# Report

Laboratory Work 2

Dmitry Ladutsko

August 09, 2022

## 1. Prerequisites

### 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. Business analyses tasks – Reports

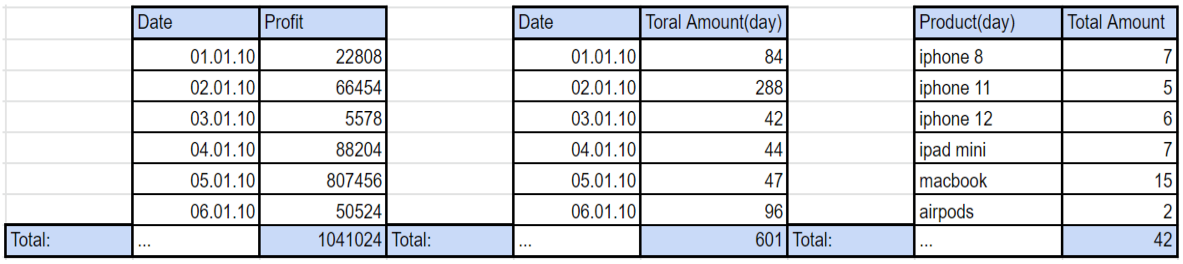
### 2.1. Task 01: CREATE Daily Reports Layouts

**The Main Task** is to create Reports Layouts according your Business Solution Proposal, which was developed on Exit Task Module 6.Introduction to DWH.

**Task Results:**

Create report layouts:

- Create excel report layouts
- Put report layouts on Git – Folder BI Tasks – Product Name (author) - Repots



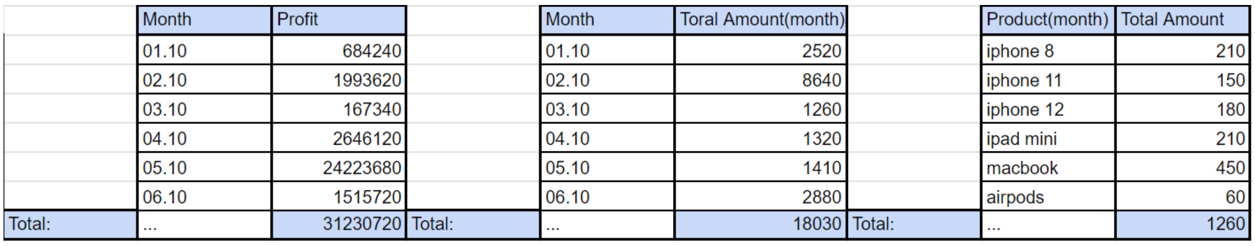
| | Date | Profit | | Date | Toral Amount(day) | | Product(day) | Total Amount |
|---|---|---|---|---|---|---|---|---|
| | 01.01.10 | 22808 | | 01.01.10 | 84 | | iphone 8 | 7 |
| | 02.01.10 | 66454 | | 02.01.10 | 288 | | iphone 11 | 5 |
| | 03.01.10 | 5578 | | 03.01.10 | 42 | | iphone 12 | 6 |
| | 04.01.10 | 88204 | | 04.01.10 | 44 | | ipad mini | 7 |
| | 05.01.10 | 807456 | | 05.01.10 | 47 | | macbook | 15 |
| | 06.01.10 | 50524 | | 06.01.10 | 96 | | airpods | 2 |
| Total: | ... | 1041024 | Total: | ... | 601 | Total: | ... | 42 |

*Picture 1 - Daily Reports Layout*

## 2.2. Task 02: CREATE Monthly Reports Layouts

**The Main Task** is to create Reports Layouts according your Business Solution Proposal, which was developed on Exit Task Module 6.Introduction to DWH.
**Task Results:**
Create report layouts:

- Create excel report layouts
- Put report layouts on Git – Folder BI Tasks – Product Name (author) - Repots



| | Month | Profit | | Month | Toral Amount(month) | | Product(month) | Total Amount |
|---|---|---|---|---|---|---|---|---|
| | 01.10 | 684240 | | 01.10 | 2520 | | iphone 8 | 210 |
| | 02.10 | 1993620 | | 02.10 | 8640 | | iphone 11 | 150 |
| | 03.10 | 167340 | | 03.10 | 1260 | | iphone 12 | 180 |
| | 04.10 | 2646120 | | 04.10 | 1320 | | ipad mini | 210 |
| | 05.10 | 24223680 | | 05.10 | 1410 | | macbook | 450 |
| | 06.10 | 1515720 | | 06.10 | 2880 | | airpods | 60 |
| Total: | ... | 31230720 | Total: | ... | 18030 | Total: | ... | 1260 |

*Picture 2 - Monthly Reports Layout*

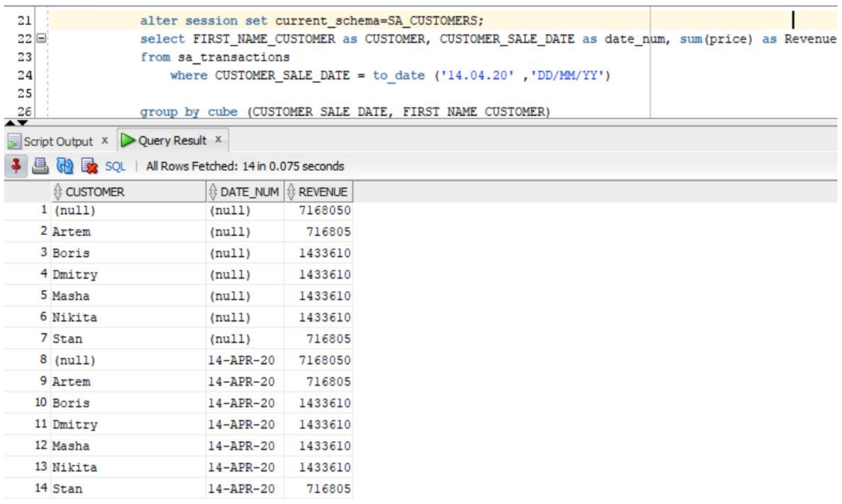
## 3. Advanced Grouping tasks – Reports

### 3.1. Task 03: CREATE Test AdHoc SQL - Daily Reports (CUBE)

**The Main Task** is to create adhoc SQL script, which will calculate Daily Reports (According report layouts on task 01).

**Requirements:**

- USE: CUBE Extension



```
alter session set current_schema=SA_CUSTOMERS;
select FIRST_NAME_CUSTOMER as CUSTOMER, CUSTOMER_SALE_DATE as date_num, sum(price) as Revenue
from sa_transactions
    where CUSTOMER_SALE_DATE = to_date ('14.04.20' ,'DD/MM/YY')

group by cube (CUSTOMER_SALE_DATE, FIRST_NAME_CUSTOMER)
```

Script Output × | Query Result ×

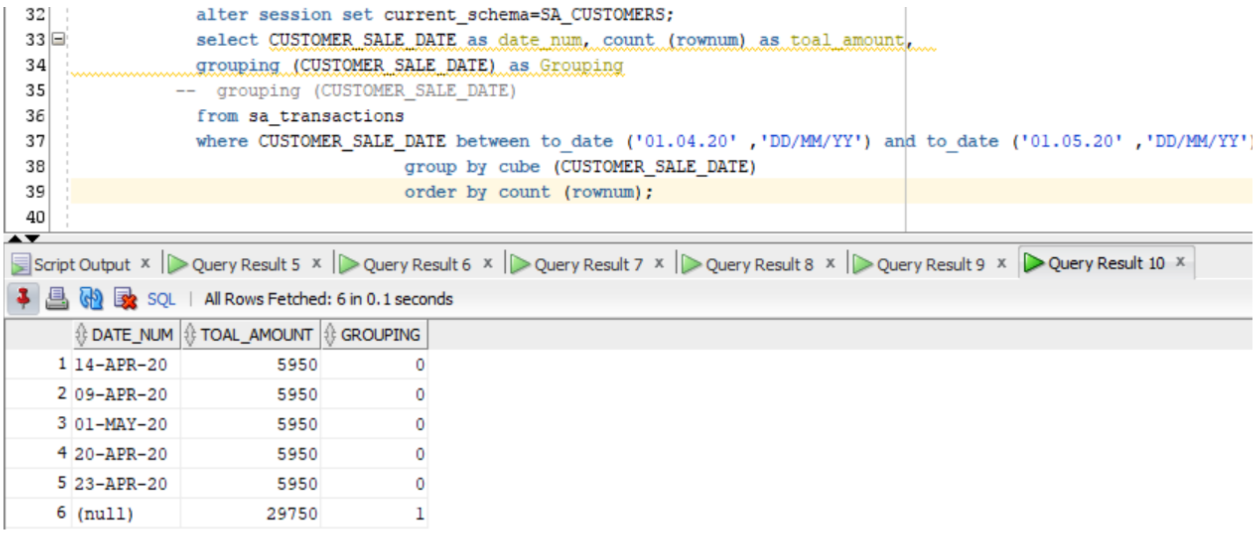
SQL | All Rows Fetched: 14 in 0.075 seconds

```
   CUSTOMER   DATE_NUM   REVENUE
 1 (null)     (null)     7168050
 2 Artem      (null)      716805
 3 Boris      (null)     1433610
 4 Dmitry     (null)     1433610
 5 Masha      (null)     1433610
 6 Nikita     (null)     1433610
 7 Stan       (null)      716805
 8 (null)     14-APR-20  7168050
 9 Artem      14-APR-20   716805
10 Boris      14-APR-20  1433610
11 Dmitry     14-APR-20  1433610
12 Masha      14-APR-20  1433610
13 Nikita     14-APR-20  1433610
14 Stan       14-APR-20   716805
```

*Picture 3 - Group by cube*

**CUBE** extension will generate subtotals for all combinations of the dimensions specified. If "n" is the number of columns listed in the CUBE, there will be 2n subtotal combinations.

- USE: Grouping() function



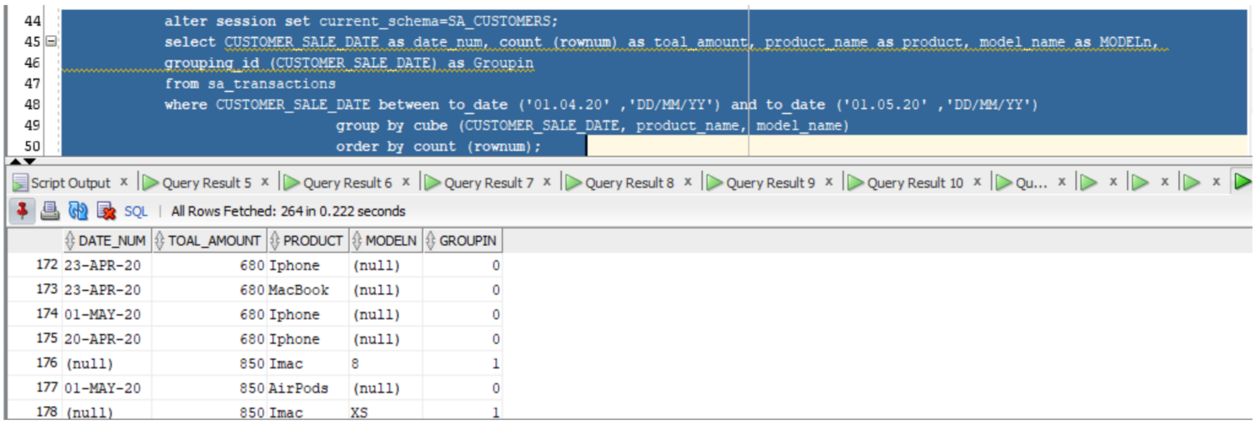
```
            alter session set current_schema=SA_CUSTOMERS;
            select CUSTOMER_SALE_DATE as date_num, count (rownum) as toal_amount,
            grouping (CUSTOMER_SALE_DATE) as Grouping
         --  grouping (CUSTOMER_SALE_DATE)
            from sa_transactions
            where CUSTOMER_SALE_DATE between to_date ('01.04.20' ,'DD/MM/YY') and to_date ('01.05.20' ,'DD/MM/YY'
                                group by cube (CUSTOMER_SALE_DATE)
                                order by count (rownum);
```

```
All Rows Fetched: 6 in 0.1 seconds

   DATE_NUM   TOAL_AMOUNT  GROUPING
 1 14-APR-20         5950         0
 2 09-APR-20         5950         0
 3 01-MAY-20         5950         0
 4 20-APR-20         5950         0
 5 23-APR-20         5950         0
 6 (null)           29750         1
```

*Picture 4 – Grouping*

It can be quite easy to visually identify subtotals generated by rollups and cubes, but to do it programatically you really need something more accurate than the presence of null values in the grouping columns. This is where the **GROUPING** function comes in. It accepts a single column as a parameter and returns "1" if the column contains a null value generated as part of a subtotal by a ROLLUP or CUBE operation or "0" for any other value, including stored null values.

- USE: Grouping_ID function



```
            alter session set current_schema=SA_CUSTOMERS;
            select CUSTOMER_SALE_DATE as date_num, count (rownum) as toal_amount, product_name as product, model_name as MODELn,
            grouping_id (CUSTOMER_SALE_DATE) as Groupin
            from sa_transactions
            where CUSTOMER_SALE_DATE between to_date ('01.04.20' ,'DD/MM/YY') and to_date ('01.05.20' ,'DD/MM/YY')
                                group by cube (CUSTOMER_SALE_DATE, product_name, model_name)
                                order by count (rownum);
```

```
All Rows Fetched: 264 in 0.222 seconds

     DATE_NUM   TOAL_AMOUNT PRODUCT  MODELN  GROUPIN
 172 23-APR-20          680 Iphone   (null)        0
 173 23-APR-20          680 MacBook  (null)        0
 174 01-MAY-20          680 Iphone   (null)        0
 175 20-APR-20          680 Iphone   (null)        0
 176 (null)             850 Imac     8             1
 177 01-MAY-20          850 AirPods  (null)        0
 178 (null)             850 Imac     XS            1
```

*Picture 5 - Grouping_id*

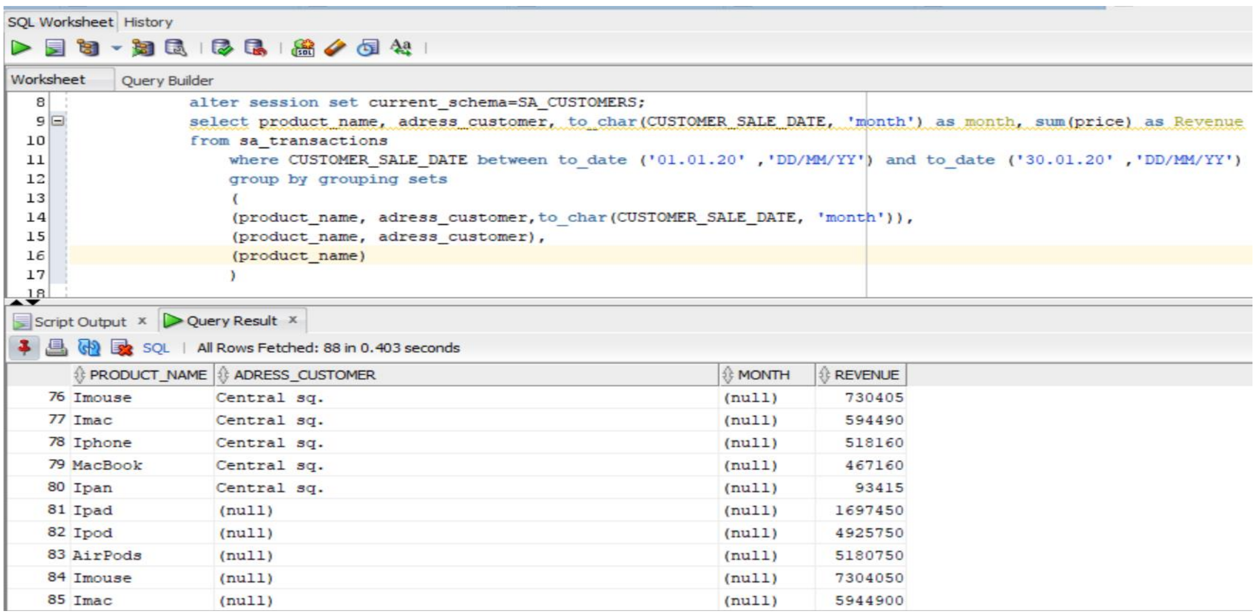
The **GROUPING_ID** function provides an alternate and more compact way to identify subtotal rows. Passing the dimension columns as arguments, it returns a number indicating the GROUP BY level.

## 3.2. Task 04: CREATE Test AdHoc SQL - Monthly Reports (ROLLUP & GROUPING SETS)

**<u>The Main Task</u>** is to create adhoc SQL script, which will calculate Monthly Reports (According report layouts on task 01).

**<u>Requirements:</u>**

- USE: ROLLUP or GROUPING SETS Extension

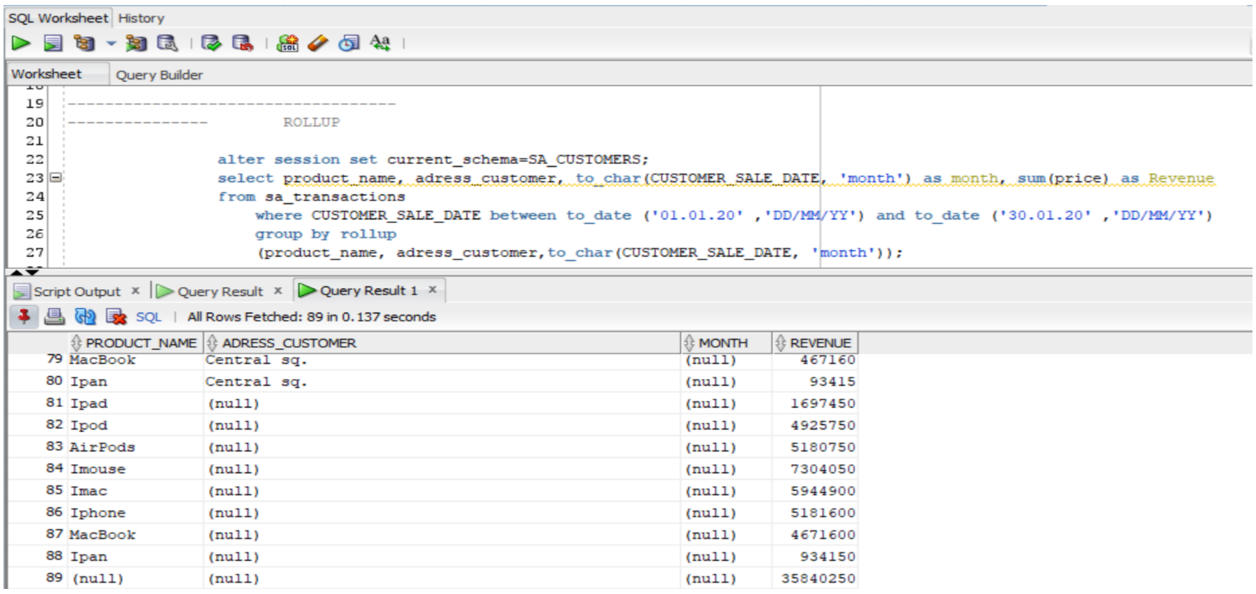


```sql
alter session set current_schema=SA_CUSTOMERS;
select product_name, adress_customer, to_char(CUSTOMER_SALE_DATE, 'month') as month, sum(price) as Revenue
from sa_transactions
    where CUSTOMER_SALE_DATE between to_date ('01.01.20' ,'DD/MM/YY') and to_date ('30.01.20' ,'DD/MM/YY')
    group by grouping sets
    (
    (product_name, adress_customer,to_char(CUSTOMER_SALE_DATE, 'month')),
    (product_name, adress_customer),
    (product_name)
    )
```

```
All Rows Fetched: 88 in 0.403 seconds

   PRODUCT_NAME  ADRESS_CUSTOMER    MONTH    REVENUE
76 Imouse        Central sq.        (null)    730405
77 Imac          Central sq.        (null)    594490
78 Iphone        Central sq.        (null)    518160
79 MacBook       Central sq.        (null)    467160
80 Ipan          Central sq.        (null)     93415
81 Ipad          (null)             (null)   1697450
82 Ipod          (null)             (null)   4925750
83 AirPods       (null)             (null)   5180750
84 Imouse        (null)             (null)   7304050
85 Imac          (null)             (null)   5944900
```

*Picture 6 - Group by grouping sets*

**GROUPING SETS** calculating all possible subtotals in a cube, especially those with many dimensions, can be quite an intensive process. If you don't need all the subtotals, this can represent a considerable amount of wasted effort. The following cube with three dimensions gives 8 levels of subtotals (GROUPING_ID: 0-7)



```sql
-----------------------------------
---------------          ROLLUP

                alter session set current_schema=SA_CUSTOMERS;
                select product_name, adress_customer, to_char(CUSTOMER_SALE_DATE, 'month') as month, sum(price) as Revenue
                from sa_transactions
                    where CUSTOMER_SALE_DATE between to_date ('01.01.20' ,'DD/MM/YY') and to_date ('30.01.20' ,'DD/MM/YY')
                    group by rollup
                    (product_name, adress_customer,to_char(CUSTOMER_SALE_DATE, 'month'));
```

```
All Rows Fetched: 89 in 0.137 seconds

   PRODUCT_NAME  ADRESS_CUSTOMER    MONTH    REVENUE
79 MacBook       Central sq.        (null)    467160
80 Ipan          Central sq.        (null)     93415
81 Ipad          (null)             (null)   1697450
82 Ipod          (null)             (null)   4925750
83 AirPods       (null)             (null)   5180750
84 Imouse        (null)             (null)   7304050
85 Imac          (null)             (null)   5944900
86 Iphone        (null)             (null)   5181600
87 MacBook       (null)             (null)   4671600
88 Ipan          (null)             (null)    934150
89 (null)        (null)             (null)  35840250
```

*Picture 7 - Group by rollup*

*In addition to the regular aggregation* results we expect from the **GROUP BY** clause, the **ROLLUP** extension produces group subtotals from right to left and a grand total. If "n" is the number of columns listed in the **ROLLUP**, there will be n+1 levels of subtotals.

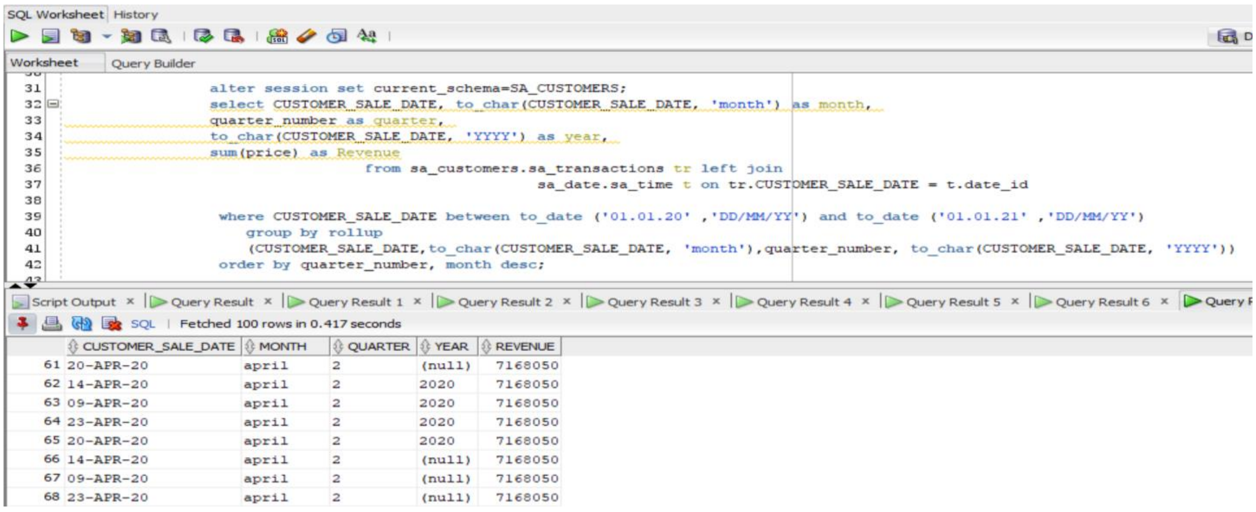
## 3.3. Task 05: CREATE Test AdHoc SQL – ROLLUP by Time

__**The Main Task**__ is to create adhoc SQL script, which will calculate Time Based Reports

Calculate measurements by next levels:

- DAY
- MONTH
- QUARTER
- YEAR

### Sum Revenue



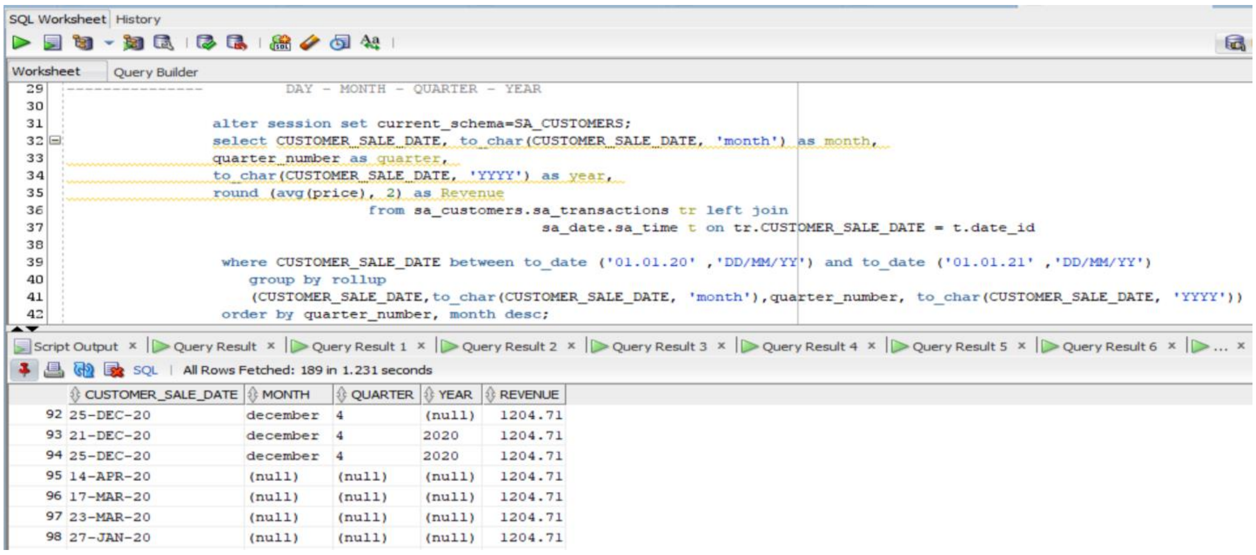
```
                alter session set current_schema=SA_CUSTOMERS;
                select CUSTOMER_SALE_DATE, to_char(CUSTOMER_SALE_DATE, 'month') as month,
                quarter_number as quarter,
                to_char(CUSTOMER_SALE_DATE, 'YYYY') as year,
                sum(price) as Revenue
                             from sa_customers.sa_transactions tr left join
                                             sa_date.sa_time t on tr.CUSTOMER_SALE_DATE = t.date_id

                 where CUSTOMER_SALE_DATE between to_date ('01.01.20' ,'DD/MM/YY') and to_date ('01.01.21' ,'DD/MM/YY')
                    group by rollup
                    (CUSTOMER_SALE_DATE,to_char(CUSTOMER_SALE_DATE, 'month'),quarter_number, to_char(CUSTOMER_SALE_DATE, 'YYYY'))
                 order by quarter_number, month desc;
```

```
Fetched 100 rows in 0.417 seconds
```

| | CUSTOMER_SALE_DATE | MONTH | QUARTER | YEAR | REVENUE |
|---|---|---|---|---|---|
| 61 | 20-APR-20 | april | 2 | (null) | 7168050 |
| 62 | 14-APR-20 | april | 2 | 2020 | 7168050 |
| 63 | 09-APR-20 | april | 2 | 2020 | 7168050 |
| 64 | 23-APR-20 | april | 2 | 2020 | 7168050 |
| 65 | 20-APR-20 | april | 2 | 2020 | 7168050 |
| 66 | 14-APR-20 | april | 2 | (null) | 7168050 |
| 67 | 09-APR-20 | april | 2 | (null) | 7168050 |
| 68 | 23-APR-20 | april | 2 | (null) | 7168050 |

*Picture 8 - Group by rollup day - month - quarter – year*

### Average Revenue



```
----------------            DAY - MONTH - QUARTER - YEAR

                alter session set current_schema=SA_CUSTOMERS;
                select CUSTOMER_SALE_DATE, to_char(CUSTOMER_SALE_DATE, 'month') as month,
                quarter_number as quarter,
                to_char(CUSTOMER_SALE_DATE, 'YYYY') as year,
                round (avg(price), 2) as Revenue
                             from sa_customers.sa_transactions tr left join
                                             sa_date.sa_time t on tr.CUSTOMER_SALE_DATE = t.date_id

                 where CUSTOMER_SALE_DATE between to_date ('01.01.20' ,'DD/MM/YY') and to_date ('01.01.21' ,'DD/MM/YY')
                    group by rollup
                    (CUSTOMER_SALE_DATE,to_char(CUSTOMER_SALE_DATE, 'month'),quarter_number, to_char(CUSTOMER_SALE_DATE, 'YYYY'))
                 order by quarter_number, month desc;
```

```
All Rows Fetched: 189 in 1.231 seconds
```

| | CUSTOMER_SALE_DATE | MONTH | QUARTER | YEAR | REVENUE |
|---|---|---|---|---|---|
| 92 | 25-DEC-20 | december | 4 | (null) | 1204.71 |
| 93 | 21-DEC-20 | december | 4 | 2020 | 1204.71 |
| 94 | 25-DEC-20 | december | 4 | 2020 | 1204.71 |
| 95 | 14-APR-20 | (null) | (null) | (null) | 1204.71 |
| 96 | 17-MAR-20 | (null) | (null) | (null) | 1204.71 |
| 97 | 23-MAR-20 | (null) | (null) | (null) | 1204.71 |
| 98 | 27-JAN-20 | (null) | (null) | (null) | 1204.71 |

*Picture 9 - Group by rollup day - month - quarter – year*

**Laboratory work summary:**

**At this lab** we have learned how we can use different group statements to group data in useful Reports. We used following group statements:

- Group by
- Group by cube
- Grouping
- Grouping_id
- Group by rollup
- Group by grouping sets

**Now we have** much more understanding about grouping data in decision – making reports.