Hospital Management System

DBMS Project

Prepared by :
Srishti Sharma (102003780)
Davjot Singh (102003143)
Arshvir Singh Sandhu (102003151)
Ashmeet Singh Sandhu (102183005)
Aavish Sidana (102183058)

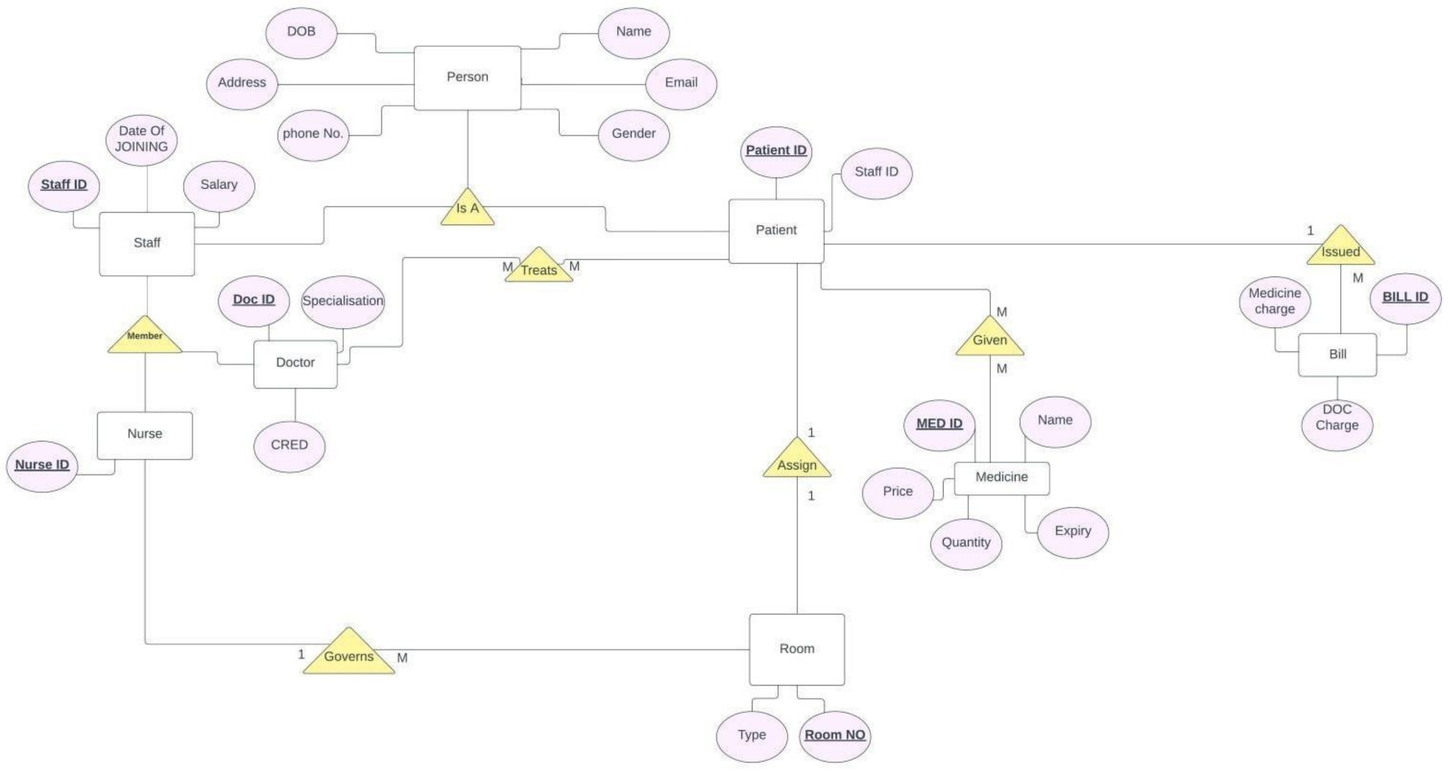
Submitted to: Pragya Mishra

Table of Contents

1. Problem Statement

A system to manage the activities in a hospital.

2. ER Diagram



3. ER to Table

	PERSON	
PK	Person_ID	NUMBER
	PERSON_FULL_NAME	VARCHAR
	PERSON_DOB	DATE
	PERSON_EMAIL	VARCHAR
	PERSON_PHONE	NUMBER
	PERSON_GENDER	VARCHAR
	PERSON_BLOOD_GROUP	VARCHAR
	PERSON_ROLE	VARCHAR

	PATIENT	
PK	PT_ID	NUMBER
FK	PERSON_ID	NUMBER

	DOCTOR	
FK	ST_ID	NUMBER
PK	DOC_ID	NUMBER
	DOC_Speciality	VARCHAR

	NURSE	
PK	N_ID	NUMBER
FK	ST_ID	NUMBER
	N_SHIFT	VARCHAR

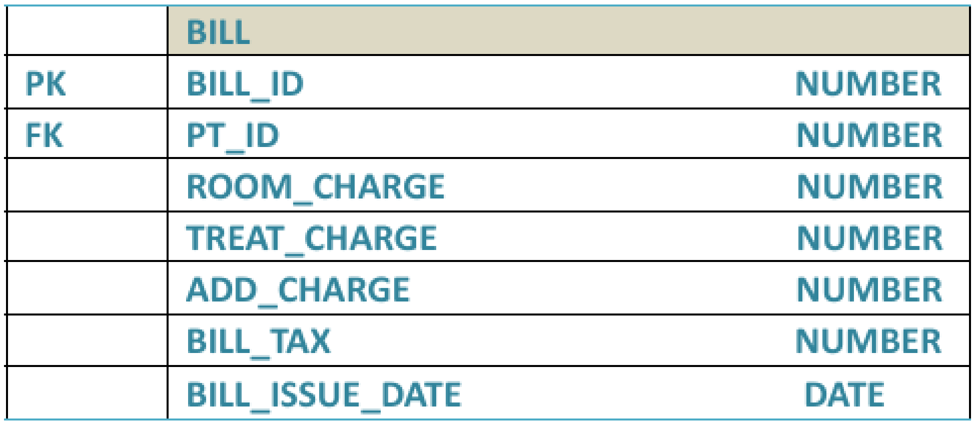
	MEDICINE	
PK	MED_ID	NUMBER
	MED_NAME	VARCHAR

	PATIENT_GIVEN_MEDICINE	
FK,PK	MED_ID	NUMBER
FK,PK	PT_ID	NUMBER
	MED_QTY	NUMBER

	DOC_TREAT_PATIENT	
FK,PK	DOC_ID	NUMBER
FK,PK	PT_ID	NUMBER
	DISEASE	VARCHAR
	DOC_COMMENTS	VARCHAR

	ROOM	
PK	ROOM_ID	NUMBER
	ROOM_TYPE	VARCHAR
	ROOM_PRICE	NUMBER
	ROOM_AVAILABLE	NUMBER

	ALLOT_ROOM	
FK,PK	PT_ID	NUMBER
FK	N_ID	NUMBER
	ROOM_CHECKIN	DATE
	ROOM_CHECKOUT	DATE



	BILL	
PK	BILL_ID	NUMBER
FK	PT_ID	NUMBER
	ROOM_CHARGE	NUMBER
	TREAT_CHARGE	NUMBER
	ADD_CHARGE	NUMBER
	BILL_TAX	NUMBER
	BILL_ISSUE_DATE	DATE

4. Code + Outputs

--Creating Tables using SQL commands

```
 CREATE TABLE PERSON(
   PERSON_ID NUMBER,
   PERSON_FULL_NAME VARCHAR2(20) NOT NULL,
   PERSON_DOB DATE,
   PERSON_EMAIL VARCHAR2(30),
   PERSON_PHONE NUMBER(10) NOT NULL,
   PERSON_GENDER VARCHAR2(10),
   PERSON_BLOOD_GROUP VARCHAR2(5),
   PERSON_ROLE VARCHAR2(5) CHECK(PERSON_ROLE IN ('PT', 'DOC', 'N')),

   CONSTRAINT PERSON_PK PRIMARY KEY(PERSON_ID),
   CONSTRAINT PERSON_BLOODGROUP_CH CHECK(PERSON_BLOOD_GROUP =
ANY('A+','A-','B-','B+','AB+','AB-','O-','O+','AEL')),
   CONSTRAINT PERSON_GENDER_CH CHECK(PERSON_GENDER =
ANY('MALE','FEMALE','OTHERS'))
);

CREATE TABLE
   PATIENT( PERSON_ID
   NUMBER,

   CONSTRAINT PT_PK PRIMARY KEY(PERSON_ID),
   CONSTRAINT PT_FK FOREIGN KEY(PERSON_ID) REFERENCES PERSON(PERSON_ID)
);

CREATE TABLE
   STAFF( PERSON_ID NUMBER
   UNIQUE,
   ST_ID NUMBER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1),
   ST_SALARY NUMBER(10, 3),
   ST_JOIN_DATE DATE NOT NULL,
   ST_YRS_EXPERIENCE NUMBER NOT NULL,
   ST_QUALIFICATION VARCHAR2(20) NOT NULL,

   CONSTRAINT ST_PK PRIMARY KEY(ST_ID),
   CONSTRAINT ST_FK FOREIGN KEY(PERSON_ID) REFERENCES PERSON(PERSON_ID)
);

CREATE TABLE DOCTOR(
   ST_ID NUMBER,
   DOC_ID NUMBER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1),
   DOC_SPECIALITY VARCHAR2(20),

   CONSTRAINT DOC_PK PRIMARY KEY(DOC_ID),
   CONSTRAINT DOC_FK FOREIGN KEY(ST_ID) REFERENCES STAFF(ST_ID)
);

CREATE TABLE NURSE(
   N_ID NUMBER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1),
   ST_ID NUMBER,
   N_SHIFT VARCHAR2(5) CHECK(N_SHIFT IN ('DAY', 'NIGHT')),

   CONSTRAINT N_PK PRIMARY KEY(N_ID),
   CONSTRAINT N_FK FOREIGN KEY(ST_ID) REFERENCES STAFF(ST_ID)
);

CREATE TABLE MEDICINE(
   MED_ID NUMBER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1),
   MED_NAME VARCHAR2(40) NOT NULL,

   CONSTRAINT MED_PK PRIMARY KEY(MED_ID)
);

CREATE TABLE
   PATIENT_GIVEN_MEDICINE( MED_ID
   NUMBER,
   PERSON_ID NUMBER,
   MED_QTY NUMBER NOT NULL,

   CONSTRAINT PGM_MED_FK FOREIGN KEY(MED_ID) REFERENCES MEDICINE(MED_ID),
   CONSTRAINT PGM_PT_FK FOREIGN KEY(PERSON_ID) REFERENCES
PATIENT(PERSON_ID),
   CONSTRAINT PGM_PK PRIMARY KEY(PERSON_ID, MED_ID)
);

CREATE TABLE
   DOC_TREAT_PATIENT( DOC_ID
   NUMBER,
   PERSON_ID NUMBER,
   DISEASE VARCHAR2(30),
   DOC_COMMENTS VARCHAR2(40),

   CONSTRAINT DTP_DOC_FK FOREIGN KEY(DOC_ID) REFERENCES DOCTOR(DOC_ID),
   CONSTRAINT DTP_PT_FK FOREIGN KEY(PERSON_ID) REFERENCES
PATIENT(PERSON_ID),
   CONSTRAINT DTP_PK PRIMARY KEY(DOC_ID, PERSON_ID)
);

CREATE TABLE
   ROOM( ROOM_ID NUMBER,
   ROOM_TYPE VARCHAR2(20),
   ROOM_PRICE NUMBER(7, 3),
   ROOM_AVAILABLE NUMBER(1),

   CONSTRAINT ROOM_PK PRIMARY KEY(ROOM_ID),
   CONSTRAINT ROOM_TYPE_CH CHECK(ROOM_TYPE IN ('SUITE', 'SINGLE', 'DELUXE'))
);

CREATE TABLE
   ALLOT_ROOM( PERSON_ID
   NUMBER,
   N_ID NUMBER,
   ROOM_ID NUMBER,
   ROOM_CHECKIN DATE DEFAULT SYSDATE,
   ROOM_CHECKOUT DATE,
   CONSTRAINT AR_PT_FK FOREIGN KEY(PERSON_ID) REFERENCES
PATIENT(PERSON_ID),
   CONSTRAINT AR_N_FK FOREIGN KEY(N_ID) REFERENCES NURSE(N_ID),
   CONSTRAINT AR_PK PRIMARY KEY(PERSON_ID),
   CONSTRAINT ROOM_ID_FK FOREIGN KEY(ROOM_ID) REFERENCES ROOM(ROOM_ID)
);

CREATE TABLE BILL(
   BILL_ID NUMBER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1),
   PERSON_ID NUMBER,
   --DEPT_CHARGE NUMBER(5, 3),
   ROOM_CHARGE NUMBER(7, 3),
   TREAT_CHARGE NUMBER(10, 3) DEFAULT 100,
   ADD_CHARGE NUMBER(5, 3),
   BILL_TAX NUMBER(10, 3),
   BILL_ISSUE_DATE DATE DEFAULT SYSDATE,

   CONSTRAINT BILL_PK PRIMARY KEY(BILL_ID),
   CONSTRAINT BILL_FK FOREIGN KEY(PERSON_ID) REFERENCES PATIENT(PERSON_ID)
);
/
```

```
CREATE PROCEDURE PT_CHECKIN(
    ID IN PERSON.PERSON_ID%TYPE,
    FULL_NAME IN PERSON.PERSON_FULL_NAME%TYPE,
    DOB IN PERSON.PERSON_DOB%TYPE,
    EMAIL IN PERSON.PERSON_EMAIL%TYPE,
    PHONE IN PERSON.PERSON_PHONE%TYPE,
    GENDER IN PERSON.PERSON_GENDER%TYPE,
    BLOOD_GRP IN PERSON.PERSON_BLOOD_GROUP%TYPE
) IS
    PHONE_INVALID EXCEPTION;
    DOB_INVALID EXCEPTION;
    CHECK_CONSTRAINT_VIOLATED EXCEPTION;
    PRAGMA EXCEPTION_INIT(CHECK_CONSTRAINT_VIOLATED, -2290);
BEGIN
    IF SUBSTR(TO_CHAR(PHONE), 1, 1) NOT IN ('9', '8', '7', '6') THEN
        RAISE PHONE_INVALID;
    ELSIF TRIM(TO_CHAR(DOB, 'YYYY')) < 1900 THEN
        RAISE DOB_INVALID;
    ELSE
        INSERT INTO
        PERSON(
            PERSON_ID,
            PERSON_FULL_NAME,
            PERSON_DOB,
            PERSON_EMAIL,
            PERSON_PHONE,
            PERSON_GENDER,
            PERSON_BLOOD_GROUP,
            PERSON_ROLE
            )
        VALUES(
            ID,
            FULL_NAME,
            DOB,
            EMAIL,
            PHONE,
            GENDER,
            BLOOD_GRP,
            'PT'
        );

        INSERT INTO
        PATIENT(
            PERSON_ID
        )
        VALUES(
            ID
        );
    END IF;

EXCEPTION
    WHEN PHONE_INVALID THEN
        DBMS_OUTPUT.PUT_LINE('PHONE NUMBER INVALID');
    WHEN DOB_INVALID THEN
        DBMS_OUTPUT.PUT_LINE('DOB INVALID');
    WHEN CHECK_CONSTRAINT_VIOLATED THEN
        DBMS_OUTPUT.PUT_LINE('*ERROR INSERTING*, CHECK YOUR INPUTS IN ROLE COL,
BLOOD GROUP, GENDER');
END;
/
CREATE OR REPLACE PROCEDURE INSERT_DOC(
    PID IN PERSON.PERSON_ID%TYPE,
    FULL_NAME IN PERSON.PERSON_FULL_NAME%TYPE,
    DOB IN PERSON.PERSON_DOB%TYPE,
    EMAIL IN PERSON.PERSON_EMAIL%TYPE,
    PHONE IN PERSON.PERSON_PHONE%TYPE,
    GENDER IN PERSON.PERSON_GENDER%TYPE,
    BLOOD_GRP IN PERSON.PERSON_BLOOD_GROUP%TYPE,
    SAL IN STAFF.ST_SALARY%TYPE,
    DATE_OF_JOIN IN STAFF.ST_JOIN_DATE%TYPE,
    EXPER IN STAFF.ST_YRS_EXPERIENCE%TYPE,
    QUALIF IN STAFF.ST_QUALIFICATION%TYPE,
    SPEC IN DOCTOR.DOC_SPECIALITY%TYPE
) IS
    CHECK_CONSTRAINT_VIOLATED EXCEPTION;
    PRAGMA EXCEPTION_INIT(CHECK_CONSTRAINT_VIOLATED, -2290);
    PHONE_INVALID EXCEPTION;
    DOB_INVALID EXCEPTION;

    S_ID STAFF.ST_ID%TYPE;
BEGIN
    IF SUBSTR(TO_CHAR(PHONE), 1, 1) NOT IN ('9', '8', '7', '6') THEN
        RAISE PHONE_INVALID;
    ELSIF TRIM(TO_CHAR(DOB, 'YYYY')) < 1900 THEN
        RAISE DOB_INVALID;
    ELSE
        INSERT INTO
            PERSON( PERSON_ID,
            PERSON_FULL_NAME,
            PERSON_DOB,
            PERSON_EMAIL,
            PERSON_PHONE,
            PERSON_GENDER,
            PERSON_BLOOD_GROUP,
            PERSON_ROLE
        )
        VALUES(
            PID,
            FULL_NAME,
            DOB,
            EMAIL,
            PHONE,
            GENDER,
            BLOOD_GRP,
            'DOC'
        );
        INSERT INTO
            STAFF( PERSON_ID,
            ST_SALARY,
            ST_JOIN_DATE,
            ST_YRS_EXPERIENCE,
            ST_QUALIFICATION
        )
        VALUES(
            PID,
            SAL,
            DATE_OF_JOIN,
            EXPER,
            QUALIF
        );
        SELECT ST_ID INTO S_ID FROM STAFF WHERE STAFF.PERSON_ID = PID;

        INSERT INTO DOCTOR(
            ST_ID,
            DOC_SPECIALITY
        )
        VALUES(
            S_ID,
            SPEC
        );
    END IF;
EXCEPTION
    WHEN PHONE_INVALID THEN
        DBMS_OUTPUT.PUT_LINE('PHONE NUMBER INVALID');
    WHEN DOB_INVALID THEN
        DBMS_OUTPUT.PUT_LINE('DOB INVALID');
    WHEN CHECK_CONSTRAINT_VIOLATED THEN
        DBMS_OUTPUT.PUT_LINE('*ERROR INSERTING*, CHECK YOUR INPUTS!');
END;
/
CREATE OR REPLACE PROCEDURE
    INSERT_NURSE( PID IN PERSON.PERSON_ID%TYPE,
    FULL_NAME IN PERSON.PERSON_FULL_NAME%TYPE,
    DOB IN PERSON.PERSON_DOB%TYPE,
    EMAIL IN PERSON.PERSON_EMAIL%TYPE,
    PHONE IN PERSON.PERSON_PHONE%TYPE,
    GENDER IN PERSON.PERSON_GENDER%TYPE,
    BLOOD_GRP IN PERSON.PERSON_BLOOD_GROUP%TYPE,
    SAL IN STAFF.ST_SALARY%TYPE,
    DATE_OF_JOIN IN STAFF.ST_JOIN_DATE%TYPE,
    EXPER IN STAFF.ST_YRS_EXPERIENCE%TYPE,
    QUALIF IN STAFF.ST_QUALIFICATION%TYPE,
    SHIFT IN NURSE.N_SHIFT%TYPE
) IS
    CHECK_CONSTRAINT_VIOLATED EXCEPTION;
    PRAGMA EXCEPTION_INIT(CHECK_CONSTRAINT_VIOLATED, -2290);
    PHONE_INVALID EXCEPTION;
    DOB_INVALID EXCEPTION;
    S_ID STAFF.ST_ID%TYPE;
BEGIN
    IF SUBSTR(TO_CHAR(PHONE), 1, 1) NOT IN ('9', '8', '7', '6') THEN
        RAISE PHONE_INVALID;
    ELSIF TRIM(TO_CHAR(DOB, 'YYYY')) < 1900 THEN
        RAISE DOB_INVALID;
    ELSE
        INSERT INTO
            PERSON( PERSON_ID,
            PERSON_FULL_NAME,
            PERSON_DOB,
            PERSON_EMAIL,
            PERSON_PHONE,
            PERSON_GENDER,
            PERSON_BLOOD_GROUP,
            PERSON_ROLE
        )
        VALUES(
            PID,
            FULL_NAME,
            DOB,
            EMAIL,
            PHONE,
            GENDER,
            BLOOD_GRP,
            'DOC'
        );

        INSERT INTO
            STAFF( PERSON_ID,
            ST_SALARY,
            ST_JOIN_DATE,
            ST_YRS_EXPERIENCE,
            ST_QUALIFICATION
        )
        VALUES(
            PID,
            SAL,
            DATE_OF_JOIN,
            EXPER,
            QUALIF
        );
        SELECT ST_ID INTO S_ID FROM STAFF WHERE STAFF.PERSON_ID = PID;

        INSERT INTO NURSE(
            ST_ID,
            N_SHIFT
        )
        VALUES(
            S_ID,
            SHIFT
        );
    END IF;
```

```
EXCEPTION
    WHEN PHONE_INVALID THEN
        DBMS_OUTPUT.PUT_LINE('PHONE NUMBER INVALID');
    WHEN DOB_INVALID THEN
        DBMS_OUTPUT.PUT_LINE('DOB INVALID');
    WHEN CHECK_CONSTRAINT_VIOLATED THEN
        DBMS_OUTPUT.PUT_LINE('*ERROR INSERTING*, CHECK YOUR INPUTS!');
END;
/
CREATE OR REPLACE PROCEDURE INSERT_MEDICINE(
    MNAME IN MEDICINE.MED_NAME%TYPE
) IS
    TOO_LARGE_EXCEP_VIOL EXCEPTION;
    PRAGMA EXCEPTION_INIT(TOO_LARGE_EXCEP_VIOL, -12899);
    NOT_NULL_VIOL EXCEPTION;
    PRAGMA EXCEPTION_INIT(NOT_NULL_VIOL, -01400);
BEGIN
    INSERT INTO MEDICINE(
        MED_NAME
    )
    VALUES(
        MNAME
    );
EXCEPTION
    WHEN TOO_LARGE_EXCEP_VIOL THEN
        DBMS_OUTPUT.PUT_LINE('ENTERED NAME IS TOO LARGE.');
    WHEN NOT_NULL_VIOL THEN
        DBMS_OUTPUT.PUT_LINE('NAME CANNOT BE NULL');
END;
/
CREATE OR REPLACE PROCEDURE
    DOC_TREATS_PAT( PID IN NUMBER,
    DID IN NUMBER,
    MID IN NUMBER,
    MQTY IN NUMBER,
    DIS IN VARCHAR2,
    DOC_COMM IN VARCHAR2
) IS
    PCOUNT  INT;
    MCOUNT INT;
    DCOUNT INT;
BEGIN
    SELECT COUNT(*) INTO PCOUNT FROM PATIENT WHERE PERSON_ID = PID;
    SELECT COUNT(*) INTO MCOUNT FROM MEDICINE WHERE MED_ID = MID;
    SELECT COUNT(*) INTO DCOUNT FROM DOCTOR WHERE DOC_ID = DID;

    IF PCOUNT <= 0 THEN
        DBMS_OUTPUT.PUT_LINE('NO PATIENT EXISTS');
    ELSIF DCOUNT <= 0 THEN
        DBMS_OUTPUT.PUT_LINE('NO DOCTOR EXISTS');
    ELSIF MCOUNT <= 0 THEN
        DBMS_OUTPUT.PUT_LINE('NO MEDICINE EXISTS');
    ELSE
        INSERT INTO PATIENT_GIVEN_MEDICINE(
            MED_ID,
            PERSON_ID,
            MED_QTY
        )
        VALUES(
            MID,
            PID,
            MQTY
        );
        INSERT INTO
            DOC_TREAT_PATIENT( DOC_ID,
            PERSON_ID,
            DISEASE,
            DOC_COMMENTS
        )
        VALUES(
            DID,
            PID,
            DIS,
            DOC_COMM
        );
    END IF;
END;
/
CREATE OR REPLACE PROCEDURE ROOM_ALLOC(
    PID IN NUMBER,
    N_ID IN NUMBER,
    -- R_ID IN NUMBER,
    INDATE IN DATE,
    OUTDATE IN DATE
) IS
    PCOUNT INT;
    MCOUNT INT;
    DCOUNT INT;
    AVAIL INT;
    RID INT;
BEGIN
    SELECT COUNT(*) INTO PCOUNT FROM PATIENT WHERE PERSON_ID = PID;
    SELECT COUNT(*) INTO AVAIL FROM ROOM WHERE ROOM_AVAILABLE = 1;

    IF PCOUNT <= 0 THEN
        DBMS_OUTPUT.PUT_LINE('NO PATIENT EXISTS');
    ELSIF AVAIL <= 0 THEN
        DBMS_OUTPUT.PUT_LINE('NO ROOM IS AVAILABLE');
    ELSE
        SELECT ROOM_ID INTO RID FROM ROOM WHERE ROOM_AVAILABLE = 1 FETCH
FIRST 1 ROW ONLY;
        INSERT INTO ALLOT_ROOM(
            PERSON_ID,
            N_ID,
            ROOM_ID,
          ROOM_CHECKIN,
          ROOM_CHECKOUT
        )
        VALUES(
          PID,
          N_ID,
          RID,
          INDATE,
          OUTDATE
        );
        UPDATE ROOM SET ROOM_AVAILABLE=0 WHERE ROOM_ID=RID;
    END IF;
END;
/
CREATE OR REPLACE PROCEDURE DISP_PT_ROOM(PID IN NUMBER)
IS
ROOM_NO NUMBER;
PERSID NUMBER;
BEGIN
SELECT PERSON_ID INTO PERSID FROM PATIENT WHERE PERSON_ID=PID;
SELECT ALLOT_ROOM.ROOM_ID INTO ROOM_NO FROM ROOM INNER JOIN ALLOT_ROOM
ON ROOM.ROOM_ID=ALLOT_ROOM.ROOM_ID WHERE ALLOT_ROOM.PERSON_ID = PID;
DBMS_OUTPUT.PUT_LINE('PATIENT IN ROOM '|| ROOM_NO);
EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('PATIENT DOES NOT EXIST!!');
END;
/
CREATE OR REPLACE PROCEDURE DISCHARGE_PT(PID IN NUMBER)
IS
TEMP NUMBER;
BEGIN
SELECT ROOM_ID INTO TEMP FROM ALLOT_ROOM WHERE PERSON_ID = PID;
UPDATE ALLOT_ROOM SET ROOM_CHECKOUT = SYSDATE WHERE PERSON_ID=PID;
UPDATE ROOM SET ROOM_AVAILABLE = 1 WHERE ROOM_ID=(SELECT ROOM_ID FROM
ALLOT_ROOM WHERE PERSON_ID = PID);
DBMS_OUTPUT.PUT_LINE('PATIENT DISCHARGED !');
EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('PATIENT DOES NOT EXIST!!');
END;
/
```

```
-- ALL DETAILS OF PATIENT
CREATE OR REPLACE VIEW DISP_PATIENT AS
SELECT * FROM PERSON P1 NATURAL JOIN PATIENT WHERE P1.PERSON_ROLE = 'PT';
/
-- ALL DETAILS OF DOCTOR
CREATE OR REPLACE VIEW DISP_DOCTOR AS
SELECT * FROM PERSON P1 NATURAL JOIN STAFF WHERE P1.PERSON_ROLE = 'DOC';
--- TRIGGERS
/
CREATE OR REPLACE TRIGGER CREATE_BILL
AFTER UPDATE OF ROOM_AVAILABLE ON ROOM
FOR EACH ROW
BEGIN
IF :NEW.ROOM_AVAILABLE=1 THEN
   DBMS_OUTPUT.PUT_LINE('BILL ISSUE DATE '||SYSDATE);
   DBMS_OUTPUT.PUT_LINE('PATIENT TREATMENT CHARGE IS - 200');
   DBMS_OUTPUT.PUT_LINE('TAX % = 5%');
END IF;
END;
/
CREATE TABLE LOGS(
    USER_ID NUMBER GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY
1),
    USER_NAME VARCHAR2(50),
    OPN VARCHAR2(50),
    PERSON_ID NUMBER,

    CONSTRAINT LOGS_FK FOREIGN KEY(PERSON_ID) REFERENCES
PERSON(PERSON_ID)
);
/
CREATE OR REPLACE TRIGGER AI_LOGS
AFTER INSERT ON PERSON
FOR EACH ROW
DECLARE
    USERNAME VARCHAR2(50);
BEGIN
    SELECT USER INTO USERNAME FROM DUAL;
    INSERT INTO LOGS(
        USER_NAME,
        OPN,
        PERSON_ID
    )
    VALUES(
        USERNAME,
        'INSERT',
        :NEW.PERSON_ID
    );
END;
/
```

---   CURSORS

```
CREATE OR REPLACE PROCEDURE CURS_PT_DOC(
    DID IN NUMBER
) IS
    CURSOR DISP_DOC_DETAILS IS
    SELECT PERSON_FULL_NAME FROM PERSON WHERE PERSON_ID = (SELECT
PERSON_ID FROM STAFF WHERE ST_ID = (SELECT ST_ID FROM DOCTOR WHERE
DOC_ID = DID));

    CURSOR DISP_PT_DETAILS IS
    SELECT PERSON_FULL_NAME FROM PERSON WHERE PERSON_ID = (SELECT
PERSON_ID FROM DOC_TREAT_PATIENT WHERE DOC_ID = DID);

    DOC_NAME VARCHAR2(20);
    PT_NAME VARCHAR2(20);
BEGIN
    OPEN DISP_DOC_DETAILS;
    LOOP
        FETCH DISP_DOC_DETAILS INTO DOC_NAME;
        EXIT WHEN DISP_DOC_DETAILS%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('DOCTOR NAME: '||DOC_NAME);
    END LOOP;
    CLOSE DISP_DOC_DETAILS;

    OPEN DISP_PT_DETAILS;
    LOOP
        FETCH DISP_PT_DETAILS INTO PT_NAME;
        EXIT WHEN DISP_PT_DETAILS%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('PATIENT NAME: '||PT_NAME);
    END LOOP;
    CLOSE DISP_PT_DETAILS;
END;
/
```

-----   INSERTIONS

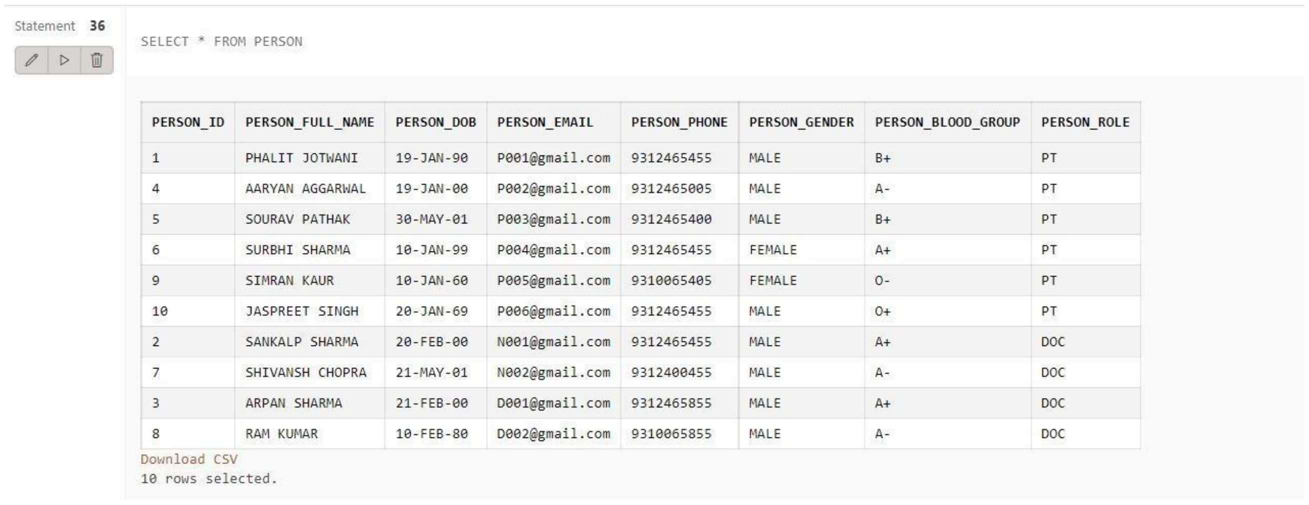
```
EXEC PT_CHECKIN(1,'PHALIT JOTWANI',TO_DATE('19-JAN-1990','DD-MM-
YYYY'),'P001@gmail.com',9312465455,'MALE','B+');
EXEC PT_CHECKIN(4,'AARYAN AGGARWAL',TO_DATE('19-JAN-2000','DD-MM-
YYYY'),'P002@gmail.com',9312465005,'MALE','A-');
EXEC PT_CHECKIN(5,'SOURAV PATHAK',TO_DATE('30-MAY-2001','DD-MM-
YYYY'),'P003@gmail.com',9312465400,'MALE','B+');
EXEC PT_CHECKIN(6,'SURBHI SHARMA',TO_DATE('10-JAN-1999','DD-MM-
YYYY'),'P004@gmail.com',9312465455,'FEMALE','A+');
EXEC PT_CHECKIN(9,'SIMRAN KAUR',TO_DATE('10-JAN-1960','DD-MM-
YYYY'),'P005@gmail.com',9310065405,'FEMALE','O-');
EXEC PT_CHECKIN(10,'JASPREET SINGH',TO_DATE('20-JAN-1969','DD-MM-
YYYY'),'P006@gmail.com',9312465455,'MALE','O+');
```

```
EXEC INSERT_NURSE(2,'SANKALP SHARMA',TO_DATE('20-FEB-2000','DD-MM-
YYYY'),'N001@gmail.com',9312465455,'MALE','A+',2000,TO_DATE('20-FEB-2020','DD-MM-
YYYY'),2,'ABC EFGHI','DAY');
EXEC INSERT_NURSE(7,'SHIVANSH CHOPRA',TO_DATE('21-MAY-2001','DD-MM-
YYYY'),'N002@gmail.com',9312400455,'MALE','A-',3000,TO_DATE('20-MAY-2020','DD-MM-
YYYY'),1,'AB EFGHIGH','DAY');
EXEC INSERT_DOC(3,'ARPAN SHARMA',TO_DATE('21-FEB-2000','DD-MM-
YYYY'),'D001@gmail.com',9312465855,'MALE','A+',10000,TO_DATE('20-FEB-2019','DD-MM-
YYYY'),3,'MBBS ','CARDIO');
EXEC INSERT_DOC(8,'RAM KUMAR',TO_DATE('10-FEB-1980','DD-MM-
YYYY'),'D002@gmail.com',9310065855,'MALE','A-',100000,TO_DATE('10-FEB-2000','DD-MM-
YYYY'),3,'MBBS','ENDOCRINOLOGY');

SELECT * FROM PERSON;
SELECT * FROM STAFF;
SELECT * FROM NURSE;
SELECT * FROM DOCTOR;


---- ROOM ALLOCATION
INSERT INTO ROOM VALUES(1,'SINGLE',500,1);
INSERT INTO ROOM VALUES(2,'DELUXE',1500,1);
INSERT INTO ROOM VALUES(3,'SUITE',2500,1);
EXEC ROOM_ALLOC(1,1,SYSDATE,SYSDATE);
EXEC ROOM_ALLOC(4,1,SYSDATE,SYSDATE);
SELECT * FROM ALLOT_ROOM;
EXEC DISP_PT_ROOM(1);
```

OUTPUT



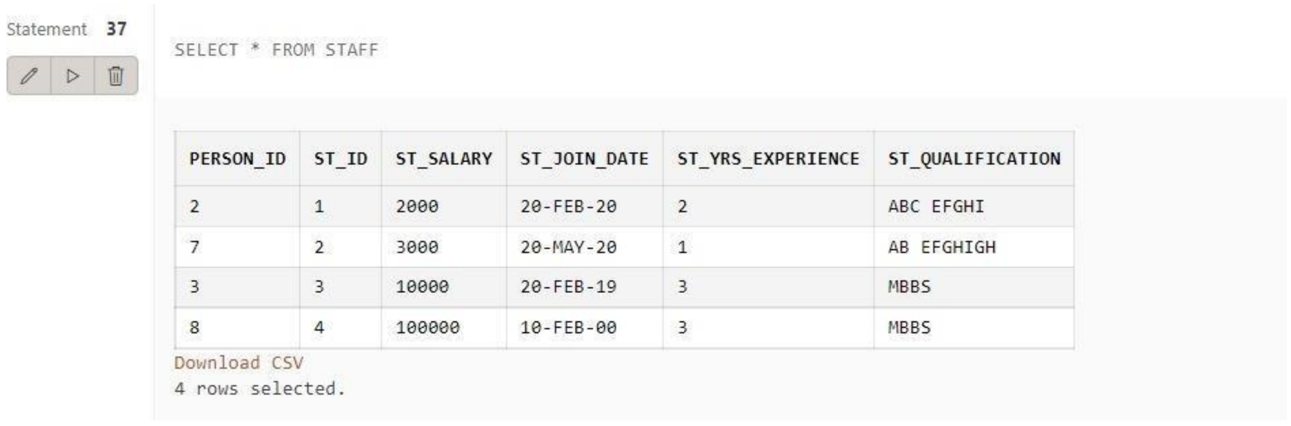
Statement 36

```
SELECT * FROM PERSON
```

PERSON_ID	PERSON_FULL_NAME	PERSON_DOB	PERSON_EMAIL	PERSON_PHONE	PERSON_GENDER	PERSON_BLOOD_GROUP	PERSON_ROLE
1	PHALIT JOTWANI	19-JAN-90	P001@gmail.com	9312465455	MALE	B+	PT
4	AARYAN AGGARWAL	19-JAN-00	P002@gmail.com	9312465005	MALE	A-	PT
5	SOURAV PATHAK	30-MAY-01	P003@gmail.com	9312465400	MALE	B+	PT
6	SURBHI SHARMA	10-JAN-99	P004@gmail.com	9312465455	FEMALE	A+	PT
9	SIMRAN KAUR	10-JAN-60	P005@gmail.com	9310065405	FEMALE	O-	PT
10	JASPREET SINGH	20-JAN-69	P006@gmail.com	9312465455	MALE	O+	PT
2	SANKALP SHARMA	20-FEB-00	N001@gmail.com	9312465455	MALE	A+	DOC
7	SHIVANSH CHOPRA	21-MAY-01	N002@gmail.com	9312400455	MALE	A-	DOC
3	ARPAN SHARMA	21-FEB-00	D001@gmail.com	9312465855	MALE	A+	DOC
8	RAM KUMAR	10-FEB-80	D002@gmail.com	9310065855	MALE	A-	DOC

Download CSV

10 rows selected.



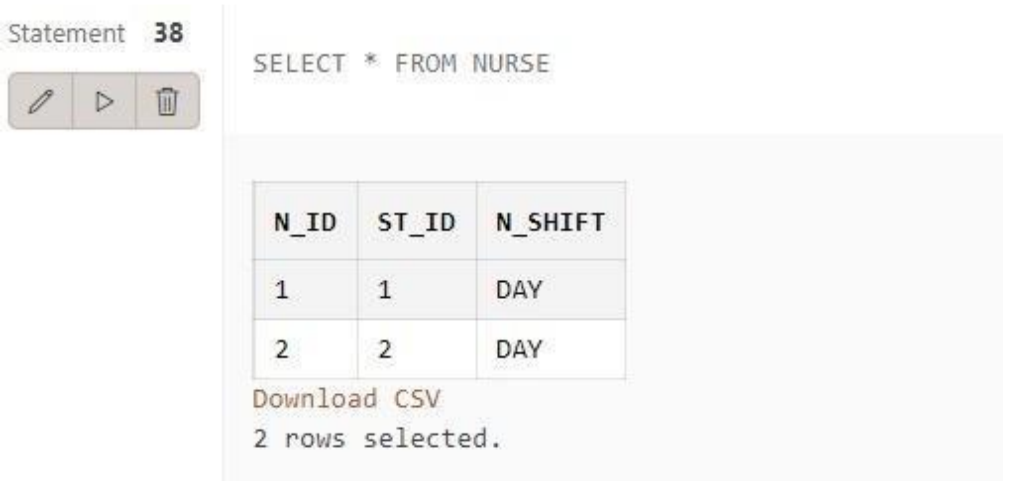
Statement 37

```
SELECT * FROM STAFF
```

PERSON_ID	ST_ID	ST_SALARY	ST_JOIN_DATE	ST_YRS_EXPERIENCE	ST_QUALIFICATION
2	1	2000	20-FEB-20	2	ABC EFGHI
7	2	3000	20-MAY-20	1	AB EFGHIGH
3	3	10000	20-FEB-19	3	MBBS
8	4	100000	10-FEB-00	3	MBBS

Download CSV

4 rows selected.



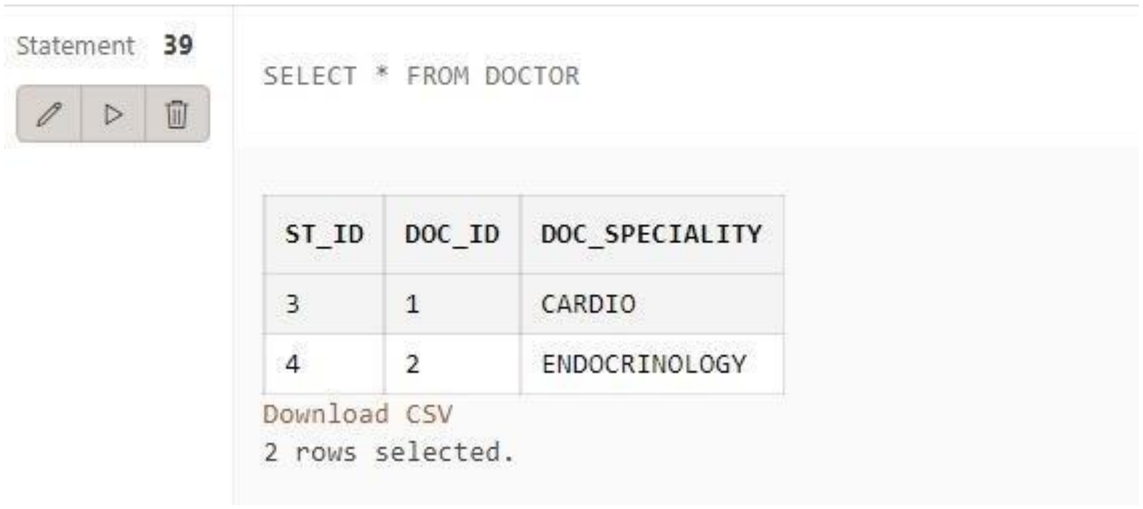
Statement 38

```
SELECT * FROM NURSE
```

N_ID	ST_ID	N_SHIFT
1	1	DAY
2	2	DAY

Download CSV

2 rows selected.

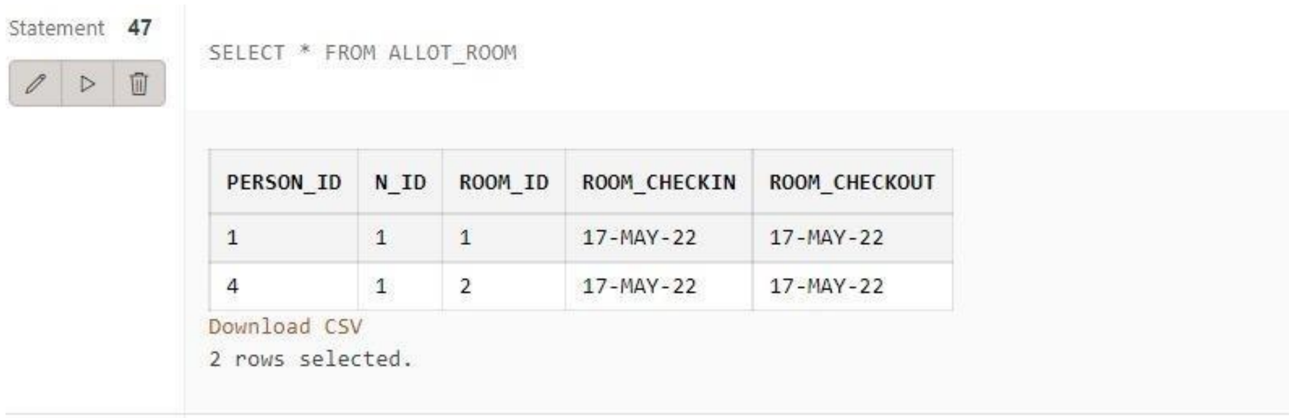


Statement 39

```
SELECT * FROM DOCTOR
```

ST_ID	DOC_ID	DOC_SPECIALITY
3	1	CARDIO
4	2	ENDOCRINOLOGY

```
Download CSV
2 rows selected.
```

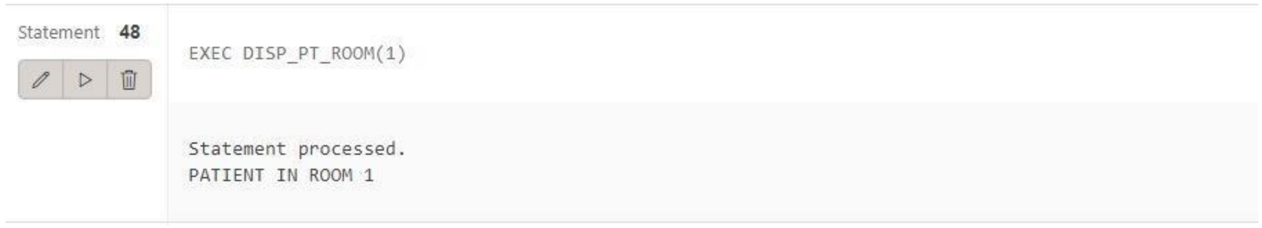


Statement 47

```
SELECT * FROM ALLOT_ROOM
```

PERSON_ID	N_ID	ROOM_ID	ROOM_CHECKIN	ROOM_CHECKOUT
1	1	1	17-MAY-22	17-MAY-22
4	1	2	17-MAY-22	17-MAY-22

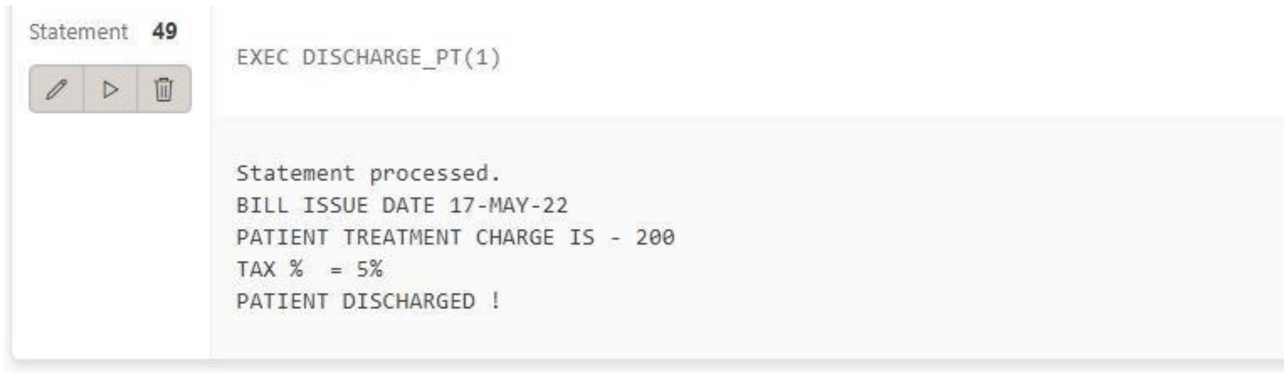
```
Download CSV
2 rows selected.
```



Statement 48

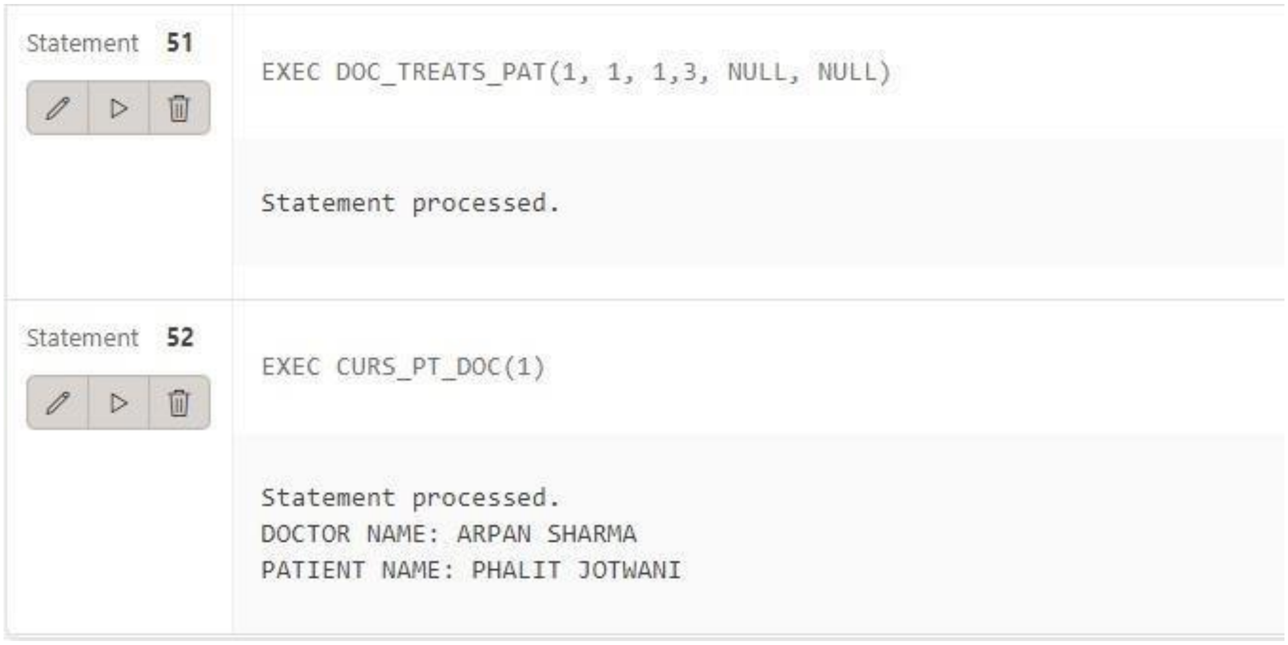
```
EXEC DISP_PT_ROOM(1)
```

```
Statement processed.
PATIENT IN ROOM 1
```



```
EXEC DISCHARGE_PT(1)
```

```
Statement processed.
BILL ISSUE DATE 17-MAY-22
PATIENT TREATMENT CHARGE IS - 200
TAX %  = 5%
PATIENT DISCHARGED !
```

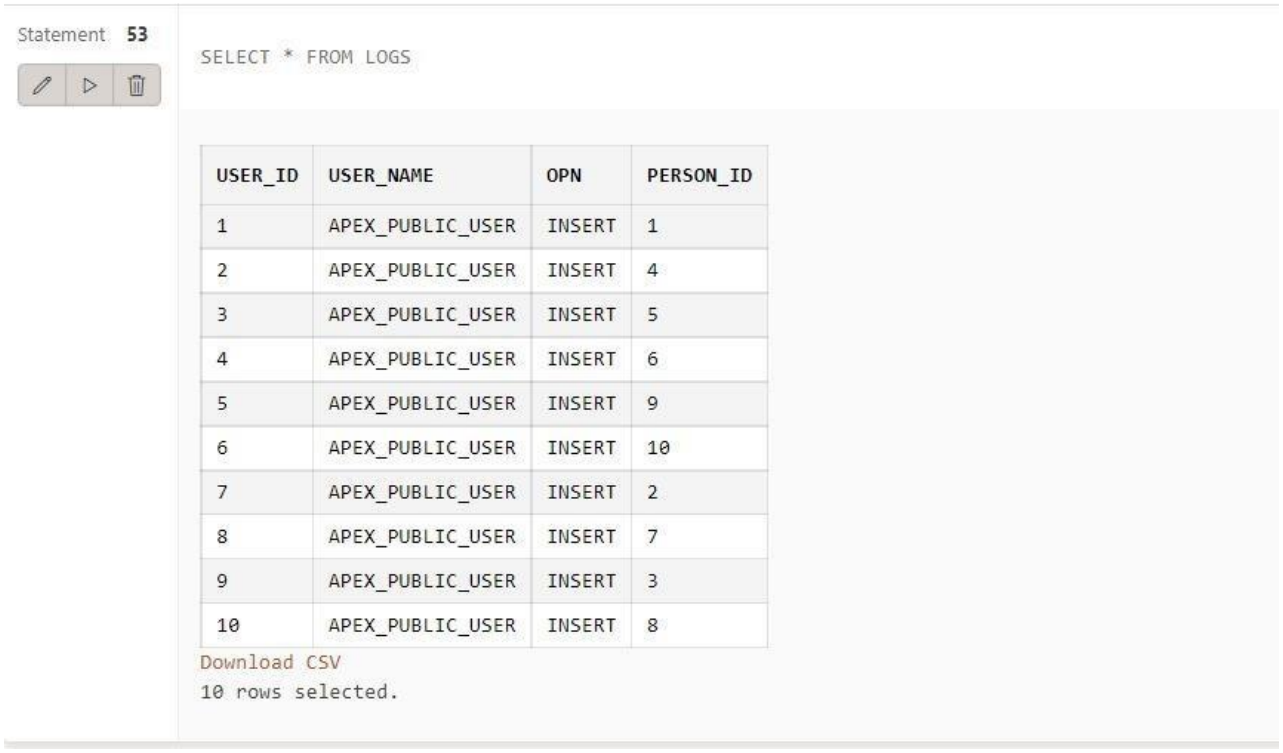


```
EXEC DOC_TREATS_PAT(1, 1, 1,3, NULL, NULL)
```

```
Statement processed.
```

```
EXEC CURS_PT_DOC(1)
```

```
Statement processed.
DOCTOR NAME: ARPAN SHARMA
PATIENT NAME: PHALIT JOTWANI
```



Statement 53

```
SELECT * FROM LOGS
```

USER_ID	USER_NAME	OPN	PERSON_ID
1	APEX_PUBLIC_USER	INSERT	1
2	APEX_PUBLIC_USER	INSERT	4
3	APEX_PUBLIC_USER	INSERT	5
4	APEX_PUBLIC_USER	INSERT	6
5	APEX_PUBLIC_USER	INSERT	9
6	APEX_PUBLIC_USER	INSERT	10
7	APEX_PUBLIC_USER	INSERT	2
8	APEX_PUBLIC_USER	INSERT	7
9	APEX_PUBLIC_USER	INSERT	3
10	APEX_PUBLIC_USER	INSERT	8

Download CSV

```
10 rows selected.
```