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 int, NAME varchar2(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 int(7), LAST_NAME varchar2(25), FIRST_NAME varchar2(25), DEPT_ID int(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 coloumns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

```
Create table EMPLOYEES2 (Employee-id int, First-Name varchar(50),
                         Last-Name varchar(50), Salary int, Dept-id int);
```

5. Drop the EMP table.

```
Drop table EMP
```

6. Rename the EMPLOYEES2 table as EMP.

```
Alter table Employee2
Rename To EMP;
```

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

```
Comment on table DEPT IS
Comment on table EMP. IS
```

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	

table

1) Create MY-EMPLOYEE (ID int(4), Last_name varchar(25), First_name varchar(25), Userid varchar(25), Salary int (9,2));

2) Insert into MY-Employee (ID, Last_name, First_name, userid, Salary)
values (1, 'Patel', 'Ralph', 'rpatel', 895),
(2, 'Dancs', 'Betty', 'bdancs', 860);

Find the Solution for the following:

1. Create MY_EMPLOYEE table with the following structure

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

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(ID, Last_name, first_Name, Userid, Salary)
values (3, 'Biri', 'Ben', LOWER (SUBSTR ('Ben',1,1)|| SUBSTR ('Biri',1,7)), 1100),
  (4, 'Newman', 'Chad', LOWER (SUBSTR ('Chad',1,1)|| SUBSTR ('Newman',1,7)), 750);
```

5. Make the data additions permanent.

```
COMMIT;
```

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

```
Update MY-EMPLOYEE
SET Last-Name = 'Drexler'
WHERE ID = 3;
```

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

```
Update MY-EMPLOYEE
SET Salary = 1000
WHERE Salary < 900;
```

8. Delete Betty dancs from MY _EMPLOYEE table.

```
Delete from MY-EMPLOYEE
WHERE First_name = 'Betty' AND Last_name = 'Dancs';
```

9. Empty the fourth row of the emp table.

```
Update MY-EMPLOYEE
SET Last_name = NULL,
First_name = NULL,
UserId = NULL,
Salary = NULL,
Where ID = 4';
```

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

Example:

Assume table TEST1 with the following structure

CREATE TABLE test1 (pk number PRIMARY KEY, fk number, col1 number,col2 number, CONTRAINT fk_constraint FOREIGN KEY(fk) references test1, CONSTRAINT ck1 CHECK (pk>0 and col1>0), CONSTRAINT ck2 CHECK (col2>0));

An error is returned for the following statements

ALTER TABLE test1 DROP (pk);

ALTER TABLE test1 DROP (col1);

The above statement can be written with CASCADE CONSTRAINT

ALTER TABLE test 1 DROP(pk) CASCADE CONSTRAINTS;
(OR)
ALTER TABLE test 1 DROP(pk, fk, col1) 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 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.

Create table Emp(ID NUMBER (4),
Constraint my-emp-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.

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)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	

3) Alter table EMP
Add DEPT_ID Number (4);

Alter table EMP
Add Constraint CHKINT
my-emp-dept-id-fk

Foriegn key (DEPT_ID)
References DEPT(ID);

Example:3

SELECT last_name, job_id, salary, commission_pct FROM employees;

Example:4

SELECT last_name, job_id, salary, 12*salary*commission_pct FROM employees;

Using Column Alias

- To rename a column heading with or without AS keyword.

Example:1
SELECT last_name AS Name
FROM employees;
Example: 2
SELECT last_name "Name" salary*12 "Annual Salary "
FROM employees;

Concatenation Operator

- Concatenates columns or character strings to other columns
- Represented by two vertical bars (||)
- Creates a resultant column that is a character expression

Example:
SELECT last_name||job_id AS "EMPLOYEES JOB" FROM employees;

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 deparment_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; FALSE

Queries

Select employee_id, last_name, Salary *12 as "ANNUAL SALARY" From employee

2. Show the structure of departments the 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_id, last_name, job_id, hire_date
from employees;

4. Provide an alias STARTDATE for the hire date.

Select employee_id, last_name, job_id, hire_date
as "STARTDATE" from employees;

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

select distinct job_id from employees;

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;

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||','||last_name||','||job_id||','||salary as The_output from employees;

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

Example:2

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

Example:3
Sorting by column alias

```
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_id
FROM employees
WHERE employee_id = 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. (hints: 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.(hints: between)

```
SELECT last_name, job_id, hire_date
FROM employees
WHERE hire_date BETWEEN TO_DATA ('20-FEB-1998', 'DD-MM-YYYY')
AND
```

```
TO_DATE ('01-MAY-1998', 'DD-MON-YYYY')
ORDER BY hire_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_id
FROM employees
WHERE depastment_id 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 depastment_id 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 TO_CHAR (hire_date, 'YYYY')='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_id
FROM employees
WHERE manager_id ISNULL;
```

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 DESC, commission_pct DESC;
```

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%';
```

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_id, salary
FROM employees
WHERE job_id IN ('SA_REP', 'ST_CLERK')
AND salary NOT IN (2500, 3500, 7000);
```

13. Display the last name, salary, and commission for all employees whose commission amount is 20%.(hints:use predicate logic)

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct = 0.2;
```

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

Find the Solution for the following:

1. Write a query to display the current date. Label the column Date.

```
SELECT SYSDATE AS "Date"
FROM DUAL;
```

2. The HR department needs a report to display the employee number, last name, salary, and increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

```
SELECT employee_id, last_name, salary, ROUND (salary * 1.155) AS
                                                "New Salary"
FROM employees;
```

3. Modify your query lab_03_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

```
SELECT employee_id, last_name, salary, ROUND(salary *1.155) AS "New Salary",
                                ROUND (salary * 1.155) - salary AS "Increase"
FROM employees;
```

4. Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results by the employees' last names.

```
SELECT INITCAP(last_name) AS "Formatted Name", LENGTH (last_name) AS "Name Length"
FROM employees
WHERE UPPER (SUBSTR (last_name, 1,1)) in ('J', 'A', 'M')
ORDER BY last_name;
```

5. Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

```
SELECT INITCAP (last_name) AS "Formatted Name", LENGTH (last_name) AS "NAME Length"
FROM employees
WHERE UPPER (SUBSTR (last_name, 1,1)) =UPPER ('&start_letter')
ORDER BY last_name;
```

6. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

```
SELECT last_name,
        CEIL (MONTHS_BETWEEN (SYSDATE, hire_date))
                AS "MONTH_WORKED"
FROM employees
ORDER BY "MONTH_WORKED";
```

Note: Your results will differ.

7. Create a report that produces the following for each employee:
<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

```
SELECT last_name || 'earns' || salary || 'monthly but wants' || (salary *3) AS "Dream Salaries"
FROM employees;
```

8. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the $ symbol. Label the column SALARY.

```
SELECT last_name,
       LPAD(TO_CHAR(salary), 15, '$') AS "salary"
FROM employees;
```

9. Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

10. Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

```
SELECT last_name,
       hire_date,
       TO_CHAR (hire_date, 'day') AS "DAY"
FROM employees
ORDER BY TO_CHAR (hire_date, 'D');
```

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

9) SELECT last_name,

TO_CHAR(hire_date, 'Day, "the" Ddspth "of" Month, YYYY') AS "Hire Date",
TO_CHAR(
NEXT_DAY(ADD_MONTHS(hire_date, 6)-1, 'MONDAY'),
) AS "REVIEW"
FROM employees;

Practice Questions

Introduction to Functions

1. For each task, choose whether a single-row or multiple row function would be most appropriate:
   a. Showing all of the email addresses in upper case letters
   b. Determining the average salary for the employees in the sales department
   c. Showing hire dates with the month spelled out *(September 1, 2004)*
   d. Finding out the employees in each department that had the most seniority (the earliest hire date)
   e. Displaying the employees' salaries rounded to the hundreds place
   f. Substituting zeros for null values when displaying employee commissions.

2. The most common multiple-row functions are: AVG, COUNT, MAX, MIN, and SUM. Give your own definition for each of these functions.

3. Test your definitions by substituting each of the multiple-row functions in this query.
SELECT FUNCTION(salary)
FROM employees
Write out each query and its results.

This query retrieves all rows in the EMPLOYEES table, even if there is no match in the DEPARTMENTS table. It alsIso 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
JOIN departments d ON 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 e.job_id, l.location_id, l.city
FROM employees e
JOIN departments d ON e.department_id= d.department_id
JOIN locations l ON d.location_id = l.location_id
WHERE e.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, l.location_id, l.city
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id=l.location_id
WHERE e.commission_pct IS NOT NULL;
```

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

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_id, d.department_id, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN location l ON d.location_id = l.location_id
WHERE l.city = 'Toronto';
```

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 e.last_name AS Employee, e.employee_id AS "Emp#",
       m.last_name AS manager, m.employee_id AS "Mgr#"
FROM employees e
LEFT JOIN employees m ON e.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 e.last_name AS Employee, e.employee_id AS "Emp#",
       m.last_name AS Manages, m.employee_id AS "Mgr#"
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id
ORDER BY e.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 e1.last_name AS "Employee", e1.department_id AS "Dept#", e2.last_name AS "Colleague"
FROM employees e1
JOIN employees e2 ON e1.department_id = e2.department_id
WHERE e1.employee_id <> e2.employee_id
ORDER BY e1.department_id, e1.last_name;
```

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_grades;
```

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 > (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 "Manages", 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)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	

4)

```
SELECT e.last_name, d.department_name
FROM employeese
JOIN departments d ON e.department_id = d.department_id
WHERE e.last_name LIKE '%a%';
```

SELECT department_id, AVG(salary) FROM employees GROUP BY department_id
HAVING max(salary)>10000;

Example displays the job ID and total monthly salary for each job that has a total payroll exceeding $13,000. The example excludes sales representatives and sorts the list by the total monthly salary.

SELECT job_id, SUM(salary) PAYROLL FROM employees WHERE job_id NOT LIKE '%REP%'
GROUP BY job_id HAVING SUM(salary) > 13000 ORDER BY SUM(salary);

Nesting Group Functions

Display the maximum average salary:
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

2. Group functions include nulls in calculations.
True/False

3. The WHERE clause restricts rows prior to inclusion in a group calculation.
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

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

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

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 COUNT (*) AS "Number of Employees"
FROM employees
WHERE Job_id = '&Job_Title';
```

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 manger_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" DES;
```

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') = '1995' THEN 1 ELSE 0 END) AS
                                                        "Hired in 1995",
SUM (CASE WHEN TO_CHAR (hire_date, 'YYYY') = '1996' THEN 1 ELSE 0 END) AS "Hired in
                                                                    1996",
```

```
SUM (CASE WHEN TO_CHAR(hire_date, 'YYYY')='1997' THEN 1 ELSE 0 END) AS "Hired in 1997",
SUM (CASE WHEN TO_CHAR(hire_date, 'YYYY')='1997' THEN 1 ELSE 0 END) AS "Hired in 1998",
FROM employees;
```

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
        Job_id,
SUM (CASE WHEN department_id= 20 THEN Salary ELSE 0 END) AS Dept_20,
SUM (CASE WHEN department_id = 50 THEN Salary ELSE 0 END) AS Dept_50,
SUM (CASE WHEN department_id= 80 THEN Salary ELSE 0 END) AS Dept_80,
SUM (Salary) AS Total_Salary
```

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.department_name||' - '||l.city AS "Name - Location",
        COUNT(e.employee_id) AS "Number of people",
        ROUND (AVG (e.Salary),2) AS Salary
FROM departments d
JOIN location l ON d.location_id=l.location_id
LEFT JOIN employees e ON d.department_id=e.department_id
GROUP BY d.department_name
        ,l.city;
```

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

Displays employees who are not IT programmers and whose salary is less than that of any IT programmer. The maximum salary that a programmer earns is $9,000.

< ANY means less than the maximum. >ANY means more than the minimum. =ANY is equivalent to IN.

Using the ALL Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ALL (SELECT salary FROM employees WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Displays employees whose salary is less than the salary of all employees with a job ID of IT_PROG and whose job is not IT_PROG.

- ALL means more than the maximum, and <ALL means less than the minimum.

The NOT operator can be used with IN, ANY, and ALL operators.

Null Values in a Subquery

```
SELECT emp.last_name FROM employees emp
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 e.last_name , e.hire_date
FROM employees e,
Join employees Supervisor . department_id on e.department_id
= Supervisor . department_id WHERE Supervisor . last_name = employee_last_name.
```

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 *u*.

```
select employee_id, last_name,
from employees where department_id
   Select distinct department_id from employees
   Where last_name like '%u%';
```

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
Join department d ON e.department_id = d.department_id
WHERE d.location_id = 1700;
```

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 = (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 e.department_id, e.last.name, e.job_id
from employees e
Join departments d on e.department_id = d.department_id
WHERE d.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 DISTINCT department_id
FROM employees
WHERE last_name like '%u%');
```

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

• 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
WHERE Country_id IN(
  SELECT Country_id FROM Countries MINUS
  SELECT Country_id FROM locations );
```

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 =10 UNIONALL
SELECT Job_id ,department_id FROM employees WHERE department_id=50
SELECT Job_id , department_id FROM employees WHERE department_id=20 ;
```

UNIONALL

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 e.employee_id , e.job_id
FROM employees e
JOIN Job_history jh ON e.employee_id =jh.employee_id
WHERE e.Job_id = jh.Job_id
AND e.hire_date < jh.Start_date;
```

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, NULL AS department_name
FROM EMPLOYEES

UNION ALL
SELECT NULL AS last_name, department_id, department_name
FROM DEPARTMENTS;
```

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

- Group By clause
- Distinct keyword
- Columns contain by expressions
- NOT NULL columns in the base table that are not selected by the view

Example: (Using the WITH CHECK OPTION clause)

```
CREATE OR REPLACE VIEW empvu20
 AS    SELECT *
 FROM employees
 WHERE department_id=20
 WITH CHECK OPTION CONSTRAINT empvu20_ck;
```

Note: Any attempt to change the department number for any row in the view fails because it violates the WITH CHECK OPTION constraint.

Example – (Execute this and note the error)

```
UPDATE empvu20 SET department_id=10 WHERE employee_id=201;
```

Denying DML operations

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 VIEW EMPLOYEE_VU AS
SELECT employee_number , employee_name AS EMPLOYEE, department_
number FROM employees;
```

2. Display the contents of the EMPLOYEES_VU view.

```
SELECT * FROM EMPLOYEE_VU;
```

3. Select the view name and text from the USER_VIEWS data dictionary views.

```
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, DEPTNO 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 DEPT50 AS
SELECT EMPNO, LAST_NAME AS EMPLOYEE, DEPARTMENT_ID AS
FROM employees                                    DEPTNO
WHERE DEPARTMENT_ID = 50
WITH CHECK OPTION;
```

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

```
DESCRIBE DEPT50;
SELECT * FROM DEPT50;
```

7. Attempt to reassign Matos to department 80.

```
UPDATE employees
SET DEPARTMENT_ID = 80
WHERE LAST_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.DEPARTMENT_NAME AS "Department",
    e.SALARY AS "Salary",
    J.GRADE AS "GRADE"
FROM
    employees e
JOIN
    departments d ON e.DEPARTMENT_ID = d.department_ID
JOIN
    Job_grades J ON e.SALARY BETWEEN J.LOWSALARY AND J.
                                    HIGH_SALARY
```

EXERCISE 12

Intro to Constraints; NOT NULL and UNIQUE Constraints

Global Fast Foods has been very successful this past year and has opened several new stores. They need to add a table to their database to store information about each of their store's locations. The owners want to make sure that all entries have an identification number, date opened, address, and city and that no other entry in the table can have the same email address. Based on this information, answer the following questions about the global_locations table. Use the table for your answers.

Global Fast Foods global_locations Table						
NAME	TYPE	LENGTH	PRECISION	SCALE	NULLABLE	DEFAULT
Id						
name						
date_opened						
address						
city						
zip/postal code						
phone						
email						
manager_id						
Emergency contact						

1. What is a "constraint" as it relates to data integrity?
A Constrain is a rule enforced by database to maintain data integrity and accuracy. It restricts the type of data that can be inseted into table.
2. What are the limitations of constraints that may be applied at the column level and at the table level? column level Constraints cannot reference other colums whereas table level Constraints can.
3. Why is it important to give meaningful names to constraints?
It makes debugging easier
it helps the identity purpose of each constraint
it improves maintability
4. Based on the information provided by the owners, choose a datatype for each column. Indicate the length, precision, and scale for each NUMBER datatype.

Id. manager_id - Number(s) name, email, emergency_contact - VARCHAR(50)
address - VARCHAR(100) City - VARCHAR (30)

5. Use "(nullable)" to indicate those columns that can have null values. Zip_postal_code - VARCHAR(15)
Phone - VARCHAR2(15)

Zip_Postal_Code, Phone, email, manager_id, emergences_Contact.

6. Write the CREATE TABLE statement for the Global Fast Foods locations table to define the constraints at the column level.

create table global_locations (id NUMBER (6) Const gl_id_pk primary key, name VARCHAR2 (50), CONSTRAINT gl_name_nn NOT NULL, date_opened DATE CONSTRAINT gl_date_nn NOT NULL, address VARCHAR2 (100), CONSTRAINT gl_email_id UNIQE, manager_id Number (6).

7. Execute the CREATE TABLE statement in Oracle Application Express.

TABLE GLOBAL_LOCATIONS Created

8. Execute a DESCRIBE command to view the Table Summary information.

DESC global_locations;

9. Rewrite the CREATE TABLE statement for the Global Fast Foods locations table to define the UNIQUE constraints at the table level. Do not execute this statement.

NAME	TYPE	LENGTH	PRECISION	SCALE	NULLABLE	DEFAULT
id	number	4				
loc_name	varchar2	20			X	
	date					
address	varchar2	30				
city	varchar2	20				
zip_postal	varchar2	20			X	
phone	varchar2	15			X	
email	varchar2	80			X	
manager_id	number	4			X	
contact	varchar2	40			X	

CREATE TABLE global_fast_food.locations (
id NUMBER (6) CONSTRAINT gl_id_PK PRIMARY KEY.
loc_Name VARCHAR gl_name_in Not NULL,
date DATE CONSTRAINT gl_date_nn NOT_NULL,
address VARCHAR 2 (30) CONSTRAINT gl_address NOT NULL,
zip_postal VARCHAR (20),
Phone VARCHAR2(15),
email VARCHAR 2 (100),
);

PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a
- PRIMARY KEY - Ensure each row in a table is uniquely identifiable
- FOREIGN KEY - Establish a link b/w two tables. It enforces referential integrity.
- CHECK CONSTRAINT

2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal_id). The license_tag_number must be unique. The admit_date and vaccination_date columns cannot contain null values.

animal_id NUMBER(6) — primary key
name VARCHAR2(25) — unique
license_tag_number NUMBER(10)
admit_date DATE — NOT NULL
adoption_id NUMBER(5),
vaccination_date DATE — NOT NULL

3. Create the animals table. Write the syntax you will use to create the table.

(reate table animal (animal_id NUMBER(6), name, VARCHAR2(25), license_tag_no Number(10), admit_date DATE Not NULL.
);

4. Enter one row into the table. Execute a SELECT * statement to verify your input. Refer to the graphic below for input.

ANIMAL_ID	NAME	LICENSE_TAG_NUMBER	ADMIT_DATE	ADOPTION_ID	VACCINATION_DATE
101	Spot	35540	10-Oct-2004	205	12-Oct-2004

Insert into animal (animal_id Number(6), license_tag_number, admit_date) ('10-oct-2004', 'DD-mon-YYYY')
205, TO_DATE('12-oct-2004', 'DD-Mon-YYYY');

5. Write the syntax to create a foreign key (adoption_id) in the animals table that has a corresponding primary-key reference in the adoptions table. Show both the column-level and table-level syntax. Note that because you have not actually created an adoptions table, no adoption_id primary key exists, so the foreign key cannot be added to the animals table.

adoption_id Number(5) CONSTRAINT _adoption REFERENCE (apoption_id)

6. What is the effect of setting the foreign key in the ANIMAL table as:

a. ON DELETE CASCADE → If a referenced row in the parent table is deleted.
b. ON DELETE SET NULL

↳ If a referenced row in the parent table is deleted.

7. What are the restrictions on defining a CHECK constraint?

→ Cannot reference other tables - only columns in the same row

↳ must be logically valid

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

EXERCISE 13
Creating Views

1. What are three uses for a view from a DBA's perspective?

A view is a virtual table based on the result-set of an SQL statement. A view contains rows & columns, just like a real table.

2. Create a simple view called view_d_songs that contains the ID, title and artist from the DJs on Demand table for each "New Age" type code. In the subquery, use the alias "Song Title" for the title column.

```
CREATE VIEW view_d_songs AS
SELECT ID, title AS "Song Title", artist
WHERE type_code = 'New Age';
```

3. SELECT * FROM view_d_songs. What was returned?

The query would return the ID, Song Title, and artist for all songs that have a type code of New Age from the DJs on Demand table.

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns.

Or use alias after the CREATE statement as shown.

```
CREATE OR REPLACE VIEW view_d_songs AS
SELECT ID AS song_id, title AS song_title, artist, type_code AS
                                                    song_type
FROM DJs_On_Demand
WHERE type_cod = 'New Age'
```

5. Jason Tsang, the disk jockey for DJs on Demand, needs a list of the past events and those planned for the coming months so he can make arrangements for each event's equipment setup. As the company manager, you do not want him to have access to the price that clients paid for their events. Create a view for Jason to use that displays the name of the event, the event date, and the theme description. Use aliases for each column name.

```
CREATE VIEW Jason_events_views AS
SELECT
        event_name AS event_title,
        event_date AS event_day,
        theme_description AS theme_details
FROM events;
```

6. It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by department. Use the Oracle database to prepare a view that displays the needed information for department managers.

```
CREATE VIEW department_Salary_Summary AS
SELECT
    department_id,
    MIN(Salary) AS min_Salary,
    MAX(Salary) AS max_Salary,
    AVG(Salary) AS avg_Salary
FROM
    employees
GROUP BY
    department_id;
```

DML Operations and Views

Use the DESCRIBE statement to verify that you have tables named copy_d_songs, copy_d_events, copy_d_cds, and copy_d_clients in your schema. If you don't, write a query to create a copy of each.

1. Query the data dictionary USER_UPDATABLE_COLUMNS to make sure the columns in the base tables will allow UPDATE, INSERT, or DELETE. All table names in the data dictionary are stored in uppercase.

```
dftable
  FROM user_updatable_columns
  Where table_name = 'COPY_D_SONGS';
```

Use the same syntax but change table_name of the other tables.

2. Use the CREATE or REPLACE option to create a view of *all* the columns in the copy_d_songs table called view_copy_d_songs.

```
create (or) Replace view copy_d_songs AS
  Select * from copy_d_Songs
```

3. Use view_copy_d_songs to INSERT the following data into the underlying copy_d_songs table. Execute a SELECT * from copy_d_songs to verify your DML command. See the graphic.

ID	TITLE	DURATION	ARTIST	TYPE_CODE
88	Mello Jello	2	The What	4

```
insert into view_copy_d_songs (id, title,
duration, artist, type_code);
Select * from copy_d_songs;
```

4. Create a view based on the DJs on Demand COPY_D_CDS table. Name the view read_copy_d_cds. Select all columns to be included in the view. Add a WHERE clause to restrict the year to 2000. Add the WITH READ ONLY option.

```
Create View read_copy_dJ
Select *
FROM copy_d_cds
WHERE year=2000;
```

5. Using the read_copy_d_cds view, execute a DELETE FROM read_copy_d_cds WHERE cd_number = 90;

```
Delete from read_copy_cds where number=90;
```

6. Use REPLACE to modify read_copy_d_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds. Execute a SELECT * statement to verify that the view exists.

```
create or replace view read_cop_d_cds Adp
Select * from copy_d_cds.
```

7. Use the read_copy_d_cds view to delete any CD of year 2000 from the underlying copy_d_cds.

```
Delete from read_copy_d_cds WHERE year=2000;
```

8. Use the read_copy_d_cds view to delete cd_number 90 from the underlying copy_d_cds table.

```
Delete from read_copy_d where cd_number=90;
```

9. Use the read_copy_d_cds view to delete year 2001 records.

```
Delete from read copy_d_ as where year=2001;
```

10. Execute a SELECT * statement for the base table copy_d_cds. What rows were deleted?

Select * from copy_d_cds;

11. What are the restrictions on modifying data through a view?

Read only views block all DML operation.

12. What is Moore's Law? Do you consider that it will continue to apply indefinitely? Support your opinion with research from the internet.

more's law state that the no. of transistors on a microchip.

* 3D chip stacking
* photonic

13. What is the "singularity" in terms of computing?

It refers hypothetical future point.

Managing Views

1. Create a view from the copy_d_songs table called view_copy_d_songs that includes only the title and artist. Execute a SELECT * statement to verify that the view exists.

```
Create view view_copy_d_songs
Select title, artist
FROM Copy_d_songs;
```

2. Issue a DROP view_copy_d_songs. Execute a SELECT * statement to verify that the view has been deleted.

```
Drop view view_copy_d_songs;
Select * FROM view_copy_d_songs
```

3. Create a query that selects the last name and salary from the Oracle database. Rank the salaries from highest to lowest for the top three employees.

```
Select last_name, salary from (select last_name,
salary RANK() OVER (order by salary DESC) AS rank
from employees)
```

4. Construct an inline view from the Oracle database that lists the last name, salary, department ID, and maximum salary for each department. Hint: One query will need to calculate maximum salary by department ID.

```
Select e.last_name, e.salary, e.department_id, d.top_
salary. Top (Select department_id, MAX(Salary)).
```

5. Create a query that will return the staff members of Global Fast Foods ranked by salary from lowest to highest.

```
Select last_name, salary
RANK() over (order by salary ASC) AS Rank.
```

Indexes and Synonyms

1. What is an index and what is it used for?

An index is a database object that improves the speed of data.

2. What is a ROWID, and how is it used?

Row is a unique identifier for each row in oracle. database.

3. When will an index be created automatically?

Create INDEX idx_cd_number ON track_listing.

4. Create a nonunique index (foreign key) for the DJs on Demand column (cd_number) in the D_TRACK_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

Select index_name, unique from dts.

5. Use the join statement to display the indexes and uniqueness that exist in the data dictionary for the DJs on Demand D_SONGS table.

Select index_name, table_name, unique.
from user_indexes
WHERE table_name = 'D_Events';

6. Use a SELECT statement to display the index_name, table_name, and uniqueness from the data dictionary USER_INDEXES for the DJs on Demand D_EVENTS table.

Select index_name, table_name, unique.
from user_indexes.

7. Write a query to create a synonym called dj_tracks for the DJs on Demand d_track_listings table.

Create Synonym dl_tracks
For dl_track_listings

8. Create a function-based index for the last_name column in DJs on Demand D_PARTNERS table that makes it possible not to have to capitalize the table name for searches. Write a SELECT statement that would use this index.

Create INDES_idx_upper_last_name
ON d_partists (UPPER (Cust_name));

9. Create a synonym for the D_TRACK_LISTINGS table. Confirm that it has been created by querying the data dictionary.

Create Synonym d_track_listing_syn
For d_track_listings

10.Drop the synonym that you created in question

Drop Synonym d_track_listing_syn;

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

DROP INDEX *index*;

Find the Solution for the following:

1. Create a sequence to be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1000. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

2. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number

3. Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.
4. Create a nonunique index on the foreign key column (DEPT_ID) in the EMP table.

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

Solutions:

1) CREATE SEQUENCE DEPT_ID_SEQ START WITH 200 MAXVALUE 1000 INCREMENT BY 10';

2) SELECT SEQUENCE_NAME, MAX_VALUE, INCREMENT_BY, LAST_NUMBER, FROM USER_SEQUENCES WHERE SEQUENCE_NAME = 'DEPT_ID_SEQ';

3) INSERT INTO DEPT (DEPT_ID, DEPT_NAME) VALUES (DEPT_ID_SEQ.NXTVAL, 'Education');

INSERT INTO DEPT (DEPT_ID, DEPT_NAME) VALUES (DEPT_ID_SEQ_NEXTVAL, 'Administration');

4) CREATE INDEX EMP_DEPT_ID_IX ON EMP (DEPT_ID);

5) SELECT INDEX_NAME, TABLE_NAME, UNIQNESS FROM USER_INDEXES WHERE TABLE_NAME = 'EMP';

DROP INDEX *index*;

Find the Solution for the following:

1. Create a sequence to be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1000. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

2. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number

3. Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.
4. Create a nonunique index on the foreign key column (DEPT_ID) in the EMP table.

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

Solutions:

1) CREATE SEQUENCE DEPT_ID_SEQ START WITH 200 MAXVALUE 1000 INCREMENT BY 10';

2) SELECT SEQUENCE_NAME, MAX_VALUE, INCREMENT_BY, LAST_NUMBER, FROM USER_SEQUENCES WHERE SEQUENCE_NAME = 'DEPT_ID_SEQ';

3) INSERT INTO DEPT (DEPT_ID, DEPT_NAME) VALUES (DEPT_ID_SEQ.NXTVAL, 'Education');

INSERT INTO DEPT (DEPT_ID, DEPT_NAME) VALUES (DEPT_ID_SEQ_NEXTVAL, 'Administration');

4) CREATE INDEX EMP_DEPT_ID_IX ON EMP (DEPT_ID);

5) SELECT INDEX_NAME, TABLE_NAME, UNIQNESS FROM USER_INDEXES WHERE TABLE_NAME = 'EMP';

Find the Solution for the following:

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 onto oracle server is CREATE SESSION

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

CREATE TABLE also a System privilege.

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

WITH GRANT OPTION Clause in the GRANT

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?

To make Job easier when Creating many users who require Same System privileges used role.

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

ALTER USER. Syntax 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.

7. Query all the rows in your DEPARTMENTS table.

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.

9. Query the USER_TABLES data dictionary to see information about the tables that you own.

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

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

6) GRANT SELECT ON DEPARTMENTS TO user_b;
GRANT SELECT ON DEPARTMENTS TO your_username;

7) SELECT * FROM DEPARTMENTS;

8) INSERT INTO DEPARTMENTS (department_number, department_name) VALUES (500, 'Education');

INSERT INTO DEPARTMENTS (department_number, department_name) VALUES (510, 'Human Resources');

9) SELECT * FROM USER_TABLES;

10) REVOKE SELECT ON your_table_name FROM other_team_user;

11) DELETE FROM DEPARTMENTS WHERE department_number = 500;

COMMIT;

Evaluation Procedure	Marks awarded
Practice Evaluation (5)	5
Viva(5)	5
Total (10)	10
Faculty Signature	

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION fact (n NUMBER) RETURN NUMBER IS
 f NUMBER := 1;
BEGIN
   FOR i IN 1..n LOOP f := f * i; END LOOP;
   RETURN f;
  END;
  BEGIN
    DBMS_OUTPUT.PUT_LINE ('Factorial = '|| fact(5));
END;
```

Program 2

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

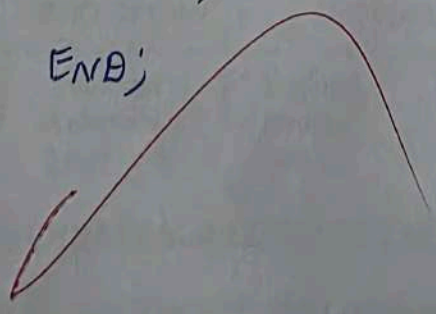
```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE
    get_book_ info(
     p_bookid IN NUMBER
     p_ title OUT VARCHAR
     p_ author OUT VARCHAR
     P_ price INOUT NUMBER
     )IS
     BEGIN
       SELECT title , author ,price INTO p_file , p. author , p-price FROM library
WHERE book_id= p_book id;
     EXCEPTION
          WHEN NO_DATA_FOUND THEN DBMS_OUTPUT .PUT_LINE ('Book not found.');
     END;

     DECLARE
          v_title VARCHAR2(50);
          v_author VARCHAR 2(50);
v_price NUMBER :=0;
BEGIN
get_book_info(1or.v_title
v_author , v_price);
DBMS_OUTPUT,PUT_LINE ('
Title: '||v_title||'. Author:'||
v_ author||', price'
'||v_price);
END';
```

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

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.

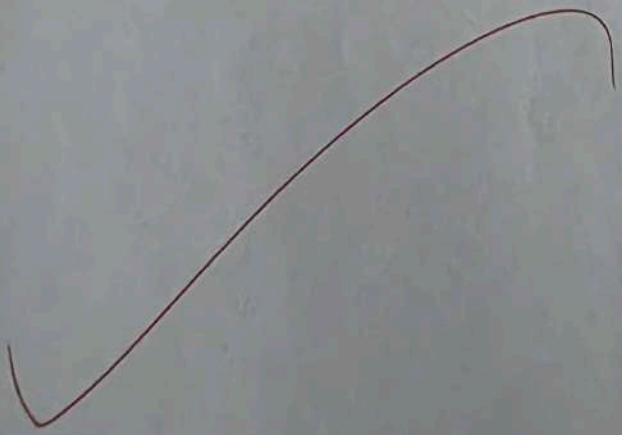
```
CREATE OR REPLACE TRIGGER
prevent_parent_delete
BEFORE DELETE ON parent
FOR EACH ROW
DECLARE
 v_count NUMBER;
BEGIN
   SELECT COUNT (*) INTO v_count FROM
child WHERE parent_id
= :OLD.parent_id;
IF v_count > 0 THEN
  RAISE_APPLICATION_ERROR (-20001, 'cannot delete parent
record : child records exists.');
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_duplicate_title
BEFORE INSERT OR UPDATE ON books
DECLARE
  SELECT COUNT (*) INTO V_Count FROM books
    WHERE title =:NEW.title;

  IF V_Count > 0 THEN

  RAISE_APPLICATION_ERROR (-20002, 'Duplicate title not
                          allowed.');

    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
limit_total_salary
BEFORE INSERT ON employees
FOR EACH ROW
DECLARE
    v_total NUMBER;
BEGIN
    SELECT SUM(salary) INTO v_total
FROM employees;
  IF v_total + :NEW.salary > 1000000
  THEN
RAISE_APPLICATION_ERROR (-20003, 'Total salary limit
    exceeded!');
END IF;
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 TABLE audit (emp NUMBER, oldsal NUMBER, newsal
    NUMBER, dt DATE);

CREATE TRIGGER try_audit
AFTER UPDATE OF Salary ON employees FOR EACH ROW
BEGIN
    INSERT INTO audit
VALUES (:OLD.employee_id, :OLD.Salary, :NEW.Salary, );

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 TABLE audit_log (user_name VARCHAR(30), action
VARCHAR(10, log_date DATE);

CREATE TRIGGER trg_user_audit
AFTER INSERT OR UPDATE ON DELETE ON employees.
FOR EACH ROW
  INSERT INTO audit_log VALUE (USER, CASE WHEN INSERTION
  THEN 'INSERT' WHEN UPDATING THEN 'UPDATE' ELSE
  'DELETE' END, SYSDATE);
END;
```

Program 7

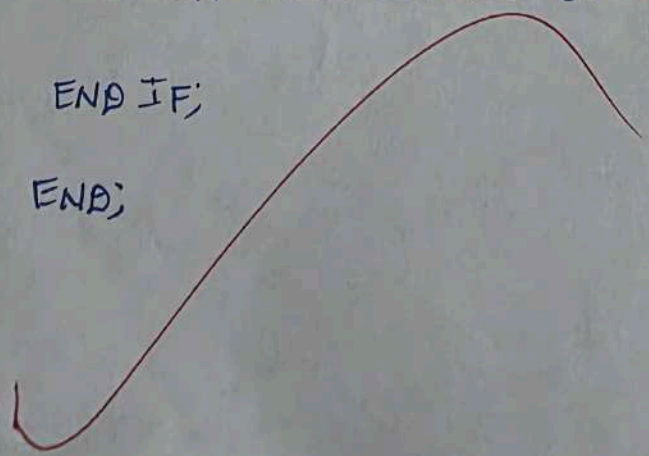
Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE TABLE Sales (sid NUMBER, amount NUMBER, total NUMBER);

CREATE TRIGGER trg_total_update AFTER INSERT ON Sales
FOR EACH ROW
BEGIN
    UPDATE Sales SET total = NVL(total,0) + :NEW.amount;
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 TRIGGER trg_Check_Stock BEFORE INSERT ON orders

FOR EACH ROW

DECLARE
   v_Stock NUMBER;

  BEGIN
      SELECT quantity INTO v_Stock FROM items WHERE
  item_id = : NEW item_id;
  IF v_stock < :NEW .qty THEN

  RAISE_APPLICATION_ERROR (-20010, 'Insufficient Stock');

   END IF;

  END;
```



EXERCISE 18

Structure of 'restaurants' collection:


```
{
 "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 'Chinees' or restaurant's name begins with letter 'Wil'. db.restaurants.find({$or: {"cuisine": {"$in: ["American", "Chinese"]}}, {"name" /^Wil/i}});

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, "date": ISODate("2014-08-11T00:00:00Z")}}});

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.grade": "A", "grades.1.score": 9, "grades": ISODate("2014-08-11T00:00:00Z")});

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

db.restaurants.find({"address.coord.1": 40.848447});

which is more than 42 and upto 52..

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 along with all the columns.

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

7. Write a MongoDB query to arranged 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});
```

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.score":{$mod:[7,0]}});
```

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":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/i}},{name:
1, borough:1, "address.coord"});
```

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}});

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"]}});

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"}});

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}, boroughof "manhattan", "Brooklyn" cuisine:["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": {$all:[2,6]}});

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({"grades.score": {$all:[2,6]}: "Mahattan"});

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({"grades.score": {$all:[2,6]}

["Mahattan","Brooklyn"]});

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({"grades.score": {$all: [2,6]}, borough: ["Manhattan", "Brooklyn"]});

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({"grades.score": {$all: [2,6]}, cuisine: {$all: ["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({"grades.score": {$in: [2,6]}});

Sample document of 'movies' collection

{

_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_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: 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 '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})

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

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

5. 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"})
```

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

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

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

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

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

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

9. 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}})
```

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

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

11. 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": {$gte: 1}})
```

12. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast

```
db.movies.find({cast: "Charles Kayser"})
```

including "Charles Kayser".

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

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.

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

14) db.movies.find(
{title: /Scene/i},
{title:1, language:1, released:1,
directors:1, writers:1, countries:1}
).

PROGRAM 1

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

```
SET SERVEROUTPUT ON;

DECLARE

    v_Salary NUMBER;

    v_incentive NUMBER;

BEGIN

    SELECT Salary INTO v_Salary FROM employees WHERE employee_id=110;

    IF v_Salary >= 50000 THEN

        v_incentive := v_Salary * 0.10;

    ELSIF v_Salary >= 30000 THEN

        v_incentive := v_Salary * 0.07;

    ELSE

        v_incentive := v_Salary * 0.05;

    END IF;

    DBMS_OUTPUT.PUT_LINE ('incentive: ' || v_incentive);

END;

/
```

PROGRAM 2

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

```
SET SERVEROUTPUT ON;

DECLARE

  "Name" VARCHAR2(20) := 'vikat';

BEGIN

   DBMS_OUTPUT.PUT_LINE(Name);

END;

/
```

PROGRAM 3

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

Sample table: employees

```
SET SERVEROUTPUT ON;

DECLARE
    v_old_Salary employees.Salary%TYPE;
BEGIN
    SELECT Salary INTO v_old_Salary
    FROM employees
    WHERE employee_id=122;

    UPDATE employees
    SET Salary =v_old_Salary * 1.10
    WHERE employee_id=122;

    DBMS_OUTPUT.PUT_LINE('Salary adjusted Successfully for Employee ID 122.');
DBMS_OUTPUT.PUT_LINE('old Salary : ' || v_old_Salary);
DBMS_OUTPUT.PUT_LINE('New Salary : ' || (v_old_Salary * 1.10)

EXPEOTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee ID 122 not found.');
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.

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE Check_employee_status IS
  v_name VARCHAR2(30) := 'Rishi';
  v_bonus NUMBER := 1000;
BEGIN
  IF (v_name IS NOT NULL) AND (v_bonus > 5000) THEN
    DBMS_OUTPUT.PUT_LINE('Both Condition are TRUE -AND return
                          TRUE.');
  ELSE
    DBMS_OUTPUT.LINE('Atleast one Condition is FALSE -AND
                      return FALSE.');
  END IF;
END;
/
```

PROGRAM 5

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

```
SET SERVEROUTPUT ON;
BEGIN
    IF 'S_Kumar' LIKE 'S%' THEN DBMS_OUTPUT.PUT_LINE('starts with S');
    END IF;
IF ('S_Kumar' LIKE 'S_%' THEN DBMS_OUTPUT.PUT_LINE('Has one char after S');
    END IF;
IF 'S_Kumar' LIKE 'S\_%' ESCAPE '\' THEN DBMS_OUTPUT.PUT_LINE
    ('matched underscore');
    END IF;
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.

```
SET SERVEROUTPUT ON;
DECLARE a NUMBER :=30; b NUMBER:= 60; s NUMBER ; l NUMBER;
BEGIN
  IF a<b THEN s:=a; l:=b; ELSE s:=b; l:=a; END IF;
  DBMS_OUTPUT.PUT_LINE ('Small=' || s || ' Large=' || l);
END;
```

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.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE inc ((p_id NUMBER, p_t NUMBER) IS
BEGIN
  UPDATE employees SET salary = salary +(p_t *0.05) WHERE
                                  employee_id=p_id;
  IF SQL%ROWCOUNT >0 THEN DBMS_OUTPUT.PUT_LINE
                          ('Record updated');
ELSE DBMS_OUTPUT.PUT_LINE ('No record updated'); END IF
END;
/
BEGIN inc(110, 20000); END;
/
```

PROGRAM 8

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

```
CREATE OR REPLACE PROCEDURE
incentive (p_sale IN NUMBER) IS
  inc NUMBER;
BEGIN
  IF p_sale >=100000 THEN inc :=
p_sale * 0.1;
ELSIF p_sale >= 500000 THEN inc:=
p_sale * 0.05;
  ELSE inc := p_sale *0.02;
END IF;
DBMS_OUTPUT.PUT_LINE ('incentive =' || inc);
END;
/
BEGIN
  incentive (750000);
END;
/
```

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
    C NUMBER;
BEGIN
    SELECT COUNT(*) INTO C FROM employees
WHERE department_id = 50;
    IF C < 45 THEN
        DBMS_OUTPUT.PUT_LINE('Vacancies: ' || (45-C));
    ELSE
        DBMS_OUTPUT.PUT_LINE('No Vacancies');
    END IF;
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
    dno NUMBER := 60;
    c NUMBER;
    total NUMBER := 50;
BEGIN
    SELECT COUNT(*) INTO c FROM employees
WHERE department_id = dno;
    IF c < total THEN
    DBMS_OUTPUT.PUT_LINE ('vacannies: ' || (total - c));
ELSE
    DBMS_OUTPUT.PUT_LINE ('No vacancies');
    END IF;
    END;
/
```

PROGRAM 11

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

```
BEGIN
    FOR rec IN (SELECT employee_id, first_name||' '||last_name AS
                name, job_title, hire_date, salary FROM employees)

    LOOP
    DBMS_OUTPUT.PUT_LINE ('ID: '||rec.employee_id||', Name: '
    ||rec.name||', Job: '||rec.job_title||', HireDate: '||
    rec.hire_date||', Salary: '||rec.salary);

    END LOOP;

END;
/
```

PROGRAM 12

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

```
BEGIN
  FOR rec IN ( SELECT e.employee_id, e.first_name || ' ' || e.last_name AS
               name ; d.department name FROM employees e
               JOIN departments d ON e.department_id =
                 d.department_id)
  Loop
  DBMS_OUTPUT.PUT_LINE ('ID:' || rec.employee_id || ', Name:' ||
                 rec.name|| ', Department: ' || rec.department_
                 name);
  END Loop;
END;
/
```

PROGRAM 13

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

```
BEGIN
  FOR rec IN (SELECT job_id, job_title, min_salary FROM jobs)
Loop
  DBMS_OUTPUT.PUT_LINE ('Job ID: ' || rec.job_id ||
                        ', Title: ' || rec.job_title || ', Min Salary: ' ||
    rec.min_salary);
  END Loop;
END;
/
```

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 e.employee_id, e.first_name||' '||
                e.last_name AS name, jh.start_date
    FROM employees e, job_history jh
    WHERE e.employee_id=jh.employee_id)
  Loop
    DBMS_OUTPUT.PUT_LINE('ID:'||rec.employee_id||
                ',Name:'||rec.name||', Start Date:'||
    rec.start_date);
  END Loop;
  END;
  /
```

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 e.employee_id, e.first_name || ' ' || e.last_name
        AS name, jh.end_date
        FROM employees e, job_history jh
        WHERE e.employee_id = jh.employee_id)
            ', Name: ' || rec.name || ', End Date: ' ||
rec.end_date);

    END LOOP;

END;
/
```

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