Ex.No.: 5		CREATING VIEWS
Date:		

After the completion of this exercise, students will be able to do the following:

- Describe a view
- Create, alter the definition of, and drop a view
- Retrieve data through a view
- Insert, update, and delete data through a view
- Create and use an inline view

View

A view is a logical table based on a table or another view. A view contains no data but is like a window through which data from tables can be viewed or changed.The tables on which a view is based are called base tables.

Advantages of Views

- To restrict data access
- To make complex queries easy
- To provide data independence
- To present different views of the same data

Classification of views

1. Simple view
2. Complex view

Feature	Simple	Complex
No. of tables	One	One or more
Contains functions	No	Yes
Contains groups of data	No	Yes
DML operations thr' view	Yes	Not always

Creating a view

Syntax

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-id , lastname
As employee , department-id 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
where view-name = "Employee-vu';

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

Select Employee, department_id
From Employee-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 As Empno,
last_name As employee .., department_id as DEPT NO
from employees 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 DEPT50 . SET .DEPTNO = 80 WHERE Employee
= '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 Emplo,
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

in Job-grade j
n e.salary
tween j. lowest_salary and j. highest_salary;

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

Ex.No.: 6		RESTRICTING AND SORTING DATA
Date:		

After the completion of this exercise, the students will be able to do the following:

- Limit the rows retrieved by the queries
- Sort the rows retrieved by the queries
-

Limiting the Rows selected

- Using WHERE clause
- Alias cannot used in WHERE clause

Syntax

```
SELECT----------
FROM----------
WHERE condition;
```

Example:

```
SELECT employee_id,last_name, job_id, deparment_id FROM employees WHERE
department_id=90;
```

Character strings and Dates

Character strings and date values are enclosed in single quotation marks.

Character values are case sensitive and date values are format sensitive.

Example:

```
SELECT employee_id,last_name, job_id, deparment_id FROM employees
WHERE last_name=’WHALEN”;
```

Comparison Conditions

All relational operators can be used. (=, >, >=, <, <= ,<>,!=)

Example:

```
SELECT last_name, salary
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 '20-Feb-1998' AND
'01 - May- 1998' order by hire_date;
```

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 department_id  IN (20,50)
 order by  last-name;
```

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 "Employee", salary "monthly salary"
 from  employees
where  salary Between  5000  and  12000
 AND   department_id IN (20,50);
```

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 '%94';
```

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  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 nul,orderby)

```
select last_name, salary, commission_pet
from employees
where commission_pet is not null
orderby salary Desc, commission_pet 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 ali 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 "Employee". salary "Montly salary",
                                       commission_pet
from employees
where commission_pet = .20;
```

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

Ex.No.: 7		USING SET OPERATORS
Date:		

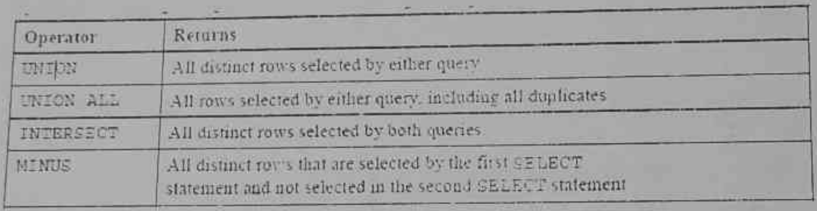
Objectives

After the completion this exercise, the students should be able to do the following:

- Describe set operators
- Use a set operator to combine multiple queries into a single query
- Control the order of rows returned

The set operators combine the results of two or more component queries into one result.

Queries containing set operators are called *compound queries.*



Operator	Returns
UNION	All distinct rows selected by either query
UNION ALL	All rows selected by either query, including all duplicates
INTERSECT	All distinct rows selected by both queries
MINUS	All distinct rows that are selected by the first SELECT statement and not selected in the second SELECT statement

The tables used in this lesson are:

- EMPLOYEES: Provides details regarding all current employees

- JOB_HISTORY: Records the details of the start date and end date of the former job, and the job
identification number and department when an employee switches jobs

UNION Operator

Guidelines

- The number of columns and the data types of the columns being selected must be identical in all the SELECT statements used in the query. The names of the columns need not be identical.

- UNION operates over all of the columns being selected.

- NULL values are not ignored during duplicate checking.

- The IN operator has a higher precedence than the UNION operator.

Display 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 (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 employees
MINUS
select department-id from employees where UPPER(job-ia)
      = UPPER('ST_CLERK') orber by 1;
```

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 c
Join localitions l using (country-id) Join departments d
using ( location_id) where department_id
                                    IS NOT NULL;'
```

Qn. 3 QUERY

```
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 10

UNION ALL

SELECT DISTINCT job-id, department-id
FRON employeen
WHERE department_id = 50 -

UNION ALL

SELECT DISTINCT job_id, department-id
FROM employees
WHERE department-id = 20 ;
```

Qn. 4 QUERY

```
SELECT employee-id , job-id
FROM employees

INTERSECT
SELECT employee_id , job_id
FROM ~~employees~~ job_history
ORDER BY 1 ;
```

Qn. 5 QUERY

```
SELECT last_name , department_id , TO_CHAR ('null')
FROM employees

UNION
SELECT TO_CHAR ('null'); department_id , department_no
FROM departments
ORDER BY 1 ;
```

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

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).

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

Ex.No.: 8		WORKING WITH MULTIPLE TABLES
Date:		

Objective

After the completion of this exercise, the students will be able to do the following:

- Write SELECT statements to access data from more than one table using equality and nonequality joins
- View data that generally does not meet a join condition by using outer joins
- Join a table to itself by using a self join

Sometimes you need to use data from more than one table.

Cartesian Products

- A Cartesian product is formed when:

– A join condition is omitted

– A join condition is invalid

– All rows in the first table are joined to all rows in the second table

- To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

A Cartesian product tends to generate a large number of rows, and the result is rarely useful. You should always include a valid join condition in a WHERE clause, unless you have a specific need to combine all rows from all tables.

Cartesian products are useful for some tests when you need to generate a large number of rows to simulate a reasonable amount of data.

Example:

To displays employee last name and department name from the EMPLOYEES and DEPARTMENTS tables.

```
SELECT last_name, department_name dept_name
FROM   employees, departments;
```

Types of Joins

- Equijoin
- Non-equijoin
- Outer join
- Self join
- Cross joins
- Natural joins
- Using clause
- Full or two sided outer joins
- Arbitrary join conditions for outer joins

Joining Tables Using Oracle Syntax

```
SELECT table1.column, table2.column
```

```
2.  SELECT DISTINCT  e.job_id,
    d. 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.  SELECT  e.last_name ,
      d.department_name ,
      d.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;
```

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 alslso 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.departments_id ~~ON~~ =
                                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.

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

```
4.  SELECT  e.last_name;
     d. depart ment_name
    FROM employees  e
    JOIN  departments  d  ON
    e.department_id = d.department_id
    WHERE  e.last_name  LIKE  '%a%';

5.  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  locations  l ON  d.location_id = l.location_id
    WHERE  l.city  =  'Toronto';

6.  SELECT  e.last_name AS  Employee,  e.employee_id  AS
    Emp #  , m.last_name  AS  Manager,  m.employee_id
    AS  Manager  , m.employee_id  AS  Mgr#
    From  employee  e
    LEFT  JOIN   employees  m  ON
    e.manager_id  =  m.employee_id;

7.  SELECT  employee_id , employee_name,  manager_id
    FROM  employees
    ORDER BY  employee_id ,'

8.  SELECT  e1.last_name   AS  "Employee",  e1.department id
    AS "dept No",  e2.last_name   AS  "Colleague"
    FROM   employees  e 1
    Join  employees  e2  ON
    e1. department _id  = e2.department_id
    WHERE    e1.employee_id != e2.employee_id
    ORDER BY  e1.last_name;
```

```
9.  DESCRIBE  JOB-GRADES;'
    SELECT  e.name  AS " Employee Name", e.job  As "Job Title",
d.department_name  AS  "Department", e.salary AS
"Salary",  j.grade_level  AS  "Grade"  FROM
employees  e   JOIN  departments  d  ON
e.department_id = d.department_id   JOIN  job_grades
j  ON  e.salary   BETWEEN  j. lowest_sal   AND
 j.highest_sal;


10.  SELECT  e.name   AS " Employee" , e.hire_date AS
                                       "Hire date".
 FROM  employees  e
 WHERE   e.hire_date > (SELECT  hire_date   FROM employees
WHERE   name  =  " Davies");


11.  SELECT   e1.name   As "Employee",
 e.hire_date AS "Emp Hired", e2.name
 AS "Manager",  e2.hire_date AS "Mgr Hired"
 FROM  employees  e1
 JOIN   employees  e2  ON   e1.manager_id = e2.employee_id
WHERE e1. hire_date  <  e2.hire_date
 ORDER  By   e1.name ;
```

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

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

Ex.No.: 9		SUB QUERIES
Date:		

Objectives

After completing this lesson, you should be able to do the following:
- Define subqueries
- Describe the types of problems that subqueries can solve
- List the types of subqueries
- Write single-row and multiple-row subqueries

Using a Subquery to Solve a Problem
Who has a salary greater than Abel's?

Main query:
Which employees have salaries greater than Abel's salary?

Subquery:
What is Abel's salary?

Subquery Syntax

SELECT *select_list* FROM *table* WHERE *expr operator* (SELECT *select_list* FROM *table*);

- The subquery (inner query) executes once before the main query (outer query).

- The result of the subquery is used by the main query.

A subquery is a SELECT statement that is embedded in a clause of another SELECT statement. You can build powerful statements out of simple ones by using subqueries. They can be very useful when you need to select rows from a table with a condition that depends on the data in the table itself.

You can place the subquery in a number of SQL clauses, including the following:

- WHERE clause
- HAVING clause
- FROM clause

In the syntax:
operator includes a comparison condition such as >, =, or IN

Note: Comparison conditions fall into two classes: single-row operators

```
1. SELECT e.last_name, e.hire_date FROM employees e
   WHERE e.department_id = (SELECT department_id
   FROM employees
   WHERE last_name = '& input_last_name')
   AND e.last_name != '& input_last_name';

2. SELECT employee_id, last_name, salary
   FROM employees
   WHERE salary > (SELECT AVG(salary) FROM employees)
   ORDER BY salary ASC;

3. SELECT e.employee_id, e.last_name
   FROM employees e
   WHERE e.department_id IN (
   SELECT department_id
   FROM employees
   WHERE last_name LIKE '%u%');

4. SELECT e.last_name,
   e.department_id, e.job_id
   FROM employees e
   JOIN departments d ON
   e.departments d ON
   e.department_id = d.department_id
   WHERE d.location_id = 1700;
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).

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.

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*.

```
5· SELECT e·last_name , e·salary
FROM employees e
WHERE e·manager_id = (SELECT employee_id FROM
employees WHERE last_name = "king");
```

```
6· 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 = "Exceutive";
```

```
7. SELECT e· employee_id , e·last_name,
e· salary
FROM employees e
WHERE e· salary > (SELECT AVG(salary) FROM
                                          employees)
AND e· department_id IN (
        SELECT 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	