Lab report #10 Sadovskaya Veronika

GitHub: https://github.com/sdveronika/DataMola22

Task 1 - Transformation Description

Transforming Data Using PL/SQL

In a data warehouse environment, you can use procedural languages such as PL/SQL to implement complex transformations in the Oracle Database. Whereas CTAS operates on entire tables and emphasizes parallelism, PL/SQL provides a row-based approached and can accommodate very sophisticated transformation rules. For example, a PL/SQL procedure could open multiple cursors and read data from multiple source tables, combine this data using complex business rules, and finally insert the transformed data into one or more target table. It would be difficult or impossible to express the same sequence of operations using standard SQL statements.

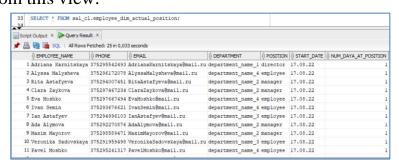
Using a procedural language, a specific transformation (or number of transformation steps) within a complex ETL processing can be encapsulated, reading data from an intermediate staging area and generating a new table object as output. A previously generated transformation input table and a subsequent transformation will consume the table generated by this specific transformation. Alternatively, these encapsulated transformation steps within the complete ETL process can be integrated seamlessly, thus streaming sets of rows between each other without the necessity of intermediate staging. You can use table functions to implement such behavior.

Task 2 - Loading to SAL Layer Data

Let's create a view that contains information about the duration of work of all employees in the current position:



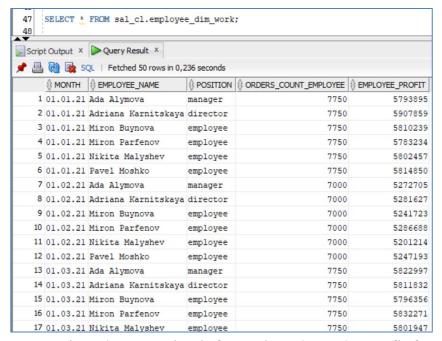
Select from this view:



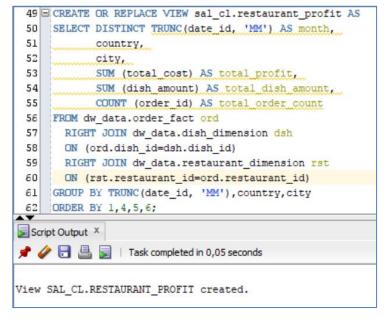
Let's create a view that contains information about the amount of work of each employee per month (number of orders and total profit):

```
35 CREATE OR REPLACE VIEW sal_cl.employee_dim_work AS
     SELECT DISTINCT TRUNC(date_id, 'MM') AS month,
 37
             first name | | ' ' | | last name AS employee name,
             job title AS position,
 38
 39
            COUNT(order id) AS orders count employee,
 40
            SUM(total_cost) AS employee profit
 41 FROM dw_data.order_fact ord
      RIGHT JOIN dw data.employee dimension emp
 42
       ON (ord.employee_id=emp.employee_id)
     GROUP BY TRUNC(date_id, 'MM'), first_name|| ' ' || last_name, job_title
 44
     ORDER BY 1,2,3,4,5;
📌 🧼 🔡 遏 | Task completed in 0,045 seconds
View SAL CL.EMPLOYEE DIM WORK created.
```

Select from this view:



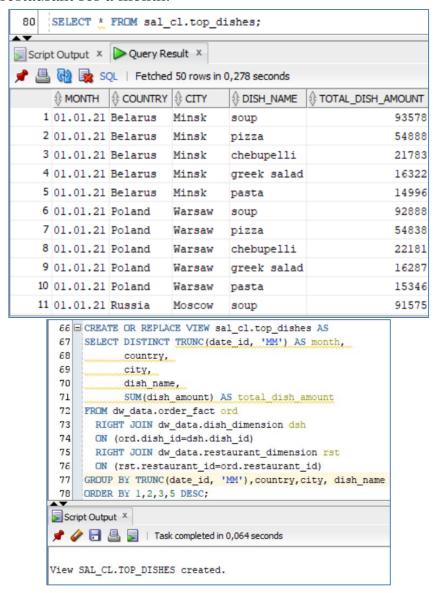
Let's create a view that contains information about the profit for the month for each restaurant, the number of dishes sold and the total number of orders:



Select from this view:

64 SELECT * FROM sal_cl.restaurant_profit;						
Script Output × Query Result ×						
📌 🚇 🔞 sql. Fetched 50 rows in 0,253 seconds						
	∯ MONTH			↑ TOTAL_PROFIT	TOTAL_DISH_AMOUNT	↑ TOTAL_ORDER_COUNT
1	01.01.21	Ukraine	Kiev	6949272	200707	9300
2	01.01.21	Russia	Moscow	6963031	200022	9300
3	01.01.21	USA	New York	6973329	199000	9300
4	01.01.21	Belarus	Minsk	6992781	201567	9300
5	01.01.21	Poland	Warsaw	7034121	201540	9300
6	01.02.21	Russia	Moscow	6231573	179280	8400
7	01.02.21	USA	New York	6259544	179442	8400
8	01.02.21	Ukraine	Kiev	6324705	181011	8400
9	01.02.21	Poland	Warsaw	6352629	181906	8400
10	01.02.21	Belarus	Minsk	6362699	183313	8400
11	01.03.21	Belarus	Minsk	6942576	197324	9300
12	01.03.21	Ukraine	Kiev	6960043	199358	9300
13	01.03.21	Russia	Moscow	6971340	198941	9300
14	01.03.21	USA	New York	6974337	200391	9300
15	01.03.21	Poland	Warsaw	6991878	201533	9300

Let's create a view that contains information about the popularity of each dish in each restaurant for a month:

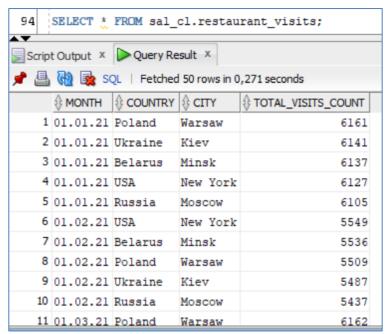


Select from this view:

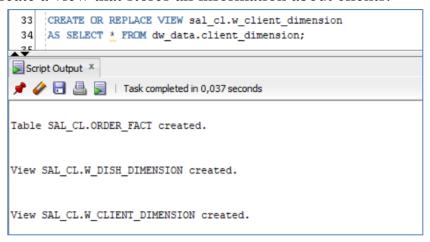
Let's create a view that contains information about the attendance of each restaurant for a month:

```
82 E CREATE OR REPLACE VIEW sal cl.restaurant visits AS
 83
     SELECT DISTINCT TRUNC (date id, 'MM') AS month,
 84
            country,
 85
           city,
 86
           COUNT (order id) AS total visits count
 87
    FROM dw_data.order_fact ord
      RIGHT JOIN dw_data.restaurant_dimension rst
 88
 89
       ON (rst.restaurant_id=ord.restaurant_id)
 90 WHERE delivery='N'
    GROUP BY TRUNC (date id, 'MM'), country, city
 91
 92
    ORDER BY 1,4 DESC;
Script Output X
📌 🥟 🔒 💂 📗 | Task completed in 0,055 seconds
View SAL CL.RESTAURANT VISITS created.
```

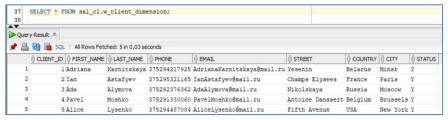
Select from this view:



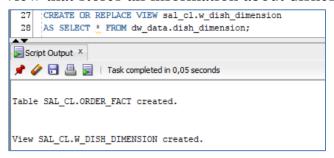
Let's create a view that stores all information about clients:



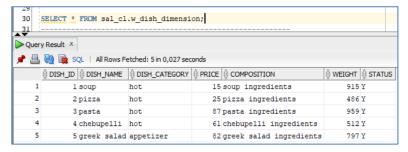
Select from this view:



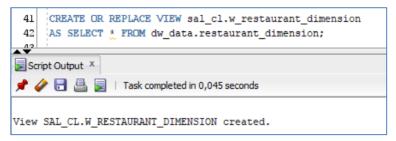
Let's create a view that stores all information about dishes:



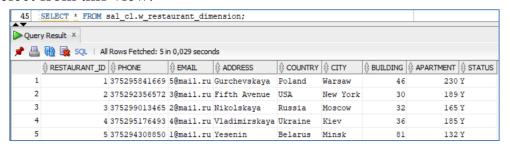
Select from this view:



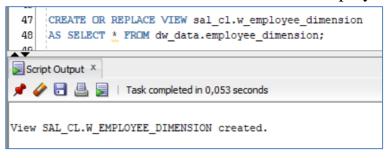
Let's create a view that stores all information about restaurants:



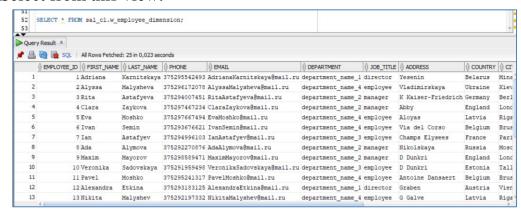
Select from this view:



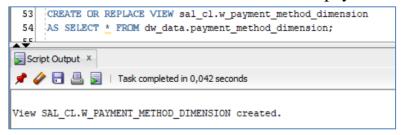
Let's create a view that stores all information about employees:



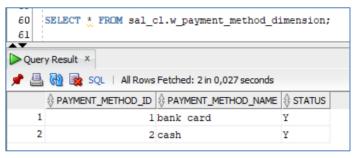
Select from this view:



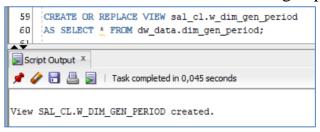
Let's create a view that stores all information about payment methods:



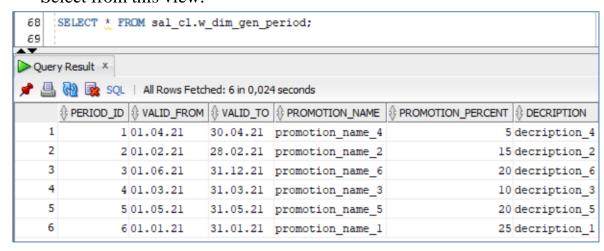
Select from this view:



Let's create a view that stores all information about gen periods:



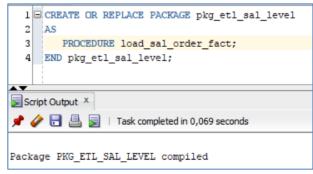
Select from this view:

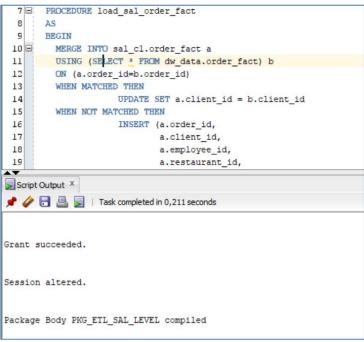


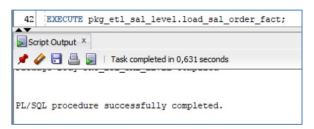
Let's create a table that stores all information about orders:

```
3 CREATE TABLE sal cl.order fact (
  4 order_id NUMBER,
  5 client id NUMBER NOT NULL,
  6 employee id NUMBER NOT NULL,
  7 restaurant id NUMBER NOT NULL,
  8 date id DATE NOT NULL,
  9 period_id NUMBER NOT NULL,
 10 payment method id NUMBER NOT NULL,
 11 dish id NUMBER NOT NULL,
 12 dish amount INT NOT NULL,
 13 total cost DECIMAL (11,2) NOT NULL,
 14 delivery CHAR(1) NOT NULL CHECK (delivery IN ('N', 'Y')))
 15 PARTITION BY RANGE (date id)
 16
     --subpartition by hash(client_id) subpartitions 4
 17 (
     PARTITION quarter_1 VALUES LESS THAN(to date('01.04.2021','DD.MM.YYYY')),
 18
     PARTITION quarter_2 VALUES LESS THAN(to date('01.07.2021','DD.MM.YYYY')),
     PARTITION quarter_3 VALUES LESS THAN(to date('01.10.2021','DD.MM.YYYY')),
     PARTITION quarter 4 VALUES LESS THAN(to date('01.01.2022','DD.MM.YYYY'))
 21
 22
Script Output X
📌 🤌 🔡 📕 | Task completed in 0,066 seconds
Table SAL CL.ORDER FACT created.
```

Let's create package and procedure to load data to this table:







Select from this table:

