

**STEP-8:** The truncate command deletes all rows from the table. Only the structure of the table remains.

Syntax:

```
TRUNCATE TABLE <table name>;
```

**STEP-9:** Alter the existing table using ALTER statement.

Syntax:

Add Column:

```
ALTER TABLE <table name> ADD (column data type  
[DEFAULT Texpr][,column data type]);
```

Modify Column:

```
ALTER TABLE <table name> MODIFY (column data type  
[DEFAULT expr], [,column data type]);
```

Drop Column:

```
ALTER TABLE <table name> DROP COLUMN <column name>;
```

**STEP-10:** To drop the entire table using DROP statement.

Syntax:

```
DROP TABLE <table name>;
```

**STEP-11:** Exit.

1. Create MY\_EMPLOYEE table with the following structure

```
create table MY_employee (empid number(4) not null, last_name varchar(10)  
, first_name varchar(10), user_id varchar(20), salary number(20));
```

NAME	NULL?	TYPE
ID	Not null	Number(4)
Last_name		Varchar(25)
First_name		Varchar(25)
Userid		Varchar(25)
Salary		Number(9,2)

2. Add the first and second rows data to MY\_EMPLOYEE table from the following sample data.

ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

insert into my\_employee values('1', 'Patel', 'Ralph', 'rpatel', '895')  
 insert into my\_employee values('2', 'Dancs', 'Betty', 'bdancs', '860')

3. Display the table with values.

select \* from my\_employee

4. Populate the next two rows of data from the sample data. Concatenate the first letter of the first\_name with the first seven characters of the last\_name to produce Userid.

insert into my\_employee ('3', 'Biri', 'Ben', 'bbiri', '1100');  
 insert into my\_employee ('4', 'Newman', 'Chad', 'Cnewman', '750')

5. Delete Betty dancs from MY\_EMPLOYEE table.

delete from my\_employee where first\_name = 'betty';

6. Empty the fourth row of the emp table.

~~delete from my-employee1 where emp-id = '4';~~

7. Make the data additions permanent.

commit;

8. Change the last name of employee 3 to Drexler.

update my-employee1 set lastname = 'drexler' where emp-id = 3;

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

update my-employee1 set salary = '1000' where salary < 900;

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	



Create the following tables with the given structure.

EMPLOYEES TABLE

NAME	NULL?	TYPE
Employee_id	Not null	Number(6)
First_Name		Varchar(20)
Last_Name	Not null	Varchar(25)
Email	Not null	Varchar(25)
Phone_Number		Varchar(20)
Hire_date	Not null	Date
Job_id	Not null	Varchar(10)
Salary		Number(8,2)
Commission_pct		Number(2,2)
Manager_id		Number(6)
Department_id		Number(4)

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

select employee\_id, first\_name, last\_name, salary  
from employees;

(b) List out the employees who works under manager 100

select \* from employees where manager\_id=100

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

select \* from employees where salary >= 4800;

(d) List out the employees whose last name is 'AUSTIN'

$\text{select * from emp where last\_name = 'Austin'}$ ;

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

$\text{select * from emp where Department\_id in (60, 70, 80)}$ ;

(f) Display the unique Manager\_Id.

$\text{select * distinct manager\_id from emp}$ ;

Create an Emp table with the following fields: (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay) (Calculate DA as 30% of Basic and HRA as 40% of Basic)

$\text{create table emp (empno number(6), empname varchar(15), job varchar(15), basic decimal(10,2), da decimal(10,2), hra decimal(10,2), pf decimal(10,2), grosspay decimal(10,2), netpay decimal(10,2))}$

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

$\text{update emp table set grosspay = basic + da + hra + pf}$

$\text{netpay = grosspay - pf}$ ;

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

$\text{after table emp add dept varchar(20)}$ ;

$\text{select min (basic) as lowest from emp group by department}$

(c) If Net Pay is less than

10,000 list those the emp\_id, emp\_name, netpay;

$\text{select emp name, netpay from emp where netpay < 10000}$

DEPARTMENT TABLE

NAME	NULL?	TYPE
Dept_id	Not null	Number(6)
Dept_name	Not null	Varchar(20)
Manager_id		Number(6)
Location_id		Number(4)

JOB\_GRADE TABLE

NAME	NULL?	TYPE
Grade_level		Varchar(2)
Lowest_sal		Number
Highest_sal		Number

LOCATION TABLE

NAME	NULL?	TYPE
Location_id	Not null	Number(4)
St_addr		Varchar(40)
Postal_code		Varchar(12)
City	Not null	Varchar(30)
State_province		Varchar(25)
Country_id		Char(2)

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

Column name	ID	NAME
Key Type		
Nulls/Unique		
FK table		
FK column		
Data Type	Number	Varchar2
Length	7	25

Create table Dept (

ID number(7),

name varchar(25),

);

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

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK table				
FK column				
Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

create table emp (ID number(7), first\_name varchar(25),  
last\_name varchar(25), dept\_id number(7));

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

alter table emp modify last\_name varchar(50);

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 (emp\_id number(4), first\_name  
varchar(20), last\_name varchar(20), salary number(6,2),  
dept\_id number(4));

5. Drop the EMP table.

drop table emp;

6. Rename the EMPLOYEES2 table as EMP.

alter table employees2 rename to emp;



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

comment on table dept is 'this Department table'  
comment on table emp is 'this is employee table';  
desc emp from user - has - comment;

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

alter table emp drop column first name

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	(R)



### Using Literal Character String

- A literal is a character, a number, or a date included in the SELECT list.
- Date and character literal values must be enclosed within single quotation marks.

#### Example:

SELECT last\_name||' is a '||job\_id AS "EMPLOYEES JOB" FROM employees;

### Eliminating Duplicate Rows

- Using DISTINCT keyword.

#### Example:

SELECT DISTINCT department\_id FROM employees;

### Displaying Table Structure

- Using DESC keyword.

#### Syntax

DESC table\_name;

#### Example:

DESC employees;

### 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;
```

### Queries

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

*desc departments;*

*select \* from departments*

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 - number, last name, jobcode,  
hire\_data from employees;

4. Provide an alias STARTDATE for the hire date.

select hire date as start date from employees

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


select unique (jobcodes) from employee

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

select job\_id || ' ' || last name as "employee and  
title" from employees;

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 - number || ' ' || last - name || ' ' ||  
job - code || ' ' || ' ' || hire date || ' ' || salary as "the op"  
from employees

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	

(OR)

ALTER TABLE test1 DROP(pk, fk, coll) CASCADE CONSTRAINTS;

### VIEWING CONSTRAINTS

Query the USER\_CONSTRAINTS table to view all the constraints definition and names.

#### Example:

```
SELECT constraint_name, constraint_type, search_condition FROM user_constraints  
WHERE table_name='employees';
```

#### Viewing the columns associated with constraints

```
SELECT constraint_name, constraint_type, FROM user_cons_columns  
WHERE table_name='employees';
```

### Find the Solution for the following:

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 emp

Add constraint my\_emp\_id\_pk Primary Key(id);

2. Create a PRIMARY KEY constraint to the DEPT table using the ID column. 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 department. Name the constraint my\_emp\_dept\_id\_fk.

Alter table emp

Add constraint my\_emp\_dept\_id\_fk Foreign Key(dept  
References dept(id);



Alter table emp

add dept\_id Int;

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 chk-commission-positive check (commission > 0);

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

Use of WITH READ ONLY option.

Any attempt to perform a DML on any row in the view results in an oracle server error.

Try this code:

```
CREATE OR REPLACE VIEW empvu10(employee_number, employee_name, job_title)
AS SELECT employee_id, last_name, job_id
FROM employees
WHERE department_id=10
WITH READ ONLY;
```

Find the Solution for the following:

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 or Replace View employee\_vu As  
Select employee\_id, last\_name employee, department\_id  
from employees

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

Select \* from employees\_vu;

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

Set long 600  
Select view name, text  
from user\_views;

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

Select employee, department from employees\_vu;

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 dept 50 as  
 select employee\_id empno, last\_name employee,  
 department\_id - id deptno  
 from employees where department\_id = 50  
 with check option  
 constraint emp\_dept

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

describe dept 50  
 select \* from dept 50;

7. Attempt to reassign Matos to department 80.

update dept 50  
 set deptno = 80  
 where empno = '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.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	



```
SELECT last_name, salary*12 annsal, job_id, department_id, hire_date  
FROM employees  
ORDER BY annsal;
```

Example: 4

Sorting by Multiple columns

```
SELECT last_name, salary, job_id, department_id, hire_date  
FROM employees  
ORDER BY department_id, salary DESC;
```

Find the Solution for the following:

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

Select last\_name, salary  
from employees

where salary > 12000;

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

Select last\_name, department  
from employees

where employee\_number = 176;

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

Select last\_name, salary  
from employees

where salary not between 5000 and 12000;

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 (hint: between)

Select last\_name, job\_id, start\_date

from employees

where start\_date between Date '1998-02-20' and  
Date '1998-05-01',  
order by start\_date asc;

5. Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name. (hints: in, orderby)

```
select last_name, department_number  
from employees  
where department_number in (20, 50)  
order by last_name asc;
```

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. (hints: between, in)

```
select last_name as 'employee', salary as 'monthly salary'  
from employees  
where salary between 5000 and 12000  
and department_number in (20, 50)  
order by last_name asc;
```

7. Display the last name and hire date of every employee who was hired in 1994. (hints: like)

```
select last_name, hire_date from employees  
where hire_date like '1994%';
```

8. Display the last name and job title of all employees who do not have a manager. (hints: is null)

```
select last_name, job_title from employees  
where manager_id is null;
```

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 null, order by)

~~select~~ last\_name, salary, commission from employees where commission is not null order by salary commission

10. Display the last name of all employees where the third letter of the name is a (hints: like)

~~select~~ last\_name from employees where last\_name like ' \_a \_' and last\_name like ' \_a \_' ;

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

~~select~~ last\_name from employees where last\_name like ' \_a \_' and last\_name like ' \_e \_' ;

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. (hints: in, not in)

~~select~~ last\_name, job\_title, salary from employees where job\_title in ('Sales Representative', 'Stock Clerk', 'Sales Representative', 'Stock Clerk') and salary not in (2500, 3500, 7000)

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	



the same as their job title when they were initially hired (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;
```

#### Example

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

#### MINUS Operator Guidelines

- The number of columns and the data types of the columns being selected by the SELECT statements in the queries must be identical in all the SELECT statements used in the query. The names of the columns need not be identical.
- All of the columns in the WHERE clause must be in the SELECT clause for the MINUS operator to work.

#### Example:

Display the employee IDs of those employees who have not changed their jobs even once.

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

#### Find the Solution for the following:

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 department\_id from departments minus select department\_id from employees where job\_id = 'ST-CLERK';*

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 country\_id, country\_name from countries minus  
select country\_id, country\_name from countries join departments  
on country\_id = department\_id  
select country\_id  
select last\_name, dept\_id from employees union  
select dept\_name, dept\_id from departments*

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, dept_id from employee where dept_id = 10
union all select job_id, dept_id from employee where dept_id = 50
union all select job_id, dept_id from employee where dept_id = 20
```

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 employee where job_id =
(select job_id from jobs where emp_id = emp_id
order by start_date asc fetch first 1 rows
only);
```

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.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

This query was completed in earlier releases as follows:

```
SELECT e.last_name, e.department_id, d.department_name  
FROM employees e, departments d  
WHERE d.department_id = e.department_id (+);
```

### FULL OUTER JOIN

Example:

```
SELECT e.last_name, e.department_id, d.department_name  
FROM employees e  
FULL OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```

This query retrieves all rows in the EMPLOYEES table, even if there is no match in the DEPARTMENTS table. It also retrieves all rows in the DEPARTMENTS table, even if there is no match in the EMPLOYEES table.

### Find the Solution for the following:

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.department_name  
from employees e, departments d  
where e.department_id = d.department_id;
```

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

```
select distinct job_id, location_id  
from employees, departments  
where employees.department_id = departments.department_id  
and employees.department_id = 80;
```

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.department_name, d.location_id, c  
from employees, departments, locations  
where e.department_id = d.department_id  
and  
d.location_id = l.location_id  
and e.commission_pct is not null;
```



2. Display the employee last name and department name for all employees who have an at (lowercase) in their last names. P

~~select last\_name, department\_name, d.department\_id~~  
~~from employees, departments~~  
~~where employees.department\_id = departments.department\_id~~

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, e.job, e.department\_id, d.department\_name~~  
~~from employees e join departments d~~

~~on (e.department\_id = d.department\_id)~~

~~where (e.location\_id = 8)~~

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 w.last\_name "Employee", w.employee\_id "Emp#"~~

~~m.last\_name "Manager", m.employee\_id "Mgr#"~~

~~from employees w join employees m~~

~~on w.manager\_id = m.employee\_id;~~

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

~~select w.last\_name "Employee", w.employee\_id "Emp#"~~

~~m.last\_name "Manager", m.employee\_id "Mgr#"~~

~~from employees w~~

~~left outer join employees m~~

~~on (w.manager\_id = m.employee\_id);~~

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.department\_id department, e.last\_name~~

~~c.last\_name colleague~~

~~from employees e join employee c~~

~~on (e.department\_id = c.department\_id)~~

~~where e.employee\_id < c.employee\_id~~

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

~~select e.last\_name, e.job\_id, d.department\_name,~~

~~e.salary, j.grade\_level~~

~~on (e.department\_id = d.department\_id)~~

~~join job\_grades~~

~~on (e.salary between j.lowest\_sal and j.highest\_sal);~~

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

select e.last name, e.hire\_date from employee where e.hiredate > (select hire date from employees where last name = ('Davies'));

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 e.last name as employee, e.hire\_date as emp\_hired, m.last name as manager, m.hire\_date as mgr\_hired from employees e join employees m on e.manager\_id = m.employee\_id where e.hire\_date < m.hire\_date

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

WHERE emp.employee\_id NOT IN (SELECT mgr.manager\_id FROM employees mgr);

Notice that the null value as part of the results set of a subquery is not a problem if you use the IN operator. The IN operator is equivalent to =ANY. For example, to display the employees who have subordinates, use the following SQL statement:

```
SELECT emp.last_name  
FROM employees emp  
WHERE emp.employee_id IN (SELECT mgr.manager_id FROM employees mgr);
```

Display all employees who do not have any subordinates:

```
SELECT last_name FROM employees  
WHERE employee_id NOT IN (SELECT manager_id FROM employees WHERE manager_id  
IS NOT NULL);
```

Find the Solution for the following:

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 = (SELECT department_id FROM employees  
WHERE last_name = 'Zlotkey') and last_name != 'Zlotkey';
```

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;
```

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 *n*.

```
SELECT employee_id (last_name FROM  
employees WHERE department_id IN  
(SELECT department_id FROM employees WHERE  
last_name LIKE '%n%'));
```



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 last_name, department_number, job_id  
from employees where department_id = 17  
select department_id from departments where  
location
```

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

```
select last_name, salary from employees where  
manager_id = (select employee_id from employees  
where last_name = 'King');
```

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

```
select department_id, last_name, job_id from  
employees where department_id = (select department_  
id from departments where department_name = 'Executive');
```

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 employee_id, last_name, salary from  
employees where salary > (select avg(salary) from employees  
AND department_id in (select department_id from  
employees where last_name like '%u%'));
```

Group functions can be nested to a depth of two. The slide example displays the maximum average salary.

```
SELECT MAX(AVG(salary)) FROM employees GROUP BY department_id;
```

#### Summary

In this exercise, students should have learned how to:

- Use the group functions COUNT, MAX, MIN, and AVG.
- Write queries that use the GROUP BY clause
- Write queries that use the HAVING clause

```
SELECT column, group_function  
FROM table  
[WHERE condition]  
[GROUP BY group_by_expression]  
[HAVING group_condition]  
[ORDER BY column];
```

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~~ False
2. Group functions include nulls in calculations.  
True/False ~~False~~ True
3. The WHERE clause restricts rows prior to inclusion in a group calculation.  
True/False ~~True~~ False

The HR department needs the following reports:

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.

Soln  $\text{Round}(\text{max}(\text{salary}))$  as maximum,  $\text{Round}(\text{min}(\text{salary}))$  as minimum

$\text{Round}(\text{sum}(\text{salary}))$  as sum,  $\text{Round}(\text{avg}(\text{salary}))$  as Average from employees

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

Soln  $\text{job\_id}$ ,  $\text{Round}(\text{max}(\text{salary}))$  as maximum,  $\text{Round}(\text{min}(\text{salary}))$  as minimum,  $\text{Round}(\text{sum}(\text{salary}))$  as sum,  $\text{Round}(\text{avg}(\text{salary}))$  as Average, from employees group by job\_id;

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 job_id, count(*) as number_of_employees from
where job_id = 12 job_title group by job_id;
```

7. Determine the number of managers without listing them. Label the column Number of Managers. *Hint: Use the MANAGER\_ID column to determine the number of managers.*

```
select count(distinct manager_id) as Number of
managers" from employees
where manager_id is not null;
```

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

```
select max(salary) - min(salary) as Difference from employees
```

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 minimum_salary
from employees where
manager_id is not null order by manager_id
having min(salary) > 6000 order by min salary Desc;
```

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 count(*) as total_employees, sum(case when
to_char(hire_date, 'YYYY') = '1993' then 0 End) as
Hired_1995, sum(case when to_char(hire_date, 'YYYY')
= '1995' then 1 else 0 End) as
Hired_1996, sum(case when to_char(hire_date, 'YYYY')
= '1996' then 1 else 0 End) as
Hired_1997, sum(case when to_char(hire_date, 'YYYY')
= '1997' then 1 else 0 End) as
Hired_1998
```



11. select job\_id, sum(case when department\_id = 20 then salary else 0 end) as Dept\_20\_Salary, sum(case when department\_id = 50 then salary else 0 end) as Dept\_50\_Salary, sum(case when department\_id = 80 then salary else 0 end) as Dept\_80\_Salary, sum(case when department\_id = 90 then salary else 0 end) as Dept\_90\_Salary from employees where department\_id in (20, 50, 80, 90) group by job\_id;

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.

13. select d.department\_name as department\_name, l.location, count(e.employee\_id) as number\_of\_people, round(avg(e.salary), 2) as average\_salary from employees e join department of d on d.department\_id = e.department\_id join location l on l.location\_id = e.location\_id group by d.department\_name, l.location;

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

# PROGRAM 1

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

```

Declare
    v_employee_id employee.emp_id := 110;
    v_salary employee.salary %Type;
    v_incentive number;
    v_incentive_pct constant number := 0.10
begin
    select salary into v_salary from employees
    where emp_id = v_employee_id;
    v_incentive = v_salary * v_incentive_pct;
    DBMS_output.put_line('Incentive for empID',
        v_employee_id || ' is ||
        v_incentive);
    
```

# PROGRAM 2

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

```

Declare
    v_test_variable number := 100;
begin
    execute immediate 'create / rename function
    "my function" return number is begin return 1; end';
    Declare
        v_result number;
    begin
        v_result = my function
        DBMS_output.put_line('Result: ' || v_result);
    Exception
        when others then
        DBMS_output.put_line('Error, raiseError');
    end
    DBMS_output.put_line('value' || (mytest-variable));
    
```

### PROGRAM 3

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

```

DECLARE
v_employee_id employees.employee_id%TYPE := 122;
v_new_salary employees.salary%TYPE;
v_current_salary employees.salary%TYPE;
v_adj number := 500;

BEGIN
select salary into v_current_salary
employee_id = v_employee_id;

v_new_salary = v_current_salary + v_adj;

update employees set salary = v_new_salary
employee_id = v_employee_id;

commit;

DBMS_OUTPUT.PUT_LINE('Salary updated!');

END

```

### 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 update_emp_status(p_emp_id
employees emp_id%TYPE) AS
v_sal employees.salary%TYPE;
v_dept_id employees.department_id%TYPE;

BEGIN
select sal, dept_id into v_sal, v_dept_id
where emp_id = p_emp_id;

IF EXCEPTION THEN
DBMS_OUTPUT.PUT_LINE('Error: ' || v_sal || v_dept_id);
END;

DBMS_OUTPUT.PUT_LINE('Value: ' || v_sal || v_dept_id);

```



# PROGRAM 5

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

Declare

v - employee\_id employees.employee\_id %type := 22;

v - new\_salary employees.salary %type

v - current\_sal employees.salary %type;

v - add number = 500;

begin

select salary into v - current from employees where

employee\_id = v - employee\_id;

v - new\_sal = v - current - sal v - add;

update employees set salary = v - new\_sal

where employee\_id = v - employee\_id;

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.

create or replace procedure update\_emp\_status(p\_emp

employees emp\_id %type) as

v\_sal employees.salary %type

v\_dept\_id employees.department\_id %type

begin

select sal, dept\_id into v\_sal, v\_dept\_id from

employees where emp\_id = p\_emp\_id;

if v\_sal is null and v\_dept\_id is not null then

update employees set status = 'Active' where emp\_id =

p\_emp\_id;

DBMS\_output.put\_line('Status updated to Active');

else DBMS\_output.put\_line('Status can't be updated');

end if;

#### 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.

~~Declare~~

v - emp - name emp.name / type ;

begin

for rec in (select name from emp where  
~~for~~ name like it - 1. escape '\')

loop

DBMS\_output.put\_line(rec.name);

End loop;

End;

#### PROGRAM 8

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

create or replace Procedure cal-inc (p - salary in  
number) as v in c - num ;

begin

if p - sal - and > = 5000 then

v - inc := p - sal - and \* 0.15 ;

else

v - inc := p - sal - and \* 0.05

end if;

DBMS\_output.put\_line (p - and || v - inc);

DECLARE

V - emp - count number;  
V - vacancies constant number = 45

begin

select count (+) into V - emp - count from emp where  
dept - id = 50;

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.

if DBMS - output . put - line ( v emp - count );

V - emp - count < V - vacancies THEN

else DBMS . output . put - line ( 'So vacancies available' );

End if DBMS - output . put - line ( 'No vacancies' );

End;

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 - dept - id number := 50;

V - emp - count number;

V - total - pos number = 100;

V - new - number;

begin

select count (\*) into V - emp - count from

dept - id = V - dept - id;

V - vacancies = V - total - pos - V - emp count;

if V - vac > 0 then



# PROGRAM 11

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

```

DECLARE
begin
    for rec in (select empid, job-title, hire date, salary from emp
    where emp-id = jobs-id)
    loop
        DBMS_OUTPUT.PUT_LINE(rec.emp-id || rec.emp-name)
        || char(rec.hiredate, 'DD-MON-YY') || rec.salary
    end loop;
end;

```

# PROGRAM 12

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

```

DECLARE
begin
    for rec in (select e.empid, e.fname || ' ' ||
    e.lname as emp, d.dept-name from employees
    join departments d on e.dept-id = d.dept-id)
    loop
        DBMS_OUTPUT.PUT_LINE(rec.emp-id || rec.emp-
        name || rec.dept-name);
    end loop;
end;

```

PROGRAM 13  
Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```

Declare
begin
  for rec in (select job_id, job_title min sal
              from jobs)
  loop
    DBMS-Output-put_line (rec.job_id || rec.job_title ||
                          rec.min_sal);
  end loop;
end

```

PROGRAM 14  
Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```

Declare
begin
  for rec in (select e.emp_id, e.first_name || ' ' ||
                    e.last_name as emp_name, jh.start_date from
                    employees e join job_history jh on e.emp_id =
                    jh.emp_id)
  loop
    DBMS-Output-put_line (rec.emp_id,
                          rec.emp_name,
                          to_char (rec.start_date, 'DD-MON-YY'));
  end loop;
end

```

End

V - employee - id employee s. employee - id 1. Type;  
V - first name employees first name 1. Type

V - last name employees last name 1. Type

Cursor EMP\_CUR IS select e. employee - id, e. first  
name, e. last name, e. hire date from employee join e on last name

PROGRAM 15  
Write a PL/SQL program to display the employee IDs, names, and job history and dates of all  
employees.

begin

for emp\_rec in emp\_cursor loop

V - employee - id := emp\_rec.employee - id

V - first name = emp\_rec.first name;

V - last name := emp\_rec.last name;

DBMS\_output.put\_line ('Employee ID: ' ||

V - employee - id || ' Name: ' || V - first name || ' ' ||

V - last name || ', Job history End

end loop;

End;

Evaluation Procedure	Marks awarded
PL/SQL Procedure (5)	
Program/Execution (5)	
Viva (5)	
Total (15)	
Faculty Signature	



Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
Create or Replace function cal_fac(n number)
Return number is fac number := 1;
begin
  if n = 0 or n = 1 then
    return fac;
  else
    for i in 1...n loop
      fac = fac * i;
    end loop;
    return fac;
  end if;
end;

/declare
num number := 5
result number;

begin
  result := cal_fac(num)
  DBMS_output.put_line(result);
end;
```

## Program 2

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

```

create or replace procedure bookinfo(
    P_book_id in number,
    P_author out varchar2,
    P_avail_top in out number);
begin
    select author, cop from library where
        book_id = P-book_id
    and P_book_id <= 1000000;
end;

```

TO WRITE A PL/SQL BLOCK TO DISPLAY THE EMPLOYEE ID AND EMPLOYEE NAME WHERE DEPARTMENT NUMBER IS 11 USING EXPLICIT CURSORS

```

1 declare
2 cursor cent is select eid, sal from sscmp where dno=11;
3 ecode sscmp.eid%type;
4 esal sscmp.sal%type;
5 begin
6 open cent;
7 loop
8 fetch cent into ecode, esal;
9 exit when cent%notfound;
10 dbms_output.put_line('Employee code and employee salary are ' || ecode || ' and ' || esal);

```

End

book\_id number = 2

author

varchar2(100);

begin

select info(book\_id, author, avail\_top)

### 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 P-dep-par before Delete on dept
for each row
declare
    C-count number;
begin
    select count(*) into C-count from emp where
    dept-id = old.dept-id;
    if C-count > 0 then
        raise_application_error(-2000, 'cant Delete dept');
    end if;
end;

```

### 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 check dup-name
before insert
on users
declare
    V-count number;
begin
    select count(*) into V-count from users
    where username = New.username;
    if V-count > 0 then
        raise_error(-200, 'Duplicate, username found');
    end if;
end;

```



### 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 tot_sal Before Insert on
salaries for each row
Declare
    tot_sal Number;
    Threshold constant number := 10000;
Begin
    select sum(sal) into tot_sal from sal;
    if tot_sal > tot_sal then --> Threshold then
        Raise_application_error(-2000, 'cannot insert');
    end;

```

### 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 log_emp_changes
After update of sal, dept_id on emp
for each row
Begin
    Insert into emp_audit (emp_id, emp_dept_id, new_sal,
    old_sal, new_dept_id, old_dept_id)
    values (emp_id, emp_dept_id, new_sal, old_sal,
    new_dept_id, old_dept_id);
End;

```

# 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 log-audit-emp
for each row
begin
    insert into audit_log (action_type;
    case, when inserting the insert when updating then update;
    else 'Delete' END) employees, last when updating or deleting
use when updating or inserting the: row: END);
end
    
```

# Program 7

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 or replace trigger trg
after insert or delete
for each row
declare
    total number;
begin
    select null(max) running_total, 0: new and
    update sales SET running_total = total + total - total -
    new id;
end;
    
```

# Program 8

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 or replace Trigger log
before insert on orders
for each Row
Declare
V - stock - level number;
Begin
select stock_level into v-stock-level from
inventory where itemid = :new.itemid;
if v-stock_level < :NE quantity then
end if
End;

```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	



date: Date()

})

Structure of 'restaurants' collection:

### EXERCISE 18

```
{
  "address": {
    "building": "1007",
    "coord": [-73.856077, 40.848447],
    "street": "Morris Park Ave",
    "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
    { "date": {"$date": "1393804800000"}, "grade": "A", "score": 2 },
    { "date": {"$date": "1378857600000"}, "grade": "A", "score": 6 },
    { "date": {"$date": "1358985600000"}, "grade": "A", "score": 10 },
    { "date": {"$date": "1322006400000"}, "grade": "A", "score": 9 },
    { "date": {"$date": "1299715200000"}, "grade": "B", "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

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

*db.movers.find({'\$year': 10938})*

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.movers.find({'year': {'\$gt': 12038}})*

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.movies.find({'grades': {'$eq': 'A'}})`

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.movies.find({'directors': 'William H. L. Williams'})`

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})`

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

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

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', 'borough': -1})`

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

`db.restaurants.find({'address.street': {'$exists': true}})`

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'}})

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': {'\$elementSize': {'\$score': {'\$mod': [7, 0]}}}} {'restaurant\_id': 1, 'name': 1, 'grades': 1})

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'}}) {'name': 1, 'borough': 1, 'address.coord': 1, 'longitude': 1}

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'}}) {'name': 1, 'borough': 1, 'address.coord': 1, 'longitude': 1, 'cuisine': 1}

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}})

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},  
location: {'\$in': ['Manhattan', 'Brooklyn']}})

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},  
location: {'\$in': ['Manhattan', 'Brooklyn']},  
cuisine: {'\$ne': 'American'}})

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},  
location: {'\$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.score': {'\$eq': 2}, {'grades.score': {'\$eq': 6}}})

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(
  { grade: { $or: [ { score: 2 }, { score: 6 } ] },
    borough: "Manhattan" } );

```

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(
  { grade: { $or: [ { score: 2 }, { score: 6 } ] },
    borough: { $in: [ "Manhattan", "Brooklyn" ] },
    cuisine: { $ne: "American" } },

```

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(
  { grade: { $or: [ { score: 2 }, { score: 6 } ] },
    borough: { $in: [ "Manhattan", "Brooklyn" ] },
    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( { grade: { \$in: [5, 6] }, location: { \$in: ['Manhattan', 'Brooklyn'] }, cuisine: { \$not: ['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( { grade: { \$in: [2, 6] } })

### Sample document of 'movies' collection

```
{
  "_id": ObjectId("573a1390f29313caabed42e8"),
  "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/MV5BMjU3NjE5NzYyYTYyNS00MDVmlWlwyYjg6MmYyWywXZD  
YyNzU2XkE5XkFpQcGdQXVYyNzQzNzQxNzIj@\_V1\_SY1000\_SX677\_AL\_ipg',  
title: 'The Great Train Robbery',

fulplot: "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: ISODate("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: ISODate("2015-08-08T19:16:10.000Z")
}
1. Find all movies with full information from the 'movie'
1893.

```

de . movies . find (f you : 189937

do. movies. find (1 ~~great~~ ranking: 1594  
: 12642)

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

db.movies.find({'genre': "Short"})

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({'director': "William K.L. Dickson"}, {"\_id": 1})

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

db.movies.find({'country': "USA"}, {"\_id": 1})

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

db.movies.find({'\_id': 1000000})

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

db.movies.find({'rated': 'unrated'})

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}})

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\_rating': {'\$gt': 4}})

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

db.movies.find({'awards\_nominations': {'\$gt': 1}},  
{'\_id': 1, 'lang': 1, 'release': 1, 'director': 1, 'countries': 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({'cast': 1, 'directors': 1, 'writers': 1, 'release': 1, 'year': 1, 'countries': 1})



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('$released:1 $date ("1993-05-09 00:00:00Z")')
title:1, lang:1, released:1, director:1, writers: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('$title:1 $year: "1893" $options: "1"')
title:1, lang:1, released:1, directors: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.

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

1) Create Sequence DEP-ID\_SEQ  
START WITH 200  
INCREMENT BY 10  
MAX VALUE 1000  
NO CYCLE;

2) select  
dept\_name, max sal, min sal, last name  
from  
users, sequences  
where  
seq\_name = 'DEP-ID\_SEQ';

3) insert into dept (dept\_id, dept\_name) values  
(dept\_id\_seq.nextval, 'Education');  
insert into dept (dept\_id, dept\_name) values  
(dept\_id\_seq.nextval, 'Admin');  
select \* from dept;

4) Create index indx\_id on Emp(dept\_id);

5) select  
index\_name,  
sequence  
from  
users, indexes  
where  
seq\_name = 'Emp';

Q1) Create session privilege to log onto the Oracle server. This is a system privilege.

2) Create table privilege

3) Owner of the table (user who created it)

4) Use roles to group system privileges

5) Alter user user\_name IDENTIFIED BY new password

6) Grant select, insert, update, delete on departments to other user;

7) select \* from Departments

8) insert into Departments (DeptID, DeptName, values (500, 'education');

insert into Departments (DeptID, DeptName, values (540, 'HQ');

9) select \* from user tables;

10) Reverse select on Departments from other user;

11) Delete from Departments where DeptID = 500;