Report

Laboratory Work 6

Dmitry Ladutsko

August 14, 2022

1. Prerequisites Task Information

1.1. Passwords Index

Password Group	Login Name	Password
Operation System	root	“rootadmin”
	oracle	“oracleadmin”
Oracle System	sys	“sysadmin”
	system	“sysadmin”
Oracle Users	All DB users	“%PWD%”

1.2. Folder Paths Index

Path Group	Path Description	Path
Operation System	Oracle RDBMS – BIN	/oracle/app/oracle
	Oracle Inventory	/oracle/app/oraInventory
	Oracle Database Storage	/oracle/oradata
	Oracle Install Directory	/oracle/install
Oracle	ORACLE_BASE	/oracle/app/oracle
	ORACLE_HOME	$ORACLE_BASE/product/11.2
FTP	ftp Incoming Folder	/ftp/incoming

2. Analytic Functions - Basic

2.1. Task 01: Create Ad Hoc SQL FIRST_VALUE, LAST_VALUE

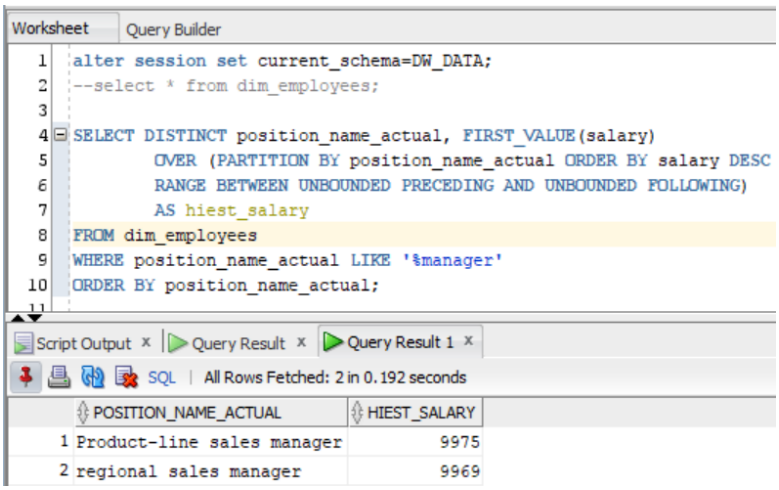
The Main Task is to create ad hoc SQL, which will analyze measurement using Analytic Functions

Required points:

Use Analytic Function:

- FIRST_VALUE, LAST_VALUE

Note. Firs_value() is a function which shows the first value in selected partition
So, lets see the highest salary in managers position



```
16  SELECT DISTINCT product_name, LAST_VALUE(price)
17          OVER (PARTITION BY product_name ORDER BY price DESC
18          RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
19          AS lowest_price_product
20  FROM dim_products
21  WHERE product_name IN('Iphone')
22  ORDER BY product_name;
```

Script Output | Query Result | Query Result 1 | Query Result 2

Task completed in 0.082 seconds

PRODUCT_NAME	LOWEST_PRICE_PRODUCT
Iphone	599

Note. Here we used Last_value() function to see the minimum price for iPhone

2.2. Task 02: Create Ad Hoc SQL RANK, DENSE_RANK, ROWNUM

The Main Task is to create ad hoc SQL, which will split rows of data by Groups using Analytic Functions

Required points:

Use Analytic Function:

- RANK, DENSE_RANK, ROWNUM

```
4
5  select FIRST_NAME_EMPLOYEE,
6         salary,
7         RANK() OVER (PARTITION BY POSITION_NAME_ACTUAL ORDER BY salary) SALARY_RANK
8    from dim_employees
9   where POSITION_NAME_ACTUAL = 'Manager';
```

Script Output | Query Result

SQL | Fetched 250 rows in 0.13 seconds

	FIRST_NAME_EMPLOYEE	SALARY	SALARY_RANK
1	Ali	1012	1
2	Anibal	1021	2
3	Maxim	1053	3
4	Stasy	1083	4
5	Sara	1135	5
6	Abrigel	1222	6
7	Salamon	1296	7
8	Abrigel	1370	8
9	Samuel	1402	9
10	Sandy	1514	10
11	Mohammed	1527	11
12	Vika	1535	12
13	Sara	1623	13
14	Kumill	1636	14

Note. Here we use **RANK** analytic function to see ranked employees salaries

```
11  ----------DENSE_RANK----------
12
13  select FIRST_NAME_EMPLOYEE,
14         salary,
15         DENSE_RANK() OVER (PARTITION BY POSITION_NAME_ACTUAL ORDER BY salary) SALARY_RANK
16    from dim_employees
17   where POSITION_NAME_ACTUAL = 'Manager';
18
```

Script Output | Query Result

SQL | Fetched 100 rows in 0.049 seconds

	FIRST_NAME_EMPLOYEE	SALARY	SALARY_RANK
1	Ali	1012	1
2	Anibal	1021	2
3	Maxim	1053	3
4	Stasy	1083	4
5	Sara	1135	5
6	Abrigel	1222	6
7	Salamon	1296	7
8	Abrigel	1370	8
9	Samuel	1402	9
10	Sandy	1514	10
11	Mohammed	1527	11
12	Vika	1535	12

Note. Here we use **DENSE_RANK** analytic function to see **DENSE**_ranked employees' salaries

Note. The difference between this two analytic functions is that **RANK** function may return a **non-consecutive ranking** if the values being tested **are the same**. Whereas, the **DENSE_RANK** function will always result in a **sequential ranking**.

2.3. Task 03: Create Ad Hoc SQL AGGREAGATE FUNCS

The Main Task is to create ad hoc SQL, which will analyze measurement using Analytic Functions

Required points:

Use Analytic Function:

- AGGREAGATE FUNCS (MAX, MIN, AVG)

```
2  ----------SUM----------
3  SELECT SUM(salary) AS TOTAL_SALARIES
4  FROM dim_employees
5  WHERE position_name_actual LIKE '%manager';
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.031 seconds

	TOTAL_SALARIES
1	2849501

Note. Here we can see what **amount of money** have we paid to all of our Manages

```
7  ----------------------------AVG---------------------------------
8  SELECT POSITION_NAME_ACTUAL AS DEPARTAMENT, trunc(AVG(salary)) AS AVERAGE_SALARY
9  FROM dim_employees
10 GROUP BY POSITION_NAME_ACTUAL;
```

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 4 in 0.023 seconds

	DEPARTAMENT	AVERAGE_SALARY
1	Manager	5723
2	Sales Manager	5479
3	Product-line sales manager	5735
4	regional sales manager	5843

Note. Here we can see **AVERAGE** salary we paid to our employees grouped by departments

```
11 ----------------------------MIN---------------------------------
12 SELECT trunc(MIN(salary)) AS MINIMAL_SALARY
13 FROM dim_employees;
```

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 1 in 0.022 seconds

	MINIMAL_SALARY
1	1012

Note. Here we can see **MINIMAL** salary we paid in company

```
15 ----------------------------MAX---------------------------------
16 SELECT trunc(MAX(salary)) AS min_salary_in_company
17 FROM dim_employees;
```

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 1 in 0.014 seconds

	MIN_SALARY_IN_COMPANY
1	9995

Note. Here we can see **MAXIMUM** salary we paid in company

Laboratory Work Summary: At this laboratory work we practised usage of analytic and aggregative functions, such as:

- SUM
- AVG
- MAX
- MIN
- FIRST_VALUE
- LAST_VALUE
- RANK
- DENSE_RANK
- ROWNUM