5. A patient should have a proper diagnosis for claim purposes.
6. A patient should have their age, contact details, gender and address specified for diagnosis.
7. Hospital should maintain records of each patient.
8. Hospital should calculate charges and generate bills for each patient.
9. Each lab in the hospital should maintain lab reports for the patients

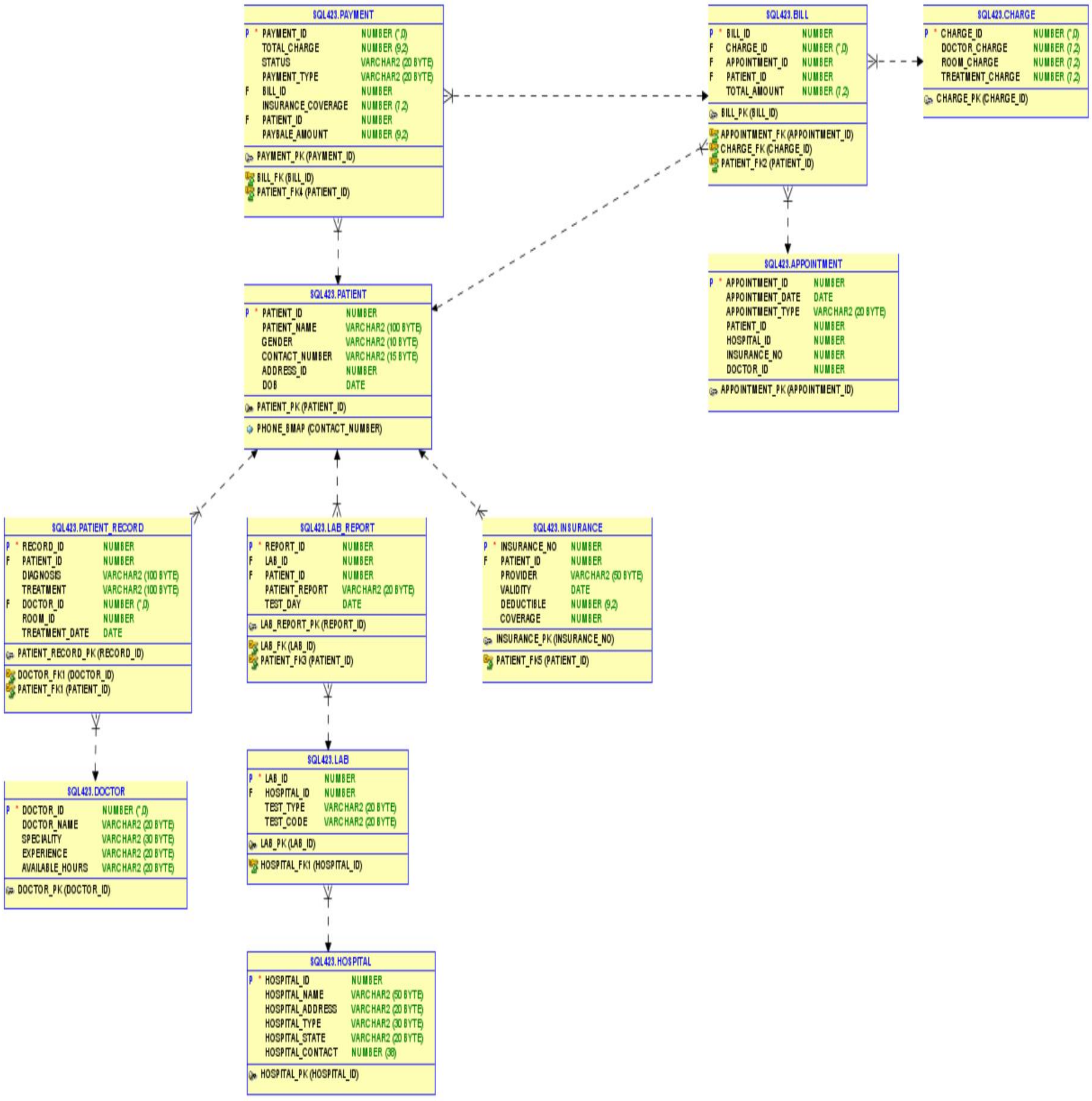
Assumptions

1. A patient can be diagnosed by multiple doctors.
2. A Doctor should have only one specialty.
3. Appointment time for a patient should be in between the available hours of a doctor.
4. Total Amount in the bill is the sum of doctor charges, Room Charges, and treatment charges.

Topic Area Distribution

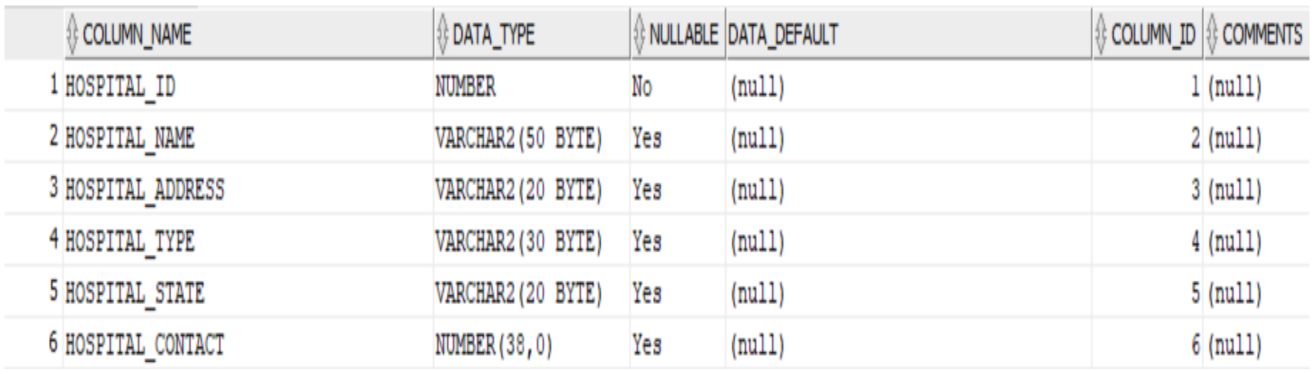
Topic Area	**Description**	**Points**
Database Design	This part should include a logical database design (for the relational model), using normalization to control redundancy and integrity constraints for data quality.	30
Query Writing	This part is another chance to write SQL queries, explore transactions, and even do some database programming for stored procedures.	30
Performance Tuning	In this section, you can capitalize and extend your prior experiments with indexing, optimizer modes, partitioning, parallel execution and any other techniques you want to further explore.	30
Other Topics	Here you are free to explore any other topics of interest. Suggestions include DBA scripts, database security, interface design, data visualization, data mining, and NoSQL databases.	10

Entity Relationship Diagram


SQL423.PAYMENT
P * PAYMENT_ID NUMBER (*,0)
TOTAL_CHARGE NUMBER (9,2)
STATUS VARCHAR2 (20 BYTE)
PAYMENT_TYPE VARCHAR2 (20 BYTE)
F BILL_ID NUMBER
INSURANCE_COVERAGE NUMBER (7,2)
F PATIENT_ID NUMBER
PAYBALE_AMOUNT NUMBER (9,2)
PAYMENT_PK (PAYMENT_ID)
BILL_FK (BILL_ID)
PATIENT_FK4 (PATIENT_ID)
SQL423.BILL
P * BILL_ID NUMBER
F CHARGE_ID NUMBER (*,0)
F APPOINTMENT_ID NUMBER
F PATIENT_ID NUMBER
TOTAL_AMOUNT NUMBER (7,2)
BILL_PK (BILL_ID)
APPOINTMENT_FK (APPOINTMENT_ID)
CHARGE_FK (CHARGE_ID)
PATIENT_FK2 (PATIENT_ID)
SQL423.CHARGE
P * CHARGE_ID NUMBER (*,0)
DOCTOR_CHARGE NUMBER (7,2)
ROOM_CHARGE NUMBER (7,2)
TREATMENT_CHARGE NUMBER (7,2)
CHARGE_PK (CHARGE_ID)
SQL423.APPOINTMENT
P * APPOINTMENT_ID NUMBER
APPOINTMENT_DATE DATE
APPOINTMENT_TYPE VARCHAR2 (20 BYTE)
PATIENT_ID NUMBER
HOSPITAL_ID NUMBER
INSURANCE_NO NUMBER
DOCTOR_ID NUMBER
APPOINTMENT_PK (APPOINTMENT_ID)
SQL423.PATIENT
P * PATIENT_ID NUMBER
PATIENT_NAME VARCHAR2 (100 BYTE)
GENDER VARCHAR2 (10 BYTE)
CONTACT_NUMBER VARCHAR2 (15 BYTE)
ADDRESS_ID NUMBER
DOB DATE
PATIENT_PK (PATIENT_ID)
PHONE_BMAP (CONTACT_NUMBER)
SQL423.PATIENT_RECORD
P * RECORD_ID NUMBER
F PATIENT_ID NUMBER
DIAGNOSIS VARCHAR2 (100 BYTE)
TREATMENT VARCHAR2 (100 BYTE)
F DOCTOR_ID NUMBER (*,0)
ROOM_ID NUMBER
TREATMENT_DATE DATE
PATIENT_RECORD_PK (RECORD_ID)
DOCTOR_FK1 (DOCTOR_ID)
PATIENT_FK1 (PATIENT_ID)
SQL423.LAB_REPORT
P * REPORT_ID NUMBER
F LAB_ID NUMBER
F PATIENT_ID NUMBER
PATIENT_REPORT VARCHAR2 (20 BYTE)
TEST_DAY DATE
LAB_REPORT_PK (REPORT_ID)
LAB_FK (LAB_ID)
PATIENT_FK3 (PATIENT_ID)
SQL423.INSURANCE
P * INSURANCE_NO NUMBER
F PATIENT_ID NUMBER
PROVIDER VARCHAR2 (50 BYTE)
VALIDITY DATE
DEDUCTIBLE NUMBER (9,2)
COVERAGE NUMBER
INSURANCE_PK (INSURANCE_NO)
PATIENT_FK5 (PATIENT_ID)
SQL423.DOCTOR
P * DOCTOR_ID NUMBER (*,0)
DOCTOR_NAME VARCHAR2 (20 BYTE)
SPECIALITY VARCHAR2 (30 BYTE)
EXPERIENCE VARCHAR2 (20 BYTE)
AVAILABLE_HOURS VARCHAR2 (20 BYTE)
DOCTOR_PK (DOCTOR_ID)
SQL423.LAB
P * LAB_ID NUMBER
F HOSPITAL_ID NUMBER
TEST_TYPE VARCHAR2 (20 BYTE)
TEST_CODE VARCHAR2 (20 BYTE)
LAB_PK (LAB_ID)
HOSPITAL_FK1 (HOSPITAL_ID)
SQL423.HOSPITAL
P * HOSPITAL_ID NUMBER
HOSPITAL_NAME VARCHAR2 (50 BYTE)
HOSPITAL_ADDRESS VARCHAR2 (20 BYTE)
HOSPITAL_TYPE VARCHAR2 (30 BYTE)
HOSPITAL_STATE VARCHAR2 (20 BYTE)
HOSPITAL_CONTACT NUMBER (38)
HOSPITAL_PK (HOSPITAL_ID)

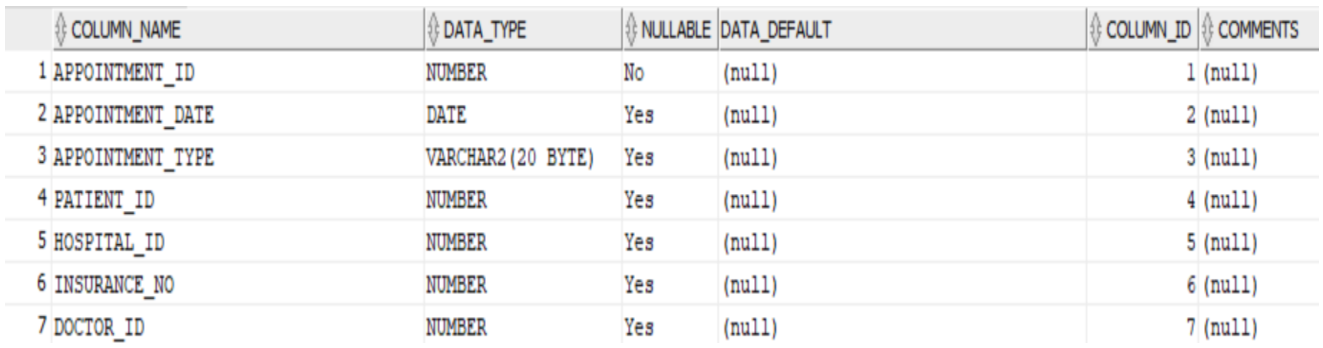
Data Dictionary

Hospital:



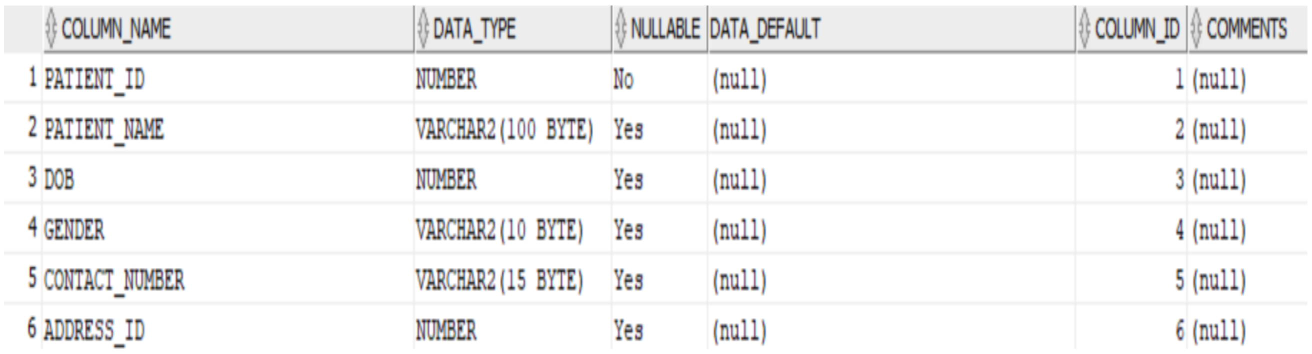
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	HOSPITAL_ID	NUMBER	No	(null)	1	(null)
2	HOSPITAL_NAME	VARCHAR2(50 BYTE)	Yes	(null)	2	(null)
3	HOSPITAL_ADDRESS	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	HOSPITAL_TYPE	VARCHAR2(30 BYTE)	Yes	(null)	4	(null)
5	HOSPITAL_STATE	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)
6	HOSPITAL_CONTACT	NUMBER(38,0)	Yes	(null)	6	(null)

Appointment:



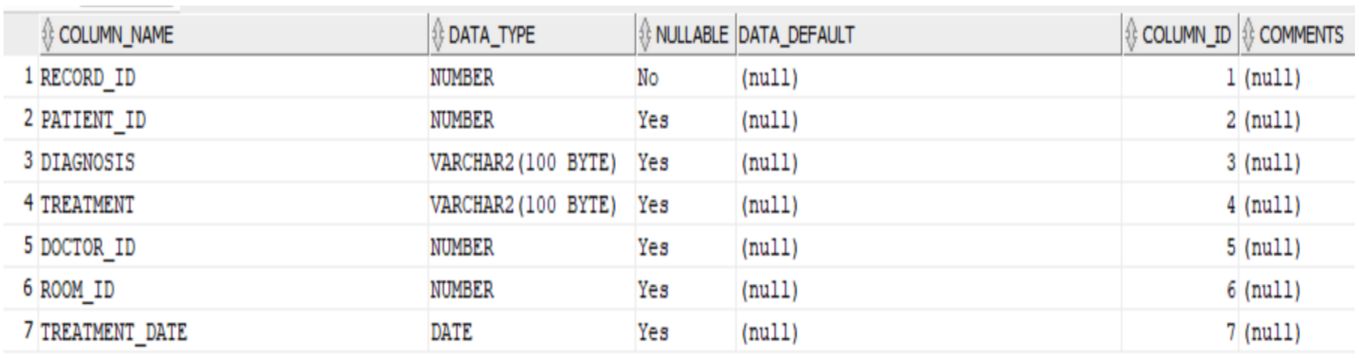
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	APPOINTMENT_ID	NUMBER	No	(null)	1	(null)
2	APPOINTMENT_DATE	DATE	Yes	(null)	2	(null)
3	APPOINTMENT_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	PATIENT_ID	NUMBER	Yes	(null)	4	(null)
5	HOSPITAL_ID	NUMBER	Yes	(null)	5	(null)
6	INSURANCE_NO	NUMBER	Yes	(null)	6	(null)
7	DOCTOR_ID	NUMBER	Yes	(null)	7	(null)

Patient:



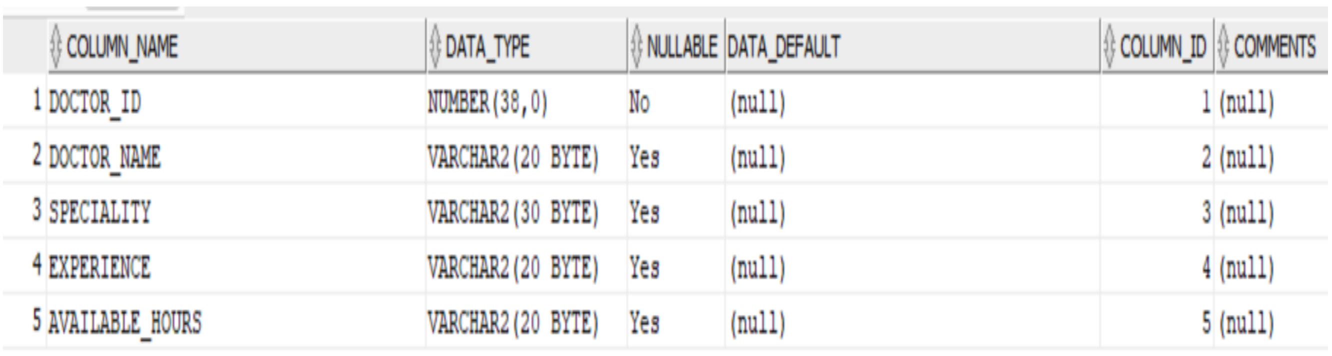
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	PATIENT_ID	NUMBER	No	(null)	1	(null)
2	PATIENT_NAME	VARCHAR2(100 BYTE)	Yes	(null)	2	(null)
3	DOB	NUMBER	Yes	(null)	3	(null)
4	GENDER	VARCHAR2(10 BYTE)	Yes	(null)	4	(null)
5	CONTACT_NUMBER	VARCHAR2(15 BYTE)	Yes	(null)	5	(null)
6	ADDRESS_ID	NUMBER	Yes	(null)	6	(null)

Patient Record:



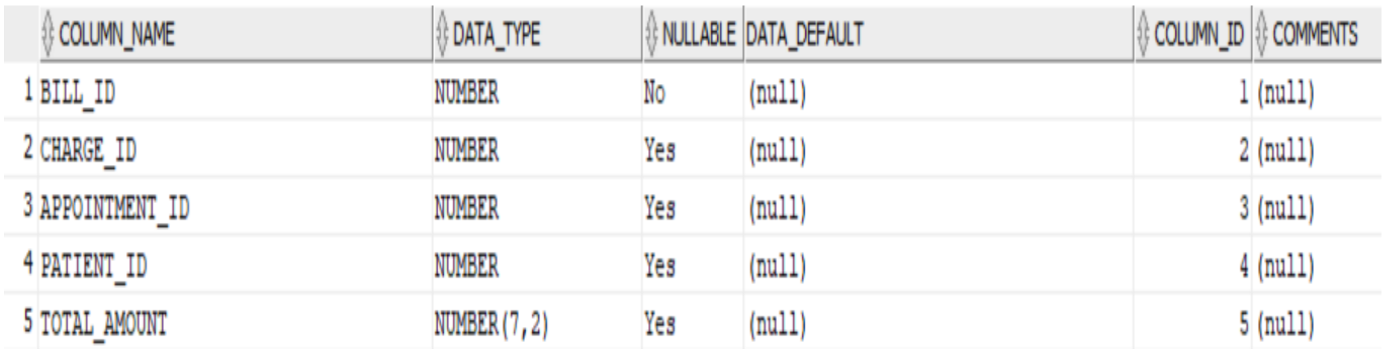
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	RECORD_ID	NUMBER	No	(null)	1	(null)
2	PATIENT_ID	NUMBER	Yes	(null)	2	(null)
3	DIAGNOSIS	VARCHAR2(100 BYTE)	Yes	(null)	3	(null)
4	TREATMENT	VARCHAR2(100 BYTE)	Yes	(null)	4	(null)
5	DOCTOR_ID	NUMBER	Yes	(null)	5	(null)
6	ROOM_ID	NUMBER	Yes	(null)	6	(null)
7	TREATMENT_DATE	DATE	Yes	(null)	7	(null)

Doctor:



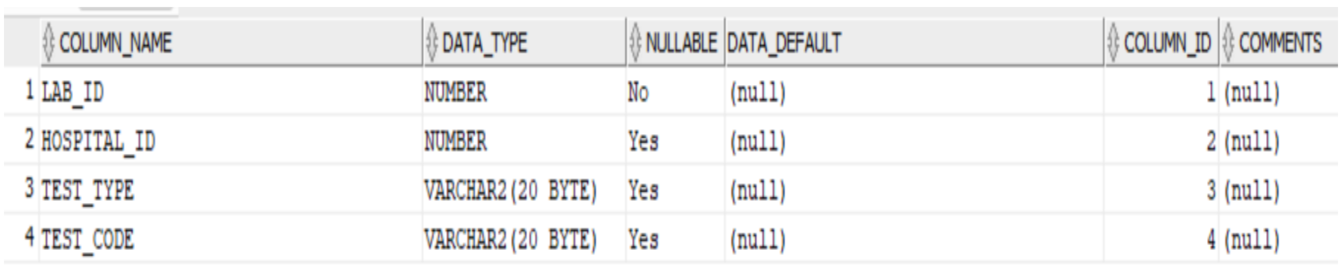
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	DOCTOR_ID	NUMBER(38,0)	No	(null)	1	(null)
2	DOCTOR_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	SPECIALITY	VARCHAR2(30 BYTE)	Yes	(null)	3	(null)
4	EXPERIENCE	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	AVAILABLE_HOURS	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)

Bill:



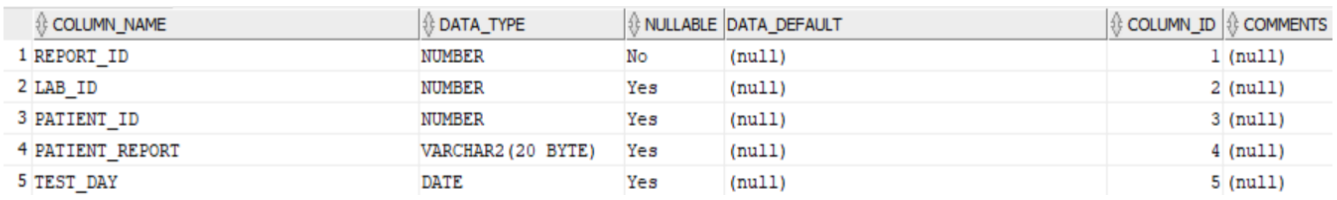
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	BILL_ID	NUMBER	No	(null)	1	(null)
2	CHARGE_ID	NUMBER	Yes	(null)	2	(null)
3	APPOINTMENT_ID	NUMBER	Yes	(null)	3	(null)
4	PATIENT_ID	NUMBER	Yes	(null)	4	(null)
5	TOTAL_AMOUNT	NUMBER(7,2)	Yes	(null)	5	(null)

Lab:



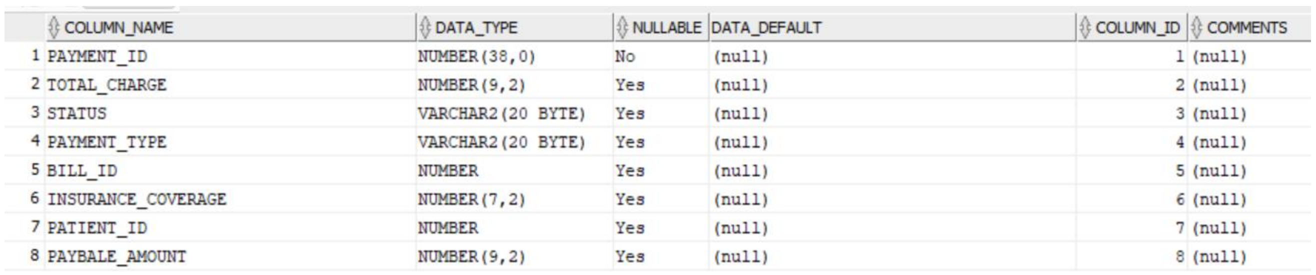
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	LAB_ID	NUMBER	No	(null)	1	(null)
2	HOSPITAL_ID	NUMBER	Yes	(null)	2	(null)
3	TEST_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	TEST_CODE	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)

Lab Report:



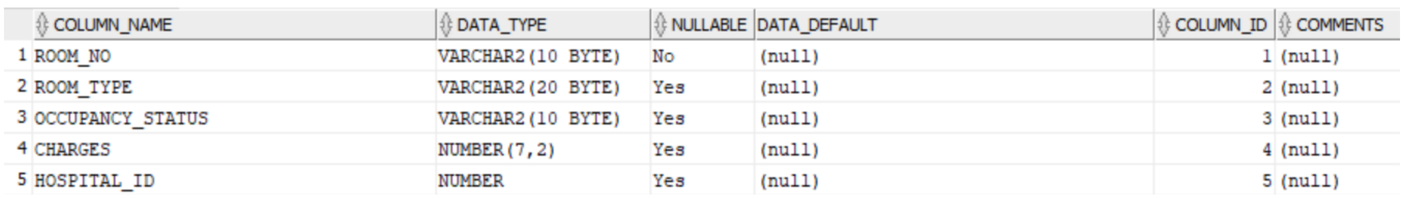
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	REPORT_ID	NUMBER	No	(null)	1	(null)
2	LAB_ID	NUMBER	Yes	(null)	2	(null)
3	PATIENT_ID	NUMBER	Yes	(null)	3	(null)
4	PATIENT_REPORT	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	TEST_DAY	DATE	Yes	(null)	5	(null)

Payment:



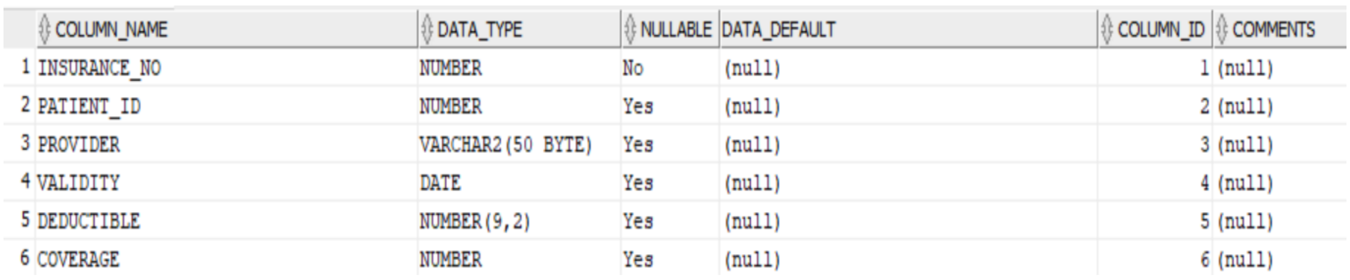
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	PAYMENT_ID	NUMBER(38,0)	No	(null)	1	(null)
2	TOTAL_CHARGE	NUMBER(9,2)	Yes	(null)	2	(null)
3	STATUS	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	PAYMENT_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	BILL_ID	NUMBER	Yes	(null)	5	(null)
6	INSURANCE_COVERAGE	NUMBER(7,2)	Yes	(null)	6	(null)
7	PATIENT_ID	NUMBER	Yes	(null)	7	(null)
8	PAYBALE_AMOUNT	NUMBER(9,2)	Yes	(null)	8	(null)

Room:



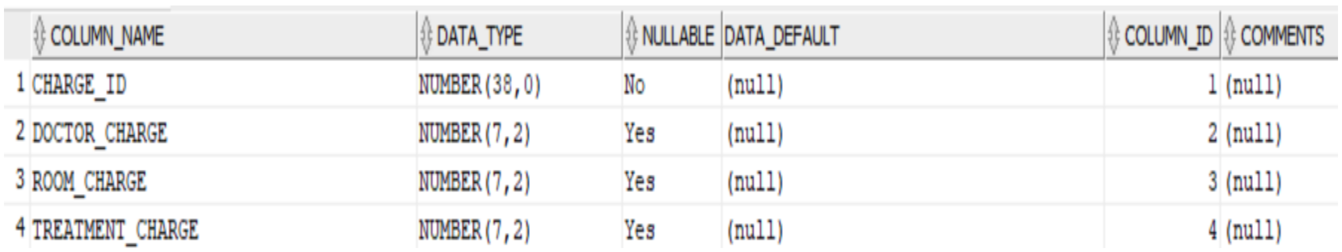
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ROOM_NO	VARCHAR2(10 BYTE)	No	(null)	1	(null)
2	ROOM_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	OCCUPANCY_STATUS	VARCHAR2(10 BYTE)	Yes	(null)	3	(null)
4	CHARGES	NUMBER(7,2)	Yes	(null)	4	(null)
5	HOSPITAL_ID	NUMBER	Yes	(null)	5	(null)

Insurance:



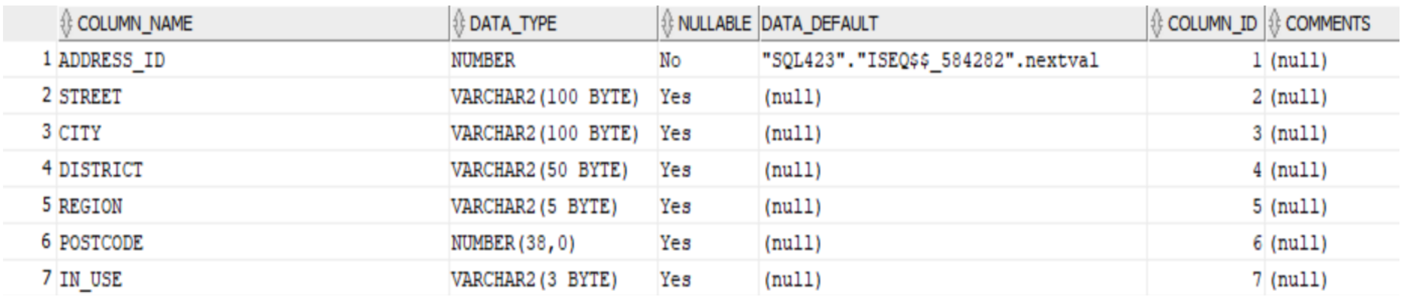
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	INSURANCE_NO	NUMBER	No	(null)	1	(null)
2	PATIENT_ID	NUMBER	Yes	(null)	2	(null)
3	PROVIDER	VARCHAR2(50 BYTE)	Yes	(null)	3	(null)
4	VALIDITY	DATE	Yes	(null)	4	(null)
5	DEDUCTIBLE	NUMBER(9,2)	Yes	(null)	5	(null)
6	COVERAGE	NUMBER	Yes	(null)	6	(null)

Charge:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CHARGE_ID	NUMBER(38,0)	No	(null)	1	(null)
2	DOCTOR_CHARGE	NUMBER(7,2)	Yes	(null)	2	(null)
3	ROOM_CHARGE	NUMBER(7,2)	Yes	(null)	3	(null)
4	TREATMENT_CHARGE	NUMBER(7,2)	Yes	(null)	4	(null)

Address:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ADDRESS_ID	NUMBER	No	"SQL423"."ISEQ$$_584282".nextval	1	(null)
2	STREET	VARCHAR2(100 BYTE)	Yes	(null)	2	(null)
3	CITY	VARCHAR2(100 BYTE)	Yes	(null)	3	(null)
4	DISTRICT	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)
5	REGION	VARCHAR2(5 BYTE)	Yes	(null)	5	(null)
6	POSTCODE	NUMBER(38,0)	Yes	(null)	6	(null)
7	IN_USE	VARCHAR2(3 BYTE)	Yes	(null)	7	(null)

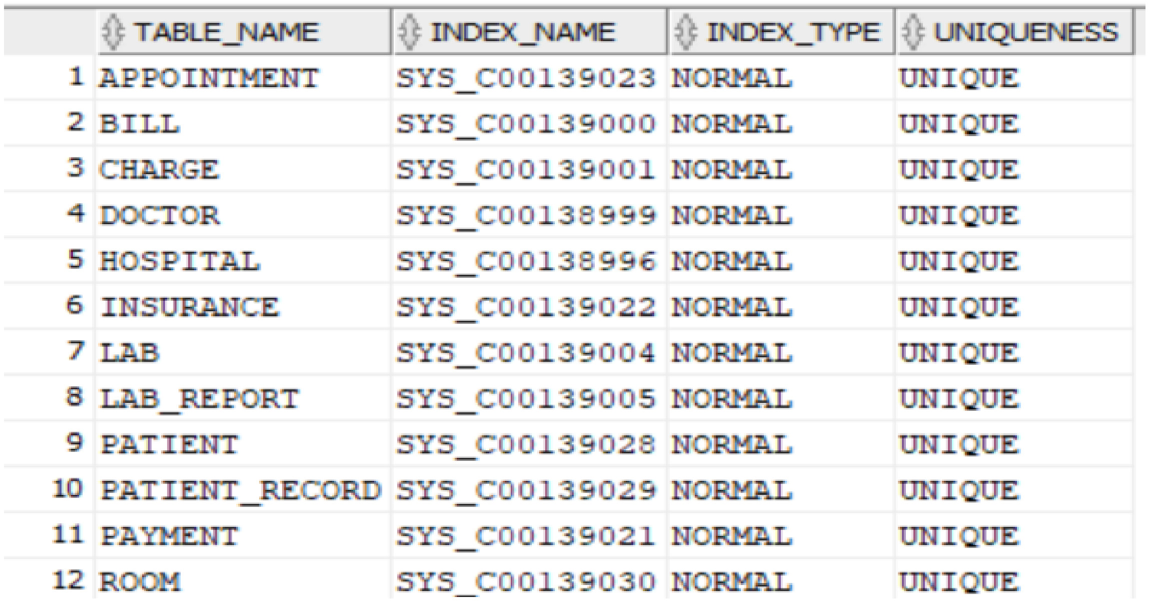
Physical Design

Indexing-

While creation of database, we created primary key for each table, and they were used as indexes for each table in the database. These primary keys are listed below.

Display Indexes when database was initially created (Primary Keys are used as indexes)

```
SELECT table_name,
       index_name,
       index_type,
       uniqueness
FROM   user_indexes
ORDER  BY table_name;
```



	TABLE_NAME	INDEX_NAME	INDEX_TYPE	UNIQUENESS
1	APPOINTMENT	SYS_C00139023	NORMAL	UNIQUE
2	BILL	SYS_C00139000	NORMAL	UNIQUE
3	CHARGE	SYS_C00139001	NORMAL	UNIQUE
4	DOCTOR	SYS_C00138999	NORMAL	UNIQUE
5	HOSPITAL	SYS_C00138996	NORMAL	UNIQUE
6	INSURANCE	SYS_C00139022	NORMAL	UNIQUE
7	LAB	SYS_C00139004	NORMAL	UNIQUE
8	LAB_REPORT	SYS_C00139005	NORMAL	UNIQUE
9	PATIENT	SYS_C00139028	NORMAL	UNIQUE
10	PATIENT_RECORD	SYS_C00139029	NORMAL	UNIQUE
11	PAYMENT	SYS_C00139021	NORMAL	UNIQUE
12	ROOM	SYS_C00139030	NORMAL	UNIQUE

Data Generation and Loading

Created stored Procedure to insert data into patient table

```
CREATE
OR      replace PROCEDURE insert_patient_data IS lcntr number;

p_name    varchar2(100);add_id    number;p_gender varchar2(3);n_id      number;p_i
d      number;BEGIN
  for lcntr IN 1..3000
  loop
  SELECT firstname
                ||' '
                || lastname ,
         gender,
         name_id
  INTO   p_name ,
         p_gender,
         n_id
  FROM   names
  WHERE  in_use = 'N'
  FETCH next 1 rows only;

  select patient_seq.nextval
  INTO   p_id
  FROM   dual;

  select address_id
  INTO   add_id
  FROM   addresses
  WHERE  in_use = 'N'
  FETCH next 1 rows only;

  dbms_output.put_line('Creating Patient '|| p_id ||' NAME : ' || p_name || '
GENDER= ' || p_gender || ' ADDRESS = ' || add_id);
  insert INTO patient
              (
                          patient_id,
                          patient_name,
                          gender,
                          address_id
              )
              VALUES
              (
                          p_id,
                          p_name,
                          p_gender,
                          add_id
              );

  update addresses
  SET    in_use = 'Y'
  WHERE  address_id = add_id;

  update names
  SET    in_use = 'Y'
  WHERE  name_id = n_id;

endLOOP;EXCEPTION
WHEN others THEN
  raise_application_error(-20001,'An error was encountered - '||sqlcode||' -
ERROR- '||sqlerrm);END;
```

Updating Patient Details Table:

Updating contact Number:

```
SQL>UPDATE patient
SET    contact_number = (Decode(Round(dbms_random.value(1, 6)), 1, '(813) '
              || To_char(Round(dbms_random.value(345, 991)))
              || '-'
              || To_char(Round(dbms_random.value(1111, 9999))), 2, '(727) '
              || To_char(Round(dbms_random.value(254, 347)))
              || '-'
              || To_char(Round(dbms_random.value(1111, 9999))), 3, '(212) '
              || To_char(Round(dbms_random.value(222, 999)))
              || '-'
              || To_char(Round(dbms_random.value(1111, 9999))), 4, '(202) '
              || To_char(Round(dbms_random.value(333, 888)))
              || '-'
              || To_char(Round(dbms_random.value(1111, 9999))), 5, '(617) '
              || To_char(Round(dbms_random.value(444, 777)))
              || '-'
              || To_char(Round(dbms_random.value(1111, 9999))), '(813) '
              || To_char(Round(dbms_random.value(345, 991)))
              || '-'
              || To_char(Round(dbms_random.value(1111, 9999))) ));
```

Updating date of Birth

```
SQL>
UPDATE patient
SET    dob = to_date(decode(round(dbms_random.value(1, 12)),
                           1, round(dbms_random.value(1, 31))
                                 || '-JAN-'
                                 || round(dbms_random.value(1975, 2021)),
                           2, round(dbms_random.value(1, 28))
                                 || '-FEB-'
                                 || round(dbms_random.value(1975, 2021)),
                           3, round(dbms_random.value(1, 31))
                                 || '-MAR-'
                                 || round(dbms_random.value(1975, 2021)),
                           4, round(dbms_random.value(1, 30))
                                 || '-APR-'
                                 || round(dbms_random.value(1975, 2021)),
                           5, round(dbms_random.value(1, 31))
                                 || '-MAY-'
                                 || round(dbms_random.value(1975, 2021)),
                           6, round(dbms_random.value(1, 30))
                                 || '-JUN-'
                                 || round(dbms_random.value(1975, 2021)),
                           7, round(dbms_random.value(1, 31))
                                 || '-JUL-'
                                 || round(dbms_random.value(1975, 2021)),
                           8, round(dbms_random.value(1, 31))
                                 || '-AUG-'
                                 || round(dbms_random.value(1975, 2021)),
                           9, round(dbms_random.value(1, 30))
                                 || '-SEP-'
                                 || round(dbms_random.value(1975, 2021)),
                           10, round(dbms_random.value(1, 31))
                                 || '-OCT-'
                                 || round(dbms_random.value(1975, 2021)),
                           11, round(dbms_random.value(1, 30))
                                 || '-NOV-'
                                 || round(dbms_random.value(1975, 2021)),
                           12, round(dbms_random.value(1, 31))
                                 || '-DEC-'
                      || round(dbms_random.value(1975, 2021))), 'DD-MON-YYYY');
```

PL/SQL block for creating PATIENT _REPORT

```
DECLARE
    p_id        NUMBER;
    d_id        NUMBER;
    r_id        NUMBER;
    p_illness   VARCHAR2(100);
    p_treatment VARCHAR2(120);
    t_date      DATE;
    CURSOR c1 IS
      SELECT patient_id,
             dob
      FROM   patient sample(60)
      WHERE  ROWNUM < 100;
BEGIN
    FOR lcntr IN 1..10 LOOP
        FOR data IN c1 LOOP
            p_id := data.patient_id;

            t_date := data.dob;

            SELECT doctor_id
            INTO   d_id
            FROM   doctor sample(20)
            WHERE  ROWNUM = 1;

            SELECT room_no
            INTO   r_id
            FROM   room sample(20)
            WHERE  ROWNUM = 1;

            SELECT d_name,
                   treatment
            INTO   p_illness, p_treatment
            FROM   disease sample(20)
            WHERE  ROWNUM = 1;

            INSERT INTO patient_record
                        (record_id, patient_id diagnosis,treatment,doctor_id,
                         room_id, treatment_date)


            VALUES      (patient_record_seq.NEXTVAL,
                         p_id,
                         p_illness,
                         p_treatment,
                         d_id,
                         r_id,
                         t_date);
        END LOOP;
    END LOOP;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
      Raise_application_error(-20001, 'An error was encountered - '
                                      ||SQLCODE
                                      ||' -ERROR- '
                                      ||SQLERRM);
END;
```

PROCEDURE TO INSERT LAB DATA :

```
DECLARE
    p_id        NUMBER;
    d_id        NUMBER;
    r_id        NUMBER;
    p_illness   VARCHAR2(100);
    p_treatment VARCHAR2(120);
    t_date      DATE;
    CURSOR c1 IS
      SELECT patient_id,
             dob
      FROM   patient sample(60)
      WHERE  ROWNUM < 100;
BEGIN
    FOR lcntr IN 1..10 LOOP
        FOR data IN c1 LOOP
            p_id := data.patient_id;

            t_date := data.dob;

            SELECT doctor_id
            INTO   d_id
            FROM   doctor sample(20)
            WHERE  ROWNUM = 1;

            SELECT room_no
            INTO   r_id
            FROM   room sample(20)
            WHERE  ROWNUM = 1;

            SELECT d_name,
                   treatment
            INTO   p_illness, p_treatment
            FROM   disease sample(20)
            WHERE  ROWNUM = 1;

            INSERT INTO patient_record
                        (record_id,
                         patient_id,
                         diagnosis,
                         treatment,
                         doctor_id,
                         room_id,
                         treatment_date)
            VALUES      (patient_record_seq.NEXTVAL,
                         p_id,
                         p_illness,
                         p_treatment,
                         d_id,
                         r_id,
                         t_date);
        END LOOP;
    END LOOP;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
      Raise_application_error(-20001, 'An error was encountered - '
                                      ||SQLCODE
                                      ||' -ERROR- '
                                      ||SQLERRM);
END;
```

PROCEDURE TO INSERT INSURANCE DATA :

```
DECLARE
    p_id        NUMBER;
    d_id        NUMBER;
    prov        VARCHAR2(100);
    valid_till  VARCHAR2(120);
    ded         NUMBER;
    cover_per   NUMBER;
    r_value     NUMBER;
    CURSOR c1 IS
      SELECT patient_id,
             dob
      FROM   patient
      WHERE  patient_id NOT IN (SELECT patient_id
                                FROM   insurance)
             AND ROWNUM < 200;
BEGIN
    FOR lcntr IN 1..10 LOOP
        FOR data IN c1 LOOP
            p_id := data.patient_id;

            INSERT INTO insurance
                        (insurance_no,
                         patient_id,
                         validity)
            VALUES      ( insurance_seq.NEXTVAL,
                              p_id,
                              SYSDATE + lcntr * 20 );
        END LOOP;
    END LOOP;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
      Raise_application_error(-20001, 'An error was encountered - '
                                      ||SQLCODE
                                      ||' -ERROR- '
                                      ||SQLERRM);
END;
```

PROCEDURE TO INSERT APPOINTMENTS:

```
DECLARE
    p_id    NUMBER;
    d_id    NUMBER;
    h_id    NUMBER;
    t_date  DATE;
    r_value NUMBER;
    ins_no  NUMBER;
    CURSOR c1 IS
      SELECT patient_id,
             dob
      FROM   patient sample(60)
      WHERE  ROWNUM < 100;
BEGIN
    ins_no := 1;

    FOR lcntr IN 1..10 LOOP
        FOR data IN c1 LOOP
            p_id := data.patient_id;

            t_date := data.dob;

            r_value := Round(dbms_random.Value(50, 347));

            SELECT doctor_id
            INTO   d_id
            FROM   doctor sample(20)
            WHERE  ROWNUM = 1;

            SELECT hospital_id
            INTO   h_id
            FROM   hospital sample(20)
            WHERE  ROWNUM = 1;

            SELECT insurance_no
            INTO   ins_no
            FROM   insurance
            WHERE  patient_id = p_id;

            INSERT INTO appointment
                        (appointment_id,
                         appointment_date,
                         patient_id,
                         doctor_id,
                         hospital_id,
                         insurance_no)
            VALUES      (appointment_seq.NEXTVAL,
                         SYSDATE - r_value,
                         p_id,
                         d_id,
                         h_id,
                         ins_no);
        END LOOP;

        COMMIT;
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
      Raise_application_error(-20001, 'An error was encountered - '
                                      ||SQLCODE
                                      ||' -ERROR- '
                                      ||SQLERRM);
END;
```

Creating appointment table-

```
DECLARE
    p_id     NUMBER;
    d_id     NUMBER;
    h_id     NUMBER;
    t_date   DATE;
    r_value NUMBER;
    ins_no   NUMBER;
    CURSOR c1 IS
      SELECT patient_id,
             dob
      FROM   patient sample(60)
      WHERE  ROWNUM < 100;
BEGIN
    ins_no := 1;
    FOR lcntr IN 1..10 LOOP
        FOR data IN c1 LOOP
            p_id := data.patient_id;

            t_date := data.dob;

            r_value := Round(dbms_random.Value(50, 347));

            SELECT doctor_id
            INTO   d_id
            FROM   doctor sample(20)
            WHERE  ROWNUM = 1;

            SELECT hospital_id
            INTO   h_id
            FROM   hospital sample(20)
            WHERE  ROWNUM = 1;

            SELECT insurance_no
            INTO   ins_no
            FROM   insurance
            WHERE  patient_id = p_id;

            INSERT INTO appointment
                        (appointment_id,
                         appointment_date,
                         patient_id,
                         doctor_id,
                         hospital_id,
                         insurance_no)
            VALUES      (appointment_seq.NEXTVAL,
                         SYSDATE - r_value,
                         p_id,
                         d_id,
                         h_id,
                         ins_no);
        END LOOP;

        COMMIT;
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
      Raise_application_error(-20001, 'An error was encountered - '
                                      ||SQLCODE
                                      ||' -ERROR- '
                                      ||SQLERRM);
END;
```

Generating Foreign key constraints-

```
ALTER TABLE patient_record
  ADD CONSTRAINT patient_fk1 FOREIGN KEY (patient_id) REFERENCES patient(
  patient_id);

ALTER TABLE patient_record
  ADD CONSTRAINT doctor_fk1 FOREIGN KEY (doctor_id) REFERENCES doctor(doctor_id
)
;

ALTER TABLE patient_record
  ADD CONSTRAINT room_fk FOREIGN KEY (room_id) REFERENCES room(room_id);

ALTER TABLE bill
  ADD CONSTRAINT charge_fk FOREIGN KEY (charge_id) REFERENCES charge(charge_id)
;

ALTER TABLE bill
  ADD CONSTRAINT appointment_fk FOREIGN KEY (appointment_id) REFERENCES
  appointment(appointment_id);

ALTER TABLE bill
  ADD CONSTRAINT patient_fk2 FOREIGN KEY (patient_id) REFERENCES patient(
  patient_id);

ALTER TABLE lab
  ADD CONSTRAINT hospital_fk1 FOREIGN KEY (hospital_id) REFERENCES hospital(
  hospital_id);

ALTER TABLE lab_report
  ADD CONSTRAINT lab_fk FOREIGN KEY (lab_id) REFERENCES lab(lab_id);

ALTER TABLE lab_report
  ADD CONSTRAINT patient_fk3 FOREIGN KEY (patient_id) REFERENCES patient(
  patient_id);

ALTER TABLE payment
  ADD CONSTRAINT bill_fk FOREIGN KEY (bill_id) REFERENCES bill(bill_id);

ALTER TABLE payment
  ADD CONSTRAINT patient_fk4 FOREIGN KEY (patient_id) REFERENCES patient(
  patient_id);

ALTER TABLE insurance
  ADD CONSTRAINT patient_fk5 FOREIGN KEY (patient_id) REFERENCES patient(
  patient_id);
```

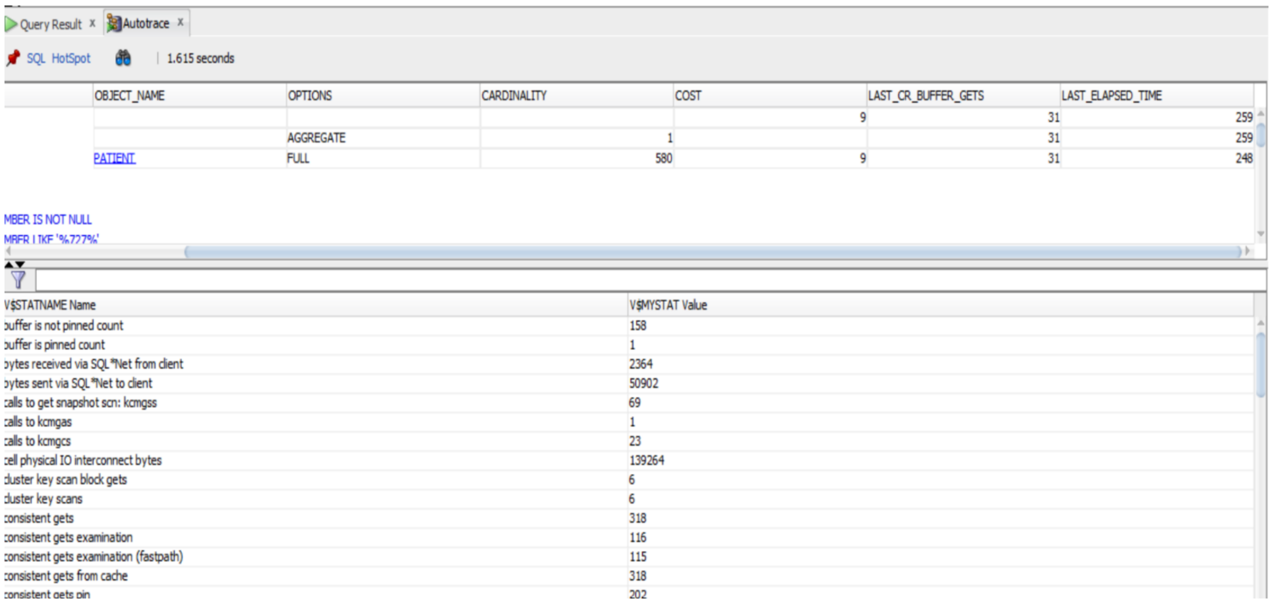
Performance Tuning

Indexing-

No index (data with high cardinality)-

```
SELECT Count(*)
FROM   patient
WHERE  contact_number LIKE '%727%';
```

Count (*) 580



Query Result | Autotrace

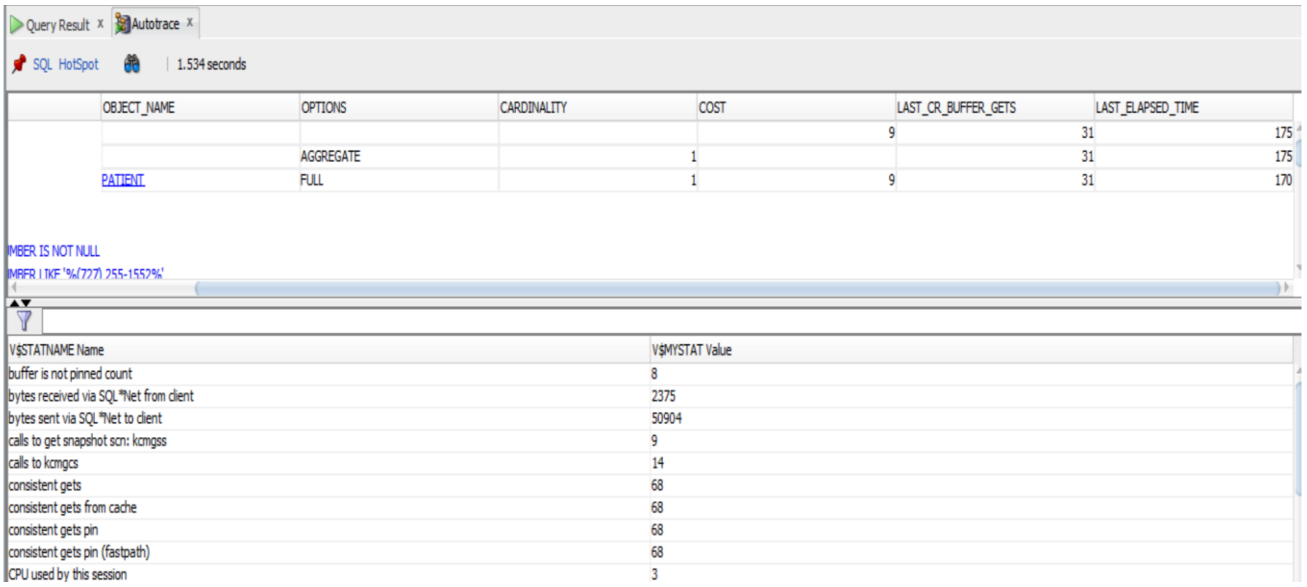
SQL HotSpot | 1.615 seconds

OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
			9	31	259
	AGGREGATE	1		31	259
PATIENT	FULL	580	9	31	248

```
MBER IS NOT NULL
MBER LIKE '%727%'
```

V$STATNAME Name	V$MYSTAT Value
buffer is not pinned count	158
buffer is pinned count	1
bytes received via SQL*Net from client	2364
bytes sent via SQL*Net to client	50902
calls to get snapshot scn: kcmgss	69
calls to kcmgas	1
calls to kcmgcs	23
cell physical IO interconnect bytes	139264
cluster key scan block gets	6
cluster key scans	6
consistent gets	318
consistent gets examination	116
consistent gets examination (fastpath)	115
consistent gets from cache	318
consistent gets pin	202

```
SELECT Count(*)
FROM   patient
WHERE  contact_number LIKE '%(727) 255-1552%';
```
Count (*) 1


Query Result × | Autotrace ×

SQL HotSpot | 1.534 seconds

	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
				9	31	175
		AGGREGATE	1		31	175
	PATIENT	FULL	1	9	31	170

```
MBER IS NOT NULL
MBER LIKE '%(727) 255-1552%'
```

V$STATNAME Name	V$MYSTAT Value
buffer is not pinned count	8
bytes received via SQL*Net from client	2375
bytes sent via SQL*Net to client	50904
calls to get snapshot scn: kcmgss	9
calls to kcmgcs	14
consistent gets	68
consistent gets from cache	68
consistent gets pin	68
consistent gets pin (fastpath)	68
CPU used by this session	3

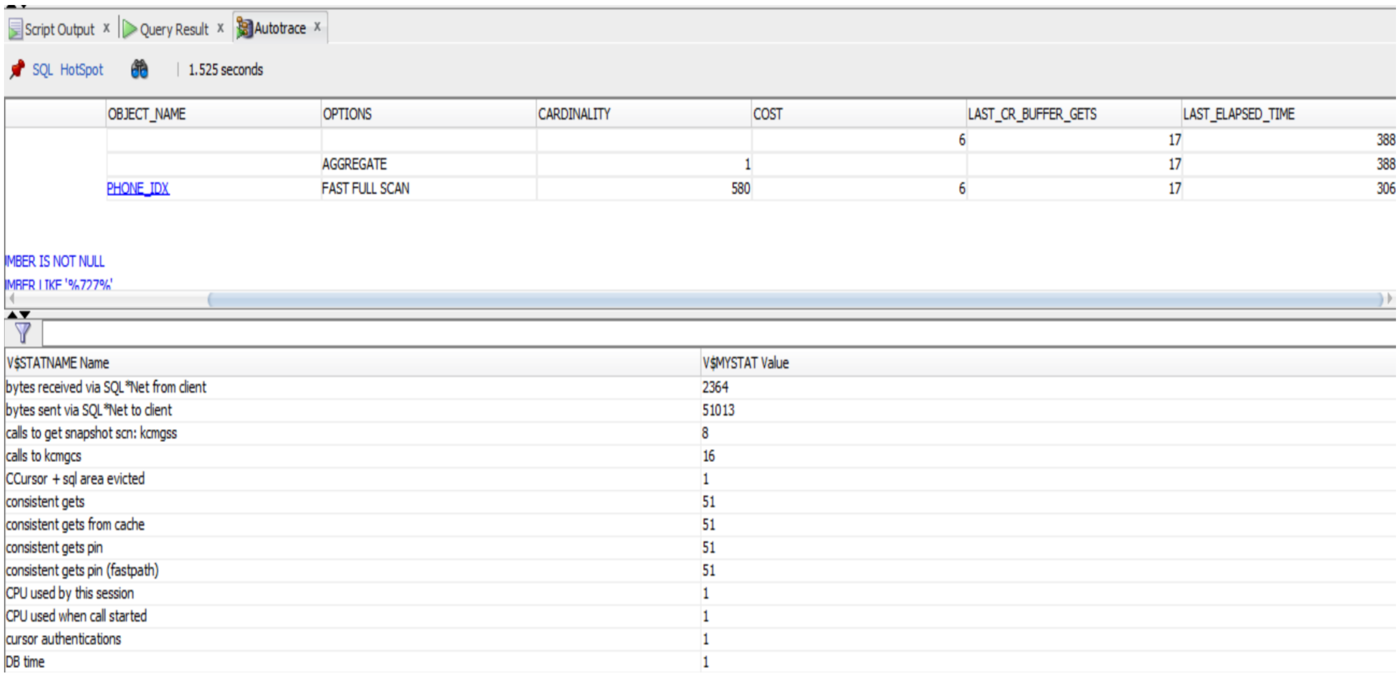
B-tree

index (for data with high cardinality)-

```
CREATE INDEX phone_idx
  ON patient(contact_number);

SELECT Count(*)
FROM   patient
WHERE  contact_number LIKE '%727%';
```

Count (*) 580


Script Output × | Query Result × | Autotrace ×

SQL HotSpot | 1.525 seconds

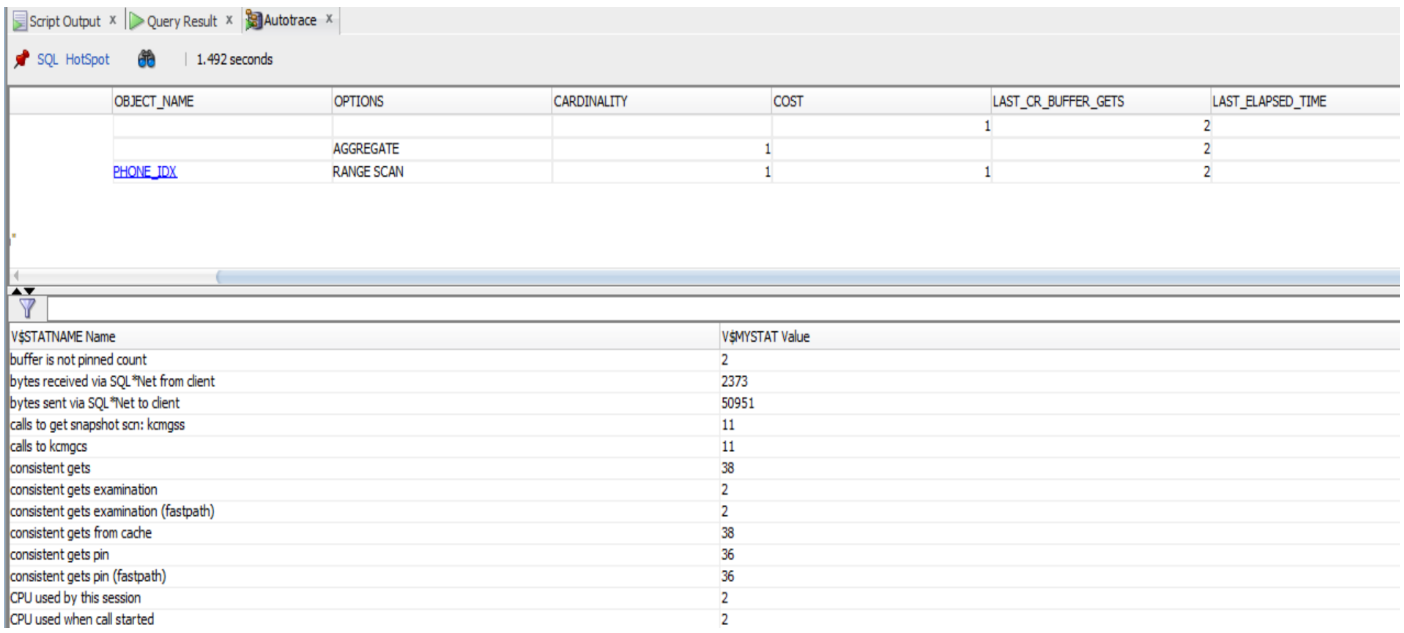
	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
				6	17	388
		AGGREGATE	1		17	388
	PHONE_IDX	FAST FULL SCAN	580	6	17	306

```
MBER IS NOT NULL
MBER LIKE '%727%'
```

V$STATNAME Name	V$MYSTAT Value
bytes received via SQL*Net from client	2364
bytes sent via SQL*Net to client	51013
calls to get snapshot scn: kcmgss	8
calls to kcmgcs	16
CCursor + sql area evicted	1
consistent gets	51
consistent gets from cache	51
consistent gets pin	51
consistent gets pin (fastpath)	51
CPU used by this session	1
CPU used when call started	1
cursor authentications	1
DB time	1

```
SELECT Count(*)
FROM   patient
WHERE  contact_number LIKE '(727) 255-1552';
```

Count (*) 1



Script Output × | Query Result × | Autotrace ×

SQL HotSpot | 1.492 seconds

OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
			1	2	
	AGGREGATE	1		2	
PHONE_IDX	RANGE SCAN	1	1	2	

V$STATNAME Name	V$MYSTAT Value
buffer is not pinned count	2
bytes received via SQL*Net from client	2373
bytes sent via SQL*Net to client	50951
calls to get snapshot scn: kcmgss	11
calls to kcmgcs	11
consistent gets	38
consistent gets examination	2
consistent gets examination (fastpath)	2
consistent gets from cache	38
consistent gets pin	36
consistent gets pin (fastpath)	36
CPU used by this session	2
CPU used when call started	2

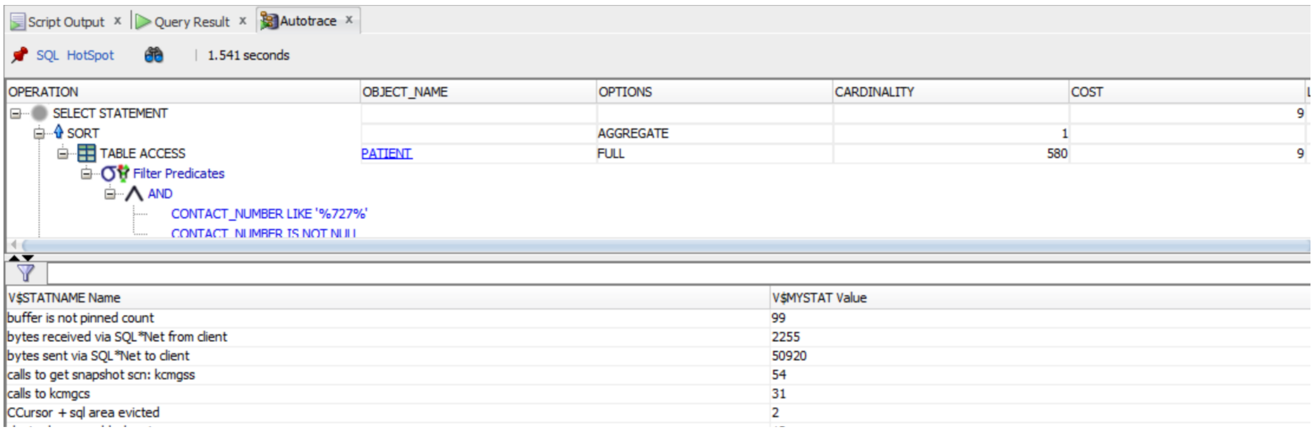
Bitmap Index (Data with high cardinality)-

```
DROP INDEX phone_idx;

CREATE bitmap INDEX phone_bmap
  ON patient(contact_number);

SELECT Count(*)
FROM   patient
WHERE  contact_number LIKE '%727%';
```

Count (*) 580



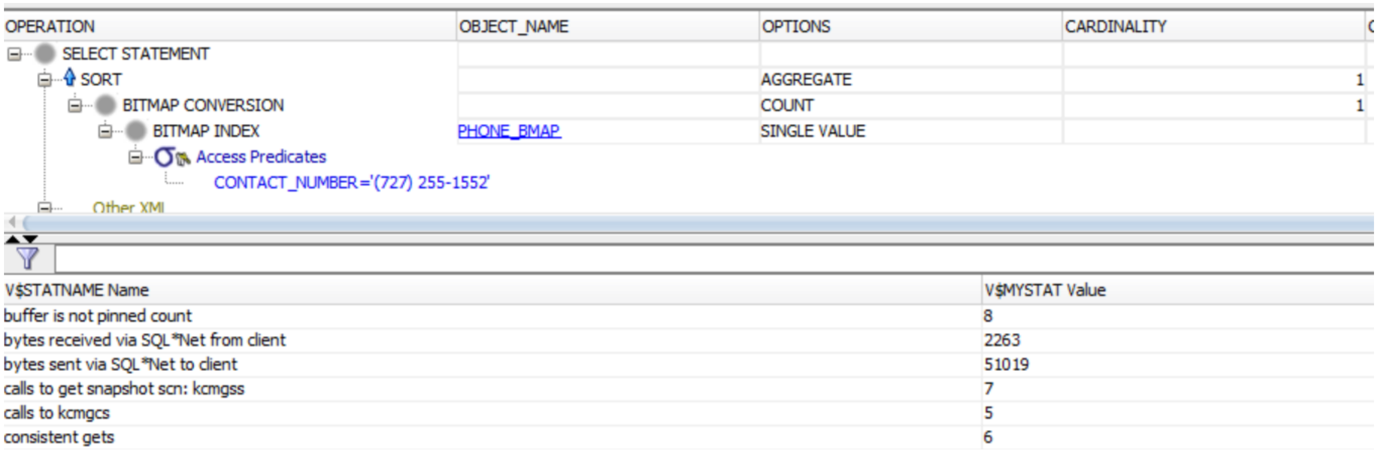
Script Output × | Query Result × | Autotrace ×

SQL HotSpot | 1.541 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				9
SORT		AGGREGATE	1	
TABLE ACCESS	PATIENT	FULL	580	9
Filter Predicates				
AND				
CONTACT_NUMBER LIKE '%727%'				
CONTACT_NUMBER IS NOT NULL				

V$STATNAME Name	V$MYSTAT Value
buffer is not pinned count	99
bytes received via SQL*Net from client	2255
bytes sent via SQL*Net to client	50920
calls to get snapshot scn: kcmgss	54
calls to kcmgcs	31
CCursor + sql area evicted	2

```sql
SELECT Count(*)
FROM   patient
WHERE  contact_number LIKE '(727) 255-1552';
```



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY
SELECT STATEMENT			
SORT		AGGREGATE	1
BITMAP CONVERSION		COUNT	1
BITMAP INDEX	PHONE_BMAP	SINGLE VALUE	
Access Predicates			
CONTACT_NUMBER='(727) 255-1552'			
Other XML			

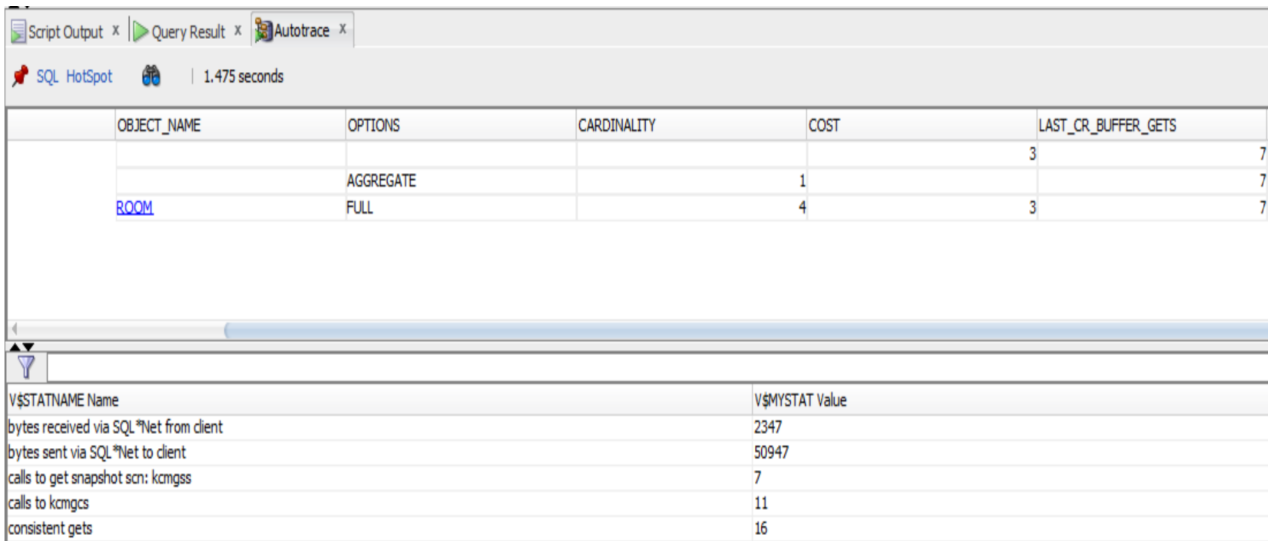
V$STATNAME Name	V$MYSTAT Value
buffer is not pinned count	8
bytes received via SQL*Net from client	2263
bytes sent via SQL*Net to client	51019
calls to get snapshot scn: kcmgss	7
calls to kcmgcs	5
consistent gets	6

The check was performed on table with high-cardinality attributes and B-Tree index performed better than Bitmap. Behavior of these two indexing was tested against point and range scan. B-Tree index and Bitmap index both helped minimize the cost of the point scan query. Now focusing on the range scan, B-Tree index dramatically decreased the consistent gets from 31 to 17 and the cost from 9 to 6.

No-Index (Data with low cardinality)-

```sql
SELECT Count(*)
FROM   room
WHERE  hospital_id = 1;
```

Count (*) 4

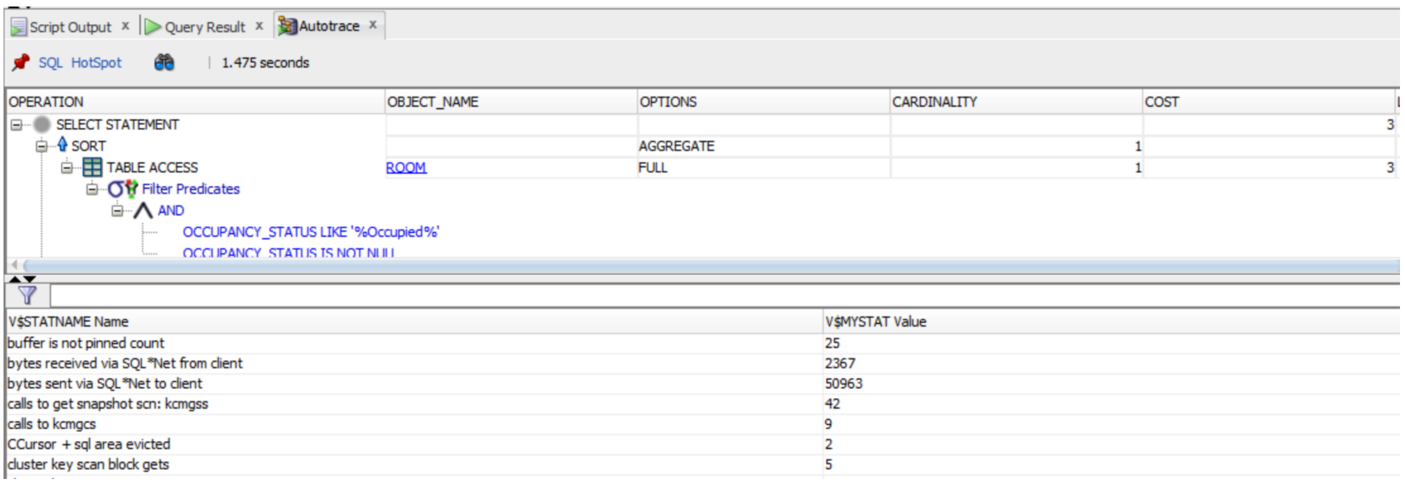


OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS
			3	7
	AGGREGATE	1		7
ROOM	FULL	4	3	7

V$STATNAME Name	V$MYSTAT Value
bytes received via SQL*Net from client	2347
bytes sent via SQL*Net to client	50947
calls to get snapshot scn: kcmgss	7
calls to kcmgcs	11
consistent gets	16

```sql
SELECT Count(*)
FROM   room
WHERE  occupancy_status LIKE '%Occ%';
```

Count (*) 17



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3
SORT		AGGREGATE	1	
TABLE ACCESS	ROOM	FULL	1	3
Filter Predicates				
AND				
OCCUPANCY_STATUS LIKE '%Occupied%'				
OCCUPANCY_STATUS IS NOT NULL				

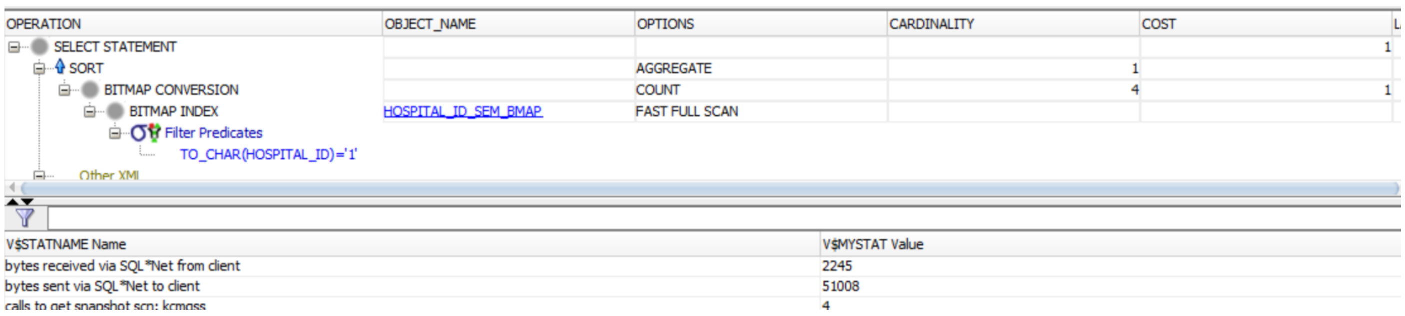
V$STATNAME Name	V$MYSTAT Value
buffer is not pinned count	25
bytes received via SQL*Net from client	2367
bytes sent via SQL*Net to client	50963
calls to get snapshot scn: kcmgss	42
calls to kcmgcs	9
CCursor + sql area evicted	2
cluster key scan block gets	5

B-Tree Index (Data with low cardinality)-

```sql
CREATE INDEX hospital_id_idx
   ON room(hospital_id);

SELECT Count(*)
FROM   room
WHERE  hospital_id LIKE '1';
```

Count (*) 4



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
SORT		AGGREGATE	1	
BITMAP CONVERSION		COUNT	4	1
BITMAP INDEX	HOSPITAL_ID_SEM_BMAP	FAST FULL SCAN		
Filter Predicates				
TO_CHAR(HOSPITAL_ID)='1'				
Other XML				

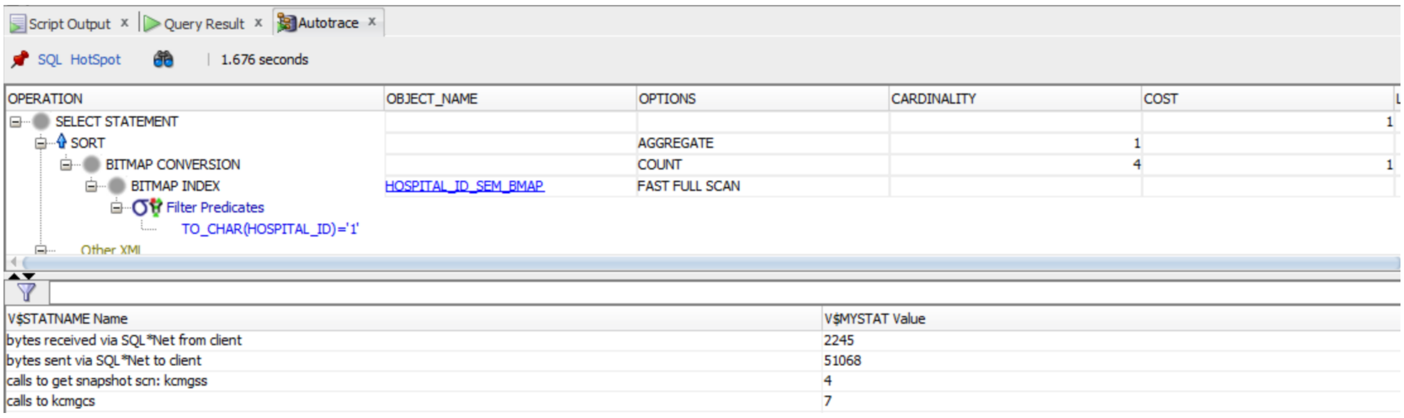
V$STATNAME Name	V$MYSTAT Value
bytes received via SQL*Net from client	2245
bytes sent via SQL*Net to client	51008
calls to get snapshot scn: kcmgss	4

Bitmap Index (Data with low cardinality)-

```
DROP INDEX hospital_id_idx;

CREATE bitmap INDEX hospital_id_sem_bmap
  ON room(hospital_id);
```

Count (*) 4


Script Output × | Query Result × | Autotrace ×

SQL HotSpot | 1.676 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
SORT		AGGREGATE	1	
BITMAP CONVERSION		COUNT	4	1
BITMAP INDEX	HOSPITAL_ID_SEM_BMAP	FAST FULL SCAN		
Filter Predicates				
TO_CHAR(HOSPITAL_ID)='1'				
Other XML				

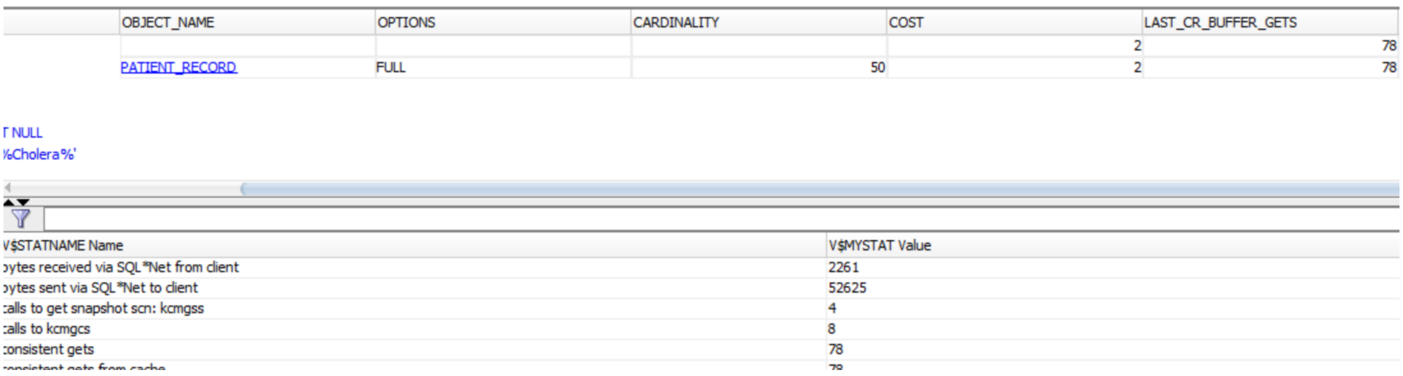
V$STATNAME Name	V$MYSTAT Value
bytes received via SQL*Net from client	2245
bytes sent via SQL*Net to client	51068
calls to get snapshot scn: kcmgss	4
calls to kcmgcs	7

After performing Bitmap and B-tree index on the high and low cardinality data, it was observed that the cost and cardinality was reduced.

<u>Partitioning</u>

Before Partitioning

```
SELECT *
FROM   patient_record
WHERE  diagnosis LIKE '%Cholera%';
```



OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS
			2	78
PATIENT_RECORD	FULL	50	2	78

T NULL
%Cholera%'

V$STATNAME Name	V$MYSTAT Value
bytes received via SQL*Net from client	2261
bytes sent via SQL*Net to client	52625
calls to get snapshot scn: kcmgss	4
calls to kcmgcs	8
consistent gets	78
consistent gets from cache	78

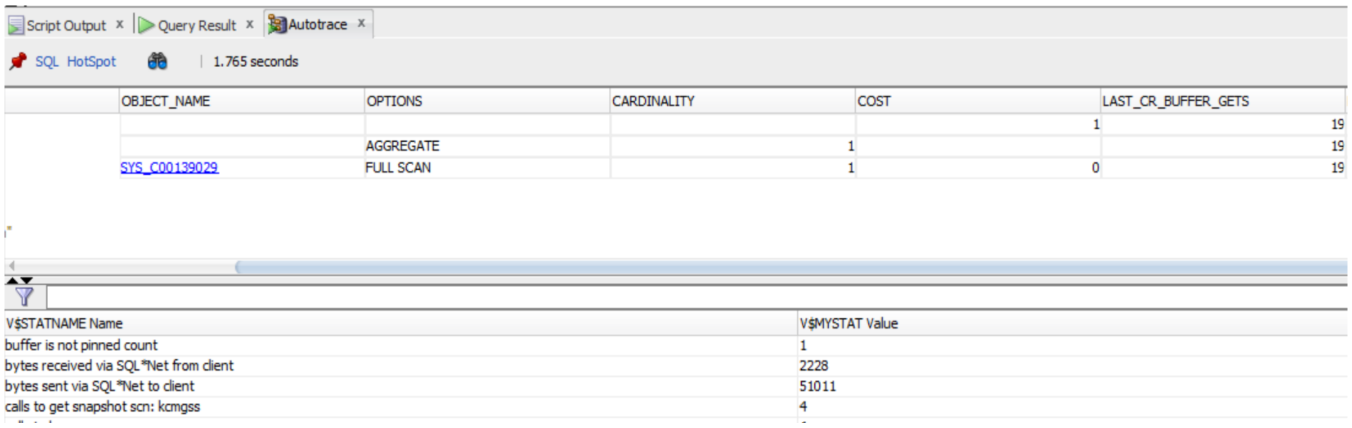
After list partitioning-

```
      CREATE TABLE patient_record_by_diagnosis
   (
     record_id      NUMBER PRIMARY KEY,
     patient_id     NUMBER,
     diagnosis      VARCHAR2(100),
     treatment      VARCHAR2(100),
     doctor_id      NUMBER,
     room_id        NUMBER,
     treatment_date DATE
   ) partition BY list(diagnosis) (PARTITION dia_cho VALUES ('Cholera'),
PARTITION dia_chl VALUES ('Chlamydia'), PARTITION dia_pst VALUES ('Psittacosis'
)
, PARTITION dia_rb VALUES ('Rabies'), PARTITION dia_pa VALUES ('Paralytic'),
PARTITION dia_lis VALUES ('Listeriosis'), PARTITION dia_dip VALUES ('Diphtheria
'
), PARTITION dia_tb VALUES ('Tuberculosis'), PARTITION dia_bot VALUES (
'Botulism'), PARTITION dia_var VALUES ('Varicella'), PARTITION dia_rub VALUES (
'Rubella'), PARTITION dia_bab VALUES ('Babesiosis'), PARTITION dia_hep VALUES (
'Hepatitis A'), PARTITION dia_per VALUES ('Pertussis'), PARTITION dia_lep VALUE
S
('Leprosy'), PARTITION dia_syp VALUES ('Early Syphilis'));




>
INSERT INTO patient_record_by_diagnosis
            (
                        record_id,
                        patient_id ,
                        diagnosis ,
                        treatment ,
                        doctor_id,
                        room_id,
                        treatment_date
            )
SELECT record_id,
       patient_id ,
       diagnosis ,
       treatment ,
       doctor_id,
       room_id,
       treatment_date
FROM   patient_record;>


SELECT *
FROM   patient_record_by_diagnosis
WHERE  diagnosis LIKE '%Cholera%';
```



OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS
			1	19
	AGGREGATE	1		19
SYS_C00139029	FULL SCAN	1	0	19

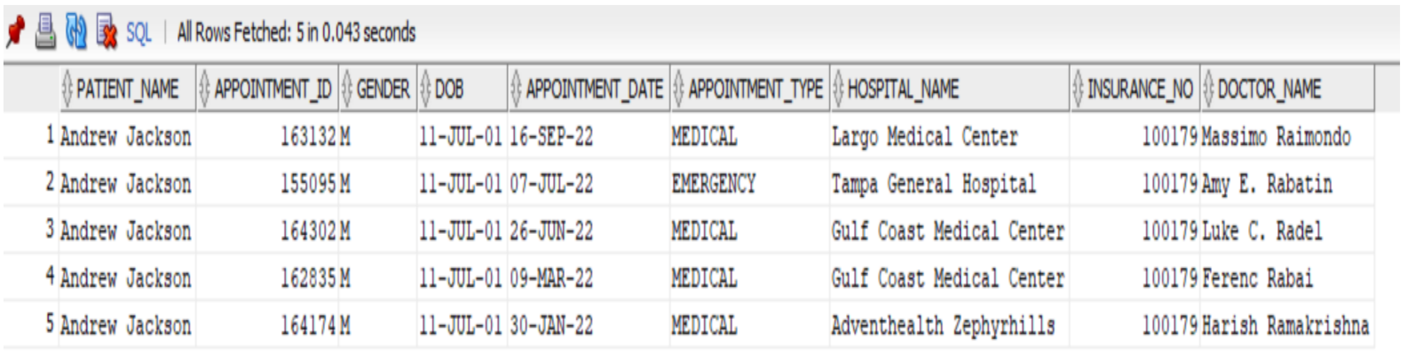
V$STATNAME Name	V$MYSTAT Value
buffer is not pinned count	1
bytes received via SQL*Net from client	2228
bytes sent via SQL*Net to client	51011
calls to get snapshot scn: kcmgss	4

It is observed that the cost of the operation as well as the consistent gets lower once we apply partitioning on the table. The cost fell from 2 to 0 and the consistent fell from 78 to 19.

Scripts

Patients appointment History-

```
SELECT P.patient_name,
       A.appointment_id,
       P.gender,
       P.dob,
       A.appointment_date,
       appointment_type,
       H.hospital_name,
       insurance_no,
       D.doctor_name
FROM   patient P
       INNER JOIN appointment A
               ON P.patient_id = A.patient_id
       JOIN hospital H
         ON A.hospital_id = H.hospital_id
       JOIN doctor D
         ON A.doctor_id = D.doctor_id
WHERE  P.patient_id = 11797
ORDER  BY A.appointment_date DESC;
```

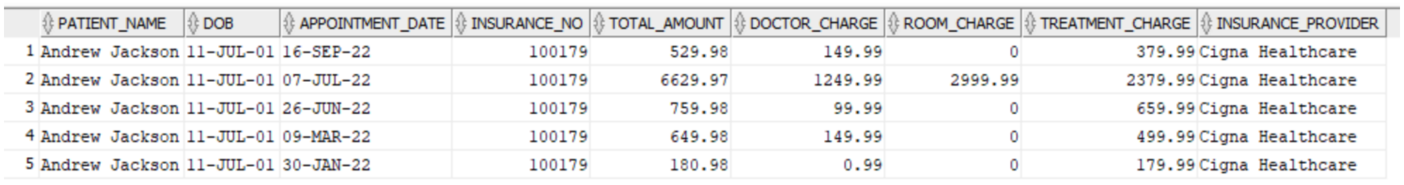


All Rows Fetched: 5 in 0.043 seconds

	PATIENT_NAME	APPOINTMENT_ID	GENDER	DOB	APPOINTMENT_DATE	APPOINTMENT_TYPE	HOSPITAL_NAME	INSURANCE_NO	DOCTOR_NAME
1	Andrew Jackson	163132	M	11-JUL-01	16-SEP-22	MEDICAL	Largo Medical Center	100179	Massimo Raimondo
2	Andrew Jackson	155095	M	11-JUL-01	07-JUL-22	EMERGENCY	Tampa General Hospital	100179	Amy E. Rabatin
3	Andrew Jackson	164302	M	11-JUL-01	26-JUN-22	MEDICAL	Gulf Coast Medical Center	100179	Luke C. Radel
4	Andrew Jackson	162835	M	11-JUL-01	09-MAR-22	MEDICAL	Gulf Coast Medical Center	100179	Ferenc Rabai
5	Andrew Jackson	164174	M	11-JUL-01	30-JAN-22	MEDICAL	Adventhealth Zephyrhills	100179	Harish Ramakrishna

Patients Billing History-

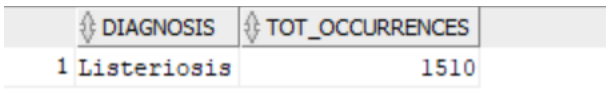
```
SELECT P.patient_name,
       P.dob,
       A.appointment_date,
       A.insurance_no,
       B.total_amount,
       C.doctor_charge,
       C.room_charge,
       C.treatment_charge,
       INS.provider AS INSURANCE_PROVIDER
FROM   patient P,
       appointment A,
       bill B,
       charge C,
       insurance INS
WHERE  P.patient_id = A.patient_id
       AND B.patient_id = P.patient_id
       AND B.appointment_id = A.appointment_id
       AND B.charge_id = C.charge_id
       AND A.insurance_no = INS.insurance_no
       AND p.patient_id = 11797;
```



	PATIENT_NAME	DOB	APPOINTMENT_DATE	INSURANCE_NO	TOTAL_AMOUNT	DOCTOR_CHARGE	ROOM_CHARGE	TREATMENT_CHARGE	INSURANCE_PROVIDER
1	Andrew Jackson	11-JUL-01	16-SEP-22	100179	529.98	149.99	0	379.99	Cigna Healthcare
2	Andrew Jackson	11-JUL-01	07-JUL-22	100179	6629.97	1249.99	2999.99	2379.99	Cigna Healthcare
3	Andrew Jackson	11-JUL-01	26-JUN-22	100179	759.98	99.99	0	659.99	Cigna Healthcare
4	Andrew Jackson	11-JUL-01	09-MAR-22	100179	649.98	149.99	0	499.99	Cigna Healthcare
5	Andrew Jackson	11-JUL-01	30-JAN-22	100179	180.98	0.99	0	179.99	Cigna Healthcare

MOST DIAGNOSED_ILLNESS

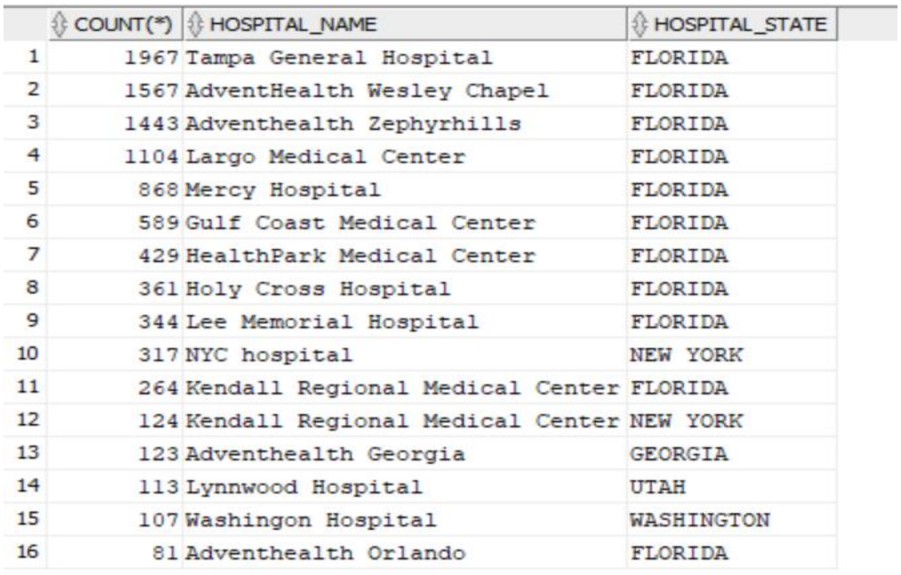
```
SELECT diagnosis,
       tot_occurrences
FROM   (SELECT diagnosis,
               COUNT(*) AS tot_occurrences
        FROM   patient_record
        GROUP  BY diagnosis
        ORDER  BY COUNT(*) desc)
FETCH first 1 ROW only;
```



	DIAGNOSIS	TOT_OCCURRENCES
1	Listeriosis	1510

NO OF PATIENTS SERVED BY HOSPITAL THIS YEAR-

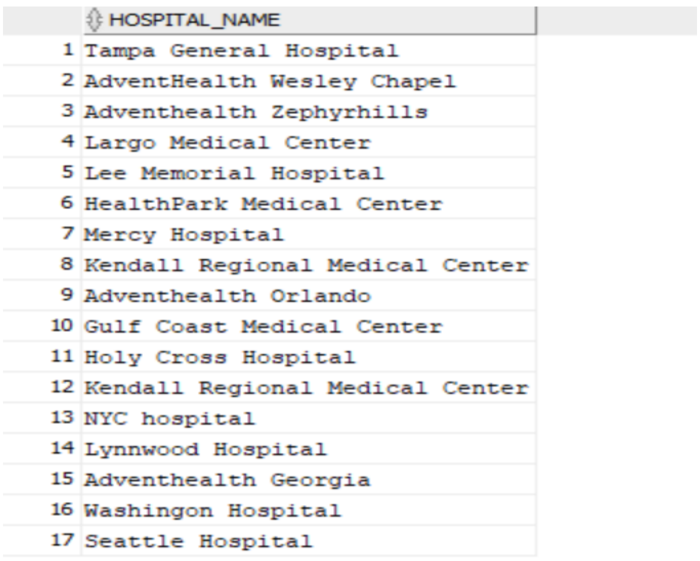
```
SELECT Count(*),
       H.hospital_name,
       H.hospital_state
FROM   appointment A,
       hospital H
WHERE  A.hospital_id = H.hospital_id
       AND A.appointment_date > To_date('01/01/22', 'DD/MM/YYYY')
GROUP  BY H.hospital_name,
          H.hospital_state
ORDER  BY Count(*) DESC
```



	COUNT(*)	HOSPITAL_NAME	HOSPITAL_STATE
1	1967	Tampa General Hospital	FLORIDA
2	1567	AdventHealth Wesley Chapel	FLORIDA
3	1443	Adventhealth Zephyrhills	FLORIDA
4	1104	Largo Medical Center	FLORIDA
5	868	Mercy Hospital	FLORIDA
6	589	Gulf Coast Medical Center	FLORIDA
7	429	HealthPark Medical Center	FLORIDA
8	361	Holy Cross Hospital	FLORIDA
9	344	Lee Memorial Hospital	FLORIDA
10	317	NYC hospital	NEW YORK
11	264	Kendall Regional Medical Center	FLORIDA
12	124	Kendall Regional Medical Center	NEW YORK
13	123	Adventhealth Georgia	GEORGIA
14	113	Lynnwood Hospital	UTAH
15	107	Washingon Hospital	WASHINGTON
16	81	Adventhealth Orlando	FLORIDA

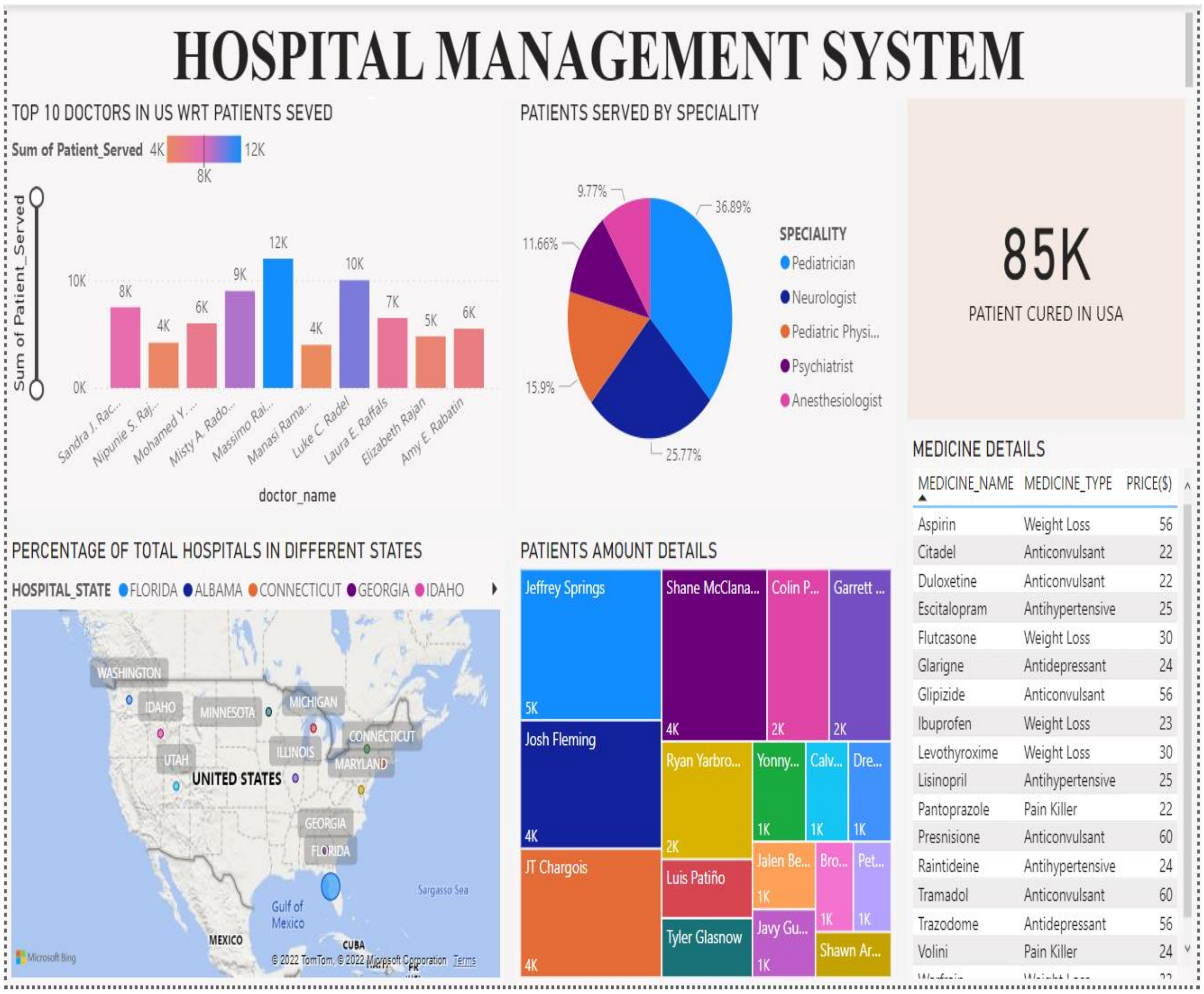
List of all the hospitals-

```
SELECT hospital_name
FROM   hospital;
```



	HOSPITAL_NAME
1	Tampa General Hospital
2	AdventHealth Wesley Chapel
3	Adventhealth Zephyrhills
4	Largo Medical Center
5	Lee Memorial Hospital
6	HealthPark Medical Center
7	Mercy Hospital
8	Kendall Regional Medical Center
9	Adventhealth Orlando
10	Gulf Coast Medical Center
11	Holy Cross Hospital
12	Kendall Regional Medical Center
13	NYC hospital
14	Lynnwood Hospital
15	Adventhealth Georgia
16	Washingon Hospital
17	Seattle Hospital

Data Visualization



MEDICINE_NAME	MEDICINE_TYPE	PRICE($)
Aspirin	Weight Loss	56
Citadel	Anticonvulsant	22
Duloxetine	Anticonvulsant	22
Escitalopram	Antihypertensive	25
Flutcasone	Weight Loss	30
Glarigne	Antidepressant	24
Glipizide	Anticonvulsant	56
Ibuprofen	Weight Loss	23
Levothyroxime	Weight Loss	30
Lisinopril	Antihypertensive	25
Pantoprazole	Pain Killer	22
Presnisione	Anticonvulsant	60
Raintideine	Antihypertensive	24
Tramadol	Anticonvulsant	60
Trazodome	Antidepressant	56
Volini	Pain Killer	24

Analysis of the Power BI Report:

1. *Top 10 Doctors in the USA with Respect to patients Served:*
   (Bar Graph)
   The graph shows which all doctors served the most patients. The blue colour which says 12k patients served by 'Massimo Raimondo' is the highest among all and 'Manasi Ramakrishnan' served 4200 patients and was least among the top 10 doctors.
   Also, the graph has a filter to its right from which we can change the visualization of the patients served.

2. *Patients Served by Specialty:*
   (Pie Chart)

This graph shows the percentage-wise patients served by different doctors' specialties. Out of which most patients are served by the Pediatrician category doctors with a total.

percentage of 36.89% and the least are served by pulmonologists which are 3.31%. This showcase that most doctors are required in the pediatrician category as most of the patients needs treatment for the same.

Patients Cured in USA:
(Card)
The top right corner card showcase the total number of people who have been cured in the USA and that number turns out to be 85K. The numbers are based on the sample data and have to be revised with actual data if the project goes live.

Number of Total Hospitals in Different States across USA:
(Map)
The bottom left visual is a map of the USA which showcases the total number of hospitals across different states. The maximum comes out to be in 'Florida' as seen by the width of the bubble in blue and the least is 'Connecticut' in Green. This can help us to directly view the hospital numbers and plan for emergencies.

Patient Amount Details:
(Tree Map)
The bottom middle graph showcases the total amount which the hospital has taken from patients and among them who were the top ones. This graph can help the hospitals to bring down the charges through which their hospitals can be highlighted and reached out by patients. Moreover, the patients can also keep an eye on what the hospital is charging them.

MEDICINE DETAILS:
(Table)
The last graph showcase the table with different medicines, the names of the medicine, different medicine types as in why it is used and also the cost of the medicine. This table is a single point of truth for the pharmacy department and can be expanded by placing the total number of medicines left in the pharmacy. This will help the pharmacist to keep the track of the medicines.

All these graphs if utilized wisely can help the hospital to manage and predict various factors like:
Which all states need more hospitals.
Which all medicines are consumed the most, ultimately can help in reverse engineering to bring the best cure for the disease for which the medicine is use.
Patients can keep an eye on the total bill given to them by the hospital.
Hospitals collaboratively can cure more patients to maximize the Card number.
Which all are the famous doctors.