

**DEPARTMENT OF COMPUTER SCIENCE
RAJAGIRI COLLEGE OF SOCIAL SCIENCES
(Autonomous)
KALAMASSERY - KOCHI - 683104**



MASTER OF COMPUTER APPLICATIONS

DBMS LAB RECORD

NAME : KHADEEJA BEEVI C N

SEMESTER : I

REGISTER NO. :



**DEPARTMENT OF COMPUTER SCIENCE
RAJAGIRI COLLEGE OF SOCIAL SCIENCES
(Autonomous)
KALAMASSERY - KOCHI - 683104**

MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

NAME : KHADEEJA BEEVI C N

SEMESTER : I

REGISTER NO. :

Certified that this is a bonafide record of work done by the student in the Software Laboratory of Rajagiri Department of Computer Science, Kalamassery.

Faculty in Charge

Dean, Computer Science

Internal Examiner

External Examiner

Place : Kalamassery

Date :

Table of Contents

Activity	Page No
1. E-R Diagram & Table Design	1
2. Practice SQL Data Definition Language(DDL) commands	3
2.1 Table creation and alteration	
3. Practice SQL Data Manipulation Language (DML) commands	10
3.1 Row insertion, deletion and updating	10
3.2 Retrieval of data (Simple select query and select with where options (include all relational and logical operators)	21
3.3 Functions: Numeric Data, Character Conversion and Group functions	27
3.4 Data manipulations using date functions	30
3.5 Set Operations	35
3.6 Illustration of Group by Having Clause	39
3.7 Sub Queries	41
3.8 Views	44
4. Practice PL/SQL	50
4.1 Introduction to PL/SQL	50
4.2 Illustration of Cursors	55
4.3 Illustration of Procedures	60
4.4 Illustration of Functions	66
4.5 Illustration of Triggers	69

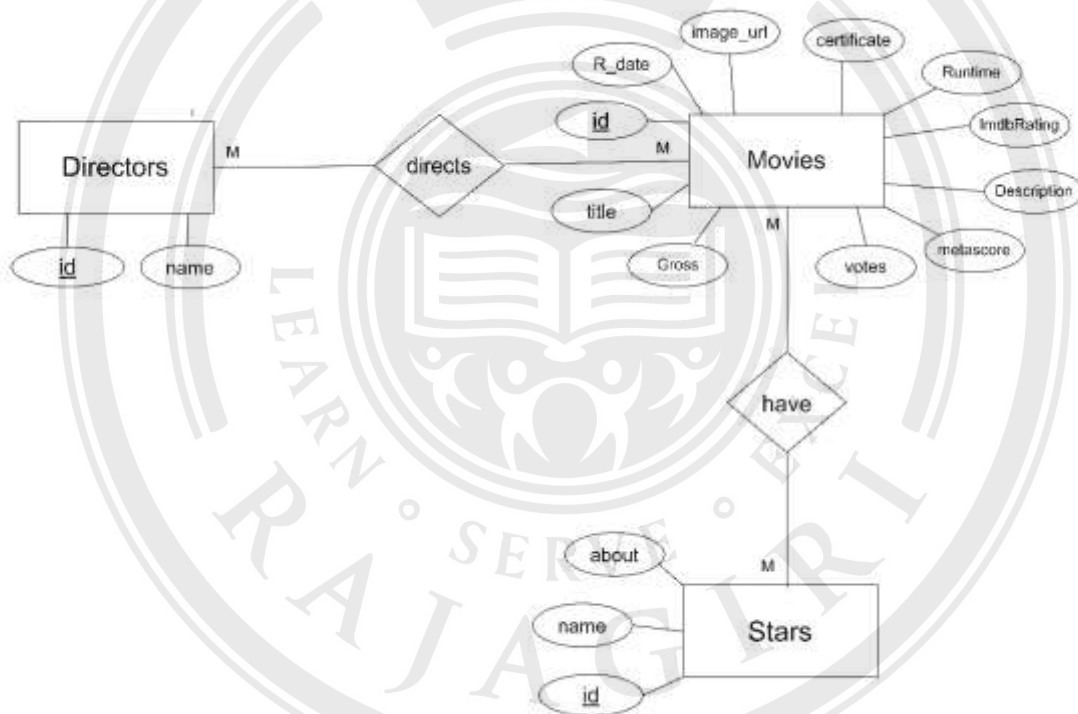
Activity # 1**1. ER Diagram & Table Design**

Description :

- E-R Diagram and table reduction
- Table descriptions

Date : 14/08/2023

- **ER DIAGRAM**

**TABLE DESIGN****Table name: Directors****Description: Used to store Directors Information**

Attribute	Data Type	Constraints
Id	Int	Primary Key/ Not Null
Name	Varchar2(40)	Not Null

Table name: Stars**Description: Used to store Stars Information**

Attribute	Data Type	Constraints
Id	Int	Primary Key/ Not Null
Name	Varchar2(40)	Unique
About	Varchar2(100)	

Table name: Movies**Description: Used to store Movies Information**

Attribute	Data Type	Constraints
Id	Int	Primary Key/ Not Null
Title	Varchar2(40)	Not Null
R_date	Date	
Image_url	Varchar2(100)	
Certificate	Varchar2(20)	
Runtime	Number(3,2)	
ImdbRating	Number (3,1)	By default 0
Description	Text(100)	By default Null
Metascore	Number (3,1)	By default 0
Votes	Int	By default 0
Gross	Number(10,2)	Gross amount should be greater than 10000

Table name: MoviesDirectors**Description: Used to store Movie Directors Information**

Attribute	Data Type	Constraints	
MoviesId	Int	Foreign Key references Id of Movies table	Primary Key
DirectorsId	Int	Foreign Key references Id of Directors table	

Table name: MoviesStars**Description: Used to store Movie Stars Information**

Attribute	Data Type	Constraints	
MoviesId	Int	Foreign Key references Id of Movies table	Primary Key
StarsId	Int	Foreign Key references Id of Stars table	

Activity # 2**2. Practice SQL Data Definition Language(DDL) commands**

Description : Table creation and alteration

Date:14/08/2023

Query

- **Create the tables based on the above description.**

SQL> create table directors(id number(5) primary key,name varchar2(40) not null);

Table created.

SQL> create table stars(id number(5) primary key,name varchar2(40) unique,about varchar2(100));

Table created.

SQL> create table movies(id number(5),title varchar2(40) not null,r_date date,image_url varchar2(100),certificate varchar2(20),runtime number(3,2),imdbrating number(3,1) default 0,description varchar2(100) default NULL,metascore number(3,1) default 0,votes number(5) default 0,gross number(10,2),constraint pkmovie primary key(id),constraint chkgross check(gross>10000));

Table created.

SQL> create table moviesdirectors(moviesid number(5),directorsid number(5),constraint fkmov foreign key(moviesid) references movies(id),constraint fkdir foreign key(directorsid) references directors(id),constraint pkmovdir primary key(moviesid,directorsid));

Table created.

SQL> create table moviesstars(moviesid number(5),starsid number(5),constraint fksmov foreign key(moviesid) references movies(id),constraint fkstar foreign key(starsid) references stars(id),constraint pkstar primary key(moviesid,starsid));

Table created.

SQL> desc directors;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
NAME	NOT NULL	VARCHAR2(40)

SQL> desc stars;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
NAME		VARCHAR2(40)
ABOUT		VARCHAR2(100)

SQL> desc movies;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
R_DATE		DATE
IMAGE_URL		VARCHAR2(100)
CERTIFICATE		VARCHAR2(20)
RUNTIME		NUMBER(3,2)
IMDBRATING		NUMBER(3,1)
DESCRIPTION		VARCHAR2(100)
METAScore		NUMBER(3,1)
VOTES		NUMBER(5)
GROSS		NUMBER(10,2)

SQL> desc moviesdirectors;

Name	Null?	Type
MOVIESID	NOT NULL	NUMBER(5)
DIRECTORSID	NOT NULL	NUMBER(5)

SQL> desc moviesstars;

Name	Null?	Type
MOVIESID	NOT NULL	NUMBER(5)
STARSID	NOT NULL	NUMBER(5)

- **Add a column 'DOB' to Stars table**

SQL> alter table stars add dob date;

Table altered.

SQL> desc stars;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
NAME		VARCHAR2(40)
ABOUT		VARCHAR2(100)
DOB		DATE

- **Drop the column 'Gross' in Movies table**

SQL> alter table movies drop column gross;

Table altered.

SQL> desc movies;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
R_DATE		DATE
IMAGE_URL		VARCHAR2(100)
CERTIFICATE		VARCHAR2(20)
RUNTIME		NUMBER(3,2)
IMDBRATING		NUMBER(3,1)
DESCRIPTION		VARCHAR2(100)
METAScore		NUMBER(3,1)
VOTES		NUMBER(5)

- **Add column 'Language' in Movies table.**

SQL> alter table movies add language varchar2(40);

Table altered.

SQL> desc movies;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
R_DATE		DATE
IMAGE_URL		VARCHAR2(100)
CERTIFICATE		VARCHAR2(20)
RUNTIME		NUMBER(3,2)
IMDBRATING		NUMBER(3,1)
DESCRIPTION		VARCHAR2(100)
METAScore		NUMBER(3,1)

VOTES	NUMBER(5)
LANGUAGE	VARCHAR2(40)

- **Add column Gross Number(10,2) in Movies table.**

SQL> alter table movies add gross number(12,2);

Table altered.

SQL> desc movies;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
R_DATE		DATE
IMAGE_URL		VARCHAR2(100)
CERTIFICATE		VARCHAR2(20)
RUNTIME		NUMBER(3,2)
IMDBRATING		NUMBER(3,1)
DESCRIPTION		VARCHAR2(100)
METAScore		NUMBER(3,1)
VOTES		NUMBER(5)
LANGUAGE		VARCHAR2(40)
GROSS		NUMBER(12,2)

- **Change the name of the column 'R_date' in Movies table to Releasedate. Releasedate.**

SQL> alter table movies rename column r_date to releasedate;

Table altered.

SQL> desc movies;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
RELEASEDATE		DATE
IMAGE_URL		VARCHAR2(100)
CERTIFICATE		VARCHAR2(20)
RUNTIME		NUMBER(3,2)
IMDBRATING		NUMBER(3,1)
DESCRIPTION		VARCHAR2(100)
METAScore		NUMBER(3,1)
VOTES		NUMBER(5)
LANGUAGE		VARCHAR2(40)
GROSS		NUMBER(12,2)

- **Add a column 'Age' in Directors table as Number. Age must be 7 years or above.**

SQL> alter table directors add age number(5) constraint chkage check(age>7);

Table altered.

SQL> desc directors;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
NAME	NOT NULL	VARCHAR2(40)
AGE		NUMBER(5)

- **Add a new column 'Hit' in Movies table with datatype Number(1) and by default 0.**

SQL> alter table movies add hit number(1) default 0;

Table altered.

SQL> desc movies;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
RELEASEDATE		DATE
IMAGE_URL		VARCHAR2(100)
CERTIFICATE		VARCHAR2(20)
RUNTIME		NUMBER(3,2)
IMDBRATING		NUMBER(3,1)
DESCRIPTION		VARCHAR2(100)
METAScore		NUMBER(3,1)
VOTES		NUMBER(5)
LANGUAGE		VARCHAR2(40)
GROSS		NUMBER(12,2)
HIT		NUMBER(1)

- **Add a new column 'Entry_date' in Movies table to record the date on which the movie details are entered in the data base.**

SQL> alter table movies add entry_date date;

Table altered.

SQL> desc movies;

Name	Null?	Type

ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
RELEASEDATE		DATE
IMAGE_URL		VARCHAR2(100)
CERTIFICATE		VARCHAR2(20)
RUNTIME		NUMBER(3,2)
IMDBRATING		NUMBER(3,1)
DESCRIPTION		VARCHAR2(100)
METAScore		NUMBER(3,1)
VOTES		NUMBER(5)
LANGUAGE		VARCHAR2(40)
GROSS		NUMBER(12,2)
HIT		NUMBER(1)
ENTRY_DATE		DATE

- **Destroy the table MoviesStars and recreate it.**

SQL> drop table moviesstars;

Table dropped.

SQL> create table moviesstars(moviesid number(5),starsid number(5),constraint fksmov foreign key(moviesid) references movies(id),constraint fkstar foreign key(starsid) references stars(id),constraint pkstar primary key(moviesid,starsid));

Table created.

- **Change the size of the Director's name to 30.**

SQL> alter table directors modify name varchar2(30);

Table altered.

SQL> desc directors;

Name	Null?	Type

ID	NOT NULL	NUMBER(5)
NAME	NOT NULL	VARCHAR2(30)
AGE		NUMBER(5)

- **Add the following check constraints:**
- **Releasedate should be less than the Entry_date in the Movies table.**

SQL> alter table movies add constraint chkdate check(releasedate<entry_date);

Table altered.

```
SQL> select constraint_name,constraint_type from user_constraints where
table_name='MOVIES';
```

CONSTRAINT_NAME	
SYS_C0011211	C
PKMOVIE	P
CHKDATE	C

- **Language of movies should be Malayalam, English, Tamil or Hindi.**

```
SQL> alter table movies add constraint chklang check(language
in('malayalam','english','tamil','hindi'));
```

Table altered.

```
SQL> select constraint_name,constraint_type from user_constraints where
table_name='MOVIES';
```

CONSTRAINT_NAME	
SYS_C0011211	C
PKMOVIE	P
CHKDATE	C
CHKLANG	C

Activity # 3**3. Practice SQL Data Manipulation Language (DML) commands**

Description: Row insertion, deletion and updating

Date: 27/08/2023

Query

- **Insert the appropriate data (10 rows) for the tables with respect to defined datatypes, size and constraints.**

SQL> insert into directors values(101,'Nahas Hidayath',45);

1 row created.

SQL> insert into directors values(102,'Alphonse Puthran',50);

1 row created.

SQL> insert into directors values(103,'Anwar Rasheed',40);

1 row created.

SQL> insert into directors values(104,'Rojin Thomas',35);

1 row created.

SQL> insert into directors values(105,'James Cameroon',35);

1 row created.

SQL> insert into directors values(106,'Quentin Tarantino',55);

1 row created.

SQL> insert into directors values(107,'Mohit Suri',48);

1 row created.

SQL> insert into directors values(108,'Rohit Shetty',51);

1 row created.

SQL> insert into directors values(109,'Gautham Vasudev Menon',53);

1 row created.

SQL> insert into directors values(110,'Atlee',37);

1 row created.

SQL> select * from directors;

ID NAME	AGE
101 Nahas Hidayath	45
102 Alphonse Puthran	50
103 Anwar Rasheed	40
104 Rojin Thomas	35
105 James Cameroon	35
106 Quentin Tarantino	55
107 Mohit Suri	48
108 Rohit Shetty	51
109 Gautham Vasudev Menon	53
110 Atlee	37

10 rows selected.

SQL> insert into stars values(201,'Shane Nigam','Good Acting','12-feb-1990');

1 row created.

SQL> insert into stars values(202,'Nivin Pauly','Super performance','20-apr-1993');

1 row created.

SQL> insert into stars values(203,'Dulquer Salman','Fantastic','24-jun-1995');

1 row created.

SQL> insert into stars values(204,'Indrans','Great characterization','04-may-1980');

1 row created.

SQL> insert into stars values(205,'Sully','Thriller','16-oct-1986');

1 row created.

SQL> insert into stars values(206,'Leonardo Dicaprio','Excellent','31-aug-1978');

1 row created.

SQL> insert into stars values(207,'Aditya Roy kapur','Amazing','10-sep-1983');

1 row created.

SQL> insert into stars values(208,'Shah Rukh Khan','Super Acting','02-nov-1987');

1 row created.

SQL> insert into stars values(209,'Surya','Best actor','21-dec-1981');

1 row created.

SQL> insert into stars values(210,'Vijay','Famous Actor','18-jan-1979');

1 row created.

SQL> select * from stars;

ID NAME DOB	ABOUT
201 Shane Nigam 12-FEB-90	Good Acting
202 Nivin Pauly 20-APR-93	Super performance
203 Dulquer Salman 24-JUN-95	Fantastic
204 Indrans 04-MAY-80	Great characterization
205 Sully 16-OCT-86	Thriller
206 Leonardo Dicaprio 31-AUG-78	Excellent
207 Aditya Roy kapur 10-SEP-83	Amazing
208 Shah Rukh Khan 02-NOV-87	Super Acting
209 Surya 21-DEC-81	Best actor
210 Vijay 18-JAN-79	Famous Actor

10 rows selected.

SQL> insert into movies values(301,'Rdx','25-aug-2023','Rdx.png','U/A',2.5,5.9,'trio and the fight scenes',61,443,'malayalam',14000000,1,'27-aug-2023');

1 row created.

```
SQL> insert into movies values(302,'premam','29-may-2015','premam.png','U/A',2.5,8.3,'Romantic journey',74,21991,'malayalam',76000000,2,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(303,'Usthad Hotel','29-jun-2012','usthadhotel.jpg','U/A',2,9,'Aspiring chef',0,452,'malayalam',41000000,1,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(304,'Home','19-aug-2021','home.jpg','U/A',2.8,9.1,'Life of a family',67,95,'malayalam',3000000,0,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(305,'Avatar','16-dec-2022','avatar.jpg','A',3.2,7.6,'NULL',67,100,'english',230000000,3,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(306,'Once upon a time in Hollywood','26-jul-2019','once_upon_a_time_in.jpg','A',2.8,7.6,'hope of acting career',62,92,'english',37000000,2,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(307,'Ashiqui 2','26-apr-2013','ashiqui2.png','U',2.2,7,'singer falls in love',74,84,'hindi',109000000,1,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(308,'Chennai express','08-aug-2013','chennaiexpress.jpg','U/A',2.4,6,'Train story',45,120,'hindi',109000000,2,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(309,'Vaaranam Aayiram','14-nov-2008','vaaranam.jpg','U/A',2.8,8.2,'Rescue mission and fathers death',0,49,'tamil',40000000,0,'27-aug-2023');
```

1 row created.

```
SQL> insert into movies values(310,'Mersal','18-oct-2017','mersal.jpg','U/A',2.8,7.5,'corruption and arresting doctor',0,300,'tamil',2600000000,0,'27-aug-2023');
```


SQL> select * from movies;

ID	TITLE	RELEASEDA
301	Rdx	25-AUG-23
302	premam	09-MAY-15
303	Usthad Hotel	29-JUN-12
304	Home	19-AUG-21
305	Avatar	16-DEC-22
306	Once upon a time in Hollywood	26-JUL-19
307	Ashiqui 2	26-APR-13
308	Chennai express	08-AUG-13
309	Vaaranam Aayiram	14-NOV-08
310	Mersal	18-OCT-17

IMAGE_URL	CERTIFICATE
Rdx.png	U/A
premam.png	U/A
usthadhotel.jp	U/A
home.jpg	U/A
avatar.jpg	A
once_upon_a_time_in.jpg	A
ashiqui2.png	U
chennaiexpress.jpg	U/A
vaaranam.jpg	U/A
mersal.jpg	U/A

RUNTIME	IMDBRATING
2.5	5.9
2.5	8.3
2	9
2.8	9.1
2.8	9.1
2.8	7.6
2.2	7
2.4	6
2.8	8.2
2.8	7.5

DESCRIPTION

trio and the fight scenes
 Romantic journey
 Aspiring chef
 Life of a family
 NULL
 hope of acting career
 singer falls in love
 Train story
 Rescue mission and fathers death

corruption and arresting doctor

METAScore	VOTES	LANGUAGE
61	443	malayalam
74	21991	malayalam
0	452	malayalam
67	95	malayalam
67	100	english
62	92	english
74	84	hindi
45	120	hindi
0	49	tamil
0	300	tamil

GROSS	HIT ENTRY_DAT
14000000	1 27-AUG-23
76000000	1 27-AUG-23
41000000	1 27-AUG-23
730000000	1 27-AUG-23
109500000	0 27-AUG-23
3000000	1 27-AUG-23
230000000	1 27-AUG-23
37000000	1 27-AUG-23
109000000	1 27-AUG-23
40000000	0 27-AUG-23
2600000000	1 27-AUG-23

10 rows selected.

1 row created.

SQL> insert into moviesdirectors values(301,101);

1 row created.

SQL> insert into moviesdirectors values(302,102);

1 row created.

SQL> insert into moviesdirectors values(303,103);

1 row created.

SQL> insert into moviesdirectors values(304,104);

1 row created.

SQL> insert into moviesdirectors values(305,105);

1 row created.

SQL> insert into moviesdirectors values(306,106);

1 row created.

SQL> insert into moviesdirectors values(307,107);

1 row created.

SQL> insert into moviesdirectors values(308,108);

1 row created.

SQL> insert into moviesdirectors values(309,109);

1 row created.

SQL> insert into moviesdirectors values(310,110);

1 row created.

SQL> select * from moviesdirectors;

MOVIESID	DIRECTORSID
301	101
302	102
303	103
304	104
305	105
306	106
307	107
308	108
309	109
310	110

10 rows selected.

SQL> insert into moviesstars values(301,201);

1 row created.

SQL> insert into moviesstars values(302,202);

1 row created.

SQL> insert into moviesstars values(303,203);

1 row created.

SQL> insert into moviesstars values(304,204);

1 row created.

SQL> insert into moviesstars values(305,205);

1 row created.

SQL> insert into moviesstars values(306,206);

1 row created.

SQL> insert into moviesstars values(307,207);

1 row created.

SQL> insert into moviesstars values(308,208);

1 row created.

SQL> insert into moviesstars values(309,209);

1 row created.

SQL> insert into moviesstars values(310,210);

1 row created.

SQL> select * from moviesstars;

MOVIESID	STARSID
301	201
302	202
303	203
304	204
305	205
306	206
307	207
308	208
309	209
310	210

10 rows selected.

- **Change value of Hit to 1 where 'Votes' greater than or equal to 90.**

SQL> update movies set hit=1 where votes>=90;

8 rows updated.

- **Create table IndustryHit with the following columns:**

Id

Title

Releasedate

Language

Votes

Gross

The data types and null characteristics for these columns should be the same as the corresponding columns in the Movies table described at the beginning of the lab exercise.

```
SQL> create table industryhit(id number(5) primary key,title varchar2(40) not null,releasedate date,language varchar2(40),votes number(5),gross number(12,2));
```

Table created.

```
SQL> desc industryhit;
```

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
RELEASEDATE		
DATE		
LANGUAGE		VARCHAR2(40)
VOTES		NUMBER(5)
GROSS		NUMBER(12,2)

- **New movies hit the box office; their data is as follows:**

Id: 1014, 1021, 1032

Title: 2018: Everyone is a Hero, Oppenheimer, Maamannan

Releasedate: 5 May 2023, 21 July 2023, 29 June 2023

Language: Malayalam, English, Tamil

Votes: 97, 96, 95

Gross: 750000000, 500000000, 505000000

Add the new employees to the IndustryHit table.

```
SQL> insert into industryhit values(1014,'Everyone is a Hero','05-may-2023','malayalam',97,750000000);
```

1 row created.

```
SQL> insert into industryhit values(1021,'Oppenheimer','21-jul-2023','english',96,500000000);
```

1 row created.

```
SQL> insert into industryhit values(1032,'Maamannan','29-jun-2023','hindi',95,505000000);
```

1 row created.

```
SQL> select * from industryhit;
```

ID	TITLE	RELEASEDATE	LANGUAGE	VOTES	GROSS
1014	Everyone is a Hero	05-MAY-23	malayalam	97	750000000
1021	Oppenheimer	21-JUL-23	english	96	50000000
1032	Maamannan	29-JUN-23	hindi	95	50500000

- Insert data into the new IndustryHit table.**

```
SQL> insert into industryhit(select id,title,release date,language,votes,gross from movies where votes>=95);
```

7 rows created.

```
SQL> select * from industryhit;
```

ID	TITLE	RELEASEDATE	LANGUAGE	VOTES	GROSS
1014	Everyone is a Hero	05-MAY-23	malayalam	97	750000000
1021	Oppenheimer	21-JUL-23	english	96	500000000
1032	Maamannan	29-JUN-23	hindi	95	505000000
301	Rdx	25-AUG-23	malayalam	443	14000000
302	premam	29-MAY-15	malayalam	21991	76000000
303	Usthad Hotel	29-JUN-12	malayalam	452	41000000
304	Home	19-AUG-21	malayalam	95	3000000
305	Avatar	16-DEC-22	english	100	230000000
308	Chennai express	08-AUG-13	hindi	120	109000000
310	Mersal	18-OCT-17	tamil	300	2600000000

10 rows selected.

- Insert data into the IndustryHit table by copying the appropriate columns in the Movies table for those Movies that have Votes greater than or equal to 95.**

```
SQL> insert into industryhit(select id,title,release date,language,votes,gross from movies where votes>=95);
```

7 rows created.

SQL> select * from industryhit;

ID	TITLE	RELEASEDA	LANGUAGE	VOTES	GROSS
1014	Everyone is a Hero	05-MAY-23	malayalam	97	750000000
1021	Oppenheimer	21-JUL-23	english	96	500000000
1032	Maamannan	29-JUN-23	hindi	95	505000000
301	Rdx	25-AUG-23	malayalam	443	14000000
302	premam	29-MAY-15	malayalam	21991	76000000
303	Usthad Hotel	29-JUN-12	malayalam	452	41000000
304	Home	19-AUG-21	malayalam	95	3000000
305	Avatar	16-DEC-22	english	100	230000000
308	Chennai express	08-AUG-13	hindi	120	109000000
310	Mersal	18-OCT-17	tamil	300	2600000000

10 rows selected.

- **Movie Oppenheimer got a Metascore of 80. Make the appropriate data change..**

SQL> update movies set metascore=80 where title='Oppenheimer';

1 row updated

- **Movie ‘Voice Of Sathyanathan’ was released. For ‘Voice Of Sathyanathan’ enter the following data:**

Id: 1015

Title: Voice Of Sathyanathan

Releasedate: 28 July 2023

Image_url: https://m.media-amazon.com/imak2M_.jpg

Certificate: U

Runtime: 2.10

ImdbRating: 7.4

Description: A man's life becomes increasingly complicated after his neighbor is injured in a dispute over a fence.

Metascore: 60

Votes: 90

Gross: 109500000

SQL> insert into

movies(id,title,releasedate,image_url,certificate,runtime,imdbrating,description,metascore,votes,gross) values(1015,'Voice of Sathyanathan','28-jul-2023','https://m.media-amazon.com/imake2M_.jpg','U',2.10,7.4,'A Mans life becomes increasingly complicated after his neighbor is injured in a dispute over a fence',60,90,109500000);

1 row created.

SQL> select * from movies where id=1015;

ID	TITLE	RELEASEDA	IMAGE_URL	CERTIFICATE	RUNTIME	IMDBRATING
DESCRIPTION	METAScore	VOTES	LANGUAGE	GROSS	HIT	
1015	Voice of Sathyanathan	28-JUL-23	https://m.media-amazon.com/ima2M_.jpg	U	2.17.4	A Mans life becomes increasingly complicated after his neighbor is injured in a dispute over a fence
	60	90	109500000	0		

- **Delete all rows from IndustryHit and drop the IndustryHit table.**

SQL> delete from industryhit;

10 rows deleted.

SQL> drop table industryhit;

Table dropped.

Description: Retrieval of data (Simple select query and select with 'where' options (include all relational and logical operators))

Date:28/08/2023

Query

- **List details of all movies**

SQL> select * from movies;

ID	TITLE	RELEASEDA
301	Rdx	25-AUG-23
302	premam	09-MAY-15
303	Usthad Hotel	29-JUN-12
311	Oppenheimer	21-JUL-23
1015	Voice of Sathyanathan	28-JUL-23
304	Home	19-AUG-21
305	Avatar	16-DEC-22
306	Once upon a time in Hollywood	26-JUL-19
307	Ashiqui 2	26-APR-13
308	Chennai express	08-AUG-13
309	Vaaranam Aayiram	14-NOV-08
310	Mersal	18-OCT-17

IMAGE_URL	CERTIFICATE
Rdx.png	U/A
premam.png	U/A
usthadhotel.jp	U/A
oppenheimer.jpg	U/A
https://m.media-amazon.com/imake2M_.jpg	U/A
home.jpg	U/A
avatar.jpg	A
once_upon_a_time_in.jpg	A
ashiqui2.png	U
chennaiaexpress.jpg	U/A
vaaranam.jpg	U/A
mersal.jpg	U/A

RUNTIME	IMDBRATING
2.5	5.9
2.5	8.3
2	9

3	8.6
2.1	7.4
2.8	9.1
2.8	9.1
2.8	7.6
2.2	7
2.4	6
2.8	8.2
2.8	7.5

DESCRIPTION

trio and the fight scenes
 Romantic journey
 Aspiring chef
 Devolpment of atomic bomb
 A Mans life becomes increasingly complicated after his neighbor is injured in a dispute over a fence
 Life of a family
 NULL
 hope of acting career
 singer falls in love
 Train story
 Rescue mission and fathers death
 corruption and arresting doctor

METAScore	VOTES	LANGUAGE
61	443	malayalam
74	21991	malayalam
0	452	malayalam
80	400	english
60	90	malayalam
67	95	malayalam
67	100	english
62	92	english
74	84	hindi
45	120	hindi
1	49	tamil
0	300	tamil

GROSS	HIT ENTRY_DAT
14000000	1 27-AUG-23
76000000	1 27-AUG-23
41000000	1 27-AUG-23
730000000	1 27-AUG-23
109500000	0 27-AUG-23
3000000	1 27-AUG-23
230000000	1 27-AUG-23

37000000	1 27-AUG-23
109000000	1 27-AUG-23
40000000	0 27-AUG-23
2600000000	1 27-AUG-23

12 rows selected.

- **List Title, Votes, Releasedate, Gross where Gross collection greater than 5000,000,00. Sequence the results in descending order by Gross.**

SQL> select title,votes,releasedate,gross from movies where gross>5000000000 order by gross desc;

TITLE	VOTES RELEASEDA	GROSS
Mersal	300 18-OCT-17	2600000000
Oppenheimer	400 21-JUL-23	7300000000

- **Retrieve the titles and years of Tamil movies released in 2022.**

SQL> select title,extract(year from releasedate)as year from movies where language='tamil' and extract(year from releasedate)='2022';

TITLE	YEAR
Dejavu	2022
Diary	2022

- **Get the titles, years, and meta scores of movies sorted in descending order of meta scores.**

SQL> select title,extract(year from releasedate)as year,metascore from movies order by metascore desc;

TITLE	YEAR	METAScore
Oppenheimer	2023	80
Ashiqui 2	2013	74
premam	2015	74
Diary	2022	67
Avatar	2022	67
Home	2021	67
Once upon a time in Hollywood	2019	62
Rdx	2023	61
Voice of Sathyanathan	2023	60
Chennai express	2013	45
Mersal	2017	0

TITLE	YEAR	METAScore
Vaaranam Aayiram	2008	0
Dejavu	2022	0
Usthad Hotel	2012	0

- **List titles, years, languages, dates and votes of all Malayalam and English movies released before 2022 and ImdbRating less than 7. The list should be ordered by Title.**

SQL> select title,extract(year from releasedate)as year,language,releasedate,votes from movies where (language='malayalam' or language='english') and extract(year from releasedate)<2022 and imdbrating<7 order by title;

TITLE	YEAR	LANGUAGE	RELEASEDATE	VOTES
Once upon a time in Hollywood	2019	english	26-JUL-19	92
premam	2015	malayalam	29-MAY-15	21991

- **List all the movies whose title starts with 'Open'. Order the result by descending order of their id.**

SQL> select id,title from movies where title like 'Open%' order by id desc;

ID	TITLE
315	Open grave
314	Open the door please

- **List Hit movies released in 2022 and 2023. Order the result by ascending order of their Titles.**

SQL> select title,extract(year from releasedate)as year from movies where extract(year from releasedate) between 2022 and 2023 order by title;

TITLE	YEAR
Avatar	2022
Dejavu	2022
Diary	2022
Oppenheimer	2023
Rdx	2023
Voice of Sathyanathan	2023

6 rows selected.

- **Retrieve movies with a runtime between 1.5 and 2.5 hours.**

SQL> select title ,runtime from movies where runtime between 1.5 and 2.5;

TITLE	RUNTIME
Rdx	2.5
premam	2.5
Usthad Hotel	2
Voice of Sathyanathan	2.1
Dejavu	1.9
Diary	2
Open grave	1.8
Ashiqui 2	2.2
Chennai express	2.4

9 rows selected.

- **Retrieve movies with Metascore ratings below 50 and IMDb ratings above 6.0.**

SQL> select title,metascore,imdbrating from movies where metascore<50 and imdbrating>6.0;

TITLE	METAScore	IMDBRATING
Usthad Hotel	0	9
Dejavu	0	6.7
Open the door please	0	6.6
Open grave	33	6.2
Vaaranam Aayiram	0	8.2
Mersal	0	7.5

6 rows selected.

- **Retrieve movies with no description provided.**

SQL> select title,description from movies where description='NULL';

TITLE	DESCRIPTION
Avatar	NULL

Description: Functions: Numeric Data, Character Conversion and Group functions

Date:28/08/2023

Query

- Illustrate the different numeric functions using dual table (power, round, ceil, floor, abs, exp, greatest, least, mod, trunc, round, sign, sqrt etc.)

SQL> select power(4,2),round(10.3445,2),ceil(24.7),floor(24.7),abs(-20) from dual;

POWER(4,2) ROUND(10.3445,2) CEIL(24.7) FLOOR(24.7) ABS(-20)

16 10.34 25 24 20

SQL> select exp(2),greatest(7,-3,9),least(-7,3,9),mod(15,7) from dual;

EXP(2) GREATEST(7,-3,9) LEAST(-7,3,9) MOD(15,7)

7.3890561 9 -7 1

SQL> select trunc(15.6321,3),sign(3),sqrt(25) from dual;

TRUNC(15.6321,3) SIGN(3) SQRT(25)

15.632 1 5

- Illustrate the character functions (upper, lower, initcap, length, concat, ascii, substr, ltrim, rtrim, trim, translate, instr, chr,Lpad,Rpadetc) using the table Movies.

SQL> select upper('chennai express')as upper, lower('CHENNAI EXPRESS')as lower,initcap('chennai express')as initcap,length('chennai express')as length,concat('chennaiexpress','movie')as concat,ascii('chennai express')as ascii from movies where id=308;

UPPER	LOWER	INITCAP	LENGTH	CONCAT	ASCII
CHENNAI EXPRESS	chennai express	Chennai Express	15	chennaiexpressmovie	99

```
SQL> select substr('chennaiexpress',3,6)as substr,ltrim(' chennaiexpress ')as ltrim,rtrim('
chennaiexpress ')as rtrim,trim(' chennaiexpress ')as
trim,translate('chennaiexpress','nna','wow')as translate from movies where id=308;
```

SUBSTR	LTRIM	RTRIM	TRIM	TRANSLATE
ennaie	chennaiexpress	chennaiexpress	chennaiexpress	chewwwiexpress

```
SQL> select instr('chennaiexpress','n')as instr,chr(97),lpad('chennaiexpress',20,'*')as
lpad,RPAD('chennaiexpress',20,'*')as rpad from movies where id=308;
```

INSTR C	LPAD	RPAD
4 a	*****chennaiexpress	chennaiexpress*****

- **Illustration of conversion functions- to_number, to_char(numberconversion), to_char(dateconversion)**

```
SQL> select to_number('1342.67', '9999.99')as to_number,to_char(1342.67,'9999.9')as
to_char ,to_char(sysdate,'dd-mm-yyyy') as to_char from dual;
```

TO_NUMBER	TO_CHAR	TO_CHAR
1342.67	1342.7	28-08-2023

- **Count the total no. of Movies**

```
SQL> select count(id) as Totalno from movies;
```

TOTALNO
16

- **Calculate the average votes of movies.**

```
SQL> select avg(votes)from movies;
```

AVG(VOTES)
1544.125

- **Determine the maximum and minimum collection of movies. Rename the output as Max_Coll and Min_Coll respectively.**

```
SQL> select max(gross)as maxcoll,min(gross)as mincoll from movies;
```

MAXCOLL	MINCOLL
---------	---------

```
-----
2600000000  100000
```

- **Count the number of movies crossed the collection 50,00,00,000.**

SQL> select count(id) as no_of_movies from movies where gross>500000000;

NO_OF_MOVIES

```
-----
2
```

- **Count the hit movies of 2021.**

SQL> select count(id) as no_of__movies from movies where hit=1 and extract(year from releasedate)=2021;

NO_OF__MOVIES

```
-----
1
```

Description: Functions: Data manipulations using date functions

Date:28/08/2023

Query

- **Provide a list of all movies which were released on June 16, 2020. Display the year and month of the released date and the Id. Sort the result by Id. Name the derived columns YEAR and MONTH.**

SQL> select id,extract(year from releasedate)year,to_char(releasedate,'month')as month
from movies where releasedate='16-jun-2020' order by id;

ID	YEAR	MONTH
316	2020	june
317	2020	june

- **List the number of months between release date and entry date of each movie.**

SQL> select abs(ceil(months_between(releasedate,entry_date))) as no_of_months from
movies;

NO_OF_MONTHS
0
98
133
1
38
13
12
193
120

NO_OF_MONTHS
38
24
8

49
124
120
177
70

18 rows selected.

- **List the Entry_date in the format 'DD-Month-YY'.**

SQL> select to_char(entry_date,'DD-month-YY') as entry_date from movies;

ENTRY_DATE

27-august -23
27-august -23
27-august -23
27-august -23

28-august -23
28-august -23
28-august -23
28-august -23
28-august -23
28-august -23

ENTRY_DATE

27-august -23
27-august -23
27-august -23
27-august -23
27-august -23
27-august -23
27-august -23

18 rows selected

- **List the date, 8 days after today's date.**

SQL> select sysdate+8 from dual;

SYSDATE+8

05-SEP-23

- **List all the movies which were released in the month of February.**

SQL> select title,releasedate from movies where to_char(releasedate,'mm')='02';

TITLE	RELEASEDA
Santhosham	24-FEB-23

- **Illustrate the different date functions using dual table (to_date, Add_months, last_day, months_between, next_day, round etc.)**

SQL> select sysdate,to_date('27/05/2002','dd-mm-yy') from dual;

SYSDATE	TO_DATE('
28-AUG-23	27-MAY-02

SQL> select sysdate,add_months(sysdate,4) from dual;

SYSDATE	ADD_MONTH
28-AUG-23	28-DEC-23

SQL> select last_day('05/may/02') from dual;

LAST_DAY('
31-MAY-02

SQL> select abs(ceil(months_between('22-sep-17','25-jul-18'))) as months_between from dual;

MONTHS_BETWEEN
10

SQL> select sysdate,next_day(sysdate,'monday') from dual;

SYSDATE	NEXT_DAY('
28-AUG-23	04-SEP-23

SQL> select round(to_date('28-sep-2024'),'month')from dual;

ROUND(TO_
01-OCT-24

- **Illustration of special date formats using to_char function (use of th,sp,sph)**

SQL> select to_char(releasedate,'ddth-mon-yyyy') as th_function from movies ;

TH_FUNCTION

 25th-aug-2023
 29th-may-2015
 29th-jun-2012
 21st-jul-2023
 28th-jul-2023
 16th-jun-2020
 22nd-jul-2022
 24th-feb-2023
 26th-aug-2022
 04th-jul-2007
 14th-aug-2013

TH_FUNCTION

 16th-jun-2020
 19th-aug-2021
 16th-dec-2022
 26th-jul-2019
 26th-apr-2013
 08th-aug-2013
 14th-nov-2008
 18th-oct-2017

19 rows selected.

SQL> select to_char(releasedate,'ddsp-mm-yyyy') as sp_function from movies ;

SP_FUNCTION

 twenty-five-08-2023
 twenty-nine-05-2015
 twenty-nine-06-2012
 twenty-one-07-2023
 twenty-eight-07-2023
 sixteen-06-2020
 twenty-two-07-2022
 twenty-four-02-2023
 twenty-six-08-2022
 four-07-2007
 fourteen-08-2013

SP_FUNCTION

```

-----
sixteen-06-2020
nineteen-08-2021
sixteen-12-2022
twenty-six-07-2019
twenty-six-04-2013
eight-08-2013
fourteen-11-2008
eighteen-10-2017

```

19 rows selected.

SQL> select to_char(releasedate,'ddspth-mm-yyyy') as spth_function from movies ;

SPTH_FUNCTION

```

-----
twenty-fifth-08-2023
twenty-ninth-05-2015
twenty-ninth-06-2012
twenty-first-07-2023
twenty-eighth-07-2023
sixteenth-06-2020
twenty-second-07-2022
twenty-fourth-02-2023
twenty-sixth-08-2022
fourth-07-2007
fourteenth-08-2013

```

SPTH_FUNCTION

```

-----
sixteenth-06-2020
nineteenth-08-2021
sixteenth-12-2022
twenty-sixth-07-2019
twenty-sixth-04-2013
eighth-08-2013
fourteenth-11-2008
eighteenth-10-2017

```

19 rows selected.

- **Calculate the total gross earnings for movies released after June 16, 2020.**

SQL> select sum(gross) as total_earnings from movies where releasedate>to_date('16-jun-2020','dd-mon-yyyy') ;

TOTAL_EARNINGS

```

-----
110550000

```

Description: Functions: Set Operations

Date:28/08/2023

Query

- **Create a new table IndustryHit (Id, title, genre, Certificate, Gross, Releasedate). Insert some movies from Movies table and some new movies in the new table IndustryHit.**

SQL> create table industryhit(id number(5) primary key,title varchar(40) not null,genre varchar2(40),certificate varchar2(20),gross number(10,2),releasedate date);

Table created.

SQL> desc industryhit;

Name	Null?	Type
ID	NOT NULL	NUMBER(5)
TITLE	NOT NULL	VARCHAR2(40)
GENRE		VARCHAR2(40)
CERTIFICATE		VARCHAR2(20)
GROSS		NUMBER(10,2)
RELEASEDATE		DATE

SQL> insert into industryhit(id,title,genre,certificate,gross,releasedate)select id,title,'unknown' as genre,certificate,gross,releasedate from movies where id=301;

1 row created.

SQL> insert into industryhit(id,title,genre,certificate,gross,releasedate)select id,title,'unknown' as genre,certificate,gross,releasedate from movies where id=307;

1 row created.

SQL> insert into industryhit(id,title,genre,certificate,gross,releasedate)select id,title,'unknown' as genre,certificate,gross,releasedate from movies where id=309;

1 row created.

SQL> insert into industryhit(id,title,genre,certificate,gross,releasedate)select id,title,'unknown' as genre,certificate,gross,releasedate from movies where id=311;

1 row created.

SQL> insert into industryhit(id,title,genre,certificate,gross,releasedate)select id,title,'unknown' as genre,certificate,gross,releasedate from movies where id=315;

SQL> update industryhit set genre='adventure' where id=301;

1 row updated.

SQL> update industryhit set genre='musical drama' where id=307;

1 row updated.

SQL> update industryhit set genre='romantic and musical' where id=309;

1 row updated.

SQL> update industryhit set genre='thriller' where id=311;

1 row updated.

SQL> update industryhit set genre='horror' where id=315;

1 row updated.

SQL> select * from industryhit;

ID	TITLE	GENRE	CERTIFICATE	GROSS	RELEASEDA
301	Rdx	adventure	U/A	14000000	25-AUG-23
307	Ashiqui 2	musical drama	U	109000000	26-APR-13
309	Vaaranam Aayiram	romantic and musical	U/A	40000000	14-NOV-08
311	Oppenheimer	thriller	U/A	730000000	21-JUL-23
315	Open grave	horror	U/A	3000000	14-AUG-13

SQL> insert into industryhit values(302,'Neram','Romance','U',5.30000000,'10-may-2013');

1 row created.

SQL> insert into industryhit values(303,'Kadal','Romance','U',6.70000000,'31-jan-2013');

1 row created.

SQL> insert into industryhit values(304,'Bheed','Drama','U/A',3.30000000,'24-mar-2013');

1 row created.

SQL> insert into industryhit values(308,'Anjam Pathira','Thriller','U/A',28000000,'10-jan-2020');

1 row created.

SQL> select * from industryhit;

ID	TITLE	GENRE	CERTIFICATE	GROSS	RELEASEDA
301	Rdx	adventure	U/A	14000000	25-AUG-23
307	Ashiqui 2	musical drama	U	109000000	26-APR-13
309	Vaaranam Aayiram	romantic and musical	U/A	40000000	14-NOV-08
311	Oppenheimer	thriller	U/A	730000000	21-JUL-23
315	Open grave	horror	U/A	3000000	14-AUG-13
302	Neram	Romance	U	5.3	10-MAY-13
303	Kadal	Romance	U	6.7	31-JAN-13
304	Bheed	Drama	U/A	3.3	24-MAR-13
308	Anjam Pathira	thriller	U/A	28000000	10-JAN-20

9 rows selected.

- **Retrieve the titles of all movies and industry hits which are in the action thriller genre.**

SQL> select title from movies INTERSECT select title from industryhit where genre='thriller';

TITLE

Oppenheimer

- **Retrieve the titles of all movies including industry hits.**

SQL> select title from movies UNION ALL select title from industryhit;

TITLE

Rdx
 premam
 Usthad Hotel
 Oppenheimer
 Voice of Sathyanathan
 Driven
 Dejavu
 Santhosham
 Diary
 Open the door please
 Open grave

TITLE

The Marshes
 Home
 Avatar
 Once upon a time in Hollywood
 Ashiqui 2
 Chennai express
 Vaaranam Aayiram
 Mersal
 Rdx
 Ashiqui 2
 Vaaranam Aayiram

TITLE

Oppenheimer
 Open grave
 Neram
 Kadal
 Bheed
 Anjam Pathira

28 rows selected.

- **Retrieve the titles of all movies which are not industry hits.**

SQL> select title from movies MINUS select title from industryhit;

TITLE

Avatar
 Chennai express
 Dejavu
 Diary
 Driven
 Home
 Mersal
 Once upon a time in Hollywood
 Open the door please
 Santhosham
 The Marshes

TITLE

Usthad Hotel
 Voice of Sathyanathan
 premam

14 rows selected.

Description: Illustration of Group By having clause

Date:29/08/2023

Query

- For all genres, display genre type and the sum of all Gross for each genre. Name the derived column SUM_COLL.

SQL> select genre,sum(gross) as sum_coll from industryhit group by genre;

GENRE	SUM_COLL
thriller	758000000
adventure	14000000
Romance	12
Drama	3.3
musical drama	109000000
romantic and musical	40000000
horror	3000000

- For all genres, display the genre type and the number of titles. Name the derived column TITLE_COUNT.

SQL> select genre,count(title)title_count from industryhit group by genre;

GENRE	TITLE_COUNT
thriller	2
adventure	1
Romance	2
Drama	1
musical drama	1
romantic and musical	1
horror	1

7 rows selected.

- Display the genres which have more than 3 titles.

SQL> select genre,count(title) from industryhit group by genre having count(title)>3;

GENRE	COUNT(TITLE)
thriller	4

- **Retrieve the total number of movies released in each year, only for years with at least 5 movies.**

SQL> select extract(year from releasedate) year,count(*)total_movies from industryhit group by extract(year from releasedate) having count(*)>=5;

YEAR	TOTAL_MOVIES
2013	5

- **List the certificates along with the number of movies for each certificate, but only show certificates with more than 3 movies.**

SQL> select certificate,count(*)total_movies from industryhit group by certificate having count(*)>3;

CERTIFICATE	TOTAL_MOVIES
U/A	7
U	4

- **Show the total gross earnings for each certificate, but only for certificates with total gross greater than \$1 million.**

SQL> select certificate,sum(gross) from industryhit group by certificate having sum(gross)>1000000;

CERTIFICATE	SUM(GROSS)
U/A	902000003
U	213000012

- **List the release years with the highest number of movies and the corresponding movie count, limited to the top 3 years.**

SQL> select * from (select extract(year from releasedate)year,count(*) from industryhit group by extract(year from releasedate) order by count(*) desc) where rownum<=3;

YEAR	COUNT(*)
2013	5
2023	2
2020	2

Description: Sub Queries

Date:29/08/2023

Query

- Retrieve the titles and runtime of movies with the highest Metascore.

SQL> select title, runtime from movies where metascore in (select max(metascore) from movies);

TITLE	RUNTIME
Oppenheimer	3

- List the titles of movies with a Gross amount greater than the average Gross amount of all movies.

SQL> select title from movies where gross > (select avg(gross) from movies);

TITLE
Oppenheimer
Avatar
Mersal

- Retrieve the titles and descriptions of movies with a Metascore lower than the average Metascore.

SQL> select title, description from movies where metascore < (select avg(metascore) from movies);

TITLE	DESCRIPTION
Usthad Hotel	Aspiring chef
Dejavu	Crime novelist
Santhosham	Bond between siblings
Open the door please	photo frame
Open grave	killer
The Marshes	research and survival
Vaaranam Aayiram	Rescue mission and fathers death
Mersal	corruption and arresting doctor

8 rows selected.

- **List the movie titles and their IMDb ratings for movies released in the year with the highest average IMDb rating.**

SQL> select * from (select title,extract(year from releasedate)year,avg(imdbrating) from movies group by title,extract(year from releasedate) order by avg(imdbrating) desc) where rownum=1;

TITLE	YEAR	AVG(IMDBRATING)
Home	2021	9.1

- **Retrieve the movie titles and their IMDb ratings for movies that have a Metascore greater than twice their IMDb rating.**

SQL> select title,imdbrating from movies where (metascore,imdbrating) in (select metascore,imdbrating from movies where metascore>2*imdbrating);

TITLE	IMDBRATING
Rdx	5.9
premam	12
Oppenheimer	8.6
Voice of Sathyanathan	7.4
Driven	9
Diary	7.3
Open grave	6.2
Home	9.1
Avatar	7.6
Once upon a time in Hollywood	6.7
Ashiqui 2	14

TITLE	IMDBRATING
Chennai express	6

12 rows selected .

- **Find the title and gross amount of the top 3 highest-grossing movies.**

SQL> select * from(select title,gross from movies order by gross desc) where rownum<=3;

TITLE	GROSS
Mersal	2600000000
Oppenheimer	730000000

Avatar

230000000

- **Calculate the total number of votes received by movies released in the year 2022.**

SQL> select * from(select sum(votes) from movies where extract(year from releasedate)=2022);

SUM(VOTES)

400

- **List the titles and certificate ratings of movies that have an IMDb rating below the average IMDb rating.**

SQL> select title,imdbrating,certificate from movies where imdbrating<(select avg(imdbrating) from movies);

TITLE	IMDBRATING CERTIFICATE	
Rdx	5.9	U/A
Voice of Sathyanathan	7.4	U
Dejavu	6.7	U/A
Santhosham	7.4	U/A
Diary	7.3	U/A
Open the door please	6.6	U/A
Open grave	6.2	U/A
The Marshes	7.4	A
Avatar	7.6	A
Once upon a time in Hollywood	6.7	A
Chennai express	6	U/A

TITLE	IMDBRATING CERTIFICATE	
Mersal	7.5	U/A

12 rows selected.

Description: Views

Date:22/08/2023

Query:

- **Create a view called MovieDetails that combines information from the Movies, Directors, and Stars tables to display movie titles, directors' names, and the names of stars who acted in those movies.**

SQL> create view moviedetails as select m.title as movies,d.name as directors,s.name as stars from movies m,directors d,stars s,moviesdirectors md,moviesstars ms where m.id=md.moviesid and m.id=ms.moviesid and md.directorsid=d.id and ms.starsid=s.id;

View created.

SQL> select * from moviedetails;

MOVIES	DIRECTORS	STARS
-----	-----	-----
Rdx	Nahas Hidayath	Shane Nigam
premam	Alphonse Puthran	Nivin Pauly
Usthad Hotel	Anwar Rasheed	Dulquer Salman
Home	Rojin Thomas	Indrans
Avatar	James Cameroon	Sully
Once upon a time in Hollywood	Quentin Tarantino	Leonardo Dicaprio
Ashiqui 2	Mohit Suri	Aditya Roy kapur
Chennai express	Rohit Shetty	Shah Rukh Khan
Vaaranam Aayiram	Gautham Vasudev Menon	Surya
Mersal	Atlee	Vijay

10 rows selected.

- **Create a view called HighlyRatedMovies that displays movies with IMDb ratings greater than 8.0, including their titles and ratings.**

SQL> create view highlyratedmovies as select title,imdbrating from movies where imdbrating>8.0;

View created.

SQL> select * from highlyratedmovies;

TITLE	IMDBRATING
premam	12
Usthad Hotel	9
Oppenheimer	8.6
Driven	9
Home	9.1
Ashiqui 2	14
Vaaranam Aayiram	8.2

7 rows selected.

- **Create a view called DirectorMovies that provides a list of directors along with the number of movies they have directed.**

SQL> create view directormovies as select d.name
directors,count(md.moviesid)no_of_movies from directors d,moviesdirectors md where
d.id=md.directorsid group by d.name;

View created.

SQL> select * from directormovies;

DIRECTORS	NO_OF_MOVIES
Mohit Suri	1
James Cameroon	1
Anwar Rasheed	1
Rojin Thomas	1
Atlee	1
Gautham Vasudev Menon	1
Nahas Hidayath	1
Alphonse Puthran	1
Rohit Shetty	1
Quentin Tarantino	1

10 rows selected.

- **Create a view called StarMovies that displays stars' names and the titles of movies they have acted in.**

SQL> create view starmovies as select s.name,m.title from movies m,stars s,moviesstars
ms where m.id=ms.moviesid and s.id=ms.starsid;

View created.

SQL> select * from starmovies;

NAME	TITLE
Shane Nigam	Rdx
Nivin Pauly	premam
Dulquer Salman	Usthad Hotel
Indrans	Home
Sully	Avatar
Leonardo Dicaprio	Once upon a time in Hollywood
Aditya Roy kapur	Ashiqui 2
Shah Rukh Khan	Chennai express
Surya	Vaaranam Aayiram
Vijay	Mersal

10 rows selected.

- **Create a view called LongestMovies that lists the titles of movies with the longest runtimes (duration).**

SQL> create view longestmovies as select title,duration from(select title,runtime as duration from movies order by runtime desc)where rownum<=5;

View created.

SQL> select * from longestmovies;

TITLE	DURATION
Avatar	3.2
Oppenheimer	3
Home	2.8
Vaaranam Aayiram	2.8
Once upon a time in Hollywood	2.8

- **Create a view called LanguageDistribution that shows the distribution of movies based on the languages they were released in, including the count of movies for each language.**

SQL> create view languagedistribution as select language,count(*)movies from movies group by language;

View created.

SQL> select * from languagedistribution;

LANGUAGE	MOVIES
malayalam	5
	1
english	7
tamil	4
hindi	2

- **Create a view called GrossEarnings that displays movies with their titles and gross earnings, sorted by earnings in descending order.**

SQL> create view grossearnings as select title,gross as grossearnings from movies order by gross desc;

View created.

SQL> select * from grossearnings;

TITLE	GROSSEARNINGS
Mersal	2600000000
Oppenheimer	730000000
Avatar	230000000
Voice of Sathyanathan	109500000
Chennai express	109000000
Ashiqui 2	109000000
premam	76000000
Usthad Hotel	41000000
Vaaranam Aayiram	40000000
Once upon a time in Hollywood	37000000
Dejavu	15000000

TITLE	GROSSEARNINGS
Rdx	14000000
The Marshes	8000000
Diary	4000000
Open grave	3000000
Driven	3000000
Home	3000000
Open the door please	100000
Santhosham	.24

19 rows selected.

- **Create a view called IndustryHitMovies that shows the titles and release dates of industry hit movies.**

SQL> create view industryhitmovies as select title,releasedate from industryhit;

View created.

SQL> select * from industryhitmovies;

TITLE	RELEASEDA
Rdx	25-AUG-23
Ashiqui 2	26-APR-13
Vaaranam Aayiram	14-NOV-08
Oppenheimer	21-JUL-23
Open grave	14-AUG-13
Neram	10-MAY-13
Kadal	31-JAN-13
Bheed	24-MAR-13
Anjam Pathira	10-JAN-20
Cold Case	30-JUN-21
Forensic	28-FEB-20

11 rows selected.

- **Create a view called MovieVotes that displays movies along with their titles and the number of votes they have received.**

SQL> create view movievotes as select title,votes from movies;

View created.

SQL> select * from movievotes;

TITLE	VOTES
Rdx	443
premam	21991
Usthad Hotel	452
Oppenheimer	400
Voice of Sathyanathan	90
Driven	104
Dejavu	100
Santhosham	440
Diary	200
Open the door please	90
Open grave	100

TITLE	VOTES
The Marshes	89
Home	95
Avatar	100
Once upon a time in Hollywood	92
Ashiqui 2	84
Chennai express	120
Vaaranam Aayiram	49
Mersal	300

19 rows selected.

- **Create a view called CertifiedMovies that lists movies with their titles and certificates (e.g., U/A, U).**

SQL> create view certifiedmovies as select title,certificate from movies;

View created.

SQL> select * from certifiedmovies;

TITLE	CERTIFICATE
Rdx	U/A
premam	U/A
Usthad Hotel	U/A
Oppenheimer	U/A
Voice of Sathyanathan	U
Driven	U/A
Dejavu	U/A
Santhosham	U/A
Diary	U/A
Open the door please	U/A
Open grave	U/A

TITLE	CERTIFICATE
The Marshes	A
Home	U/A
Avatar	A
Once upon a time in Hollywood	A
Ashiqui 2	U
Chennai express	U/A
Vaaranam Aayiram	U/A
Mersal	U/A

19 rows selected.

Activity # 4**4.Practice PL/SQL**

Description : Introduction to PL/SQL**Date: 22/08/2023**

Query:

- **Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and corresponding values of calculated area in an empty table named Areas, consisting of two columns Radius and Area.**

```
SQL> create table area(radius number(10,2),area number(10,2));
```

```
Table created.
```

```
declare
```

```
i number;
```

```
a number(10,2);
```

```
begin
```

```
for i in 3..7
```

```
loop
```

```
a:=3.14*i*i;
```

```
insert into Areas values(i,a);
```

```
end loop;
```

```
dbms_output.put_line('Area is :'||a);
```

```
end;
```

```
output:
```

```
SQL> @"C:\Users\KHADEEJA C N\Desktop\plsql\area.sql"
```

```
12 /
```

```
Area is :153.86
```

```
PL/SQL procedure successfully completed.
```

```
SQL> select * from Areas;
```

RADIUS	AREA
3	28.26
4	50.24
5	78.5
6	113.04
7	153.86

- **Write a PL/SQL block of code for inverting a number accepted from the console.**

```

declare
num number;
rev number;
rem number;
begin
rev:=0;
num:=&num;
while num!=0
loop
rem:=mod(num,10);
rev:=rev*10+rem;
num:=trunc(num/10);
end loop;
dbms_output.put_line('Inverse is :'||rev);
end;

```

output:

SQL> @"C:\Users\KHADEEJA C N\Desktop\plsql\inverse.sql"

16 /

Enter value for num: 456

old 7: num:=#

new 7: num:=456;

Inverse is :654

PL/SQL procedure successfully completed.

- **Write a PL/SQL code block that will accept an account number from the user and debit an amount of Rs.2000 from the account if the account has a minimum balance of 500 after the amount is debited. The process is fired on the Accounts table.**

SQL> create table accounts(accno number(10),balance number(10,2));

Table created.

SQL> insert into accounts values(101,5000);

1 row created.

SQL> insert into accounts values(102,1000);

1 row created.

SQL> insert into accounts values(103,10000);

1 row created.

SQL> insert into accounts values(104,3000);

1 row created.

SQL> insert into accounts values(105,500);

1 row created.

SQL> select * from accounts;

ACCNO	BALANCE
101	5000
102	1000
103	10000
104	3000
105	500

declare

ano number;

b number(10,5);

begin

ano:=&ano;

select balance into b from accounts where accno=ano;

if b-2000<500 then

dbms_output.put_line('Transaction not possible,insufficient balance');

else

update accounts set balance=balance-2000 where accno=ano;

dbms_output.put_line('Transaction successful');

end if;

end;

output:

```
SQL> @"C:\Users\KHADEEJA C N\Desktop\plsql\accounts.sql"
```

```
14 /
```

```
Enter value for ano: 101
```

```
old 5: ano:=&ano;
```

```
new 5: ano:=101;
```

```
Transaction successful
```

```
PL/SQL procedure successfully completed.
```

```
SQL> @"C:\Users\KHADEEJA C N\Desktop\plsql\accounts.sql"
```

```
14 /
```

```
Enter value for ano: 102
```

```
old 5: ano:=&ano;
```

```
new 5: ano:=102;
```

```
Transaction not possible,insufficient balance
```

```
PL/SQL procedure successfully completed.
```

```
SQL> select * from accounts;
```

ACCNO	BALANCE
101	3000
102	1000
103	10000
104	3000
105	500

- **Write a PL/SQL block of code that updates the salaries of Maria Jacob and Albert by Rs. 2000/- and Rs.2500/- respectively. Then check to see that the total salary does not exceed 75000. If the total salary is greater than 75000, then undo the updates made to salaries of both. (Use savepoint, rollback and commit).**

```
SQL> create table salary(id number(5),name varchar2(40),sal number(12,2));
```

```
Table created.
```

```
SQL> insert into salary values(101,'Mariya Jacob',10000);
```

```
1 row created.
```

```
SQL> insert into salary values(102,'Albert',20000);
```

```
1 row created.
```

```
SQL> insert into salary values(103,'Khadeeja',35000);
```


1 row created.

SQL> insert into salary values(104,'Faris',6000);

1 row created.

SQL> select * from salary;

ID NAME	SAL
101 Mariya Jacob	10000
102 Albert	20000
103 Khadeeja	35000
104 Faris	6000

declare

total number;

begin

savepoint s1;

update salary set sal=sal+2000 where id =102;

update salary set sal=sal+2500 where id =103;

select sum(sal) into total from salary;

dbms_output.put_line('total : '|| total);

if total>75000 then

rollback to s1;

else

commit;

end if;

end;

ouput:

SQL> @"C:\Users\KHADEEJA C N\Desktop\plsql\salary.sql"

13 /

total : 75500

PL/SQL procedure successfully completed.

SQL> select * from salary;

ID NAME	SAL
101 Mariya Jacob	10000
102 Albert	20000
103 Khadeeja	35000
104 Faris	6000

Description : Illustration of cursors

Date: 22/08/2023

Query:

Illustration of Implicit cursor.

- Write a PL/SQL block to accept an employee number and update the salary of that employee to raise the salary by 0.15. Display appropriate message based on the existence of the record in the employee table.

```
SQL> create table employee(empno number(5),empname varchar2(40),job
varchar2(40),salary number(12,2));
```

Table created.

```
SQL> insert into employee values(101,'Khadeeja','analyst',50000);
```

1 row created.

```
SQL> insert into employee values(102,'Faris','developer',100000);
```

1 row created.

```
SQL> insert into employee values(103,'Sahala','designer',40000);
```

1 row created.

```
SQL> insert into employee values(104,'Aparna','analyst',60000);
```

1 row created.

```
SQL> insert into employee values(105,'Arun','engineer',80000);
```

1 row created.

```
SQL> select * from employee;
```

EMPNO	EMPNAME	JOB	SALARY
101	Khadeeja	analyst	50000
102	Faris	developer	100000
103	Sahala	designer	40000
104	Aparna	analyst	60000
105	Arun	analyst	80000

```

declare
begin
update employee set salary=salary+(salary*0.15) where empno=&empno;
if sql%found then
dbms_output.put_line(sql%rowcount ||'record is updated');
end if;
end;
```

output:

```
SQL> @"C:\Users\KHADEEJA C N\Desktop\plsql\implicit1.sql"
```

```
8 /
```

Enter value for empno: 101

```
old 3: update employee set salary=salary+(salary*0.15) where empno=&empno;
```

```
new 3: update employee set salary=salary+(salary*0.15) where empno=101;
```

```
1record is updated
```

```
PL/SQL procedure successfully completed.
```

```
SQL> select * from employee;
```

EMPNO	EMPNAME	JOB	SALARY
101	Khadeeja	analyst	57500
102	Faris	developer	100000
103	Sahala	designer	40000
104	Aparna	analyst	60000
105	Arun	analyst	80000

- The HRD manager decides to raise the salary of employees working as 'analyst' by 0.15. Write a cursor to update the salary of the employees. Display the no. of employee records that has been modified.

```

declare
begin
update employee set salary=salary+(salary*0.15) where job='analyst';
```

```

if sql%found then
dbms_output.put_line(sql%rowcount ||'record is updated');
end if;
end;

```

output:

```
SQL> @"C:\Users\KHADEEJA C N\Desktop\plsql\implicit2.sql"
```

```
8 /
```

```
3record is updated
```

```
PL/SQL procedure successfully completed.
```

```
SQL> select * from employee;
```

EMPNO	EMPNAME	JOB	SALARY
101	Khadeeja	analyst	66125
102	Faris	developer	100000
103	Sahala	designer	40000
104	Aparna	analyst	69000
105	Arun	analyst	92000

Illustration of explicit cursor.

- Write an explicit cursor to display the name, department, salary of the first 5 employees getting the highest salary.

```

declare
cursor empcursor is select empname,deptname,salary from (select
empname,deptname,salary from employee e,department d where e.deptno=d.deptno order
by salary desc) where rownum<=5;
vemp empcursor%rowtype;
begin
dbms_output.put_line('*****employee details*****');
open empcursor;
fetch empcursor into vemp;
while empcursor%found
loop
dbms_output.put_line('Empname:'|| vemp.empname);
dbms_output.put_line('Departmentname:'|| vemp.deptname);
dbms_output.put_line('Salary:'|| vemp.salary);
dbms_output.put_line('*****');
fetch empcursor into vemp;

```

```
end loop;
close empcursor;
end;
```

output:

```
SQL> @ C:\Users\CCL35\Desktop\pls\expilicite_record1.sql;
50 /
```

*****employee details*****

```
Empname:ARNOLD LEONARD AMON
Departmentname:COMPUTER SERVICE DIVISION
Salary:152750
```

```
Empname:DONA ANICE SIBY
Departmentname:COMPUTER SERVICE DIVISION
Salary:46500
```

```
Empname:PHILIP VINCENT
Departmentname:PLANNING
Salary:41250
```

```
Empname:ALFRIN LUIZ
Departmentname:SUPPORT SERVICES
Salary:40175
```

```
Empname:SHILVY K K
Departmentname INFORMATION CENTER
Salary:39250
```

PL/SQL procedure successfully completed.

- The HRD manager decides to raise the salary of employees working as 'analyst' by 0.15. Whenever any such raise is given to the employees, a record for the same is maintained in the emp_raise table. It includes the employee number, the date when the raise was given and actual raise. Write a PL/SQL block to update the salary of the employees and insert a record in the emp_raise table.

Emp_raise(empcode, raisedate,raise_amt)

```

declare
v_raise_amt number;
v_raisedate date:=sysdate;
cursor empcursor is select * from employee where job='analyst';
vemp empcursor%rowtype;
begin
open empcursor;
fetch empcursor into vemp;
while empcursor%found
loop
update employee set salary=salary+salary*0.15 where empno=vemp.empno;
insert into emp_raise(empcode,raisedate,raise_amt)
values(vemp.empno,v_raisedate,vemp.salary*0.15);
fetch empcursor into vemp;
end loop;
close empcursor;
dbms_output.put_line('salary updated and records inserted in the new table successfully');
end;

```

output:

```

SQL> @ "C:\Users\KHADEEJA C N\Desktop\record
2023_dbms\plsql\explicit_record2.sql"

```

18 /

PL/SQL procedure successfully completed.

```

SQL> select * from employee;

```

EMPNO	EMPNAME	JOB	SALARY	DEPARTMENT
<hr/>				
101	Khadeeja	analyst	69000	marketing
102	Faris	developer	100000	IT
103	Sahala	designer	40000	production
104	Aparna	analyst	80500	marketing
105	Arun	analyst	11500	marketing
106	Mariya	engineer	75000	IT
107	Ravi	clerk	45000	finance

7 rows selected.

```

SQL> select * from emp_raise;

```

EMPCODE	RAISEDATE	RAISE_AMT
<hr/>		
101	23-SEP-23	9000
104	23-SEP-23	10500
105	23-SEP-23	1500

Description : Illustration of Procedures

Date: 20/09/2023

Query:

- Write a PL/SQL block which makes use of a stored procedure Proj_emp (emp_name varchar2(50)) which finds all the details of the projects involved by the given employee.

```
create or replace procedure Proj_emp(ename in varchar2)
as
cursor p is select projname from emp_proj ep inner join employee e on e.empno
=ep.empno inner join project p on ep.projno=p.projno where empname=ename;
project_name varchar2(50);
begin
open p;
loop
fetch p into project_name;
exit when p%notfound;
DBMS_OUTPUT.PUT_LINE(PROJECT_NAME);
end loop;
close p;
end;
declare
e varchar2(50);
begin
e:=&e;
Proj_emp(e);
end;
```

output:

```
SQL> @ C:\Users\CCL35\Desktop\khadeeja\plsql\procedures\project.sql
20 /
Procedure created.
SQL> @ C:\Users\CCL35\Desktop\khadeeja\plsql\procedures\pro.sql
12 /
Enter value for e: 'PHILIP VINCENT'
old 7: e:=&e;
```

```

new 7: e:='PHILIP VINCENT';
ADMIN SERVICES
WELD LINE PLANNING
W L PROGRAM DESIGN
PL/SQL procedure successfully completed.

```

- **Write a procedure to check whether a string is a palindrome . Call the procedure to list all the palindrome names in the employee table.**

```

create or replace procedure palindron_checker(arg in varchar2)
as
rev varchar2(30);
begin
select reverse(arg) into rev from dual;
if arg = rev
then
dbms_output.put_line(arg);
end if;
end ;
declare
cursor empnames is select upper(empname) from employee;
empname varchar2(30);
begin
open empnames;
loop
fetch empnames into empname;
exit when empnames%notfound;
palindron_checker(empname);
end loop;
close empnames;
end;

```

output:

```

SQL> @ C:\Users\CCL35\Desktop\khadeeja\plsql\procedures\palindrome.sql
49 /
Procedure created.
SQL> @ C:\Users\CCL35\Desktop\khadeeja\plsql\procedures\pal.sql
15 /
ANNA
PL/SQL procedure successfully completed.

```


- **Write a PL/SQL block which retrieve all the employee into a cursor and display the details of all assigned projects for each employee using a stored procedure Proj_emp (emp_name varchar2(50)).**

```

create or replace procedure Proj_emp(ename in varchar2)
as
cursor p is select projname from emp_proj ep inner join employee e on e.empno
=ep.empno inner join project p on ep.projno=p.projno where empname=ename;
project_name varchar2(50);
begin
open p;
loop
fetch p into project_name;
exit when p%notfound;
DBMS_OUTPUT.PUT_LINE(PROJECT_NAME);
end loop;
close p;
end;
declare
cursor p is select empname from employee;
e employee.empname%type;
begin
open p;
loop
fetch p into e;
exit when p%notfound;
dbms_output.put_line('Name='||e);
proj_emp(e);
dbms_output.put_line('*****');
end loop;
end;

```

output:

```
SQL> @ C:\Users\CCL35\Desktop\khadeeja\plsql\emp.sql
```

```
20 /
```

```
Procedure created.
```

```
SQL> @ C:\Users\CCL35\Desktop\khadeeja\plsql\employee.sql
```

```
17 /
```

```
Name=ARNOLD LEONARD AMON
```

```
ADMIN SERVICES
```

```
WELD LINE AUTOMATION
```

```
*****
```

Name=PHILIP VINCENT
ADMIN SERVICES
WELD LINE PLANNING
W L PROGRAM DESIGN

Name=SHILVY K K
USER EDUCATION
QUERY SERVICES

Name=ALFRIN LUIZ
GEN SYSTEM SERVICES
OPERATION SUPPORT

Name=OSHINA ANTONY
W L PROGRAMMING
ADMIN SERVICES

Name=BINCY PAUL
GENERAL AD SYSTEMS

Name=ANAMIKA PAUL
GENERAL AD SYSTEMS
OPERATION

Name=ANEESH DENNY
GENERAL AD SYSTEMS
SYSTEMS SUPPORT

Name=DONA ANICE SIBY

Name=JUNAID K V
W L PROGRAM DESIGN

Name=CHRISTEENA THOMAS
W L PROGRAM DESIGN

Name=JEFFIN DOMINIC
ADMIN SERVICES

Name=JEWEL BIJOY
W L ROBOT DESIGN
SYSTEMS SUPPORT
GEN SYSTEM SERVICES

Name=MELLOW REEBA THOMAS
GEN SYSTEM SERVICES

Name=JOHN VARGHESE

Name=ASHREENA HASSAN

Name=VISHAK VIJAYAKUMAR
WELD LINE PLANNING

Name=MERLIN M.D
W L ROBOT DESIGN

Name=MARIA JOHN

Name=VISHALAKSHI V PRABHU
PAYROLL PROGRAMMING

Name=ANGEL PAUL
PAYROLL PROGRAMMING

Name=RIYA TONEY
USER EDUCATION

Name=PRIYA TOMY
QUERY SERVICES
USER EDUCATION

Name=ARYAMOL ASOKAN
USER EDUCATION

Name=GEO GEORGE

Name=JIMMY THOMSON
OPERATION

Name=ALAN PAYYAPPILLY
SCP SYSTEM SUPPORT

Name=BEN PETER MATHEW
OPERATION SUPPORT

Name=KRISHNANUNNI S
SCP SYSTEM SUPPORT

Name=AHALYA V A

APPLICATIONS SUPPORT

WELD LINE AUTOMATION

Name=ANNA

W L PROGRAMMING

PL/SQL procedure successfully completed.



Description : Illustration of Functions

Date: 21/09/2023

Query:

- Write a function to find the reverse of EmpNo in Employee table and display the EmpNo and Reversed(Emp No) of the first 5 employees using an SQL Query.

```

create or replace function
rev(empno in char)
return char
is
r char(10);
begin
select reverse(empno) into r from dual;
return r;
end;
/
declare
cursor c1 is select empno from employee where rownum <=5;
emp c1%rowtype;
rev1 char(10);
begin
open c1;
fetch c1 into emp;
while c1%found
loop
rev1:=rev(emp.empno);
dbms_output.put_line(emp.empno||' '||rev1);
fetch c1 into emp;
end loop;
close c1;
end;
/

```

output:

```
SQL> @ "C:\Users\KHADEEJA C N\Desktop\record 2023_dbms\plsql\fun_rev.sql"
```

Function created.

```
SQL> @ "C:\Users\KHADEEJA C N\Desktop\record  
2023_dbms\plsql\function_reverse.sql"
```

```
E0010      0100E  
E0020      0200E  
E0030      0300E  
E0050      0500E  
E0060      0600E
```

PL/SQL procedure successfully completed.

- **Write a function that would check for the existence of an employee in the employee table given an EmpNo. If existing employee, check whether he is the manager of any department and display messages accordingly.**

```
create or replace function chk_manager(empno in varchar2) return varchar2
as
empname employee.empname%type;
deptname department.deptname%type;
cursor emp_cur is select empname,deptname from employee e inner join department d on
e.deptno=d.deptno where empno=empno and job like 'MANAGER%';
begin
select empname into empname from employee where empno=empno;
If sql%found
then
dbms_output.put_line('Employee Found : '||empname);
open emp_cur;
fetch emp_cur into empname,deptname;
if emp_cur%notfound
then
return 'Not Manager';
else
return empname||' Manager of '||deptname;
end if;
else
return 'Employee Not Found';
end if;
close emp_cur;
end;
```

/

output:

```
SQL> execute dbms_output.put_line(chk_manager('E0030'));
```

Employee Found : SHILVY K K

SHILVY K K Manager of INFORMATION CENTER

PL/SQL procedure successfully completed.



Description : Illustration of Triggers

Date: 21/09/2023

Query:

- Consider the table Employee. Write PL/SQL statements to create a trigger when fired checks the operation performed on a table and based on the operation, a variable is assigned the value 'update' or 'delete'. Previous values of the modified record of the table Employee are stored into the appropriate variables declared and inserted to the audit table AuditEmployee.

```
create or replace trigger emptrig before delete or update on employee
for each row
declare
ch varchar2(15);
begin
if deleting then
ch:='delete';
insert into auditemployee
values(:old.empno,:old.empname,:old.mobile,:old.salary,:old.job,ch);
elsif updating then
ch:='update';
insert into auditemployee
values(:old.empno,:old.empname,:old.mobile,:old.salary,:old.job,ch);
end if;
end;
```

output:

```
SQL> create table auditemployee(empno char(10),empname varchar2(30),mobile
number(10),salary number(10,2),job varchar2(15));
```

Table created.

```
SQL> @ C:\Users\CCL35\Desktop\khadeeja\plsql\triggers\emptrigger.sql
14 /
```

Trigger created.

```
SQL> update employee set empname='KHADEEJA' where empno='E0340';
```

1 row updated.

```
SQL> select * from auditemployee;
```


EMPNO	EMPNAME	MOBILE	SALARY JOB	STATUS
E0340	ANJALI NAIR	9466976574	20370 FIELDREP	update

- **Write PL/SQL statements to create a trigger which generates an error messages if the salary is below or beyond the valid range 0-5000 on the employee table. The triggering events are update and insert.**

```
CREATE OR REPLACE TRIGGER check_salary_range
BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW
DECLARE
BEGIN
IF :NEW.salary < 0 OR :NEW.salary > 5000 THEN
RAISE_APPLICATION_ERROR(-20001, 'Salary is out of the valid range (0-5000).');
END IF;
END;
/
```

output:

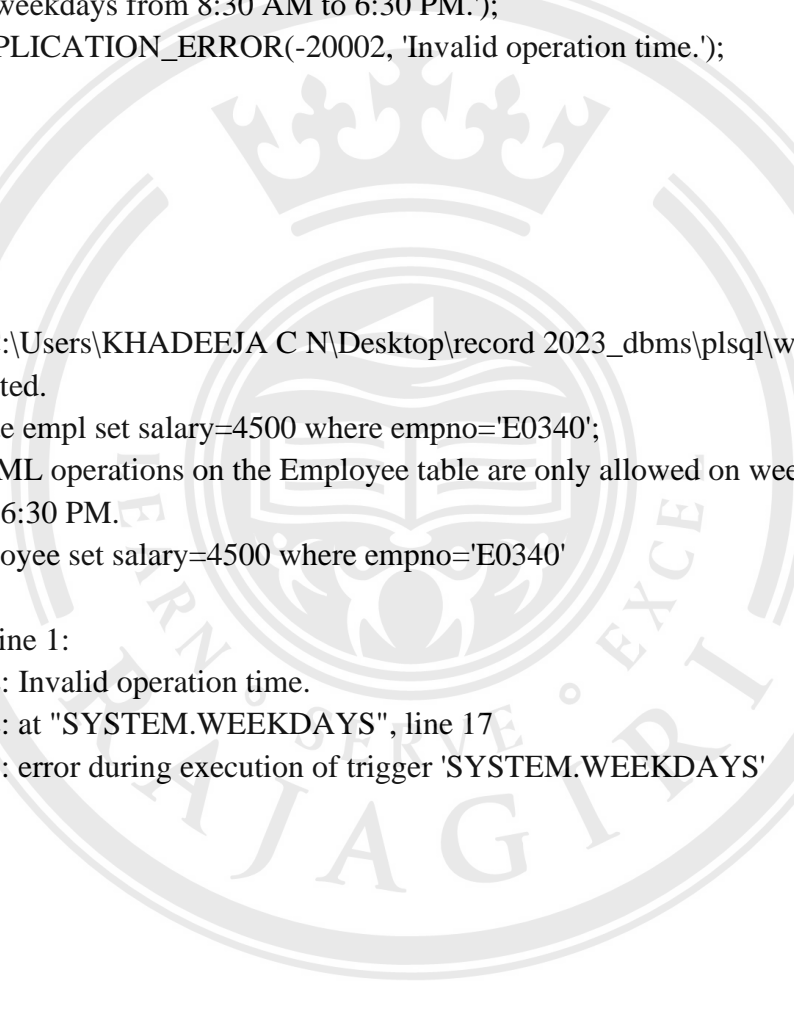
```
SQL> @ "C:\Users\KHADEEJA C N\Desktop\record
2023_dbms\plsql\trigger_salary.sql"
Trigger created.
SQL> update empl set salary=5500 where empno='E0340';
update employee set salary=5500 where empno='E0340'
*
ERROR at line 1:
ORA-20001: Salary is out of the valid range (0-5000).
ORA-06512: at "SYSTEM.CHECK_SALARY_RANGE", line 4
ORA-04088: error during execution of trigger 'SYSTEM.CHECK_SALARY_RANGE'
SQL> update employee set salary=4500 where empno='E0340';
1 row updated.
```

- **Write PL/SQL statements to create a trigger that limits the DML actions to the Employee table to weekdays from 8.30am to 6.30pm. If a user tries to insert/update/delete a row in the Employee table, a warning message will be prompted.**

```
CREATE OR REPLACE TRIGGER weekdays
BEFORE INSERT OR UPDATE OR DELETE ON Employee
FOR EACH ROW
DECLARE
```

```
current_day NUMBER;  
current_time NUMBER;  
BEGIN  
current_day := TO_NUMBER(TO_CHAR(SYSDATE, 'D'));  
current_time := TO_NUMBER(TO_CHAR(SYSDATE, 'HH24MI'));  
IF current_day >= 2 AND current_day <= 6  
AND current_time >= 830 AND current_time <= 1830 THEN  
NULL;  
ELSE
```

```
DBMS_OUTPUT.PUT_LINE('Warning: DML operations on the Employee table are only  
allowed on weekdays from 8:30 AM to 6:30 PM.');
```



```
RAISE_APPLICATION_ERROR(-20002, 'Invalid operation time.');
```

```
END IF;  
END;  
/
```

output:

```
SQL> @ "C:\Users\KHADEEJA C N\Desktop\record 2023_dbms\plsql\week.sql"  
Trigger created.  
SQL> update empl set salary=4500 where empno='E0340';  
Warning: DML operations on the Employee table are only allowed on weekdays from  
8:30 AM to 6:30 PM.  
update employee set salary=4500 where empno='E0340'  
*  
ERROR at line 1:  
ORA-20002: Invalid operation time.  
ORA-06512: at "SYSTEM.WEEKDAYS", line 17  
ORA-04088: error during execution of trigger 'SYSTEM.WEEKDAYS'
```