Practice 11: Reporting Aggregated Data Using the Group Functions

Jesús Manuel Juárez Pasillas

October 28, 2021

1 Introduction

Group functions allow us to obtain results from a group of elements with which information of great relevance can be extracted since all the rows of a table can be taken, or the rows can be grouped so that they are several results depending on several groups formed.
The functions in Oracle allow us to carry out a great variety of operations with the data that we have, they also allow us to modify, represent them in another format, among other things.

2 Development

2.1 Activity 1:

Read all the choices carefully because there might be more than one correct answer. Choose all the correct answers for each question.

1. What result is returned by the following statement?
   SELECT COUNT(*) FROM DUAL;

   A. NULL

   B. 0

   C. **1**

   D. None of the above

   Since the dual table only has one record, the count function will return 1.

2. Choose one correct statement regarding group functions.

   A. Group functions may only be used when a GROUP BY clause is present.

   B. Group functions can operate on multiple rows at a time.

C. Group functions only operate on a single row at a time.

D. **Group functions can execute multiple times within a single group.**

You can run as many functions as you want in a single group.

3. What value is returned after executing the following statement?
SELECT SUM(SALARY) FROM EMPLOYEES;

   A. **900**

   B. 1000

   C. NULL

   D. None of the above

   The result will be 900 since the sum function does not take null values into account.

4. Which values are returned after executing the following statement?
SELECT COUNT(*), COUNT(SALARY) FROM EMPLOYEES;

   A. 10 and 10

   B. 10 and NULL

   C. **10 and 9**

   D. None of the above

   As there are 10 records, 10 will appear in the first count, in the second count 9 will appear since a value is null and it does not count.

5. What value is returned after executing the following statement?
SELECT AVG(NVL(SALARY,100)) FROM EMPLOYEES;

   A. NULL

   B. 90

   C. **100**

   D. None of the above

   The nvl function makes all that null salary value become 100 so this would take the average of the 10 records where all the values entered into the avg function are 100.

6. What value is returned after executing the following statement?
SELECT SUM((AVG(LENGTH(NVL(SALARY,0)))))
FROM EMPLOYEES GROUP BY SALARY;

   A. An error is returned

   B. 3

C. **4**

D. None of the above

It would come out 3 in the average of the salaries that are 100, since it is the sum of the length of their salaries and divided by the 9 elements that would come out, and it would come out 1 in the averages that are null, therefore, the sum of these 2 numbers would be 4.

7. How many records are returned by the following query?
SELECT SUM(SALARY), DEPARTMENT_ID
FROM EMPLOYEES GROUP BY DEPARTMENT_ID;

A. **12**

B. 11

C. NULL

D. None of the above

It returns 12 rows since there are 11 with different values for department_id and a null value with which it also groups.

8. What values are returned after executing the following statement?
SELECT JOB_ID, MAX_SALARY
FROM JOBS GROUP BY MAX_SALARY;

A. One row of output with the values DBA, 100

B. Ten rows of output with the values DBA, 100

C. **An error is returned**

D. None of the above

Returns an error because it does not have a group function with which to group the results.

9. How many rows of data are returned after executing the following statement?
SELECT DEPT_ID, SUM(NVL(SALARY,100)) FROM EMP
GROUP BY DEPT_ID HAVING SUM(SALARY) > 400;

A. Two rows

B. **One row**

C. Zero rows

D. None of the above

It returns a row since with the nvl function it makes no salary value null and it would always have 100 in salary, this groups it by the id that in this case there are only 2 different id and as in the having section it does not use the nvl function to remove the null, it only returns a row that would be where the salary is not null.

10. How many rows of data are returned after executing the following statement?
SELECT DEPT_ID, SUM(SALARY) FROM EMP
GROUP BY DEPT_ID HAVING SUM(NVL(SALARY,100)) > 400;

    A. **Two rows**

    B. One row

    C. Zero rows

    D. None of the above

    It returns two rows since in the having clause it contains a function with which no salary value is null, so the resulting sums of the groupings give more than 400.

2.2 Activity 2:

Propose an answer to the following issues:

- You would like to retrieve the earliest date from a column that stores DATE information. Can a group function be utilized to retrieve this value?
  - **Yes, with the min function which works for dates, the oldest date required is obtained.**
- Summary statistics are required by senior management. This includes details like number of employees, total staff salary cost, lowest salary, and highest salary values. Can such a report be drawn using one query?
  - **Yes, this can be done using the count, sum, min and max functions with which you will get all this information.**
- You are asked to list the number of unique jobs performed by employees in the organization. Counting the JOB_ID records will give you all the jobs. Is it possible to count the unique jobs?
  - **Yes, with the count function you can count the unique jobs performed, in addition to including the group by clause to group the results for each employee.**

- You wish to print name badges for the staff who work as sales representatives. Can the length of the shortest and longest LAST_NAME values be determined for these employees?
  - **Yes, with the max and min function where inside they have the length function and this function contains the last_name column.**
- Is it possible to count the records in each group, first by dividing the employee records by year of employment, then by job, and finally by salary?
  - **Yes, the group by clause sets the columns by which you want to group in order separated by commas.**
- Is there a limit to the number of groups within groups that can be formed?
  - **There is no defined limit on the groups within the group, but it is better not to do this to avoid confusion.**

2.3 Activity 3:

Connect to the OE schema and complete the following tasks.
Using SQL Developer, connect to the OE schema and complete the following tasks. The PRODUCT_INFORMATION table lists items that are orderable and others that are planned, obsolete, or under development. You are required to prepare a report that groups the non-orderable products by their PRODUCT_STATUS and shows the number of products in each group and the sum of the LIST_PRICE of the products per group. Further, only the group-level rows, where the sum of the LIST_PRICE is greater than 4000, must be displayed. A product is non-orderable if the PRODUCT_STATUS value is not equal to the string 'orderable'.

- **Select count(*), sum(list_price), product_status from Product_Information where product_status != 'orderable' group by product_status having sum(list_price) > 4000;**

	COUNT(*)	SUM(LIST_PRICE)	PRODUCT_STATUS
1	11	4922	under development
2	16	7389	obsolete

Figure 1: Consult the oe scheme.

2.4 Activity 4:

This exercise must be performed using HR schema.

- The COUNTRIES table stores a list of COUNTRY_NAME values. You are required to calculate the average length of all the country names. Any fractional components must be rounded to the nearest whole number.
  - **Select round(avg(length(country_name))) average_country_name_length from Countries;**

	AVERAGE_COUNTRY_NAME_LENGTH
1	8

Figure 2: Average length of all the country names.

- Analysis of staff turnover is a common reporting requirement. You are required to create a report containing the number of employees who left their jobs, grouped by the year in which they left. The jobs they performed are also required. The results must be sorted in descending order based on the number of employees in each group. The report must list the year, the JOB_ID, and the number of employees who left a particular job in that year.
  - **Select to_char(end_date,'yyyy') "Quitting Year", job_id, count(*) "Number of Employees" from Job_History group by to_char(end_date,'yyyy'), job_id order by count(*) desc;**

	Quitting Year	JOB_ID	Number of Employees
1	2007	ST_CLERK	2
2	2001	AC_ACCOUNT	1
3	2005	AC_MGR	1
4	2006	AC_ACCOUNT	1
5	2006	SA_REP	1
6	2001	AD_ASST	1
7	2006	IT_PROG	1
8	2007	MK_REP	1
9	2007	SA_MAN	1

Figure 3: Quitting Year.

- The company is planning a recruitment drive and wants to identify the days of the week on which 15 or more staff members were hired. Your

report must list the days and the number of employees hired on each of them.

- **Select to_char(hire_date,'day'), count(*)**
  from Employees group by to_char(hire_date,'day')
  having count(*) >= 15;

	TO_CHAR(HIRE_DATE,'DAY')	COUNT(*)
1	sábado	19
2	jueves	16
3	miércoles	15
4	domingo	15
5	viernes	19

Figure 4: Days of the week on which 15 or more staff members were hired.

2.5 Activity 5:

Determine the validity of the following three statements. Circle either True or False and explain the reason.

1. Group functions work across many rows to produce one result per group.
   True/False
   - **True since the group functions what they do is receive several rows with which they obtain a single result.**
2. Group functions include nulls in calculations.
   True/**False**
   - **Group functions don't count columns that have a null value, it just ignores them.**
3. The WHERE clause restricts rows before inclusion in a group calculation.
   True/False
   - **True, the where clause is executed first than the group clause since it is the condition that the rows have to fulfill in order to show those results, then they are added to the groups.**
4. Find the highest, lowest, sum, and average salary of all employees. Label the columns as Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number. Save your SQL statement as lab_11_04.sql. Run the query.
   - **The file is located inside the file called "Plungins.zip".**

	Maximum	Minimum	Sum	Average
1	24000	2100	691416	6462

Figure 5: Highest, lowest, sum, and average salary.

5. Modify the query in lab_11_04.sql to display the minimum, maximum, sum, and average salary for each job type. Resave lab_11_04.sql as lab_11_05.sql. Run the statement in lab_11_05.sql.

   - **The file is located inside the file called "Plungins.zip".**

	JOB_ID	Maximum	Minimum	Sum	Average
1	IT_PROG	9000	4200	28800	5760
2	AC_MGR	12008	12008	12008	12008
3	AC_ACCOUNT	8300	8300	8300	8300
4	ST_MAN	8200	5800	36400	7280
5	PU_MAN	11000	11000	11000	11000
6	AD_ASST	4400	4400	4400	4400
7	AD_VP	17000	17000	34000	17000
8	SH_CLERK	4200	2500	64300	3215
9	FI_ACCOUNT	9000	6900	39600	7920
10	FI_MGR	12008	12008	12008	12008
11	PU_CLERK	3100	2500	13900	2780

Figure 6: Highest, lowest, sum, and average salary for each job type.

6. Write a query to display the number of people with the same job.

   - **Select job_id, count(*) from Employees group by job_id;**

	JOB_ID	COUNT(*)
1	AC_ACCOUNT	1
2	AC_MGR	1
3	AD_ASST	1
4	AD_PRES	1
5	AD_VP	2
6	FI_ACCOUNT	5
7	FI_MGR	1
8	HR_REP	1
9	IT_PROG	5
10	MK_MAN	1

Figure 7: Number of people with the same job.

Generalize the query so that the user in the HR department is prompted for a job title. Save the script to a file named lab_11_06.sql. Run the query. Enter IT_PROG when prompted.

- **The file is located inside the file called "Plungins.zip".**

	JOB_ID	COUNT(*)
1	IT_PROG	5

Figure 8: Prompted for a job title.

7. Determine the number of managers without listing them. Label the column as Number of Managers. Hint: Use the MANAGER_ID column to determine the number of managers.

   - **Select count(distinct manager_id) "Number of Managers" from Employees;**

	Number of Managers
1	18

Figure 9: Number of managers.

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

	DIFERENCE
1	21900

Figure 10: Difference between the highest and lowest salaries.

- **Select (max(salary)-min(salary)) Diference from Employees;**

9. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

   - **Select manager_id, min(salary) from Employees where manager_id is not null group by manager_id having min(salary) > 6000 order by min(salary) desc;**

	MANAGER_ID	MIN(SALARY)
1	102	9000
2	205	8300
3	145	7000
4	146	7000
5	108	6900
6	147	6200
7	149	6200
8	148	6100

Figure 11: Manager number and the salary of the lowest-paid.

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 (the years were changed because the records we have are from the year 2001.).

    - **Select count(*) total,**
      count(nvl2(nullif(to_char(hire_date,'yyyy'),'2001'),
      null,hire_date)) "2001",
      count(nvl2(nullif(to_char(hire_date,'yyyy'),'2002'),
      null,hire_date)) "2002",
      count(nvl2(nullif(to_char(hire_date,'yyyy'),'2003'),
      null,hire_date)) "2003",
      count(nvl2(nullif(to_char(hire_date,'yyyy'),'2004'),

null,hire_date)) "2004"
from Employees where to_char(hire_date,'yyyy')
in ('2001','2002','2003','2004');

	TOTAL	2001	2002	2003	2004
1	24	1	7	6	10

Figure 12: The number of employees hired in 2001, 2002, 2003, and 2004.

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 "Job",**
      nvl2(nullif(department_id,20),null,max(salary)) "Dept 20",
      nvl2(nullif(department_id,50),null,max(salary)) "Dept 50",
      nvl2(nullif(department_id,80),null,max(salary)) "Dept 80",
      nvl2(nullif(department_id,90),null,max(salary)) "Dept 90",
      sum(salary) "Total" from Employees where department_id
      in (20,50,80,90) group by job_id,department_id;

	Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
1	ST_MAN	(null)	8200	(null)	(null)	36400
2	SA_REP	(null)	(null)	11500	(null)	243500
3	AD_VP	(null)	(null)	(null)	17000	34000
4	MK_MAN	13000	(null)	(null)	(null)	13000
5	MK_REP	6000	(null)	(null)	(null)	6000
6	SA_MAN	(null)	(null)	14000	(null)	61000
7	SH_CLERK	(null)	4200	(null)	(null)	64300
8	AD_PRES	(null)	(null)	(null)	24000	24000
9	ST_CLERK	(null)	3600	(null)	(null)	55700

Figure 13: Job, salary for that job based on department and salary total.

3 Pre-Assessment:

- Practices pre-Assessment for Database Systems Laboratory II

Practice	Pre-Assessment
COMPLIES WITH THE REQUESTED FUNCTIONALITY	X
HAS THE CORRECT INDENTATION	X
HAS AN EASY WAY TO ACCESS THE PROVIDED FILES	X
HAS A REPORT WITH IDC FORMAT	X
REPORT INFORMATION IS FREE OF SPELLING ERRORS	X
DELIVERED IN TIME AND FORM	X
IS FULLY COMPLETED (SPECIFY THE PERCENTAGE COMPLETED)	100%

Table 1: Pre-Assessment.

4 Conclusion:

Group functions are essential when you want to know important information about a complete table or about some groups that can be formed according to different characteristics that the data of these groups share. Group functions can be supplemented with single-line functions which makes for more concrete results.