Database Management Systems Lab
Lab 9
CSE 4308

A Z Hasnain Kabir
200042102
Software Engineering

10 November 2022

Task 1

Analysis of the problem

We are to write PL/SQL statements to execute given queries

Code

```
SET SERVEROUPUT ON
BEGIN
DBMS_OUTPUT . PUT_LINE ( ' HASNAIN KABIR ');
END;
/

DECLARE
    STUDENTID NUMBER(9);
BEGIN
    STUDENTID := '&studentid';
    DBMS_OUTPUT.PUT_LINE(STUDENTID);
END;
/

DECLARE
    NUM1 NUMBER(4);
    NUM2 NUMBER(4);
```

```
18 BEGIN
19     NUM1 := &num1;
20     NUM2 := &num2;
21     DBMS_OUTPUT.PUT_LINE(NUM1 * NUM2);
22 END;
23 /
24
25 DECLARE
26     CURRENT DATE := SYSDATE;
27 BEGIN
28     DBMS_OUTPUT.PUT_LINE(TO_CHAR(CURRENT, 'HH12:MI:SS'));
29 END;
30 /
31
32 DECLARE
33     NUM NUMERIC(5,2);
34
35 BEGIN
36     NUM := &NUM;
37     CASE NUM - TRUNC(NUM)
38         WHEN 0 THEN
39             DBMS_OUTPUT.PUT_LINE('WHOLE NUMBER');
40         ELSE
41             DBMS_OUTPUT.PUT_LINE('FRACTION');
42     END CASE;
43 END;
44 /
45
46 DECLARE
47     NUM NUMERIC(5,2);
48
49 BEGIN
50     NUM := &NUM;
51         IF (NUM-TRUNC(NUM) = 0) THEN
52             DBMS_OUTPUT.PUT_LINE('WHOLE NUMBER');
53         ELSE
54             DBMS_OUTPUT.PUT_LINE('FRACTION');
55     END IF;
56 END;
57 /
58
59 CREATE OR REPLACE PROCEDURE FIND_COMPOSITE(NUM IN NUMBER,
      RESULT OUT VARCHAR2)
```

```
AS
I NUMBER;
TEMP NUMBER;
BEGIN
    I := 2;
    TEMP := 1;

    FOR I IN 2..NUM/2
    LOOP
        IF(MOD(NUM, I) = 0) THEN
            TEMP :=0;
            EXIT;
        END IF;
    END LOOP;

    IF TEMP = 1
    THEN
        RESULT := 'PRIME';
    ELSE
        RESULT := 'COMPOSITE';
    END IF;
END;
/

DECLARE
    NUM NUMBER(3);
    RESULT VARCHAR2(10);
BEGIN
    NUM:=&NUM;
    FIND_COMPOSITE(NUM, RESULT);
    DBMS_OUTPUT.PUT_LINE(RESULT);
END;
/
```

Explanation of the solution

The solution here uses PL/SQL blocks such as BEGIN, END, DBMS_OUTPUT, Procedures and functions to execute sql queries.

Findings

It is very difficult to identify errors while executing PL/SQL blocks through command line. We can use show error to identify error.

Task 2

Analysis of the problem

We are to use procedures and functions to solve given problems.

Code

```
1  CREATE
2  OR REPLACE PROCEDURE TOP_MOVIES(NUM IN NUMBER) AS MOV_NUM
       NUMBER;
3  BEGIN
4  SELECT
5    COUNT(*) INTO MOV_NUM
6  FROM
7    MOVIE;
8  IF NUM > MOV_NUM THEN DBMS_OUTPUT.PUT_LINE('ERROR');
9  ELSE FOR ROW IN (
10   SELECT
11     *
12   FROM
13     (
14       SELECT
15         MOVIE.MOV_ID,
16         MOVIE.MOV_TITLE,
17         MOVIE.MOV_YEAR,
18         MOVIE.MOV_LANGUAGE,
19         MOVIE.MOV_RELEASEDATE,
20         MOVIE.MOV_COUNTRY,
21         SUM(RATING.REV_STARS)/ COUNT(RATING.REV_STARS) AS
               AVG_RATING
22       FROM
23         RATING,
24         MOVIE
25       WHERE
26         RATING.MOV_ID = MOVIE.MOV_ID
27       GROUP BY
```

```
28          MOVIE.MOV_ID,
29          MOVIE.MOV_TITLE,
30          MOVIE.MOV_YEAR,
31          MOVIE.MOV_LANGUAGE,
32          MOVIE.MOV_RELEASEDATE,
33          MOVIE.MOV_COUNTRY
34        ORDER BY
35          AVG_RATING DESC
36      )
37    WHERE
38      ROWNUM <= NUM
39  ) LOOP DBMS_OUTPUT.PUT_LINE(
40    ROW.MOV_ID || ' ' || ROW.MOV_TITLE || ' ' || ROW.MOV_YEAR
          || ' ' || ROW.MOV_LANGUAGE || ' ' || ROW.
        MOV_RELEASEDATE || ' ' || ROW.MOV_COUNTRY || ' ' ||
        ROW.AVG_RATING
41  );
42  END LOOP;
43  END IF;
44  END;
45  /
46
47  BEGIN TOP_RATED_MOVIES(10);
48  END;
49  /
50
51  CREATE
52  OR REPLACE FUNCTION MOVIE_STAT (TITLE MOVIE.MOV_TITLE %
      TYPE) RETURN VARCHAR AS ACT_NUM NUMBER;
53  BEGIN
54  SELECT
55    COUNT(*) INTO ACT_NUM
56  FROM
57    MOVIE,
58    CASTS,
59    ACTOR
60  WHERE
61    MOVIE.MOV_ID = CASTS.MOV_ID
62    AND CASTS.ACT_ID = ACTOR.ACT_ID
63    AND MOVIE.MOV_TITLE = TITLE
64  GROUP BY
65    MOVIE.MOV_ID;
66  IF ACT_NUM > 1 THEN RETURN 'ENSEMBLE';
```

```
67 ELSE RETURN 'SOLO';
68 END IF;
69 END;
70 /
71
72
73 BEGIN DBMS_OUTPUT.PUT_LINE(
74   MOVIE_STATUS('The Prestige')
75 );
76 END;
77 /
78
79
80 CREATE
81 OR REPLACE PROCEDURE OSCAR_NOMINATIONS AS BEGIN FOR ROW IN
       (
82   SELECT
83     DIRECTOR.DIR_FIRSTNAME,
84     DIRECTOR.DIR_LASTNAME,
85     RATED_MOVIES.MOV_ID
86   FROM
87     (
88       SELECT
89         MOVIE.MOV_ID
90       FROM
91         RATING,
92         MOVIE
93       WHERE
94         RATING.MOV_ID = MOVIE.MOV_ID
95       GROUP BY
96         MOVIE.MOV_ID,
97         MOVIE.MOV_TITLE
98       HAVING
99         SUM(RATING.REV_STARS)/ COUNT(RATING.REV_ID) >= 7
100        AND COUNT(*) >= 10
101    ) RATED_MOVIES,
102    DIRECTION,
103    DIRECTOR
104  WHERE
105    RATED_MOVIES.MOV_ID = DIRECTION.MOV_ID
106    AND DIRECTION.DIR_ID = DIRECTOR.DIR_ID
107 ) LOOP DBMS_OUTPUT.PUT_LINE(
108   ROW.DIR_FIRSTNAME || ' ' || ROW.DIR_LASTNAME
```

```
109 );
110 END LOOP;
111 END;
112 /
113
114 BEGIN OSCAR_NOMINATIONS;
115 END;
116 /
117
118
119 CREATE
120 OR REPLACE FUNCTION MOVIE_CATEGORY(TITLE MOVIE.MOV_TITLE %
       TYPE) RETURN VARCHAR AS RELEASEDATE DATE;
121 AVG_RATING RATING.REV_STARS % TYPE;
122 YEAR VARCHAR(5);
123 BEGIN
124 SELECT
125   MOVIE.MOV_RELEASEDATE INTO RELEASEDATE
126 FROM
127   MOVIE
128 WHERE
129   MOVIE.MOV_TITLE = TITLE;
130 SELECT
131   SUM(RATING.REV_STARS)/ COUNT(*) INTO AVG_RATING
132 FROM
133   MOVIE,
134   RATING
135 WHERE
136   MOVIE.MOV_ID = RATING.MOV_ID
137   AND MOVIE.MOV_TITLE = TITLE;
138 YEAR := TO_CHAR(RELEASEDATE, 'YYYY');
139 IF YEAR >= 1950
140 AND YEAR < 1960
141 AND AVG_RATING > 6.5 THEN RETURN 'Fantastic Fifties';
142 ELSIF YEAR >= 1960
143 AND YEAR < 1970
144 AND AVG_RATING > 6.7 THEN RETURN 'Sweet Sixties';
145 ELSIF YEAR >= 1970
146 AND YEAR < 1980
147 AND AVG_RATING > 6.9 THEN RETURN 'Super Seventies';
148 ELSIF YEAR >= 1980
149 AND YEAR < 1990
150 AND AVG_RATING > 7.1 THEN RETURN 'Ecstatic Eighties';
```

```
151 ELSIF YEAR >= 1990
152 AND YEAR < 2000
153 AND AVG_RATING > 7.3 THEN RETURN 'Neat Nineties';
154 ELSE RETURN 'Garbage';
155 END IF;
156 END;
157 /
158
159 BEGIN DBMS_OUTPUT.PUT_LINE(
160   MOVIE_CATEGORY('Chinatown')
161 );
162 END;
163 /
```

Explanation of the solution

The solution here uses PL/SQL blocks such as BEGIN, END, Procedures and functions to execute sql queries.

Findings

It is very difficult to identify errors while executing PL/SQL blocks through command line. We can use show error to identify error.