HOMEWORK 4
KAIJUN HE

```
CREATE OR REPLACE PROCEDURE CREATE_ALL_TABLES IS
   user_table VARCHAR2(500):=  'CREATE TABLE USERS(
                USERID INT,
                GENDER VARCHAR2(5),
                AGECODE VARCHAR2(50),
                OCCIPATION VARCHAR2(50),
                ZIPCODE VARCHAR2(50),
                PRIMARY KEY(USERID))';
   movie_table VARCHAR2(500)  :=  'CREATE TABLE MOVIES (
                                MOVIEID INT,
                                TITLE VARCHAR2(200),
                                YEAR INT,
                                PRIMARY KEY(MOVIEID))';
   movie_genres_table VARCHAR2(500):=  'CREATE TABLE MOVIEGENRES (
                                        MOVIEID INT,
                                        GENRES VARCHAR2(100),
                                        PRIMARY KEY (MOVIEID, GENRES),
                    FOREIGN KEY (MOVIEID) REFERENCES MOVIES(MOVIEID))';
   rating_table VARCHAR2(500) := 'CREATE TABLE RATINGS (
                                 USERID INT,
                                 MOVIEID INT,
                                 RATING INT,
                                 TIMESTAMPS TIMESTAMP,
                                 PRIMARY KEY (USERID, MOVIEID),
                                 FOREIGN KEY (MOVIEID) REFERENCES MOVIES(MOVIEID))';
   BEGIN
      DBMS_OUTPUT.PUT_LINE('CREATE PROCEDURE TO CREATE 4 TABLES ACCORDING README FILE
AND HOMEWORK INSTRUCTION');
```

HOMEWORK 4
KAIJUN HE

```
    EXECUTE IMMEDIATE user_table;

    EXECUTE IMMEDIATE movie_table;

    EXECUTE IMMEDIATE movie_genres_table;

    EXECUTE IMMEDIATE rating_table;

        END CREATE_ALL_TABLES;


CREATE OR REPLACE PROCEDURE DROP_ALL_TABLES IS

  drop_rating_table VARCHAR2(500) := 'DROP TABLE RATINGS';

   drop_movie_genres_table VARCHAR2(500)  := 'DROP TABLE MOVIEGENRES';

        drop_user_table VARCHAR2(500)  := 'DROP TABLE USERS';

  drop_movie_table VARCHAR2(500):= 'DROP TABLE MOVIES';

  BEGIN

    DBMS_OUTPUT.PUT_LINE('CREATE PROCEDURE TO DROP ALL CREATED TABLE');

    EXECUTE IMMEDIATE drop_movie_genres_table;

    EXECUTE IMMEDIATE drop_rating_table;

        EXECUTE IMMEDIATE drop_movie_table;

        EXECUTE IMMEDIATE drop_user_table;

  END DROP_ALL_TABLES;



CREATE OR REPLACE PROCEDURE PARSE_USERS_DATA IS

BEGIN

        DECLARE

                age_code VARCHAR2(500);

                occupation VARCHAR2(50);

    store_data VARCHAR2(500) := 'INSERT INTO USERS
VALUES(:USERID, :GENDER, :AGECODE, :OCCUPATION, :ZIPCODE)';

                CURSOR z_user_info IS

                        SELECT *
```

HOMEWORK 4
KAIJUN HE

```
FROM usersxlsx;
        r_user_info z_user_info%ROWTYPE;
  BEGIN
DBMS_OUTPUT.PUT_LINE('START TO FETCH CREATED CURSOR TO DO THE DECODING AGECODE AND OCCUPATION BY GIVEN INFORMATION IN README, WHERE I WILL USE CASE TO CONVERT GIVEN AGECODE TO A RANGE OF AGE AND REALL OCCUPATION ');
        OPEN z_user_info;
        LOOP
            FETCH z_user_info INTO r_user_info;
            EXIT WHEN z_user_info%NOTFOUND;
            CASE r_user_info.occupation
            WHEN 0 THEN
                occupation := 'other';
            WHEN 1 THEN
                occupation := 'academic/educator';
            WHEN 2 THEN
                occupation := 'artist';
            WHEN 3 THEN
                occupation := 'clerical/admin';
            WHEN 4 THEN
                occupation := 'college/grad student';
            WHEN 5 THEN
                occupation := 'customer service';
            WHEN 6 THEN
                occupation := 'doctor/health care';
            WHEN 7 THEN
                occupation := 'executive/managerial';
            WHEN 8 THEN
                occupation := 'farmer';
```

```
WHEN 9 THEN
        occupation := 'homemaker';
WHEN 10 THEN
        occupation := 'K-12 student';
WHEN 11 THEN
        occupation := 'lawyer';
WHEN 12 THEN
        occupation := 'programmer';
WHEN 13 THEN
        occupation := 'retired';
WHEN 14 THEN
        occupation := 'sales/marketing';
WHEN 15 THEN
        occupation := 'scientist';
WHEN 16 THEN
        occupation := 'self-employed';
WHEN 17 THEN
        occupation := 'technician/engineer';
WHEN 18 THEN
        occupation := 'tradesman/craftsman';
WHEN 19 THEN
        occupation := 'unemployed';
WHEN 20 THEN
        occupation := 'writer';
END CASE;
CASE r_user_info.AGE
WHEN 1 THEN
        age_code := 'Under 18';
WHEN 18 THEN
```

```
                    age_code := '18-24';
                WHEN 25 THEN
                    age_code := '25-34';
                WHEN 35 THEN
                    age_code := '35-44';
                WHEN 45 THEN
                    age_code := '45-49';
                WHEN 50 THEN
                    age_code := '50 -55';
                WHEN 56 THEN
                    age_code := '56+';
                END CASE;
                EXECUTE IMMEDIATE store_data
                    USING r_user_info.USERID, r_user_info.GENDER, age_code, occupation,
r_user_info.ZIPCODE;
            END LOOP;
            CLOSE z_user_info;
        END;
END PARSE_USERS_DATA;


CREATE OR REPLACE PROCEDURE PARSE_MOVIES_DATA IS
BEGIN
        DECLARE
            movie_data VARCHAR2(500) := 'INSERT INTO movies VALUES(:movieid, :title, :years)';
            titles VARCHAR2(500);
            years INT;
    movie_genres VARCHAR2(500);
            movie_genres_data VARCHAR2(100)  := 'INSERT INTO MOVIEGENRES
VALUES(:MOVIEID, :GENRES)';
```

HOMEWORK 4
KAIJUN HE

```
        CURSOR z_movie_info IS
            SELECT *
            FROM moviesxlsx;
        r_movie_info z_movie_info%ROWTYPE;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('THIS PROCEDURE IS USED TO STORED GIVEN DATA INTO MY
OWN CREATED TABLE BY USING SUBSTR FUNCTION IN ORACLE DATABASE
            BY OBSERVE THE DATA GIVEN WE KNOW THE LAST SIX DIGIT DATA IS (YYYY),
THEREFORE WE CAN SIMPLEY DIVIDED INTO TWO PART TITLE NAME AND YEAR
            , AND I CREATE A MOVIEID-GENRES RELATIONSHIP TABLE AND WE COULD
SEPERATE THE GENRES BY THE | SIGN, WHERE I AM USING REGEXP TO DO THIS');
        OPEN z_movie_info;
        LOOP
            FETCH z_movie_info INTO r_movie_info;
            EXIT WHEN z_movie_info%NOTFOUND;
            years := SUBSTR(r_movie_info.TITLE, -5, 4);
            titles := SUBSTR(r_movie_info.TITLE, 1, LENGTH(r_movie_info.TITLE) - 6);
            EXECUTE IMMEDIATE movie_data
                USING r_movie_info.movieid, titles, years;
            FOR i IN 1..10 LOOP
                movie_genres := regexp_substr(r_movie_info.GENRES, '[^|]+', 1, i);
                IF LENGTH(movie_genres) > 0 THEN
                    EXECUTE IMMEDIATE movie_genres_data
                        USING r_movie_info.MOVIEID, movie_genres;
                END IF;

            END LOOP;
        END LOOP;
  CLOSE z_movie_info;
    END;
```

HOMEWORK 4
KAIJUN HE

```
END PARSE_MOVIES_DATA;


CREATE OR REPLACE PROCEDURE PARSE_RATINGS_DATA IS

BEGIN

	DECLARE

		RATINGS_DATA VARCHAR2(500) := 'INSERT INTO RATINGS VALUES
(:USERID, :MOVIEID, :RATING, :TIMESTAMPS)';

		CURSOR z_ratings_info IS

			SELECT *

			FROM RATINGSXLSX;

		r_ratings_info z_ratings_info%ROWTYPE;

	BEGIN

		DBMS_OUTPUT.PUT_LINE('THI IS SIMPLEY FETCH GIVEN data only thing i should do is to
find out how to current epoch time by convert timestamps in second plus 01/01/1970 time ');

		OPEN z_ratings_info;

		LOOP

			FETCH z_ratings_info INTO r_ratings_info;

			EXIT WHEN z_ratings_info%NOTFOUND;

			EXECUTE IMMEDIATE RATINGS_DATA

				USING r_ratings_info.USERID, r_ratings_info.MOVIEID,
r_ratings_info.RATING,  to_date('01011970', 'MMDDYYYY') + (1/24/60/60)*r_ratings_info.TIMESTAMPS;

		END LOOP;

		CLOSE z_ratings_info;

	END;

END PARSE_RATINGS_DATA;


CREATE OR REPLACE PROCEDURE PROGRESS IS

BEGIN

 DROP_ALL_TABLES();

 CREATE_ALL_TABLES();
```

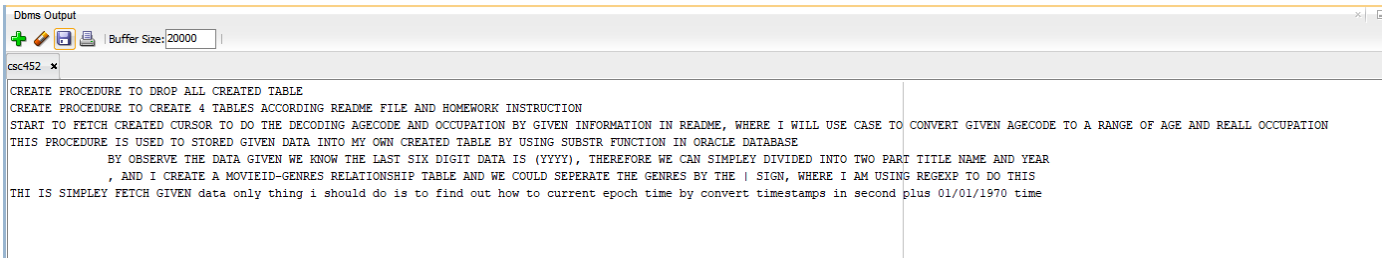
HOMEWORK 4
KAIJUN HE

```
 PARSE_USERS_DATA();

 PARSE_MOVIES_DATA();

 PARSE_RATINGS_DATA();

END;


begin

   progress();

   end;

-- queries i would like to do


-- COMPARE NUMBER OF HIGH RATING MOVIES BETWEEN 1993 AND 2000 FOR MALE USER AND
FEMALE USER

-- WE COULD OBSERVE THE RESULT THE NUMBER OF RATING IS DECREASING BUT THE RATIO OF
FEMALE AND MALE

--USER IS ALMOST KEEPING 3:1



SELECT COUNT(RATINGS.RATING) AS NUMBER_FEMALE_HIGH_RATING_IN_1993

FROM MOVIES

        INNER JOIN RATINGS ON RATINGS.MOVIEID = MOVIES.MOVIEID

        INNER JOIN USERS ON USERS.USERID = RATINGS.USERID

WHERE USERS.GENDER = 'F' AND MOVIES.YEAR = '1993' AND RATINGS.RATING = '5';


SELECT COUNT(RATINGS.RATING) AS NUMBER_MALE_HIGH_RATING_IN_1993

FROM MOVIES

        INNER JOIN RATINGS ON RATINGS.MOVIEID = MOVIES.MOVIEID

        INNER JOIN USERS ON USERS.USERID = RATINGS.USERID

WHERE USERS.GENDER = 'M' AND MOVIES.YEAR = '1993' AND RATINGS.RATING = '5';
```

HOMEWORK 4
KAIJUN HE

```
SELECT COUNT(RATINGS.RATING) AS NUMBER_FEMALE_HIGH_RATING_IN_2000

FROM MOVIES

        INNER JOIN RATINGS ON RATINGS.MOVIEID = MOVIES.MOVIEID

        INNER JOIN USERS ON USERS.USERID = RATINGS.USERID

WHERE USERS.GENDER = 'F' AND MOVIES.YEAR = '2000' AND RATINGS.RATING = '5';


SELECT COUNT(RATINGS.RATING) AS NUMBER_MALE_HIGH_RATING_IN_2000

FROM MOVIES

        INNER JOIN RATINGS ON RATINGS.MOVIEID = MOVIES.MOVIEID

        INNER JOIN USERS ON USERS.USERID = RATINGS.USERID

WHERE USERS.GENDER = 'M' AND MOVIES.YEAR = '2000' AND RATINGS.RATING = '5';
```

DBMS output and query output:



HOMEWORK 4
KAIJUN HE

Task completed in 0.894 seconds

```
NUMBER_FEMALE_HIGH_RATING_IN_1993
---------------------------------
                              537


NUMBER_MALE_HIGH_RATING_IN_1993
-------------------------------
                             1461


NUMBER_FEMALE_HIGH_RATING_IN_2000
---------------------------------
                              434


NUMBER_MALE_HIGH_RATING_IN_2000
-------------------------------
                             1326
```