EXERCISE-10

USING THE SET OPERATORS

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.
- By default, the output is sorted in ascending order of the first column of the SELECT clause.

Example:
Display the current and previous job details of all employees. Display each employee only once.
```
SELECT employee_id, job_id FROM employees UNION SELECT employee_id, job_id
FROM job_history;
```

Example:
```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history;
```

UNION ALL Operator

Guidelines
The guidelines for UNION and UNION ALL are the same, with the following two exceptions that pertain to UNION ALL:
- Unlike UNION, duplicate rows are not eliminated and the output is not sorted by default.
- The DISTINCT keyword cannot be used.

Example:
Display the current and previous departments of all employees.

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

INTERSECT 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.
- Reversing the order of the intersected tables does not alter the result.
- INTERSECT does not ignore NULL values.

Example:

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 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

SELECT country_id, country_name FROM countries MINUS SELECT c.country_id, c.country_name FROM countries c JOIN locations l ON c.country_id = l.country_id JOIN department d ON l.location_id = d.location_id;

report.

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 UNION SELECT job_id, department_id FROM employees WHERE department_id = 50 UNION SELECT job_id, department_id FROM employees WHERE department_id = 20;

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

SELECT e.employee_id, e.job_id FROM employees e JOIN job_history j on e.employee_id = j.employee_id WHERE j.start_date = (SELECT MIN(start_date) FROM job_history WHERE employee_id = e.employee_id);

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 SELECT NULL, department_id, department_name FROM departments;

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

Practice Exercise

NULL Functions

1. Create a report that shows the Global Fast Foods promotional name, start date, and end date from the f_promotional_menus table. If there is an end date, temporarily replace it with "end in two weeks". If there is no end date, replace it with today's date.

SELECT promo_name, promo_begin_date AS start_Date,
NVL(TO_CHAR(promo_end_date, 'end in two weeks'),
TO_CHAR(SYSDATE, 'DD-MON-YYYY')) AS END_Date FROM
f_promotional_menus;

2. Not all Global Fast Foods staff members receive overtime pay. Instead of displaying a null value for these employees, replace null with zero. Include the employee's last name and overtime rate in the output. Label the overtime rate as "Overtime Status".

SELECT last_name, NVL(overtime_rate, 0) AS "Overtime
Status" FROM f_staffs;

3. The manager of Global Fast Foods has decided to give all staff who currently do not earn overtime an overtime rate of $5.00. Construct a query that displays the last names and the overtime rate for each staff member, substituting $5.00 for each null overtime value.

SELECT last_name, NVL(overtime_rate, 5) AS
Overtime_Rate FROM f_staffs;

4. Not all Global Fast Foods staff members have a manager. Create a query that displays the employee last name and 9999 in the manager ID column for these employees.

SELECT last_name, NVL(manager_id, 9999) AS
Manager_ID FROM f_staffs;

5. Which statement(s) below will return null if the value of v_sal is 50?

a. SELECT nvl(v_sal, 50) FROM emp;
b. SELECT nvl2(v_sal, 50) FROM emp;
c. SELECT nullif(v_sal, 50) FROM emp;
d. SELECT coalesce (v_sal, Null, 50) FROM emp;

6. What does this query on the Global Fast Foods table return?

SELECT COALESCE(last_name, to_char(manager_id)) as NAME FROM
f_staffs;

SELECT COALESCE(last_name, TO_CHAR(manager_id))
AS Name FROM f_staffs;

7a. Create a report listing the first and last names and month of hire for all employees in the EMPLOYEES table (use TO_CHAR to convert hire_date to display the month).

SELECT first_name, last_name, TO_CHAR(hire_date,
'Month') AS FROM employees;

b.Modify the report to display null if the month of hire is September. Use the NULLIF function.

SELECT first_name, last_name,
NULLIF(TO_CHAR(hire_date, 'Month'), 'September')
AS Month_Hired FROM employees;

8. For all null values in the specialty column in the DJs on Demand d_partners table, substitute "No Specialty." Show the first name and specialty columns only.

```
SELECT first_name, NVL(specialty, 'NO specialty') AS
specialty FROM d_partners
```

Conditional Expressions

1. From the DJs on Demand d_songs table, create a query that replaces the 2-minute songs with "shortest" and the 10-minute songs with "longest". Label the output column "Play Times".

```
SELECT title CASE duration WHEN 2 THEN 'shortest'
WHEN 10 THEN 'longest' ELSE TO_CHAR(duration)
END AS "play.Times" FROM d_songs;
```

2. Use the Oracle database employees table and CASE expression to decode the department id. Display the department id, last name, salary and a column called "New Salary" whose value is based on the following conditions:

If the department id is 10 then 1.25 * salary If the department id is 90 then 1.5 * salary
If the department id is 130 then 1.75 * salary Otherwise, display the old salary.

```
SELECT department_id, last_name, salary,
  CASE department_id,
    WHEN 10 THEN salary * 1.25
    WHEN 90 THEN salary * 1.50
    WHEN 130 THEN salary * 1.75
  ELSE salary.
  END AS "New Salary"
  FROM employees;
```

3. Display the first name, last name, manager ID, and commission percentage of all employees in departments 80 and 90. In a 5th column called "Review", again display the manager ID. If they don't have a manager, display the commission percentage. If they don't have a commission, display 99999.

```
SELECT first_name, last_name, manager_id, commission_pct
NVL(manager_id,
NVL(commission_pct, 99999)) As Review FROM employees WHERE
department_id IN(80,90);
```

Cross Joins and Natural Joins

Use the Oracle database for problems 1-4.

1. Create a cross-join that displays the last name and department name from the employees and departments tables.

```
SELECT e.last_name, d.department_name FROM
employees e CROSS JOIN department d;
```

2. Create a query that uses a natural join to join the departments table and the locations table. Display the department id, department name, location id, and city.

```
SELECT department_id, department_name,
location_id, city FROM departments.
NATURAL JOIN locations;
```

3. Create a query that uses a natural join to join the departments table and the locations table. Restrict the output to only department IDs of 20 and 50. Display the department id, department name, location id, and city.

```
SELECT department_id, department_name,
location_id, city FROM departments
NATURAL JOIN locations
WHERE department_id IN(20,50);
```

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