// drop all tables first & sequences first

Drop table accounts cascade constraints;

Drop table company cascade constraints;

Drop table job\_seeker cascade constraints;

Drop table skills cascade constraints;

Drop table job\_seeker\_skills cascade constraints;

Drop table job\_posts cascade constraints;

Drop table job\_app cascade constraints;

Drop table message cascade constraints;

-- Drop existing procedure if it exists

DROP PROCEDURE apply\_for\_job\_post;

DROP PROCEDURE assign\_initial\_score;

--Drop existing sequence if it exists

drop sequence seq\_account\_id;

drop sequence seq\_company\_id;

drop sequence seq\_job\_id;

drop sequence seq\_skill\_id;

drop sequence seq\_job\_app\_id;

drop sequence seq\_message\_id;

// create statements

Create table accounts

(accid int,

acc\_holder\_name varchar (50),

address varchar (50),

zipcode varchar (50),

state varchar (50),

email varchar (50),

phone\_number varchar (50),

account\_type varchar (50),

registration\_date date,

primary key(accid)

);

Create table company (

cid int primary key,

company\_name varchar(255)

);

Create table job\_seeker

(accid integer,

highest\_degree int CHECK (highest\_degree IN (1,2,3,4)),

major varchar (50),

school varchar(100),

years\_exp integer,

primary key (accid),

foreign key (accid) references accounts(accid)

);

Create table skills (

sid int primary key,

skill\_name varchar(255)

);

Create table job\_seeker\_skills

(sid int,

accid int,

slevel int CHECK (slevel IN (1,2,3)),

primary key (sid, accid),

foreign key (sid) references skills(sid),

foreign key (accid) references job\_seeker (accid)

);

CREATE TABLE job\_posts(

job\_id NUMBER PRIMARY KEY,

company\_id NUMBER REFERENCES company(cid),

recruiter\_account\_id NUMBER REFERENCES accounts(accid),

job\_title VARCHAR2(100) NOT NULL, description VARCHAR2(500),

job\_type VARCHAR2(20) CHECK(job\_type IN ('full time', 'part-time')),

min\_pay\_range NUMBER,

max\_pay\_range NUMBER,

job\_city VARCHAR2(50),

job\_state VARCHAR2(20),

min\_degree\_level NUMBER CHECK(min\_degree\_level IN (1, 2, 3, 4)),

min\_experience NUMBER,

status NUMBER CHECK(status IN (0, 1))

);

create table job\_app(

appid int,

accid int,

job\_id int,

app\_date date,

app\_status int,

primary key(appid),

foreign key (accid) references job\_seeker(accid),

foreign key (job\_id) references job\_posts(job\_id)

);

create table message(

m\_id int,

accid int,

m\_time timestamp,

m\_body varchar (300),

foreign key (accid) references accounts(accid)

);

//sequence for account ID creation

CREATE SEQUENCE seq\_account\_id

start with 1

increment by 1

Minvalue 0

Maxvalue 100

Cycle;

//sequence for companyID creation

CREATE SEQUENCE seq\_company\_id

start with 1

increment by 1

Minvalue 0

Maxvalue 100

Cycle;

//sequence for job\_post ID creation

CREATE SEQUENCE seq\_job\_id

start with 1

increment by 1

Minvalue 0

Maxvalue 100

Cycle;

//sequence for skill ID creation

CREATE SEQUENCE seq\_skill\_id

start with 1

increment by 1

Minvalue 0

Maxvalue 100

Cycle;

//sequence for job app ID creation

CREATE SEQUENCE seq\_job\_app\_id

start with 1

increment by 1

Minvalue 0

Maxvalue 100

Cycle;

//sequence for newly generated messages

CREATE SEQUENCE seq\_message\_id

start with 1

increment by 1

Minvalue 0

Maxvalue 100

Cycle;

//insert statements

Insert into accounts

values (seq\_account\_id.NEXTVAL, 'John Bennett', '123 Main Streett', 56723, 'Oklahoma', 'john\_bennet@yahoo.mail', '539-421-9809', 'applicant', date '2023-3-9');

Insert into accounts

values (seq\_account\_id.NEXTVAL, 'Jane Smith', '456 Oak Avenue', 23214, 'New York', 'janesmith123@gmail.com', '212-421-9809', 'applicant', date '2023-9-19' );

Insert into accounts

values (seq\_account\_id.NEXTVAL, 'Michael Harper', '753 Elms Drive', 92021, 'California', 'michael.h@gmail.com', '415-404-8888', 'applicant', date '2023-5-10');

Insert into company

values (seq\_company\_id.NEXTVAL, 'Company A');

Insert into company

values(seq\_company\_id.NEXTVAL, 'Company B');

Insert into company

values(seq\_company\_id.NEXTVAL, 'Company C');

Insert into job\_seeker

values (1, 2, 'Computer Science', 'UMBC', 3);

Insert into job\_seeker

Values (2, 3, 'Biology', 'UMD', 1);

Insert into job\_seeker

Values (3, 2, 'Chemistry', 'UCLA', 0);

Insert into skills

values(seq\_skill\_id.NEXTVAL, 'Java');

Insert into skills

values(seq\_skill\_id.NEXTVAL, 'Python');

Insert into skills

values(seq\_skill\_id.NEXTVAL, 'SQL');

Insert into skills

values(seq\_skill\_id.NEXTVAL, 'PL/SQL');

insert into job\_seeker\_skills

values(1, 1, 3 );

Insert into job\_seeker\_skills

values(2, 1, 2);

Insert into job\_seeker\_skills

Values(3, 2, 1);

INSERT INTO job\_posts VALUES (seq\_job\_id.NEXTVAL, 1, 2, 'Software Developer', 'Join our innovative team!', 'full time', 80000, 100000, 'Fairfax', 'VA', 2, 2, 1);

INSERT INTO job\_posts VALUES (seq\_job\_id.NEXTVAL, 2, 2, 'Data Scientist', 'Exciting opportunity in data science!', 'part-time', 60000, 80000, 'Los Angeles', 'CA', 3, 3, 1);

INSERT INTO job\_posts VALUES (seq\_job\_id.NEXTVAL, 3, 1, 'Product Manager', 'Exciting opportunity to lead product development!', 'full time', 90000, 120000, 'San Francisco', 'CA', 3, 5, 1);

Insert into job\_app

values (seq\_job\_app\_id.NEXTVAL, 1, 1, date '2024-03-08', 1);

Insert into job\_app

values(seq\_job\_app\_id.NEXTVAL, 2, 2, date '2024-03-08', 1);

Insert into job\_app

values(seq\_job\_app\_id.NEXTVAL, 3, 3, date '2024-03-08', 1);

Insert into message

values (seq\_message\_id.NEXTVAL, 1, timestamp '2024-03-08 10:00:00', 'I would like to discuss your interest in this role and explore further opportunities. Looking forward to hearing from you.');

Insert into message

values(seq\_message\_id.NEXTVAL, 2, timestamp '2024-03-08 10:05:00', 'Feel free to check out our job listings or let me know if you would like more information about specific roles.');

Insert into message

values(seq\_message\_id.NEXTVAL, 3, timestamp '2024-03-08 10:10:00', 'If you are open to new opportunities, I would love to discuss the details with you. Looking forward to your response.');

-- Procedure to apply for job posts with validation checks for user ID and job post ID

CREATE OR REPLACE PROCEDURE apply\_for\_job\_posts (v\_accid IN INT, v\_job\_id IN INT, v\_date IN DATE) IS

v\_count INT; -- Variable to store count of valid user IDs

v\_count\_2 INT; -- Variable to store count of valid job post IDs

BEGIN

-- Check if the provided account ID exists and is of type 'applicant'

SELECT COUNT(\*) INTO v\_count FROM accounts WHERE accid = v\_accid AND account\_type = 'applicant';

IF v\_count = 0 THEN

-- If the user ID is invalid, display error message and exit the procedure

DBMS\_OUTPUT.PUT\_LINE('Invalid user ID');

RETURN;

END IF;

-- Check if the provided job ID exists and is active

SELECT COUNT(\*) INTO v\_count\_2 FROM job\_posts WHERE job\_id = v\_job\_id AND status = 1;

IF v\_count\_2 = 0 THEN

-- If the job post ID is invalid or inactive, display error message and exit the procedure

DBMS\_OUTPUT.PUT\_LINE('Invalid job post ID');

RETURN;

END IF;

-- Insert new job application into job\_app table

INSERT INTO job\_app (appid, accid, job\_id, app\_date, app\_status)

VALUES (seq\_job\_app\_id.NEXTVAL, v\_accid, v\_job\_id, v\_date, 1);

-- Display success message with the ID of the newly created job application

DBMS\_OUTPUT.PUT\_LINE('New application ' || seq\_job\_app\_id.CURRVAL || ' created');

END;

/

set serveroutput on;

-- Test Case 1: Applying for a job post with an invalid account ID

-- Display the details of the job post

SELECT \* FROM job\_posts WHERE job\_id = 9999;

-- Execute the procedure with an invalid account ID

EXEC apply\_for\_job\_posts(1, 9999, SYSDATE);

-- Display the updated job application table to verify the new application

SELECT \* FROM job\_app WHERE accid = 1;

-- Test Case 2: Applying for a job post with an invalid job post ID

-- Display the details of the job post

SELECT \* FROM job\_posts WHERE accid = 9999;

-- Execute the procedure with an invalid job post ID

EXEC apply\_for\_job\_posts(9999, 1, SYSDATE);

-- Display the updated job application table to verify no new application is added

SELECT \* FROM job\_app WHERE job\_id = 1;

-- Test Case 3: Applying for a job post with a valid account ID and job post ID

-- Display the details of the job post

SELECT \* FROM job\_posts WHERE job\_id = 1;

-- Execute the procedure with a valid account ID and job post ID

EXEC apply\_for\_job\_posts(1, 1, SYSDATE);

-- Display the updated job application table to verify the new application

SELECT \* FROM job\_app WHERE accid = 1;

CREATE OR REPLACE PROCEDURE assign\_initial\_score(job\_post\_id\_in IN NUMBER) IS

v\_min\_degree\_level NUMBER;

v\_min\_experience NUMBER;

v\_job\_title VARCHAR2(100);

v\_job\_type VARCHAR2(20);

BEGIN

-- Step 1: Check if the job post exists and is active

BEGIN

SELECT min\_degree\_level, min\_experience, job\_title, job\_type

INTO v\_min\_degree\_level, v\_min\_experience, v\_job\_title, v\_job\_type

FROM job\_posts

WHERE job\_id = job\_post\_id\_in AND status = 1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid job post ID');

RETURN;

END;

-- Step 2: Find eligible job seekers

FOR job\_seeker\_rec IN (

SELECT js.accid, js.highest\_degree, js.years\_exp, a.acc\_holder\_name

FROM job\_seeker js

JOIN job\_app ja ON js.accid = ja.accid

JOIN accounts a ON js.accid = a.accid

WHERE ja.job\_id = job\_post\_id\_in

AND js.highest\_degree >= v\_min\_degree\_level

AND js.years\_exp >= v\_min\_experience

)

LOOP

-- Step 3: Compute score for each job seeker

DECLARE

v\_score NUMBER := 0;

v\_skills\_count NUMBER := 0;

BEGIN

-- Calculate the basic score

v\_score := job\_seeker\_rec.years\_exp + (job\_seeker\_rec.highest\_degree - v\_min\_degree\_level) \* 2;

-- Calculate the number of required skills satisfied by the job seeker

SELECT COUNT(\*)

INTO v\_skills\_count

FROM job\_seeker\_skills jss

JOIN skills s ON jss.sid = s.sid

WHERE jss.accid = job\_seeker\_rec.accid

AND jss.slevel >= 1;

-- Update the score with the number of skills

v\_score := v\_score + v\_skills\_count;

-- Step 4: Print job seeker's name and score

DBMS\_OUTPUT.PUT\_LINE('Job Seeker: ' || job\_seeker\_rec.acc\_holder\_name || ', Score: ' || v\_score);

END;

END LOOP;

END;

/

SET SERVEROUTPUT ON;

-- Execute the procedure for a valid job post ID (e.g., job\_id = 1)

EXEC assign\_initial\_score(1);

-- Verify the accounts table to see job seeker details

SELECT \* FROM job\_posts;

SET SERVEROUTPUT ON;

-- Execute the procedure for an invalid job post ID (e.g., job\_id = 99, which does not exist)

EXEC assign\_initial\_score(99);

-- Verify the accounts table to confirm no changes

SELECT \* FROM job\_posts;

SET SERVEROUTPUT ON;

-- Assuming job\_id = 2 has no applicants meeting the minimum degree and experience requirements

EXEC assign\_initial\_score(2);

-- Verify the accounts table to see job seeker details

SELECT \* FROM job\_posts;

SET SERVEROUTPUT ON;

-- Execute the procedure for a job post with mixed applicants (e.g., job\_id = 3)

EXEC assign\_initial\_score(3);

-- Verify the accounts table to see job seeker details

SELECT \* FROM job\_posts;