

**RAJALAKSHMI ENGINEERING  
COLLEGE RAJALAKSHMI NAGAR,  
THANDALAM – 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**  
An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai

**CS23332 DATABASE MANAGEMENT  
SYSTEMS LAB**

**Laboratory Record Notebook**

Name: **PRAVEEN.M**

Year / Branch / Section: **2<sup>nd</sup> year / B.E CSE – 'D'**

University Register No: **2116230701243**

College Roll No: **230701243**

Semester: **3<sup>rd</sup> Semester**

Academic Year: **2023 - 2024**

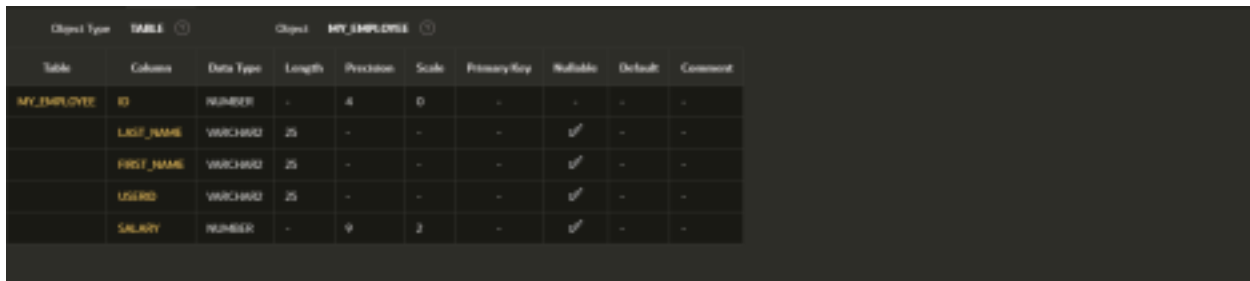
## **CS23332 DATABASE MANAGEMENT SYSTEMS**

|                 |                      |
|-----------------|----------------------|
| <b>NAME</b>     | <b>PRAVEEN.M</b>     |
| <b>ROLL NO.</b> | <b>2116230701243</b> |
| <b>DEPT</b>     | <b>CSE</b>           |
| <b>SEC</b>      | <b>‘D’</b>           |

|           |            |  |
|-----------|------------|--|
| Ex.No.: 1 |            | CREATION OF BASE TABLE AND<br>DML OPERATIONS |
| Date:     | 01/08/2024 |  |

1) Create MY\_EMPLOYEE table with the following structure

```
CREATE TABLE MY_EMPLOYEE(
ID Number(4) NOT NULL,
Last_name Varchar(25),
First_name Varchar(25),
Userid Varchar(25),
Salary Number(9,2)
);
```



| Table       | Columns    | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------------|------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| MY_EMPLOYEE | ID         | NUMBER    | 4      |           | 0     | -           | -        | -       | -       |
|             | LAST_NAME  | VARCHAR2  | 25     |           |       | -           | ✓        | -       | -       |
|             | FIRST_NAME | VARCHAR2  | 25     |           |       | -           | ✓        | -       | -       |
|             | USERID     | VARCHAR2  | 25     |           |       | -           | ✓        | -       | -       |
|             | SALARY     | NUMBER    | 9      |           | 2     | -           | ✓        | -       | -       |

2) Add the first row and second rows data to MY\_EMPLOYEE table from the sample table

```
Insert into
MY_EMPLOYEE(&ID,&LAST_NAME,&FIRST_NAME,&USERID,&SALARY
)
values(1,"Patel","Ralph","rpatel",895
2,"Dancs","Betty","bdancs",860);
```

3) Display the table with values

```
Select * from MY_EMPLOYEE;
```

| ID | LAST_NAME | FIRST_NAME | USERID    | SALARY |
|----|-----------|------------|-----------|--------|
| 2  | Dancs     | Betty      | bdancs    | 850    |
| 4  | Newman    | Chad       | Chewman   | 750    |
| 1  | Patel     | Rajiv      | rpatel    | 895    |
| 3  | Dei       | Ben        | BDeri     | 1000   |
| 5  | Roperbur  | Andrey     | aroperbur | 1050   |

5 rows returned in 0.00 seconds [Download](#)

- 4) populate the next two rows of data from the sample data. Concatenate the first letter of the first\_NAME with first seven letters of the last\_name to produce Userid

Update MY\_EMPLOYEES

Set Userid = substr(first\_name,1,1) || substr(last\_name,1,7)

Where ID in (3,4);

- 5) delete Betty dancs from my\_employee

table`1Delete from MY\_EMPLOYEE

Where FIRST\_NAME = 'Betty' and LAST\_NAME = 'Dancs';

| ID | LAST_NAME | FIRST_NAME | USERID    | SALARY |
|----|-----------|------------|-----------|--------|
| 1  | Patel     | Rajiv      | rpatel    | 895    |
| 5  | Dei       | Ben        | BDeri     | 1000   |
| 4  | Newman    | Chad       | Chewman   | 750    |
| 5  | Roperbur  | Andrey     | aroperbur | 1050   |

4 rows returned in 0.00 seconds [Download](#)

- 6) Empty the fourth row of the emp table

Delete from MY\_EMPLOYEE

Where ID = 5;

| ID | LAST_NAME | FIRST_NAME | USERID  | SALARY |
|----|-----------|------------|---------|--------|
| 1  | Patel     | Rajiv      | rpatel  | 895    |
| 3  | Dei       | Ben        | BDeri   | 1000   |
| 4  | Newman    | Chad       | Chewman | 750    |

- 7) Make the data additions

permanentCommit;

8) Change the last name of employee 3 to Drexler

Update MY\_EMPLOYEE

Set LAST\_NAME = "Drexler"

Where ID = 3;

| ID | LAST_NAME | FIRST_NAME | USERID  | SALARY |
|----|-----------|------------|---------|--------|
| 1  | Patel     | Ralph      | rpatel  | 895    |
| 3  | Drexler   | Ben        | Eden    | 1000   |
| 4  | Hewman    | Chad       | CHewman | 790    |

3 rows returned in 0.01 seconds [Download](#)

9) Change the salary to 1000 for all the employees with a salary less than 900.

Update MY\_EMPLOYEE

Set salary = 1000

Where salary < 900;

| ID | LAST_NAME | FIRST_NAME | USERID  | SALARY |
|----|-----------|------------|---------|--------|
| 1  | Patel     | Ralph      | rpatel  | 1000   |
| 3  | Drexler   | Ben        | Eden    | 1000   |
| 4  | Hewman    | Chad       | CHewman | 1000   |

3 rows returned in 0.00 seconds [Download](#)

|           |            |                    |
|-----------|------------|--------------------|
| Ex.No.: 2 |            | DATA MANIPULATIONS |
| Date:     | 08/08/2024 |                    |

a) Find out the employee id, names, salaries of all the employees `select`

`Employee_id, First_Name, Salary from EMPLOYEES;`

| EMPLOYEE_ID | FIRST_NAME | SALARY |
|-------------|------------|--------|
| 1           | Justin     | 4900   |
| 2           | Emma       | 5500   |
| 3           | Robert     | 9000   |
| 4           | Scarlett   | 8000   |
| 5           | Chris      | 7500   |
| 6           | Mark       | 7200   |
| 7           | Chris      | 7800   |
| 8           | Jeremy     | 3800   |
| 9           | Tom        | 6000   |

b) List out the employees who works under manager 100

`select First_Name || ' ' || Last_Name as name from EMPLOYEES where manager_id =100;`

| NAME          |
|---------------|
| Cate Austin   |
| Justin Beiber |

2 rows returned in 0.04 seconds [Download](#)

c) Find the names of the employees who have a salary greater than or equal to 4800

`select First_Name || ' ' || Last_Name as name from EMPLOYEES  
Where salary >= 4800;`

| NAME            |
|-----------------|
| Emma Stone      |
| Brie Larson     |
| Elizabeth Olsen |
| Cate Austin     |
| Robert Downey   |
| Karen Gillan    |
| Sebastian Stan  |
| Karl Austin     |
| Chris Evans     |

d) List out the employees whose last name is \_AUSTIN

```
select First_Name || ' ' || Last_Name as name from EMPLOYEES  
where Last_Name = 'Austin';
```

| NAME            |
|-----------------|
| Cate Austin     |
| Karl Austin     |
| Jeremy Austin   |
| Chris Austin    |
| Zoe Austin      |
| Scarlett Austin |

6 rows returned in 0.00 seconds [Download](#)

e) Find the names of the employees who works in departments 60,70 and 80

```
select First_Name || ' ' || Last_Name as name from EMPLOYEES  
where Department_id in (60,70,80);
```

| NAME             |
|------------------|
| Chadwick Boseman |
| Jeremy Austin    |
| Tessa Thompson   |
| Zoe Austin       |
| Pom Klementieff  |

5 rows returned in 0.01 seconds [Download](#)

f) Display the unique Manager\_Id.

```
select DISTINCT(manager_id) from EMPLOYEES;
```

| MANAGER_ID |
|------------|
| 400        |
| 200        |
| 350        |
| 300        |
| 250        |
| 450        |
| 600        |
| 550        |
| 900        |
| 800        |

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds [Download](#)

(a) Insert Five Records and calculate GrossPay and NetPay.

```
INSERT INTO Emp (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)  
VALUES (  
101, 'John Doe', 'Manager', 50000, 15000, 20000, 6000,0,0 ,
```

```

102, 'Jane Smith', 'Developer', 40000, 12000, 16000, 4800,0,0 ,
103, 'Alice Johnson', 'Analyst', 35000, 10500, 14000, 4200,0,0 ,
104, 'Bob Brown', 'Designer', 30000, 9000, 12000, 3600,0,0 ,
105, 'Charlie Davis', 'Tester', 25000, 7500, 10000, 3000,0,0
)

```

```

update emp
set GrossPay = Basic+DA+HRA
where Grosspay = 0;

```

```

update emp
set NetPay = Grosspay - PF
where Netpay = 0;

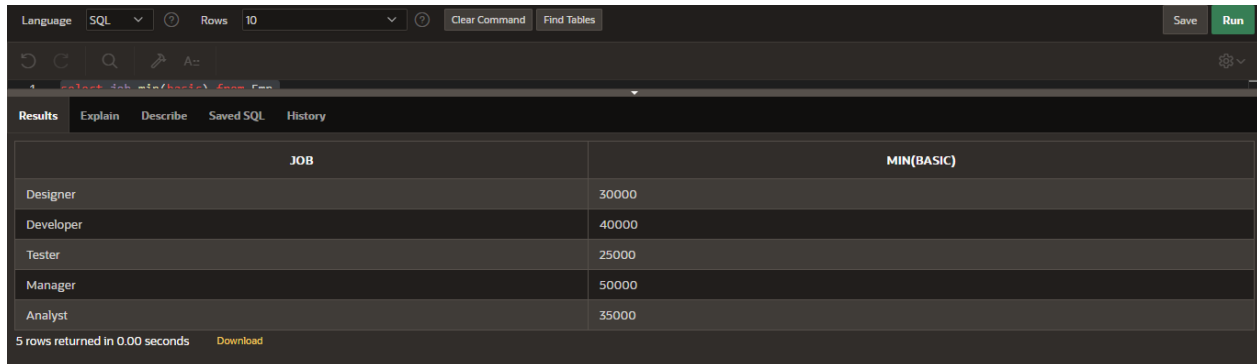
```

(b) Display the employees whose Basic is lowest in each department.

```

select job,min(basic) from Emp
group by Job;

```



| JOB       | MIN(BASIC) |
|-----------|------------|
| Designer  | 30000      |
| Developer | 40000      |
| Tester    | 25000      |
| Manager   | 50000      |
| Analyst   | 35000      |

5 rows returned in 0.00 seconds [Download](#)

1. Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

Create table DEPT(



```

ID Number(7),
Name varchar(25)
);

```

```
Desc DEPT;
```

Results

Explain

Describe

Saved SQL

History

Object Type

TABLE

Object

DEPT

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|--------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| DEPT  | ID     | NUMBER    | -      | 7         | 0     | -           |          | -       | -       |
|       | NAME   | VARCHAR2  | 25     | -         | -     | -           |          | -       | -       |

2) Create the EMP1 table based on the following instance chart. Confirm that the table is created.

```

create table EMP1(
  ID Number(7),
  First_name varchar(25),
  Last_name varchar(25),
  Dept_id Number(7)
);

```

```
Desc EMP1;
```

Results

Explain

Describe

Saved SQL

History

Object Type

TABLE

Object

EMP1

| Table | Column     | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| EMP1  | ID         | NUMBER    | -      | 7         | 0     | -           |          | -       | -       |
|       | FIRST_NAME | VARCHAR2  | 25     | -         | -     | -           |          | -       | -       |
|       | LAST_NAME  | VARCHAR2  | 25     | -         | -     | -           |          | -       | -       |
|       | DEPT_ID    | NUMBER    | -      | 7         | 0     | -           |          | -       | -       |

3) Modify the EMP1 table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

```

ALTER TABLE EMP1
modify Last_name varchar(50);

```

ResultsExplainDescribeSaved SQLHistory

Object TypeTABLEObjectEMP1

| Table | Column     | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| EMP1  | ID         | NUMBER    | -      | 7         | 0     | -           | ✓        | -       | -       |
|       | FIRST_NAME | VARCHAR2  | 25     | -         | -     | -           | ✓        | -       | -       |
|       | LAST_NAME  | VARCHAR2  | 50     | -         | -     | -           | ✓        | -       | -       |
|       | DEPT_ID    | NUMBER    | -      | 7         | 0     | -           | ✓        | -       | -       |

4) Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee\_id, First\_name, Last\_name, Salary and Dept\_id columns. Name the columns Id, First\_name, Last\_name, salary and Dept\_id respectively.

```
create table EMPLOYEES2(
  ID Number(10),
  First_name varchar(50),
  Last_name varchar(50),
  Salary Number(10),
  Dept_id Number(10)
);
```

5) Drop the EMP1 table.

```
drop table EMP1;
```

6) Rename the EMPLOYEES2 table as EMP1.

```
ALTER TABLE EMPLOYEES2 RENAME TO EMP1;
```

7) Add a comment on DEPT and EMP1 tables. Confirm the modification by describing the table.

```
comment on TABLE DEPT IS 'this table contains the fields ID and NAME..';
```

```
SELECT TABLE_NAME, COMMENTS
FROM USER_TAB_COMMENTS
WHERE TABLE_NAME = 'DEPT';
```

Results

Explain

Describe

Saved SQL

History

| TABLE_NAME | COMMENTS                                     |
|------------|--|
| DEPT       | this table contains the fields ID and NAME.. |

1 rows returned in 0.06 seconds [Download](#)

comment on TABLE EMP1 IS 'this table contains the fields ID,first name,last name,salary,DEPT\_id..';

```
SELECT TABLE_NAME, COMMENTS
FROM USER_TAB_COMMENTS
WHERE TABLE_NAME = 'EMP1';
```

| Results Explain Describe Saved SQL History               |   |
|--|---|
| TABLE_NAME   | COMMENTS  |
| EMP1   | this table contains the fields ID,first name,last name,salary,DEPT_id.. |
| 1 rows returned in 0.04 seconds <a href="#">Download</a> |   |

8) Drop the First\_name column from the EMP table and confirm it.

```
ALTER TABLE EMP1
drop column First_name;
```

| Results Explain Describe Saved SQL History |           |             |        |           |       |             |          |         |         |
|--|-----------|-------------|--------|-----------|-------|-------------|----------|---------|---------|
| Object Type                                |           | Object EMP1 |        |           |       |             |          |         |         |
| Table                                      | Column    | Data Type   | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
| EMP1                                       | ID        | NUMBER      | -      | 10        | 0     | -           |          | -       | -       |
|  | LAST_NAME | VARCHAR2    | 50     | -         | -     | -           |          | -       | -       |
|  | SALARY    | NUMBER      | -      | 10        | 0     | -           |          | -       | -       |
|  | DEPT_ID   | NUMBER      | -      | 10        | 0     | -           |          | -       | -       |

|                         |  |  |
|-------------------------|--|--|
| <b>Ex.No.: 3</b>        | <b>WRITING BASIC SQL SELECT STATEMENTS</b> |  |
| <b>Date:</b> 10/08/2024 |  |  |

Find the Solution for the following:

True OR False

1. The following statement executes successfully.

Identify the Errors

```
SELECT employee_id, last_name
sal*12 ANNUAL SALARY
FROM employees;
```

FALSE

The columns in select statement should be separated by commas and the column alias should be given by using a keyword "as"

```
SELECT employee_id, last_name, salary*12 as "ANNUAL SALARY"
FROM employees;
```

| Results     | Explain   | Describe      | Save SQL | History |
|-------------|-----------|---------------|----------|---------|
| EMPLOYEE_ID | LAST_NAME | ANNUAL SALARY |          |         |
| 2           | Stone     | 66000         |          |         |
| 10          | Rudd      | 30000         |          |         |
| 11          | Larson    | 86400         |          |         |
| 20          | Olsen     | 87600         |          |         |
| 25          | Austin    | 116400        |          |         |
| 27          | Gabiblum  | 42000         |          |         |
| 3           | Downey    | 108000        |          |         |
| 18          | Gillan    | 82800         |          |         |
| 21          | Mackie    | 48000         |          |         |
| 22          | Stan      | 108000        |          |         |

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.02 seconds [Download](#)

2) Show the structure of departments the table. Select all the data from it.

Desc employees;

Results Explain Describe Saved SQL History

Object Type TABLE ? Object EMPLOYEES ?

| Table     | Column         | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-----------|----------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| EMPLOYEES | EMPLOYEE_ID    | NUMBER    | -      | 6         | 0     | 1           | -        | -       | -       |
|           | FIRST_NAME     | VARCHAR2  | 20     | -         | -     | -           | ✓        | -       | -       |
|           | LAST_NAME      | VARCHAR2  | 25     | -         | -     | -           | -        | -       | -       |
|           | EMAIL          | VARCHAR2  | 25     | -         | -     | -           | -        | -       | -       |
|           | PHONE_NUMBER   | VARCHAR2  | 20     | -         | -     | -           | ✓        | -       | -       |
|           | HIRE_DATE      | DATE      | 7      | -         | -     | -           | -        | -       | -       |
|           | JOB_ID         | VARCHAR2  | 10     | -         | -     | -           | -        | -       | -       |
|           | SALARY         | NUMBER    | -      | 8         | 2     | -           | ✓        | -       | -       |
|           | COMMISSION_PCT | NUMBER    | -      | 2         | 2     | -           | ✓        | -       | -       |
|           | MANAGER_ID     | NUMBER    | -      | 6         | 0     | -           | ✓        | -       | -       |
|           | DEPARTMENT_ID  | NUMBER    | -      | 4         | 0     | -           | ✓        | -       | -       |

3. Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first.

```
select employee_id , job_id , last_name , hire_date from employees;
```

| Results  | Explain | Describe  | Saved SQL  | History |
|--|---------|-----------|------------|---------|
| EMPLOYEE_ID  | JOB_ID  | LAST_NAME | HIRE_DATE  |         |
| 2  | #es002  | Stone     | 11/06/1990 |         |
| 10   | #pr010  | Rudd      | 04/06/1969 |         |
| 11   | #bl011  | Larson    | 10/01/1989 |         |
| 20   | #eo020  | Olsen     | 02/16/1989 |         |
| 25   | #cb025  | Austin    | 05/14/1969 |         |
| 27   | #lg027  | Goldblum  | 10/22/1992 |         |
| 3  | #rd003  | Downey    | 04/04/1965 |         |
| 18   | #kg018  | Gillan    | 11/28/1987 |         |
| 21   | #am021  | Mackie    | 09/23/1978 |         |
| 22   | #st022  | Stan      | 08/13/1982 |         |
| More than 10 rows available. Increase rows selector to view more rows. |         |           |            |         |
| 10 rows returned in 0.01 seconds <a href="#">Download</a>              |         |           |            |         |

4) Provide an alias STARTDATE for the hire date.

```
select hire_date as "STARTDATE" from employees;
```

| Results  | Explain | Describe | Save SQL | History |
|--|---------|----------|----------|---------|
| STARTDATE  |         |          |          |         |
| 11/06/1990   |         |          |          |         |
| 04/06/1969   |         |          |          |         |
| 10/01/1989   |         |          |          |         |
| 02/16/1989   |         |          |          |         |
| 05/14/1969   |         |          |          |         |
| 10/22/1952   |         |          |          |         |
| 04/04/1965   |         |          |          |         |
| 11/28/1987   |         |          |          |         |
| 09/25/1978   |         |          |          |         |
| 08/15/1982   |         |          |          |         |
| More than 10 rows available. Increase rows selector to view more rows. |         |          |          |         |
| 10 rows returned in 0.04 seconds <a href="#">Download</a>              |         |          |          |         |

5) Create a query to display unique job codes from the employee table.

```
select distinct(job_id) from employees;
```

| Results  | Explain | Describe | Save SQL | History |
|--|---------|----------|----------|---------|
| JOB_ID   |         |          |          |         |
| #cc005   |         |          |          |         |
| #mr006   |         |          |          |         |
| #th024   |         |          |          |         |
| #th009   |         |          |          |         |
| #sa004   |         |          |          |         |
| #tw030   |         |          |          |         |
| #kg018   |         |          |          |         |
| #sa028   |         |          |          |         |
| #p001  |         |          |          |         |
| #ch007   |         |          |          |         |
| More than 10 rows available. Increase rows selector to view more rows. |         |          |          |         |
| 10 rows returned in 0.00 seconds <a href="#">Download</a>              |         |          |          |         |

6) Display the last name concatenated with the job ID , separated by a comma and space, and name the column EMPLOYEE and TITLE.

```
select last_name || ' ' || ',' || ' ' || job_id as "EMPLOYEE AND TITLE" from employees;
```

| Results  | Explain | Describe | Save SQL | History |
|--|---------|----------|----------|---------|
| EMPLOYEE AND TITLE   |         |          |          |         |
| Stone , #es002   |         |          |          |         |
| Rudd , #pr010  |         |          |          |         |
| Larson , #tl011  |         |          |          |         |
| Olsen , #eo020   |         |          |          |         |
| Austin , #cb025  |         |          |          |         |
| Goldblum , #jg027  |         |          |          |         |
| Downey , #nd003  |         |          |          |         |
| Gillan , #kg018  |         |          |          |         |
| Mackie , #am021  |         |          |          |         |
| Stan , #ss022  |         |          |          |         |
| More than 10 rows available. Increase rows selector to view more rows. |         |          |          |         |
| 10 rows returned in 0.00 seconds <a href="#">Download</a>              |         |          |          |         |

7. Create a query to display all the data from the employees table. Separate each column by a comma. Name the column THE\_OUTPUT.

```
select employee_id || ' , ' || first_name || ' , ' || last_name || ' , ' || email || ' , ' || phone_number || ' ,  
' || hire_date || ' , ' || job_id || ' , ' || salary || ' , ' || commission_pct || ' , ' || manager_id || ' , ' ||  
department_id as "THE_OUTPUT"  
from employees;
```

| Results   | Explain | Describe | Saved SQL | History |
|---|---------|----------|-----------|---------|
| THE_OUTPUT  |         |          |           |         |
| 2 , Emma , Stone , emma002@gmail.com , 9840237515 , 11/06/1990 , #es002 , 5500 , 15 , 200 , 15            |         |          |           |         |
| 10 , Paul , Rudd , paul010@gmail.com , 9840237521 , 04/06/1969 , #pr010 , 2500 , 16 , 250 , 30            |         |          |           |         |
| 11 , Brie , Larson , brie011@gmail.com , 9840237522 , 10/01/1989 , #bl011 , 7200 , 18 , 400 , 35          |         |          |           |         |
| 20 , Elizabeth , Olsen , elizabeth020@gmail.com , 9840237531 , 02/16/1989 , #eo020 , 7300 , 12 , 800 , 90 |         |          |           |         |
| 25 , Cate , Austin , cate025@gmail.com , 9840237536 , 05/14/1969 , #ca025 , 9700 , 11 , 100 , 55          |         |          |           |         |
| 27 , Jeff , Goldblum , jeff027@gmail.com , 9840237538 , 10/22/1952 , #jg027 , 3300 , 13 , 200 , 75        |         |          |           |         |
| 3 , Robert , Downey , robert003@gmail.com , 9840237534 , 04/04/1965 , #rd003 , 9000 , 2 , 350 , 40        |         |          |           |         |
| 18 , Karen , Gillan , karen018@gmail.com , 9840237529 , 11/28/1987 , #kg018 , 6900 , 16 , 600 , 95        |         |          |           |         |
| 21 , Anthony , Mackie , anthony021@gmail.com , 9840237532 , 09/25/1978 , #am021 , 4000 , 13 , 850 , 30    |         |          |           |         |
| 22 , Sebastian , Stan , sebastian022@gmail.com , 9840237535 , 08/15/1982 , #ss022 , 9000 , 14 , 550 , 75  |         |          |           |         |
| More than 10 rows available. Increase rows selector to view more rows.                                    |         |          |           |         |
| 10 rows returned in 0.01 seconds <a href="#">Download</a>   |         |          |           |         |

|                  |            |                                 |
|------------------|------------|---------------------------------|
| <b>Ex.No.: 4</b> |            | <b>WORKING WITH CONSTRAINTS</b> |
| <b>Date:</b>     | 16/08/2024 |                                 |

- 1) Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my\_emp\_id\_pk.

```
alter table EMP1
add constraint my_emp_id_pk PRIMARY KEY(ID);
```

- 2) Create a PRIMAY KEY constraint to the DEPT table using the ID colum. The constraint should be named at creation. Name the constraint my\_dept\_id\_pk.

```
alter table DEPT
add constraint my_dept_id_pk PRIMARY KEY(ID);
```

- 3) Add a column DEPT\_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent deparment. Name the constraint my\_emp\_dept\_id\_fk.

```
alter table emp
add DEPT_ID Numbe(10);
```

```
alter table emp
add constraint my_emp_dept_id_fk FOREIGN KEY(DEPT_ID) references dept(ID);
```

- 4) Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

```
alter table emp
add COMMISSION Number(2,2);
```

```
alter table emp
add CONSTRAINT commission_gt_zero CHECK(COMMISSION > 0);
```



|                         |                       |  |
|-------------------------|-----------------------|--|
| <b>Ex.No.: 5</b>        | <b>CREATING VIEWS</b> |  |
| <b>Date:</b> 23/08/2024 |                       |  |

- 1) Create a view called EMPLOYEE\_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.

```
create view EMPLOYEE_VU as
select employee_id , first_name || ' ' || last_name as "EMPLOYEE", department_id
from employees;
```

- 2) Display the contents of the EMPLOYEES\_VU view.

```
select * from EMPLOYEE_VU;
```

| Results     | Explain         | Describe      | Saved SQL | History |
|-------------|-----------------|---------------|-----------|---------|
| EMPLOYEE_ID | EMPLOYEE        | DEPARTMENT_ID |           |         |
| 1           | Justin Bieber   | 10            |           |         |
| 2           | Emma Stone      | 15            |           |         |
| 3           | Robert Downey   | 40            |           |         |
| 4           | Scarlett Austin | 45            |           |         |
| 5           | Chris Evans     | 55            |           |         |
| 6           | Mark Ruffalo    | 40            |           |         |
| 7           | Chris Hemsworth | 65            |           |         |
| 8           | Jeremy Austin   | 70            |           |         |
| 9           | Tom Holland     | 50            |           |         |

- 3) Select the view name and text from the USER\_VIEWS data dictionary views.

```
select VIEW_NAME, TEXT
from USER_VIEWS
where VIEW_NAME = 'EMPLOYEE_VU';
```

| VIEW_NAME  | TEXT  |
|--|---|
| EMPLOYEE_VU  | select employee_id , first_name    ' '    last_name as "EMPLOYEE", department_id from employees |
| 1 rows returned in 0.04 seconds <a href="#">Download</a> |   |

- 4) Using your EMPLOYEES\_VU view, enter a query to display all employees names and Department.

```
SELECT employee, department_id
```

FROM EMPLOYEE\_VU;

| EMPLOYEE        | DEPARTMENT_ID |
|-----------------|---------------|
| Emma Stone      | 15            |
| Paul Rudd       | 30            |
| Brie Larson     | 35            |
| Elizabeth Olsen | 90            |
| Cate Austin     | 55            |
| Jeff Goldblum   | 75            |
| Robert Downey   | 40            |
| Karen Gillan    | 95            |
| Anthony Mackie  | 30            |
| Sebastian Stan  | 75            |

More than 10 rows available. Increase rows selector to view more rows.

- 5) Create a view named DEPT50 that contains the employee number, employee last names and department numbers for all employees in department 50. Label the view columns EMPNO, EMPLOYEE and DEPTNO. Do not allow an employee to be reassigned to another department through the view.

```
CREATE VIEW DEPT50 AS
SELECT employee_id AS EMPNO,
       employee AS EMPLOYEE,
       department_id AS DEPTNO
FROM EMPLOYEE_VU
WHERE department_id = 50
WITH READ ONLY;
```

| EMPNO | EMPLOYEE             | DEPTNO |
|-------|----------------------|--------|
| 9     | Tom Holland          | 50     |
| 18    | Chris Austin         | 50     |
| 23    | Benedict Cumberbatch | 50     |

3 rows returned in 0.01 seconds [Download](#)

- 6) Display the structure and contents of the DEPT50 view.

Desc dept50;

| Results Explain Describe Saved SQL History |          |               |        |           |       |             |          |         |         |
|--|----------|---------------|--------|-----------|-------|-------------|----------|---------|---------|
| Object Type                                |          | Object DEPT50 |        |           |       |             |          |         |         |
| Table                                      | Column   | Data Type     | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
| DEPT50                                     | EMPNO    | NUMBER        | -      | 6         | 0     | -           | -        | -       | -       |
|  | EMPLOYEE | VARCHAR2      | 46     | -         | -     | -           | ✓        | -       | -       |
|  | DEPTNO   | NUMBER        | -      | 4         | 0     | -           | ✓        | -       | -       |

- 7) Attempt to reassign Matos to department 80.

```

UPDATE EMPLOYEES
SET department_id = 80
WHERE first_name = 'Matos';

```

- 8) Create a view called SALARY\_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and JOB\_GRADE tables. Label the column Employee, Department, salary, and Grade respectively.

```

CREATE VIEW SALARY_VU AS
SELECT e.last_name AS Employee,
       d.dept_name AS Department,
       e.salary AS Salary,
       j.grade_level AS Grade
FROM EMPLOYEES e
JOIN DEPARTMENT d
ON e.department_id = d.dept_id
JOIN JOB_GRADE j
ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;

```

| Results  | Explain          | Describe | Save SQL | History |
|--|------------------|----------|----------|---------|
|  |                  |          |          |         |
| EMPLOYEE   | DEPARTMENT       |          | SALARY   | GRADE   |
| Austin   | manager          |          | 6800     | 3       |
| Baustista  | HR               |          | 6500     | 3       |
| Holland  | manager          |          | 6000     | 3       |
| Mackie   | accounts manager |          | 4000     | 2       |
| Goldblum   | HR               |          | 3500     | 2       |
| Goldblum   | HR               |          | 3500     | 4       |
| Rudd   | accounts manager |          | 2500     | 2       |
| Rudd   | accounts manager |          | 2500     | 4       |
| 8 rows returned in 0.00 seconds <a href="#">Download</a> |                  |          |          |         |

|                  |            |                                     |
|------------------|------------|-------------------------------------|
| <b>Ex.No.: 6</b> |            | <b>RESTRICTING AND SORTING DATA</b> |
| <b>Date:</b>     | 29/08/2024 |                                     |

- 1) Create a query to display the last name and salary of employees earning more than 12000.

```
select salary , last_name from employees
where salary > 12000;
```

| SALARY | LAST_NAME |
|--------|-----------|
| 13500  | Austin    |
| 13500  | Austin    |
| 13500  | Austin    |
| 13500  | Austin    |
| 13500  | Austin    |
| 13500  | Austin    |

6 rows returned in 0.01 seconds [Download](#)

- 2) Create a query to display the employee last name and department number for employee number 176.

```
select last_name , department_id from employees
where employee_id = 176;
```

| LAST_NAME | DEPARTMENT_ID |
|-----------|---------------|
| Evans     | 55            |

1 rows returned in 0.00 seconds [Download](#)

- 3) Create a query to display the last name and salary of employees whose salary is not in the range of 5000 and 12000.

```
select last_name , salary from employees
where salary not between 5000 and 12000;
```

| LAST_NAME | SALARY |
|-----------|--------|
| Rudd      | 2500   |
| Austin    | 13500  |
| Goldblum  | 3500   |
| Mackie    | 4000   |
| Austin    | 13500  |
| Beiber    | 4900   |
| Austin    | 13500  |
| Austin    | 13500  |
| Austin    | 13500  |

|             |       |
|-------------|-------|
| Klementieff | 1100  |
| Austin      | 13500 |
| Cooper      | 4500  |

12 rows returned in 0.00 seconds [Download](#)

- 4) Display the employee last name, job ID, and start date of employees hired between February 20,1998 and May 1,1998.order the query in ascending order by start date.(hints: between)

```
select last_name, job_id, hire_date from employees
where hire_date between '02-20-1998' and '05-01-1998';
```

| LAST_NAME | JOB_ID | HIRE_DATE  |
|-----------|--------|------------|
| Evans     | #ce005 | 04/01/1998 |

1 rows returned in 0.00 seconds [Download](#)

- 5) Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.

```
select last_name, department_id from employees
where department_id = 20 or department_id = 50
order by last_name;
```

| LAST_NAME   | DEPARTMENT_ID |
|-------------|---------------|
| Austin      | 50            |
| Cumberbatch | 50            |
| Holland     | 50            |

3 rows returned in 0.04 seconds [Download](#)

- 6) Display the last name and salary of all employees who earn between 5000 and 12000 and are in departments 20 and 50 in alphabetical order by name. Label the columns EMPLOYEE, MONTHLY SALARY respectively.

```
select last_name as "EMPLOYEE" , salary as "MONTHLY SALARY" from employees
where department_id in (20,50) and salary between 5000 and 12000
order by last_name;
```

| EMPLOYEE    | MONTHLY SALARY |
|-------------|----------------|
| Cumberbatch | 8200           |
| Holland     | 6000           |

2 rows returned in 0.04 seconds [Download](#)

- 7) Display the last name and hire date of every employee who was hired in 1994.

```
select last_name, hire_date from employees
```

```
where hire_date like '%1994%';
```

| LAST_NAME  | HIRE_DATE  |
|--|------------|
| Evans  | 05/07/1994 |
| 1 rows returned in 0.00 seconds <a href="#">Download</a> |            |

- 8) Display the last name and job title of all employees who do not have a manager

```
select e.last_name, d.dept_name from employees e
join department d
on e.department_id = d.dept_id
where not(dept_name = 'manager');
```

| LAST_NAME  | DEPT_NAME        |
|--|------------------|
| Rudd   | accounts manager |
| Olson  | ethical hacker   |
| Austin   | data analyst     |
| Goldblum   | HR               |
| Mackie   | accounts manager |
| Stan   | HR               |
| Evans  | data analyst     |
| Bautista   | HR               |
| 8 rows returned in 0.03 seconds <a href="#">Download</a> |                  |

- 9) Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.(hints: is not nul,orderby)

```
select last_name,salary,commission_pct from employees
where commission_pct is not null
order by salary,commission_pct desc;
```

| LAST_NAME   | SALARY | COMMISSION_PCT |
|-------------|--------|----------------|
| Klementieff | 1100   | .1             |
| Rudd        | 2500   | .16            |
| Goldblum    | 3500   | .13            |
| Mackie      | 4000   | .13            |
| Cooper      | 4500   | .13            |
| Beiber      | 4900   | .1             |
| Thompson    | 5200   | .12            |
| Stone       | 5500   | .15            |
| Holland     | 6000   | .13            |
| Bautista    | 6500   | .15            |

- 10) Display the last name of all employees where the third letter of the name is a.

```
select last_name from employees
where last_name like '_a%';
```

| Results  | Explain | Describe | Saved SQL | History |
|--|---------|----------|-----------|---------|
| LAST_NAME  |         |          |           |         |
| Stan   |         |          |           |         |
| Evans  |         |          |           |         |
| charles  |         |          |           |         |
| 3 rows returned in 0.00 seconds <a href="#">Download</a> |         |          |           |         |

11) Display the last name of all employees who have an a and an e in their last name.

```
SELECT last_name FROM employees
WHERE last_name LIKE '%a%' AND last_name LIKE '%e%';
```

| LAST_NAME  |
|--|
| Mackie   |
| Boseman  |
| Cumberbatch  |
| charles  |
| 4 rows returned in 0.00 seconds <a href="#">Download</a> |

12) Display the last name and job and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2500 ,3500 or 7000/.

```
SELECT e.last_name,e.salary,d.dept_name FROM employees e
join department d on e.department_id = d.dept_id
WHERE (dept_name in ('stock clerk','sales representative')) and (salary not
in(2500,3500,7000));
```

| LAST_NAME                       | SALARY                   | DEPT_NAME   |
|---------------------------------|--------------------------|-------------|
| Olsen                           | 7300                     | stock clerk |
| 1 rows returned in 0.01 seconds | <a href="#">Download</a> |             |

|                  |            |                            |
|------------------|------------|----------------------------|
| <b>Ex.No.: 7</b> |            | <b>USING SET OPERATORS</b> |
| <b>Date:</b>     | 30/08/2024 |                            |

- 1) The HR department needs a list of department IDs for departments that do not contain the job ID ST\_CLERK. Use set operators to create this report.

```
select dept_id from department
minus
select department_id from employees
where job_id = 'ST_CLERK';
```

| DEPT_ID |
|---------|
| 55      |
| 90      |

2 rows returned in 0.03 seconds [Download](#)

- 2) The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use set operators to create this report.

```
SELECT c.country_id, c.country_name
FROM countries c
LEFT JOIN department d ON c.country_id = d.country_id
WHERE d.country_id IS NULL;
```

| COUNTRY_ID | COUNTRY_NAME |
|------------|--------------|
| IS         | Iceland      |

1 rows returned in 0.01 seconds [Download](#)

- 3) Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

```
SELECT job_id, department_id
FROM employees
WHERE department_id IN (10, 50, 20)
ORDER BY department_id;
```



| JOB_ID   | DEPARTMENT_ID |
|----------|---------------|
| ST_CLERK | 10            |
| #ca013   | 50            |
| #bc023   | 50            |
| ST_CLERK | 50            |

4 rows returned in 0.01 seconds [Download](#)

- 4) Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

| EMPLOYEE_ID | JOB_ID   |
|-------------|----------|
| 2           | #pr010   |
| 20          | #bl011   |
| 30          | #eo020   |
| 7           | #cb025   |
| 1           | ST_CLERK |

5 rows returned in 0.01 seconds [Download](#)

- 5) The HR department needs a report with the following specifications:

- Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.
- Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them Write a compound query to accomplish this.

```
SELECT last_name, department_id FROM employees
UNION
SELECT dept_name, dept_id FROM department;
```

| LAST_NAME  | DEPARTMENT_ID |
|--|---------------|
| Austin   | 25            |
| Austin   | 45            |
| Austin   | 50            |
| Austin   | 55            |
| Austin   | 60            |
| Austin   | 70            |
| More than 20 rows available. Increase rows selector to view more rows. |               |
| 20 rows returned in 0.00 seconds <a href="#">Download</a>              |               |

|                  |            |                                     |
|------------------|------------|-------------------------------------|
| <b>Ex.No.: 8</b> |            | <b>WORKING WITH MULTIPLE TABLES</b> |
| <b>Date:</b>     | 05/09/2024 |                                     |

- 1) Write a query to display the last name, department number, and department name for all Employees.

```
select e.last_name , e.department_id , d.dept_name
from employees e
join department d on e.department_id = d.dept_id;
```

| LAST_NAME  | DEPARTMENT_ID | DEPT_NAME        |
|------------|---------------|------------------|
| Rudd       | 30            | accounts manager |
| Olsen      | 90            | stock clerk      |
| Austin     | 55            | data analyst     |
| Goldblum   | 75            | HR               |
| Mackie     | 30            | accounts manager |
| Stan       | 75            | HR               |
| Evans      | 55            | data analyst     |
| Boseman    | 70            | HR               |
| Hiddleston | 100           | sales manager    |

- 2) Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

```
select d.dept_name,d.location_id
from department d
join employees e on d.dept_id = e.department_id
where department_id = 80;
```

| DEPT_NAME     | LOCATION_ID |
|---------------|-------------|
| Sales manager | 10          |
| IT support    | 13          |
| admin manager | 16          |
| Sales manager | 10          |
| IT support    | 13          |
| admin manager | 16          |
| Sales manager | 10          |
| IT support    | 13          |
| admin manager | 16          |

9 rows returned in 0.04 seconds [Download](#)

- 3) Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission

```
select e.last_name,d.dept_name,d.location_id,l.city
from (department d
inner join employees e on d.dept_id = e.department_id
inner join location l on d.location_id = l.location_id)
where commission_pct is not null;
```

| LAST_NAME | DEPT_NAME        | LOCATION_ID | CITY       |
|-----------|------------------|-------------|------------|
| Rudd      | accounts manager | 7           | melbourne  |
| Austin    | data analyst     | 10          | Washington |
| Goldblum  | HR               | 4           | New York   |
| Mackie    | accounts manager | 7           | melbourne  |
| Stan      | HR               | 4           | New York   |
| Evans     | data analyst     | 10          | Washington |
| Boseman   | HR               | 2           | Atlanta    |

21 rows returned in 0.01 seconds [Download](#)

- 4) Display the employee last name and department name for all employees who have an a(lowercase) in their last names.

```
select e.last_name,d.dept_name
from department d
inner join employees e on d.dept_id = e.department_id
where last_name like '%a%';
```

| LAST_NAME   | DEPT_NAME        |
|-------------|------------------|
| Mackie      | accounts manager |
| Stan        | HR               |
| Evans       | data analyst     |
| Boseman     | HR               |
| Holland     | manager          |
| Bautista    | HR               |
| Cumberbatch | manager          |
| charles     | Sales manager    |
| charles     | IT support       |

- 5) Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.

```
select e.last_name,d.dept_name,e.department_id
from (department d
inner join employees e on d.dept_id = e.department_id
inner join location l on l.location_id = d.location_id)
where city = 'Toronto';
```

| LAST_NAME   | DEPT_NAME  | DEPARTMENT_ID |
|-------------|------------|---------------|
| Boseman     | HR         | 70            |
| Austin      | HR         | 70            |
| Thompson    | HR         | 70            |
| Klementieff | IT support | 80            |
| roy         | IT support | 80            |
| charles     | IT support | 80            |

6 rows returned in 0.01 seconds [Download](#)

- 6) Display the employee last name and employee number along with their manager's last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, Respectively

```
select last_name as "Employee",employee_id as "Emp#",manager_id as "Mgr#" from
employees;
```

| Employee | Emp# | Mgr# |
|----------|------|------|
| Stone    | 2    | 200  |
| Rudd     | 10   | 250  |
| Larson   | 11   | 400  |
| Olsen    | 20   | 800  |
| Austin   | 25   | 100  |
| Goldblum | 27   | 200  |
| Downey   | 3    | 350  |
| Gillan   | 18   | 600  |
| Mackie   | 21   | 850  |

- 7) Modify lab4\_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

```
SELECT last_name AS "Employee",employee_id AS "Emp#",manager_id AS "Mgr#"
FROM employees ORDER BY employee_id;
```

| Employee  | Emp# | Mgr# |
|-----------|------|------|
| Beiber    | 1    | 100  |
| Stone     | 2    | 200  |
| Downey    | 3    | 350  |
| Austin    | 4    | 300  |
| Ruffalo   | 6    | 250  |
| Hemsworth | 7    | 600  |
| Austin    | 8    | 350  |
| Holland   | 9    | 400  |
| Rudd      | 10   | 250  |

- 8) Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label

```
select e.last_name as "Employee",d.dept_name as "department_name",e.department_id
as "department_no" from employees e
inner join department d on e.department_id = d.dept_id;
```

| Employee   | department_name  | department_no |
|------------|------------------|---------------|
| Rudd       | accounts manager | 30            |
| Olsen      | stock clerk      | 90            |
| Austin     | data analyst     | 55            |
| Goldblum   | HR               | 75            |
| Mackie     | accounts manager | 30            |
| Stan       | HR               | 75            |
| Evans      | data analyst     | 55            |
| Boseman    | HR               | 70            |
| Hiddleston | sales manager    | 100           |

- 9) Show the structure of the JOB\_GRADES table. Create a query that displays the name, job, department name, salary, and grade for all employees

```
desc job_grade;
```

```
SELECT e.first_name || ' ' || last_name AS  
"Employee",d.dept_name,e.salary,g.grade_level as"GRADE"  
FROM (employees e  
inner join department d on e.department_id = d.dept_id  
inner join job_grade g on e.department_id = g.department_id);
```

| Employee        | DEPT_NAME    | SALARY | GRADE |
|-----------------|--------------|--------|-------|
| Elizabeth Olsen | stock clerk  | 7300   | 3     |
| Cate Austin     | data analyst | 13500  | 4     |
| Chris Evans     | data analyst | 7500   | 4     |
| Jeff Goldblum   | HR           | 3500   | 2     |
| Sebastian Stan  | HR           | 9000   | 2     |
| Dave Bautista   | HR           | 6500   | 2     |

6 rows returned in 0.01 seconds   [Download](#)

10) Create a query to display the name and hire date of any employee hired after employee Davies.

```
SELECT last_name,hire_date FROM employees  
where hire_date > '05-03-1986';
```

| LAST_NAME | HIRE_DATE  |
|-----------|------------|
| Stone     | 11/06/1990 |
| Larson    | 10/01/1989 |
| Olsen     | 02/16/1989 |
| Gillan    | 11/28/1987 |
| Evans     | 05/07/1994 |
| Beiber    | 09/21/1996 |
| Holland   | 06/01/1996 |
| roy       | 02/23/1991 |
| charles   | 09/18/1993 |

11) Display the names and hire dates for all employees who were hired before their managers, along with their manager's names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively.

```
SELECT last_name as "employee",hire_date as "employee hired" FROM employees;
```

| employee | employee hired |
|----------|----------------|
| Stone    | 11/06/1990     |
| Rudd     | 04/06/1969     |
| Larson   | 10/01/1989     |
| Olsen    | 02/16/1989     |
| Austin   | 05/14/1969     |
| Goldblum | 10/22/1952     |
| Downey   | 04/04/1965     |
| Gillan   | 11/28/1987     |
| Mackie   | 09/23/1978     |



|                  |            |                    |
|------------------|------------|--------------------|
| <b>Ex.No.: 9</b> |            | <b>SUB QUERIES</b> |
| <b>Date:</b>     | 06/09/2024 |                    |

- 1) The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

```
SELECT last_name, hire_date
FROM employees
WHERE department_id = ALL(
    SELECT department_id
    FROM employees
    WHERE last_name = 'Zlotkey'
)
AND last_name != 'Zlotkey';
```

| LAST_NAME | HIRE_DATE  |
|-----------|------------|
| Doe       | 08/10/1995 |
| Elba      | 09/06/1972 |
| charles   | 09/18/1993 |

3 rows returned in 0.01 seconds [Download](#)

- 2) Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY
FROM employees
WHERE SALARY > (
    SELECT AVG(SALARY)
    FROM employees
)
ORDER BY SALARY ASC;
```

| EMPLOYEE_ID | LAST_NAME | SALARY |
|-------------|-----------|--------|
| 7           | Hemsworth | 7800   |
| 16          | Diesel    | 8000   |
| 12          | Boseman   | 8000   |
| 23          | Carlos    | 8200   |
| 41          | charles   | 8900   |
| 22          | Stan      | 9000   |
| 3           | Downey    | 9000   |
| 8           | Wilson    | 13500  |
| 25          | Austin    | 13500  |

- 3) Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a u.

```
SELECT EMPLOYEE_ID, LAST_NAME
FROM employees
WHERE DEPARTMENT_ID IN (
  SELECT DEPARTMENT_ID
  FROM employees
  WHERE LAST_NAME LIKE '%a%' and LAST_NAME LIKE '%u%');
```

| EMPLOYEE_ID | LAST_NAME |
|-------------|-----------|
| 3           | Downey    |
| 6           | Ruffalo   |
| 30          | Waititi   |
| 27          | Goldblum  |
| 22          | Stan      |
| 17          | Bautista  |
| 25          | Abu       |
| 176         | Morris    |
| 23          | andru     |

9 rows returned in 0.01 seconds [Download](#)

- 4) The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.

```
SELECT e.last_name, e.department_id, e.job_id
```

```

FROM employees e
INNER JOIN department d ON e.department_id = d.dept_id
WHERE e.department_id IN (
    SELECT dept_id
    FROM department
    WHERE location_id = 1700);

```

| LAST_NAME | DEPARTMENT_ID | JOB_ID |
|-----------|---------------|--------|
| Abu       | 55            | #cb025 |
| Morris    | 55            | #ce005 |
| andru     | 55            | #bc023 |

3 rows returned in 0.02 seconds [Download](#)

- 5) Create a report for HR that displays the last name and salary of every employee who reports to King.

```

SELECT e.last_name, e.salary
FROM employees e
WHERE e.manager_id IN (
    SELECT d.manager_id
    FROM department d
    WHERE d.manager_name = 'king');

```

| LAST_NAME  | SALARY |
|------------|--------|
| Zlotkey    | 7200   |
| Hiddleston | 6500   |
| Holland    | 6000   |
| Austin     | 13500  |
| Austen     | 5500   |
| Goldblum   | 3500   |

6 rows returned in 0.01 seconds [Download](#)

- 6) Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

```

SELECT e.department_id, e.last_name, e.job_id
FROM employees e
JOIN department d on e.department_id = d.dept_id
WHERE d.dept_name = 'executive';

```

| DEPARTMENT_ID | LAST_NAME | JOB_ID   |
|---------------|-----------|----------|
| 75            | Goldblum  | ST_CLERK |
| 75            | Stan      | #ss022   |
| 25            | Austin    | #ka028   |
| 75            | Bautista  | #db017   |
| 25            | Diesel    | #vd016   |

5 rows returned in 0.02 seconds [Download](#)

- 7) Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a u.

```

SELECT e.employee_id, e.last_name, e.salary
FROM employees e
WHERE e.salary > (
    SELECT AVG(salary)
    FROM employees
)
AND e.department_id IN (
    SELECT x.department_id
    FROM employees x
    WHERE x.last_name LIKE '%a%' AND x.last_name LIKE '%u%'
);

```

| EMPLOYEE_ID | LAST_NAME | SALARY |
|-------------|-----------|--------|
| 3           | Downey    | 9000   |
| 22          | Stan      | 9000   |
| 25          | Abu       | 13500  |
| 23          | andru     | 8200   |

4 rows returned in 0.01 seconds [Download](#)

|                         |   |
|-------------------------|---|
| <b>Ex.No.: 10</b>       | <b>AGGREGATING DATA USING GROUP FUNCTIONS</b> |
| <b>Date:</b> 12/09/2024 |   |

**Find the Solution for the following:**

Determine the validity of the following three statements. Circle either True or False.

1. Group functions work across many rows to produce one result per group. True/False - **TRUE**

2. Group functions include nulls in calculations. True/False - **FALSE**

3. The WHERE clause restricts rows prior to inclusion in a group calculation. True/False - **FALSE**

4) Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number

```
SELECT ROUND(MAX(salary)) AS Maximum, ROUND(MIN(salary)) AS Minimum,
ROUND(SUM(salary)) AS Sum, ROUND(AVG(salary)) AS Average
FROM employees;
```

| MAXIMUM | MINIMUM | SUM    | AVERAGE |
|---------|---------|--------|---------|
| 15500   | 1100    | 254300 | 7706    |

1 rows returned in 0.02 seconds [Download](#)

5) Modify the above query to display the minimum, maximum, sum, and average salary for each job type.

```
SELECT ROUND(MAX(salary)) AS Maximum, ROUND(MIN(salary)) AS Minimum,
ROUND(SUM(salary)) AS Sum, ROUND(AVG(salary)) AS Average
FROM employees
join department
on department.dept_id = employees.department_id
group by dept_name;
```

| MAXIMUM | MINIMUM | SUM   | AVERAGE |
|---------|---------|-------|---------|
| 4000    | 2500    | 6500  | 3250    |
| 13500   | 13500   | 13500 | 13500   |
| 7800    | 4500    | 12300 | 6150    |
| 13500   | 5200    | 26700 | 8900    |
| 7000    | 1100    | 8100  | 4050    |
| 6500    | 5500    | 12000 | 6000    |
| 13500   | 6000    | 19500 | 9750    |
| 13500   | 13500   | 13500 | 13500   |
| 13500   | 3500    | 40500 | 8100    |

6) Write a query to display the number of people with the same job. Generalize the query so that the user in the HR department is prompted for a job title.

```
SELECT d.dept_name , COUNT(*) AS NumberOfEmployees
FROM Employees e
join department d on e.department_id = d.dept_id
group by d.dept_name;
```

| DEPT_NAME        | NUMBEROFEMPLOYEES |
|------------------|-------------------|
| accounts manager | 2                 |
| IT support       | 1                 |
| admin manager    | 2                 |
| HR               | 3                 |
| stock clerk      | 2                 |
| sales manager    | 2                 |
| manager          | 2                 |
| developer        | 1                 |
| executive        | 5                 |
| data analyst     | 5                 |

10 rows returned in 0.01 seconds [Download](#)

7) Determine the number of managers without listing them. Label the column Number of Managers

```
SELECT COUNT(DISTINCT MANAGER_ID) AS "Number of Managers"
FROM Employees
WHERE MANAGER_ID IS NOT NULL;
```

| Number of Managers                                       |
|--|
| 15   |
| 1 rows returned in 0.01 seconds <a href="#">Download</a> |

8) Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

```
select max(salary) - min(salary) as "DIFFERENCE"
from employees;
```

| DIFFERENCE   |
|--|
| 12400  |
| 1 rows returned in 0.01 seconds <a href="#">Download</a> |

9) Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is \$6,000 or less. Sort the output in descending order of salary.

```
SELECT MANAGER_ID, MIN(SALARY) AS "Lowest Salary"
FROM Employees
WHERE MANAGER_ID IS NOT NULL
GROUP BY MANAGER_ID
HAVING MIN(SALARY) > 6000
ORDER BY "Lowest Salary" DESC;
```

| MANAGER_ID   | Lowest Salary |
|--|---------------|
| 350  | 8000          |
| 150  | 7700          |
| 500  | 7500          |
| 800  | 7300          |
| 600  | 6900          |
| 550  | 6500          |
| 6 rows returned in 0.01 seconds <a href="#">Download</a> |               |

10) Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

```
SELECT EXTRACT(YEAR FROM hire_date) AS "yearly wise employment", COUNT(*)
FROM employees
GROUP BY EXTRACT(YEAR FROM hire_date)
HAVING EXTRACT(YEAR FROM hire_date) IN (1995, 1996, 1997, 1998);
```

| yearly wise employment | COUNT(*) |
|------------------------|----------|
| 1996                   | 2        |
| 1995                   | 1        |

2 rows returned in 0.01 seconds [Download](#)

11) Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

```
select d.dept_name , sum(e.salary)
from employees e
join department d on e.department_id = d.dept_id
where department_id in (20,50,80,90)
group by d.dept_name;
```

| DEPT_NAME   | SUM(E.SALARY) |
|-------------|---------------|
| stock clerk | 8100          |
| manager     | 19500         |

2 rows returned in 0.02 seconds [Download](#)

12) Write a query to display each department's name, location, number of employees, and the average salary for all the employees in that department. Label the column name- Location, Number of people, and salary respectively. Round the average salary to two decimal places.

```
SELECT d.dept_name AS "Name", d.Location_id AS "Location",
COUNT(e.department_id) AS "Number of People", ROUND(AVG(e.Salary), 2) AS
"Salary"
FROM department d
JOIN employees e ON d.dept_id = e.department_id
```



GROUP BY d.dept\_name, d.location\_id;

| Name   | Location | Number of People | Salary  |
|--|----------|------------------|---------|
| sales manager  | 7        | 2                | 6000    |
| data analyst   | 1700     | 3                | 9733.33 |
| stock clerk  | 19       | 2                | 4050    |
| HR   | 2        | 3                | 8900    |
| admin manager  | 16       | 2                | 6150    |
| manager  | 10       | 2                | 9750    |
| accounts manager   | 7        | 2                | 3250    |
| executive  | 4        | 3                | 6333.33 |
| developer  | 1        | 1                | 13500   |
| executive  | 10       | 2                | 10750   |
| More than 10 rows available. Increase rows selector to view more rows. |          |                  |         |
| 10 rows returned in 0.03 seconds <a href="#">Download</a>              |          |                  |         |

|                         |                        |
|-------------------------|------------------------|
| <b>Ex.No.: 11</b>       | <b>PL SQL PROGRAMS</b> |
| <b>Date:</b> 13/09/2024 |                        |

### PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

```

DECLARE
pl_emp_id employees.employee_id%TYPE := 110;
pl_salary employees.salary%TYPE;
pl_incentive NUMBER;
BEGIN
SELECT salary INTO pl_salary
FROM employees
WHERE employee_id = pl_emp_id;

pl_incentive := pl_salary * 0.10;

UPDATE employees
SET incentive = pl_incentive
WHERE employee_id = pl_emp_id;

DBMS_OUTPUT.PUT_LINE('Incentive for employee ID ' || pl_emp_id || ' is ' ||
pl_incentive);

COMMIT;
END;

```

| Results                              | Explain | Describe | Saved SQL | History |
|--------------------------------------|---------|----------|-----------|---------|
| Incentive for employee ID 110 is 820 |         |          |           |         |
| 1 row(s) updated.                    |         |          |           |         |
| 0.00 seconds                         |         |          |           |         |

## PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE
    employeeName VARCHAR2(100);
    "EmployeeID" NUMBER;
BEGIN
    employeeName := 'John Doe';
    "EmployeeID" := 40;

    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employeeName);
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || "EmployeeID");
END;
```

| Results  | Explain | Describe | Saved SQL | History |
|--|---------|----------|-----------|---------|
| Employee Name: John Doe<br>Employee ID: 40<br><br>Statement processed.<br><br>0.01 seconds |         |          |           |         |

### PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

```
DECLARE
  v_employee_id NUMBER := 122;
  v_salary      NUMBER;
  v_new_salary  NUMBER;
  v_increase_percentage NUMBER := 0.40;
BEGIN
  SELECT salary INTO v_salary
  FROM employees
  WHERE employee_id = v_employee_id;

  v_new_salary := v_salary + (v_salary * v_increase_percentage / 100);

  UPDATE employees
  SET salary = v_new_salary
  WHERE employee_id = v_employee_id;

  DBMS_OUTPUT.PUT_LINE('Employee ID ' || v_employee_id || ' new salary: ' ||
v_new_salary);
END;
```

| Results                              | Explain | Describe | Saved SQL | History |
|--------------------------------------|---------|----------|-----------|---------|
| Employee ID 122 new salary: 9036.036 |         |          |           |         |
| 1 row(s) updated.                    |         |          |           |         |
| 0.01 seconds                         |         |          |           |         |

## PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

```
create or replace procedure check_null
is
    value1 number := 10;
    value2 number := null;
begin
    if value1 is not null and value2 is null then
        dbms_output.put_line('Both values are not null!!');
    else
        dbms_output.put_line('Null value found');
    end if;
end;

BEGIN
    check_null;
END;
```

| Results                    | Explain | Describe | Saved SQL | History |
|----------------------------|---------|----------|-----------|---------|
| Both values are not null!! |         |          |           |         |
| Statement processed.       |         |          |           |         |
| 0.00 seconds               |         |          |           |         |

## PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

**declare**

**v\_employeenname employees.first\_name%type;**  
**v\_employeeid NUMBER := 122;**

**begin**

**select first\_name into v\_employeenname**  
**from employees**  
**where first\_name like '%e%' and employee\_id = v\_employeeid;**

**DBMS\_OUTPUT.PUT\_LINE(v\_employeenname);**

**END;**

## PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num\_small variable and large number will store in num\_large variable.

```
declare
ab number :=10;
cd number :=20;
num_small number;
num_large number;
begin
if ab>cd then
num_small :=cd;
num_large :=ab;
else
num_small :=ab;
num_large :=cd;
end if;
dbms_output.put_line('small number = '||num_small);
dbms_output.put_line('large number = '||num_large);
End;
```

```
small number = 10
large number = 20

Statement processed.

0.01 seconds
```

## PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

```
create or replace procedure calculate_incentive(p_emp_id
employees.employee_id%type, p_target number)
is
    v_incentive number(7,2);
    v_salary employees.salary%type;
begin
    select salary into v_salary
    from employees
    where employee_id = p_emp_id;

    if p_target >= 100000 then
        v_incentive := v_salary * 0.1;
        dbms_output.put_line('Incentive of ' || v_incentive || ' calculated for employee ID ' ||
p_emp_id);
    else
        dbms_output.put_line('No incentive for employee ID ' || p_emp_id);
    end if;
End;
```

```
Incentive of 750 calculated for employee ID 176
```

```
Statement processed.
```

```
0.02 seconds
```



## PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
create or replace procedure incentive_sale(p_emp_id employees.employee_id%type,
p_sales number)
is
    v_incentive number(7,2);
begin
    if p_sales > 100000 then
        v_incentive := p_sales * 0.1;
    elsif p_sales between 50000 and 100000 then
        v_incentive := p_sales * 0.05;
    else
        v_incentive := 0;
    end if;

    dbms_output.put_line('Incentive for employee ID ' || p_emp_id || ' is: ' || v_incentive);
End;

begin
    incentive_sale(122,500000);
end;
```

```
Incentive for employee ID 122 is: 50000
```

```
Statement processed.
```

```
0.01 seconds
```

## PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
declare
no_of_emp number;
vacancies number:=45;
begin
select count(*) into no_of_emp from employees where department_id=50;
if no_of_emp<vacancies then
dbms_output.put_line('vacancies are available');
else
dbms_output.put_line('vacancies are not available');
end if;
end;
```

A screenshot of a SQL execution environment showing the output of the PL/SQL program. The text is displayed on a dark background with a light font. The output consists of three lines: 'vacancies are available', 'Statement processed.', and '0.01 seconds'.

vacancies are available

Statement processed.

0.01 seconds

## PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
declare
  v_department_id number := 55;
  v_emp_count number;
  v_vacancies number := 50;
begin
  select count(*) into v_emp_count
  from employees
  where department_id = v_department_id;

  if v_emp_count < v_vacancies then
    dbms_output.put_line('Vacancies available: ' || (v_vacancies - v_emp_count));
  else
    dbms_output.put_line('No vacancies available.');
```

end if;

end;

```
Vacancies available: 47
```

```
Statement processed.
```

```
0.01 seconds
```

## PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```
begin
  for i in (select employee_id, first_name || ' ' || last_name as name, job_id, hire_date,
salary from employees)
    loop
      dbms_output.put_line('ID: ' || i.employee_id || ', Name: ' || i.name || ', Job: ' || i.job_id
|| ', Hire Date: ' || i.hire_date || ', Salary: ' || i.salary);
    end loop;
end;
```

```
ID: 2, Name: Emma Austen, Job: ST_CLERK, Hire Date: 11/06/1990, Salary: 5500
ID: 10, Name: Paul Rudd, Job: #pr010, Hire Date: 04/06/1969, Salary: 2500
ID: 11, Name: Brie Zlotkey, Job: #bl011, Hire Date: 10/01/1989, Salary: 7200
ID: 20, Name: Elizabeth Olsen, Job: #eo020, Hire Date: 02/16/1989, Salary: 7300
ID: 25, Name: Cate Abu, Job: #cb025, Hire Date: 05/14/1969, Salary: 13500
ID: 27, Name: Jeff Goldblum, Job: ST_CLERK, Hire Date: 10/22/1952, Salary: 3500
ID: 122, Name: Robert Downey, Job: #rd003, Hire Date: 04/04/1965, Salary: 9036.04
ID: 18, Name: Karen Gillan, Job: #kg018, Hire Date: 11/28/1987, Salary: 6900
ID: 21, Name: Anthony Mackie, Job: ST_CLERK, Hire Date: 09/23/1978, Salary: 4000
ID: 22, Name: Sebastian Stan, Job: #ss022, Hire Date: 08/13/1982, Salary: 9000
ID: 28, Name: Karl Austin, Job: #ka028, Hire Date: 06/07/1972, Salary: 13500
ID: 176, Name: Chris Morris, Job: #ce005, Hire Date: 05/07/1994, Salary: 7500
ID: 6, Name: Mark Ruffalo, Job: #mr006, Hire Date: 11/22/1967, Salary: 7200
ID: 12, Name: Chadwick Boseman, Job: #cb012, Hire Date: 11/29/1976, Salary: 8000
ID: 24, Name: Tom Hiddleston, Job: #th024, Hire Date: 02/09/1981, Salary: 6500
ID: 1, Name: Justin Beiber, Job: ST_CLERK, Hire Date: 09/21/1996, Salary: 4900
ID: 8, Name: Jeremy Wilson, Job: #ja008, Hire Date: 01/07/1971, Salary: 13500
ID: 7, Name: Chris Hemsworth, Job: #ch007, Hire Date: 08/11/1983, Salary: 7800
ID: 9, Name: Tom Holland, Job: ST_CLERK, Hire Date: 06/01/1996, Salary: 6000
ID: 13, Name: Chris Austin, Job: #ca013, Hire Date: 06/21/1979, Salary: 13500
ID: 17, Name: Dave Bautista, Job: #db017, Hire Date: 01/18/1969, Salary: 6500
ID: 26, Name: Tessa Thompson, Job: ST_CLERK, Hire Date: 10/03/1983, Salary: 5200
ID: 14, Name: Zoe Austin, Job: #za014, Hire Date: 06/19/1978, Salary: 13500
ID: 19, Name: Pom Davies, Job: #pk019, Hire Date: 05/03/1986, Salary: 1100
ID: 42, Name: Matos roy, Job: #mr042, Hire Date: 02/23/1991, Salary: 7000
ID: 4, Name: Scarlett Austin, Job: #sa004, Hire Date: 11/22/1984, Salary: 13500
ID: 15, Name: Bradley Hook, Job: ST_CLERK, Hire Date: 01/05/1975, Salary: 4500
ID: 16, Name: Vin Diesel, Job: #vd016, Hire Date: 07/18/1967, Salary: 8000
ID: 110, Name: Benedict andru, Job: #bc023, Hire Date: 07/19/1976, Salary: 8200
ID: 30, Name: Taika Waititi, Job: #tw030, Hire Date: 08/16/1975, Salary: 7700
ID: 40, Name: John Doe , Job: #jd040 , Hire Date: 08/10/1995, Salary: 6000
ID: 29, Name: Idris Elba, Job: #ie029, Hire Date: 09/06/1972, Salary: 7400
ID: 41, Name: Matos charles, Job: #mc041, Hire Date: 09/18/1993, Salary: 8900
```

Statement processed.

## PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```
begin
  for i in (select e.employee_id, e.first_name || ' ' || e.last_name as name, d.dept_name
            from employees e
            join department d on e.employee_id = d.dept_id) loop
    dbms_output.put_line('ID: ' || i.employee_id || ', Name: ' || i.name || ', Department: ' ||
i.dept_name);
  end loop;
End;
```

```
ID: 25, Name: Cate Abu, Department: executive
ID: 15, Name: Bradley Hook, Department: sales manager
ID: 30, Name: Taika Waititi, Department: accounts manager
```

```
Statement processed.
```

```
0.03 seconds
```

## PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
begin
  for rec in (select e.employee_id, d.dept_name, min(salary) as min_salary from
employees
  e join department d
    on e.employee_id = d.dept_id
  group by e.employee_id , d.dept_name)
  loop
    dbms_output.put_line('Job ID: ' || rec.employee_id || ', Title: ' || rec.dept_name || ',
Min Salary: ' || rec.min_salary);
  end loop;
End;
```

```
Job ID: 30, Title: accounts manager, Min Salary: 7700
Job ID: 25, Title: executive, Min Salary: 13500
Job ID: 15, Title: sales manager, Min Salary: 4500

Statement processed.

0.05 seconds
```

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
begin
  for rec in (select e.employee_id, d.dept_name, min(salary) as min_salary from
employees
  e join department d
    on e.employee_id = d.dept_id
  group by e.employee_id , d.dept_name)
  loop
    dbms_output.put_line('Job ID: ' || rec.employee_id || ', Title: ' || rec.dept_name || ',
Min Salary: ' || rec.min_salary);
  end loop;
End;
```

```
Job ID: 30, Title: accounts manager, Min Salary: 7700
Job ID: 25, Title: executive, Min Salary: 13500
Job ID: 15, Title: sales manager, Min Salary: 4500
```

```
Statement processed.
```

```
0.05 seconds
```

## PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all Employees.

Begin

```
for rec in (select employee_id, first_name || ' ' || last_name as name, hire_date
            from employees) loop
    dbms_output.put_line('ID: ' || rec.employee_id || ', Name: ' || rec.name || ', Start Date: '
|| rec.hire_date);
end loop;
end;
```

```
ID: 2, Name: Emma Austen, Start Date: 11/06/1990
ID: 10, Name: Paul Rudd, Start Date: 04/06/1969
ID: 11, Name: Brie Zlotkey, Start Date: 10/01/1989
ID: 20, Name: Elizabeth Olsen, Start Date: 02/16/1989
ID: 25, Name: Gate Abu, Start Date: 05/14/1969
ID: 27, Name: Jeff Goldblum, Start Date: 10/22/1952
ID: 122, Name: Robert Downey, Start Date: 04/04/1965
ID: 18, Name: Karen Gillan, Start Date: 11/28/1987
ID: 21, Name: Anthony Mackie, Start Date: 09/23/1978
ID: 22, Name: Sebastian Stan, Start Date: 08/13/1982
ID: 28, Name: Karl Austin, Start Date: 06/07/1972
ID: 176, Name: Chris Morris, Start Date: 05/07/1994
ID: 6, Name: Mark Ruffalo, Start Date: 11/22/1967
ID: 12, Name: Chadwick Boseman, Start Date: 11/20/1976
ID: 24, Name: Tom Hiddleston, Start Date: 02/09/1981
ID: 1, Name: Justin Beiber, Start Date: 09/21/1996
ID: 8, Name: Jeremy Wilson, Start Date: 01/07/1971
ID: 7, Name: Chris Hemsworth, Start Date: 08/11/1983
ID: 9, Name: Tom Holland, Start Date: 06/01/1996
ID: 13, Name: Chris Austin, Start Date: 06/21/1979
ID: 17, Name: Dave Bautista, Start Date: 01/18/1969
ID: 26, Name: Tessa Thompson, Start Date: 10/03/1983
ID: 14, Name: Zoe Austin, Start Date: 06/19/1978
ID: 19, Name: Pun Davies, Start Date: 05/03/1988
ID: 42, Name: Natos roy, Start Date: 02/23/1901
ID: 4, Name: Scarlett Austin, Start Date: 11/22/1984
ID: 15, Name: Bradley Hook, Start Date: 01/05/1975
ID: 16, Name: Vin Diesel, Start Date: 07/18/1967
ID: 110, Name: Benedict andru, Start Date: 07/19/1976
ID: 30, Name: Talika Waititi, Start Date: 08/16/1975
ID: 40, Name: John Doe , Start Date: 08/10/1995
ID: 29, Name: Idris Elba, Start Date: 09/06/1972
```



## PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

**BEGIN**

**FOR** rec **IN** (SELECT employee\_id, first\_name || ' ' || last\_name AS name, end\_date  
FROM employees)

**LOOP**

dbms\_output.put\_line('ID: ' || rec.employee\_id ||  
' , Name: ' || rec.name ||  
' , End Date: ' ||  
NVL(TO\_CHAR(rec.end\_date, 'YYYY-MM-DD'), 'Still Active'));

**END LOOP;**

**END;**

```
ID: 2, Name: Emma Austen, End Date: Still Active
ID: 10, Name: Paul Rudd, End Date: Still Active
ID: 11, Name: Brie Zlotkey, End Date: Still Active
ID: 20, Name: Elizabeth Olsen, End Date: Still Active
ID: 25, Name: Cate Abuy, End Date: Still Active
ID: 27, Name: Jeff Goldblum, End Date: Still Active
ID: 122, Name: Robert Downey, End Date: Still Active
ID: 18, Name: Karen Gillan, End Date: Still Active
ID: 21, Name: Anthony Mackie, End Date: Still Active
ID: 22, Name: Sebastian Stan, End Date: Still Active
ID: 28, Name: Karl Austin, End Date: Still Active
ID: 176, Name: Chris Morris, End Date: Still Active
ID: 6, Name: Mark Ruffalo, End Date: Still Active
ID: 12, Name: Chadwick Boseman, End Date: Still Active
ID: 24, Name: Tom Hiddleston, End Date: Still Active
ID: 1, Name: Justin Beiber, End Date: Still Active
ID: 8, Name: Jeremy Wilson, End Date: Still Active
ID: 7, Name: Chris Hemsworth, End Date: Still Active
ID: 9, Name: Tom Holland, End Date: Still Active
ID: 13, Name: Chris Austin, End Date: Still Active
ID: 17, Name: Dave Bautista, End Date: Still Active
ID: 26, Name: Yessa Thompson, End Date: Still Active
ID: 14, Name: Zoe Austin, End Date: Still Active
ID: 19, Name: Pom Davies, End Date: Still Active
ID: 42, Name: Matos roy, End Date: Still Active
ID: 4, Name: Scarlett Austin, End Date: Still Active
ID: 15, Name: Bradley Hook, End Date: Still Active
ID: 16, Name: Vin Diesel, End Date: Still Active
ID: 110, Name: Benedict andru, End Date: Still Active
ID: 30, Name: Taika Waititi, End Date: Still Active
ID: 40, Name: John Doe , End Date: Still Active
ID: 29, Name: Idris Elba, End Date: Still Active
```

|                  |                 |
|------------------|-----------------|
| Ex.No.: 12       | PL SQL PROGRAMS |
| Date: 19/09/2024 |                 |

## Program 1

### FACTORIAL OF A NUMBER USING FUNCTION

```

DECLARE
  n NUMBER := 10;
  result NUMBER;

  FUNCTION itfact(num NUMBER) RETURN NUMBER IS
    fact NUMBER := 1;
  BEGIN
    FOR i IN 1..num LOOP
      fact := fact * i;
    END LOOP;
    RETURN fact;
  END;

BEGIN
  result := itfact(n);
  DBMS_OUTPUT.PUT_LINE('The factorial of ' || n || ' is ' || result);
END;

```

| Results   | Explain | Describe | Saved SQL | History |
|---|---------|----------|-----------|---------|
| <p>The factorial of 10 is 3628800</p> <p>Statement processed.</p> <p>0.01 seconds</p> |         |          |           |         |

## Program 2

Write a PL/SQL program using Procedures IN,INOUT,OUT parameters to retrieve the corresponding book information in library

```
CREATE OR REPLACE PROCEDURE book_info(  
    p_book_id IN NUMBER,  
    p_author OUT VARCHAR2,  
    p_title OUT VARCHAR2,  
    p_published_date OUT DATE  
) AS  
BEGIN  
    SELECT author, title, published_date  
    INTO p_author, p_title, p_published_date  
    FROM books  
    WHERE book_id = p_book_id;  
  
EXCEPTION  
    WHEN NO_DATA_FOUND THEN  
        p_author := NULL;  
        p_title := NULL;  
        p_published_date := NULL;  
    WHEN OTHERS THEN  
        RAISE;  
END book_info;  
  
DECLARE  
    v_author VARCHAR2(100);  
    v_title VARCHAR2(100);  
    v_published_date DATE;  
    v_book_id NUMBER := 1;  
BEGIN  
    book_info(v_book_id, v_author, v_title, v_published_date);  
  
    IF v_author IS NOT NULL THEN  
        DBMS_OUTPUT.PUT_LINE('Book ID: ' || v_book_id);  
        DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);  
        DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);  
        DBMS_OUTPUT.PUT_LINE('Published Date: ' || TO_CHAR(v_published_date, 'YYYY-  
MM-DD'));  
    ELSE  
        DBMS_OUTPUT.PUT_LINE('No book found with ID: ' || v_book_id);  
    END IF;  
END;
```

Book ID: 1  
Author: William Shaespeare  
Title: Hamlet  
Published Date: 1590-12-12

Statement processed.

0.02 seconds

|            |            |                       |
|------------|------------|-----------------------|
| Ex.No.: 13 |            | WORKING WITH TRIGGERS |
| Date:      | 20/09/2024 |                       |

### Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```

CREATE OR REPLACE TRIGGER prevent_parent_deletion
BEFORE DELETE ON employees
FOR EACH ROW
DECLARE
    pl_dept_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO pl_dept_count
    FROM department
    WHERE dept_id = :OLD.employee_id;
    IF pl_dept_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete employee record as
department records exist. ');
    END IF;
END;

DELETE FROM employees
WHERE employee_id = 70;

```

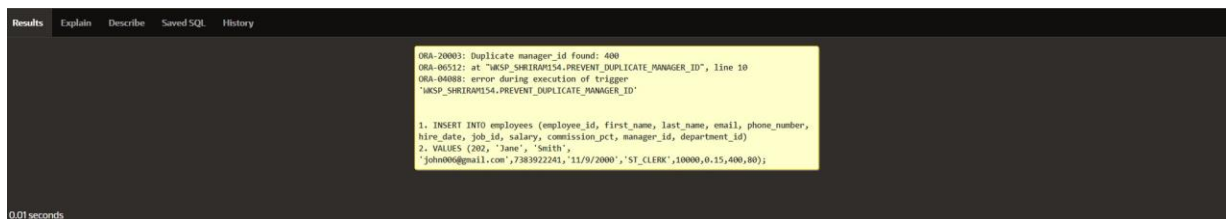
| Results   | Explain | Describe | Saved SQL | History |
|---|---------|----------|-----------|---------|
| <pre> ORA-20001: Cannot delete employee record as department records exist. ORA-06512: at "WKSP_SHRIRAM154.PREVENT_PARENT_DELETION", line 9 ORA-04088: error during execution of trigger "WKSP_SHRIRAM154.PREVENT_PARENT_DELETION" </pre> |         |          |           |         |
| 0.02 seconds  |         |          |           |         |

## Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE OR REPLACE TRIGGER prevent_duplicate_manager_id
BEFORE INSERT OR UPDATE ON employees
FOR EACH ROW
DECLARE
    pl_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO pl_count
    FROM employees
    WHERE manager_id = :NEW.manager_id
    AND employee_id != :NEW.employee_id;
    IF pl_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Duplicate manager_id found: ' ||
:NEW.manager_id);
    END IF;
END;
```

```
INSERT INTO employees (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, commission_pct, manager_id, department_id)
VALUES (202, 'Jane', 'Smith',
'john006@gmail.com',7383922241,'11/9/2000','ST_CLERK',10000,0.15,400,80);
```

A screenshot of the SQL Developer interface. The top menu bar includes 'Results', 'Explain', 'Describe', 'Save SQL', and 'History'. The main window displays an error message in a yellow box: 'ORA-20003: Duplicate manager\_id found: 400', 'ORA-00512: at "WKSP\_SHRIRAM154.PREVENT\_DUPLICATE\_MANAGER\_ID", line 10', 'ORA-04088: error during execution of trigger', and 'WKSP\_SHRIRAM154.PREVENT\_DUPLICATE\_MANAGER\_ID'. Below the error message, the SQL statement being executed is shown: '1. INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id) 2. VALUES (202, 'Jane', 'Smith', 'john006@gmail.com',7383922241,'11/9/2000','ST\_CLERK',10000,0.15,400,80);'. The bottom left corner of the window shows '0.01 seconds'.

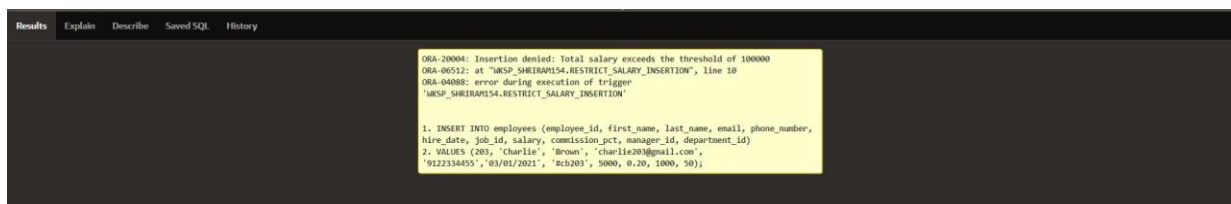
### Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
CREATE OR REPLACE TRIGGER restrict_salary_insertion
BEFORE INSERT ON employees
FOR EACH ROW
DECLARE
    total_salary NUMBER;
    threshold NUMBER := 100000;
BEGIN

    SELECT SUM(salary)
    INTO total_salary
    FROM employees;
    IF (total_salary + :NEW.salary) > threshold THEN
        RAISE_APPLICATION_ERROR(-20004, 'Insertion denied: Total salary exceeds the
threshold of ' || threshold);
    END IF;
END;
```

```
INSERT INTO employees (employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary, commission_pct, manager_id, department_id)
VALUES (203, 'Charlie', 'Brown', 'charlie203@gmail.com', '9122334455', '03/01/2021',
'#cb203', 5000, 0.20, 1000, 50);
```

A screenshot of the SQL Developer interface. The top bar shows tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The main window displays an error message in a yellow box. The error message reads: 'ORA-20004: Insertion denied: Total salary exceeds the threshold of 100000' followed by 'ORA-00512: at "MSP\_SHRIMP154.RESTRICT\_SALARY\_INSERTION", line 10' and 'ORA-04088: error during execution of trigger "MSP\_SHRIMP154.RESTRICT\_SALARY\_INSERTION"'. Below the error message, the SQL statement being executed is shown: '1. INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id) 2. VALUES (203, 'Charlie', 'Brown', 'charlie203@gmail.com', '9122334455', '03/01/2021', '#cb203', 5000, 0.20, 1000, 50);'.

## PROGRAM 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE OR REPLACE TRIGGER audit_changes
AFTER UPDATE OF salary, job_id ON employees
FOR EACH ROW
BEGIN
    IF :OLD.salary != :NEW.salary OR :OLD.job_id != :NEW.job_id THEN
        INSERT INTO employee_audit (
            employee_id,
            old_salary,
            new_salary,
            old_job_title,
            new_job_title,
            change_timestamp,
            changed_by
        ) VALUES (
            :OLD.employee_id,
            :OLD.salary,
            :NEW.salary,
            :OLD.job_id,
            :NEW.job_id,
            SYSTIMESTAMP,
            USER
        );
    END IF;
END;
```

```
UPDATE employees
SET salary = 55000, job_id = 'ST_CLERK'
WHERE employee_id = 176;
```

```
SELECT * FROM employee_audit;
```

| AUDIT_ID | EMPLOYEE_ID | OLD_SALARY | NEW_SALARY | OLD_JOB_ID       | NEW_JOB_ID      | CHANGE_TIMESTAMP             | CHANGED_BY       |
|----------|-------------|------------|------------|------------------|-----------------|------------------------------|------------------|
| 1        | 20          | 50000      | 55000      | manager          | manager         | 15-OCT-24 10:00:00.000000 AM | admin            |
| 2        | 122         | 60000      | 65000      | Manager          | Manager         | 15-OCT-24 10:15:00.000000 AM | admin            |
| 5        | 27          | 45000      | 47000      | Analyst          | Senior Analyst  | 15-OCT-24 10:30:00.000000 AM | user1            |
| 22       | 176         | 7500       | 55000      | #ce005           | ST_CLERK        | 16-OCT-24 04:25:06.252580 PM | APEX_PUBLIC_USER |
| 3        | 9           | 70000      | 75000      | Senior Developer | Lead Developer  | 15-OCT-24 10:45:00.000000 AM | user2            |
| 4        | 4           | 80000      | 85000      | Team Lead        | Project Manager | 15-OCT-24 11:00:00.000000 AM | admin            |

6 rows returned in 0.00 seconds [Download](#)



## PROGRAM 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
CREATE OR REPLACE TRIGGER trg_audit_employees
AFTER INSERT OR UPDATE OR DELETE ON employees
FOR EACH ROW
DECLARE
    v_old_values CLOB;
    v_new_values CLOB;
BEGIN
    IF INSERTING THEN
        v_old_values := NULL;
        v_new_values := 'employee_id: ' || :NEW.employee_id || ', ' ||
            'first_name: ' || :NEW.first_name || ', ' ||
            'salary: ' || :NEW.salary;

        INSERT INTO audit_log (action, table_name, record_id, changed_by, new_values)
        VALUES ('INSERT', 'employees', :NEW.employee_id, USER, v_new_values);

    ELSIF UPDATING THEN
        v_old_values := 'employee_id: ' || :OLD.employee_id || ', ' ||
            'first_name: ' || :OLD.first_name || ', ' ||
            'salary: ' || :OLD.salary;
        v_new_values := 'employee_id: ' || :NEW.employee_id || ', ' ||
            'first_name: ' || :NEW.first_name || ', ' ||
            'salary: ' || :NEW.salary;

        INSERT INTO audit_log (action, table_name, record_id, changed_by, old_values,
new_values)
        VALUES ('UPDATE', 'employees', :NEW.employee_id, USER, v_old_values,
v_new_values);

    ELSIF DELETING THEN
        v_old_values := 'employee_id: ' || :OLD.employee_id || ', ' ||
            'first_name: ' || :OLD.first_name || ', ' ||
            'salary: ' || :OLD.salary;
        v_new_values := NULL;

        INSERT INTO audit_log (action, table_name, record_id, changed_by, old_values)
        VALUES ('DELETE', 'employees', :OLD.employee_id, USER, v_old_values);
    END IF;
END trg_audit_employees;
```

```
INSERT INTO employees (employee_id, first_name, salary)
VALUES (3, 'Ball', 50000);
```

| Results            | Explain | Describe | Saved SQL | History |
|--------------------|---------|----------|-----------|---------|
| 1 row(s) inserted. |         |          |           |         |
| 0.12 seconds       |         |          |           |         |

```
UPDATE employees
SET salary = 55000
WHERE employee_id = 3;
```

|                   |  |  |  |  |
|-------------------|--|--|--|--|
| 1 row(s) updated. |  |  |  |  |
| 0.06 seconds      |  |  |  |  |

```
DELETE FROM employees
WHERE employee_id = 3;
```

```
SELECT * FROM audit_log;
```

| AUDIT_ID | ACTION | TABLE_NAME | RECORD_ID | CHANGED_BY       | CHANGE_TIMESTAMP             | OLD_VALUES                                      | NEW_VALUES                                      |
|----------|--------|------------|-----------|------------------|------------------------------|---|---|
| 1        | INSERT | employees  | 3         | APEX_PUBLIC_USER | 16-OCT-24 04:39:7957308 PM   | -   | employee_id: 3, first_name: Ball, salary: 50000 |
| 3        | DELETE | employees  | 3         | APEX_PUBLIC_USER | 16-OCT-24 04:41:49.077471 PM | employee_id: 3, first_name: Ball, salary: 55000 | -   |
| 2        | UPDATE | employees  | 3         | APEX_PUBLIC_USER | 16-OCT-24 04:40:03:193035 PM | employee_id: 3, first_name: Ball, salary: 50000 | employee_id: 3, first_name: Ball, salary: 55000 |

3 rows returned in 0.00 seconds [Download](#)

## PROGRAM 6

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE TABLE transactions (  
    transaction_id NUMBER PRIMARY KEY,  
    amount NUMBER,  
    running_total NUMBER  
);  
  
CREATE OR REPLACE TRIGGER update_running_total  
FOR INSERT ON transactions  
COMPOUND TRIGGER  
  
    TYPE amount_array IS TABLE OF NUMBER INDEX BY PLS_INTEGER;  
    new_amounts amount_array;  
  
    BEFORE EACH ROW IS  
    BEGIN  
        new_amounts(:NEW.transaction_id) := :NEW.amount;  
    END BEFORE EACH ROW;  
  
    AFTER STATEMENT IS  
    BEGIN  
        DECLARE  
            v_total NUMBER;  
        BEGIN  
            SELECT NVL(MAX(running_total), 0)  
            INTO v_total  
            FROM transactions;  
  
            FOR i IN new_amounts.FIRST .. new_amounts.LAST LOOP  
                v_total := v_total + new_amounts(i);  
                UPDATE transactions  
                SET running_total = v_total  
                WHERE transaction_id = i;  
            END LOOP;  
        END;  
    END AFTER STATEMENT;  
  
END update_running_total;  
  
INSERT INTO transactions (transaction_id, amount)
```

VALUES (1, 10000);

INSERT INTO transactions (transaction\_id, amount)  
VALUES (2, 20000);

| Results        |  |        | Explain       | Describe | Saved SQL | History |
|----------------|--|--------|---------------|----------|-----------|---------|
| TRANSACTION_ID |  | AMOUNT | RUNNING_TOTAL |          |           |         |
| 1              |  | 10000  | 10000         |          |           |         |
| 2              |  | 20000  | 30000         |          |           |         |

2 rows returned in 0.01 seconds

Download

## PROGRAM 7

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE TABLE inventory (  
    item_id NUMBER PRIMARY KEY,  
    item_name VARCHAR2(100),  
    stock_level NUMBER  
);
```

```
CREATE TABLE orders (  
    order_id NUMBER PRIMARY KEY,  
    item_id NUMBER,  
    quantity NUMBER,  
    order_status VARCHAR2(20),  
    CONSTRAINT fk_item FOREIGN KEY (item_id) REFERENCES inventory(item_id)  
);
```

```
CREATE OR REPLACE TRIGGER validate_stock_before_order  
BEFORE INSERT ON orders  
FOR EACH ROW  
DECLARE  
    v_stock_level NUMBER;  
    v_pending_orders NUMBER;  
BEGIN  
    SELECT stock_level  
    INTO v_stock_level  
    FROM inventory  
    WHERE item_id = :NEW.item_id;  
    SELECT NVL(SUM(quantity), 0)  
    INTO v_pending_orders  
    FROM orders  
    WHERE item_id = :NEW.item_id  
    AND order_status = 'Pending';  
    IF (:NEW.quantity + v_pending_orders) > v_stock_level THEN  
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient stock for item: ' || :NEW.item_id);  
    END IF;  
END;
```

INSERT INTO orders (order\_id, item\_id, quantity, order\_status)  
VALUES (1, 101, 5, 'Pending');

```
1 row(s) inserted.  
  
0.03 seconds
```

INSERT INTO orders (order\_id, item\_id, quantity, order\_status)  
VALUES (2, 103, 20, 'Pending');

| ITEM_ID  | ITEM_NAME | STOCK_LEVEL |
|--|-----------|-------------|
| 101  | hp_laptop | 10          |
| 102  | keyboard  | 20          |
| 103  | mouse     | 15          |
| rows returned in 0.01 seconds <a href="#">Download</a> |           |             |

| ORDER_ID   | ITEM_ID | QUANTITY | ORDER_STATUS |
|--|---------|----------|--------------|
| 1  | 101     | 5        | Pending      |
| 1 rows returned in 0.01 seconds <a href="#">Download</a> |         |          |              |

|            |            |          |
|------------|------------|----------|
| Ex.No.: 14 |            | MONGO DB |
| Date:      | 26/09/2024 |          |

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

```
db.restaurants.find(
{
  $or: [
    { cuisine: { $nin: ["American", "Chinees"] } },
    { name: { $regex: /^Wil/i } }
  ]
},
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
}
);
```

```
>_MONGOSH
< {
  borough: 'Bronx',
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
{
  borough: 'Bronx',
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: 30075445
}
{
  borough: 'Bronx',
  cuisine: 'Italian',
  name: 'Pasta Palace',
  restaurant_id: 30075446
}
{
  borough: 'Manhattan',
  cuisine: 'Chinese',
  name: 'Dragon Wok',
  restaurant_id: 30075447
}
```

2. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates..

```
db.restaurants.find(
{
  grades: {
    $elemMatch: {
      grade: "A",
      score: 11
    }
  }
},
{
  restaurant_id: 1,
  name: 1,
  grades: 1,
  _id: 0
}
);
```

```
< {
  grades: [
    {
      date: 2014-03-03T00:00:00.003Z,
      grade: 'A',
      score: 3
    },
    {
      date: 2013-09-11T00:00:00.003Z,
      grade: 'A',
      score: 7
    },
    {
      date: 2013-01-24T00:00:00.003Z,
      grade: 'A',
      score: 11
    },
    {
      date: 2011-11-23T00:00:00.003Z,
      grade: 'A',
      score: 5
    },
    {
      date: 2011-03-10T00:00:00.003Z,
      grade: 'B',
      score: 13
    }
  ],
}
```



3. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".

```
db.restaurants.find(
{
  "grades.1": {
    $elemMatch: {
      grade: "A",
      score: 9
    }
  }
},
{
  restaurant_id: 1,
  name: 1,
  grades: 1,
  _id: 0
});
```

4. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..

```
db.restaurants.find(
{
  "address.coord.1": { $gt: 42, $lte: 52 }
},
{
  restaurant_id: 1,
  name: 1,
  address: 1,
  _id: 0
});
```

5. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

```
db.restaurants.find().sort({ name: 1 });
```

**SAMPLE OUTPUT:-**

```
{
  _id: ObjectId('671b5e6d56ec9972ca8f5dc4'),
  address: {
    building: 5566,
    coord: [
      -73.867377,
      40.854047
    ],
    street: '28th Avenue',
    zipcode: 10490
  },
  borough: 'Bronx',
  cuisine: 'BBQ',
  grades: [
    {
      date: 2014-03-03T00:00:00.028Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2013-09-11T00:00:00.028Z,
      grade: 'A',
      score: 7
    },
    {
      date: 2013-01-24T00:00:00.028Z,
      grade: 'A',
      score: 11
    },
    {
      date: 2011-11-23T00:00:00.028Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2011-03-10T00:00:00.028Z,
      grade: 'B',

```

```
    score: 15
  }
],
name: 'BBQ Haven',
restaurant_id: 30075473
}
```

```
{
  _id: ObjectId('671b5dab56ec9972ca8f5db0'),
  address: {
    building: 5566,
    coord: [
      -73.859377,
      40.850047
    ],
    street: '8th Avenue',
    zipcode: 10470
  },
  borough: 'Manhattan',
  cuisine: 'French',
  grades: [
    {
      date: 2014-03-03T00:00:00.008Z,
      grade: 'A',
      score: 7
    },
    {
      date: 2013-09-11T00:00:00.008Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2013-01-24T00:00:00.008Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.008Z,
      grade: 'B',
      score: 15
    },
    {
      date: 2011-03-10T00:00:00.008Z,
```

```
    grade: 'A',
    score: 6
  }
],
name: 'Bistro Belle',
restaurant_id: 30075453
}
```

6. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

```
db.restaurants.find().sort({ name: -1 });
```

#### SAMPLE OUTPUT

```
{
  _id: ObjectId('671b5e9456ec9972ca8f5dc8'),
  address: {
    building: 9900,
    coord: [
      -73.868977,
      40.854847
    ],
    street: '32nd Avenue',
    zipcode: 10494
  },
  borough: 'Manhattan',
  cuisine: 'Russian',
  grades: [
    {
      date: 2014-03-03T00:00:00.032Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2013-09-11T00:00:00.032Z,
      grade: 'B',
      score: 5
    },
    {

```

```
    date: 2013-01-24T00:00:00.032Z,  
    grade: 'A',  
    score: 9  
  },  
  {  
    date: 2011-11-23T00:00:00.032Z,  
    grade: 'A',  
    score: 8  
  },  
  {  
    date: 2011-03-10T00:00:00.032Z,  
    grade: 'A',  
    score: 11  
  }  
],  
name: "Tsar's Table",  
restaurant_id: 30075477  
}
```

```
{  
  _id: ObjectId('671b5e6d56ec9972ca8f5dbe'),  
  address: {  
    building: 9900,  
    coord: [  
      -73.864977,  
      40.852847  
    ],  
    street: '22nd Avenue',  
    zipcode: 10484  
  },  
  borough: 'Bronx',  
  cuisine: 'Italian',  
  grades: [  
    {  
      date: 2014-03-03T00:00:00.022Z,  
      grade: 'A',  
      score: 8  
    },  
    {  
      date: 2013-09-11T00:00:00.022Z,  
      grade: 'B',  
      score: 5  
    },  
  ],  
}
```

```

{
  date: 2013-01-24T00:00:00.022Z,
  grade: 'A',
  score: 12
},
{
  date: 2011-11-23T00:00:00.022Z,
  grade: 'A',
  score: 9
},
{
  date: 2011-03-10T00:00:00.022Z,
  grade: 'A',
  score: 14
}
],
name: 'Trattoria Bella',
restaurant_id: 30075467
}

```

7. Write a MongoDB query to arrange the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

```
db.restaurants.find().sort({ cuisine: 1, borough: -1 });
```

**SAMPLE OUTPUT:-**

```

{
  _id: ObjectId('671b5d549d3d63480e0a64e9'),
  address: {
    building: 2233,
    coord: [
      -73.858177,
      40.849447
    ],
    street: '5th Avenue',
    zipcode: 10467
  },
  borough: 'Bronx',
  cuisine: 'American',

```

```
grades: [  
  {  
    date: 2014-03-03T00:00:00.005Z,  
    grade: 'A',  
    score: 10  
  },  
  {  
    date: 2013-09-11T00:00:00.005Z,  
    grade: 'A',  
    score: 6  
  },  
  {  
    date: 2013-01-24T00:00:00.005Z,  
    grade: 'B',  
    score: 12  
  },  
  {  
    date: 2011-11-23T00:00:00.005Z,  
    grade: 'A',  
    score: 9  
  },  
  {  
    date: 2011-03-10T00:00:00.005Z,  
    grade: 'A',  
    score: 14  
  }  
],  
name: 'Burger Bistro',  
restaurant_id: 30075450  
}  
  
{  
  _id: ObjectId('671b5e6d56ec9972ca8f5dc4'),  
  address: {  
    building: 5566,  
    coord: [  
      -73.867377,  
      40.854047  
    ],  
    street: '28th Avenue',  
    zipcode: 10490  
  },  
  borough: 'Bronx',  
  cuisine: 'BBQ',  
}
```

```
grades: [  
  {  
    date: 2014-03-03T00:00:00.028Z,  
    grade: 'A',  
    score: 10  
  },  
  {  
    date: 2013-09-11T00:00:00.028Z,  
    grade: 'A',  
    score: 7  
  },  
  {  
    date: 2013-01-24T00:00:00.028Z,  
    grade: 'A',  
    score: 11  
  },  
  {  
    date: 2011-11-23T00:00:00.028Z,  
    grade: 'A',  
    score: 9  
  },  
  {  
    date: 2011-03-10T00:00:00.028Z,  
    grade: 'B',  
    score: 15  
  }  
],  
name: 'BBQ Haven',  
restaurant_id: 30075473  
}
```

8. Write a MongoDB query to know whether all the addresses contains the street or not.

```
db.restaurants.find(  
  {  
    "address.street": { $exists: false }  
  }  
);
```



```
> db.restaurants.find(
  {
    "address.street": { $exists: false }
  }
);
<
Customers >
```

9. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

```
db.restaurants.find(
  {
    "address.coord": { $type: "double" }
  }
);
```

**SAMPLE OUTPUT:-**

```
{
  _id: ObjectId('671b92d339ec8a9bc8b6588b'),
  address: {
    building: '1007',
    coord: [
      -73.856077,
      40.848447
    ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: 2014-03-03T00:00:00.000Z,
```

```
    grade: 'A',
    score: 2
  },
  {
    date: 2013-09-11T00:00:00.000Z,
    grade: 'A',
    score: 6
  },
  {
    date: 2013-01-24T00:00:00.000Z,
    grade: 'A',
    score: 10
  },
  {
    date: 2011-11-23T00:00:00.000Z,
    grade: 'A',
    score: 9
  },
  {
    date: 2011-03-10T00:00:00.000Z,
    grade: 'B',
    score: 14
  }
],
name: 'Morris Park Bake Shop',
restaurant_id: '30075445'
}

{
  _id: ObjectId('671b5d549d3d63480e0a64e5'),
  address: {
    building: 1234,
    coord: [
      -73.856577,
      40.848647
    ],
    street: '1st Avenue',
    zipcode: 10463
  },
  borough: 'Bronx',
  cuisine: 'Italian',
  grades: [
    {
      date: 2014-03-03T00:00:00.001Z,
```

```

    grade: 'A',
    score: 5
  },
  {
    date: 2013-09-11T00:00:00.001Z,
    grade: 'A',
    score: 8
  },
  {
    date: 2013-01-24T00:00:00.001Z,
    grade: 'B',
    score: 12
  },
  {
    date: 2011-11-23T00:00:00.001Z,
    grade: 'A',
    score: 7
  },
  {
    date: 2011-03-10T00:00:00.001Z,
    grade: 'A',
    score: 15
  }
],
name: 'Pasta Palace',
restaurant_id: 30075446
}

```

10. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

```

db.restaurants.find(
  {
    "grades.score": { $mod: [7, 0] }
  },
  {
    restaurant_id: 1,
    name: 1,
    grades: 1,
    _id: 0
  }
);

```

### SAMPLE OUTPUT:-

```
{
  grades: [
    {
      date: 2014-03-03T00:00:00.000Z,
      grade: 'A',
      score: 2
    },
    {
      date: 2013-09-11T00:00:00.000Z,
      grade: 'A',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.000Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.000Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2011-03-10T00:00:00.000Z,
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
```

```
{
  grades: [
    {
      date: 2014-03-03T00:00:00.001Z,
      grade: 'A',
      score: 5
    },
    {
```

```

    date: 2013-09-11T00:00:00.001Z,
    grade: 'A',
    score: 8
  },
  {
    date: 2013-01-24T00:00:00.001Z,
    grade: 'B',
    score: 12
  },
  {
    date: 2011-11-23T00:00:00.001Z,
    grade: 'A',
    score: 7
  },
  {
    date: 2011-03-10T00:00:00.001Z,
    grade: 'A',
    score: 15
  }
],
name: 'Pasta Palace',
restaurant_id: 30075446
}

```

11. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

```

db.restaurants.find(
{
  name: { $regex: /mon/i }
},
{
  name: 1,
  borough: 1,
  "address.coord.0": 1, // Longitude
  "address.coord.1": 1, // Latitude
  cuisine: 1,
  _id: 0
}
);

```

12. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

```
db.restaurants.find(
{
  name: { $regex: /^Mad/i }
},
{
  name: 1,
  borough: 1,
  "address.coord.0": 1, // Longitude
  "address.coord.1": 1, // Latitude
  cuisine: 1,
  _id: 0
}
);
```

13. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5.

```
db.restaurants.find(
{
  "grades.score": { $lt: 5 }
}
);
```

#### SAMPLE OUTPUT:-

```
{
  _id: ObjectId('671b92d339ec8a9bc8b6588b'),
  address: {
    building: '1007',
```

```
    coord: [
      -73.856077,
      40.848447
    ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: 2014-03-03T00:00:00.000Z,
      grade: 'A',
      score: 2
    },
    {
      date: 2013-09-11T00:00:00.000Z,
      grade: 'A',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.000Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.000Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2011-03-10T00:00:00.000Z,
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}

{
  _id: ObjectId('671b5d549d3d63480e0a64e6'),
  address: {
```

```
    building: 5678,
    coord: [
      -73.856977,
      40.848847
    ],
    street: '2nd Avenue',
    zipcode: 10464
  },
  borough: 'Manhattan',
  cuisine: 'Chinese',
  grades: [
    {
      date: 2014-03-03T00:00:00.002Z,
      grade: 'B',
      score: 4
    },
    {
      date: 2013-09-11T00:00:00.002Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2013-01-24T00:00:00.002Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.002Z,
      grade: 'A',
      score: 8
    },
    {
      date: 2011-03-10T00:00:00.002Z,
      grade: 'B',
      score: 16
    }
  ],
  name: 'Dragon Wok',
  restaurant_id: 30075447
}
```



**14. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.**

```
db.restaurants.find(
{
  "grades.score": { $lt: 5 },
  borough: "Manhattan"
}
);
```

**15. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.**

```
db.restaurants.find(
{
  "grades.score": { $lt: 5 },
  borough: { $in: ["Manhattan", "Brooklyn"] }
}
);
```

```

    _id: ObjectId('671b5d549d3d63480e0a64e6'),
    address: {
      building: 5678,
      coord: [
        -73.856977,
        40.848847
      ],
      street: '2nd Avenue',
      zipcode: 10464
    },
    borough: 'Manhattan',
    cuisine: 'Chinese',
    grades: [
      {
        date: 2014-03-03T00:00:00.002Z,
        grade: 'B',
        score: 4
      },
      {
        date: 2013-09-11T00:00:00.002Z,
        grade: 'A',
        score: 9
      },
      {
        date: 2013-01-24T00:00:00.002Z,
        grade: 'A',
        score: 10
      }
    ],
  },

```

16. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```

db.restaurants.find(
{
  "grades.score": { $lt: 5 },
  borough: { $in: ["Manhattan", "Brooklyn"] },
  cuisine: { $ne: "American" }
}
);

```

```

    _id: ObjectId('671b5d549d3d63480e0a64e6'),
    address: {
      building: 5678,
      coord: [
        -73.856977,
        40.848847
      ],
      street: '2nd Avenue',
      zipcode: 10464
    },
    borough: 'Manhattan',
    cuisine: 'Chinese',
    grades: [
      {
        date: 2014-03-03T00:00:00.002Z,
        grade: 'B',
        score: 4
      },
      {
        date: 2013-09-11T00:00:00.002Z,
        grade: 'A',
        score: 9
      },
      {
        date: 2013-01-24T00:00:00.002Z,
        grade: 'A',
        score: 10
      },
      {

```

17. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```

db.restaurants.find(
{
  "grades.score": { $lt: 5 },
  borough: { $in: ["Manhattan", "Brooklyn"] },
  cuisine: { $nin: ["American", "Chinese"] }
}
);

```

18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

```

db.restaurants.find(
{
  grades: {
    $all: [
      { $elemMatch: { score: 2 } },

```

```
        { $elemMatch: { score: 6 } }  
    ]  
}  
}  
);
```

#### SAMPLE OUTPUT:-

```
{  
  _id: ObjectId('671b92d339ec8a9bc8b6588b'),  
  address: {  
    building: '1007',  
    coord: [  
      -73.856077,  
      40.848447  
    ],  
    street: 'Morris Park Ave',  
    zipcode: '10462'  
  },  
  borough: 'Bronx',  
  cuisine: 'Bakery',  
  grades: [  
    {  
      date: 2014-03-03T00:00:00.000Z,  
      grade: 'A',  
      score: 2  
    },  
    {  
      date: 2013-09-11T00:00:00.000Z,  
      grade: 'A',  
      score: 6  
    },  
    {  
      date: 2013-01-24T00:00:00.000Z,  
      grade: 'A',  
      score: 10  
    },  
    {  
      date: 2011-11-23T00:00:00.000Z,  
      grade: 'A',  
      score: 9  
    },  
    {  
      date: 2011-03-10T00:00:00.000Z,
```

```
    grade: 'B',
    score: 14
  },
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
```

```
{
  _id: ObjectId('671b5c5f9d3d63480e0a64e4'),
  address: {
    building: 1007,
    coord: [
      -73.856077,
      40.848447
    ],
    street: 'Morris Park Ave',
    zipcode: 10462
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: 2014-03-03T00:00:00.000Z,
      grade: 'A',
      score: 2
    },
    {
      date: 2013-09-11T00:00:00.000Z,
      grade: 'A',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.000Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2011-11-23T00:00:00.000Z,
      grade: 'A',
      score: 9
    },
    {

```

```
    date: 2011-03-10T00:00:00.000Z,  
    grade: 'B',  
    score: 14  
  }  
],  
name: 'Morris Park Bake Shop',  
restaurant_id: 30075445  
}
```

19. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

```
db.restaurants.find(  
  {  
    borough: "Manhattan",  
    grades: {  
      $all: [  
        { $elemMatch: { score: 2 } },  
        { $elemMatch: { score: 6 } }  
      ]  
    }  
  }  
);
```

20. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

```
db.restaurants.find(  
  {  
    borough: { $in: ["Manhattan", "Brooklyn"] },  
    grades: {  
      $all: [  
        { $elemMatch: { score: 2 } },  
        { $elemMatch: { score: 6 } }  
      ]  
    }  
  }  
);
```

```

    { $elemMatch: { score: 6 } }
  ]
}
);

```

21. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```

db.restaurants.find(
{
  borough: { $in: ["Manhattan", "Brooklyn"] },
  grades: {
    $all: [
      { $elemMatch: { score: 2 } },
      { $elemMatch: { score: 6 } }
    ]
  },
  cuisine: { $ne: "American" }
}
);

```

22. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```

db.restaurants.find(
{
  borough: { $in: ["Manhattan", "Brooklyn"] },
  grades: {
    $all: [
      { $elemMatch: { score: 2 } },
      { $elemMatch: { score: 6 } }
    ]
  },
  cuisine: { $nin: ["American", "Chinese"] }
}
);

```

23. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6.

```
db.restaurants.find(
{
  $or: [
    { "grades.score": 2 },
    { "grades.score": 6 }
  ]
}
);
```

**SAMPLE OUTPUT:-**

```
{
  _id: ObjectId('671b5d549d3d63480e0a64e9'),
  address: {
    building: 2233,
    coord: [
      -73.858177,
      40.849447
    ],
    street: '5th Avenue',
    zipcode: 10467
  },
  borough: 'Bronx',
  cuisine: 'American',
  grades: [
    {
      date: 2014-03-03T00:00:00.005Z,
      grade: 'A',
      score: 10
    },
    {
      date: 2013-09-11T00:00:00.005Z,
      grade: 'A',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.005Z,
```



```
    grade: 'B',
    score: 12
  },
  {
    date: 2011-11-23T00:00:00.005Z,
    grade: 'A',
    score: 9
  },
  {
    date: 2011-03-10T00:00:00.005Z,
    grade: 'A',
    score: 14
  }
],
name: 'Burger Bistro',
restaurant_id: 30075450
}

{
  _id: ObjectId('671b5dab56ec9972ca8f5daf'),
  address: {
    building: 4455,
    coord: [
      -73.858977,
      40.849847
    ],
    street: '7th Avenue',
    zipcode: 10469
  },
  borough: 'Bronx',
  cuisine: 'Thai',
  grades: [
    {
      date: 2014-03-03T00:00:00.007Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2013-09-11T00:00:00.007Z,
      grade: 'B',
      score: 6
    },
    {
      date: 2013-01-24T00:00:00.007Z,
```

```
    grade: 'A',
    score: 12
  },
  {
    date: 2011-11-23T00:00:00.007Z,
    grade: 'A',
    score: 8
  },
  {
    date: 2011-03-10T00:00:00.007Z,
    grade: 'B',
    score: 14
  }
],
name: 'Thai Delight',
restaurant_id: 30075452
}
```

## MOVIES COLLECTION

1. Find all movies with full information from the 'movies' collection that released in the year 1893.

```
db.movies.find({ year: 1893 });
```

2. Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.

```
db.movies.find({ runtime: { $gt: 120 } });
```

### SAMPLE OUTPUT:-

```
{
  _id: ObjectId('573a1390f29313caabcd42ec'),
  plot: 'An astronaut stranded on Mars must survive alone.',
  genres: [
    'Sci-Fi',
    'Drama'
  ],
  runtime: 135,
  cast: [
    'Matt Damon',
    'Jessica Chastain'
  ],
  poster: 'https://m.media-amazon.com/images/poster4.jpg',
  title: 'Mars Alone',
  fullplot: 'An astronaut, left alone on Mars, struggles to survive with limited resources while awaiting rescue.',
  languages: [
```

```
{
  'English':
  ],
  released: '2015-10-02T00:00:00.000Z',
  directors: [
    'Ridley Scott'
  ],
  rated: 'PG-13',
  awards: {
    wins: 8,
    nominations: 6,
    text: '8 wins & 6 nominations.'
  },
  lastupdated: '2021-08-09 17:22:30.000000000',
  year: 2015,
  imdb: {
    rating: 8,
    votes: 25650,
    id: 443
  },
  countries: [
    'USA'
  ],
  type: 'movie',
  tomatoes: {
    viewer: {
      rating: 4.5,
      numReviews: 2201,
      meter: 93
    },
    fresh: 18,
    critic: {
      rating: 8.5,
      numReviews: 25,
      meter: 96
    }
  },
}
```

```
    rotten: 1,  
    lastUpdated: 2021-07-19T21:20:55.000Z  
  }  
}
```

3. Find all movies with full information from the 'movies' collection that have "Short" genre.

```
db.movies.find({ genres: "Short" });
```

### SAMPLE OUTPUT:-

```
{  
  _id: ObjectId('573a1390f29313caabcd42e8'),  
  plot: 'A group of bandits stage a brazen train hold-up, only to find a  
determined posse hot on their heels.',  
  genres: [  
    'Short',  
    'Western'  
  ],  
  runtime: 11,  
  cast: [  
    'A.C. Abadie',  
    "Gilbert M. 'Broncho Billy' Anderson",  
    'George Barnes',  
    'Justus D. Barnes'  
  ],  
  poster: 'https://m.media-  
amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWlwYjg  
tMmYwYWlxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1  
000_SX677_AL_.jpg',  
  title: 'The Great Train Robbery',  
  fullplot: "Among the earliest existing films in American cinema -  
notable as the first film that presented a narrative story to tell - it  
depicts a group of cowboy outlaws who hold up a train and rob the
```

passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",

languages: [

'English'

],

released: 1903-12-01T00:00:00.000Z,

directors: [

'Edwin S. Porter'

],

rated: 'TV-G',

awards: {

wins: 1,

  nominations: 0,

  text: '1 win.'

},

lastupdated: '2015-08-13 00:27:59.177000000',

year: 1903,

imdb: {

  rating: 7.4,

  votes: 9847,

  id: 439

},

countries: [

'USA'

],

type: 'movie',

tomatoes: {

  viewer: {

    rating: 3.7,

    numReviews: 2559,

    meter: 75

  },

  fresh: 6,

  critic: {

    rating: 7.6,

```

    numReviews: 6,
    meter: 100
  },
  rotten: 0,
  lastUpdated: 2015-08-08T19:16:10.000Z
}
}

```

4. Retrieve all movies from the 'movies' collection that were directed by "William K.L. Dickson" and include complete information for each movie.

```
db.movies.find({ directors: "William K.L. Dickson" });
```

6. Retrieve all movies from the 'movies' collection that were released in the USA and include complete information for each movie.

```
db.movies.find({ countries: "USA" });
```

```

{
  _id: ObjectId('573a1390f29313caabdc42e8'),
  plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
  genres: [
    'Short',
    'Western'
  ],
  runtime: 11,
  cast: [
    'A.C. Abadie',
    'Gilbert M. 'Broncho Billy' Anderson',
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjESNzYtYTYyNS00MDVhLWlWYjgtMmVhYWIxZDZyNzU2XkEyXkFqcGdeQXVyMzQzNzQxNzI0._V1_SY1000_',
  title: 'The Great Train Robbery',
  fullplot: 'Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it',
  languages: [
    'English'
  ],
  released: 1903-12-01T00:00:00.000Z,
  directors: [

```

7. Retrieve all movies from the 'movies' collection that have complete information and are rated as "UNRATED".

```
db.movies.find({ rated: "UNRATED" });
```

8. Retrieve all movies from the 'movies' collection that have complete information and have received more than 1000 votes on IMDb.

```
db.movies.find({ "imdb.votes": { $gt: 1000 } });
```

```
< {
  _id: ObjectId('573a1390f29313caabcd42e8'),
  plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
  genres: [
    'Short',
    'Western'
  ],
  runtime: 11,
  cast: [
    'A.C. Abadie',
    'Gilbert M. 'Broncho Billy' Anderson',
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYjgtMmYwYWIXZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000
  title: 'The Great Train Robbery',
  fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - i
  languages: [
    'English'
  ],
  released: 1903-12-01T00:00:00.000Z,
  directors: [
    'Edwin S. Porter'
  ],
}
```

9. Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

```
db.movies.find({ "imdb.rating": { $gt: 7 } });
```



```

> db.movies.find({ "imdb.rating": { $gt: 7 } });
< {
  _id: ObjectId('573a1390f29313caabcd42e8'),
  plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
  genres: [
    'Short',
    'Western'
  ],
  runtime: 11,
  cast: [
    'A.C. Abadie',
    'Gilbert M. 'Broncho Billy' Anderson',
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTtyNS00MDVmLWIwVjgtMmYwYWIXZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000
  title: 'The Great Train Robbery',
  fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - i
  languages: [
    'English'
  ],
  released: 1903-12-01T00:00:00.000Z,
  directors: [
    'Edwin S. Porter'
  ],
  rated: 'TV-G',
  awards: {
    wins: 1,

```

**10. Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.**

**`db.movies.find({ "tomatoes.viewer.rating": { $gt: 4 } });`**

```

> db.movies.find({ "tomatoes.viewer.rating": { $gt: 4 } });
< {
  _id: ObjectId('573a1390f29313caabcd42ea'),
  plot: 'A chef tries to open a restaurant amidst a series of challenges.',
  genres: [
    'Drama',
    'Comedy'
  ],
  runtime: 120,
  cast: [
    'Emma Stone',
    'Chris Pratt',
    'Anna Kendrick'
  ],
  poster: 'https://m.media-amazon.com/images/poster2.jpg',
  title: 'The Culinary Dream',
  fullplot: "A chef's journey to make his dream restaurant come true, overcoming family and financial obstacles.",
  languages: [
    'English',
    'French'
  ],
  released: 2015-02-12T00:00:00.000Z,
  directors: [
    'Samantha Jones'
  ],
  rated: 'PG-13',
  awards: {
    wins: 1,

```

**11. Retrieve all movies from the 'movies' collection that have received an award.**

```
db.movies.find({ "awards.wins": { $gt: 0 } });
```

```

> db.movies.find({ "awards.wins": { $gt: 0 } });
< {
  _id: ObjectId('573a1390f29313caabcd42e8'),
  plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
  genres: [
    'Short',
    'Western'
  ],
  runtime: 11,
  cast: [
    'A.C. Abadie',
    "Gilbert M. 'Broncho Billy' Anderson",
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYjgtMmYwYWIXZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000
  title: 'The Great Train Robbery',
  fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - i
  languages: [
    'English'
  ],
  released: 1903-12-01T00:00:00.000Z,
  directors: [
    'Edwin S. Porter'
  ],
  rated: 'TV-G',
  awards: {
    wins: 1,

```

**12. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB that have at least one nomination.**

```

db.movies.find(
  { "awards.nominations": { $gt: 0 } },
  {
    title: 1,
    languages: 1,
    released: 1,
    directors: 1,
    writers: 1,
    awards: 1,
    year: 1,
    genres: 1,
    runtime: 1,
    cast: 1,
    countries: 1
  }
)

```

```
};
```

```
>_MONGOSH
/,
< {
  _id: ObjectId('573a1390f29313caabcd42e9'),
  genres: [
    'Adventure',
    'Fantasy'
  ],
  runtime: 95,
  cast: [
    'Ethan Hawke',
    'Jane Doe',
    'Mark Strong'
  ],
  title: 'The Amulet Quest',
  languages: [
    'English'
  ],
  released: 2008-07-15T00:00:00.000Z,
  directors: [
    'John Smith'
  ],
  awards: {
    wins: 2,
    nominations: 1,
    text: '2 wins & 1 nomination.'
  },
  year: 2008,
  countries: [
    'USA'
  ]
}
```

**13. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".**

```
db.movies.find(
  { cast: "Charles Kayser" },
  {
    title: 1,
    languages: 1,
    released: 1,
    directors: 1,
    writers: 1,
    awards: 1,
    year: 1,
  }
)
```

```
    genres: 1,  
    runtime: 1,  
    cast: 1,  
    countries: 1  
  }  
);
```

**14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that released on May 9, 1893.**

```
db.movies.find(  
  { released: ISODate("1893-05-09T00:00:00Z") },  
  {  
    title: 1,  
    languages: 1,  
    released: 1,  
    directors: 1,  
    writers: 1,  
    countries: 1  
  }  
);
```

**14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.**

```
db.movies.find(  
  { title: { $regex: /scene/i } },  
  {  
    title: 1,  
    languages: 1,  
  }  
);
```

```
released: 1,  
directors: 1,  
writers: 1,  
countries: 1  
}  
);
```



```
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Administration');
```

```
SELECT * FROM DEPT  
WHERE DEPT_NAME IN ('Education', 'Administration');
```

| DEPT_ID | DEPT_NAME      |
|---------|----------------|
| 210     | Administration |
| 200     | Education      |

2 rows returned in 0.04 seconds [Download](#)

4. Create a non unique index on the foreign key column (DEPARTMENT\_ID) in the EMPLOYEES table.

```
CREATE INDEX employees_department_id_idx  
ON EMPLOYEES (DEPARTMENT_ID);
```

5. Display the indexes and uniqueness that exist in the data dictionary for the EMP table.

```
SELECT INDEX_NAME, UNIQUENESS  
FROM USER_INDEXES  
WHERE TABLE_NAME = 'EMPLOYEES';
```

| INDEX_NAME                  | UNIQUENESS |
|-----------------------------|------------|
| EMPLOYEES_DEPARTMENT_ID_IDX | NONUNIQUE  |
| SYS_C00163680725            | UNIQUE     |

2 rows returned in 0.05 seconds [Download](#)



|                   |            |                                |
|-------------------|------------|--------------------------------|
| <b>Ex.No.: 16</b> |            | <b>CONTROLLING USER ACCESS</b> |
| <b>Date:</b>      | 03/10/2024 |                                |

1. What privilege should a user be given to log on to the Oracle Server? Is this a system or an object privilege?

The privilege a user should be given to log on to the Oracle Server is the CREATE SESSION privilege.

Type of Privilege: This is a system privilege.

GRANT CREATE SESSION TO username;

2. What privilege should a user be given to create tables?

the user needs the CREATE TABLE privilege.

The CREATE TABLE privilege allows the user to create new tables in their own schema.

GRANT CREATE TABLE TO username;

3. If you create a table, who can pass along privileges to other users on your table?

When you create a table, only you as the table owner (or a user with the ADMIN OPTION or GRANT ANY PRIVILEGE system privilege) can grant privileges on your table to other users.

GRANT SELECT ON your\_table TO other\_user;

4. You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?

As a DBA, to simplify the process of granting the same system privileges to multiple users, you should use roles.

```
CREATE ROLE my_role;
```

```
GRANT CREATE SESSION TO my_role;  
GRANT CREATE TABLE TO my_role;
```

```
GRANT my_role TO user1;  
GRANT my_role TO user2;
```

5. What command do you use to change your password?

```
ALTER USER username IDENTIFIED BY new_password;
```

6. Grant another user access to your DEPARTMENTS table. Have the user grant you query Access to his or her DEPARTMENTS table.

Grant Access to Your DEPARTMENTS Table

```
GRANT SELECT ON your_username.DEPARTMENTS TO other_user;
```

Grant Query Access to Other User's DEPARTMENTS Table

```
GRANT SELECT ON other_user.DEPARTMENTS TO your_username;
```

7. Query all the rows in your DEPARTMENTS table.

```
SELECT * FROM DEPARTMENT;
```

| Results | Explain       | Describe   | Saved SQL   | History    |              |
|---------|---------------|------------|-------------|------------|--------------|
| DEPT_ID | DEPT_NAME     | MANAGER_ID | LOCATION_ID | COUNTRY_ID | MANAGER_NAME |
| 70      | HR            | 800        | 2           | IND        | don          |
| 25      | executive     | 400        | 10          | AFG        | king         |
| 50      | manager       | 200        | 10          | US         | king         |
| 80      | stock clerk   | 150        | 19          | UK         | riyaan       |
| 45      | IT support    | 400        | 13          | IS         | bell         |
| 15      | sales manager | 250        | 7           | AFG        | mont         |

8. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team's table.

```
INSERT INTO DEPARTMENT(dept_id,
DEPT_NAME,manager_id,location_id,country_id,manager_name)
VALUES (500, 'Education',300,12,'BAN','ball');
```

```
INSERT INTO DEPARTMENT(dept_id,
DEPT_NAME,manager_id,location_id,country_id,manager_name)
VALUES (510, 'Human Resources',150,10,'AUS','john');
```

```
SELECT * FROM DEPARTMENT;
```

Results

Explain

Describe

Saved SQL

History

| DEPT_ID | DEPT_NAME       | MANAGER_ID | LOCATION_ID | COUNTRY_ID | MANAGER_NAME |
|---------|-----------------|------------|-------------|------------|--------------|
| 510     | Human Resources | 150        | 10          | AUS        | john         |
| 500     | Education       | 300        | 12          | BAN        | ball         |

9. Query the USER\_TABLES data dictionary to see information about the tables that you own.

```
SELECT * FROM USER_TABLES;
```

| Results Explain Describe Saved SQL History |                            |              |          |        |          |          |           |           |                |             |             |             |
|--|----------------------------|--------------|----------|--------|----------|----------|-----------|-----------|----------------|-------------|-------------|-------------|
| TABLE_NAME                                 | TABLESPACE_NAME            | CLUSTER_NAME | IOT_NAME | STATUS | PCT_FREE | PCT_USED | INI_TRANS | MAX_TRANS | INITIAL_EXTENT | NEXT_EXTENT | MIN_EXTENTS | MAX_EXTENTS |
| AUDIT_LOG                                  | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| BOOKS                                      | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| COUNTRIES                                  | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| DEPARTMENT                                 | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| DEPT                                       | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| EMP  | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| EMP1                                       | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | -              | -           | -           | -           |
| EMPLOYEES                                  | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| EMPLOYEE_AUDIT                             | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |
| HTMLDB_PLAN_TABLE                          | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | -              | -           | -           | -           |
| INVENTORY                                  | APEX_BIGFILE_INSTANCE_TBS3 | -            | -        | VALID  | 10       | -        | 1         | 255       | 65536          | 1048576     | 1           | 2147483645  |

10. Revoke the SELECT privilege on your table from the other team.

**REVOKE SELECT ON team1\_user.DEPARTMENTS FROM other\_user;**

11. Remove the row you inserted into the DEPARTMENTS table in step 8 and save the changes.

**DELETE FROM DEPARTMENT  
WHERE DEPT\_ID IN (500, 510);**