Educational Testing Service

DATABASE DESIGN CS6360.002

Team 11

Nishitha Vellalachervu (nxv170011)

Sachi Patel(srp170009)

Sushrut Patnaik (sxp175331)

TABLE OF CONTENTS

Introduction:

We are designing a testing service similar to the actual Educational Testing Service. We have tried to incorporate services provided by the actual ETS organization. However, our project is more generic version of a testing service and not an exact replica of the ETS system

Requirements:

We have brainstormed and finalized on a couple of entities that are required in the system.

- **Student**: A student can be considered as an integral part of the model. It has attributes like user_id, name, address, DOB, Password, etc. We have assumed that the user_id is unique for each student in the system.

- **Test:** The test is the most important entity in the system as everything else revolves around this entity. The test has attributes like test_id , test_name, test_date and score. test_id is like a uniquely generated number for each test that any student registers for. Test_name can have values like GRE, TOEFL , IELTS, and other subject specific test names. Each student has a register for test which will have a unique test_id.

- **Testing Center:** The testing Center is also an important part of our model. The test will be conducted at the testing center. It has attributes like a center_id, center name and the address.

- **Employee:** The testing center conducting the test will certainly have some employees working for them. So these entity will hold information about them. Employees have attributes like a unique ssn , Name, salary, phone numbers. Here we have classified the employee entity into two disjoint subclasses namely 'admin' and 'proctor'
    1. Admin : The admin is a subclass of employee has an attribute called role. So here role can basically mean an HR role, technical role, etc. The admin basically is a type of manager who manages not just a center but has specific management roles in it.
    2. Proctor : The proctor is responsible for monitoring and carrying out the tests at the testing center. It has an attribute called Specialization which can refer to a particular subject or exam that he or she is an expert on.

- **Payment:** Payment is an entity which has attributes like a Trans_id which is unique. The other attributes are trans_date, amount, type , status . For every test that a student registers for will have a payment with a trans_id.

- **Results:** Result is another important part of our system which has the score of the student. However from the final ER diagram and model we have removed this entity as it was a weak entity with a 1:1 relationship with the entity Test.

Relationships and Assumptions:

- Each student can **register** for multiple tests and for each test there will be one and only one student. Please note that here each test doesn't mean a particular test like GRE or TOEFL. Here each test is a unique with a unique test_id for a particular student. So student:test is a 1:N relation

- Student **makes** a payment (unique) for each test. Each student can make multiple payments and for each payment there can be only one student

- Each payment with a unique trans_id will be **received** for a single test. The relation between payment and test is a 1:1 relation.

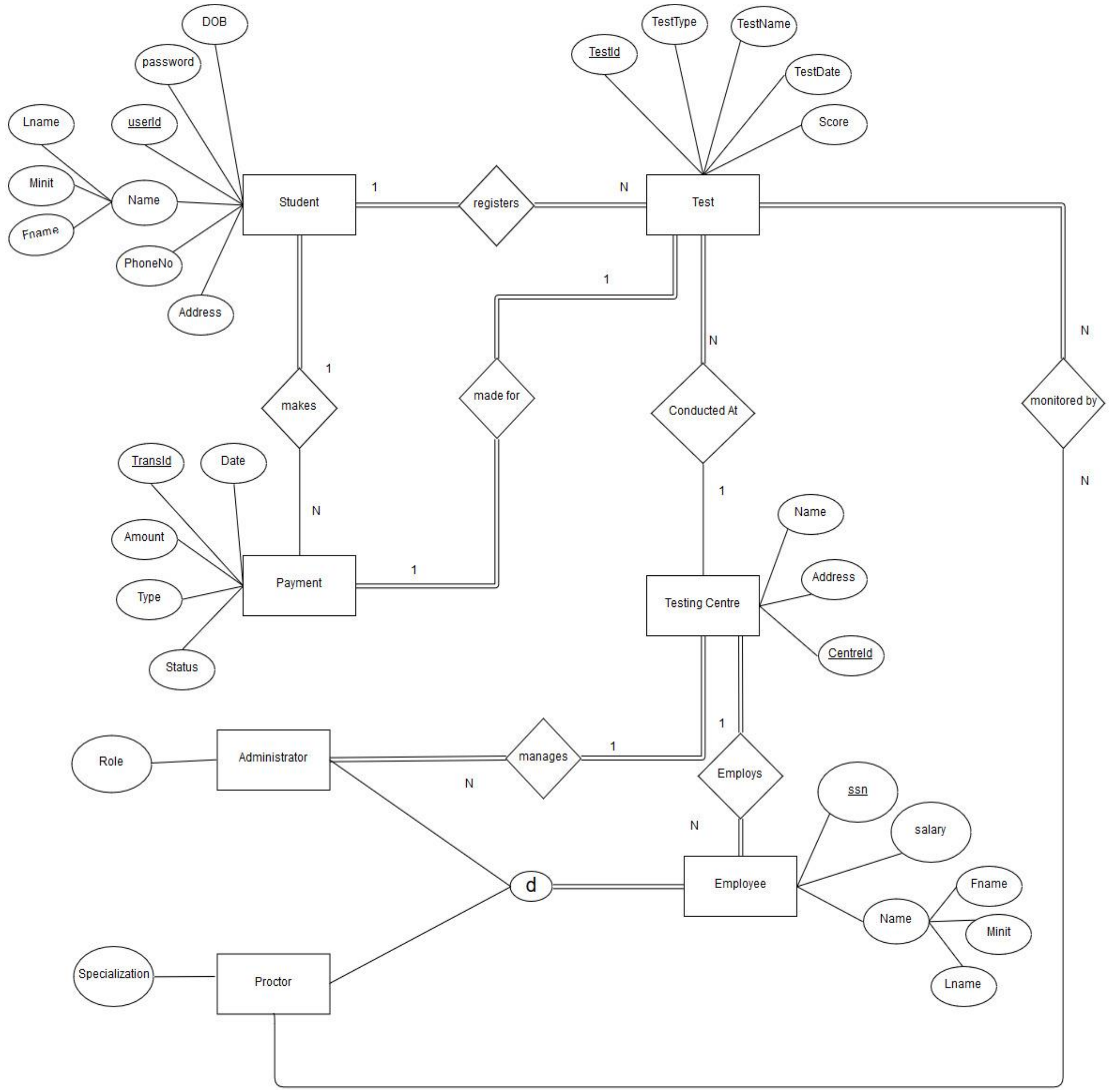
- Each test will be **conducted** at a particular testing center. Each testing center **can have** multiple tests. The relationship between testing center and test is 1:N relation

- Each testing center **employs** multiple employees while a particular employee can only **work** for a single testing center. So the relationship here also a 1:N relation.

- An administrator will be **working** at a single testing center. A testing center can have multiple administrators. This is also a 1:N relation

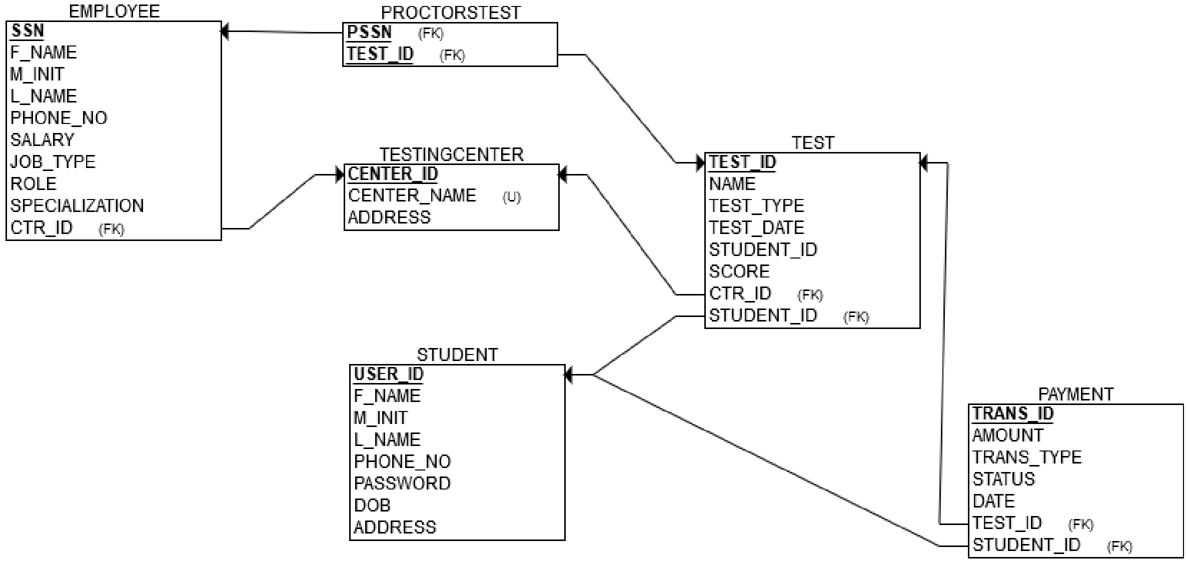
- A proctor can **monitor** multiple tests. Each test can be **monitored** by multiple proctors. So the monitors relation will be a M:N relation

Entity Relationship Diagram



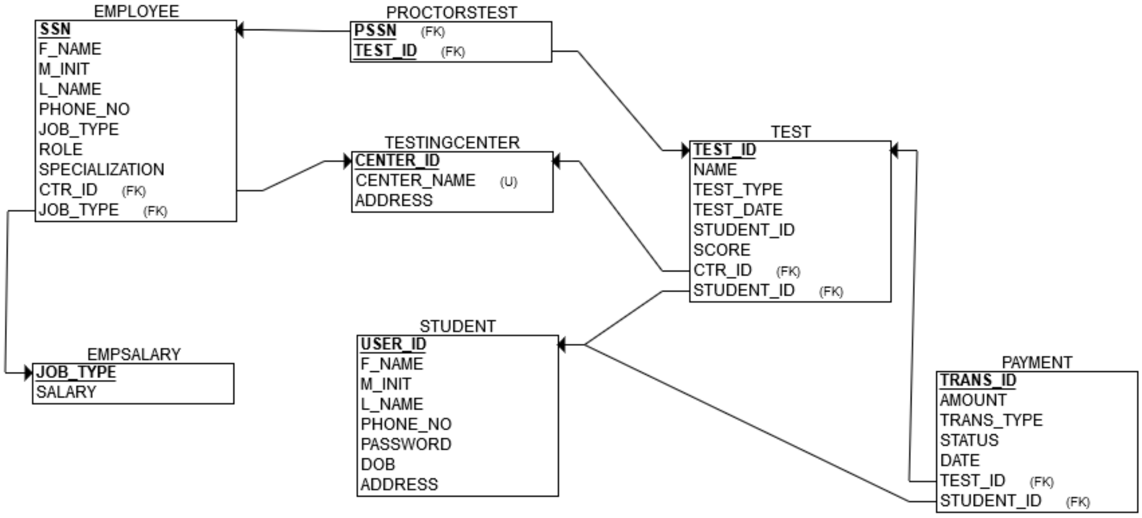
Relational Schema

The schema shown below is the schema before normalization.



We have followed relational schema mapping techniques to achieve the schema given above. After looking at each of the relations we can conclude that all the relations are in 2NF.

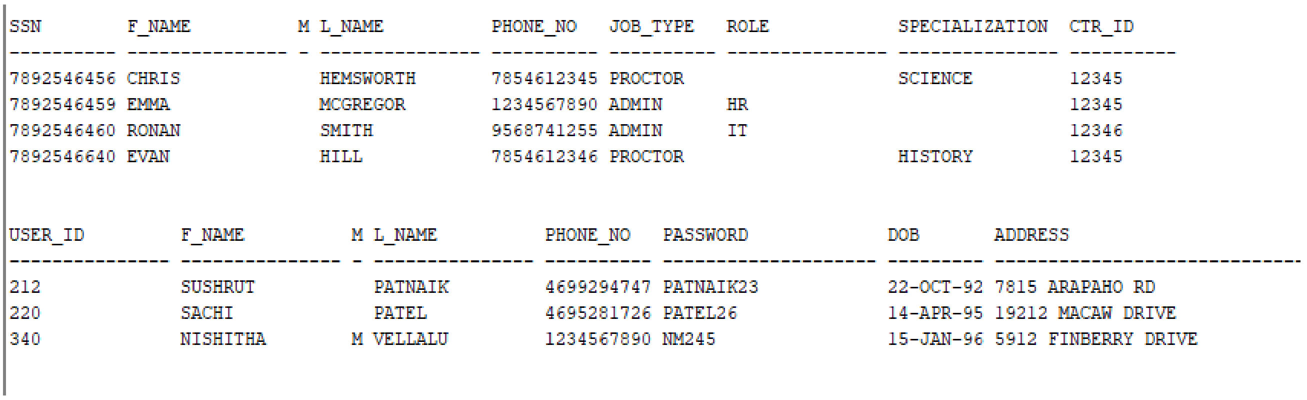
But if we look at the employee table, we can see that there is some transitive dependency. The Salary attribute of an employee will depend on the attribute 'Job_type'. To normalize this into 3NF we have created a new relation called 'EMPSALARY' and the new schema is given below.



SQL QUERIES

```
--DROP TABLE STUDENT;
CREATE TABLE STUDENT(
   USER_ID VARCHAR(15) NOT NULL,
   F_NAME VARCHAR(15) NOT NULL,
   M_INIT VARCHAR(1),
   L_NAME VARCHAR(15) NOT NULL,
   PHONE_NO CHAR(10),
   PASSWORD VARCHAR(20) NOT NULL,
   DOB DATE NOT NULL,
   ADDRESS VARCHAR(40) NOT NULL,
   PRIMARY KEY (USER_ID)

);

--DROP TABLE TESTINGCENTER;
CREATE TABLE TESTINGCENTER(
   CENTER_ID VARCHAR(10) NOT NULL,
   CENTER_NAME VARCHAR(20) NOT NULL,
   ADDRESS VARCHAR(40),
   PRIMARY KEY(CENTER_ID),
   UNIQUE(CENTER_NAME)

);

--DROP TABLE TEST;
CREATE TABLE TEST(
   TEST_ID VARCHAR(10) NOT NULL,
   NAME VARCHAR(10) NOT NULL,
   TEST_TYPE VARCHAR(10) NOT NULL,
   TEST_DATE DATE NOT NULL,
   STUDENT_ID VARCHAR(15) NOT NULL,
   SCORE DECIMAL,
   CTR_ID VARCHAR(10) NOT NULL,
   PRIMARY KEY (TEST_ID),
   FOREIGN KEY (STUDENT_ID) REFERENCES STUDENT(USER_ID),
   FOREIGN KEY (CTR_ID) REFERENCES TESTINGCENTER(CENTER_ID)
);

--DROP TABLE EMPLOYEE;
CREATE TABLE EMPLOYEE(
   SSN CHAR(10) NOT NULL,
   F_NAME VARCHAR(15) NOT NULL,
   M_INIT VARCHAR(1),
   L_NAME VARCHAR(15) NOT NULL,
   PHONE_NO CHAR(10),
   SALARY DECIMAL(10,2) NOT NULL,
   JOB_TYPE VARCHAR(10) NOT NULL,
   ROLE VARCHAR(15),
   SPECIALIZATION VARCHAR(15),
   CTR_ID VARCHAR(10) NOT NULL,
   PRIMARY KEY (SSN),
   FOREIGN KEY (CTR_ID) REFERENCES TESTINGCENTER(CENTER_ID)
);

--DROP TABLE PROCTORSTEST;
CREATE TABLE PROCTORSTEST(
   PSSN CHAR(10) NOT NULL,
   TEST_ID VARCHAR(10) NOT NULL,
   PRIMARY KEY (TEST_ID,PSSN),
   FOREIGN KEY (TEST_ID) REFERENCES TEST(TEST_ID),
   FOREIGN KEY (PSSN) REFERENCES EMPLOYEE(SSN)
);

--DROP TABLE PAYMENT;
CREATE TABLE PAYMENT(
   TRANS_ID VARCHAR(10) NOT NULL,
   AMOUNT INTEGER NOT NULL,
   TRANS_TYPE VARCHAR(10) NOT NULL,
   TRANS_DATE DATE NOT NULL,
   STATUS VARCHAR(10) DEFAULT 'PENDING' ,
   STUDENT_ID VARCHAR(15) NOT NULL,
   TEST_ID VARCHAR(10) NOT NULL,
   PRIMARY KEY(TRANS_ID),
   FOREIGN KEY (STUDENT_ID) REFERENCES STUDENT(USER_ID),
   FOREIGN KEY (TEST_ID) REFERENCES TEST(TEST_ID)
);
```

Screenshots of the Employee and Student table

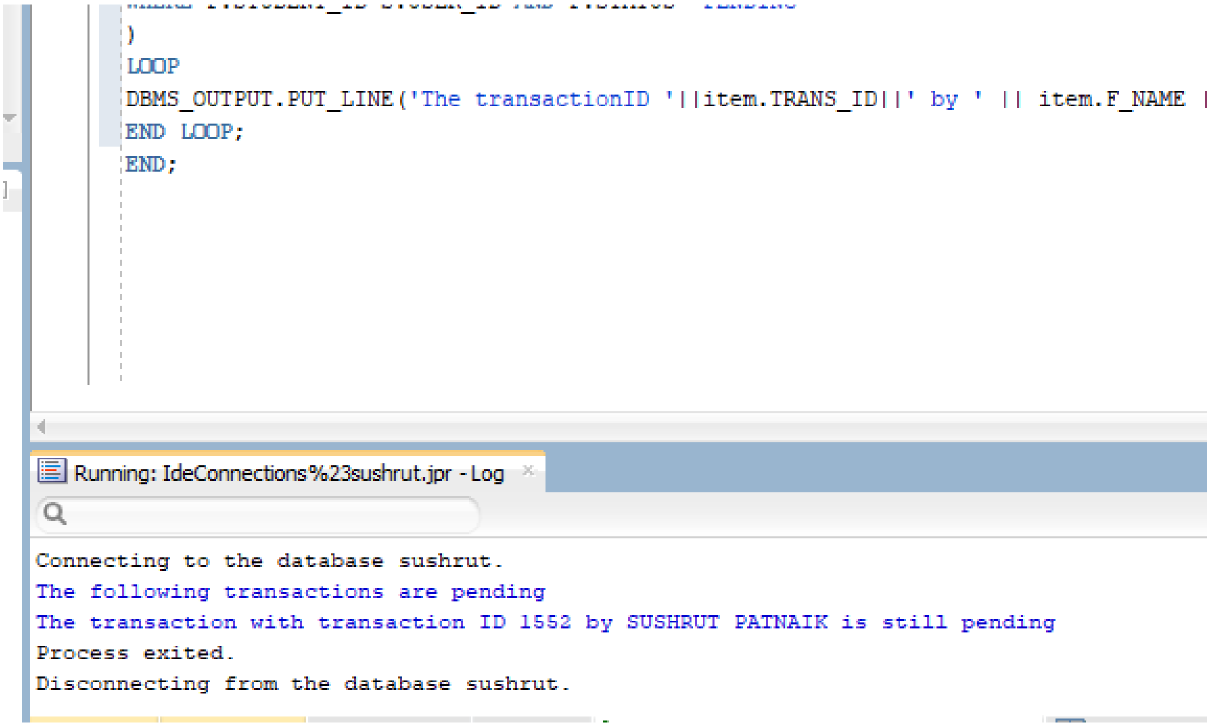


```
SSN        F_NAME          M L_NAME          PHONE_NO   JOB_TYPE   ROLE            SPECIALIZATION  CTR_ID
---------- --------------- - --------------- ---------- ---------- --------------- --------------- ----------
7892546456 CHRIS             HEMSWORTH       7854612345 PROCTOR                    SCIENCE         12345
7892546459 EMMA              MCGREGOR        1234567890 ADMIN      HR                              12345
7892546460 RONAN             SMITH           9568741255 ADMIN      IT                              12346
7892546640 EVAN              HILL            7854612346 PROCTOR                    HISTORY         12345


USER_ID         F_NAME          M L_NAME          PHONE_NO   PASSWORD             DOB       ADDRESS
--------------- --------------- - --------------- ---------- -------------------- --------- ------------------------------
212             SUSHRUT           PATNAIK         4699294747 PATNAIK23            22-OCT-92 7815 ARAPAHO RD
220             SACHI             PATEL           4695281726 PATEL26              14-APR-95 19212 MACAW DRIVE
340             NISHITHA        M VELLALU         1234567890 NM245                15-JAN-96 5912 FINBERRY DRIVE
```

Creating Stored Procedures

a) **Pending transactions:** This procedure displays all transactions that are in pending status.

```
CREATE OR REPLACE
PROCEDURE REPORT_PENDING_TRANSACTIONS
AS
BEGIN
DBMS_OUTPUT.PUT_LINE ('The following transactions are pending');
FOR item IN
(
SELECT TRANS_ID, F_NAME,L_NAME
FROM PAYMENT P,STUDENT S
WHERE P.STUDENT_ID=S.USER_ID AND P.STATUS='PENDING'
)
LOOP
DBMS_OUTPUT.PUT_LINE('The transactionID '||item.TRANS_ID||' by ' || item.F_NAME
||item.L_NAME||' is still pending');
END LOOP;
END;
```

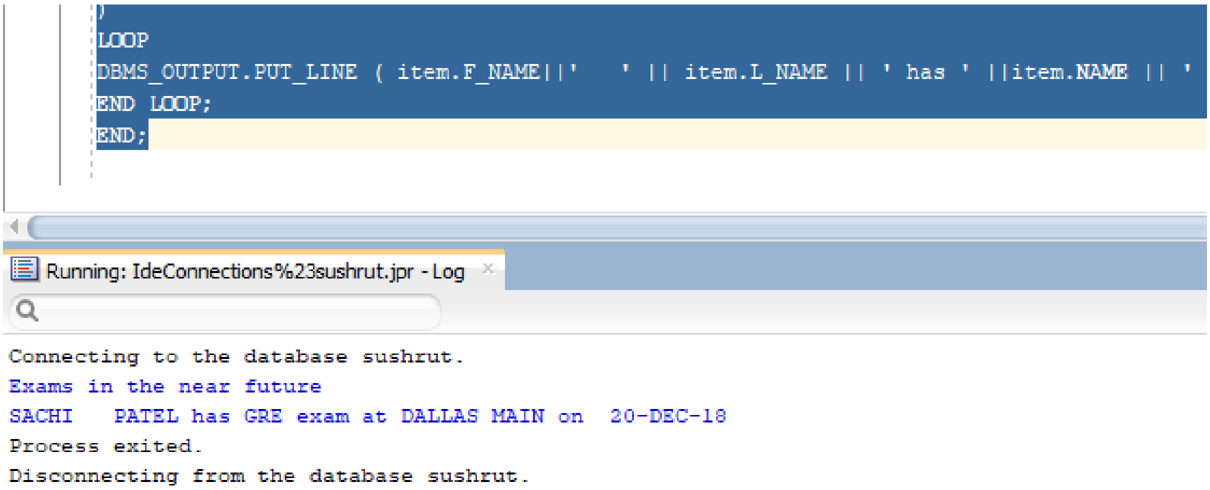


```
)
LOOP
DBMS_OUTPUT.PUT_LINE('The transactionID '||item.TRANS_ID||' by ' || item.F_NAME |
END LOOP;
END;
```

```
Connecting to the database sushrut.
The following transactions are pending
The transaction with transaction ID 1552 by SUSHRUT PATNAIK is still pending
Process exited.
Disconnecting from the database sushrut.
```

b) Display upcoming exams: This procedure displays all upcoming exams at a center

```
CREATE OR REPLACE
PROCEDURE UPCOMING_EXAMS
AS
BEGIN
DBMS_OUTPUT.PUT_LINE ('Exams in the near future');
FOR item IN
(
SELECT T.NAME,T.TEST_DATE, S.F_NAME, S.L_NAME, C.CENTER_NAME
FROM STUDENT S, TEST T, TESTINGCENTER C
WHERE S.USER_ID= T.STUDENT_ID AND C.CENTER_ID=T.CTR_ID AND T.TEST_DATE > SYSDATE
)
LOOP
DBMS_OUTPUT.PUT_LINE ( item.F_NAME||'  ' || item.L_NAME || ' has ' ||item.NAME || ' exam
at ' || item.CENTER_NAME || ' on  '|| item.TEST_DATE );
END LOOP;
END;
```



```
)
LOOP
DBMS_OUTPUT.PUT_LINE ( item.F_NAME||'    ' || item.L_NAME || ' has ' ||item.NAME || '
END LOOP;
END;
```

```
Connecting to the database sushrut.
Exams in the near future
SACHI   PATEL has GRE exam at DALLAS MAIN on  20-DEC-18
Process exited.
Disconnecting from the database sushrut.
```

Creating Triggers

a) Trigger when a new employee joins

```
CREATE OR REPLACE TRIGGER trigger_new_employee
AFTER INSERT
  on EMPLOYEE
  FOR EACH ROW
BEGIN
        DBMS_OUTPUT.PUT_LINE('New employee has been added');
END;
```

b) Trigger when a student tries to schedule a test within 21 days

```
CREATE OR REPLACE TRIGGER trigger_new_test
BEFORE INSERT ON TEST
FOR EACH ROW
DECLARE
LATEST;
BEGIN
SELECT TEST_DATE INTO LATEST FROM TEST WHERE STUDENT_ID=:NEW.STUDENT_ID;
IF(DATEDIFF(DAY, LATEST, GETDATE())<21)
THEN
RAISE_APPLICATION_ERROR(-20001, ‘CANNOT APPEAR FOR EXAM AGAIN WITHIN 21
DAYS!!');
END IF;
END;
```