Тестовое задание

Диалект (не во всех заданиях): PL/SQL Oracle Database

Код запросов в конце документа.

Ссылка на обновленное (сильно) резюме:

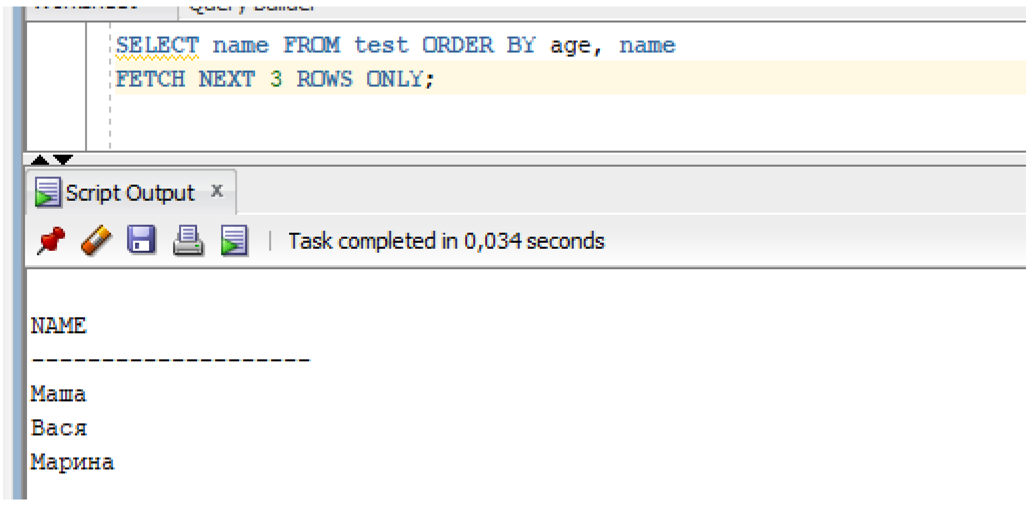
Резюме_Аналитик_Данных_Петров_Алексей.pdf

Задача 1

Задание: необходимо найти 3-х самых молодых сотрудников в коллективе и выдать их имена, предварительно отсортировав. Задачу требуется решить несколькими способами (чем больше, тем лучше).

Простые способы

Способ 1:



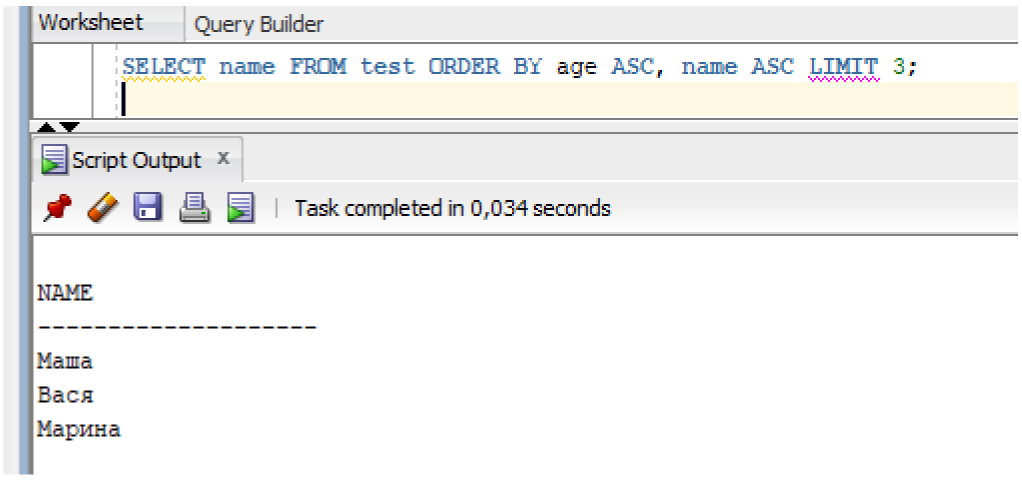
```
SELECT name FROM test ORDER BY age, name
FETCH NEXT 3 ROWS ONLY;
```

Script Output

Task completed in 0,034 seconds

```
NAME
--------------------
Маша
Вася
Марина
```

Способ 2 (в диалекте MY SQL):



Worksheet | Query Builder

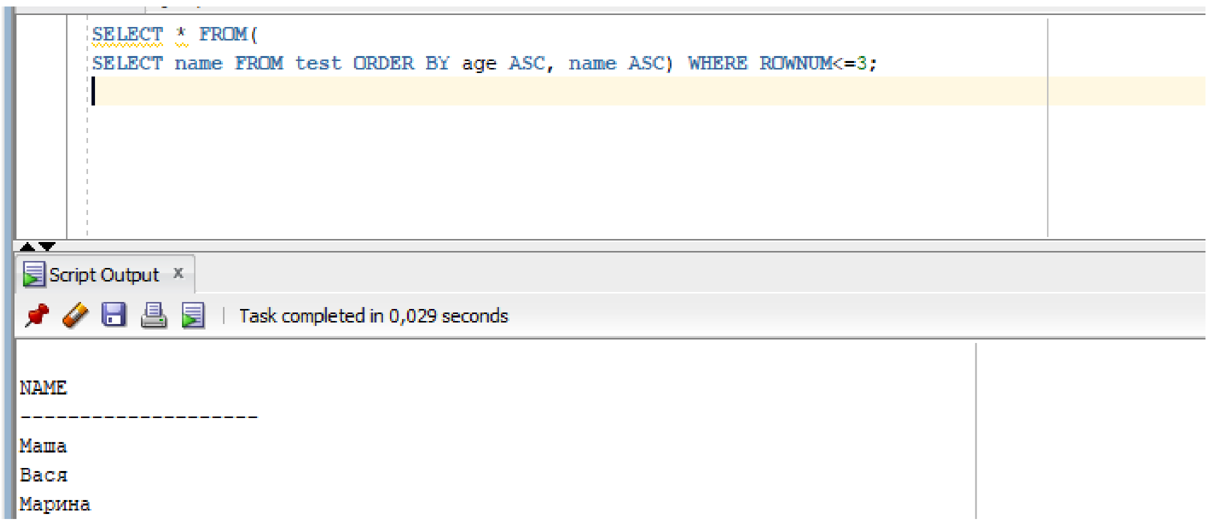
```
SELECT name FROM test ORDER BY age ASC, name ASC LIMIT 3;
```

Script Output

Task completed in 0,034 seconds

```
NAME
--------------------
Маша
Вася
Марина
```

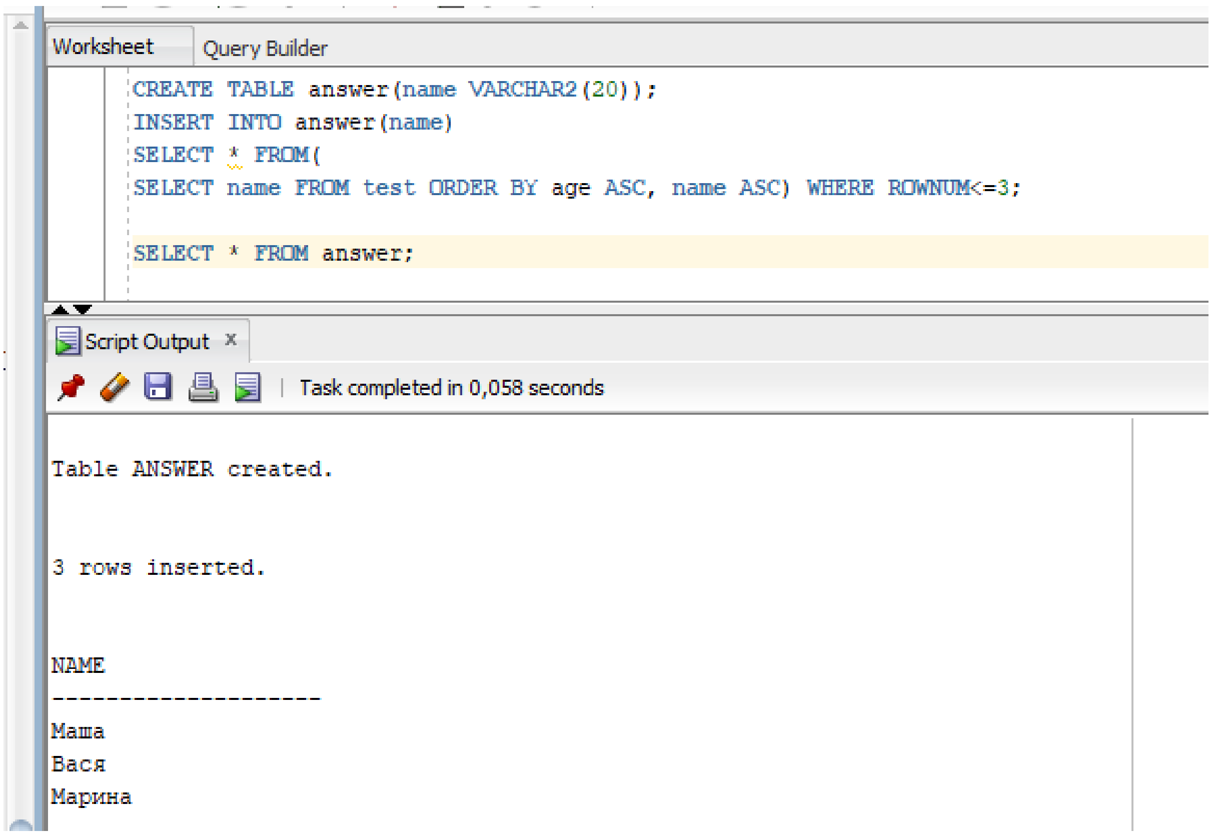
Способ 3:



```
SELECT * FROM(
SELECT name FROM test ORDER BY age ASC, name ASC) WHERE ROWNUM<=3;
```

```
NAME
--------------------
Маша
Вася
Марина
```

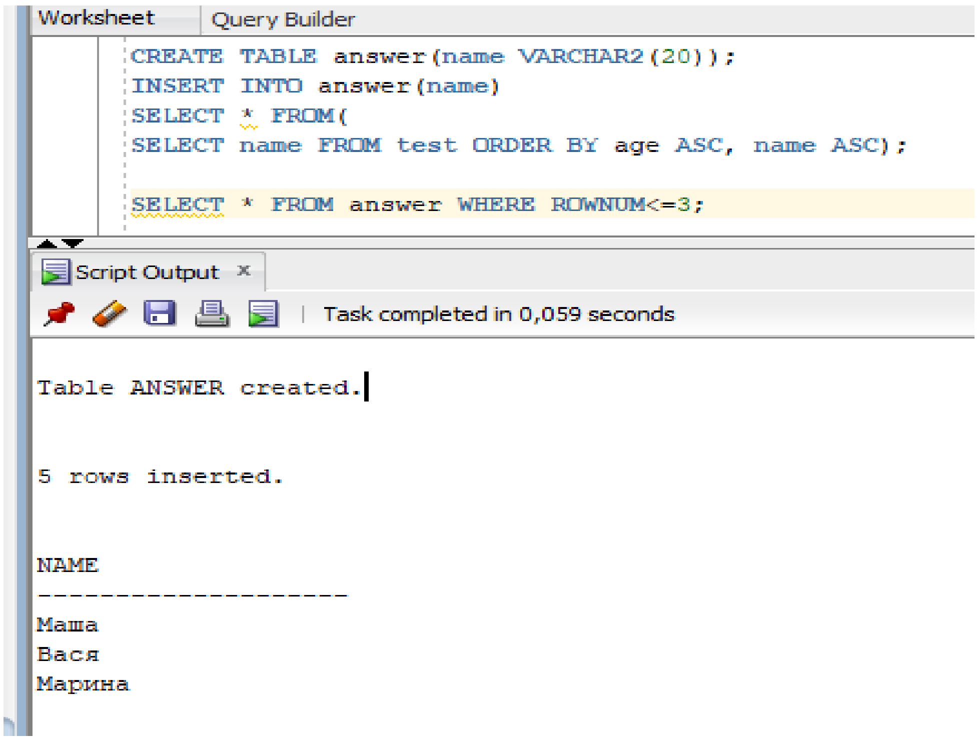
Способ 4:



```
CREATE TABLE answer(name VARCHAR2(20));
INSERT INTO answer(name)
SELECT * FROM(
SELECT name FROM test ORDER BY age ASC, name ASC) WHERE ROWNUM<=3;

SELECT * FROM answer;
```

```
Table ANSWER created.


3 rows inserted.


NAME
--------------------
Маша
Вася
Марина
```

Способ 5:

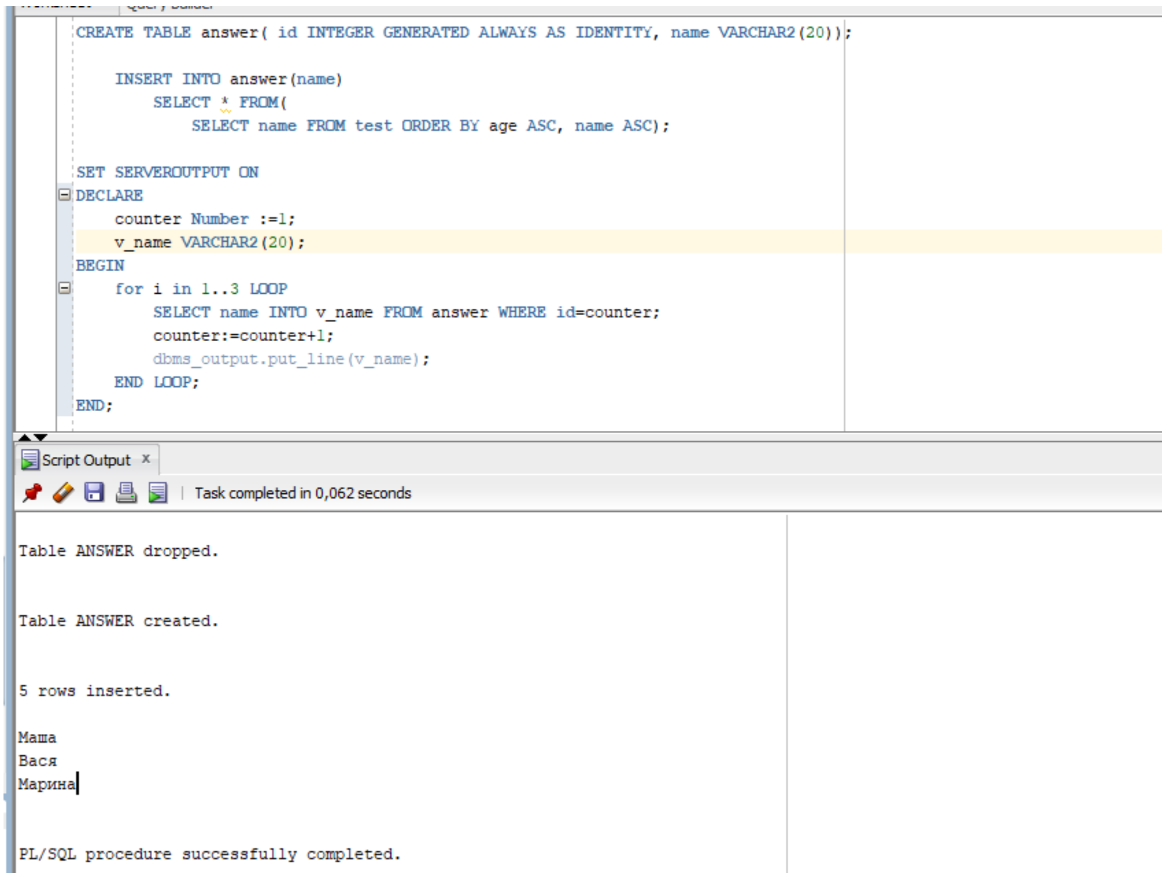


```
CREATE TABLE answer(name VARCHAR2(20));
INSERT INTO answer(name)
SELECT * FROM(
SELECT name FROM test ORDER BY age ASC, name ASC);

SELECT * FROM answer WHERE ROWNUM<=3;
```

```
Table ANSWER created.


5 rows inserted.


NAME
--------------------
Маша
Вася
Марина
```

Способы диалекта PL/SQL

Способ 6:

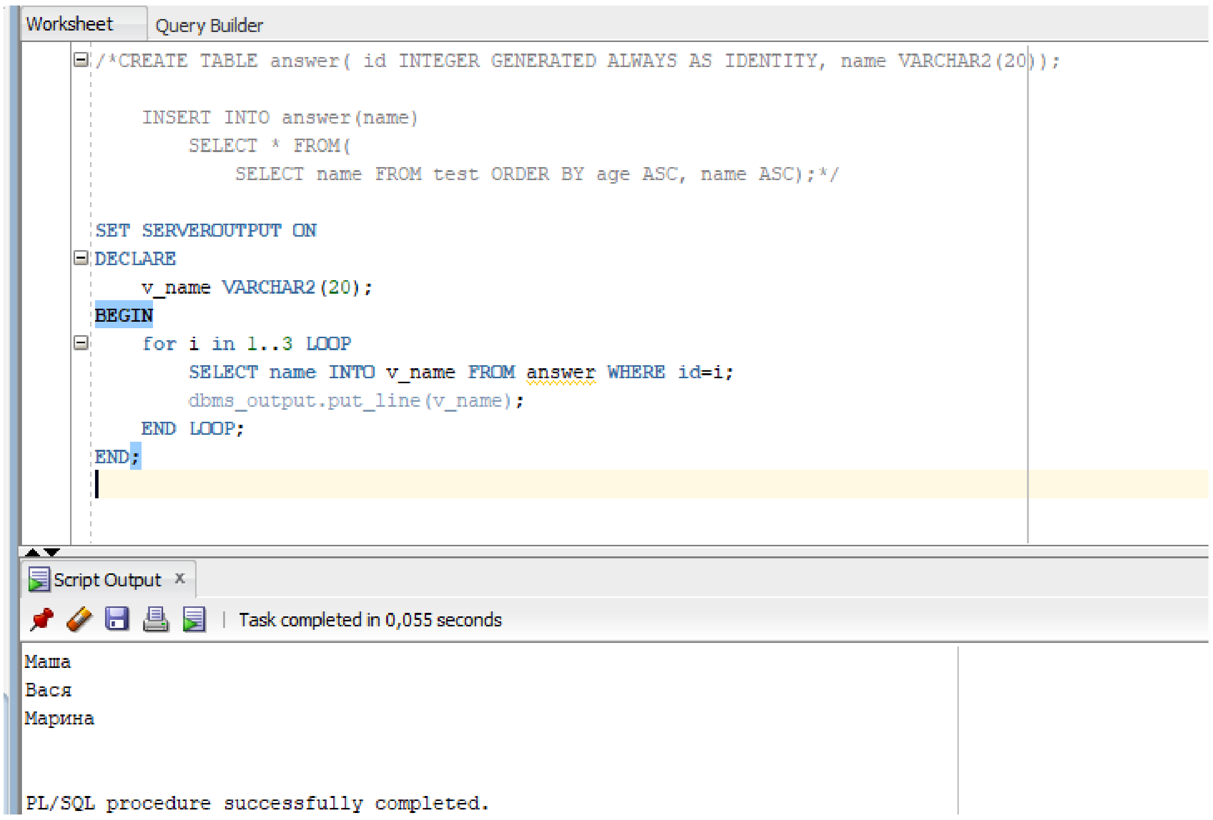


```
CREATE TABLE answer( id INTEGER GENERATED ALWAYS AS IDENTITY, name VARCHAR2(20));

    INSERT INTO answer(name)
        SELECT * FROM(
            SELECT name FROM test ORDER BY age ASC, name ASC);

SET SERVEROUTPUT ON
DECLARE
    counter Number :=1;
    v_name VARCHAR2(20);
BEGIN
    for i in 1..3 LOOP
        SELECT name INTO v_name FROM answer WHERE id=counter;
        counter:=counter+1;
        dbms_output.put_line(v_name);
    END LOOP;
END;
```

```
Table ANSWER dropped.


Table ANSWER created.


5 rows inserted.

Маша
Вася
Марина


PL/SQL procedure successfully completed.
```

Через циклы можно придумать много аналогичных способов

Способ 7 (то же самое, но с i вместо counter):

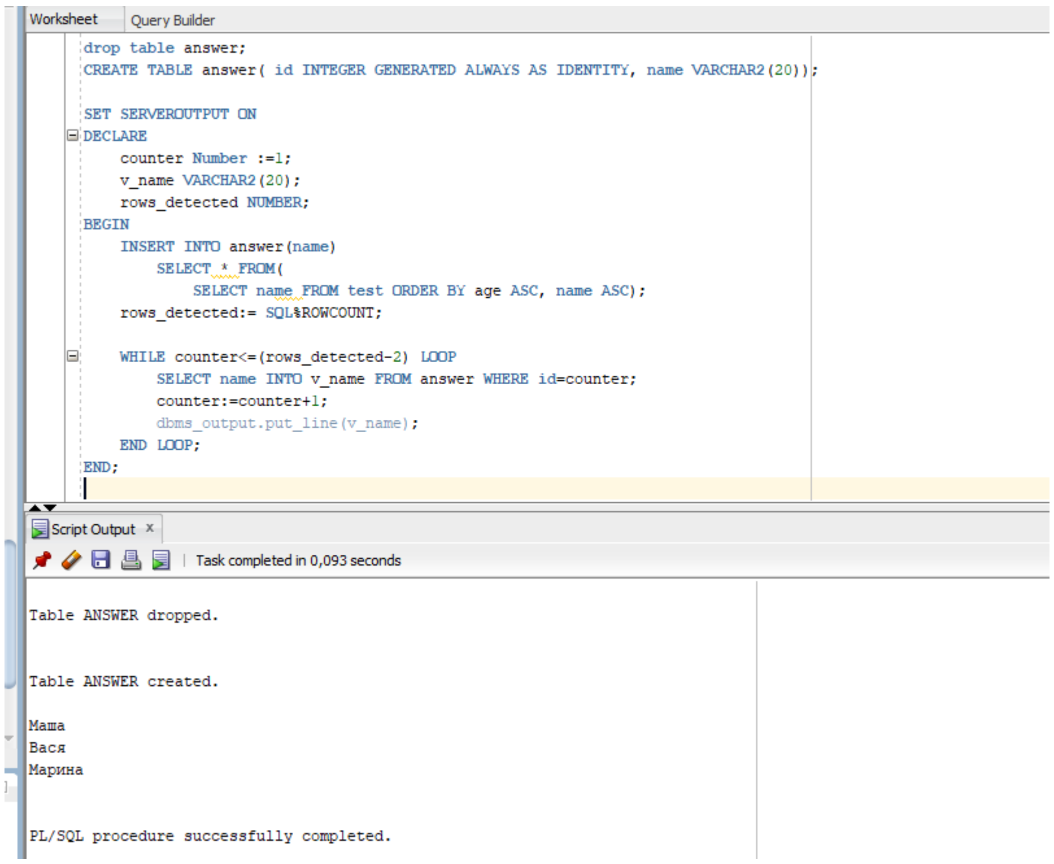

Worksheet Query Builder

```
/*CREATE TABLE answer( id INTEGER GENERATED ALWAYS AS IDENTITY, name VARCHAR2(20));

    INSERT INTO answer(name)
        SELECT * FROM(
            SELECT name FROM test ORDER BY age ASC, name ASC);*/

SET SERVEROUTPUT ON
DECLARE
    v_name VARCHAR2(20);
BEGIN
    for i in 1..3 LOOP
        SELECT name INTO v_name FROM answer WHERE id=i;
        dbms_output.put_line(v_name);
    END LOOP;
END;
```

Script Output | Task completed in 0,055 seconds

```
Маша
Вася
Марина


PL/SQL procedure successfully completed.
```

Способ 8 (использование неявного курсора):

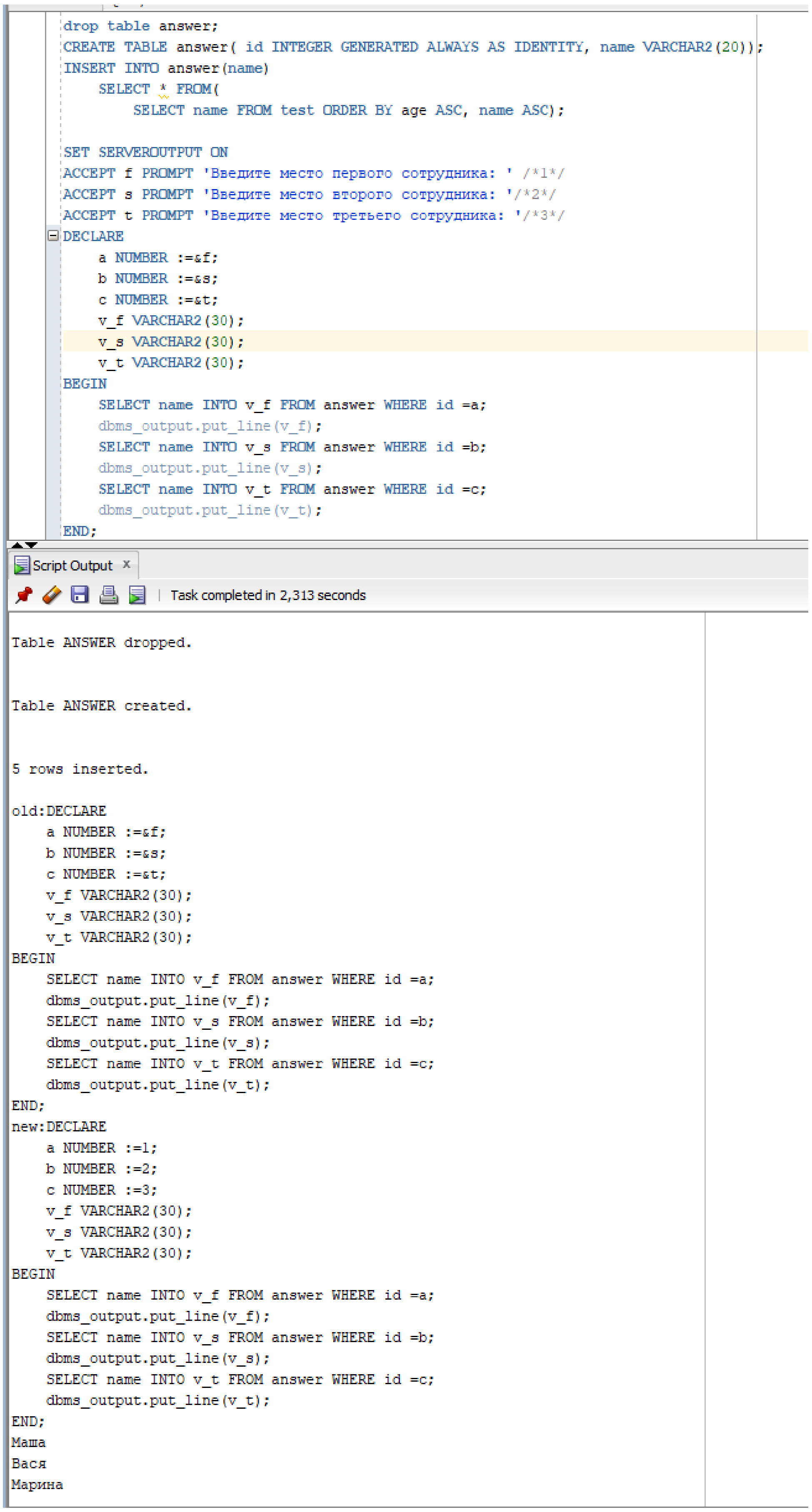

Worksheet Query Builder

```
drop table answer;
CREATE TABLE answer( id INTEGER GENERATED ALWAYS AS IDENTITY, name VARCHAR2(20));

SET SERVEROUTPUT ON
DECLARE
    counter Number :=1;
    v_name VARCHAR2(20);
    rows_detected NUMBER;
BEGIN
    INSERT INTO answer(name)
        SELECT * FROM(
            SELECT name FROM test ORDER BY age ASC, name ASC);
    rows_detected:= SQL%ROWCOUNT;

    WHILE counter<=(rows_detected-2) LOOP
        SELECT name INTO v_name FROM answer WHERE id=counter;
        counter:=counter+1;
        dbms_output.put_line(v_name);
    END LOOP;
END;
```

Script Output | Task completed in 0,093 seconds

```
Table ANSWER dropped.


Table ANSWER created.

Маша
Вася
Марина


PL/SQL procedure successfully completed.
```

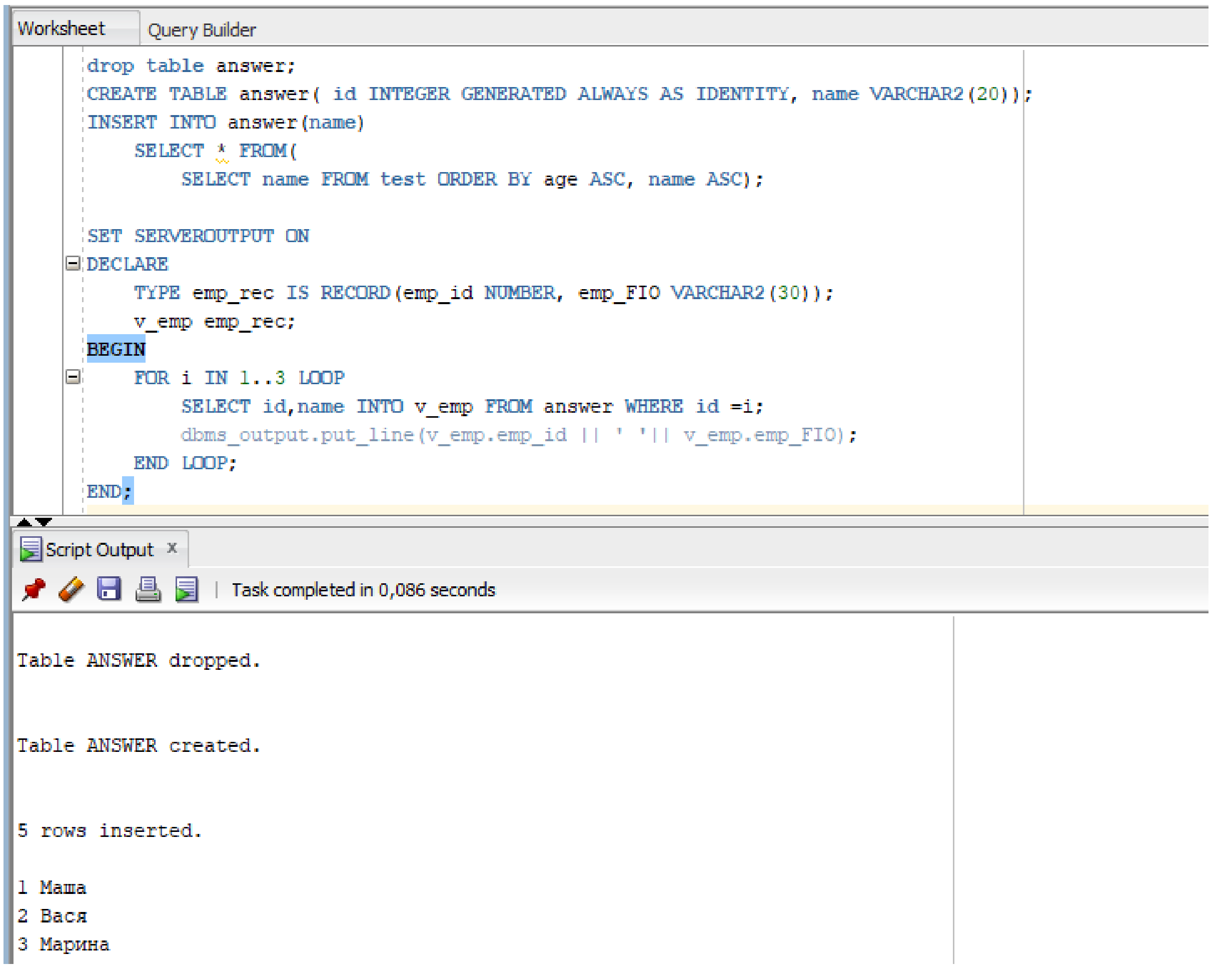
Способ 9 (ввод пользователем места в топе)



```
drop table answer;
CREATE TABLE answer( id INTEGER GENERATED ALWAYS AS IDENTITY, name VARCHAR2(20));
INSERT INTO answer(name)
    SELECT * FROM(
        SELECT name FROM test ORDER BY age ASC, name ASC);

SET SERVEROUTPUT ON
ACCEPT f PROMPT 'Введите место первого сотрудника: ' /*1*/
ACCEPT s PROMPT 'Введите место второго сотрудника: '/*2*/
ACCEPT t PROMPT 'Введите место третьего сотрудника: '/*3*/
DECLARE
    a NUMBER :=&f;
    b NUMBER :=&s;
    c NUMBER :=&t;
    v_f VARCHAR2(30);
    v_s VARCHAR2(30);
    v_t VARCHAR2(30);
BEGIN
    SELECT name INTO v_f FROM answer WHERE id =a;
    dbms_output.put_line(v_f);
    SELECT name INTO v_s FROM answer WHERE id =b;
    dbms_output.put_line(v_s);
    SELECT name INTO v_t FROM answer WHERE id =c;
    dbms_output.put_line(v_t);
END;
```

```
Table ANSWER dropped.


Table ANSWER created.


5 rows inserted.

old:DECLARE
    a NUMBER :=&f;
    b NUMBER :=&s;
    c NUMBER :=&t;
    v_f VARCHAR2(30);
    v_s VARCHAR2(30);
    v_t VARCHAR2(30);
BEGIN
    SELECT name INTO v_f FROM answer WHERE id =a;
    dbms_output.put_line(v_f);
    SELECT name INTO v_s FROM answer WHERE id =b;
    dbms_output.put_line(v_s);
    SELECT name INTO v_t FROM answer WHERE id =c;
    dbms_output.put_line(v_t);
END;
new:DECLARE
    a NUMBER :=1;
    b NUMBER :=2;
    c NUMBER :=3;
    v_f VARCHAR2(30);
    v_s VARCHAR2(30);
    v_t VARCHAR2(30);
BEGIN
    SELECT name INTO v_f FROM answer WHERE id =a;
    dbms_output.put_line(v_f);
    SELECT name INTO v_s FROM answer WHERE id =b;
    dbms_output.put_line(v_s);
    SELECT name INTO v_t FROM answer WHERE id =c;
    dbms_output.put_line(v_t);
END;
Маша
Вася
Марина
```

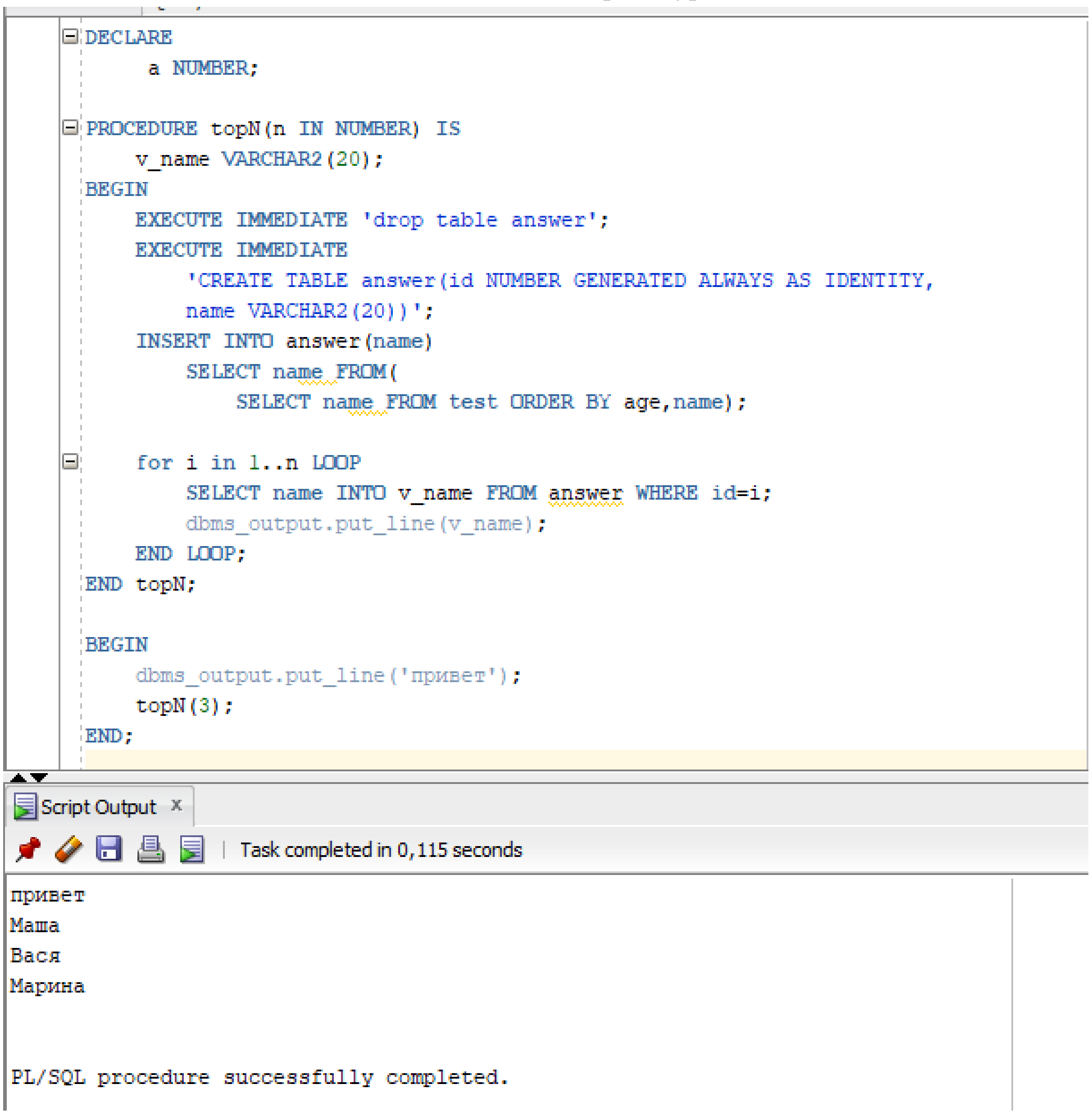
Способ 10 (конструкции):



```
drop table answer;
CREATE TABLE answer( id INTEGER GENERATED ALWAYS AS IDENTITY, name VARCHAR2(20));
INSERT INTO answer(name)
    SELECT * FROM(
        SELECT name FROM test ORDER BY age ASC, name ASC);

SET SERVEROUTPUT ON
DECLARE
    TYPE emp_rec IS RECORD(emp_id NUMBER, emp_FIO VARCHAR2(30));
    v_emp emp_rec;
BEGIN
    FOR i IN 1..3 LOOP
        SELECT id,name INTO v_emp FROM answer WHERE id =i;
        dbms_output.put_line(v_emp.emp_id || ' '|| v_emp.emp_FIO);
    END LOOP;
END;
```

```

Table ANSWER dropped.


Table ANSWER created.


5 rows inserted.

1 Маша
2 Вася
3 Марина
```

Способ 11: Процедура

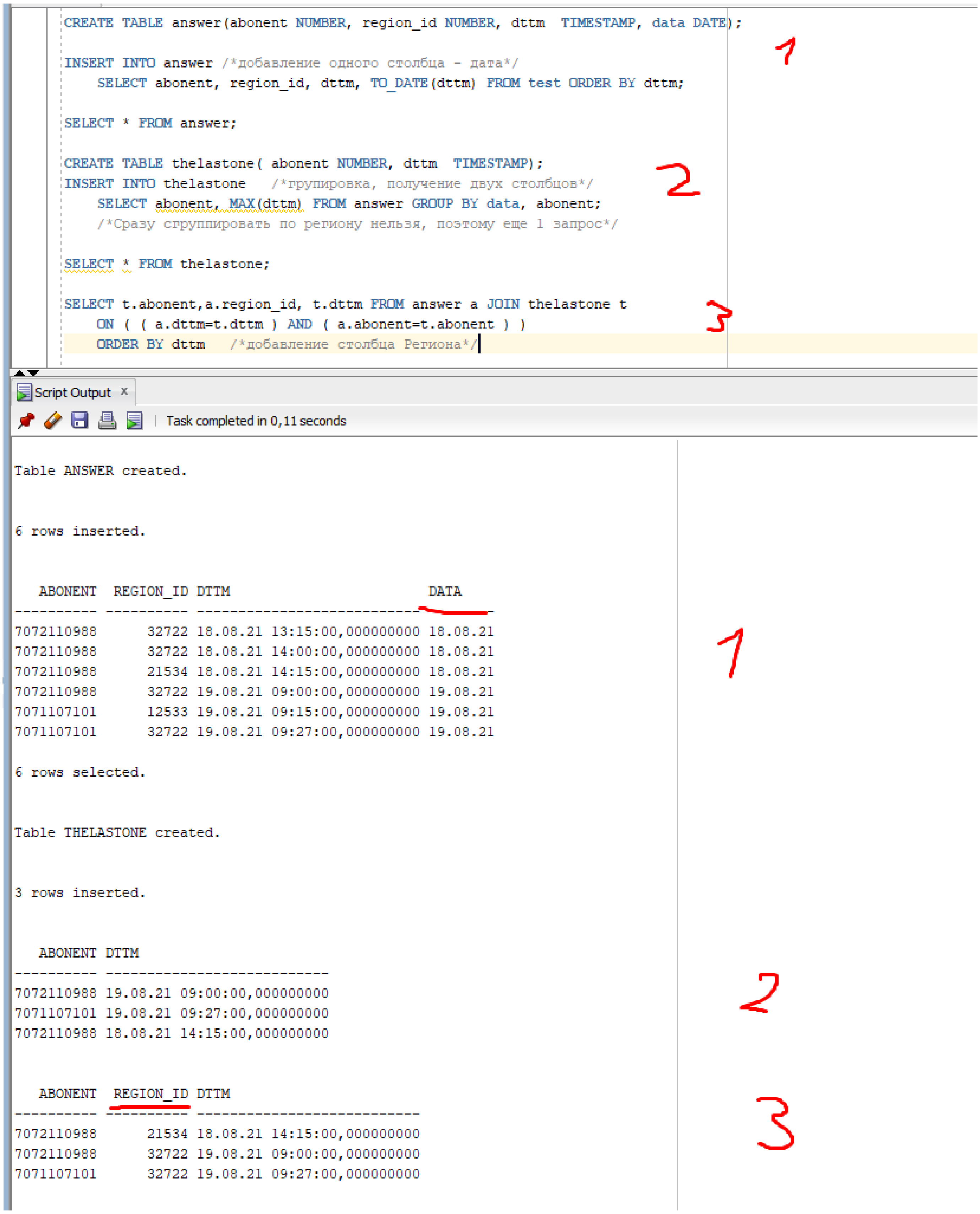


```
DECLARE
      a NUMBER;

PROCEDURE topN(n IN NUMBER) IS
     v_name VARCHAR2(20);
BEGIN
     EXECUTE IMMEDIATE 'drop table answer';
     EXECUTE IMMEDIATE
         'CREATE TABLE answer(id NUMBER GENERATED ALWAYS AS IDENTITY,
         name VARCHAR2(20))';
     INSERT INTO answer(name)
         SELECT name FROM(
             SELECT name FROM test ORDER BY age,name);

     for i in 1..n LOOP
         SELECT name INTO v_name FROM answer WHERE id=i;
         dbms_output.put_line(v_name);
     END LOOP;
END topN;

BEGIN
     dbms_output.put_line('привет');
     topN(3);
END;
```

```
привет
Маша
Вася
Марина


PL/SQL procedure successfully completed.
```

Задача 2

Задание: нужно для каждого дня определить последнее местоположение абонента.



```
CREATE TABLE answer(abonent NUMBER, region_id NUMBER, dttm  TIMESTAMP, data DATE);

INSERT INTO answer /*добавление одного столбца - дата*/
    SELECT abonent, region_id, dttm, TO_DATE(dttm) FROM test ORDER BY dttm;

SELECT * FROM answer;
```

1

```
CREATE TABLE thelastone( abonent NUMBER, dttm  TIMESTAMP);
INSERT INTO thelastone   /*групаировка, получение двух столбцов*/
    SELECT abonent, MAX(dttm) FROM answer GROUP BY data, abonent;
    /*Сразу сгруппировать по региону нельзя, поэтому еще 1 запрос*/

SELECT * FROM thelastone;
```

2

```
SELECT t.abonent,a.region_id, t.dttm FROM answer a JOIN thelastone t
    ON ( ( a.dttm=t.dttm ) AND ( a.abonent=t.abonent ) )
    ORDER BY dttm   /*добавление столбца Региона*/
```

3

Script Output ×

Task completed in 0,11 seconds

```
Table ANSWER created.


6 rows inserted.


   ABONENT  REGION_ID DTTM                        DATA
---------- ---------- --------------------------- --------
7072110988      32722 18.08.21 13:15:00,000000000 18.08.21
7072110988      32722 18.08.21 14:00:00,000000000 18.08.21
7072110988      21534 18.08.21 14:15:00,000000000 18.08.21
7072110988      32722 19.08.21 09:00:00,000000000 19.08.21
7071107101      12533 19.08.21 09:15:00,000000000 19.08.21
7071107101      32722 19.08.21 09:27:00,000000000 19.08.21

6 rows selected.
```

1

```
Table THELASTONE created.


3 rows inserted.


   ABONENT DTTM
---------- ---------------------------
7072110988 19.08.21 09:00:00,000000000
7071107101 19.08.21 09:27:00,000000000
7072110988 18.08.21 14:15:00,000000000
```

2

```
   ABONENT  REGION_ID DTTM
---------- ---------- ---------------------------
7072110988      21534 18.08.21 14:15:00,000000000
7072110988      32722 19.08.21 09:00:00,000000000
7071107101      32722 19.08.21 09:27:00,000000000
```

3

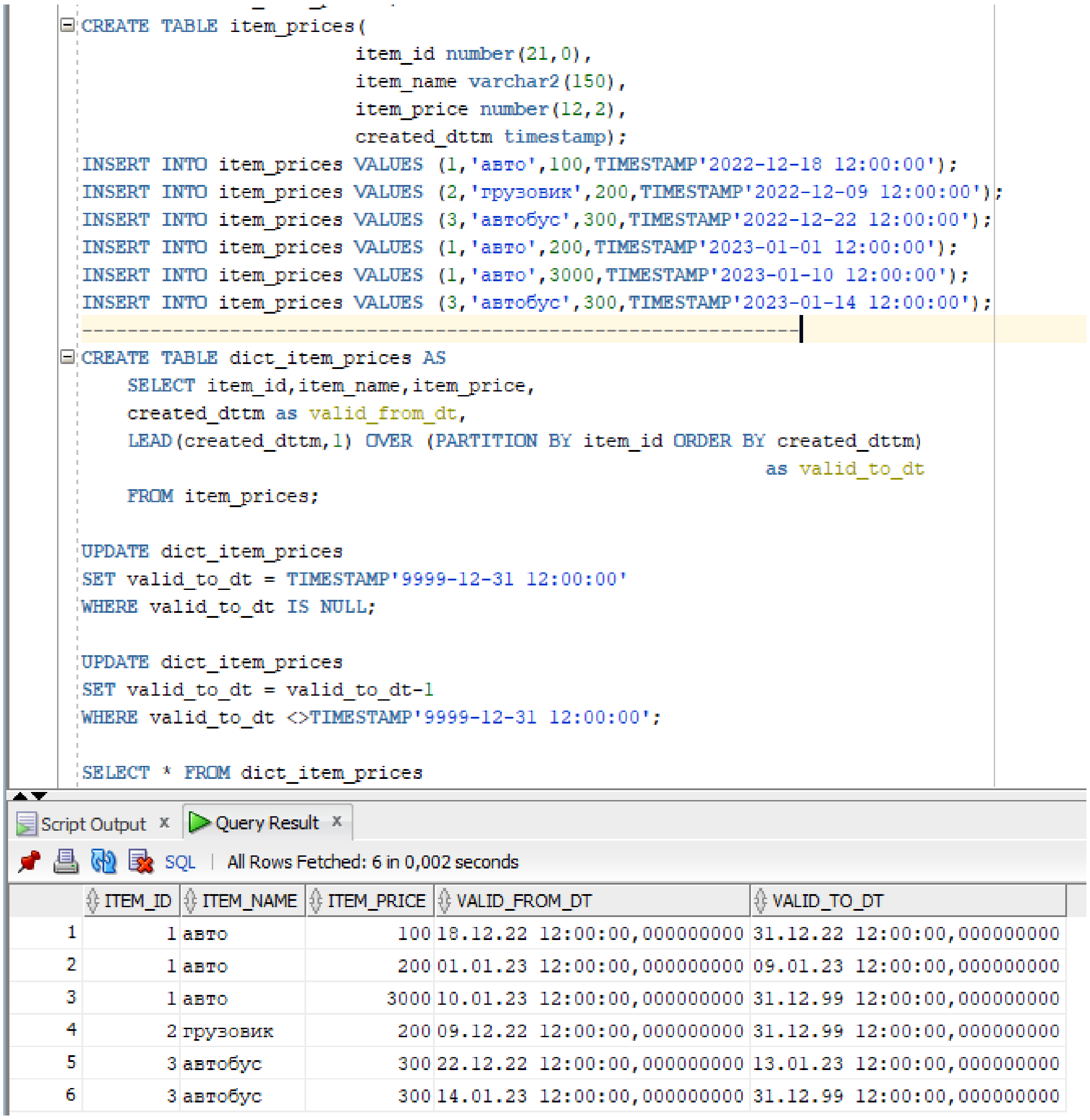
Задача 3

Задание: необходимо сформировать таблицу dict_item_prices.

Примечание: для последней (действующей) цены устанавливается дата 9999-12-31.

P.S. В этой и следующих задачах решение после полосы с дефисами.

Все что выше - данные для наглядности, проверки корректности работы.



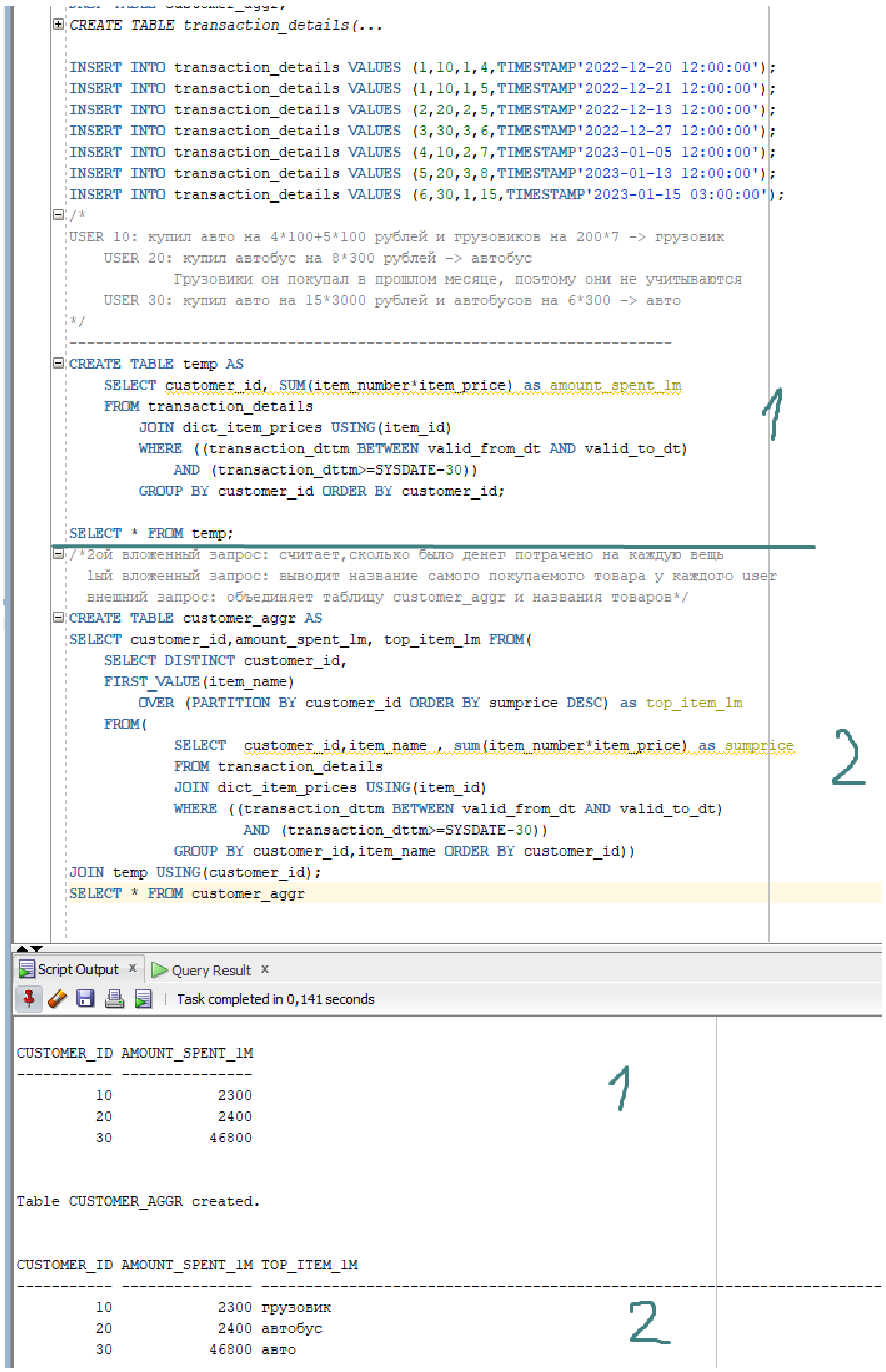
```
CREATE TABLE item_prices(
                        item_id number(21,0),
                        item_name varchar2(150),
                        item_price number(12,2),
                        created_dttm timestamp);
INSERT INTO item_prices VALUES (1,'авто',100,TIMESTAMP'2022-12-18 12:00:00');
INSERT INTO item_prices VALUES (2,'грузовик',200,TIMESTAMP'2022-12-09 12:00:00');
INSERT INTO item_prices VALUES (3,'автобус',300,TIMESTAMP'2022-12-22 12:00:00');
INSERT INTO item_prices VALUES (1,'авто',200,TIMESTAMP'2023-01-01 12:00:00');
INSERT INTO item_prices VALUES (1,'авто',3000,TIMESTAMP'2023-01-10 12:00:00');
INSERT INTO item_prices VALUES (3,'автобус',300,TIMESTAMP'2023-01-14 12:00:00');
---------------------------------------------------------------
CREATE TABLE dict_item_prices AS
    SELECT item_id,item_name,item_price,
    created_dttm as valid_from_dt,
    LEAD(created_dttm,1) OVER (PARTITION BY item_id ORDER BY created_dttm)
                                                        as valid_to_dt
    FROM item_prices;

UPDATE dict_item_prices
SET valid_to_dt = TIMESTAMP'9999-12-31 12:00:00'
WHERE valid_to_dt IS NULL;

UPDATE dict_item_prices
SET valid_to_dt = valid_to_dt-1
WHERE valid_to_dt <>TIMESTAMP'9999-12-31 12:00:00';

SELECT * FROM dict_item_prices
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 6 in 0,002 seconds

	ITEM_ID	ITEM_NAME	ITEM_PRICE	VALID_FROM_DT	VALID_TO_DT
1	1	авто	100	18.12.22 12:00:00,000000000	31.12.22 12:00:00,000000000
2	1	авто	200	01.01.23 12:00:00,000000000	09.01.23 12:00:00,000000000
3	1	авто	3000	10.01.23 12:00:00,000000000	31.12.99 12:00:00,000000000
4	2	грузовик	200	09.12.22 12:00:00,000000000	31.12.99 12:00:00,000000000
5	3	автобус	300	22.12.22 12:00:00,000000000	13.01.23 12:00:00,000000000
6	3	автобус	300	14.01.23 12:00:00,000000000	31.12.99 12:00:00,000000000

Задача 4



```
DROP TABLE customer_aggr;
CREATE TABLE transaction_details(...

INSERT INTO transaction_details VALUES (1,10,1,4,TIMESTAMP'2022-12-20 12:00:00');
INSERT INTO transaction_details VALUES (1,10,1,5,TIMESTAMP'2022-12-21 12:00:00');
INSERT INTO transaction_details VALUES (2,20,2,5,TIMESTAMP'2022-12-13 12:00:00');
INSERT INTO transaction_details VALUES (3,30,3,6,TIMESTAMP'2022-12-27 12:00:00');
INSERT INTO transaction_details VALUES (4,10,2,7,TIMESTAMP'2023-01-05 12:00:00');
INSERT INTO transaction_details VALUES (5,20,3,8,TIMESTAMP'2023-01-13 12:00:00');
INSERT INTO transaction_details VALUES (6,30,1,15,TIMESTAMP'2023-01-15 03:00:00');
/*
USER 10: купил авто на 4*100+5*100 рублей и грузовиков на 200*7 -> грузовик
    USER 20: купил автобус на 8*300 рублей -> автобус
            Грузовики он покупал в прошлом месяце, поэтому они не учитываются
    USER 30: купил авто на 15*3000 рублей и автобусов на 6*300 -> авто
*/
---------------------------------------------------------------------
CREATE TABLE temp AS
    SELECT customer_id, SUM(item_number*item_price) as amount_spent_lm
    FROM transaction_details
        JOIN dict_item_prices USING(item_id)
        WHERE ((transaction_dttm BETWEEN valid_from_dt AND valid_to_dt)
            AND (transaction_dttm>=SYSDATE-30))
        GROUP BY customer_id ORDER BY customer_id;

SELECT * FROM temp;
```

1

```
/*2ой вложенный запрос: считает,сколько было денег потрачено на каждую вещь
  1ый вложенный запрос: выводит название самого покупаемого товара у каждого user
  внешний запрос: объединяет таблицу customer_aggr и названия товаров*/
CREATE TABLE customer_aggr AS
SELECT customer_id,amount_spent_lm, top_item_lm FROM(
    SELECT DISTINCT customer_id,
    FIRST_VALUE(item_name)
        OVER (PARTITION BY customer_id ORDER BY sumprice DESC) as top_item_lm
    FROM(
            SELECT  customer_id,item_name , sum(item_number*item_price) as sumprice
            FROM transaction_details
            JOIN dict_item_prices USING(item_id)
            WHERE ((transaction_dttm BETWEEN valid_from_dt AND valid_to_dt)
                    AND (transaction_dttm>=SYSDATE-30))
            GROUP BY customer_id,item_name ORDER BY customer_id))
JOIN temp USING(customer_id);
SELECT * FROM customer_aggr
```

2

```
CUSTOMER_ID AMOUNT_SPENT_1M
----------- ---------------
         10            2300
         20            2400
         30           46800


Table CUSTOMER_AGGR created.


CUSTOMER_ID AMOUNT_SPENT_1M TOP_ITEM_1M
----------- --------------- --------------------------------------------------------------------
         10            2300 грузовик
         20            2400 автобус
         30           46800 авто
```

1

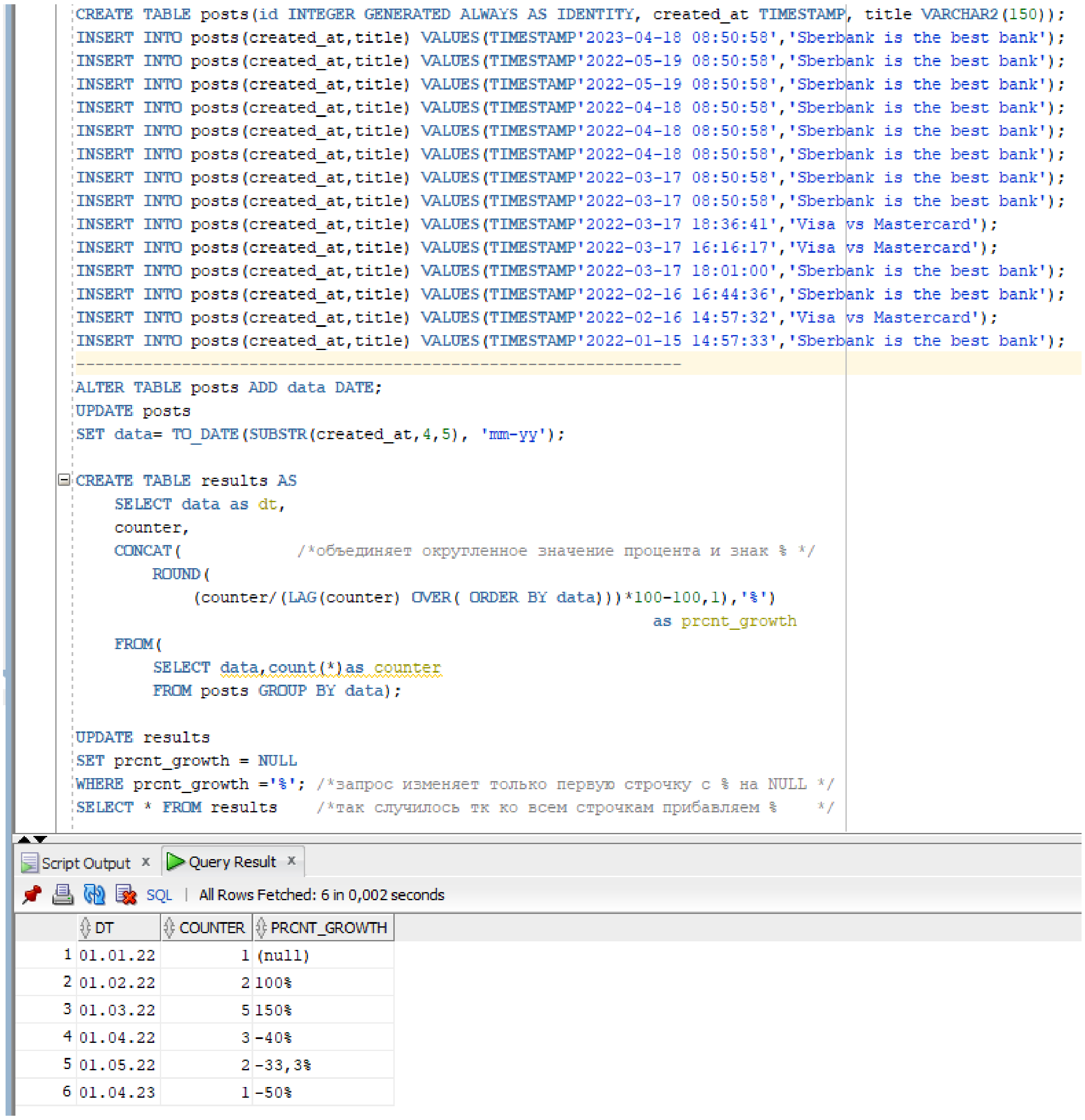
2

Вместо вложенности запросов можно создавать новые (временные) таблицы, работать с ними, и потом удалить ненужные таблицы. Но это увеличит время запроса.

Задача 5

решение - после желтой полосы с дефисами

Все что выше - данные для проверки корректности работы



```sql
CREATE TABLE posts(id INTEGER GENERATED ALWAYS AS IDENTITY, created_at TIMESTAMP, title VARCHAR2(150));
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2023-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-05-19 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-05-19 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 18:36:41','Visa vs Mastercard');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 16:16:17','Visa vs Mastercard');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 18:01:00','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-02-16 16:44:36','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-02-16 14:57:32','Visa vs Mastercard');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-01-15 14:57:33','Sberbank is the best bank');
---------------------------------------------------------------
ALTER TABLE posts ADD data DATE;
UPDATE posts
SET data= TO_DATE(SUBSTR(created_at,4,5), 'mm-yy');

CREATE TABLE results AS
    SELECT data as dt,
    counter,
    CONCAT(             /*объединяет округленное значение процента и знак % */
        ROUND(
            (counter/(LAG(counter) OVER( ORDER BY data)))*100-100,1),'%')
                                                    as prcnt_growth
    FROM(
        SELECT data,count(*)as counter
        FROM posts GROUP BY data);

UPDATE results
SET prcnt_growth = NULL
WHERE prcnt_growth ='%'; /*запрос изменяет только первую строчку с % на NULL */
SELECT * FROM results    /*так случилось тк ко всем строчкам прибавляем %     */
```

Script Output × Query Result ×

SQL | All Rows Fetched: 6 in 0,002 seconds

	DT	COUNTER	PRCNT_GROWTH
1	01.01.22	1	(null)
2	01.02.22	2	100%
3	01.03.22	5	150%
4	01.04.22	3	-40%
5	01.05.22	2	-33,3%
6	01.04.23	1	-50%

ЗАПРОСЫ

ЗАДАЧА 3

```
  CREATE TABLE item_prices(
                 item_id number(21,0),
                 item_name varchar2(150),
                 item_price number(12,2),
                 created_dttm timestamp);
INSERT INTO item_prices VALUES (1,'авто',100,TIMESTAMP'2022-12-18 12:00:00');
INSERT INTO item_prices VALUES (2,'грузовик',200,TIMESTAMP'2022-12-09 12:00:00');
INSERT INTO item_prices VALUES (3,'автобус',300,TIMESTAMP'2022-12-22 12:00:00');
INSERT INTO item_prices VALUES (1,'авто',200,TIMESTAMP'2023-01-01 12:00:00');
INSERT INTO item_prices VALUES (1,'авто',3000,TIMESTAMP'2023-01-10 12:00:00');
INSERT INTO item_prices VALUES (3,'автобус',300,TIMESTAMP'2023-01-14 12:00:00');
------------------------------------------------------------
CREATE TABLE dict_item_prices AS
   SELECT item_id,item_name,item_price,
   created_dttm as valid_from_dt,
   LEAD(created_dttm,1) OVER (PARTITION BY item_id ORDER BY created_dttm)
                                          as valid_to_dt
   FROM item_prices;

UPDATE dict_item_prices
SET valid_to_dt = TIMESTAMP'9999-12-31 12:00:00'
WHERE valid_to_dt IS NULL;

UPDATE dict_item_prices
SET valid_to_dt = valid_to_dt-1
WHERE valid_to_dt <>TIMESTAMP'9999-12-31 12:00:00';

SELECT * FROM dict_item_prices
```

ЗАДАЧА 4

```
CREATE TABLE transaction_details(
   transaction_id number(21,0),
   customer_id number(21,0),
   item_id number(21,0) ,
   item_number number(8,0),
   transaction_dttm timestamp);

INSERT INTO transaction_details VALUES (1,10,1,4,TIMESTAMP'2022-12-20 12:00:00');
INSERT INTO transaction_details VALUES (1,10,1,5,TIMESTAMP'2022-12-21 12:00:00');
INSERT INTO transaction_details VALUES (2,20,2,5,TIMESTAMP'2022-12-13 12:00:00');
INSERT INTO transaction_details VALUES (3,30,3,6,TIMESTAMP'2022-12-27 12:00:00');
INSERT INTO transaction_details VALUES (4,10,2,7,TIMESTAMP'2023-01-05 12:00:00');
INSERT INTO transaction_details VALUES (5,20,3,8,TIMESTAMP'2023-01-13 12:00:00');
INSERT INTO transaction_details VALUES (6,30,1,15,TIMESTAMP'2023-01-15 03:00:00');
/*
USER 10: купил авто на 4*100+5*100 рублей и грузовиков на 200*7 -> грузовик
   USER 20: купил автобус на 8*300 рублей -> автобус
         Грузовики он покупал в прошлом месяце, поэтому они не учитываются
   USER 30: купил авто на 15*3000 рублей и автобусов на 6*300 -> авто
*/
--------------------------------------------------------------------
CREATE TABLE temp AS
   SELECT customer_id, SUM(item_number*item_price) as amount_spent_1m
   FROM transaction_details
      JOIN dict_item_prices USING(item_id)
      WHERE ((transaction_dttm BETWEEN valid_from_dt AND valid_to_dt)
         AND (transaction_dttm>=SYSDATE-30))
      GROUP BY customer_id ORDER BY customer_id;

SELECT * FROM temp;
/*2ой вложенный запрос: считает,сколько было денег потрачено на каждую вещь
  1ый вложенный запрос: выводит название самого покупаемого товара у каждого user
  внешний запрос: объединяет таблицу customer_aggr и названия товаров*/
CREATE TABLE customer_aggr AS
SELECT customer_id,amount_spent_1m, top_item_1m FROM(
   SELECT DISTINCT customer_id,
   FIRST_VALUE(item_name)
      OVER (PARTITION BY customer_id ORDER BY sumprice DESC) as top_item_1m
   FROM(
         SELECT  customer_id,item_name , sum(item_number*item_price) as sumprice
         FROM transaction_details
         JOIN dict_item_prices USING(item_id)
         WHERE ((transaction_dttm BETWEEN valid_from_dt AND valid_to_dt)
               AND (transaction_dttm>=SYSDATE-30))
         GROUP BY customer_id,item_name ORDER BY customer_id))
JOIN temp USING(customer_id);
SELECT * FROM customer_aggr
```

ЗАДАЧА 5

```
CREATE TABLE posts(id INTEGER GENERATED ALWAYS AS IDENTITY, created_at TIMESTAMP, title VARCHAR2(150));
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2023-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-05-19 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-05-19 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-04-18 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 08:50:58','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 18:36:41','Visa vs Mastercard');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 16:16:17','Visa vs Mastercard');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-03-17 18:01:00','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-02-16 16:44:36','Sberbank is the best bank');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-02-16 14:57:32','Visa vs Mastercard');
INSERT INTO posts(created_at,title) VALUES(TIMESTAMP'2022-01-15 14:57:33','Sberbank is the best bank');
-------------------------------------------------------------
ALTER TABLE posts ADD data DATE;
UPDATE posts
SET data=TO_DATE(SUBSTR(created_at,4,5),'mm-yy');

CREATE TABLE results AS
   SELECT data as dt,
   counter,
   CONCAT(        /* объединяет округленное значение процента и знак % */
      ROUND(
         (counter/(LAG(counter) OVER (ORDER BY data)))*100-100,1),'%')
                                          as prcnt_growth
   FROM(
      SELECT data,count(*) as counter
      FROM posts GROUP BY data);

UPDATE results
SET prcnt_growth = NULL
WHERE prcnt_growth = '%';                SELECT * FROM results
```