ISM 6218 Advanced Database Management

XYZ Hospital Database System

GROUP 2:

1. Lakshay Mukhi
2. Poonam Krishna
3. Shreyas Dalvi
4. Yashjeet Singh

Table of Contents

USF UNIVERSITY OF SOUTH FLORIDA.

I. Summary

The project we have undertaken has the primary goal of updating the current database system employed at XYZ hospital to include more information than it holds now.

As of now, the system has information available only the hospital staff and that too not that comprehensive; we plan to make it available for patients too by integrating more details such as patient details and occupancy details for the hospital so that the patient can also track his requests.

The proposed changes include steps such as creating login information where the user can save their medical profile, thereby making it easy for the institution to track patients' data and their insurance related information. So, every new time when the patient visits the hospital, he wouldn't have to fill in the detail again thus reducing redundancy.

There are various roles created such as a patient, doctor, and nurse. The doctor can view his patient's medical records, status, allocated wards and other such information.

Steps are also being taken will also allow hospitals to trace occupancy of their rooms and operation theatres. Also, doctors can see which nurses are allocated to which shifts and which ward, thus increasing accountability. Only authorized users are permitted to make changes in the database. Also, a mechanism has been installed which lets the hospital contact the insurance provider directly saving time and increasing efficiency.

The above are just a few of the many improvements planned to optimize and the database system and increase its performance.

II. Project Requirements

The project requirements are as follows:

1. The project encompasses fourteen (14) tables which deal with a hospital system.

2. The system will face almost negligible downtime.

3. All the users will need to get login credential from a DBA.

4. Any jump in traffic will not have a major impact on the system performance.

5. Maintenance is done only during night times and that too infrequently.

6. A different network monitors this network, alerting the users via email regarding any downtime in this system.

7. RAID functionality is used for storing databases.

8. A backup server is in place which mirrors the data while the database is in active use (hot backup) while another server takes backup on a regular basis when the database is not in active use (cold backup).

III. Assumptions

1. It is not mandatory for all patients to have insurance.

2. Every patient will have an admission date and discharge date but not necessarily an operation date.

3. Bill can be paid in installments.

4. Doctors have full access to patient and nurse information.

5. Every patient will have a room and a ward allocated to him.

6. A patient will not be turned away by the hospital even for non-emergency cases.

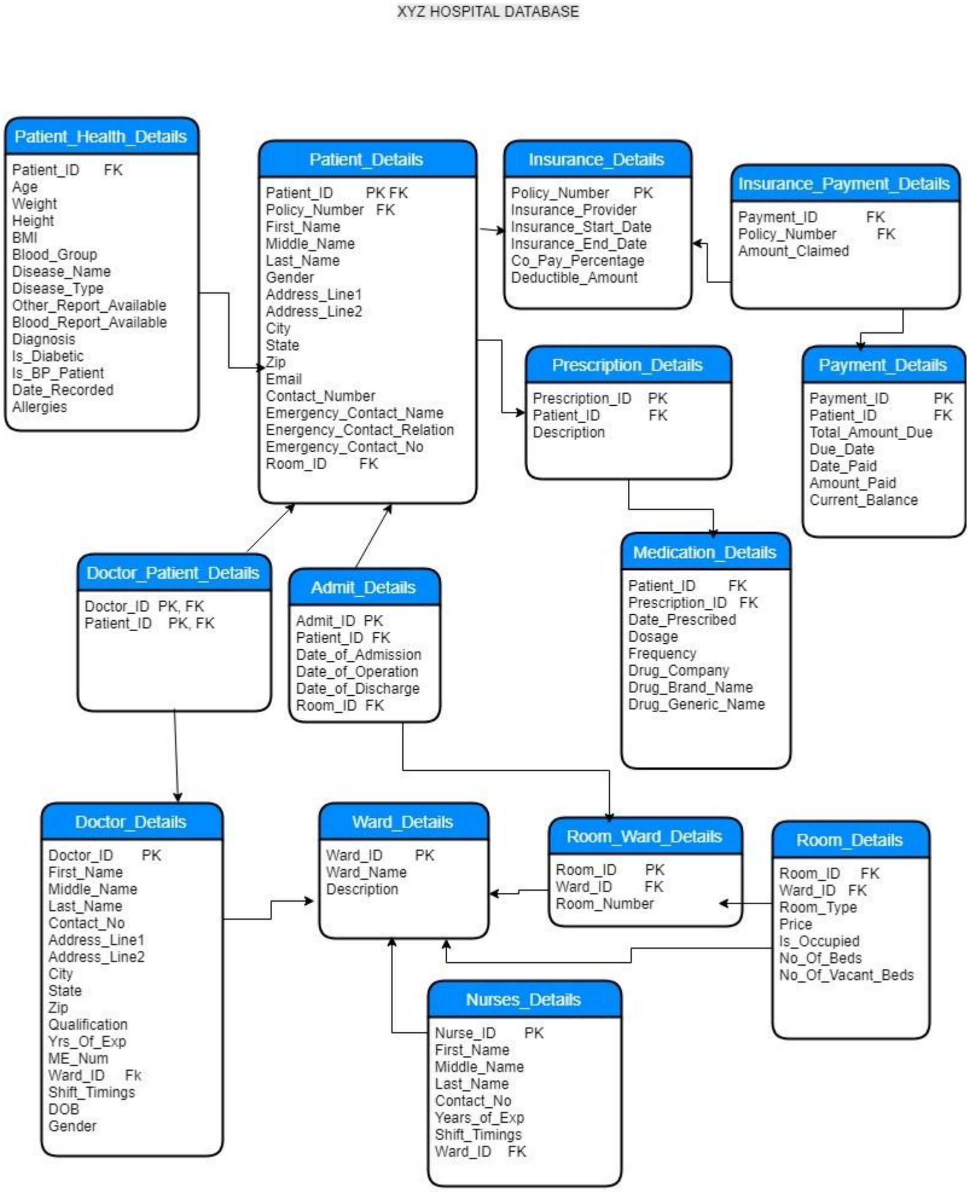
7. It is assumed that there are a hundred (100) doctors, three hundred and fifty (350) nurses and five thousand (5000) patients recorded in the database and 10,000 admit details.

8. A web portal exists from which data can be stored in and extracted from this database.

IV. Logical Database Design

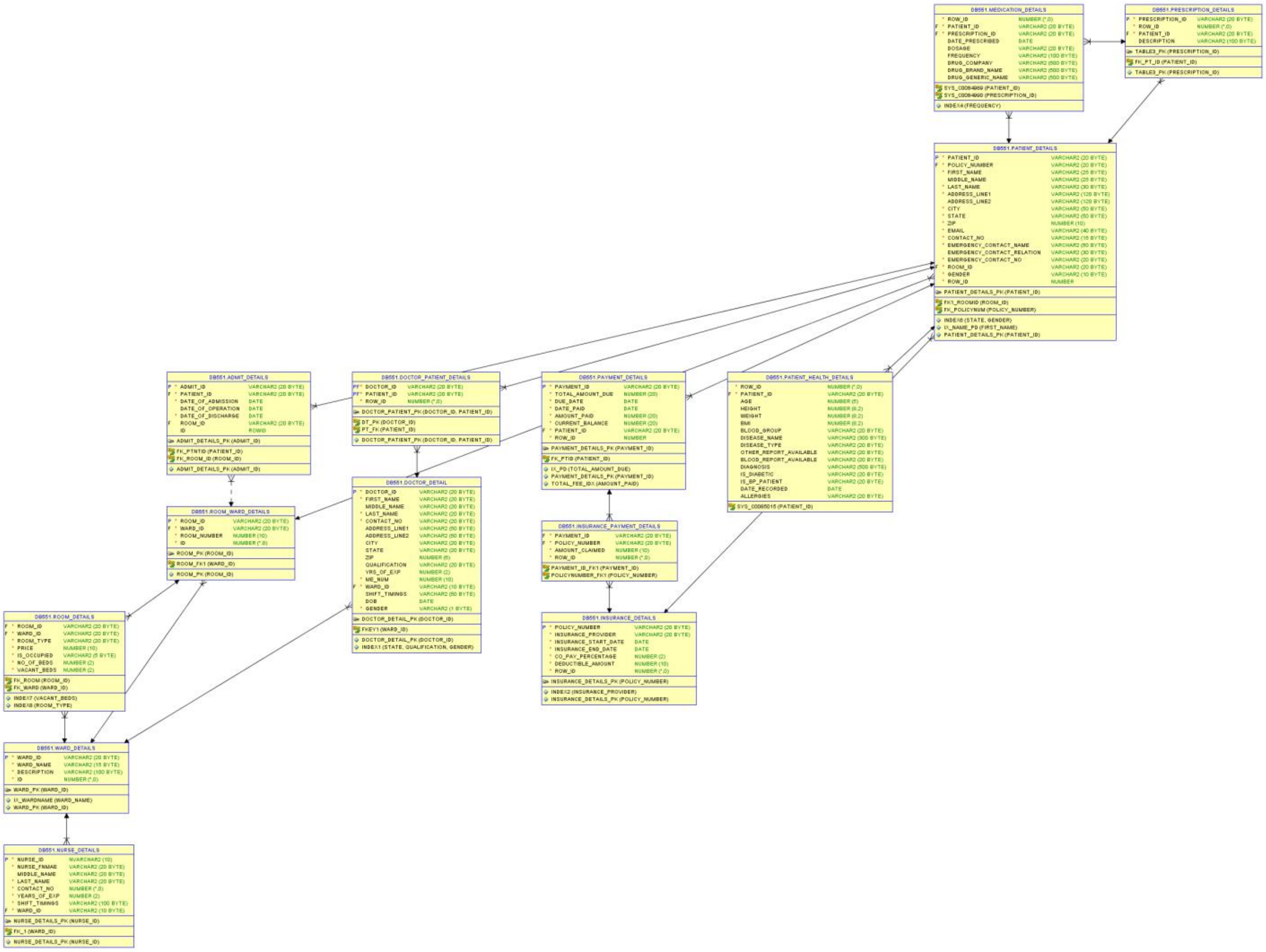
a. Entity Relationship Diagram

Based on business requirement, ER diagram is made. This Entity-Relationship Diagram identifies all the entities and attributes which have been created for the project.



b. Logical Design

A logical database design is required before beginning with physical database design. We used Oracle data modeler to create Logical design. The design is created automatically as per the table used in developing the database. Below is the diagram that shows the attributes of the table and how all the tables are logically related.



```
DB551.MEDICATION_DETAILS
  * ROW_ID                 NUMBER (*,0)
F * PATIENT_ID             VARCHAR2 (20 BYTE)
F * PRESCRIPTION_ID        VARCHAR2 (20 BYTE)
    DATE_PRESCRIBED        DATE
    DOSAGE                 VARCHAR2 (20 BYTE)
    FREQUENCY              VARCHAR2 (100 BYTE)
    DRUG_COMPANY           VARCHAR2 (500 BYTE)
    DRUG_BRAND_NAME        VARCHAR2 (500 BYTE)
    DRUG_GENERIC_NAME      VARCHAR2 (500 BYTE)
SYS_C0084989 (PATIENT_ID)
SYS_C0084990 (PRESCRIPTION_ID)
INDEX4 (FREQUENCY)

DB551.PRESCRIPTION_DETAILS
P * PRESCRIPTION_ID        VARCHAR2 (20 BYTE)
  * ROW_ID                 NUMBER (*,0)
F * PATIENT_ID             VARCHAR2 (20 BYTE)
    DESCRIPTION            VARCHAR2 (100 BYTE)
TABLE3_PK (PRESCRIPTION_ID)
FK_PT_ID (PATIENT_ID)
TABLE3_PK (PRESCRIPTION_ID)

DB551.PATIENT_DETAILS
P * PATIENT_ID                   VARCHAR2 (20 BYTE)
F * POLICY_NUMBER                VARCHAR2 (20 BYTE)
  * FIRST_NAME                   VARCHAR2 (25 BYTE)
    MIDDLE_NAME                  VARCHAR2 (25 BYTE)
  * LAST_NAME                    VARCHAR2 (30 BYTE)
  * ADDRESS_LINE1                VARCHAR2 (120 BYTE)
    ADDRESS_LINE2                VARCHAR2 (120 BYTE)
  * CITY                         VARCHAR2 (50 BYTE)
  * STATE                        VARCHAR2 (50 BYTE)
  * ZIP                          NUMBER (10)
  * EMAIL                        VARCHAR2 (40 BYTE)
  * CONTACT_NO                   VARCHAR2 (15 BYTE)
  * EMERGENCY_CONTACT_NAME       VARCHAR2 (50 BYTE)
    EMERGENCY_CONTACT_RELATION   VARCHAR2 (30 BYTE)
  * EMERGENCY_CONTACT_NO         VARCHAR2 (20 BYTE)
F * ROOM_ID                      VARCHAR2 (20 BYTE)
  * GENDER                       VARCHAR2 (10 BYTE)
  * ROW_ID                       NUMBER
PATIENT_DETAILS_PK (PATIENT_ID)
FK1_ROOMID (ROOM_ID)
FK_POLICYNUM (POLICY_NUMBER)
INDEX6 (STATE, GENDER)
IX_NAME_PD (FIRST_NAME)
PATIENT_DETAILS_PK (PATIENT_ID)

DB551.ADMIT_DETAILS
P * ADMIT_ID               VARCHAR2 (20 BYTE)
F * PATIENT_ID             VARCHAR2 (20 BYTE)
  * DATE_OF_ADMISSION      DATE
    DATE_OF_OPERATION      DATE
  * DATE_OF_DISCHARGE      DATE
F   ROOM_ID                VARCHAR2 (20 BYTE)
    ID                     ROWID
ADMIT_DETAILS_PK (ADMIT_ID)
FK_PTNTID (PATIENT_ID)
FK_ROOM_ID (ROOM_ID)
ADMIT_DETAILS_PK (ADMIT_ID)

DB551.DOCTOR_PATIENT_DETAILS
PF* DOCTOR_ID              VARCHAR2 (20 BYTE)
PF* PATIENT_ID             VARCHAR2 (20 BYTE)
  * ROW_ID                 NUMBER (*,0)
DOCTOR_PATIENT_PK (DOCTOR_ID, PATIENT_ID)
DT_PK (DOCTOR_ID)
PT_FK (PATIENT_ID)
DOCTOR_PATIENT_PK (DOCTOR_ID, PATIENT_ID)

DB551.PAYMENT_DETAILS
P * PAYMENT_ID             VARCHAR2 (20 BYTE)
  * TOTAL_AMOUNT_DUE       NUMBER (20)
  * DUE_DATE               DATE
  * DATE_PAID              DATE
  * AMOUNT_PAID            NUMBER (20)
  * CURRENT_BALANCE        NUMBER (20)
F * PATIENT_ID             VARCHAR2 (20 BYTE)
  * ROW_ID                 NUMBER
PAYMENT_DETAILS_PK (PAYMENT_ID)
FK_PTID (PATIENT_ID)
IX_PD (TOTAL_AMOUNT_DUE)
PAYMENT_DETAILS_PK (PAYMENT_ID)
TOTAL_FEE_IDX (AMOUNT_PAID)

DB551.PATIENT_HEALTH_DETAILS
  * ROW_ID                   NUMBER (*,0)
F * PATIENT_ID               VARCHAR2 (20 BYTE)
    AGE                      NUMBER (5)
    HEIGHT                   NUMBER (8,2)
    WEIGHT                   NUMBER (8,2)
    BMI                      NUMBER (8,2)
    BLOOD_GROUP              VARCHAR2 (20 BYTE)
    DISEASE_NAME             VARCHAR2 (300 BYTE)
    DISEASE_TYPE             VARCHAR2 (20 BYTE)
    OTHER_REPORT_AVAILABLE   VARCHAR2 (20 BYTE)
    BLOOD_REPORT_AVAILABLE   VARCHAR2 (20 BYTE)
    DIAGNOSIS                VARCHAR2 (500 BYTE)
    IS_DIABETIC              VARCHAR2 (20 BYTE)
    IS_BP_PATIENT            VARCHAR2 (20 BYTE)
    DATE_RECORDED            DATE
    ALLERGIES                VARCHAR2 (20 BYTE)
SYS_C0085015 (PATIENT_ID)

DB551.ROOM_WARD_DETAILS
P * ROOM_ID                VARCHAR2 (20 BYTE)
F * WARD_ID                VARCHAR2 (20 BYTE)
  * ROOM_NUMBER            NUMBER (10)
  * ID                     NUMBER (*,0)
ROOM_PK (ROOM_ID)
ROOM_FK1 (WARD_ID)
ROOM_PK (ROOM_ID)

DB551.DOCTOR_DETAIL
P * DOCTOR_ID              VARCHAR2 (20 BYTE)
  * FIRST_NAME             VARCHAR2 (20 BYTE)
    MIDDLE_NAME            VARCHAR2 (20 BYTE)
  * LAST_NAME              VARCHAR2 (20 BYTE)
  * CONTACT_NO             VARCHAR2 (20 BYTE)
    ADDRESS_LINE1          VARCHAR2 (50 BYTE)
    ADDRESS_LINE2          VARCHAR2 (50 BYTE)
    CITY                   VARCHAR2 (20 BYTE)
    STATE                  VARCHAR2 (20 BYTE)
    ZIP                    NUMBER (6)
    QUALIFICATION          VARCHAR2 (20 BYTE)
    YRS_OF_EXP             NUMBER (2)
  * ME_NUM                 NUMBER (10)
F * WARD_ID                VARCHAR2 (10 BYTE)
    SHIFT_TIMINGS          VARCHAR2 (50 BYTE)
    DOB                    DATE
  * GENDER                 VARCHAR2 (1 BYTE)
DOCTOR_DETAIL_PK (DOCTOR_ID)
FKEY1 (WARD_ID)
DOCTOR_DETAIL_PK (DOCTOR_ID)
INDEX1 (STATE, QUALIFICATION, GENDER)

DB551.INSURANCE_PAYMENT_DETAILS
F * PAYMENT_ID             VARCHAR2 (20 BYTE)
F * POLICY_NUMBER          VARCHAR2 (20 BYTE)
  * AMOUNT_CLAIMED         NUMBER (10)
  * ROW_ID                 NUMBER (*,0)
PAYMENT_ID_FK1 (PAYMENT_ID)
POLICYNUMBER_FK1 (POLICY_NUMBER)

DB551.INSURANCE_DETAILS
P * POLICY_NUMBER          VARCHAR2 (20 BYTE)
  * INSURANCE_PROVIDER     VARCHAR2 (20 BYTE)
  * INSURANCE_START_DATE   DATE
  * INSURANCE_END_DATE     DATE
  * CO_PAY_PERCENTAGE      NUMBER (2)
  * DEDUCTIBLE_AMOUNT      NUMBER (10)
  * ROW_ID                 NUMBER (*,0)
INSURANCE_DETAILS_PK (POLICY_NUMBER)
INDEX2 (INSURANCE_PROVIDER)
INSURANCE_DETAILS_PK (POLICY_NUMBER)

DB551.ROOM_DETAILS
F * ROOM_ID                VARCHAR2 (20 BYTE)
F * WARD_ID                VARCHAR2 (20 BYTE)
  * ROOM_TYPE              VARCHAR2 (20 BYTE)
  * PRICE                  NUMBER (10)
  * IS_OCCUPIED            VARCHAR2 (5 BYTE)
  * NO_OF_BEDS             NUMBER (2)
  * VACANT_BEDS            NUMBER (2)
FK_ROOM (ROOM_ID)
FK_WARD (WARD_ID)
INDEX7 (VACANT_BEDS)
INDEX8 (ROOM_TYPE)

DB551.WARD_DETAILS
P * WARD_ID                VARCHAR2 (20 BYTE)
  * WARD_NAME              VARCHAR2 (15 BYTE)
  * DESCRIPTION            VARCHAR2 (100 BYTE)
  * ID                     NUMBER (*,0)
WARD_PK (WARD_ID)
IX_WARDNAME (WARD_NAME)
WARD_PK (WARD_ID)

DB551.NURSE_DETAILS
P * NURSE_ID               NVARCHAR2 (10)
  * NURSE_FNMAE            VARCHAR2 (20 BYTE)
    MIDDLE_NAME            VARCHAR2 (20 BYTE)
  * LAST_NAME              VARCHAR2 (20 BYTE)
  * CONTACT_NO             NUMBER (*,0)
  * YEARS_OF_EXP           NUMBER (2)
  * SHIFT_TIMINGS          VARCHAR2 (100 BYTE)
F * WARD_ID                VARCHAR2 (10 BYTE)
NURSE_DETAILS_PK (NURSE_ID)
FK_1 (WARD_ID)
NURSE_DETAILS_PK (NURSE_ID)
```

c. Data Dictionary

TABLE NAME	NO. OF ROWS	ATTRIBUTES	DATA FORMAT	NULLABLE	PK
ADMIT_DETAILS	**10000**	ADMIT_ID	VARCHAR2(20 BYTE)	No	Y
		PATIENT_ID	VARCHAR2(20 BYTE)	No	
		DATE_OF_ADMISSION	DATE	No	
		DATE_OF_OPERATION	DATE	Yes	
		DATE_OF_DISCHARGE	DATE	No	
		ROOM_ID	VARCHAR2(20 BYTE)	Yes	
DOCTOR_DETAIL	**100**	DOCTOR_ID	VARCHAR2(20 BYTE)	No	Y
		FIRST_NAME	VARCHAR2(20 BYTE)	No	
		MIDDLE_NAME	VARCHAR2(20 BYTE)	Yes	
		LAST_NAME	VARCHAR2(20 BYTE)	No	
		CONTACT_NO	VARCHAR2(20 BYTE)	No	
		ADDRESS_LINE1	VARCHAR2(50 BYTE)	Yes	
		ADDRESS_LINE2	VARCHAR2(50 BYTE)	Yes	
		CITY	VARCHAR2(20 BYTE)	Yes	
		STATE	VARCHAR2(20 BYTE)	Yes	
		ZIP	NUMBER (6,0)	Yes	
		QUALIFICATION	VARCHAR2(20 BYTE)	Yes	
		YRS_OF_EXP	NUMBER (2,0)	Yes	
		ME_NUM	NUMBER (10,0)	No	
		WARD_ID	VARCHAR2(10 BYTE)	No	
		SHIFT_TIMINGS	VARCHAR2(50 BYTE)	Yes	
		DOB	DATE	Yes	
		GENDER	VARCHAR2(1 BYTE)	No	

DOCTOR_PATIENT_DETAIL	**5000**	DOCTOR_ID	VARCHAR2(20 BYTE)	No	Y
		PATIENT_ID	VARCHAR2(20 BYTE)	No	Y
INSURANCE_DETAILS	**5000**	POLICY_NUMBER	VARCHAR2(20 BYTE)	No	Y
		INSURANCE_PROVIDER	VARCHAR2(20 BYTE)	No	
		INSURANCE_START_DATE	DATE	No	
		INSURANCE_END_DATE	DATE	No	
		CO_PAY_PERCENTAGE	NUMBER (2,0)	No	
		DEDUCTIBLE_AMOUNT	NUMBER (10,0)	No	
INSURANCE_PAYMENT_DETAILS	**10000**	PAYMENT_ID	VARCHAR2(20 BYTE)	No	Y
		POLICY_NUMBER	VARCHAR2(20 BYTE)	No	
		AMOUNT_CLAIMED	NUMBER (10,0)	No	
MEDICATION_DETAILS	**10000**	PATIENT_ID	VARCHAR2(20 BYTE)	No	
		PRESCRIPTION_ID	VARCHAR2(20 BYTE)	No	
		DATE_PRESCRIBED	DATE	Yes	
		DOSAGE	VARCHAR2(20 BYTE)	Yes	
		FREQUENCY	VARCHAR2(100 BYTE)	Yes	
		DRUG_COMPANY	VARCHAR2(500 BYTE)	Yes	
		DRUG_BRAND_NAME	VARCHAR2(500 BYTE)	Yes	
		DRUG_GENERIC_NAME	VARCHAR2(500 BYTE)	Yes	
NURSE_DETAILS	**350**	NURSE_ID	NVARCHAR2(10 CHAR)	No	Y
		NURSE_FNMAE	VARCHAR2(20 BYTE)	No	
		MIDDLE_NAME	VARCHAR2(20 BYTE)	Yes	

		LAST_NAME	VARCHAR2(20 BYTE)	No	
		CONTACT_NO	NUMBER (38,0)	No	
		YEARS_OF_EXP	NUMBER (2,0)	No	
		SHIFT_TIMINGS	VARCHAR2(100 BYTE)	No	
		WARD_ID	VARCHAR2(10 BYTE)	No	
PATIENT_DETAILS	**5000**	PATIENT_ID	VARCHAR2(20 BYTE)	No	Y
		POLICY_NUMBER	VARCHAR2(20 BYTE)	No	
		FIRST_NAME	VARCHAR2(25 BYTE)	No	
		MIDDLE_NAME	VARCHAR2(25 BYTE)	Yes	
		LAST_NAME	VARCHAR2(30 BYTE)	No	
		ADDRESSLINE1	VARCHAR2(120 BYTE)	No	
		ADDRESSLINE2	VARCHAR2(120 BYTE)	Yes	
		CITY	VARCHAR2(50 BYTE)	No	
		STATE	VARCHAR2(50 BYTE)	No	
		ZIP	NUMBER (10,0)	No	
		EMAIL	VARCHAR2(40 BYTE)	No	
		CONTACT_NO	VARCHAR2(15 BYTE)	No	
		EMERGENCY_CONTACT_NAME	VARCHAR2(50 BYTE)	No	
		EMERGENCY_CONTACT_RELATION	VARCHAR2(30 BYTE)	Yes	
		EMERGENCY_CONTACT_NO	VARCHAR2(20 BYTE)	No	
		ROOM_ID	VARCHAR2(20 BYTE)	No	
		GENDER	VARCHAR2(10 BYTE)	No	
PATIENT_HEALTH_DETAILS	**5000**	PATIENT_ID	VARCHAR2(20 BYTE)	No	
		AGE	NUMBER (5,0)	Yes	
		HEIGHT	NUMBER (8,2)	Yes	

		WEIGHT	NUMBER (8,2)	Yes	
		BMI	NUMBER (8,2)	Yes	
		BLOOD_GROUP	VARCHAR2(20 BYTE)	Yes	
		DISEASE_NAME	VARCHAR2(300 BYTE)	Yes	
		DISEASE_TYPE	VARCHAR2(20 BYTE)	Yes	
		OTHER_REPORT_AVAILABLE	VARCHAR2(20 BYTE)	Yes	
		BLOOD_REPORT_AVAILABLE	VARCHAR2(20 BYTE)	Yes	
		DIAGNOSIS	VARCHAR2(500 BYTE)	Yes	
		IS_DIABETIC	VARCHAR2(20 BYTE)	Yes	
		IS_BP_PATIENT	VARCHAR2(20 BYTE)	Yes	
		DATE_RECORDED	DATE	Yes	
		ALLERGIES	VARCHAR2(20 BYTE)	Yes	
PAYMENT_DETAILS	**10000**	PAYMENT_ID	VARCHAR2(20 BYTE)	No	Y
		TOTAL_AMOUNT_DUE	NUMBER (20,0)	No	
		DUE_DATE	DATE	No	
		DATE_PAID	DATE	No	
		AMOUNT_PAID	NUMBER (20,0)	No	
		CURRENT_BALANCE	NUMBER (20,0)	No	
PRESCRIPTION_DETAILS	**10000**	PRESCRIPTION_ID	VARCHAR2(20 BYTE)	No	Y
		PATIENT_ID	VARCHAR2(20 BYTE)	No	
		DESCRIPTION	VARCHAR2(100 BYTE)	Yes	
ROOM_DETAILS	**100**	ROOM_ID	VARCHAR2(20 BYTE)	No	
		WARD_ID	VARCHAR2(20 BYTE)	No	
		ROOM_TYPE	VARCHAR2(20 BYTE)	No	

		PRICE	NUMBER (10,0)	No	
		IS_OCCUPIED	VARCHAR2(5 BYTE)	No	
		NO_OF_BEDS	NUMBER (2,0)	No	
		VACANT_BEDS	NUMBER (2,0)	No	
ROOM_WARD_DETAILS	**100**	ROOM_ID	VARCHAR2(20 BYTE)	No	Y
		WARD_ID	VARCHAR2(20 BYTE)	No	
		ROOMNUMBER	NUMBER (10,0)	No	
WARD_DETAILS	**10**	WARD_ID	VARCHAR2(20 BYTE)	No	Y
		WARD_NAME	VARCHAR2(15 BYTE)	No	
		DESCRIPTION	VARCHAR2(100 BYTE)	No	

V. Physical Database

Once the logical database design is complete, we transformed the logical design physical implementation. The very first step is to transform entities to table, attributes into column and domains into data type and constraint. There are several design strategies implemented while designing database tables. Foreign key n each table are indexed to provide faster response.

We used indexing in most of the table to reduce the cost and time of query execution. Unique index which is same as primary key is created in tables like PATIENT_DETAILS, NURSE, ROOM_DETAILS, DOCTOR_DETAILS, WARD_DETAILS etc. We also created B tree index on several columns of the table which are having huge number of record. B- tree index reduce the scan time of large tables.
We also created Bit map index on the tables like WARD_DETAILS, Gender column of PATIENT_DETAILS, DOCTOR_DETAILS as these columns has low cardinality.

a. Table Creation

i. ADMIT_DETAILS

DDL:

```
CREATE TABLE "DB551"."ADMIT_DETAILS"
 (      "ADMIT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
        "PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
        "DATE_OF_ADMISSION" DATE NOT NULL ENABLE,
        "DATE_OF_OPERATION" DATE,
        "DATE_OF_DISCHARGE" DATE NOT NULL ENABLE,
        "ROOM_ID" VARCHAR2(20 BYTE),
        "ID" ROWID,
         CONSTRAINT "ADMIT_DETAILS_PK" PRIMARY KEY ("ADMIT_ID")
```

```
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ENABLE,
              CONSTRAINT "FK_ROOM_ID" FOREIGN KEY ("ROOM_ID")
               REFERENCES "DB551"."ROOM_WARD_DETAILS" ("ROOM_ID")
ENABLE,
              CONSTRAINT "FK_PTNTID" FOREIGN KEY ("PATIENT_ID")
               REFERENCES "DB551"."PATIENT_DETAILS" ("PATIENT_ID")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "STUDENTS";
```

ii. DOCTOR_DETAIL

DDL:

```
  CREATE TABLE "DB551"."DOCTOR_DETAIL"
   (        "DOCTOR_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "FIRST_NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "MIDDLE_NAME" VARCHAR2(20 BYTE),
        "LAST_NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,
        "CONTACT_NO" VARCHAR2(20 BYTE) NOT NULL ENABLE,
        "ADDRESS_LINE1" VARCHAR2(50 BYTE),
        "ADDRESS_LINE2" VARCHAR2(50 BYTE),
        "CITY" VARCHAR2(20 BYTE),
        "STATE" VARCHAR2(20 BYTE),
        "ZIP" NUMBER (6,0),
        "QUALIFICATION" VARCHAR2(20 BYTE),
        "YRS_OF_EXP" NUMBER (2,0),
        "ME_NUM" NUMBER (10,0) NOT NULL ENABLE,
        "WARD_ID" VARCHAR2(10 BYTE) NOT NULL ENABLE,
        "SHIFT_TIMINGS" VARCHAR2(50 BYTE),
        "DOB" DATE,
        "GENDER" VARCHAR2(1 BYTE) NOT NULL ENABLE,
         CONSTRAINT "DOCTOR_DETAIL_PK" PRIMARY KEY ("DOCTOR_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ENABLE,
         CONSTRAINT "FKEY1" FOREIGN KEY ("WARD_ID")
          REFERENCES "DB551"."WARD_DETAILS" ("WARD_ID") ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";

  CREATE BITMAP INDEX "DB551"."INDEX1" ON "DB551"."DOCTOR_DETAIL"
("STATE", "QUALIFICATION", "GENDER")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";
```

iii. DOCTOR_PATIENT_DETAILS

DDL:

```
  CREATE TABLE "DB551"."DOCTOR_PATIENT_DETAILS"
   (	"DOCTOR_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"ROW_ID" NUMBER (*,0) GENERATED ALWAYS AS IDENTITY
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT
BY 1 START WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
	 CONSTRAINT "DOCTOR_PATIENT_PK" PRIMARY KEY ("DOCTOR_ID",
"PATIENT_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ENABLE,
              CONSTRAINT "DT_PK" FOREIGN KEY ("DOCTOR_ID")
               REFERENCES "DB551"."DOCTOR_DETAIL" ("DOCTOR_ID") ENABLE,
              CONSTRAINT "PT_FK" FOREIGN KEY ("PATIENT_ID")
               REFERENCES "DB551"."PATIENT_DETAILS" ("PATIENT_ID")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

iv. INSURANCE_DETAILS

DDL:

```
  CREATE TABLE "DB551"."INSURANCE_DETAILS"
   (        "POLICY_NUMBER" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "INSURANCE_PROVIDER" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "INSURANCE_START_DATE" DATE NOT NULL ENABLE,
            "INSURANCE_END_DATE" DATE NOT NULL ENABLE,
            "CO_PAY_PERCENTAGE" NUMBER(2,0) NOT NULL ENABLE,
            "DEDUCTIBLE_AMOUNT" NUMBER(10,0) NOT NULL ENABLE,
            "ROW_ID" NUMBER(*,0) GENERATED ALWAYS AS IDENTITY
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT
BY 1 START WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
           CONSTRAINT "INSURANCE_DETAILS_PK" PRIMARY KEY
("POLICY_NUMBER")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS"  ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;

  CREATE BITMAP INDEX "DB551"."INDEX2" ON
"DB551"."INSURANCE_DETAILS" ("INSURANCE_PROVIDER")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

v. INSURANCE_PAYMENT_DETAILS

DDL:

```
  CREATE TABLE "DB551"."INSURANCE_PAYMENT_DETAILS"
   (	"PAYMENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"POLICY_NUMBER" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"AMOUNT_CLAIMED" NUMBER(10,0) NOT NULL ENABLE,
	"ROW_ID" NUMBER(*,0) GENERATED ALWAYS AS IDENTITY
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT
BY 1 START WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
	 CONSTRAINT "PAYMENT_ID_FK1" FOREIGN KEY ("PAYMENT_ID")
	  REFERENCES "DB551"."PAYMENT_DETAILS" ("PAYMENT_ID")
ENABLE,
	 CONSTRAINT "POLICYNUMBER_FK1" FOREIGN KEY
("POLICY_NUMBER")
	  REFERENCES "DB551"."INSURANCE_DETAILS" ("POLICY_NUMBER")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

vi. MEDICATION_DETAILS

DDL:

```
  CREATE TABLE "DB551"."MEDICATION_DETAILS"
   (	"ROW_ID" NUMBER(*,0) GENERATED ALWAYS AS IDENTITY
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT
BY 1 START WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
	"PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"PRESCRIPTION_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"DATE_PRESCRIBED" DATE,
	"DOSAGE" VARCHAR2(20 BYTE),
	"FREQUENCY" VARCHAR2(100 BYTE),
	"DRUG_COMPANY" VARCHAR2(500 BYTE),
	"DRUG_BRAND_NAME" VARCHAR2(500 BYTE),
	"DRUG_GENERIC_NAME" VARCHAR2(500 BYTE),
	 FOREIGN KEY ("PATIENT_ID")
	  REFERENCES "DB551"."PATIENT_DETAILS" ("PATIENT_ID")
ENABLE,
	 FOREIGN KEY ("PRESCRIPTION_ID")
	  REFERENCES "DB551"."PRESCRIPTION_DETAILS"
("PRESCRIPTION_ID") ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
```

```
  TABLESPACE "STUDENTS" ;

  CREATE BITMAP INDEX "DB551"."INDEX4" ON
"DB551"."MEDICATION_DETAILS" ("FREQUENCY")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

vii. NURSE_DETAILS

DDL:

```
  CREATE TABLE "DB551"."NURSE_DETAILS"
   (        "NURSE_ID" NVARCHAR2(10) NOT NULL ENABLE,
            "NURSE_FNMAE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "MIDDLE_NAME" VARCHAR2(20 BYTE),
            "LAST_NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "CONTACT_NO" NUMBER(*,0) NOT NULL ENABLE,
            "YEARS_OF_EXP" NUMBER(2,0) NOT NULL ENABLE,
            "SHIFT_TIMINGS" VARCHAR2(100 BYTE) NOT NULL ENABLE,
            "WARD_ID" VARCHAR2(10 BYTE) NOT NULL ENABLE,
             CONSTRAINT "NURSE_DETAILS_PK" PRIMARY KEY ("NURSE_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS"  ENABLE,
              CONSTRAINT "FK_1" FOREIGN KEY ("WARD_ID")
               REFERENCES "DB551"."WARD_DETAILS" ("WARD_ID") ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

viii. PATIENT_DETAILS

DDL:

```
  CREATE TABLE "DB551"."PATIENT_DETAILS"
   (        "PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "POLICY_NUMBER" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "FIRST_NAME" VARCHAR2(25 BYTE) NOT NULL ENABLE,
            "MIDDLE_NAME" VARCHAR2(25 BYTE),
            "LAST_NAME" VARCHAR2(30 BYTE) NOT NULL ENABLE,
            "ADDRESS_LINE1" VARCHAR2(120 BYTE) NOT NULL ENABLE,
            "ADDRESS_LINE2" VARCHAR2(120 BYTE),
            "CITY" VARCHAR2(50 BYTE) NOT NULL ENABLE,
            "STATE" VARCHAR2(50 BYTE) NOT NULL ENABLE,
            "ZIP" NUMBER(10,0) NOT NULL ENABLE,
            "EMAIL" VARCHAR2(40 BYTE) NOT NULL ENABLE,
            "CONTACT_NO" VARCHAR2(15 BYTE) NOT NULL ENABLE,
            "EMERGENCY_CONTACT_NAME" VARCHAR2(50 BYTE) NOT NULL
ENABLE,
            "EMERGENCY_CONTACT_RELATION" VARCHAR2(30 BYTE),
            "EMERGENCY_CONTACT_NO" VARCHAR2(20 BYTE) NOT NULL
ENABLE,
            "ROOM_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "GENDER" VARCHAR2(10 BYTE) NOT NULL ENABLE,
            "ROW_ID" NUMBER GENERATED ALWAYS AS IDENTITY MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START
WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
             CONSTRAINT "PATIENT_DETAILS_PK" PRIMARY KEY
("PATIENT_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS"  ENABLE,
             CONSTRAINT "FK_POLICYNUM" FOREIGN KEY ("POLICY_NUMBER")
              REFERENCES "DB551"."INSURANCE_DETAILS" ("POLICY_NUMBER")
ENABLE,
             CONSTRAINT "FK1_ROOMID" FOREIGN KEY ("ROOM_ID")
              REFERENCES "DB551"."ROOM_WARD_DETAILS" ("ROOM_ID")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
```

```
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;

  CREATE BITMAP INDEX "DB551"."INDEX6" ON
"DB551"."PATIENT_DETAILS" ("STATE", "GENDER")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;

  CREATE INDEX "DB551"."IX_NAME_PD" ON "DB551"."PATIENT_DETAILS"
("FIRST_NAME")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

ix. PATIENT_HEALTH_DETAILS

DDL:

```
  CREATE TABLE "DB551"."PATIENT_HEALTH_DETAILS"
   (	"ROW_ID" NUMBER(*,0) GENERATED ALWAYS AS IDENTITY
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT
BY 1 START WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
	"PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"AGE" NUMBER(5,0),
	"HEIGHT" NUMBER(8,2),
	"WEIGHT" NUMBER(8,2),
	"BMI" NUMBER(8,2),
	"BLOOD_GROUP" VARCHAR2(20 BYTE),
	"DISEASE_NAME" VARCHAR2(300 BYTE),
	"DISEASE_TYPE" VARCHAR2(20 BYTE),
	"OTHER_REPORT_AVAILABLE" VARCHAR2(20 BYTE),
	"BLOOD_REPORT_AVAILABLE" VARCHAR2(20 BYTE),
	"DIAGNOSIS" VARCHAR2(500 BYTE),
	"IS_DIABETIC" VARCHAR2(20 BYTE),
	"IS_BP_PATIENT" VARCHAR2(20 BYTE),
	"DATE_RECORDED" DATE,
	"ALLERGIES" VARCHAR2(20 BYTE),
	 FOREIGN KEY ("PATIENT_ID")
	  REFERENCES "DB551"."PATIENT_DETAILS" ("PATIENT_ID")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
```

```
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

x. PAYMENT_DETAILS

DDL:

```
  CREATE TABLE "DB551"."PAYMENT_DETAILS"
   (	"PAYMENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"TOTAL_AMOUNT_DUE" NUMBER(20,0) NOT NULL ENABLE,
	"DUE_DATE" DATE NOT NULL ENABLE,
	"DATE_PAID" DATE NOT NULL ENABLE,
	"AMOUNT_PAID" NUMBER(20,0) NOT NULL ENABLE,
	"CURRENT_BALANCE" NUMBER(20,0) NOT NULL ENABLE,
	"PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"ROW_ID" NUMBER GENERATED ALWAYS AS IDENTITY MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START
WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
	 CONSTRAINT "PAYMENT_DETAILS_PK" PRIMARY KEY
("PAYMENT_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS"  ENABLE,
	 CONSTRAINT "FK_PTID" FOREIGN KEY ("PATIENT_ID")
```

```
                REFERENCES "DB551"."PATIENT_DETAILS" ("PATIENT_ID")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;

  CREATE INDEX "DB551"."IX_PD" ON "DB551"."PAYMENT_DETAILS"
("TOTAL_AMOUNT_DUE")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;

  CREATE INDEX "DB551"."TOTAL_FEE_IDX" ON
"DB551"."PAYMENT_DETAILS" ("AMOUNT_PAID"+"CURRENT_BALANCE")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ;
```

xi. PRESCRIPTION_DETAILS

DDL:

```
  CREATE TABLE "DB551"."PRESCRIPTION_DETAILS"
   (	"PRESCRIPTION_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"ROW_ID" NUMBER (*,0) GENERATED ALWAYS AS IDENTITY
MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT
BY 1 START WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
	"PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"DESCRIPTION" VARCHAR2(100 BYTE),
	 CONSTRAINT "TABLE3_PK" PRIMARY KEY ("PRESCRIPTION_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ENABLE,
	 CONSTRAINT "FK_PT_ID" FOREIGN KEY ("PATIENT_ID")
	  REFERENCES "DB551"."PATIENT_DETAILS" ("PATIENT_ID")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
```

```
  TABLESPACE "STUDENTS";
```

xii. ROOM_DETAILS

DDL:

```
  CREATE TABLE "DB551"."ROOM_DETAILS"
   (        "ROOM_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "WARD_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "ROOM_TYPE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "PRICE" NUMBER (10,0) NOT NULL ENABLE,
            "IS_OCCUPIED" VARCHAR2(5 BYTE) NOT NULL ENABLE,
            "NO_OF_BEDS" NUMBER (2,0) NOT NULL ENABLE,
            "VACANT_BEDS" NUMBER (2,0) NOT NULL ENABLE,
             CONSTRAINT "FK_WARD" FOREIGN KEY ("WARD_ID")
              REFERENCES "DB551"."WARD_DETAILS" ("WARD_ID") ENABLE,
             CONSTRAINT "FK_ROOM" FOREIGN KEY ("ROOM_ID")
              REFERENCES "DB551"."ROOM_WARD_DETAILS" ("ROOM_ID")
ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";

  CREATE BITMAP INDEX "DB551"."INDEX7" ON "DB551"."ROOM_DETAILS"
("VACANT_BEDS")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";

  CREATE BITMAP INDEX "DB551"."INDEX8" ON "DB551"."ROOM_DETAILS"
("ROOM_TYPE")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";
```

xiii. ROOM_WARD_DETAILS

DDL:

```
  CREATE TABLE "DB551"."ROOM_WARD_DETAILS"
   (	"ROOM_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"WARD_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
	"ROOM_NUMBER" NUMBER (10,0) NOT NULL ENABLE,
	"ID" NUMBER (*,0) GENERATED ALWAYS AS IDENTITY MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START
WITH 1 CACHE 20 NOORDER NOCYCLE NOT NULL ENABLE,
	 CONSTRAINT "ROOM_PK" PRIMARY KEY ("ROOM_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
```

```
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ENABLE,
             CONSTRAINT "ROOM_FK1" FOREIGN KEY ("WARD_ID")
              REFERENCES "DB551"."WARD_DETAILS" ("WARD_ID") ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";
```

xiv. WARD_DETAILS

DDL:

```
  CREATE TABLE "DB551"."WARD_DETAILS"
   (        "WARD_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
            "WARD_NAME" VARCHAR2(15 BYTE) NOT NULL ENABLE,
            "DESCRIPTION" VARCHAR2(100 BYTE) NOT NULL ENABLE,
            "ID" NUMBER (*,0) GENERATED ALWAYS AS IDENTITY MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START
WITH 1 CACHE 20 NOORDER  NOCYCLE  NOT NULL ENABLE,
             CONSTRAINT "WARD_PK" PRIMARY KEY ("WARD_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
STATISTICS
```

```
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS" ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";

  CREATE INDEX "DB551"."IX_WARDNAME" ON "DB551"."WARD_DETAILS"
("WARD_NAME")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
  TABLESPACE "STUDENTS";
```

b. Capacity Planning

The below table gives an idea about the space occupied by the tables.

We can see that ROOM_DETAILS occupy the smallest amount of space while PATIENT_DETAILS occupies the latest.

DATABASE	TABLE	SIZE(KB)
DB551	ADMIT_DETAILS	896
DB551	DOCTOR_DETAIL	192
DB551	DOCTOR_PATIENT_DETAILS	448
DB551	INSURANCE_DETAILS	640
DB551	INSURANCE_PAYMENT_DETAILS	384
DB551	MEDICATION_DETAILS	2112
DB551	NURSE_DETAILS	128
DB551	PATIENT_DETAILS	2368
DB551	PATIENT_HEALTH_DETAILS	2048
DB551	PAYMENT_DETAILS	1088
DB551	PRESCRIPTION_DETAILS	896
DB551	ROOM_DETAILS	64
DB551	ROOM_WARD_DETAILS	128
DB551	WARD_DETAILS	192

```
SELECT table_name, num_rows, blocks, empty_blocks, avg_space, chain_cnt,
avg_row_len
FROL all_tables
WHERE owner = 'DB551';
```

table_name	num_rows	blocks	empty_blocks	avg_space	chain_cnt	avg_row_len
ADMIT_DETAILS	10000	65	0	0	0	43
DOCTOR_DETAIL	100	5	0	0	0	159
DOCTOR_PATIENT_DETAILS	5000	20	0	0	0	19
INSURANCE_DETAILS	5000	43	0	0	0	50
INSURANCE_PAYMENT_DETAILS	10000	43	0	0	0	27
MEDICATION_DETAILS	10000	244	0	0	0	112
NURSE_DETAILS	350	5	0	0	0	81
PATIENT_DETAILS	5000	244	0	0	0	167
PATIENT_HEALTH_DETAILS	5000	244	0	0	0	166
PAYMENT_DETAILS	10000	73	0	0	0	46
PRESCRIPTION_DETAILS	10000	73	0	0	0	42
ROOM_DETAILS	100	5	0	0	0	39
ROOM_WARD_DETAILS	100	5	0	0	0	23
WARD_DETAILS	10	5	0	0	0	31

We, now have an idea about the space occupied by this project's tables.

VI. Data Generation and Loading

We were provided with login credentials to connect to USF system where we can store our data. Data was generated using open-source data generation tools available over the internet. All the generated data was stored in spreadsheets from where it was loaded into the database using SQL developer.

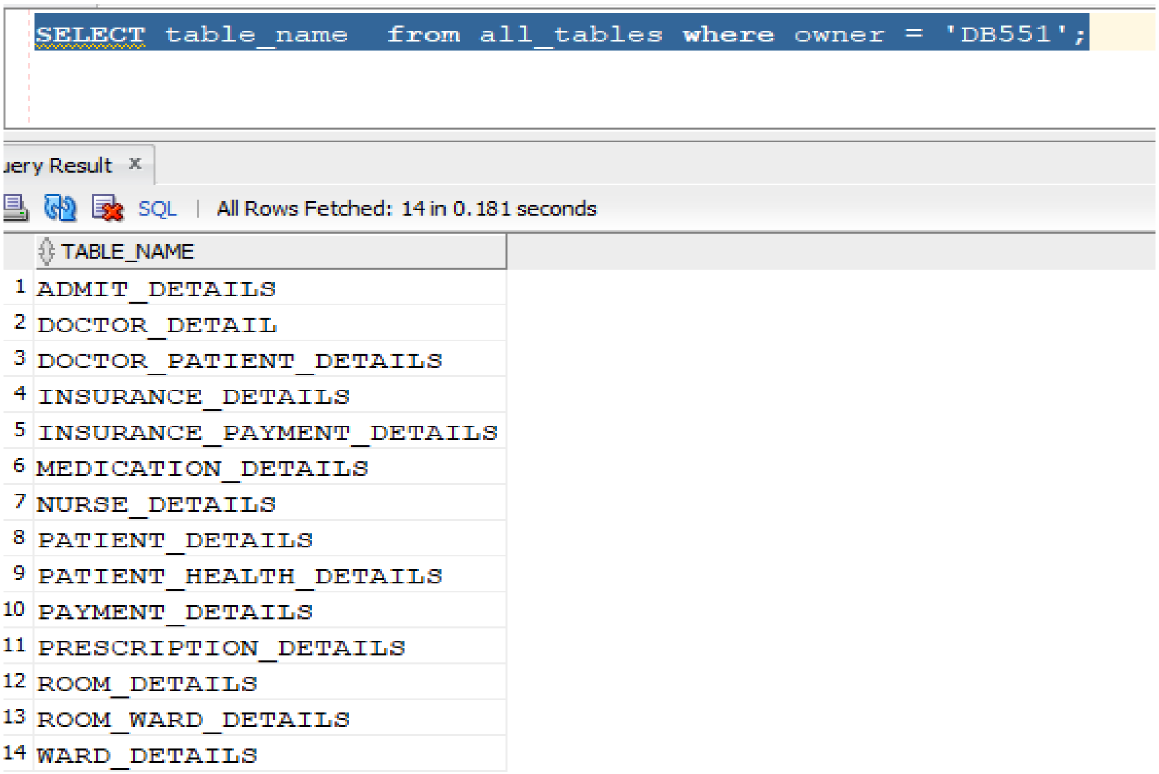
Database details:

Host: reade.fores.usf.edu

Port: 1521

SID: cdb9

DB Username: db551



```
SELECT table_name  from all_tables where owner = 'DB551';
```

All Rows Fetched: 14 in 0.181 seconds

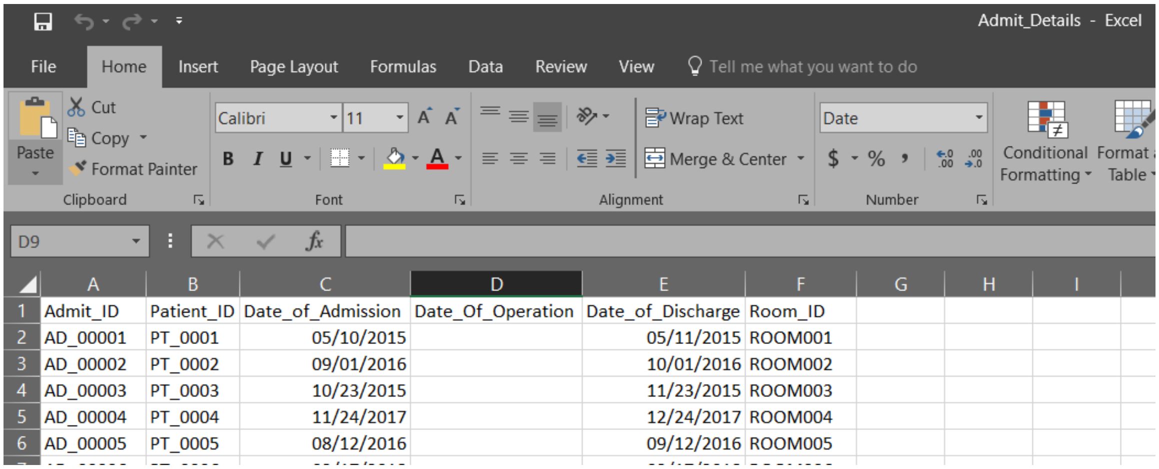
	TABLE_NAME
1	ADMIT_DETAILS
2	DOCTOR_DETAIL
3	DOCTOR_PATIENT_DETAILS
4	INSURANCE_DETAILS
5	INSURANCE_PAYMENT_DETAILS
6	MEDICATION_DETAILS
7	NURSE_DETAILS
8	PATIENT_DETAILS
9	PATIENT_HEALTH_DETAILS
10	PAYMENT_DETAILS
11	PRESCRIPTION_DETAILS
12	ROOM_DETAILS
13	ROOM_WARD_DETAILS
14	WARD_DETAILS

The tables are as follows:

1. ADMIT_DETAILS

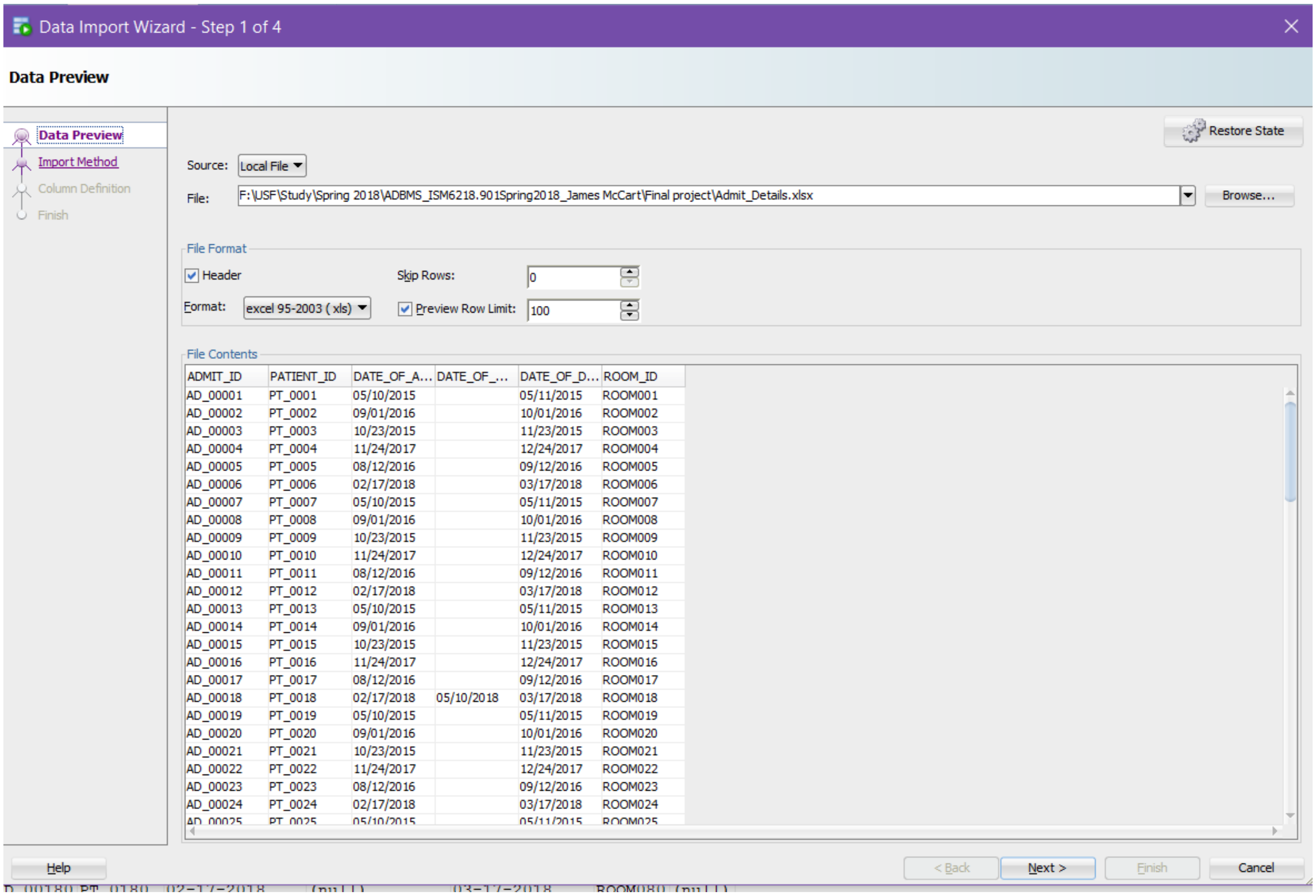
Let us take the example of this table to try and understand the process of generating and loading data.

First, the data is generated and stored in MS-Excel files, which in our case was Admit_Details.xlsx.



	A	B	C	D	E	F	G	H	I
1	Admit_ID	Patient_ID	Date_of_Admission	Date_Of_Operation	Date_of_Discharge	Room_ID			
2	AD_00001	PT_0001	05/10/2015		05/11/2015	ROOM001			
3	AD_00002	PT_0002	09/01/2016		10/01/2016	ROOM002			
4	AD_00003	PT_0003	10/23/2015		11/23/2015	ROOM003			
5	AD_00004	PT_0004	11/24/2017		12/24/2017	ROOM004			
6	AD_00005	PT_0005	08/12/2016		09/12/2016	ROOM005			

We then import and preview data the file in SQL Developer



Source: Local File

File: F:\USF\Study\Spring 2018\ADBMS_ISM6218.901Spring2018_James McCart\Final project\Admit_Details.xlsx

File Format

☑ Header   Skip Rows: 0

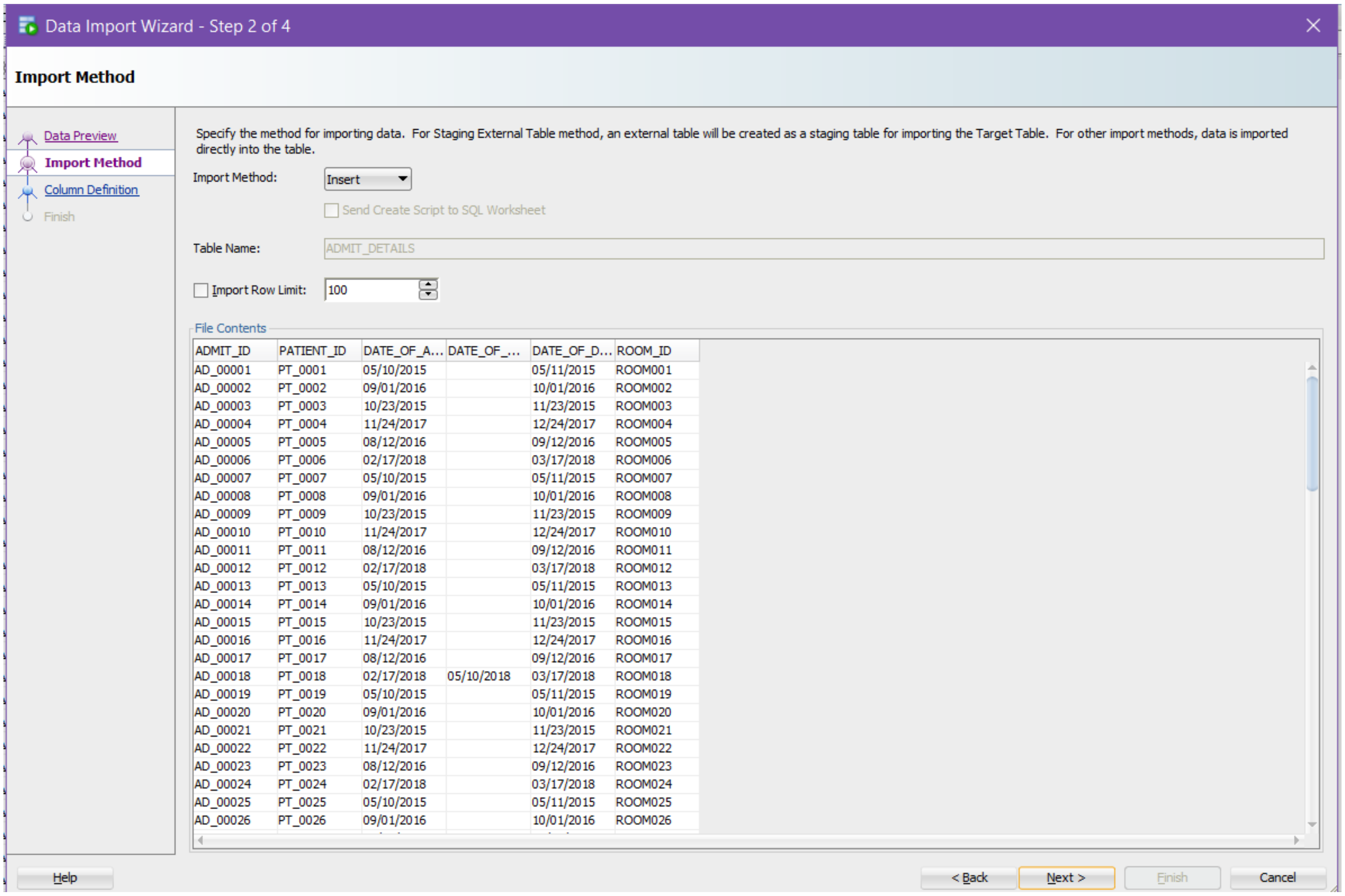
Format: excel 95-2003 (xls)   ☑ Preview Row Limit: 100

File Contents

ADMIT_ID	PATIENT_ID	DATE_OF_A...	DATE_OF_...	DATE_OF_D...	ROOM_ID
AD_00001	PT_0001	05/10/2015		05/11/2015	ROOM001
AD_00002	PT_0002	09/01/2016		10/01/2016	ROOM002
AD_00003	PT_0003	10/23/2015		11/23/2015	ROOM003
AD_00004	PT_0004	11/24/2017		12/24/2017	ROOM004
AD_00005	PT_0005	08/12/2016		09/12/2016	ROOM005
AD_00006	PT_0006	02/17/2018		03/17/2018	ROOM006
AD_00007	PT_0007	05/10/2015		05/11/2015	ROOM007
AD_00008	PT_0008	09/01/2016		10/01/2016	ROOM008
AD_00009	PT_0009	10/23/2015		11/23/2015	ROOM009
AD_00010	PT_0010	11/24/2017		12/24/2017	ROOM010
AD_00011	PT_0011	08/12/2016		09/12/2016	ROOM011
AD_00012	PT_0012	02/17/2018		03/17/2018	ROOM012
AD_00013	PT_0013	05/10/2015		05/11/2015	ROOM013
AD_00014	PT_0014	09/01/2016		10/01/2016	ROOM014
AD_00015	PT_0015	10/23/2015		11/23/2015	ROOM015
AD_00016	PT_0016	11/24/2017		12/24/2017	ROOM016
AD_00017	PT_0017	08/12/2016		09/12/2016	ROOM017
AD_00018	PT_0018	02/17/2018	05/10/2018	03/17/2018	ROOM018
AD_00019	PT_0019	05/10/2015		05/11/2015	ROOM019
AD_00020	PT_0020	09/01/2016		10/01/2016	ROOM020
AD_00021	PT_0021	10/23/2015		11/23/2015	ROOM021
AD_00022	PT_0022	11/24/2017		12/24/2017	ROOM022
AD_00023	PT_0023	08/12/2016		09/12/2016	ROOM023
AD_00024	PT_0024	02/17/2018		03/17/2018	ROOM024
AD_00025	PT_0025	05/10/2015		05/11/2015	ROOM025

```
D_00180 PT_0180  02-17-2018      (null)            03-17-2018       ROOM080 (null)
```

Then we choose the import method as "Insert".



Specify the method for importing data. For Staging External Table method, an external table will be created as a staging table for importing the Target Table. For other import methods, data is imported directly into the table.

Import Method: Insert

□ Send Create Script to SQL Worksheet

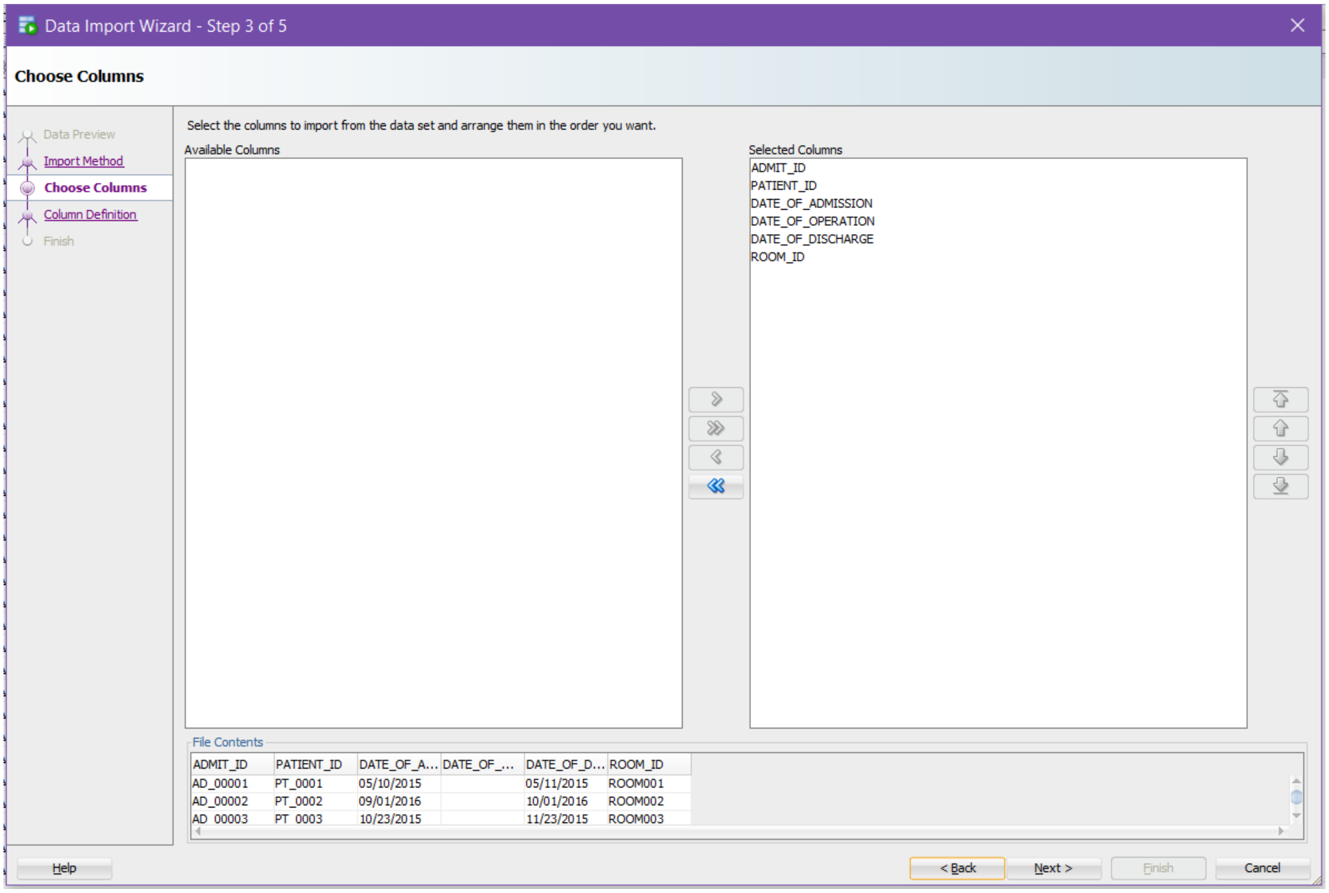
Table Name: ADMIT_DETAILS

□ Import Row Limit: 100

File Contents

ADMIT_ID	PATIENT_ID	DATE_OF_A...	DATE_OF_...	DATE_OF_D...	ROOM_ID
AD_00001	PT_0001	05/10/2015		05/11/2015	ROOM001
AD_00002	PT_0002	09/01/2016		10/01/2016	ROOM002
AD_00003	PT_0003	10/23/2015		11/23/2015	ROOM003
AD_00004	PT_0004	11/24/2017		12/24/2017	ROOM004
AD_00005	PT_0005	08/12/2016		09/12/2016	ROOM005
AD_00006	PT_0006	02/17/2018		03/17/2018	ROOM006
AD_00007	PT_0007	05/10/2015		05/11/2015	ROOM007
AD_00008	PT_0008	09/01/2016		10/01/2016	ROOM008
AD_00009	PT_0009	10/23/2015		11/23/2015	ROOM009
AD_00010	PT_0010	11/24/2017		12/24/2017	ROOM010
AD_00011	PT_0011	08/12/2016		09/12/2016	ROOM011
AD_00012	PT_0012	02/17/2018		03/17/2018	ROOM012
AD_00013	PT_0013	05/10/2015		05/11/2015	ROOM013
AD_00014	PT_0014	09/01/2016		10/01/2016	ROOM014
AD_00015	PT_0015	10/23/2015		11/23/2015	ROOM015
AD_00016	PT_0016	11/24/2017		12/24/2017	ROOM016
AD_00017	PT_0017	08/12/2016		09/12/2016	ROOM017
AD_00018	PT_0018	02/17/2018	05/10/2018	03/17/2018	ROOM018
AD_00019	PT_0019	05/10/2015		05/11/2015	ROOM019
AD_00020	PT_0020	09/01/2016		10/01/2016	ROOM020
AD_00021	PT_0021	10/23/2015		11/23/2015	ROOM021
AD_00022	PT_0022	11/24/2017		12/24/2017	ROOM022
AD_00023	PT_0023	08/12/2016		09/12/2016	ROOM023
AD_00024	PT_0024	02/17/2018		03/17/2018	ROOM024
AD_00025	PT_0025	05/10/2015		05/11/2015	ROOM025
AD_00026	PT_0026	09/01/2016		10/01/2016	ROOM026

We now, choose the columns.



Choose Columns

Select the columns to import from the data set and arrange them in the order you want.

Available Columns

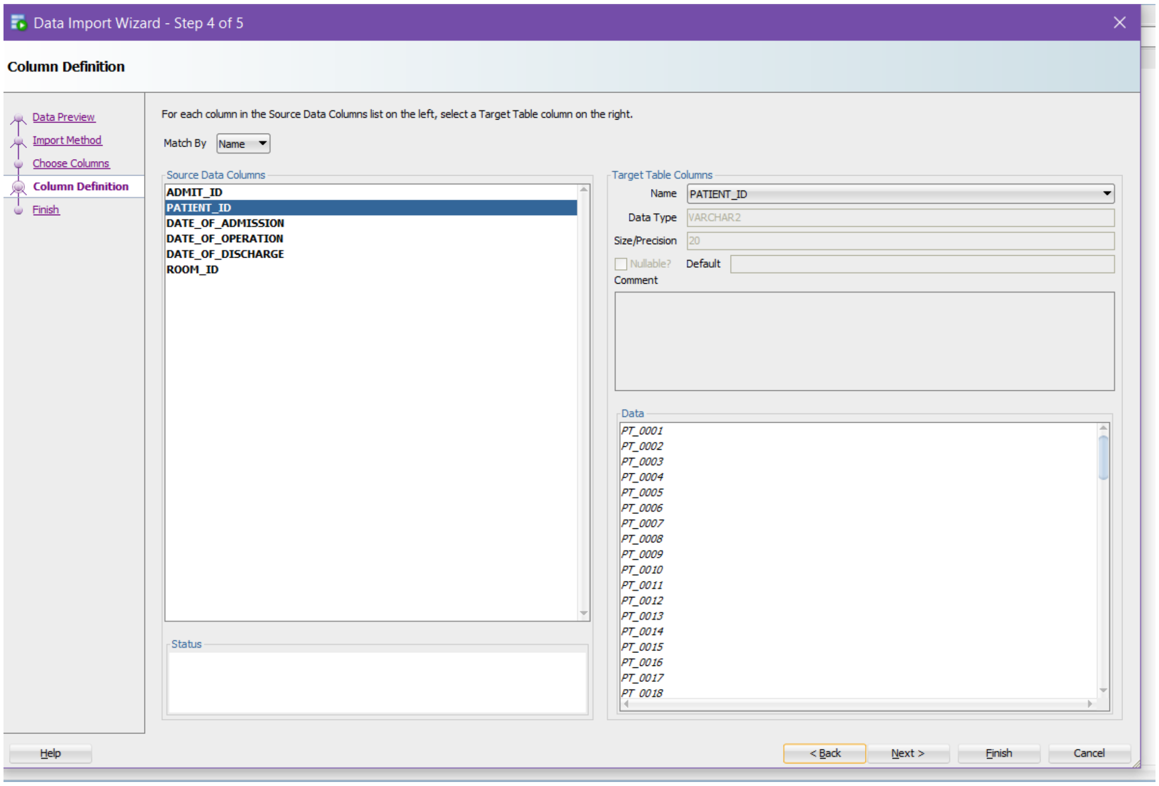
Selected Columns

ADMIT_ID
PATIENT_ID
DATE_OF_ADMISSION
DATE_OF_OPERATION
DATE_OF_DISCHARGE
ROOM_ID

File Contents

ADMIT_ID	PATIENT_ID	DATE_OF_A...	DATE_OF_...	DATE_OF_D...	ROOM_ID
AD_00001	PT_0001	05/10/2015		05/11/2015	ROOM001
AD_00002	PT_0002	09/01/2016		10/01/2016	ROOM002
AD_00003	PT_0003	10/23/2015		11/23/2015	ROOM003

Now, we decide on column definitions.



Data Import Wizard - Step 4 of 5

Column Definition

For each column in the Source Data Columns list on the left, select a Target Table column on the right.

Match By: Name

Source Data Columns:

```
ADMIT_ID
PATIENT_ID
DATE_OF_ADMISSION
DATE_OF_OPERATION
DATE_OF_DISCHARGE
ROOM_ID
```

Target Table Columns

Name: PATIENT_ID

Data Type: VARCHAR2

Size/Precision: 20

□ Nullable? Default:

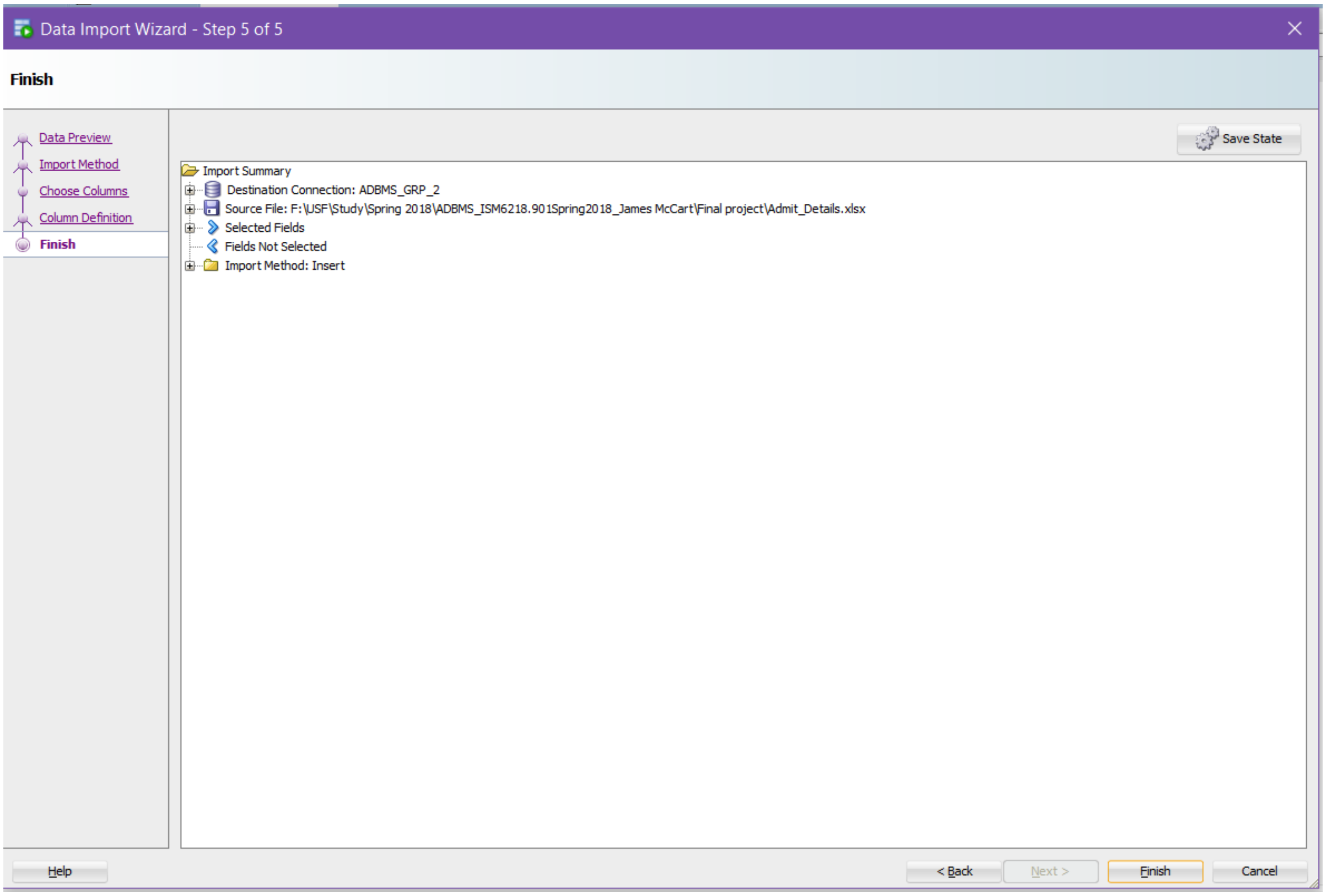
Comment:

Status:

Data:

```
PT_0001
PT_0002
PT_0003
PT_0004
PT_0005
PT_0006
PT_0007
PT_0008
PT_0009
PT_0010
PT_0011
PT_0012
PT_0013
PT_0014
PT_0015
PT_0016
PT_0017
PT_0018
```

Now, we select “Finish” and the file is imported.

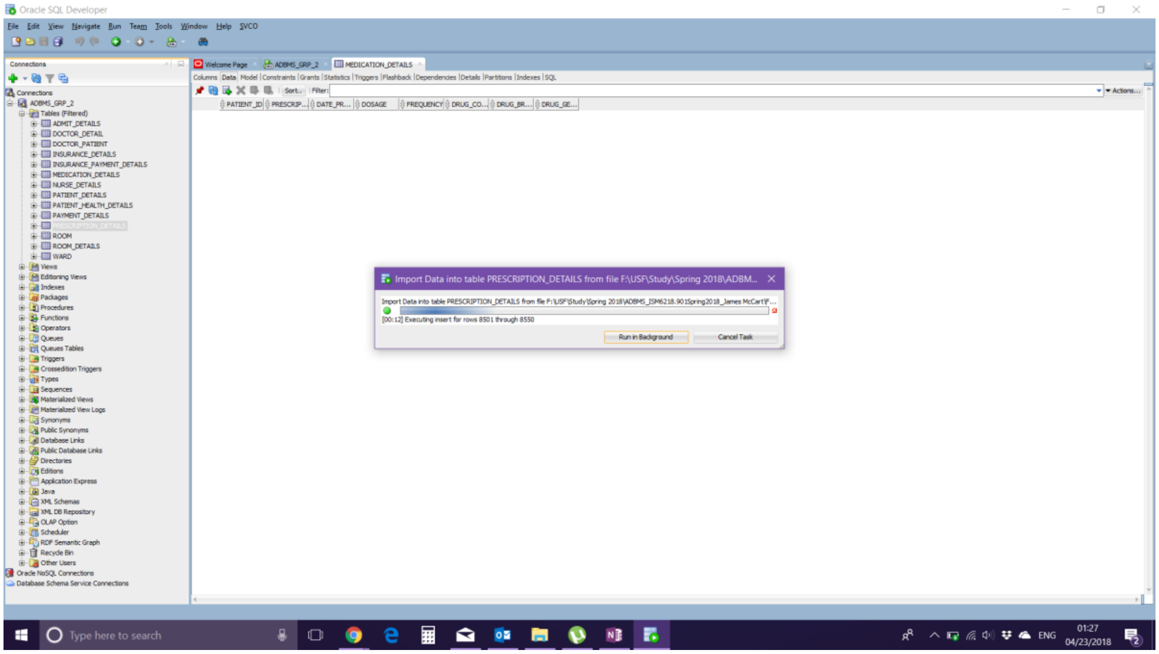


Data Import Wizard - Step 5 of 5

Finish

Import Summary
- Destination Connection: ADBMS_GRP_2
- Source File: F:\USF\Study\Spring 2018\ADBMS_ISM6218.901Spring2018_James McCart\Final project\Admit_Details.xlsx
- Selected Fields
- Fields Not Selected
- Import Method: Insert

The file is successfully imported into the database,

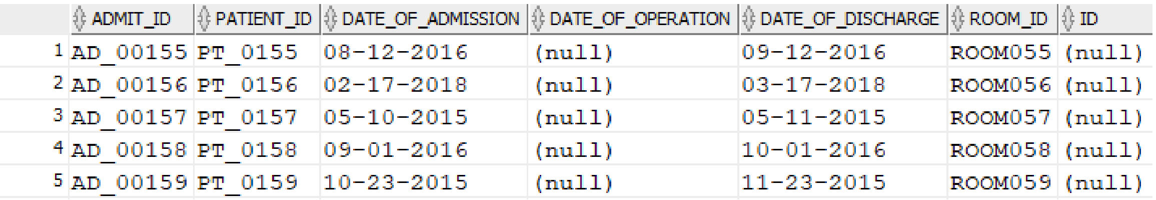


Import Data into table PRESCRIPTION_DETAILS from file F:\USF\Study\Spring 2018\ADBM...

Import Data into table PRESCRIPTION_DETAILS from file F:\USF\Study\Spring 2018\ADBMS_ISM6218.901Spring2018_James McCart\F...

[00:12] Executing insert for rows 8501 through 8550

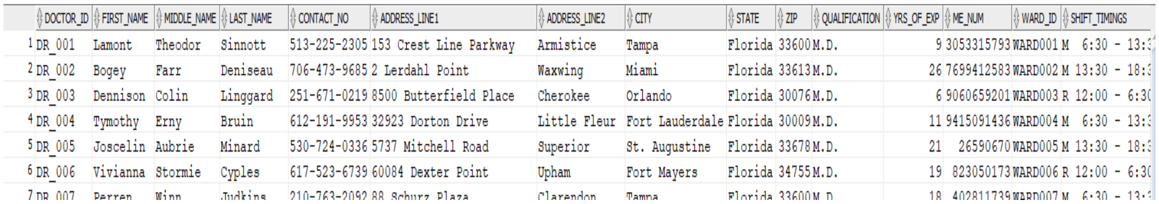
Run in Background | Cancel Task



	ADMIT_ID	PATIENT_ID	DATE_OF_ADMISSION	DATE_OF_OPERATION	DATE_OF_DISCHARGE	ROOM_ID	ID
1	AD_00155	PT_0155	08-12-2016	(null)	09-12-2016	ROOM055	(null)
2	AD_00156	PT_0156	02-17-2018	(null)	03-17-2018	ROOM056	(null)
3	AD_00157	PT_0157	05-10-2015	(null)	05-11-2015	ROOM057	(null)
4	AD_00158	PT_0158	09-01-2016	(null)	10-01-2016	ROOM058	(null)
5	AD_00159	PT_0159	10-23-2015	(null)	11-23-2015	ROOM059	(null)

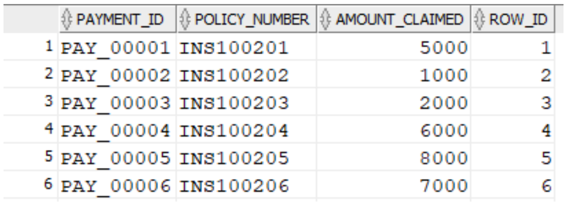
Similarly, for every file, the data gets imported.

2. DOCTOR_DETAIL



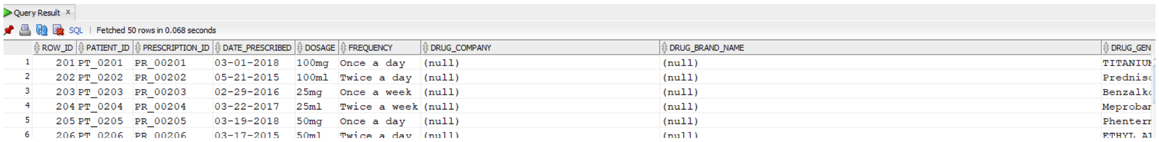
	DOCTOR_ID	FIRST_NAME	MIDDLE_NAME	LAST_NAME	CONTACT_NO	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	ZIP	QUALIFICATION	YRS_OF_EXP	ME_NUM	WARD_ID	SHIFT_TIMINGS
1	DR_001	Lamont	Theodor	Sinnott	513-225-2305	153 Crest Line Parkway	Armistice	Tampa	Florida	33600	M.D.	9	3053315793	WARD001	M 6:30 - 13:3
2	DR_002	Bogey	Farr	Deniseau	706-473-9685	2 Lerdahl Point	Waxwing	Miami	Florida	33613	M.D.	26	7699412583	WARD002	M 13:30 - 18:3
3	DR_003	Dennison	Colin	Linggard	251-671-0219	8500 Butterfield Place	Cherokee	Orlando	Florida	30076	M.D.	6	9060659201	WARD003	R 12:00 - 6:3
4	DR_004	Tymothy	Erny	Bruin	612-191-9953	32923 Dorton Drive	Little Fleur	Fort Lauderdale	Florida	30009	M.D.	11	9415091436	WARD004	M 6:30 - 13:3
5	DR_005	Joscelin	Aubrie	Minard	530-724-0336	5737 Mitchell Road	Superior	St. Augustine	Florida	33678	M.D.	21	26590670	WARD005	M 13:30 - 18:3
6	DR_006	Vivianna	Stormie	Cyples	617-523-6739	60084 Dexter Point	Upham	Fort Mayers	Florida	34755	M.D.	19	823050173	WARD006	R 12:00 - 6:3
7	DR_007	Perren	Winn	Judkins	210-763-2092	88 Schurz Plaza	Clarendon	Tampa	Florida	33600	M.D.	18	402811739	WARD007	M 6:30 - 13:3

3. DOCTOR_PATIENT_DETAILS



	PAYMENT_ID	POLICY_NUMBER	AMOUNT_CLAIMED	ROW_ID
1	PAY_00001	INS100201	5000	1
2	PAY_00002	INS100202	1000	2
3	PAY_00003	INS100203	2000	3
4	PAY_00004	INS100204	6000	4
5	PAY_00005	INS100205	8000	5
6	PAY_00006	INS100206	7000	6

4. INSURANCE_DETAILS

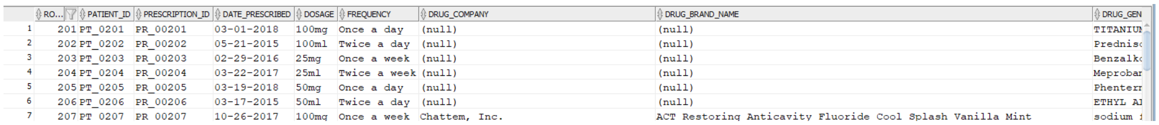


Query Result ×

SQL | Fetched 50 rows in 0.068 seconds

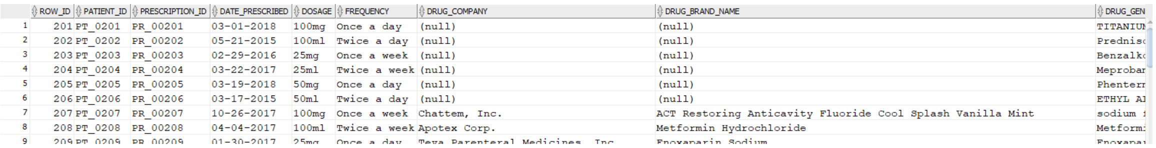
	ROW_ID	PATIENT_ID	PRESCRIPTION_ID	DATE_PRESCRIBED	DOSAGE	FREQUENCY	DRUG_COMPANY	DRUG_BRAND_NAME	DRUG_GEN
1	201	PT_0201	PR_00201	03-01-2018	100mg	Once a day	(null)	(null)	TITANIUM
2	202	PT_0202	PR_00202	05-21-2015	100ml	Twice a day	(null)	(null)	Predniso
3	203	PT_0203	PR_00203	02-29-2016	25mg	Once a week	(null)	(null)	Benzalko
4	204	PT_0204	PR_00204	03-22-2017	25ml	Twice a week	(null)	(null)	Meprobar
5	205	PT_0205	PR_00205	03-19-2018	50mg	Once a day	(null)	(null)	Phenterr
6	206	PT_0206	PR_00206	03-17-2015	50ml	Twice a day	(null)	(null)	ETHYL AI

5. INSURANCE_PAYMENT_DETAILS



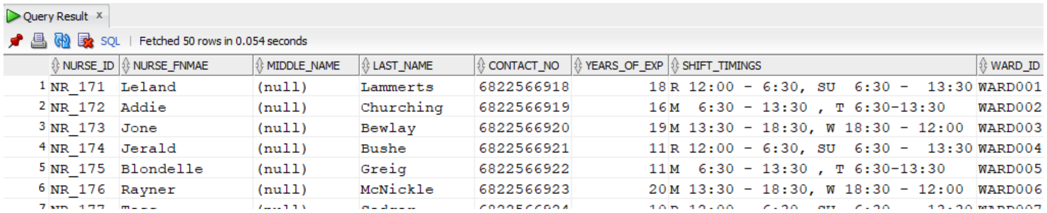
	RO...	PATIENT_ID	PRESCRIPTION_ID	DATE_PRESCRIBED	DOSAGE	FREQUENCY	DRUG_COMPANY	DRUG_BRAND_NAME	DRUG_GEN
1	201	PT_0201	PR_00201	03-01-2018	100mg	Once a day	(null)	(null)	TITANIUM
2	202	PT_0202	PR_00202	05-21-2015	100ml	Twice a day	(null)	(null)	Predniso
3	203	PT_0203	PR_00203	02-29-2016	25mg	Once a week	(null)	(null)	Benzalko
4	204	PT_0204	PR_00204	03-22-2017	25ml	Twice a week	(null)	(null)	Meprobar
5	205	PT_0205	PR_00205	03-19-2018	50mg	Once a day	(null)	(null)	Phenterr
6	206	PT_0206	PR_00206	03-17-2015	50ml	Twice a day	(null)	(null)	ETHYL AI
7	207	PT_0207	PR_00207	10-26-2017	100mg	Once a week	Chattem, Inc.	ACT Restoring Anticavity Fluoride Cool Splash Vanilla Mint	sodium f

6. MEDICATION_DETAILS



	ROW_ID	PATIENT_ID	PRESCRIPTION_ID	DATE_PRESCRIBED	DOSAGE	FREQUENCY	DRUG_COMPANY	DRUG_BRAND_NAME	DRUG_GEN
1	201	PT_0201	PR_00201	03-01-2018	100mg	Once a day	(null)	(null)	TITANIUM
2	202	PT_0202	PR_00202	05-21-2015	100ml	Twice a day	(null)	(null)	Predniso
3	203	PT_0203	PR_00203	02-29-2016	25mg	Once a week	(null)	(null)	Benzalko
4	204	PT_0204	PR_00204	03-22-2017	25ml	Twice a week	(null)	(null)	Meprobar
5	205	PT_0205	PR_00205	03-19-2018	50mg	Once a day	(null)	(null)	Phenterr
6	206	PT_0206	PR_00206	03-17-2015	50ml	Twice a day	(null)	(null)	ETHYL AI
7	207	PT_0207	PR_00207	10-26-2017	100mg	Once a week	Chattem, Inc.	ACT Restoring Anticavity Fluoride Cool Splash Vanilla Mint	sodium f
8	208	PT_0208	PR_00208	04-04-2017	100ml	Twice a week	Apotex Corp.	Metformin Hydrochloride	Metformi
9	209	PT_0209	PR_00209	01-30-2017	25mg	Once a day	Teva Parenteral Medicines, Inc	Enoxaparin Sodium	Enoxapar

7. NURSE_DETAILS

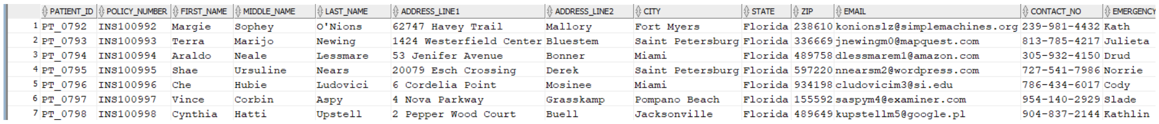


Query Result ×

SQL | Fetched 50 rows in 0.054 seconds

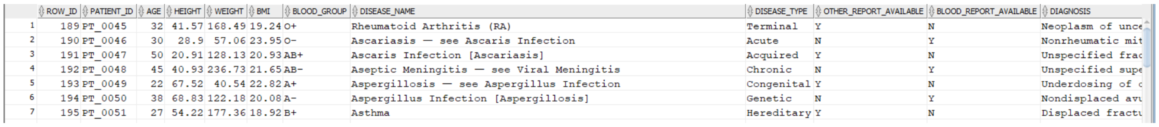
	NURSE_ID	NURSE_FNMAE	MIDDLE_NAME	LAST_NAME	CONTACT_NO	YEARS_OF_EXP	SHIFT_TIMINGS	WARD_ID
1	NR_171	Leland	(null)	Lammerts	6822566918	18	R 12:00 - 6:30, SU 6:30 - 13:30	WARD001
2	NR_172	Addie	(null)	Churching	6822566919	16	M 6:30 - 13:30 , T 6:30-13:30	WARD002
3	NR_173	Jone	(null)	Bewlay	6822566920	19	M 13:30 - 18:30, W 18:30 - 12:00	WARD003
4	NR_174	Jerald	(null)	Bushe	6822566921	11	R 12:00 - 6:30, SU 6:30 - 13:30	WARD004
5	NR_175	Blondelle	(null)	Greig	6822566922	11	M 6:30 - 13:30 , T 6:30-13:30	WARD005
6	NR_176	Rayner	(null)	McNickle	6822566923	20	M 13:30 - 18:30, W 18:30 - 12:00	WARD006
7	NR_177	[illegible]	(null)	[illegible]	6822566924	10	R 12:00 - 6:30, SU 6:30 - 13:30	WARD007

8. PATIENT_DETAILS



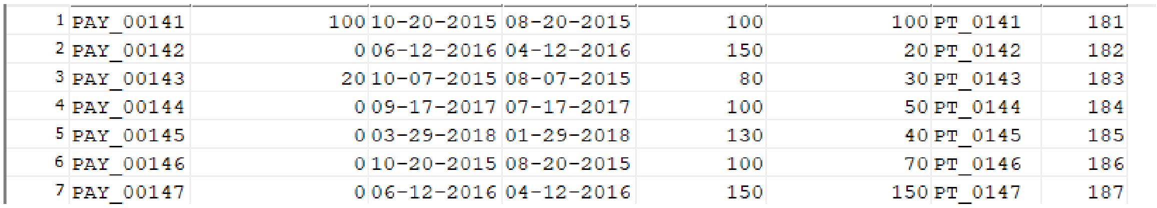
	PATIENT_ID	POLICY_NUMBER	FIRST_NAME	MIDDLE_NAME	LAST_NAME	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	ZIP	EMAIL	CONTACT_NO	EMERGENCY
1	PT_0792	INS100992	Margie	Sophey	O'Nions	62747 Havey Trail	Mallory	Fort Myers	Florida	238610	konionslz@simplemachines.org	239-981-4432	Kath
2	PT_0793	INS100993	Terra	Marijo	Newing	1424 Westerfield Center	Bluestem	Saint Petersburg	Florida	336669	jnewingm0@mapquest.com	813-785-4217	Julieta
3	PT_0794	INS100994	Araldo	Neale	Lessmare	53 Jenifer Avenue	Bonner	Miami	Florida	489758	dlessmarem1@amazon.com	305-932-4150	Drud
4	PT_0795	INS100995	Shae	Ursuline	Nears	20079 Esch Crossing	Derek	Saint Petersburg	Florida	597220	nnearsm2@wordpress.com	727-541-7986	Norrie
5	PT_0796	INS100996	Che	Hubie	Ludovici	6 Cordelia Point	Mosinee	Miami	Florida	934198	cludovicim3@si.edu	786-434-6017	Cody
6	PT_0797	INS100997	Vince	Corbin	Aspy	4 Nova Parkway	Grasskamp	Pompano Beach	Florida	155592	saspym4@examiner.com	954-140-2929	Slade
7	PT_0798	INS100998	Cynthia	Hatti	Upstell	2 Pepper Wood Court	Buell	Jacksonville	Florida	489649	kupstellm5@google.pl	904-837-2144	Kathlin

9. PATIENT_HEALTH_DETAILS



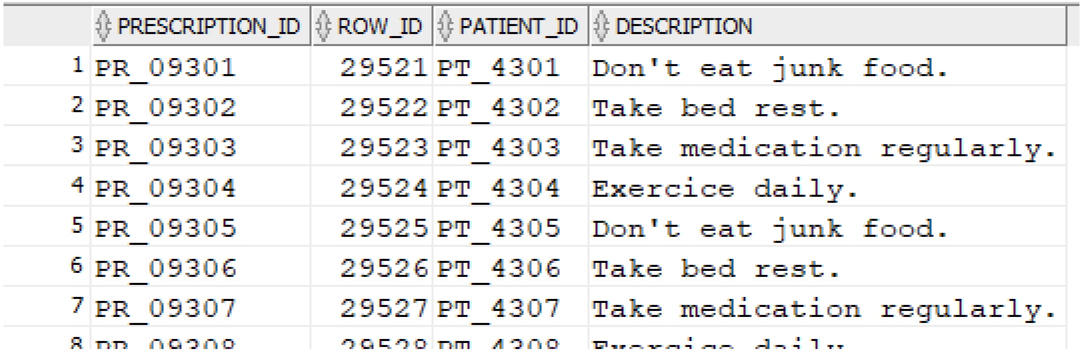
	ROW_ID	PATIENT_ID	AGE	HEIGHT	WEIGHT	BMI	BLOOD_GROUP	DISEASE_NAME	DISEASE_TYPE	OTHER_REPORT_AVAILABLE	BLOOD_REPORT_AVAILABLE	DIAGNOSIS
1	189	PT_0045	32	41.57	168.49	19.24	O+	Rheumatoid Arthritis (RA)	Terminal	Y	N	Neoplasm of unce
2	190	PT_0046	30	28.9	57.06	23.95	O-	Ascariasis — see Ascaris Infection	Acute	N	Y	Nonrheumatic mit
3	191	PT_0047	50	20.91	128.13	20.93	AB+	Ascaris Infection [Ascariasis]	Acquired	Y	N	Unspecified frac
4	192	PT_0048	45	40.93	236.73	21.65	AB-	Aseptic Meningitis — see Viral Meningitis	Chronic	N	Y	Unspecified supe
5	193	PT_0049	22	67.52	40.54	22.82	A+	Aspergillosis — see Aspergillus Infection	Congenital	Y	N	Underdosing of c
6	194	PT_0050	38	68.83	122.18	20.08	A-	Aspergillus Infection [Aspergillosis]	Genetic	N	Y	Nondisplaced avu
7	195	PT_0051	27	54.22	177.36	18.92	B+	Asthma	Hereditary	Y	N	Displaced fractu

10. PAYMENT_DETAILS



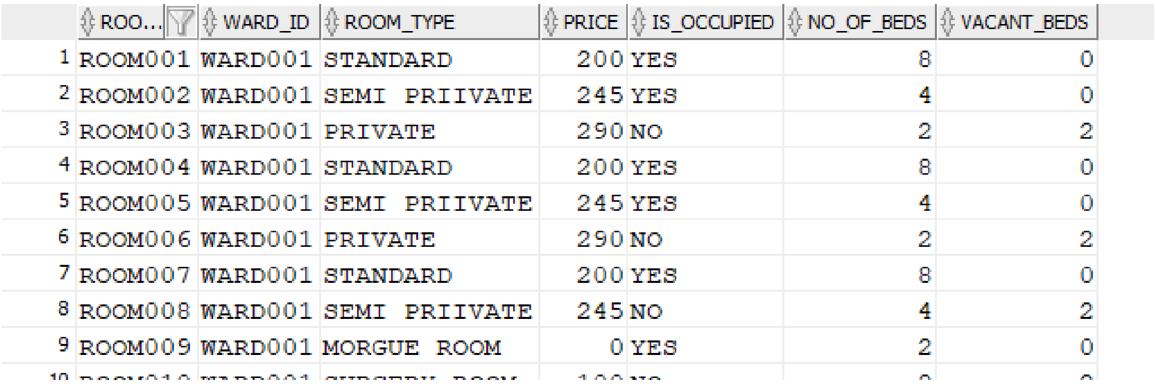
1	PAY_00141	100	10-20-2015	08-20-2015	100	100	PT_0141	181
2	PAY_00142	0	06-12-2016	04-12-2016	150	20	PT_0142	182
3	PAY_00143	20	10-07-2015	08-07-2015	80	30	PT_0143	183
4	PAY_00144	0	09-17-2017	07-17-2017	100	50	PT_0144	184
5	PAY_00145	0	03-29-2018	01-29-2018	130	40	PT_0145	185
6	PAY_00146	0	10-20-2015	08-20-2015	100	70	PT_0146	186
7	PAY_00147	0	06-12-2016	04-12-2016	150	150	PT_0147	187

11. PRESCRIPTION_DETAILS



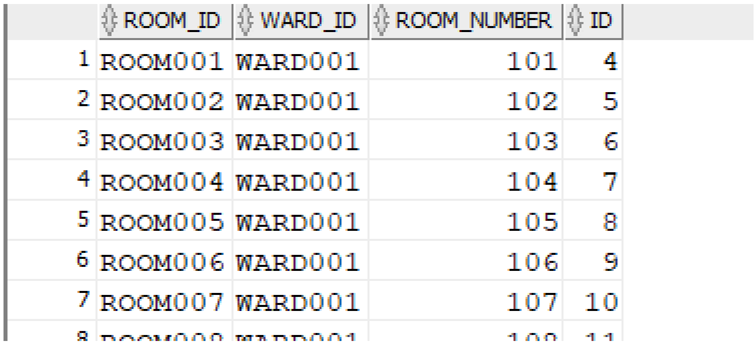
	PRESCRIPTION_ID	ROW_ID	PATIENT_ID	DESCRIPTION
1	PR_09301	29521	PT_4301	Don't eat junk food.
2	PR_09302	29522	PT_4302	Take bed rest.
3	PR_09303	29523	PT_4303	Take medication regularly.
4	PR_09304	29524	PT_4304	Exercice daily.
5	PR_09305	29525	PT_4305	Don't eat junk food.
6	PR_09306	29526	PT_4306	Take bed rest.
7	PR_09307	29527	PT_4307	Take medication regularly.
8	PR_09308	29528	PT_4308	Exercice daily.

12. ROOM_DETAILS



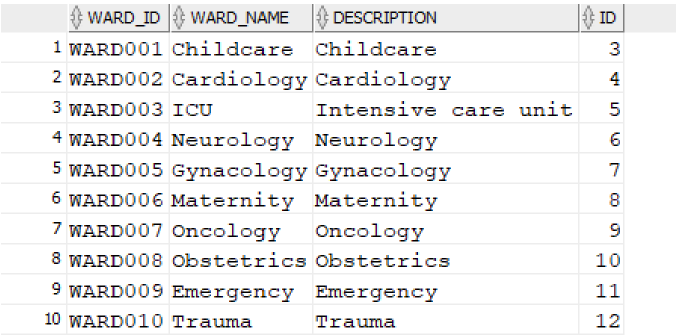
	ROO...	WARD_ID	ROOM_TYPE	PRICE	IS_OCCUPIED	NO_OF_BEDS	VACANT_BEDS
1	ROOM001	WARD001	STANDARD	200	YES	8	0
2	ROOM002	WARD001	SEMI PRIIVATE	245	YES	4	0
3	ROOM003	WARD001	PRIVATE	290	NO	2	2
4	ROOM004	WARD001	STANDARD	200	YES	8	0
5	ROOM005	WARD001	SEMI PRIIVATE	245	YES	4	0
6	ROOM006	WARD001	PRIVATE	290	NO	2	2
7	ROOM007	WARD001	STANDARD	200	YES	8	0
8	ROOM008	WARD001	SEMI PRIIVATE	245	NO	4	2
9	ROOM009	WARD001	MORGUE ROOM	0	YES	2	0
10	ROOM010	WARD001	SURGERY ROOM	100	NO	2	0

13. ROOM_WARD_DETAILS



	ROOM_ID	WARD_ID	ROOM_NUMBER	ID
1	ROOM001	WARD001	101	4
2	ROOM002	WARD001	102	5
3	ROOM003	WARD001	103	6
4	ROOM004	WARD001	104	7
5	ROOM005	WARD001	105	8
6	ROOM006	WARD001	106	9
7	ROOM007	WARD001	107	10
8	ROOM008	WARD001	108	11

14. WARD_DETAILS



	WARD_ID	WARD_NAME	DESCRIPTION	ID
1	WARD001	Childcare	Childcare	3
2	WARD002	Cardiology	Cardiology	4
3	WARD003	ICU	Intensive care unit	5
4	WARD004	Neurology	Neurology	6
5	WARD005	Gynacology	Gynacology	7
6	WARD006	Maternity	Maternity	8
7	WARD007	Oncology	Oncology	9
8	WARD008	Obstetrics	Obstetrics	10
9	WARD009	Emergency	Emergency	11
10	WARD010	Trauma	Trauma	12

VII. Performance Tuning

a. Indexing

INDEXES help to increase performance at the time of search and modification into table records.
With the help of Index, data can be located quickly, without having to search every row in a database table

i. B- Tree Index

A B-Tree index is in the form of binary tree and is the default index type. It is default index type.

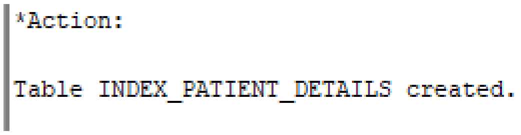
Creating Unique Index

This is same as primary constraint and is created on a Column of a database table.

Column having Unique index won't have duplicate rows

Let's create a table "INDEX" to very Unique Index functioning.

```
CREATE TABLE INDEX
as SELECT * FROM PATIENT_DETAILS;
```

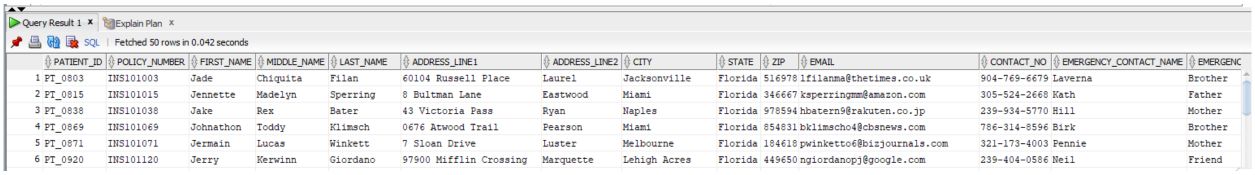


```
*Action:

Table INDEX_PATIENT_DETAILS created.
```

Querying all the Patients whose first name start with 'J'.

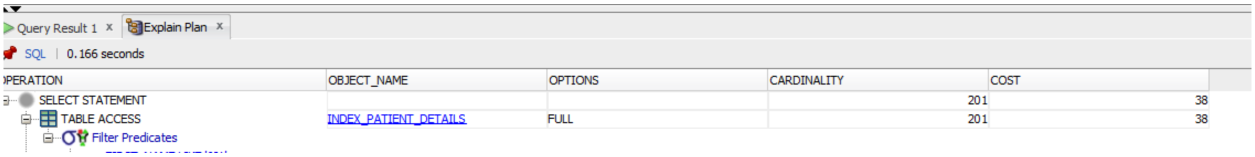
Before Setting Unique constraint,

SELECT * FROM INDEX_PATIENT_DETAILS where FIRST_NAME LIKE 'J%';



Fetched 50 rows in 0.042 seconds

	PATIENT_ID	POLICY_NUMBER	FIRST_NAME	MIDDLE_NAME	LAST_NAME	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	ZIP	EMAIL	CONTACT_NO	EMERGENCY_CONTACT_NAME	EMERGENC
1	PT_0803	INS101003	Jade	Chiquita	Filan	60104 Russell Place	Laurel	Jacksonville	Florida	516978	lfilanma@thetimes.co.uk	904-769-6679	Laverna	Brother
2	PT_0815	INS101015	Jennette	Madelyn	Sperring	8 Bultman Lane	Eastwood	Miami	Florida	346667	ksperringmm@amazon.com	305-524-2668	Kath	Father
3	PT_0838	INS101038	Jake	Rex	Bater	43 Victoria Pass	Ryan	Naples	Florida	978594	hbatern9@rakuten.co.jp	239-934-5770	Hill	Mother
4	PT_0869	INS101069	Johnathon	Toddy	Klimsch	0676 Atwood Trail	Pearson	Miami	Florida	854831	bklimscho4@cbsnews.com	786-314-8596	Birk	Brother
5	PT_0871	INS101071	Jermain	Lucas	Winkett	7 Sloan Drive	Luster	Melbourne	Florida	184618	pwinketto6@bizjournals.com	321-173-4003	Pennie	Mother
6	PT_0920	INS101120	Jerry	Kerwinn	Giordano	97900 Mifflin Crossing	Marquette	Lehigh Acres	Florida	449650	ngiordanopj@google.com	239-404-0586	Neil	Friend



Query Result 1 × | Explain Plan ×

SQL | 0.166 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			201	38
TABLE ACCESS	INDEX_PATIENT_DETAILS	FULL	201	38
Filter Predicates				

Adding Unique index

ALTER TABLE INDEX_PATIENT_DETAILS

ADD CONSTRAINT PK_UNIQUEINDEX_NAME PRIMARY KEY(PATIENT_ID);

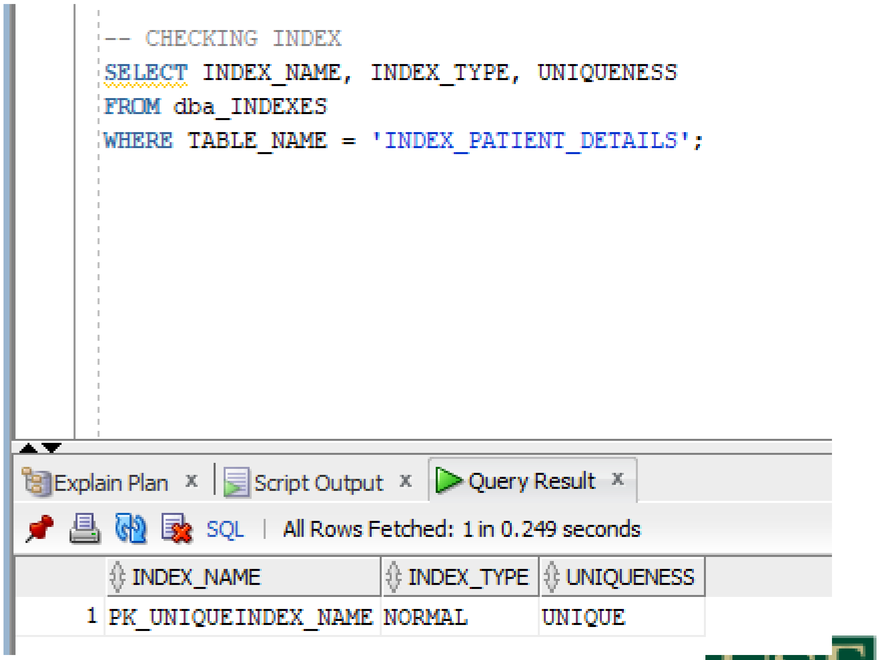
```
*Action:

Table INDEX_PATIENT_DETAILS altered.

'
```

Checking created index

SELECT INDEX_NAME, INDEX_TYPE, UNIQUENESS

FROM dba_INDEXES

WHERE TABLE_NAME = 'INDEX_PATIENT_DETAILS';



```
-- CHECKING INDEX
SELECT INDEX_NAME, INDEX_TYPE, UNIQUENESS
FROM dba_INDEXES
WHERE TABLE_NAME = 'INDEX_PATIENT_DETAILS';
```

Explain Plan × | Script Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0.249 seconds

	INDEX_NAME	INDEX_TYPE	UNIQUENESS
1	PK_UNIQUEINDEX_NAME	NORMAL	UNIQUE

Unique constraint behavior is same as primary key. All the value must have to be unique

Inserting duplicate value in table

INSERT INTO

INDEX_PATIENT_DETAILS(PATIENT_ID,POLICY_NUMBER,FIRST_NAME,

MIDDLE_NAME, LAST_NAME, ADDRESS_LINE1, ADDRESS_LINE2,

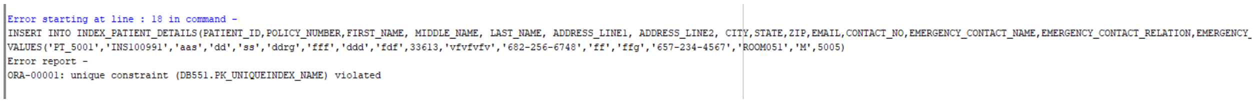
CITY,STATE,ZIP,EMAIL,CONTACT_NO,EMERGENCY_CONTACT_NAME,EMERGENCY_

CONTACT_RELATION,EMERGENCY_CONTACT_NO,ROOM_ID,GENDER,row_id)

VALUES('PT_5001','INS100991','aas','dd','ss','ddrg','fff','ddd','fdf',33613,'vfvfvfv','68

2-256-6748','ff','ffg','657-234-4567','ROOM051','M',5005);

Output : Index voilated



```
Error starting at line : 18 in command -
INSERT INTO INDEX_PATIENT_DETAILS(PATIENT_ID,POLICY_NUMBER,FIRST_NAME, MIDDLE_NAME, LAST_NAME, ADDRESS_LINE1, ADDRESS_LINE2, CITY,STATE,ZIP,EMAIL,CONTACT_NO,EMERGENCY_CONTACT_NAME,EMERGENCY_CONTACT_RELATION,EMERGENCY_
VALUES('PT_5001','INS100991','aas','dd','ss','ddrg','fff','ddd','fdf',33613,'vfvfvfv','682-256-6748','ff','ffg','657-234-4567','ROOM051','M',5005)
Error report -
ORA-00001: unique constraint (DB551.PK_UNIQUEINDEX_NAME) violated
```

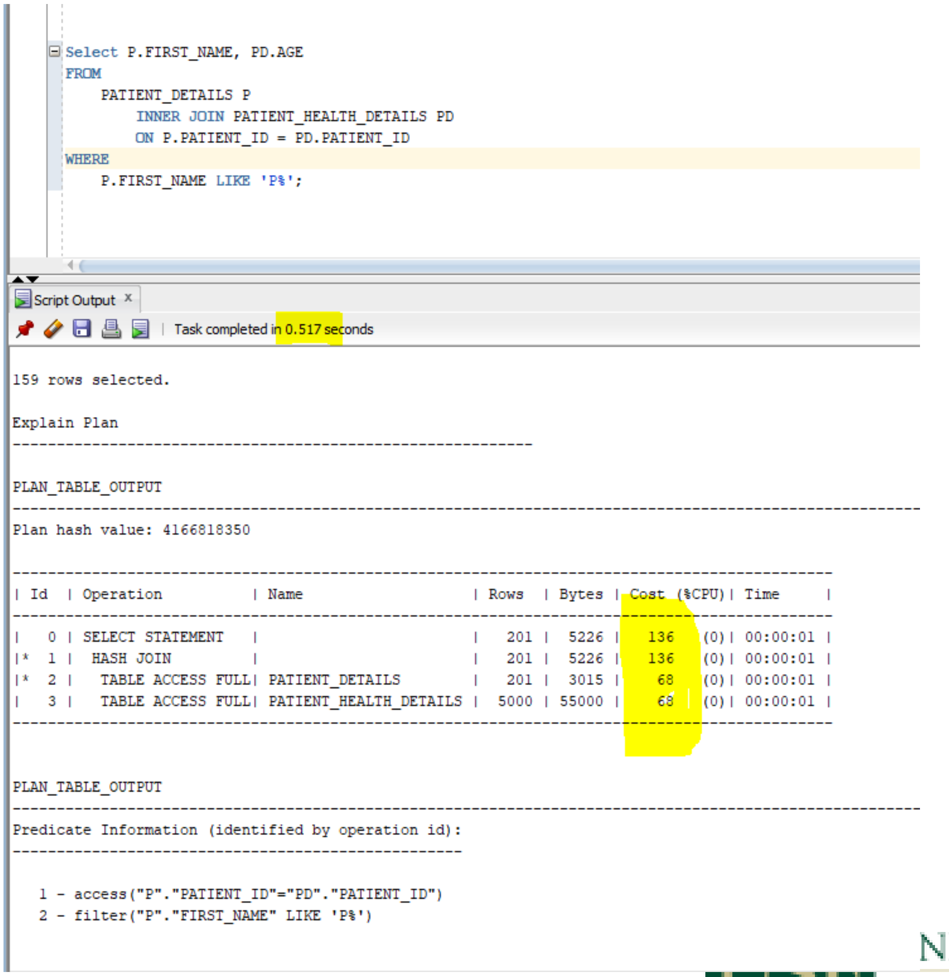
Example to show Time and Cost reduction by Indexing

The following query lists all the patients first name and Age whose first name start with P

The following query illustrates the use of B-tree index and its impact on execution plan and client statistics.

```
Select P.FIRST_NAME, PD.AGE

FROM

    PATIENT_DETAILS P

        INNER JOIN PATIENT_HEALTH_DETAILS PD

        ON P.PATIENT_ID = PD.PATIENT_ID

WHERE

    P.FIRST_NAME LIKE 'P%';
```

Time and Cost before indexing



```
Select P.FIRST_NAME, PD.AGE
FROM
    PATIENT_DETAILS P
        INNER JOIN PATIENT_HEALTH_DETAILS PD
        ON P.PATIENT_ID = PD.PATIENT_ID
WHERE
    P.FIRST_NAME LIKE 'P%';
```

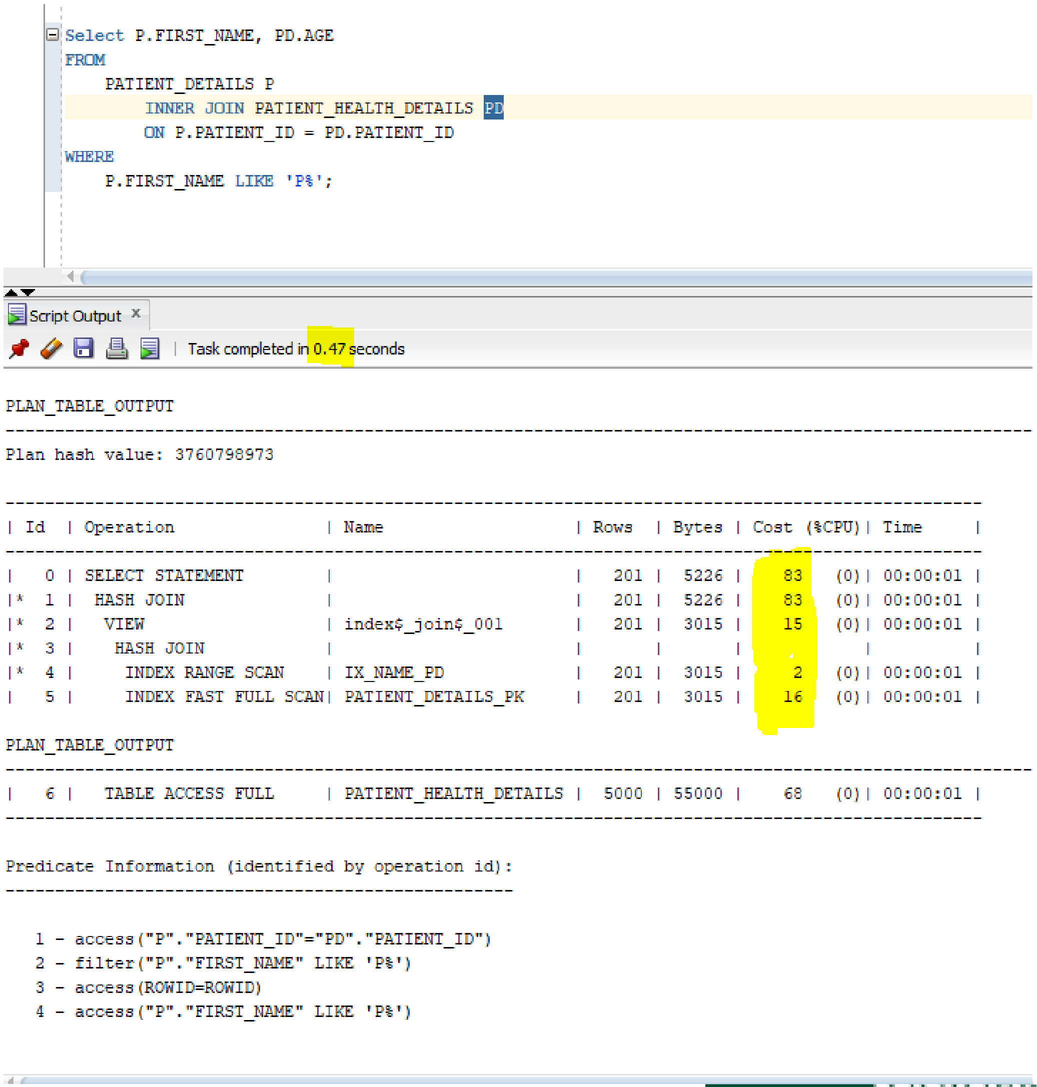
```
159 rows selected.

Explain Plan
-----------------------------------------------------------

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------
Plan hash value: 4166818350

---------------------------------------------------------------------------------------------
| Id  | Operation          | Name                   | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                        |   201 |  5226 |   136   (0)| 00:00:01 |
|*  1 |  HASH JOIN         |                        |   201 |  5226 |   136   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| PATIENT_DETAILS        |   201 |  3015 |    68   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| PATIENT_HEALTH_DETAILS |  5000 | 55000 |    68   (0)| 00:00:01 |
---------------------------------------------------------------------------------------------


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("P"."PATIENT_ID"="PD"."PATIENT_ID")
   2 - filter("P"."FIRST_NAME" LIKE 'P%')
```

Creating Index on column “FIRST_NAME”

CREATE INDEX IX_NAME_PD ON PATIENT_DETAILS (FIRST_NAME ASC);

After indexing:

Querying again to verify time and cost



```
Select P.FIRST_NAME, PD.AGE
FROM
    PATIENT_DETAILS P
        INNER JOIN PATIENT_HEALTH_DETAILS PD
        ON P.PATIENT_ID = PD.PATIENT_ID
WHERE
    P.FIRST_NAME LIKE 'P%';
```

Task completed in 0.47 seconds

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------
Plan hash value: 3760798973

---------------------------------------------------------------------------------------------
| Id  | Operation                | Name                   | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                        |   201 |  5226 |    83   (0)| 00:00:01 |
|*  1 |  HASH JOIN               |                        |   201 |  5226 |    83   (0)| 00:00:01 |
|*  2 |   VIEW                   | index$_join$_001       |   201 |  3015 |    15   (0)| 00:00:01 |
|*  3 |    HASH JOIN             |                        |       |       |            |          |
|*  4 |     INDEX RANGE SCAN     | IX_NAME_PD             |   201 |  3015 |     2   (0)| 00:00:01 |
|   5 |     INDEX FAST FULL SCAN| PATIENT_DETAILS_PK     |   201 |  3015 |    16   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------
|   6 |   TABLE ACCESS FULL      | PATIENT_HEALTH_DETAILS |  5000 | 55000 |    68   (0)| 00:00:01 |
---------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("P"."PATIENT_ID"="PD"."PATIENT_ID")
   2 - filter("P"."FIRST_NAME" LIKE 'P%')
   3 - access(ROWID=ROWID)
   4 - access("P"."FIRST_NAME" LIKE 'P%')
```

Highlighted part shows the difference in Time and Cost before and after indexing. It confirms Time and Cost of querying is reduced after adding Index

ii. Function Based Index

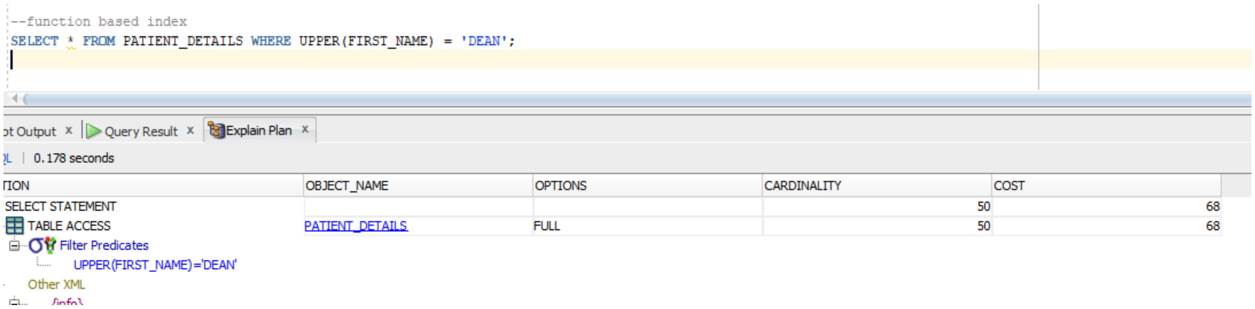
A function-based index, on the other hand, is an index that is created on the results of a function or expression. Normal index won't work on any function operator.

This index is required to increase the performance of query.

SELECT * FROM PATIENT_DETAILS WHERE UPPER(FIRST_NAME) =

'DEAN';

INDEX IX_NAME_PD which was created on FIRST_NAME column is not used here.



```
--function based index
SELECT * FROM PATIENT_DETAILS WHERE UPPER(FIRST_NAME) = 'DEAN';
```

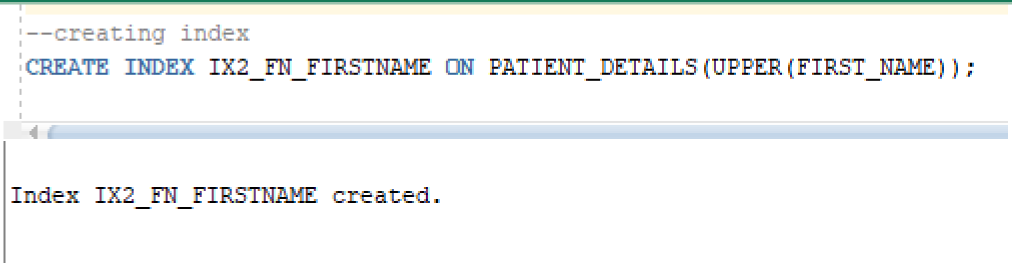
0.178 seconds

TION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			50	68
TABLE ACCESS	PATIENT_DETAILS	FULL	50	68
Filter Predicates				
UPPER(FIRST_NAME)='DEAN'				
Other XML				
{info}				

For a function (ex- UPPER) , we need to create index on function

Creating Function based index

CREATE INDEX IX2_FN_FIRSTNAME ON PATIENT_DETAILS(UPPER(FIRST_NAME));

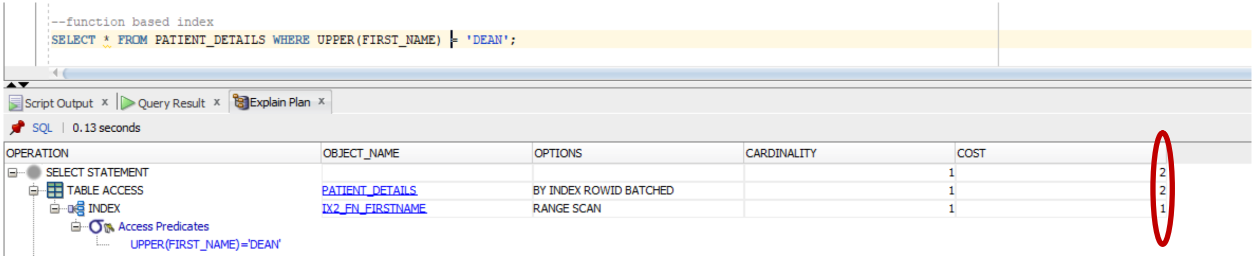


```
--creating index
CREATE INDEX IX2_FN_FIRSTNAME ON PATIENT_DETAILS(UPPER(FIRST_NAME));
```

```
Index IX2_FN_FIRSTNAME created.
```

Querying again

This time, index is used and cost and time is also reduced.



```
--function based index
SELECT * FROM PATIENT_DETAILS WHERE UPPER(FIRST_NAME) = 'DEAN';
```

0.13 seconds

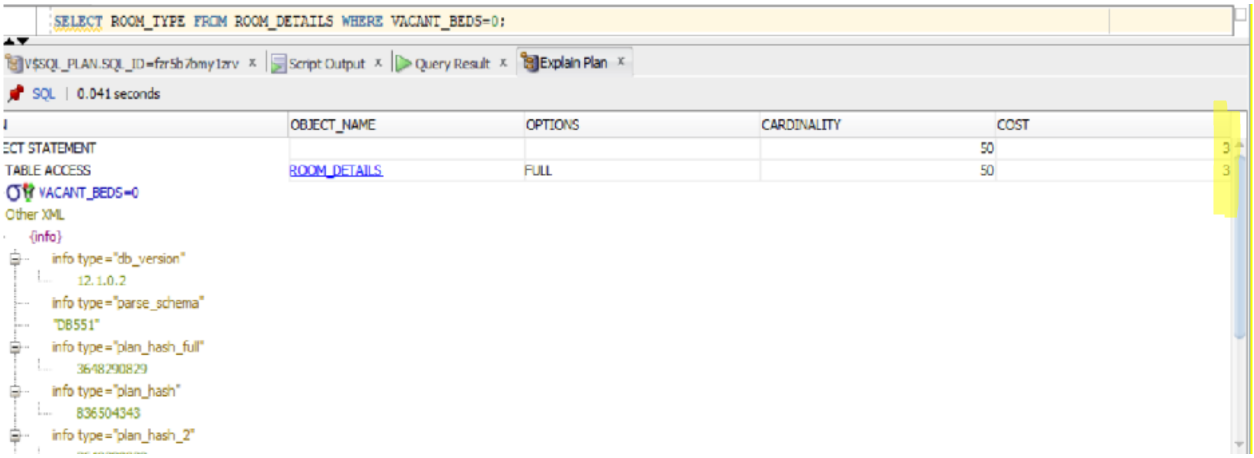
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
TABLE ACCESS	PATIENT_DETAILS	BY INDEX ROWID BATCHED	1	2
INDEX	IX2_FN_FIRSTNAME	RANGE SCAN	1	1
Access Predicates				
UPPER(FIRST_NAME)='DEAN'				

iii. Bitmap Index

Bitmap Index are used on column where number of distinct value in column is less.

SELECT ROOM_TYPE FROM ROOM_DETAILS WHERE VACANT_BEDS=0;

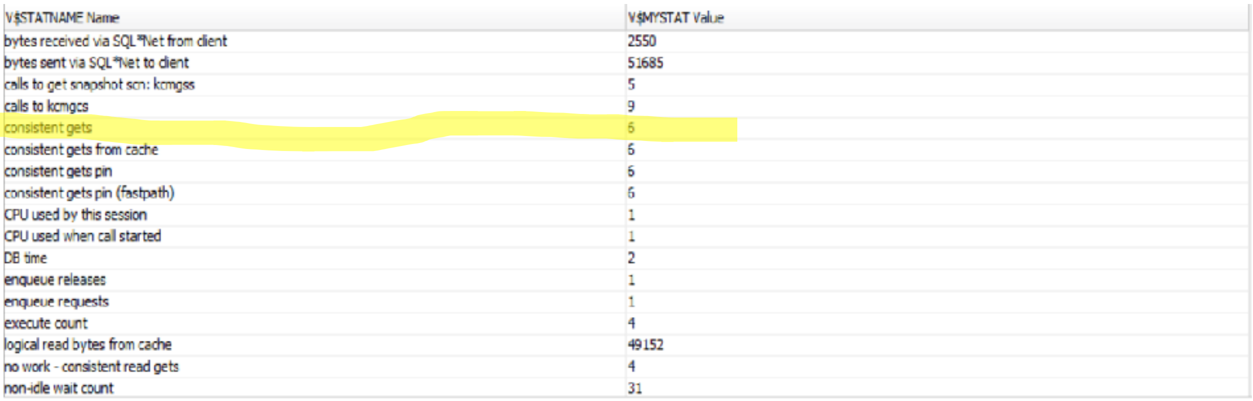
The following results were obtained before indexing:



```
SELECT ROOM_TYPE FROM ROOM_DETAILS WHERE VACANT_BEDS=0;
```

0.041 seconds

	OBJECT_NAME	OPTIONS	CARDINALITY	COST
ECT STATEMENT			50	3
TABLE ACCESS	ROOM_DETAILS	FULL	50	3
VACANT_BEDS=0				
Other XML				
{info}				
info type="db_version"				
12.1.0.2				
info type="parse_schema"				
"DB551"				
info type="plan_hash_full"				
3648290829				
info type="plan_hash"				
836504343				
info type="plan_hash_2"				



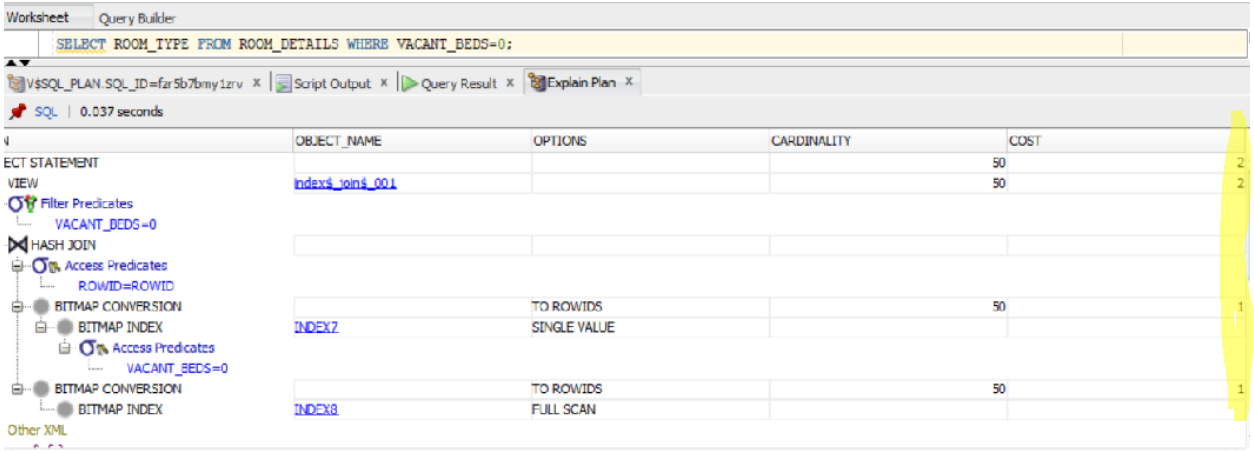
V$STATNAME Name	V$MYSTAT Value
bytes received via SQL*Net from client	2550
bytes sent via SQL*Net to client	51685
calls to get snapshot scn: kcmgss	5
calls to kcmgcs	9
consistent gets	6
consistent gets from cache	6
consistent gets pin	6
consistent gets pin (fastpath)	6
CPU used by this session	1
CPU used when call started	1
DB time	2
enqueue releases	1
enqueue requests	1
execute count	4
logical read bytes from cache	49152
no work - consistent read gets	4
non-idle wait count	31

We created bitmap index on both ROOM_TYPE and VACANT_BEDS as they both have low cardinality.

CREATE BITMAP INDEX INDEX7 ON ROOM_DETAILS(VACANT_BEDS)

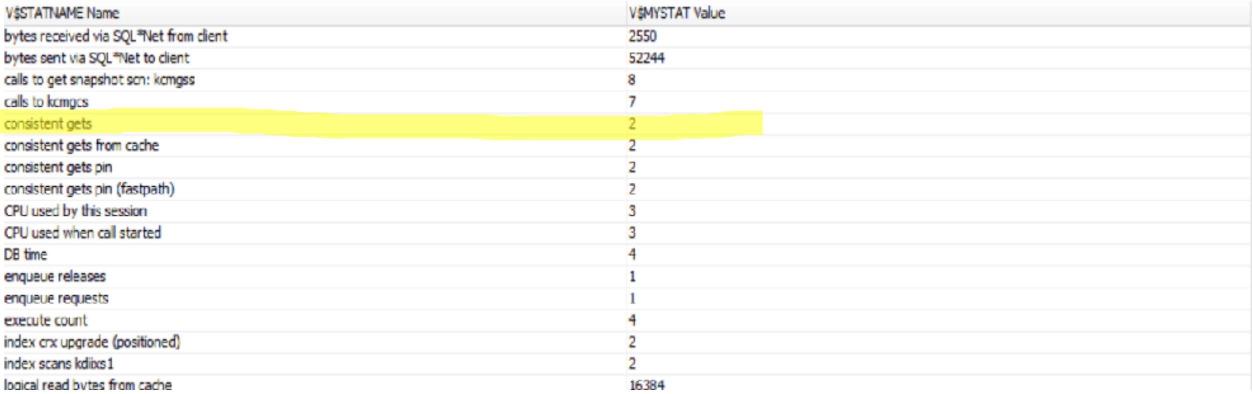
CREATE BITMAP INDEX INDEX8 ON ROOM_DETAILS (ROOM_TYPE)

The following results were obtained after indexing:



```
SELECT ROOM_TYPE FROM ROOM_DETAILS WHERE VACANT_BEDS=0;
```

N	OBJECT_NAME	OPTIONS	CARDINALITY	COST
ECT STATEMENT			50	2
VIEW	index$_join$_001		50	2
Filter Predicates				
VACANT_BEDS=0				
HASH JOIN				
Access Predicates				
ROWID=ROWID				
BITMAP CONVERSION		TO ROWIDS	50	1
BITMAP INDEX	INDEX7	SINGLE VALUE		
Access Predicates				
VACANT_BEDS=0				
BITMAP CONVERSION		TO ROWIDS	50	1
BITMAP INDEX	INDEX8	FULL SCAN		
Other XML				



V$STATNAME Name	V$MYSTAT Value
bytes received via SQL*Net from client	2550
bytes sent via SQL*Net to client	52244
calls to get snapshot scn: kcmgss	8
calls to kcmgcs	7
consistent gets	2
consistent gets from cache	2
consistent gets pin	2
consistent gets pin (fastpath)	2
CPU used by this session	3
CPU used when call started	3
DB time	4
enqueue releases	1
enqueue requests	1
execute count	4
index crx upgrade (positioned)	2
index scans kdiixs1	2
logical read bytes from cache	16384

The above proof shows the advantage of using bitmap indexes as there is significant reduction in the cost of executing the query.

Apart from this the number of consistent gets dropped from 6 to 2.

b. Query Tuning

Optimization is a key in production database. An efficient query will impact the performance or cause loss of service for other users. So optimization is important for least impact on Database performance.

Below are some of the best practices to write query

- Use SELECT field instead of SELECT *

  Inefficient:

  ```
  SELECT *
  FROM
          PATIENT_DETAILS
  ```

  Efficient:

```
SELECT
        FIRST_NAME,
        MIDDLE_NAME,
        LAST_NAME
FROM
        PATIENT_DETAILS
```

- Select more fields to avoid SELECT DISTINCT

Inefficient:

```
SELECT
        DISTINCT FIRST_NAME,
        LAST_NAME,
        STATE
FROM
        PATIENT_DETAILS
```

Efficient:

```
SELECT
        FIRST_NAME,
        LAST_NAME,
        ADDRESS_LINE1,
        CITY,
        STATE,
        ZIP
FROM
        PATIENT_DETAILS
```

- Use JOINS with INNER JOIN instead of WHERE

Inefficient:

```
SELECT
        PATIENT_DETAILS.FIRST_NAME,
        PATIENT_DETAILS.LAST_NAME,
        PATIENT_HEALTH_DETAILS.AGE
FROM
        PATIENT_DETAILS,PATIENT_HEALTH_DETAILS
WHERE
        PATIENT_DETAILS.PATIENT_ID =
PATIENT_HEALTH_DETAILS.PATIENT_ID
```

Efficient:

```
SELECT
        P.FIRST_NAME,
        P.PATIENT_DETAILS.LAST_NAME,
        PH.AGE
```

```
FROM
        PATIENT_DETAILS
                INNER JOIN PATIENT_HEALTH_DETAILS
                        ON P.PATIENT_ID = PH.PATIENT_ID
```

- Use WHERE to define filter

Inefficient:

```
SELECT
        P.FIRST_NAME,
        COUNT(A.ADMIT_ID)
FROM
        PATIENT_DETAILS P
                INNER JOIN ADMIT_DETAILS A
                        ON P.PATIENT_ID = A.PATIENT_ID
GROUP BY
        P.FIRST_NAME
HAVING
        A.DATE_OF_ADMISSION BETWEEN #1/1/2015# AND #12/31/2015#
```

Efficient:

```
SELECT
        P.FIRST_NAME,
        COUNT(A.ADMIT_ID)
FROM
        PATIENT_DETAILS P
                INNER JOIN ADMIT_DETAILS A
                        ON P.PATIENT_ID = A.PATIENT_ID
WHERE
        A.DATE_OF_ADMISSION BETWEEN #1/1/2015# AND #12/31/2015#
GROUP BY
        P.FIRST_NAME
```

- Minimize subquery usage

Inefficient:

```
SELECT
        FIRST_NAME
FROM
        PATIENT_DETAILS
WHERE
        AGE=
                (
                        SELECT
                                MAX(AGE)
```

```
            FROM
                PATIENT_HEALTH_DETAILS
        )
AND
    WEIGHT =
        (
            SELECT
                MAX(WEIGHT)
            FROM
                PATIENT_HEALTH_DETAILS
        )

AND
    CITY = ‘Miami’;
```

Efficient:

```
SELECT
    FIRST_NAME
FROM
    PATIENT_DETAILS
WHERE
    (AGE,WEIGHT) =
        (
            SELECT
                MAX(AGE),MAX(WEIGHT)
            FROM
                PATIENT_HEALTH_DETAILS
        )
AND
    CITY = ‘Miami’;
```

- Use DECODE to avoid duplicate scanning of same row

Inefficient:

```
SELECT
    DECODE(CITY, ’Tampa’, PATIENT_ID,NULL) PATIENT_ID
FROM
    PATIENT_DETAILS
WHERE
    FIRST_NAME LIKE ‘J%’;
```

Efficient:

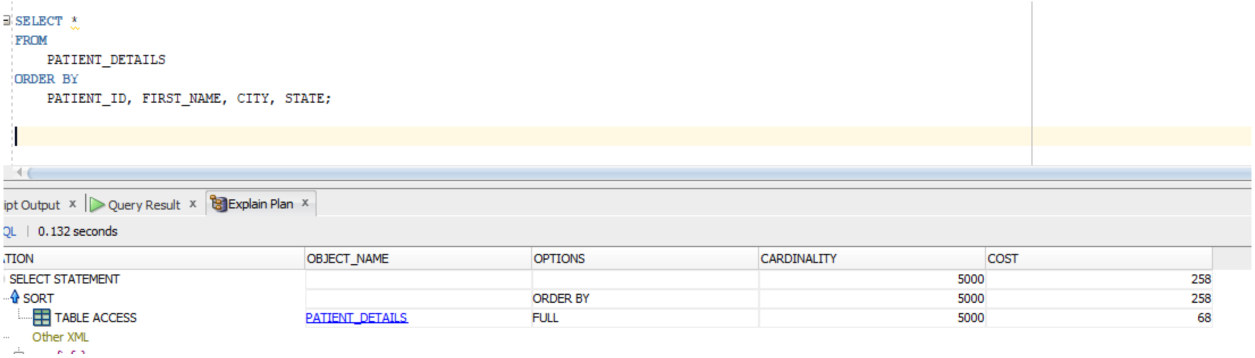
```
SELECT
    PATIENT_ID
FROM
```

```
          PATIENT_DETAILS
WHERE
          FIRST_NAME LIKE ‘j%’
AND
          CITY = ‘Tampa’;
```

c. Parallelism

For the queries, which results much records with order.

/* + PARALLEL */ works with SELECT and UPDATE statement

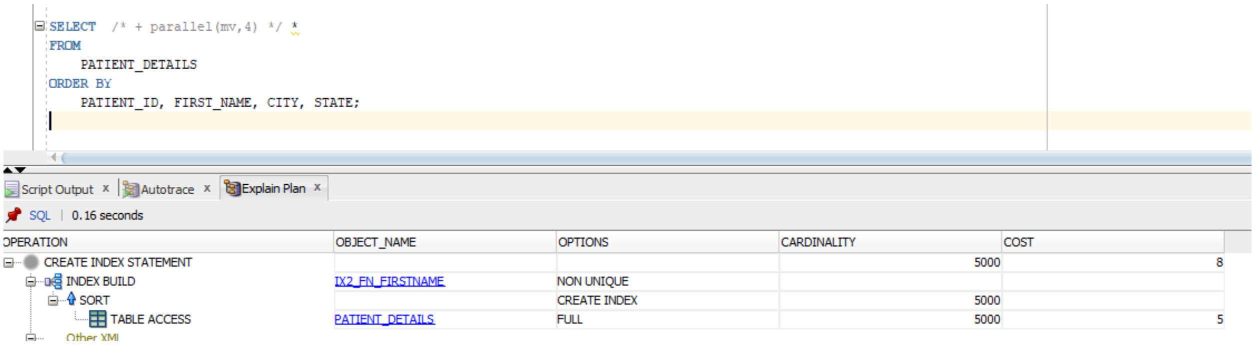


```
SELECT *
FROM
    PATIENT_DETAILS
ORDER BY
    PATIENT_ID, FIRST_NAME, CITY, STATE;
```

QL | 0.132 seconds

TION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5000	258
SORT		ORDER BY	5000	258
TABLE ACCESS	PATIENT_DETAILS	FULL	5000	68
Other XML				

Using parallelism we can see that time taken by the query reduces and is useful in retrieving large chunks of data.



```
SELECT  /* + parallel(mv,4) */ *
FROM
    PATIENT_DETAILS
ORDER BY
    PATIENT_ID, FIRST_NAME, CITY, STATE;
```

SQL | 0.16 seconds

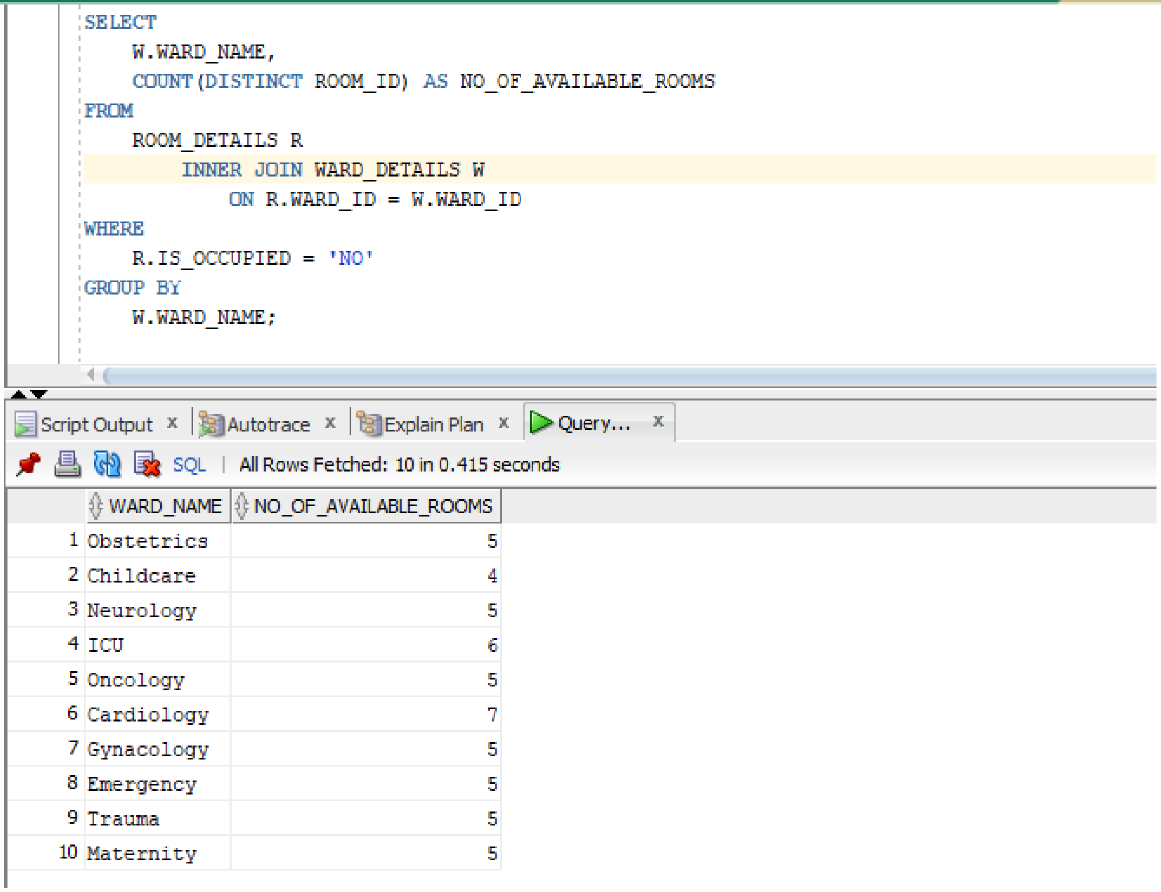
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
CREATE INDEX STATEMENT			5000	8
INDEX BUILD	IX2_FN_FIRSTNAME	NON UNIQUE		
SORT		CREATE INDEX	5000	
TABLE ACCESS	PATIENT_DETAILS	FULL	5000	5
Other XML				

VIII. Querying

Here, we will go over some interesting queries developed for this project.

1. **SQL Query to count the number of available rooms in each ward.**

```
SELECT
    W.WARD_NAME,
    COUNT(DISTINCT ROOM_ID) AS NO_OF_AVAILABLE_ROOMS
FROM
    ROOM_DETAILS R
        INNER JOIN WARD_DETAILS W
            ON R.WARD_ID = W.WARD_ID
WHERE
    R.IS_OCCUPIED = 'NO'
GROUP BY
    W.WARD_NAME;
```



```
SELECT
    W.WARD_NAME,
    COUNT(DISTINCT ROOM_ID) AS NO_OF_AVAILABLE_ROOMS
FROM
    ROOM_DETAILS R
        INNER JOIN WARD_DETAILS W
            ON R.WARD_ID = W.WARD_ID
WHERE
    R.IS_OCCUPIED = 'NO'
GROUP BY
    W.WARD_NAME;
```

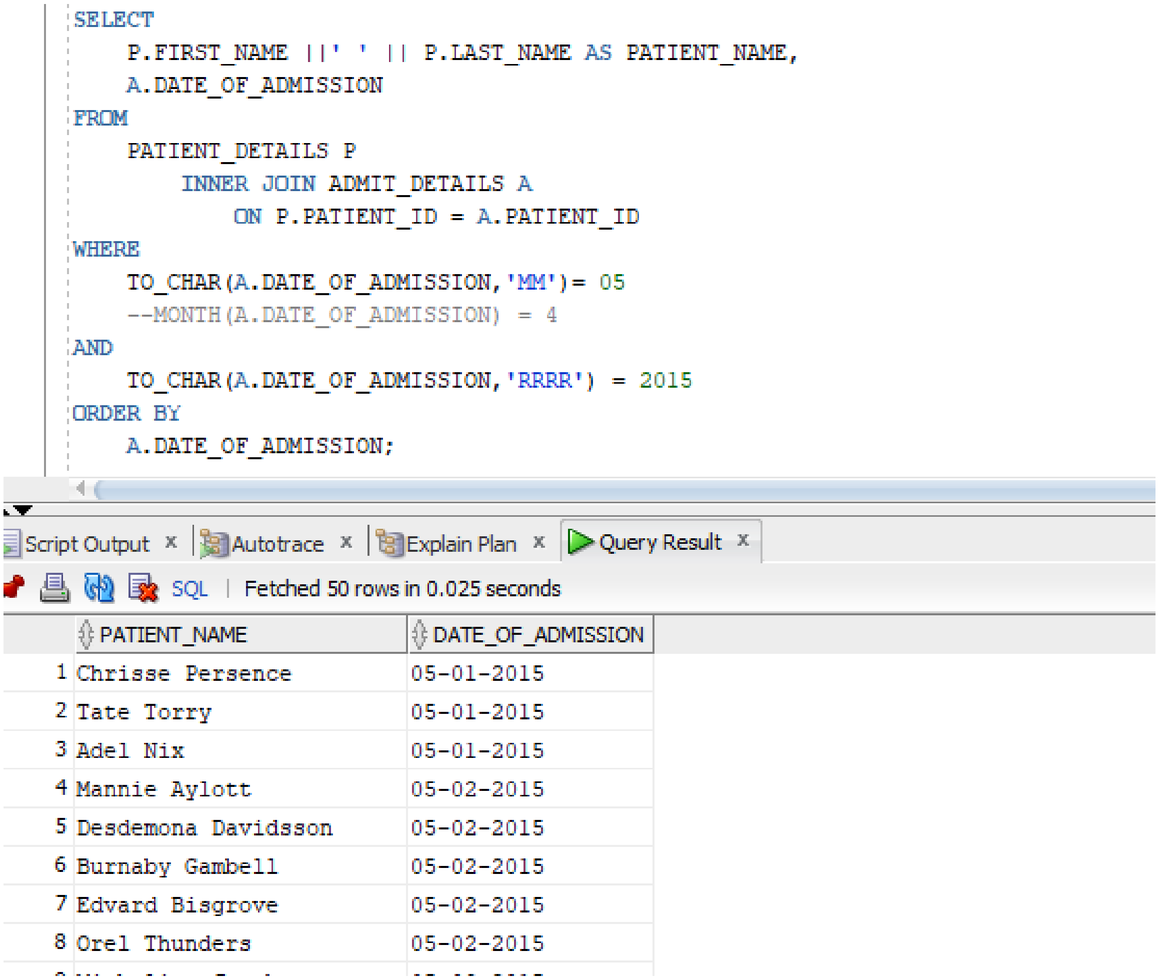
All Rows Fetched: 10 in 0.415 seconds

	WARD_NAME	NO_OF_AVAILABLE_ROOMS
1	Obstetrics	5
2	Childcare	4
3	Neurology	5
4	ICU	6
5	Oncology	5
6	Cardiology	7
7	Gynacology	5
8	Emergency	5
9	Trauma	5
10	Maternity	5

2. **SQL query to find the name of the patients who visited hospital on the month of April 2015.**

```
SELECT
   P.FIRST_NAME ||' ' || P.LAST_NAME AS PATIENT_NAME,
   A.DATE_OF_ADMISSION
FROM
   PATIENT_DETAILS P
      INNER JOIN ADMIT_DETAILS A
         ON P.PATIENT_ID = A.PATIENT_ID
WHERE
```

```
   TO_CHAR(A.DATE_OF_ADMISSION,'MM')= 05
   --MONTH(A.DATE_OF_ADMISSION) = 4
AND
   TO_CHAR(A.DATE_OF_ADMISSION,'RRRR') = 2015
ORDER BY
   A.DATE_OF_ADMISSION;
```



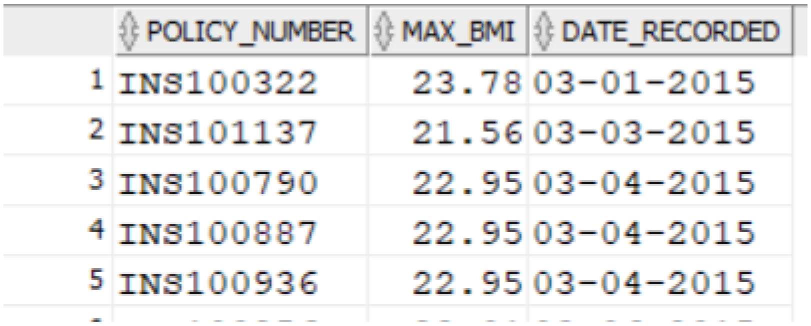
```
SELECT
    P.FIRST_NAME ||' ' || P.LAST_NAME AS PATIENT_NAME,
    A.DATE_OF_ADMISSION
FROM
    PATIENT_DETAILS P
        INNER JOIN ADMIT_DETAILS A
            ON P.PATIENT_ID = A.PATIENT_ID
WHERE
    TO_CHAR(A.DATE_OF_ADMISSION,'MM')= 05
    --MONTH(A.DATE_OF_ADMISSION) = 4
AND
    TO_CHAR(A.DATE_OF_ADMISSION,'RRRR') = 2015
ORDER BY
    A.DATE_OF_ADMISSION;
```

Fetched 50 rows in 0.025 seconds

	PATIENT_NAME	DATE_OF_ADMISSION
1	Chrisse Persence	05-01-2015
2	Tate Torry	05-01-2015
3	Adel Nix	05-01-2015
4	Mannie Aylott	05-02-2015
5	Desdemona Davidsson	05-02-2015
6	Burnaby Gambell	05-02-2015
7	Edvard Bisgrove	05-02-2015
8	Orel Thunders	05-02-2015

3. **Display policy number, maximum BMI and the date recorded for policy number which are having maximum BMI on that date.**

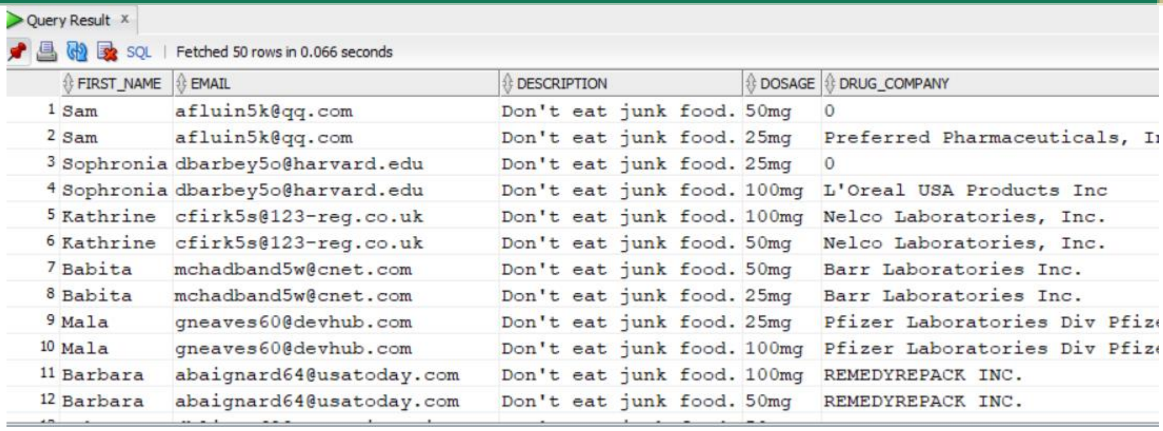
```
SELECT
    POLICY_NUMBER,
    MAX(BMI) OVER (PARTITION BY DATE_RECORDED) AS
    MAX_BMI,
    DATE_RECORDED
FROM
    PATIENT_HEALTH_DETAILS W
        INNER JOIN PATIENT_DETAILS X
        ON W.PATIENT_ID = X.PATIENT_ID
WHERE
BMI > 0;
```



	POLICY_NUMBER	MAX_BMI	DATE_RECORDED
1	INS100322	23.78	03-01-2015
2	INS101137	21.56	03-03-2015
3	INS100790	22.95	03-04-2015
4	INS100887	22.95	03-04-2015
5	INS100936	22.95	03-04-2015

4. **Display list of all patients with their description of prescription along with number of dosages suggested. The frequency of dosage should be only once a day.**

```
SELECT
    FIRST_NAME,
    EMAIL,
    DESCRIPTION,
    DOSAGE ,
    DRUG_COMPANY
FROM
    PATIENT_DETAILS P
        INNER JOIN PRESCRIPTION_DETAILS Q
            ON (P.PATIENT_ID=Q.PATIENT_ID)
                INNER JOIN
                    MEDICATION_DETAILS R
                    ON (Q.PATIENT_ID=R.PATIENT_ID)
WHERE
FREQUENCY='Once a day';
```

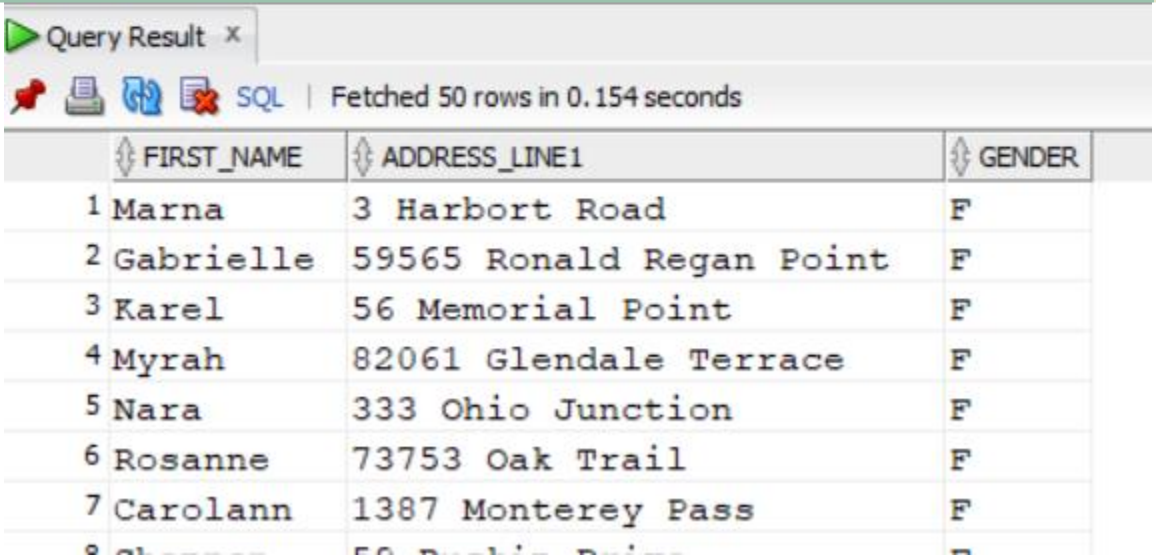


Fetched 50 rows in 0.066 seconds

	FIRST_NAME	EMAIL	DESCRIPTION	DOSAGE	DRUG_COMPANY
1	Sam	afluin5k@qq.com	Don't eat junk food.	50mg	0
2	Sam	afluin5k@qq.com	Don't eat junk food.	25mg	Preferred Pharmaceuticals, In
3	Sophronia	dbarbey5o@harvard.edu	Don't eat junk food.	25mg	0
4	Sophronia	dbarbey5o@harvard.edu	Don't eat junk food.	100mg	L'Oreal USA Products Inc
5	Kathrine	cfirk5s@123-reg.co.uk	Don't eat junk food.	100mg	Nelco Laboratories, Inc.
6	Kathrine	cfirk5s@123-reg.co.uk	Don't eat junk food.	50mg	Nelco Laboratories, Inc.
7	Babita	mchadband5w@cnet.com	Don't eat junk food.	50mg	Barr Laboratories Inc.
8	Babita	mchadband5w@cnet.com	Don't eat junk food.	25mg	Barr Laboratories Inc.
9	Mala	gneaves60@devhub.com	Don't eat junk food.	25mg	Pfizer Laboratories Div Pfize
10	Mala	gneaves60@devhub.com	Don't eat junk food.	100mg	Pfizer Laboratories Div Pfize
11	Barbara	abaignard64@usatoday.com	Don't eat junk food.	100mg	REMEDYREPACK INC.
12	Barbara	abaignard64@usatoday.com	Don't eat junk food.	50mg	REMEDYREPACK INC.

5. **Display the first name, first line of address of all female patients having insurance provider as ‘jkl insurance’**

```
SELECT
   FIRST_NAME,
   ADDRESS_LINE1,
   GENDER
   FROM
    PATIENT_DETAILS A
    INNER JOIN
    INSURANCE_DETAILS B
    ON (A.POLICY_NUMBER=B.POLICY_NUMBER)
    WHERE B.INSURANCE_PROVIDER='jkl insurance'
    AND A.GENDER='F'
```



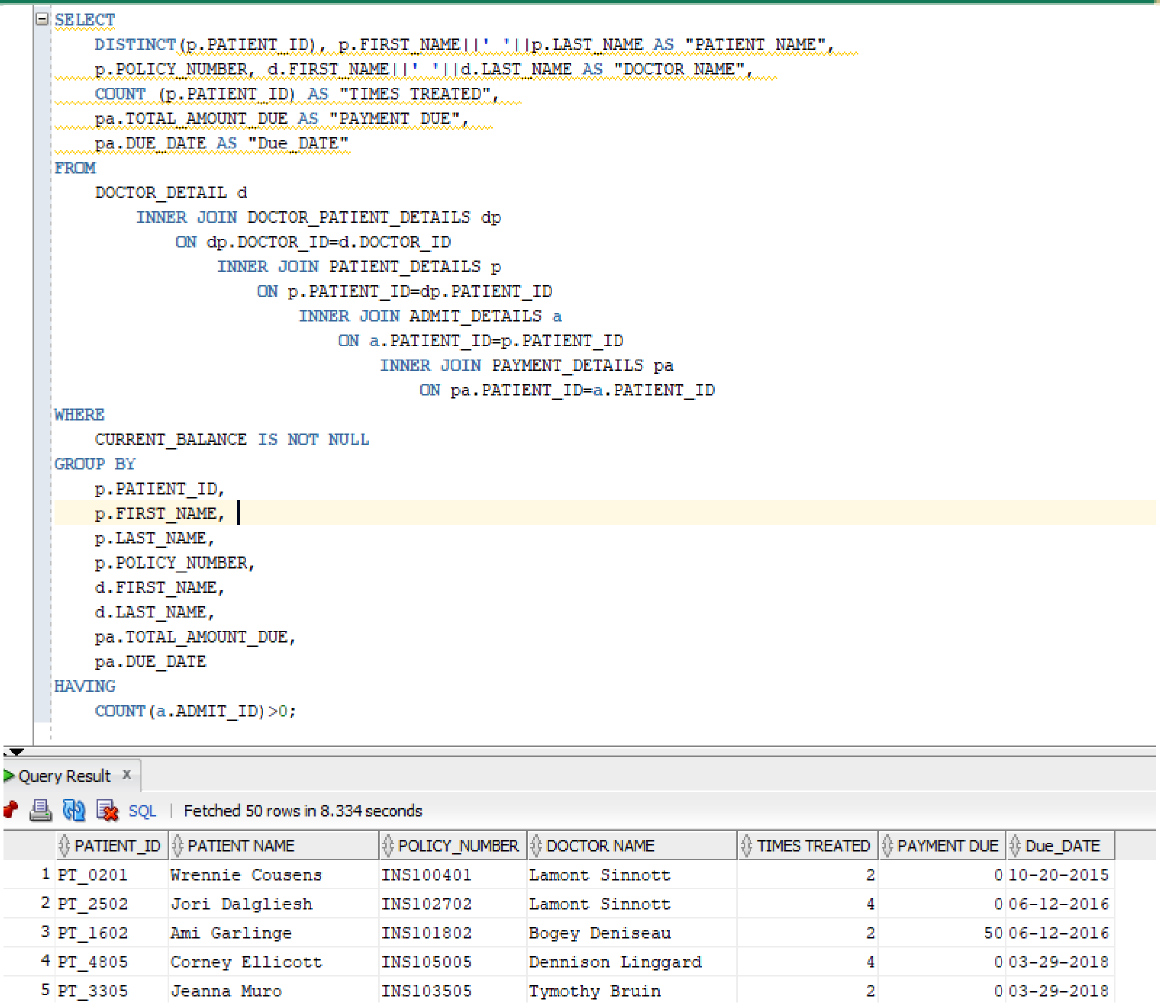
Fetched 50 rows in 0.154 seconds

	FIRST_NAME	ADDRESS_LINE1	GENDER
1	Marna	3 Harbort Road	F
2	Gabrielle	59565 Ronald Regan Point	F
3	Karel	56 Memorial Point	F
4	Myrah	82061 Glendale Terrace	F
5	Nara	333 Ohio Junction	F
6	Rosanne	73753 Oak Trail	F
7	Carolann	1387 Monterey Pass	F

6. **Find the details of all patients whose payment is due, the amount due, due date, the number of times they have been admitted along with their doctor names .**

```
SELECT
   DISTINCT(p.PATIENT_ID), p.FIRST_NAME||' '||p.LAST_NAME AS
"PATIENT NAME",
   p.POLICY_NUMBER, d.FIRST_NAME||' '||d.LAST_NAME AS "DOCTOR
NAME",
   COUNT (p.PATIENT_ID) AS "TIMES TREATED",
   pa.TOTAL_AMOUNT_DUE AS "PAYMENT DUE",
   pa.DUE_DATE AS "Due_DATE"
FROM
   DOCTOR_DETAIL d
      INNER JOIN DOCTOR_PATIENT_DETAILS dp
         ON dp.DOCTOR_ID=d.DOCTOR_ID
```

```
        INNER JOIN PATIENT_DETAILS p
          ON p.PATIENT_ID=dp.PATIENT_ID
            INNER JOIN ADMIT_DETAILS a
              ON a.PATIENT_ID=p.PATIENT_ID
                INNER JOIN PAYMENT_DETAILS pa
                  ON pa.PATIENT_ID=a.PATIENT_ID
WHERE
  CURRENT_BALANCE IS NOT NULL
GROUP BY
  p.PATIENT_ID,
  p.FIRST_NAME,
  p.LAST_NAME,
  p.POLICY_NUMBER,
  d.FIRST_NAME,
  d.LAST_NAME,
  pa.TOTAL_AMOUNT_DUE,
  pa.DUE_DATE
HAVING
  COUNT(a.ADMIT_ID)>0;
```



```
SELECT
    DISTINCT(p.PATIENT_ID), p.FIRST_NAME||' '||p.LAST_NAME AS "PATIENT NAME",
    p.POLICY_NUMBER, d.FIRST_NAME||' '||d.LAST_NAME AS "DOCTOR NAME",
    COUNT (p.PATIENT_ID) AS "TIMES TREATED",
    pa.TOTAL_AMOUNT_DUE AS "PAYMENT DUE",
    pa.DUE_DATE AS "Due_DATE"
FROM
    DOCTOR_DETAIL d
        INNER JOIN DOCTOR_PATIENT_DETAILS dp
            ON dp.DOCTOR_ID=d.DOCTOR_ID
                INNER JOIN PATIENT_DETAILS p
                    ON p.PATIENT_ID=dp.PATIENT_ID
                        INNER JOIN ADMIT_DETAILS a
                            ON a.PATIENT_ID=p.PATIENT_ID
                                INNER JOIN PAYMENT_DETAILS pa
                                    ON pa.PATIENT_ID=a.PATIENT_ID
WHERE
    CURRENT_BALANCE IS NOT NULL
GROUP BY
    p.PATIENT_ID,
    p.FIRST_NAME,
    p.LAST_NAME,
    p.POLICY_NUMBER,
    d.FIRST_NAME,
    d.LAST_NAME,
    pa.TOTAL_AMOUNT_DUE,
    pa.DUE_DATE
HAVING
    COUNT(a.ADMIT_ID)>0;
```

Fetched 50 rows in 8.334 seconds

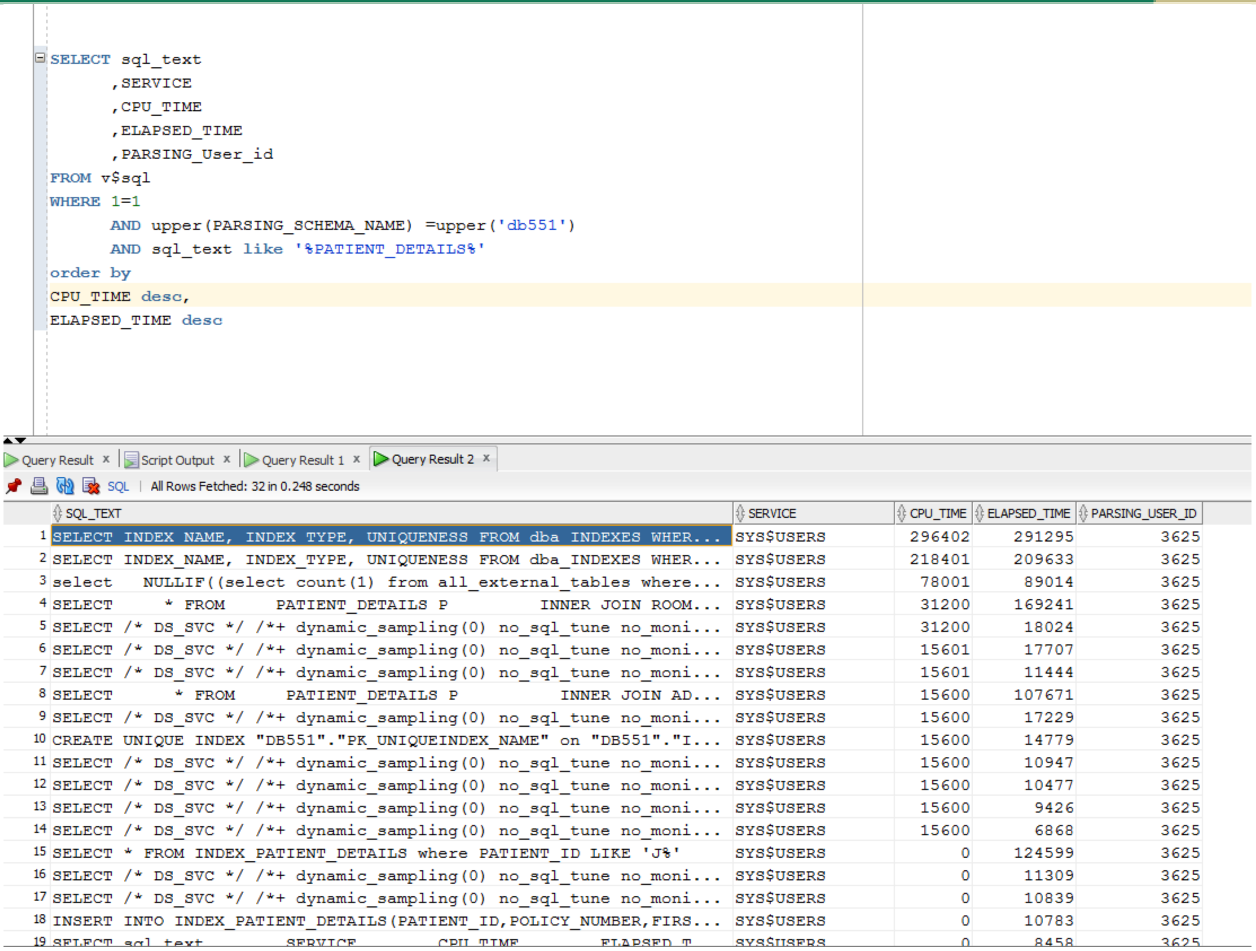
	PATIENT_ID	PATIENT NAME	POLICY_NUMBER	DOCTOR NAME	TIMES TREATED	PAYMENT DUE	Due_DATE
1	PT_0201	Wrennie Cousens	INS100401	Lamont Sinnott	2	0	10-20-2015
2	PT_2502	Jori Dalgliesh	INS102702	Lamont Sinnott	4	0	06-12-2016
3	PT_1602	Ami Garlinge	INS101802	Bogey Deniseau	2	50	06-12-2016
4	PT_4805	Corney Ellicott	INS105005	Dennison Linggard	4	0	03-29-2018
5	PT_3305	Jeanna Muro	INS103505	Tymothy Bruin	2	0	03-29-2018

IX. DBA Querying

This section deals with queries that can be used by DBA to perform database checks.

1. **Find all the SQL operations on a table order by CPU time and elapsed time.**

```
SELECT sql_text
    ,SERVICE
    ,CPU_TIME
    ,ELAPSED_TIME
    ,PARSING_User_id
FROM v$sql
WHERE 1=1
    AND upper(PARSING_SCHEMA_NAME) =upper('db551')
    AND sql_text like '%PATIENT_DETAILS%'
order by
CPU_TIME desc,
ELAPSED_TIME desc
```



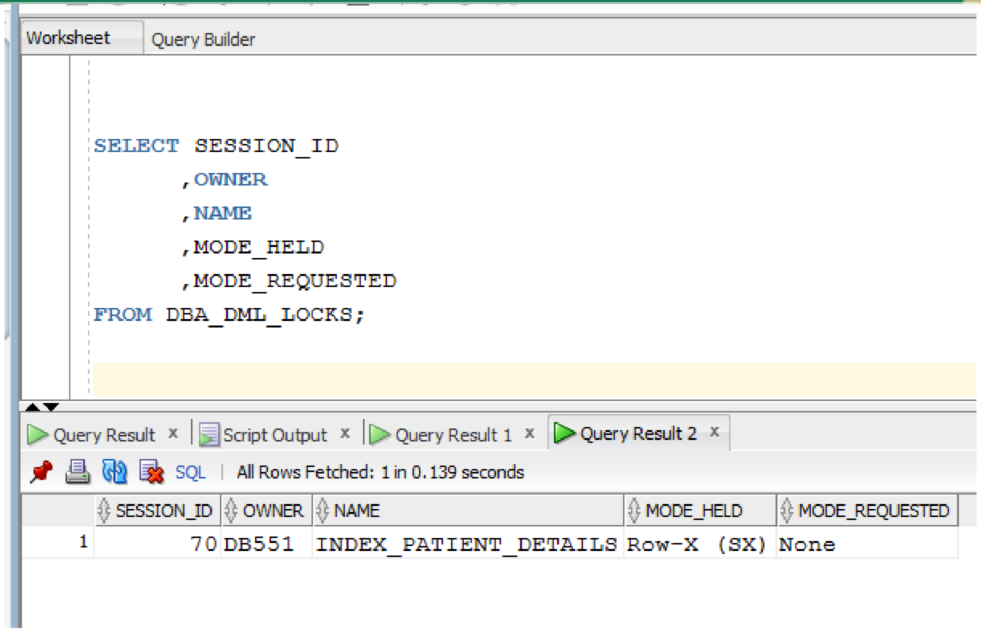
```
SELECT sql_text
       ,SERVICE
       ,CPU_TIME
       ,ELAPSED_TIME
       ,PARSING_User_id
FROM v$sql
WHERE 1=1
       AND upper(PARSING_SCHEMA_NAME) =upper('db551')
       AND sql_text like '%PATIENT_DETAILS%'
order by
CPU_TIME desc,
ELAPSED_TIME desc
```

All Rows Fetched: 32 in 0.248 seconds

```
    SQL_TEXT                                                       SERVICE      CPU_TIME  ELAPSED_TIME  PARSING_USER_ID
 1  SELECT INDEX NAME, INDEX TYPE, UNIQUENESS FROM dba INDEXES WHER... SYS$USERS    296402    291295     3625
 2  SELECT INDEX_NAME, INDEX_TYPE, UNIQUENESS FROM dba_INDEXES WHER... SYS$USERS    218401    209633     3625
 3  select   NULLIF((select count(1) from all_external_tables where... SYS$USERS     78001     89014     3625
 4  SELECT      * FROM      PATIENT_DETAILS P         INNER JOIN ROOM... SYS$USERS   31200    169241     3625
 5  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     31200     18024     3625
 6  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     15601     17707     3625
 7  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     15601     11444     3625
 8  SELECT       * FROM      PATIENT_DETAILS P          INNER JOIN AD... SYS$USERS   15600    107671     3625
 9  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     15600     17229     3625
10  CREATE UNIQUE INDEX "DB551"."PK_UNIQUEINDEX_NAME" on "DB551"."I... SYS$USERS     15600     14779     3625
11  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     15600     10947     3625
12  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     15600     10477     3625
13  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     15600      9426     3625
14  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS     15600      6868     3625
15  SELECT * FROM INDEX_PATIENT_DETAILS where PATIENT_ID LIKE 'J%'     SYS$USERS         0    124599     3625
16  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS         0     11309     3625
17  SELECT /* DS_SVC */ /*+ dynamic_sampling(0) no_sql_tune no_moni... SYS$USERS         0     10839     3625
18  INSERT INTO INDEX_PATIENT_DETAILS(PATIENT_ID,POLICY_NUMBER,FIRS... SYS$USERS         0     10783     3625
19  SELECT sql_text          SERVICE         CPU_TIME          ELAPSED_T    SYS$USERS         0      8458     3625
```

2. **See all the locks issued by DML statements.**

```
SELECT SESSION_ID
    ,OWNER
    ,NAME
    ,MODE_HELD
    ,MODE_REQUESTED
FROM DBA_DML_LOCKS;
```



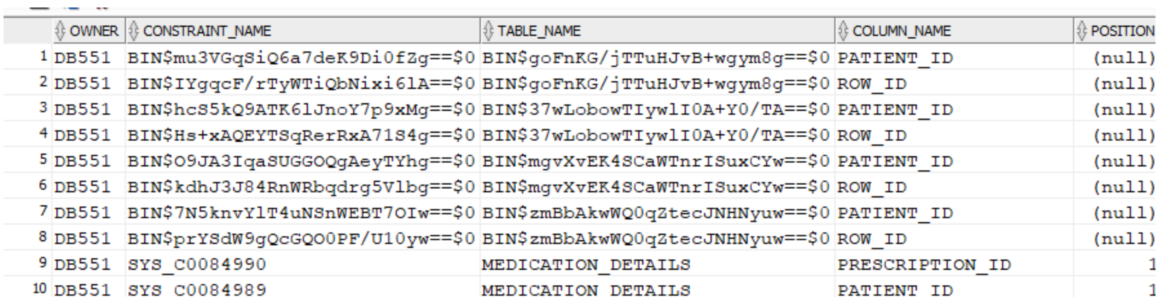
```
SELECT SESSION_ID
       ,OWNER
       ,NAME
       ,MODE_HELD
       ,MODE_REQUESTED
FROM DBA_DML_LOCKS;
```

All Rows Fetched: 1 in 0.139 seconds

	SESSION_ID	OWNER	NAME	MODE_HELD	MODE_REQUESTED
1	70	DB551	INDEX_PATIENT_DETAILS	Row-X (SX)	None

3. Find constraint in all the tables.

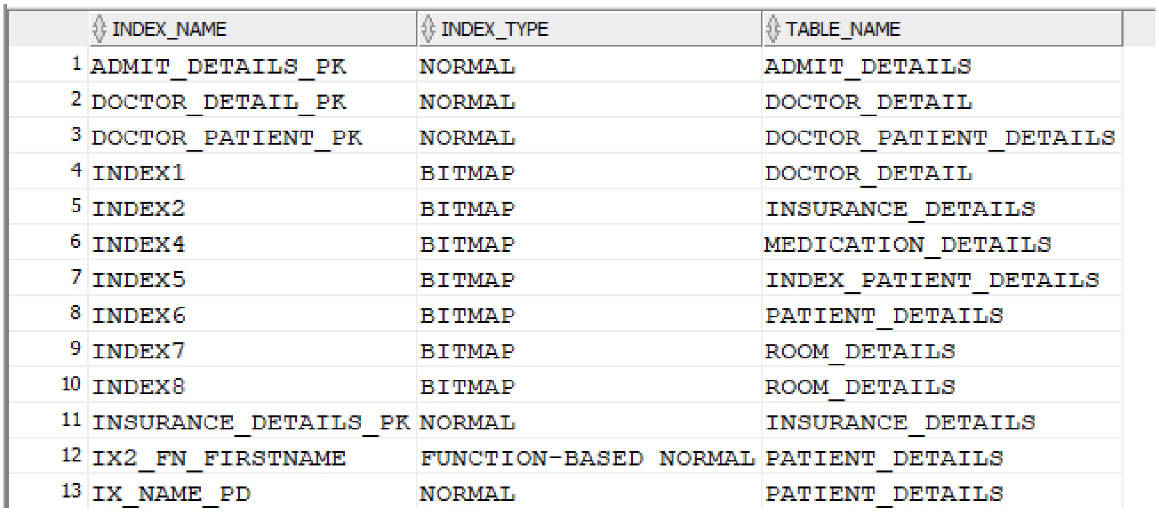
select *
from
dba_cons_columns
dba_constraints
where
owner =”DB551”



	OWNER	CONSTRAINT_NAME	TABLE_NAME	COLUMN_NAME	POSITION
1	DB551	BIN$mu3VGqSiQ6a7deK9Di0fZg==$0	BIN$goFnKG/jTTuHJvB+wgym8g==$0	PATIENT_ID	(null)
2	DB551	BIN$IYgqcF/rTyWTiQbNixi6lA==$0	BIN$goFnKG/jTTuHJvB+wgym8g==$0	ROW_ID	(null)
3	DB551	BIN$hcS5kQ9ATK6lJnoY7p9xMg==$0	BIN$37wLobowTIywlI0A+Y0/TA==$0	PATIENT_ID	(null)
4	DB551	BIN$Hs+xAQEYTSqRerRxA71S4g==$0	BIN$37wLobowTIywlI0A+Y0/TA==$0	ROW_ID	(null)
5	DB551	BIN$O9JA3IqaSUGGOQgAeyTYhg==$0	BIN$mgvXvEK4SCaWTnrISuxCYw==$0	PATIENT_ID	(null)
6	DB551	BIN$kdhJ3J84RnWRbqdrg5Vlbg==$0	BIN$mgvXvEK4SCaWTnrISuxCYw==$0	ROW_ID	(null)
7	DB551	BIN$7N5knvYlT4uNSnWEBT7OIw==$0	BIN$zmBbAkwWQ0qZtecJNHNyuw==$0	PATIENT_ID	(null)
8	DB551	BIN$prYSdW9gQcGQO0PF/U10yw==$0	BIN$zmBbAkwWQ0qZtecJNHNyuw==$0	ROW_ID	(null)
9	DB551	SYS_C0084990	MEDICATION_DETAILS	PRESCRIPTION_ID	1
10	DB551	SYS C0084989	MEDICATION DETAILS	PATIENT ID	1

4. List all the indexes of a database owner.

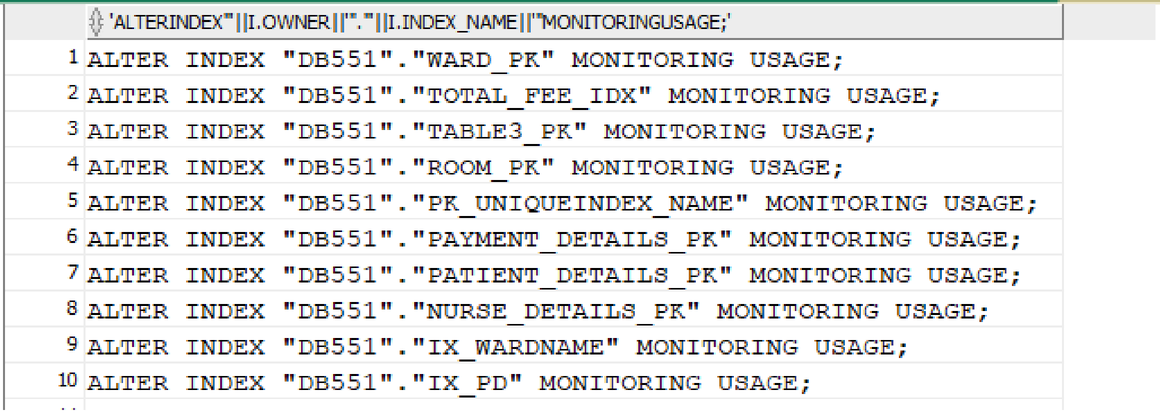
```
SELECT INDEX_NAME, INDEX_TYPE, TABLE_NAME FROM dba_indexes
WHERE owner = 'DB551' ORDER BY index_name;
```



	INDEX_NAME	INDEX_TYPE	TABLE_NAME
1	ADMIT_DETAILS_PK	NORMAL	ADMIT_DETAILS
2	DOCTOR_DETAIL_PK	NORMAL	DOCTOR_DETAIL
3	DOCTOR_PATIENT_PK	NORMAL	DOCTOR_PATIENT_DETAILS
4	INDEX1	BITMAP	DOCTOR_DETAIL
5	INDEX2	BITMAP	INSURANCE_DETAILS
6	INDEX4	BITMAP	MEDICATION_DETAILS
7	INDEX5	BITMAP	INDEX_PATIENT_DETAILS
8	INDEX6	BITMAP	PATIENT_DETAILS
9	INDEX7	BITMAP	ROOM_DETAILS
10	INDEX8	BITMAP	ROOM_DETAILS
11	INSURANCE_DETAILS_PK	NORMAL	INSURANCE_DETAILS
12	IX2_FN_FIRSTNAME	FUNCTION-BASED NORMAL	PATIENT_DETAILS
13	IX_NAME_PD	NORMAL	PATIENT_DETAILS

5. Sets monitoring on the specified table indexes.

```
SELECT 'ALTER INDEX "' || i.owner || '"."' || i.index_name || '" MONITORING
USAGE;'
FROM   dba_indexes i
WHERE owner = UPPER('&1');
```



	'ALTERINDEX"'||I.OWNER||'"."'||I.INDEX_NAME||'"MONITORINGUSAGE;'
1	ALTER INDEX "DB551"."WARD_PK" MONITORING USAGE;
2	ALTER INDEX "DB551"."TOTAL_FEE_IDX" MONITORING USAGE;
3	ALTER INDEX "DB551"."TABLE3_PK" MONITORING USAGE;
4	ALTER INDEX "DB551"."ROOM_PK" MONITORING USAGE;
5	ALTER INDEX "DB551"."PK_UNIQUEINDEX_NAME" MONITORING USAGE;
6	ALTER INDEX "DB551"."PAYMENT_DETAILS_PK" MONITORING USAGE;
7	ALTER INDEX "DB551"."PATIENT_DETAILS_PK" MONITORING USAGE;
8	ALTER INDEX "DB551"."NURSE_DETAILS_PK" MONITORING USAGE;
9	ALTER INDEX "DB551"."IX_WARDNAME" MONITORING USAGE;
10	ALTER INDEX "DB551"."IX_PD" MONITORING USAGE;

6. **List all active sessions.**

SELECT NVL(v$session.username, '(oracle)') AS username,

v$session.osuser, v$session.sid, v$session.serial#,v$process.spid,

v$session.lockwait, v$session.status,v$session.module,

v$session.machine,v$session.program,

TO_CHAR(v$session.logon_Time,'DD-MON-YYYY HH24:MI:SS') AS logon_time

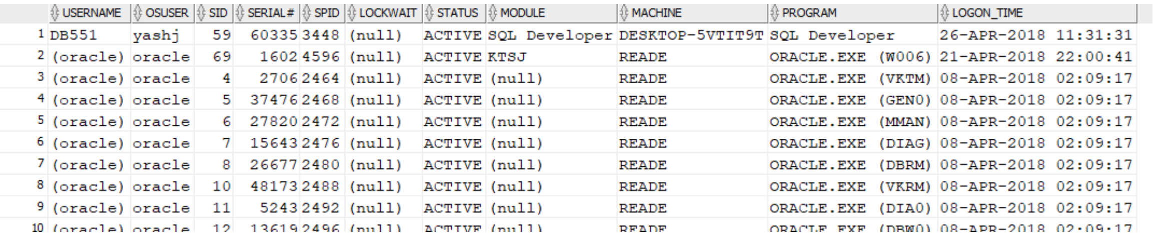
FROM v$session,

v$process

WHERE v$session.paddr = v$process.addr

AND v$session.status = 'ACTIVE'

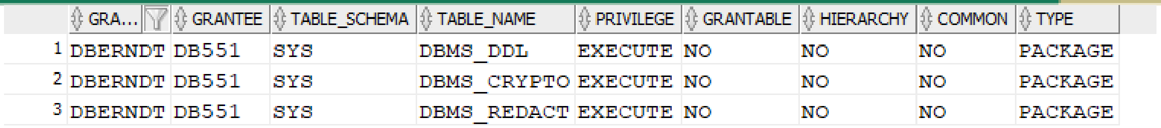
ORDER BY v$session.username, v$session.osuser;



	USERNAME	OSUSER	SID	SERIAL#	SPID	LOCKWAIT	STATUS	MODULE	MACHINE	PROGRAM	LOGON_TIME
1	DB551	yashj	59	60335	3448	(null)	ACTIVE	SQL Developer	DESKTOP-5VTIT9T	SQL Developer	26-APR-2018 11:31:31
2	(oracle)	oracle	69	1602	4596	(null)	ACTIVE	KTSJ	READE	ORACLE.EXE (W006)	21-APR-2018 22:00:41
3	(oracle)	oracle	4	2706	2464	(null)	ACTIVE	(null)	READE	ORACLE.EXE (VKTM)	08-APR-2018 02:09:17
4	(oracle)	oracle	5	37476	2468	(null)	ACTIVE	(null)	READE	ORACLE.EXE (GEN0)	08-APR-2018 02:09:17
5	(oracle)	oracle	6	27820	2472	(null)	ACTIVE	(null)	READE	ORACLE.EXE (MMAN)	08-APR-2018 02:09:17
6	(oracle)	oracle	7	15643	2476	(null)	ACTIVE	(null)	READE	ORACLE.EXE (DIAG)	08-APR-2018 02:09:17
7	(oracle)	oracle	8	26677	2480	(null)	ACTIVE	(null)	READE	ORACLE.EXE (DBRM)	08-APR-2018 02:09:17
8	(oracle)	oracle	10	48173	2488	(null)	ACTIVE	(null)	READE	ORACLE.EXE (VKRM)	08-APR-2018 02:09:17
9	(oracle)	oracle	11	5243	2492	(null)	ACTIVE	(null)	READE	ORACLE.EXE (DIA0)	08-APR-2018 02:09:17
10	(oracle)	oracle	12	13619	2496	(null)	ACTIVE	(null)	READE	ORACLE.EXE (DBW0)	08-APR-2018 02:09:17

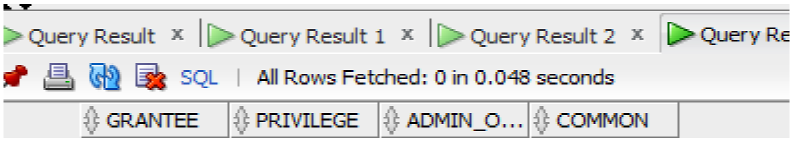
7. **Find privileges existing on the database.**

   a. select * from all_tab_privs where grantee = 'DB551';



	GRA...	GRANTEE	TABLE_SCHEMA	TABLE_NAME	PRIVILEGE	GRANTABLE	HIERARCHY	COMMON	TYPE
1	DBERNDT	DB551	SYS	DBMS_DDL	EXECUTE	NO	NO	NO	PACKAGE
2	DBERNDT	DB551	SYS	DBMS_CRYPTO	EXECUTE	NO	NO	NO	PACKAGE
3	DBERNDT	DB551	SYS	DBMS_REDACT	EXECUTE	NO	NO	NO	PACKAGE

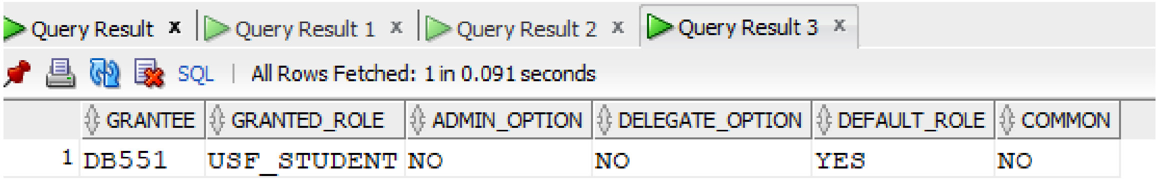
b. select * from dba_sys_privs where grantee = 'DB551';



All Rows Fetched: 0 in 0.048 seconds

GRANTEE	PRIVILEGE	ADMIN_O...	COMMON

c. select * from dba_role_privs where grantee = 'DB551';

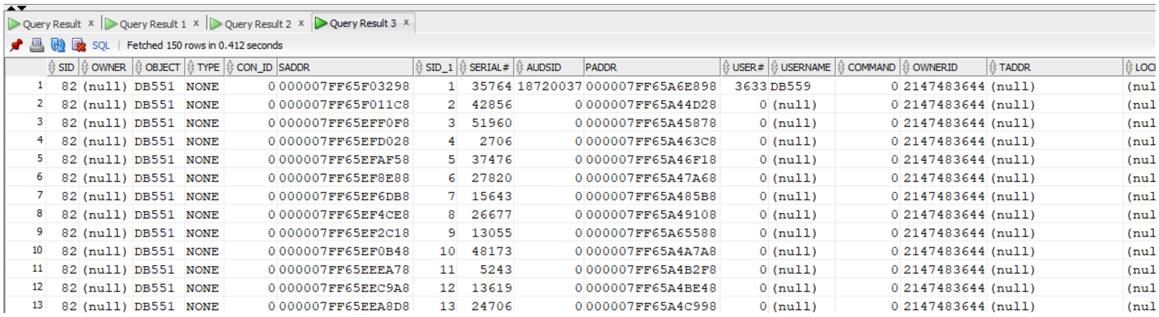


All Rows Fetched: 1 in 0.091 seconds

	GRANTEE	GRANTED_ROLE	ADMIN_OPTION	DELEGATE_OPTION	DEFAULT_ROLE	COMMON
1	DB551	USF_STUDENT	NO	NO	YES	NO

8. Lists all objects being accessed in the schema.

```
SELECT * FROM   v$access a, v$session b
 WHERE OBJECT='DB551';
```

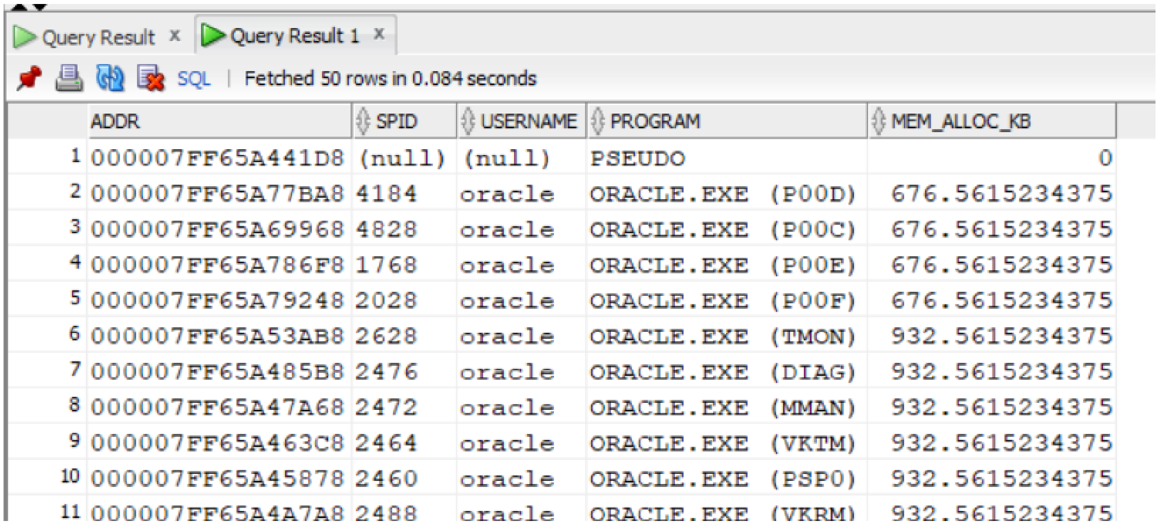


Fetched 150 rows in 0.412 seconds

	SID	OWNER	OBJECT	TYPE	CON_ID	SADDR	SID_1	SERIAL#	AUDSID	PADDR	USER#	USERNAME	COMMAND	OWNERID	TADDR	LOC
1	82	(null)	DB551	NONE	0	000007FF65F03298	1	35764	18720037	000007FF65A6E898	3633	DB559	0	2147483644	(null)	(nul
2	82	(null)	DB551	NONE	0	000007FF65F011C8	2	42856	0	000007FF65A44D28	0	(null)	0	2147483644	(null)	(nul
3	82	(null)	DB551	NONE	0	000007FF65EFF0F8	3	51960	0	000007FF65A45878	0	(null)	0	2147483644	(null)	(nul
4	82	(null)	DB551	NONE	0	000007FF65EFD028	4	2706	0	000007FF65A463C8	0	(null)	0	2147483644	(null)	(nul
5	82	(null)	DB551	NONE	0	000007FF65EFAF58	5	37476	0	000007FF65A46F18	0	(null)	0	2147483644	(null)	(nul
6	82	(null)	DB551	NONE	0	000007FF65EF8E88	6	27820	0	000007FF65A47A68	0	(null)	0	2147483644	(null)	(nul
7	82	(null)	DB551	NONE	0	000007FF65EF6DB8	7	15643	0	000007FF65A485B8	0	(null)	0	2147483644	(null)	(nul
8	82	(null)	DB551	NONE	0	000007FF65EF4CE8	8	26677	0	000007FF65A49108	0	(null)	0	2147483644	(null)	(nul
9	82	(null)	DB551	NONE	0	000007FF65EF2C18	9	13055	0	000007FF65A65588	0	(null)	0	2147483644	(null)	(nul
10	82	(null)	DB551	NONE	0	000007FF65EF0B48	10	48173	0	000007FF65A4A7A8	0	(null)	0	2147483644	(null)	(nul
11	82	(null)	DB551	NONE	0	000007FF65EEEA78	11	5243	0	000007FF65A4B2F8	0	(null)	0	2147483644	(null)	(nul
12	82	(null)	DB551	NONE	0	000007FF65EEC9A8	12	13619	0	000007FF65A4BE48	0	(null)	0	2147483644	(null)	(nul
13	82	(null)	DB551	NONE	0	000007FF65EEA8D8	13	24706	0	000007FF65A4C998	0	(null)	0	2147483644	(null)	(nul

9. Script to find all the sessions taking most PGA memory. (Can be useful in finding leaks.)

select addr,SPID,username,program,pga_alloc_mem/1024 mem_alloc_Kb from v$process order by pga_alloc_mem;



Fetched 50 rows in 0.084 seconds

	ADDR	SPID	USERNAME	PROGRAM	MEM_ALLOC_KB
1	000007FF65A441D8	(null)	(null)	PSEUDO	0
2	000007FF65A77BA8	4184	oracle	ORACLE.EXE (P00D)	676.5615234375
3	000007FF65A69968	4828	oracle	ORACLE.EXE (P00C)	676.5615234375
4	000007FF65A786F8	1768	oracle	ORACLE.EXE (P00E)	676.5615234375
5	000007FF65A79248	2028	oracle	ORACLE.EXE (P00F)	676.5615234375
6	000007FF65A53AB8	2628	oracle	ORACLE.EXE (TMON)	932.5615234375
7	000007FF65A485B8	2476	oracle	ORACLE.EXE (DIAG)	932.5615234375
8	000007FF65A47A68	2472	oracle	ORACLE.EXE (MMAN)	932.5615234375
9	000007FF65A463C8	2464	oracle	ORACLE.EXE (VKTM)	932.5615234375
10	000007FF65A45878	2460	oracle	ORACLE.EXE (PSP0)	932.5615234375
11	000007FF65A4A7A8	2488	oracle	ORACLE.EXE (VKRM)	932.5615234375

X. Database Security

Just like any valuable thing needs protection from unwarranted access, measures and protocols have been set up to protect the database from malicious attempts. An in-house IT team will be set up in place that reports to supervisors which in tur report t the hospital administrator. No outsourcing will be done, in order to avoid any incident of data getting leaked.

Before any data is migrated from the superficial server to main server proper paperwork needs to be completed where the supervisors assent is mandatory.

Every time a login takes place from a new device, DBAs get messages regarding the IP addresses and the MAC addresses. Every user, be it hospital staff or patient, needs to have a password that conforms with the security parameters. Additionally, every password will needs to be changes every 120 days lest the account gets suspended. In case of hospital staff, if the account has not been accessed the account gets suspended and can only be activated by the approval of supervisors.

ROLES	**PRIVILEGES**
Database Administrators	Administrator Privileges, maintenance, performance tuning, and recovery and create, update, delete data, users, groups and tables
Database Supervisor	Administrator Privileges, maintenance, performance tuning, and recovery and create, update, delete data, users, groups and tables Approve the work before DBAs run their jobs
Hospital Administrator	Full access to all the data

Doctors	Access to patient information apart from billing Access to nurses information like allocated ward and rooms. Privileges to create and update.
Nurses	Access to patient information apart from billing Limited acess to doctors information such as Shift timings. Privileges to create and update.
Patients	Access to his information including billing Limited access to doctors information such as Shift timings. Privileges to create and update.

All the data is being mirrored at a failover location, to prevent any loss of data in case of any catastrophe. Only Database supervisors will have access to them. Also, proper personnel are present at the data center location to avoid any unauthorized access. Every person who will enter the site will have and ID card so that records can be maintained. Regular checks will be done on the system.

XI. Conclusion

The “XYZ Hospital Database System” created above can be used by the institution as well by the patients to effectively manage their medical life. Hospital staff can review the patient records while the patient can track any and all progress. As everyone has unique login credential, steps have been taken to avoid any unauthorized alterations to the data.

We have provided a platform for reduce redundancy and increase efficiency. All updates are available to the patrons via email. This project can be extended to introduce appointment updates as well.

The users are also restricted to access only authorized documents to prevent any information breach. Any jump in data traffic will be handled smoothly by the system.

The information stored in the Data Warehouses can be analyzed using big data tools such as Rapid Miner and Apache Hadoop to better understand what further can be done in the healthcare industry.