

U2M6.LW.Analytic Functions

Shkrabatouskaya Vera

https://github.com/VeraShkrabatouskaya/DataMola_Data-Camping-2022

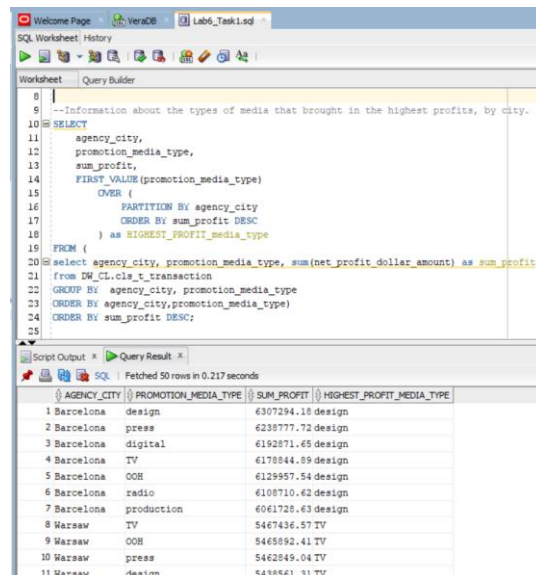
2. Analytic Functions - Basic

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

Create ad hoc SQL, which will analyze measurement using Analytic Functions:

- FIRST_VALUE, LAST_VALUE

Let's get information about the types of media that brought in the highest profits, by city.



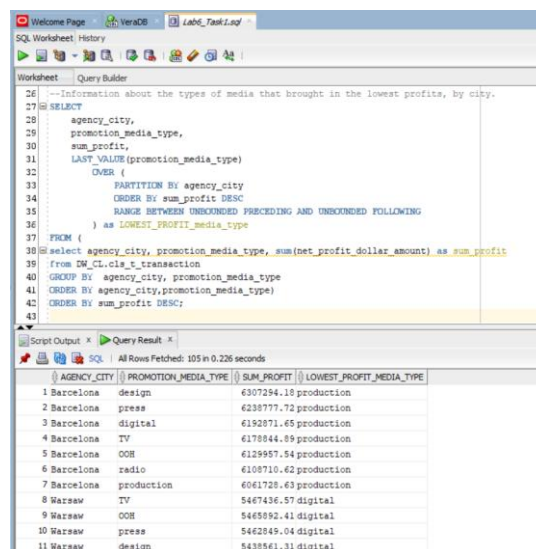
The screenshot shows a SQL Worksheet with a query that uses FIRST_VALUE to find the highest profit media type for each city. The query is as follows:

```
--Information about the types of media that brought in the highest profits, by city.
SELECT
  agency_city,
  promotion_media_type,
  sum_profit,
  FIRST_VALUE(promotion_media_type)
    OVER (
      PARTITION BY agency_city
      ORDER BY sum_profit DESC
    ) as HIGHEST_PROFIT_media_type
FROM (
  select agency_city, promotion_media_type, sum(net_profit_dollar_amount) as sum_profit
  from DW_CL.cla_t_transaction
  GROUP BY agency_city, promotion_media_type
  ORDER BY agency_city, promotion_media_type
  ORDER BY sum_profit DESC;
```

The query result shows 11 rows of data:

AGENCY_CITY	PROMOTION_MEDIA_TYPE	SUM_PROFIT	HIGHEST_PROFIT_MEDIA_TYPE
1 Barcelona	design	6307294.18	design
2 Barcelona	press	6238777.72	design
3 Barcelona	digital	6192871.65	design
4 Barcelona	TV	6178844.89	design
5 Barcelona	OOB	6129957.54	design
6 Barcelona	radio	6108710.62	design
7 Barcelona	production	6061728.63	design
8 Warsaw	TV	5467436.57	TV
9 Warsaw	OOB	5465892.41	TV
10 Warsaw	press	5462849.04	TV
11 Warsaw	design	5438561.31	TV

Let's get information about the types of media that brought in the lowest profits, by city.



The screenshot shows a SQL Worksheet with a query that uses LAST_VALUE to find the lowest profit media type for each city. The query is as follows:

```
--Information about the types of media that brought in the lowest profits, by city.
SELECT
  agency_city,
  promotion_media_type,
  sum_profit,
  LAST_VALUE(promotion_media_type)
    OVER (
      PARTITION BY agency_city
      ORDER BY sum_profit DESC
      RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
    ) as LOWEST_PROFIT_media_type
FROM (
  select agency_city, promotion_media_type, sum(net_profit_dollar_amount) as sum_profit
  from DW_CL.cla_t_transaction
  GROUP BY agency_city, promotion_media_type
  ORDER BY agency_city, promotion_media_type
  ORDER BY sum_profit DESC;
```

The query result shows 11 rows of data:

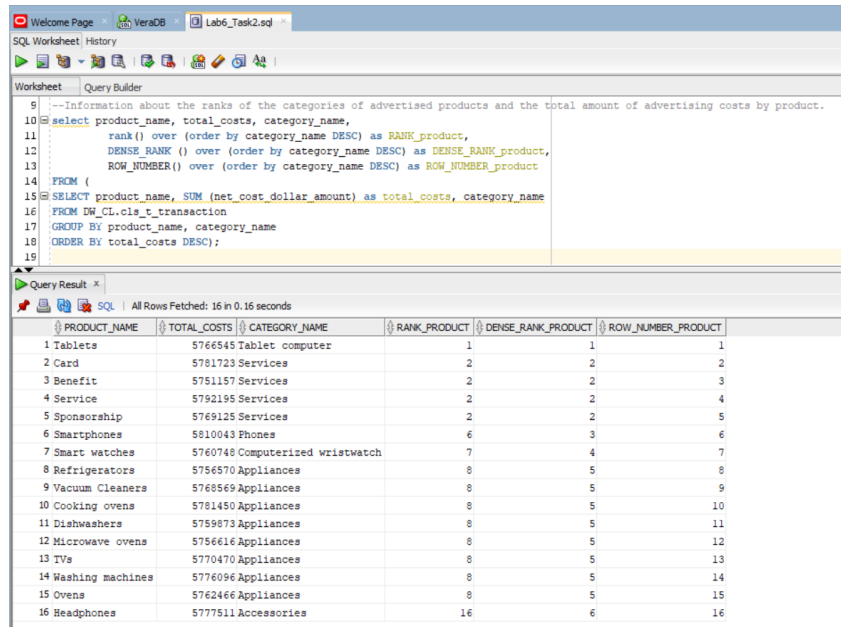
AGENCY_CITY	PROMOTION_MEDIA_TYPE	SUM_PROFIT	LOWEST_PROFIT_MEDIA_TYPE
1 Barcelona	design	6307294.18	production
2 Barcelona	press	6238777.72	production
3 Barcelona	digital	6192871.65	production
4 Barcelona	TV	6178844.89	production
5 Barcelona	OOB	6129957.54	production
6 Barcelona	radio	6108710.62	production
7 Barcelona	production	6061728.63	production
8 Warsaw	TV	5467436.57	digital
9 Warsaw	OOB	5465892.41	digital
10 Warsaw	press	5462849.04	digital
11 Warsaw	design	5438561.31	digital

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

Create ad hoc SQL, which will split rows of data by Groups using Analytic Functions:

- RANK, DENSE_RANK, ROWNUM

Let's get information about the ranks of the categories of advertised products and the total amount of advertising costs by product.



The screenshot shows a SQL Worksheet with a query and its results. The query is as follows:

```
--Information about the ranks of the categories of advertised products and the total amount of advertising costs by product.
select product_name, total_costs, category_name,
       rank() over (order by category_name DESC) as RANK_product,
       DENSE_RANK () over (order by category_name DESC) as DENSE_RANK_product,
       ROW_NUMBER() over (order by category_name DESC) as ROW_NUMBER_product
FROM (
SELECT product_name, SUM(net_cost_dollar_amount) as total_costs, category_name
FROM DW_CLS_t_transaction
GROUP BY product_name, category_name
ORDER BY total_costs DESC);
```

The query results are displayed in a table with 6 columns: PRODUCT_NAME, TOTAL_COSTS, CATEGORY_NAME, RANK_PRODUCT, DENSE_RANK_PRODUCT, and ROW_NUMBER_PRODUCT. The results are ordered by total costs in descending order.

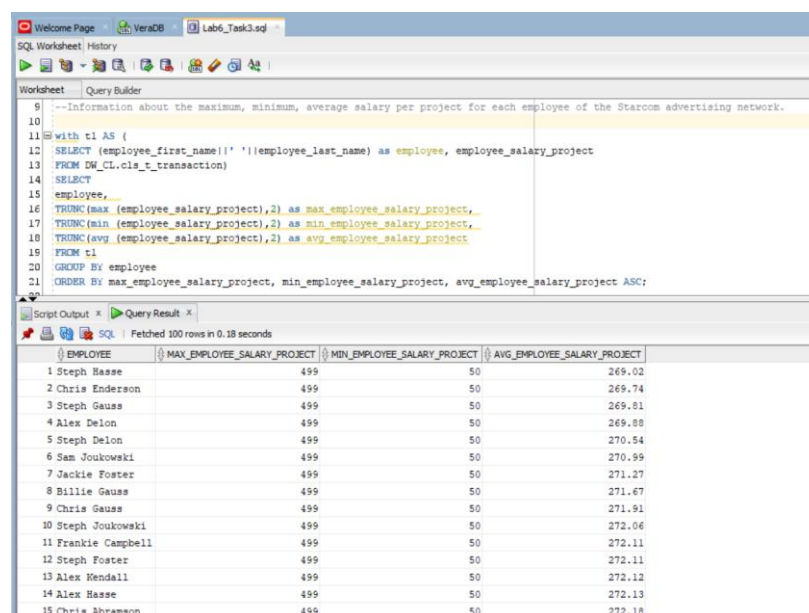
PRODUCT_NAME	TOTAL_COSTS	CATEGORY_NAME	RANK_PRODUCT	DENSE_RANK_PRODUCT	ROW_NUMBER_PRODUCT
1 Tablets	5766545	Tablet computer	1	1	1
2 Card	5781723	Services	2	2	2
3 Benefit	5751157	Services	2	2	3
4 Service	5782195	Services	2	2	4
5 Sponsorship	5769125	Services	2	2	5
6 Smartphones	5810043	Phones	6	3	6
7 Smart watches	5760748	Computerized wristwatch	7	4	7
8 Refrigerators	5756570	Appliances	8	5	8
9 Vacuum Cleaners	5768569	Appliances	8	5	9
10 Cooking ovens	5781450	Appliances	8	5	10
11 Dishwashers	5759873	Appliances	8	5	11
12 Microwave ovens	5756616	Appliances	8	5	12
13 TVs	5770470	Appliances	8	5	13
14 Washing machines	5776096	Appliances	8	5	14
15 Ovens	5762466	Appliances	8	5	15
16 Headphones	5777511	Accessories	16	6	16

2.3. Task 03: Create Ad Hoc SQL AGGREGATE FUNCS

Create ad hoc SQL, which will analyze measurement using Analytic Functions:

- AGGREGATE FUNCS (MAX, MIN, AVG)

Let's get information about the maximum, minimum, average salary per project for each employee of the Starcom advertising network.



The screenshot shows a SQL Worksheet with a query and its results. The query is as follows:

```
--Information about the maximum, minimum, average salary per project for each employee of the Starcom advertising network.
with t1 AS (
SELECT (employee_first_name||' '||employee_last_name) as employee, employee_salary_project
FROM DW_CLS_t_transaction)
SELECT
employee,
TRUNC(max(employee_salary_project),2) as max_employee_salary_project,
TRUNC(min(employee_salary_project),2) as min_employee_salary_project,
TRUNC(avg(employee_salary_project),2) as avg_employee_salary_project
FROM t1
GROUP BY employee
ORDER BY max_employee_salary_project, min_employee_salary_project, avg_employee_salary_project ASC;
```

The query results are displayed in a table with 4 columns: EMPLOYEE, MAX_EMPLOYEE_SALARY_PROJECT, MIN_EMPLOYEE_SALARY_PROJECT, and AVG_EMPLOYEE_SALARY_PROJECT. The results are ordered by maximum salary in ascending order.

EMPLOYEE	MAX_EMPLOYEE_SALARY_PROJECT	MIN_EMPLOYEE_SALARY_PROJECT	AVG_EMPLOYEE_SALARY_PROJECT
1 Steph Hasse	499	50	269.02
2 Chris Enderson	499	50	269.74
3 Steph Gauss	499	50	269.81
4 Alex Delon	499	50	269.88
5 Steph Delon	499	50	270.54
6 Sam Joukowski	499	50	270.99
7 Jackie Foster	499	50	271.27
8 Billie Gauss	499	50	271.67
9 Chris Gauss	499	50	271.91
10 Steph Joukowski	499	50	272.06
11 Frankie Campbell	499	50	272.11
12 Steph Foster	499	50	272.11
13 Alex Kendall	499	50	272.12
14 Alex Hasse	499	50	272.13
15 Chris Abramson	499	50	272.18