Ex.No.: 10		AGGREGATING DATA USING GROUP FUNCTIONS
Date:		

Objectives

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

- Identify the available group functions
- Describe the use of group functions
- Group data by using the GROUP BY clause
- Include or exclude grouped rows by using the HAVING clause

What Are Group Functions?

Group functions operate on sets of rows to give one result per group

Types of Group Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Each of the functions accepts an argument. The following table identifies the options that you can use in the syntax:

Function	Description
AVG([DISTINCT|ALL] *n*)	Average value of *n*, ignoring null values
COUNT({*| [DISTINCT|ALL] *expr*})	Number of rows, where *expr* evaluates to something other than null (count all selected rows using *, including duplicates and rows with nulls)
MAX([DISTINCT|ALL] *expr*)	Maximum value of *expr*, ignoring null values
MIN([DISTINCT|ALL] *expr*)	Minimum value of *expr*, ignoring null values
STDDEV([DISTINCT|ALL] *x*)	Standard deviation of *n*, ignoring null values
SUM([DISTINCT|ALL] *n*)	Sum values of *n*, ignoring null values
VARIANCE([DISTINCT|ALL] *x*)	Variance of *n*, ignoring null values

Group Functions: Syntax

SELECT [*column*,] *group_function(column)*, ...
FROM *table*
[WHERE *condition*]

4.
```
SELECT
 ROUND (MAX (salary)) AS
"Maximum",
 ROUND (MIN (salary)) AS
"Minimum",
 ROUND (SUM (salary)) AS "Sum",
 ROUND (AVG (salary)) AS "Average"
FROM employees;
```

5.
```
SELECT
 job_id,
 ROUND (MIN (salary)) AS
"Minimum",
 ROUND (MAX (salary)) AS
"Maximum",
 ROUND (SUM (salary)) AS "Sum",
 ROUND (AVG (salary)) AS "Average"
FROM employees
GROUP BY job_id;
```

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

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

6. SELECT job_title , COUNT (*) AS number_of_people
FROM employees
WHERE job_title = : job_title
GROUP BY job_title ;

7. SELECT COUNT (DISTINCT manager_id) AS Number_of_Managers
FROM employees
WHERE manager_id IS NOT NULL ;

8. SELECT MAX (salary) - MIN (salary) AS DIFFERENCE
FROM employees ;

9. SELECT manager_id , MIN(salary) AS lowest_salary
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY MANAGER_id
HAVING MIN (salary) > 6000
ORDER BY lowest_salary DESC ;

10.
SELECT
COUNT (*) AS Total_Employees,
SUM (CASE WHEN EXTRACT (YEAR FROM hire_date) IN
(1995, 1996, 1997, 1998) THEN 1 ELSE 0 END)
AS Employees_Hired_1995_1998
FROM employees ;

11. SELECT
job_title AS Job,
department_id AS Department_Number,
AVG (salary) AS Average_salary,
SUM (salary) AS Total_salary,

FROM employees
WHERE department_id IN (20, 50, 80, 90)
GROUP BY job_title, department_id;

12. SELECT
d.department_name AS Department_Name,
d.location AS Location,
COUNT (e.employee_id) AS Number_of_people,
ROUND (AVG (e.salary), 2) AS Average_Salary.

FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name, d.location;

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.

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.

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

Ex.No.: 11		PL SQL PROGRAMS
Date:		

PROGRAMS

TO DISPLAY HELLO MESSAGE

```
SQL> set serveroutput on;
SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:='Hello';
  5  dbms_output.put_line(a);
  6  end;
  7  /
Hello

PL/SQL procedure successfully completed.
```

TO INPUT A VALUE FROM THE USER AND DISPLAY IT

```
SQL> set serveroutput on;
SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:=&a;
  5  dbms_output.put_line(a);
  6  end;
  7  /
Enter value for a: 5
old   4: a:=&a;
new   4: a:=5;
5

PL/SQL procedure successfully completed.
```

GREATEST OF TWO NUMBERS

```
SQL> set serveroutput on;

SQL> declare
  2  a number(7);
```

PROGRAM 1

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

```
DECLARE
    v_employee_id NUMBER := 110;
    v_salary      NUMBER;
    v_incentive   NUMBER;
BEGIN
  SELECT salary
  INTO v_salary
  FROM employees
  WHERE employee_id = v_employee_id;
v_incentive := v_salary * 0.10;
DBMS_OUTPUT.PUT_LINE ('Employee ID:' || v_employee_id || ', Salary:' ||
                      v_salary || ', Incentive :' || v_incentive);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
      DBMS.OUTPUT.PUT_LINE ('No employee found with ID:' || v_emp_id);
  WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE ('An error:' || SQLERRM);
END;
/
```

PROGRAM 2

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

```
DECLARE
    myvar  Number := 10;
    "Myvar" NUMBER := 20;
BEGIN
    DBMS_OUTPUT.PUT_LINE ('Non quoted Ident :' || myvar);
    DBMS_OUTPUT.PUT_LINE ('Quoted :' || "myvar");
EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE ('An error occ:' || SQLERRM);
END;
/
```

8.
```
DECLARE
    v_employee_id NUMBER := 122;
    v_new_salary NUMBER;
    v_current_salary NUMBER;
    v_adjustment_percentage NUMBER := 0.05;

BEGIN
    SELECT salary
    INTO v_current_salary
    FROM employees
    WHERE employee_id = v_employee_id;

v_new_salary := v_current_salary + (v_current_salary *
                                    v_adjustment_perceptage);

UPDATE employees
SET salary = v_new_salary
WHERE employee_id = v_employee_id;

COMMIT;

DBMS_OUTPUT.PUT_LINE ('EmployeeID:' || v_employee_id || ', New salary' ||
                      v_new_salary);

EXCEPTION
    When No_Data_Found THEN
        DBMS_OUTPUT.PUT_LINE ('No employee :' || v_employee_id);

    When OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('An error :' || SQLERRM);

END;
```

4)
```
create OR Replace Procedure check_emp_details (
    p_emp_id IN number,
    p_emp_name IN VARCHAR2)
AS v_count Number;
BEGIN.
    IF p_employee_id IS NOT NULL AND p_emp_name IS NOT NULL
    THEN
        Select count (*)
        Into v_count
        From emp
        Where emp_id = p_emp_id AND emp_name = p_emp_name

exception
    WHERE Others THEN
        DBMS_OUTPUT.PUT_LINE ("An error :' || SQLERRM);
END check_emp_details;
```

o/p:- Pattern 1 matched
Pattern 2 matched
Patter 3 matched with escape

o/p: small : 10
Large : 20;

PROGRAM 5

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

```
SET SERVER OUTPUT ON;
BEGIN
    IF 'HelloWorld' LIKE 'H%.W%.' THEN
        DBMS_OUTPUT.PUT.LINE ('Pattern 1matched');
    END IF;
IF 'Hello 123' LIKE 'Hello_23' THEN
        DBMS_OUTPUT.PUT_LINE ('Pattern 2matched');
  END IF;
IF '50%.'discount' LIKE '50%.%.' ESCAPE '\' THEN
 DBMS_OUTPUT.PUTLINE ("pattern 3 matched
with escape.');
```

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
        num 1 NUMBER := 10;
        num 2 NUMBER := 20;
      num_small NUMBER := LEAST (num1,num2);
      num_large NUMBER := GREATEST (num1,num2);

BEGIN
    DBMS_OUTPUT.PUT_LINE ('small:'||num_small||,
                          'Large:'||

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 calc_incentive
(emp_id IN number) IS BEGIN
UPDATE employees SET incentive = target_achieved
* 0.10 WHERE emp_id = emp_id AND TARGET
DBMS_OUTPUT_PUT_LINE ('Record' || CASE WHEN
SQL %. ROW COUNT > 0 THEN 'updated'.
END;
/
```

PROGRAM 8

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

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE calc_incentive
(emp_id IN NUMBER) IS
sales_limit_NUMBER = 1000;
incentive
BEGIN
SELECT CASE WHEN total_sales >= sales_limit
THEN total_sales * 0.10 UPDATE employees
SET incentive = incentive_amount WHERE emp_id = em
DBMS_OUTPUT_LINE ('Incentive for ID' || emp_id ||
':' || incentive_amount);
Exception
WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LIN
("Employee not Found");
/
```

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.

```
SET SERVER OUTPUT ON
DECLARE
   emp-count NUMBER;
BEGIN
SELECT COUNT (*) INTO emp-COUNT FROM employees
WHERE department_id =50;
DBMS_OUTPUT_PUT_LINE ('employees in DEPT 50:'||emp-count);
DBMS_OUTPUT-PUTLINE (IIF(emp-count <45, 'vacancies
    available.',));
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.

```
SET SERVEROUTPUT ON;
DECLARE
    emp-count NUMBER;
    vacancies NUMBER :=45;
BEGIN
    SELECT COUNT (*) INTO emp-count FROM employees WHERE
                                    department =50;
DBMS-OUTPUT.PUT_LINE ('Employees in DEPT 50:'||emp-count||
                    'vacancies ||
    (vacancies - emp_count));
    END;
/
```

PROGRAM 11

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

```
SET SERVER OUTPUT ON;
BEGIN
FOR rec IN (SELECT employee_id, name, job_title,
  hire_date, salary FROM
DBMS_OUTPUT.PUTLINE ('ID:' || rec.employee_id ||
                ', Name:' || rec.name ||
                ', Job Title :' || rec.job_title ||
                ', Hire Date:' || 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.

```
SET SERVER OUTPUT ON;
BEGIN
FOR rec IN (SELECT e.employee_id, e.name,
    d.department_name FROM employees e
JOIN departments d ON e.department_id = d.department
DBMS_OUTPUT.PUT_LINE ('ID:' || rec.employees_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.

```
SET SERVER OUTPUT ON;
BEGIN
FOR rec IN (SELECT job_id, job_title, min_salary
        FROM) Loop
DBMS_OUTPUT.PUTLINE ('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.

```
SET SERVEROUTPUT ON;
BEGIN
  FOR rec (SELECT e.employee_id e.name, j.start_date
      FROM employees e
  JOIN job_history j on e.employees_id = j.employee_id) -
DBMS_OUTPUT_LINE ('ID: '||rec.employee_id ||
            ', Name: '|| rec.name ||
            ', Job 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.

```
SET SERVEROUTPUT ON;
BEGIN
FOR rec IN (SELECT e.employee_id, e.name, j.end_date
        FROM employees e
        JOIN job_history j ON e.employee_id = j.employee_id)
DBMS_OUTPUT.PUT_LINE ('ID:' || rec.employee_id ||
                ', Name:' || rec.name ||
                ', Job End Date:' || rec.end_date);

END LOOP;
END;
/
```

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

Ex.No.: 12		WORKING WITH CURSOR, PROCEDURES AND FUNCTIONS
Date:		

AIM:

Create PL/SQL Blocks to perform the Item Transaction Operations using CURSOR, FUNCTION and PROCEDUERE.

ALGORITHM:

STEP-1: Start.

STEP-2: Create two tables Item Master and Item Trans.

itemmaster(itemid , itemname, stockonhand)

itemtrans(itemid ,itemname ,dateofpurchase ,quantity)

STEP-3: Create a PROCEDURE with id, name and quantity as parameters which make a call to the FUNCTION by passing id, name, dop, and quantity as parameters dop is set as sysdate.

STEP-4: Using FUNCTION fetch each record from the table Item Master using CURSOR inside a Loop statement,

If Item Master's ItemId is equal to the entered ID value then exit the loop otherwise fetch the next record.

```
loop
        fetch master into masterrec
        exit when master%notfound
        if masterrec.itemid=id then
    exit;
  end if;
end loop;
```

STEP-5: If Itemmaster's itemid = id then,

Add the Itemmaster's stockonhand with the given quantity and update the ItemMaster table and insert the Item information into the ItemTrans table.

STEP-6: Else, if the inputed item is not present in the ItemMaster table then insert the

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
CREATE OR REPLACE FUNCTION factorial (n IN number)
    RETURN NUMBER IS
result NUMBER := 1;
BEGIN
    IF n<0 then
      RETURN NULL;
  ELSIF n=0 OR n=1 THEN
    RETURN 1;
  ELSE for i IN 2..n loop
    RESULT := result * i ;
   END LOOP;
    END IF;
  RETURN RESULT;
EXCEPTION
    WHEN OTHERS THEN
DBMS_OUTPUT_PUT_LINE('AN ERROR OCCURED :
      || SQLERRM);
RETURN NULL;
END factorial;
/
SET SERVER OUTPUT ON;
DECLARE
      num NUMBER := 5;
      fact number;
END;
/
```

2. CREATE TABLE BOOKS (
book_id Number (5) PRIMARY KEY ,
title VARCHAR 2 (100) , author VARCHAR(100) ;
publications_year NUMBER (4), available_copies NUMBER (5)
);

INSERT INTO BOOKS VALUES (1, 'ASH', 'brwell', 1949, 4);
INSERT INTO BOOKS VALUES (2, 'Mocking Bird', 'Lee', 1960, 2);
INSERT INTO BOOKS VALUES (3; 'Gatsby', 'Fitzbald', 1925, 5);
COMMIT ;

CREATE OR REPLACE PROCEDURE GetBook InfoById
(P_book_id IN
NUMBER , P_title OUT VARCHAR 2 , P_author OUT
VARCHAR 2) IS
BEGIN
SELECT title , author INTO p_title , p_author FROM
BOOKS WHERE _Id = END;

SET SERVER OUTPUT ON;
DECLARE
V_title VARCHAR (100); V_author VARCHAR 2 (100);
BEGIN
Get book In fomyId (1, V_title , V_author),
DBMS_ OUTPUT.PUTLINE ('Book': ||V_title|| ', Author :' ||V_author);
END;
/

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