Report

Laboratory Work 7

Dmitry Ladutsko

August 15, 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`

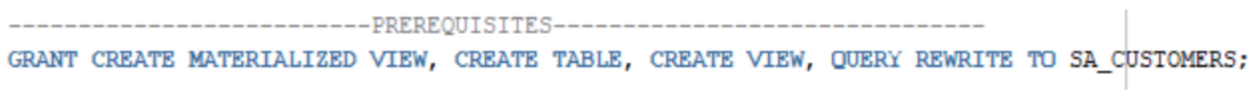
2. Materialized Views- Basic

2.1. Task 01: Create Materialized Views - ON DEMAND

<u>**The Main Task**</u> is to create Materialized Views, which will refresh ON DEMAND. You should use SQL script that was prepared by you on LabWork 02 – Report Layout Monthly.

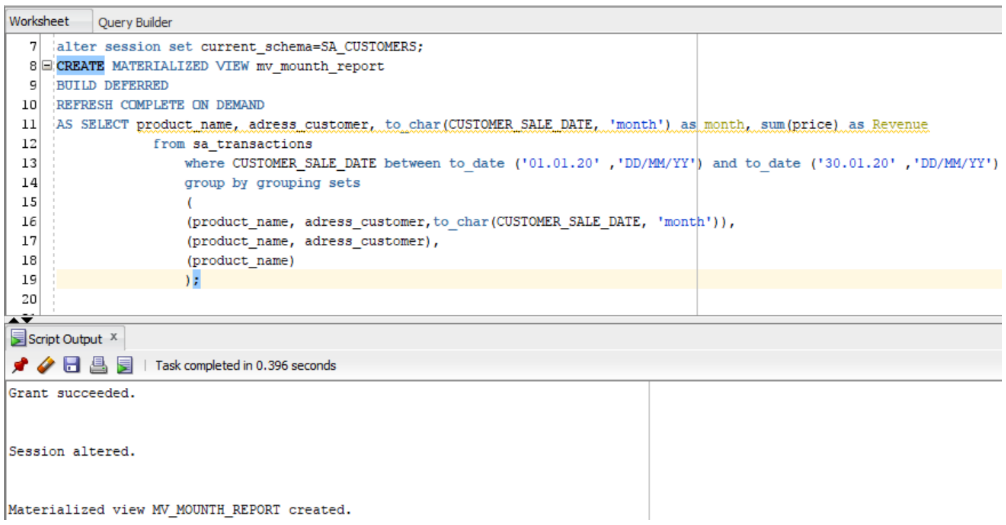
<u>Required points:</u>

- Use Standard CREATE MATERIALIZED VIEW CLAUSE
- Use DBMS_MVIEW package to run refresh MView.



```
-------------------------PREREQUISITES------------------------------
GRANT CREATE MATERIALIZED VIEW, CREATE TABLE, CREATE VIEW, QUERY REWRITE TO SA_CUSTOMERS;
```

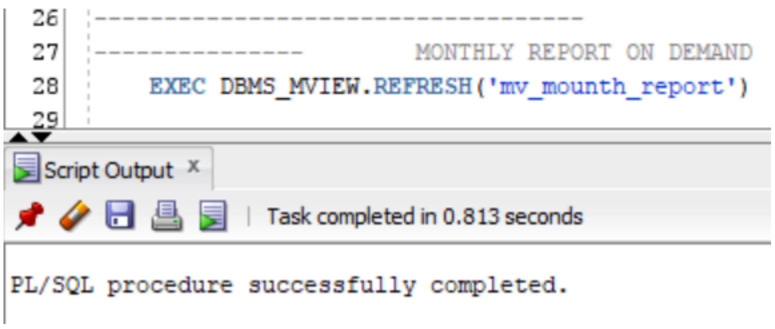
Note. Before creating Materialized Views with SA_* user, we firstly need to grant it privileges. Now let' create Mat. View as monthly report from Laboratory Work 2:



```
alter session set current_schema=SA_CUSTOMERS;
CREATE MATERIALIZED VIEW mv_mounth_report
BUILD DEFERRED
REFRESH COMPLETE ON DEMAND
AS 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)
                );
```

```
Task completed in 0.396 seconds

Grant succeeded.


Session altered.


Materialized view MV_MOUNTH_REPORT created.
```

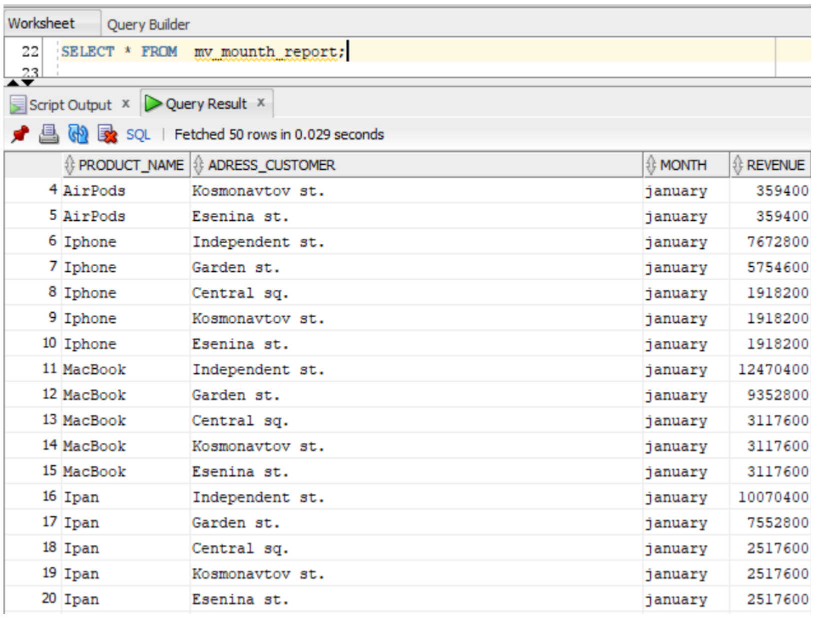
And create **Procedure** to refresh mat. View using **DBMS_MVIEW.REFRESH**



```
-------------------------------------
---------------          MONTHLY REPORT ON DEMAND
    EXEC DBMS_MVIEW.REFRESH('mv_mounth_report')
```

```
Task completed in 0.813 seconds

PL/SQL procedure successfully completed.
```

Results:



```
SELECT * FROM mv_mounth_report;
```

Fetched 50 rows in 0.029 seconds

	PRODUCT_NAME	ADRESS_CUSTOMER	MONTH	REVENUE
4	AirPods	Kosmonavtov st.	january	359400
5	AirPods	Esenina st.	january	359400
6	Iphone	Independent st.	january	7672800
7	Iphone	Garden st.	january	5754600
8	Iphone	Central sq.	january	1918200
9	Iphone	Kosmonavtov st.	january	1918200
10	Iphone	Esenina st.	january	1918200
11	MacBook	Independent st.	january	12470400
12	MacBook	Garden st.	january	9352800
13	MacBook	Central sq.	january	3117600
14	MacBook	Kosmonavtov st.	january	3117600
15	MacBook	Esenina st.	january	3117600
16	Ipan	Independent st.	january	10070400
17	Ipan	Garden st.	january	7552800
18	Ipan	Central sq.	january	2517600
19	Ipan	Kosmonavtov st.	january	2517600
20	Ipan	Esenina st.	january	2517600

<u>Task Results:</u>

Create required objects:

- Put objects script to Git.
- Prepare Document with Screenshot of Tests Data

Prepare Document with Screenshot of Refreshing MView script

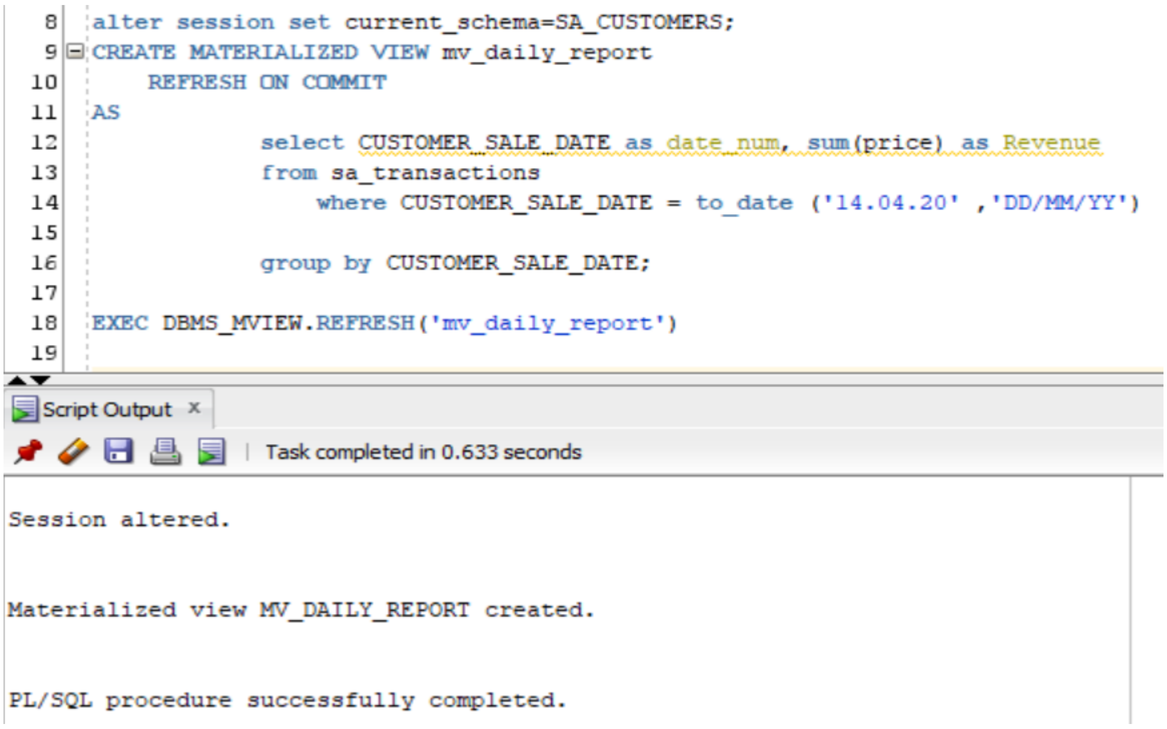
2.2. Task 02: Create Materialized Views - ON COMMIT

<u>The Main Task</u> is to create Materialized Views, which will refresh ON COMMIT. You should use SQL script that was prepared by you on LabWork 02 – Report Layout DAILY.

<u>Required points:</u>

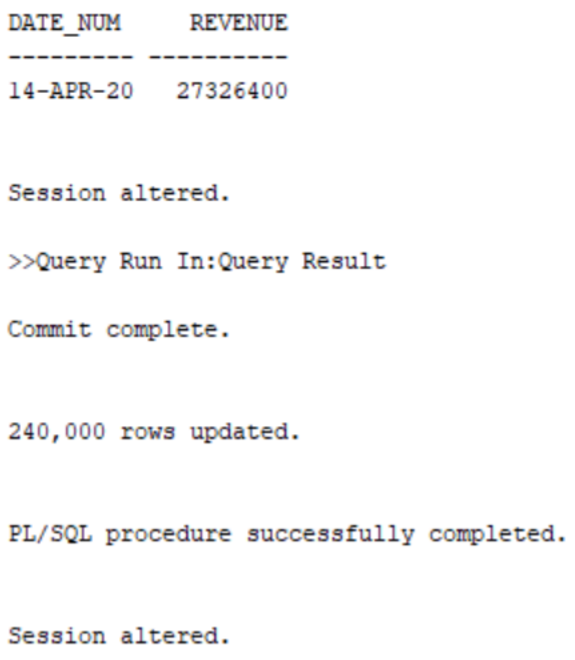
- Use Standard CREATE MATERIALIZED VIEW CLAUSE
- Tests ON COMMIT REFRESH.

Note. Let's create another Materialized View Refresh on commit using daily report from Laboratory Work 2 (to see the revenue for specified day).

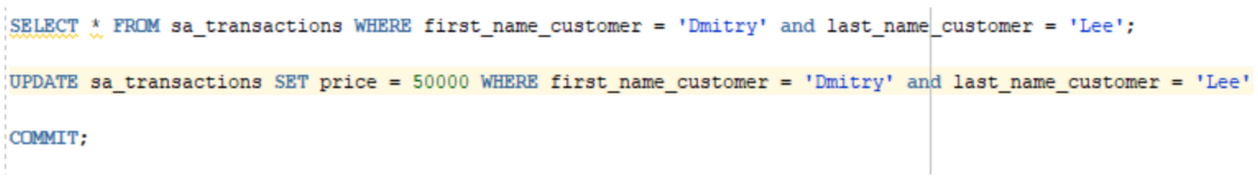


```
alter session set current_schema=SA_CUSTOMERS;
CREATE MATERIALIZED VIEW mv_daily_report
    REFRESH ON COMMIT
AS
            select CUSTOMER_SALE_DATE as date_num, sum(price) as Revenue
            from sa_transactions
                where CUSTOMER_SALE_DATE = to_date ('14.04.20' ,'DD/MM/YY')

            group by CUSTOMER_SALE_DATE;

EXEC DBMS_MVIEW.REFRESH('mv_daily_report')

```

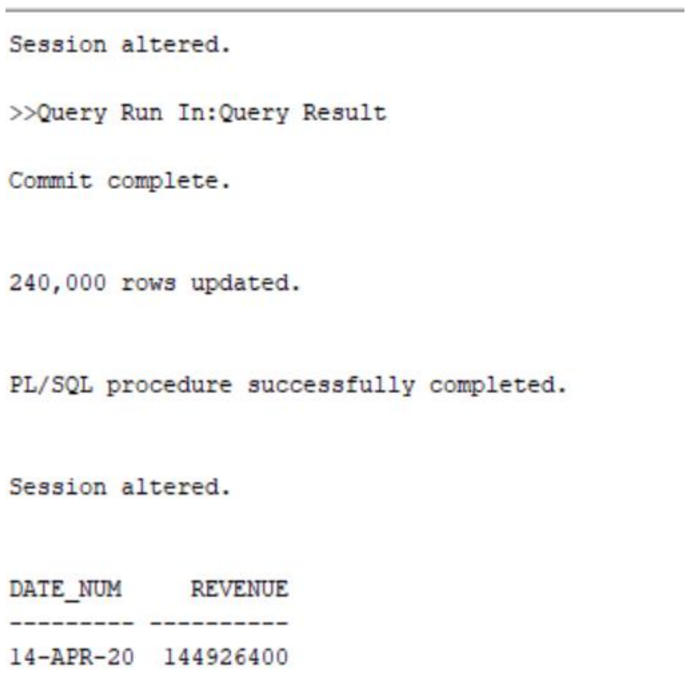
```
Session altered.


Materialized view MV_DAILY_REPORT created.


PL/SQL procedure successfully completed.
```



```
DATE_NUM      REVENUE
---------- ----------
14-APR-20    27326400


Session altered.

>>Query Run In:Query Result

Commit complete.


240,000 rows updated.


PL/SQL procedure successfully completed.


Session altered.
```



```
SELECT * FROM sa_transactions WHERE first_name_customer = 'Dmitry' and last_name_customer = 'Lee';

UPDATE sa_transactions SET price = 50000 WHERE first_name_customer = 'Dmitry' and last_name_customer = 'Lee'

COMMIT;
```



```
Session altered.

>>Query Run In:Query Result

Commit complete.


240,000 rows updated.


PL/SQL procedure successfully completed.


Session altered.


DATE_NUM      REVENUE
---------- ----------
14-APR-20   144926400
```

Note. We can see that before we commit transaction nothing have changed in Mat. View.

<u>Task Results:</u>

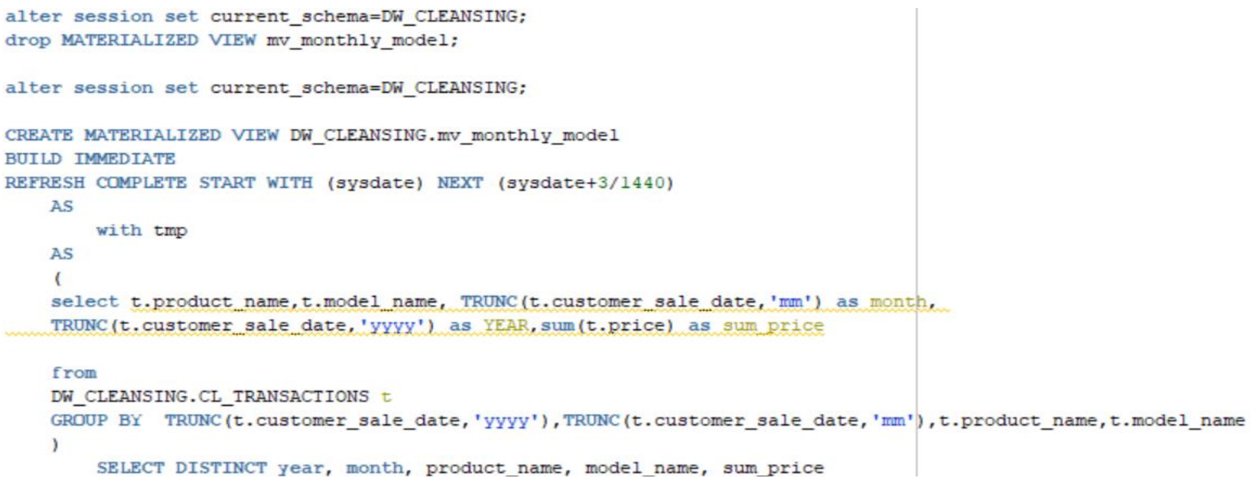
Create required objects:

- Put objects script to Git.
- Prepare Document with Screenshot of Tests Data
- Prepare Document with Screenshot of Refreshing MView script

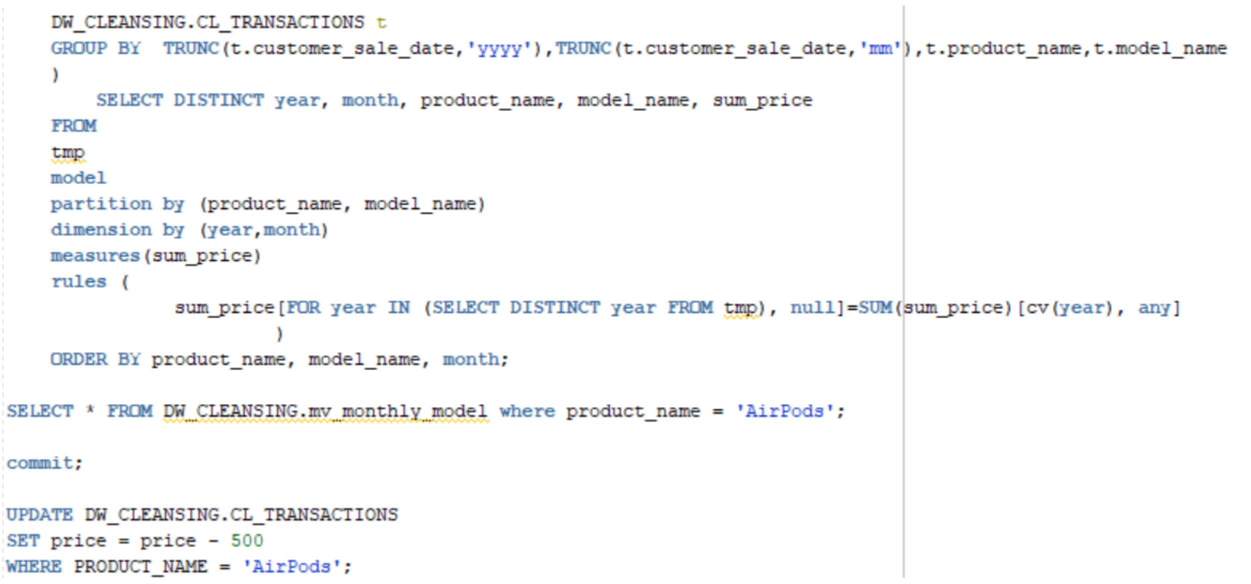
3. Materialized Views- Model Clause

3.1. Task 03: Create Materialized Views - Refreshing at definitive Time moment

<u>The Main Task</u> is to create Materialized Views, which will refresh definitive Time moment. You should use Model SQL script that was prepared by you on LabWork 05 – Report Layout Monthly.

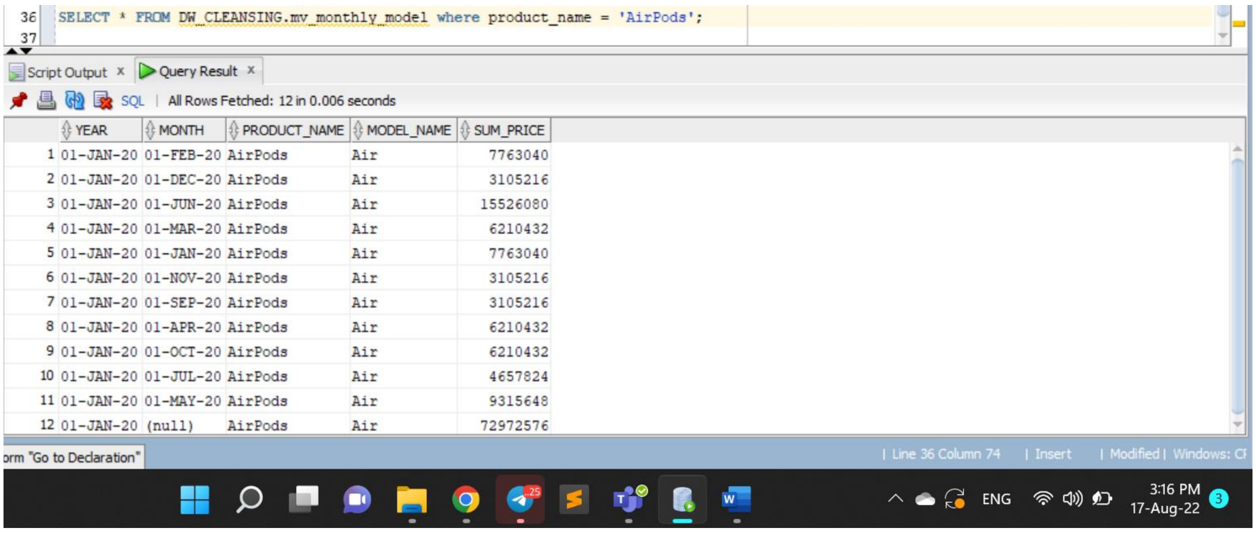


```
alter session set current_schema=DW_CLEANSING;
drop MATERIALIZED VIEW mv_monthly_model;

alter session set current_schema=DW_CLEANSING;

CREATE MATERIALIZED VIEW DW_CLEANSING.mv_monthly_model
BUILD IMMEDIATE
REFRESH COMPLETE START WITH (sysdate) NEXT (sysdate+3/1440)
    AS
        with tmp
    AS
    (
    select t.product_name,t.model_name, TRUNC(t.customer_sale_date,'mm') as month,
    TRUNC(t.customer_sale_date,'yyyy') as YEAR,sum(t.price) as sum_price

    from
    DW_CLEANSING.CL_TRANSACTIONS t
    GROUP BY  TRUNC(t.customer_sale_date,'yyyy'),TRUNC(t.customer_sale_date,'mm'),t.product_name,t.model_name
    )
        SELECT DISTINCT year, month, product_name, model_name, sum_price
```



```
    DW_CLEANSING.CL_TRANSACTIONS t
    GROUP BY  TRUNC(t.customer_sale_date,'yyyy'),TRUNC(t.customer_sale_date,'mm'),t.product_name,t.model_name
    )
        SELECT DISTINCT year, month, product_name, model_name, sum_price
    FROM
    tmp
    model
    partition by (product_name, model_name)
    dimension by (year,month)
    measures(sum_price)
    rules (
            sum_price[FOR year IN (SELECT DISTINCT year FROM tmp), null]=SUM(sum_price)[cv(year), any]
                    )
    ORDER BY product_name, model_name, month;

SELECT * FROM DW_CLEANSING.mv_monthly_model where product_name = 'AirPods';

commit;

UPDATE DW_CLEANSING.CL_TRANSACTIONS
SET price = price - 500
WHERE PRODUCT_NAME = 'AirPods';
```

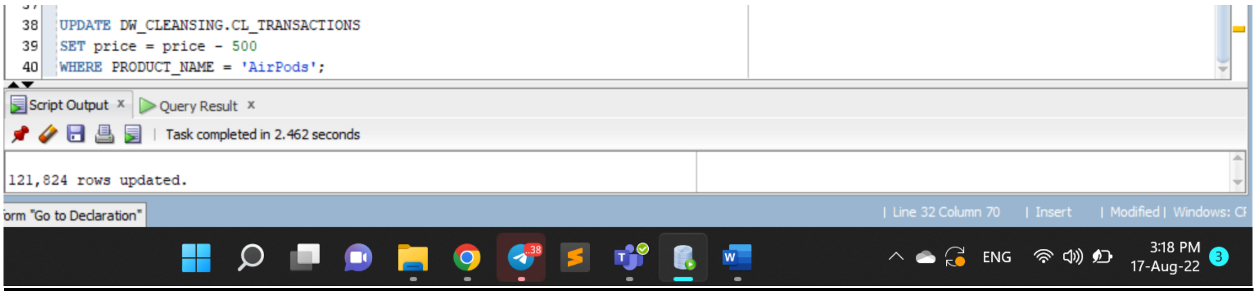
Note. As you can see we created **Materialized View refresh at definitive Time moment (in 3 minutes)**



```
SELECT * FROM DW_CLEANSING.mv_monthly_model where product_name = 'AirPods';
```

	YEAR	MONTH	PRODUCT_NAME	MODEL_NAME	SUM_PRICE
1	01-JAN-20	01-FEB-20	AirPods	Air	7763040
2	01-JAN-20	01-DEC-20	AirPods	Air	3105216
3	01-JAN-20	01-JUN-20	AirPods	Air	15526080
4	01-JAN-20	01-MAR-20	AirPods	Air	6210432
5	01-JAN-20	01-JAN-20	AirPods	Air	7763040
6	01-JAN-20	01-NOV-20	AirPods	Air	3105216
7	01-JAN-20	01-SEP-20	AirPods	Air	3105216
8	01-JAN-20	01-APR-20	AirPods	Air	6210432
9	01-JAN-20	01-OCT-20	AirPods	Air	6210432
10	01-JAN-20	01-JUL-20	AirPods	Air	4657824
11	01-JAN-20	01-MAY-20	AirPods	Air	9315648
12	01-JAN-20	(null)	AirPods	Air	72972576

orm "Go to Declaration"
| Line 36 Column 74 | Insert | Modified | Windows: CI
ENG
3:16 PM
17-Aug-22

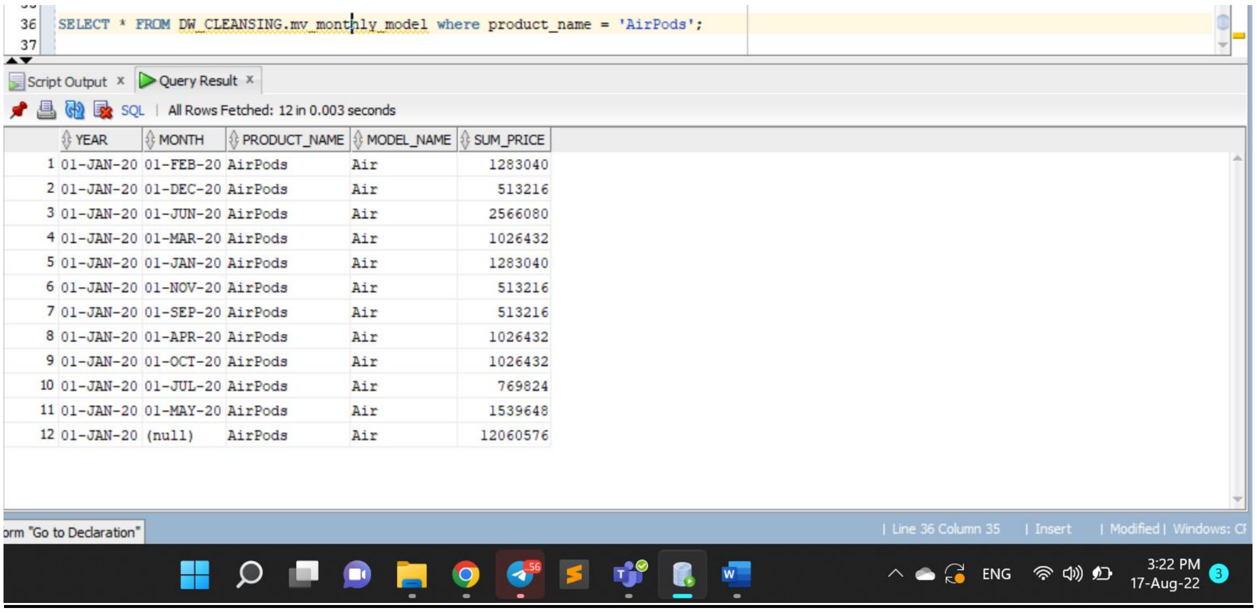


```
UPDATE DW_CLEANSING.CL_TRANSACTIONS
SET price = price - 500
WHERE PRODUCT_NAME = 'AirPods';
```

Script Output
Query Result
Task completed in 2.462 seconds

```
121,824 rows updated.
```

orm "Go to Declaration"
| Line 32 Column 70 | Insert | Modified | Windows: CI
ENG
3:18 PM
17-Aug-22



```
SELECT * FROM DW_CLEANSING.mv_monthly_model where product_name = 'AirPods';
```

	YEAR	MONTH	PRODUCT_NAME	MODEL_NAME	SUM_PRICE
1	01-JAN-20	01-FEB-20	AirPods	Air	1283040
2	01-JAN-20	01-DEC-20	AirPods	Air	513216
3	01-JAN-20	01-JUN-20	AirPods	Air	2566080
4	01-JAN-20	01-MAR-20	AirPods	Air	1026432
5	01-JAN-20	01-JAN-20	AirPods	Air	1283040
6	01-JAN-20	01-NOV-20	AirPods	Air	513216
7	01-JAN-20	01-SEP-20	AirPods	Air	513216
8	01-JAN-20	01-APR-20	AirPods	Air	1026432
9	01-JAN-20	01-OCT-20	AirPods	Air	1026432
10	01-JAN-20	01-JUL-20	AirPods	Air	769824
11	01-JAN-20	01-MAY-20	AirPods	Air	1539648
12	01-JAN-20	(null)	AirPods	Air	12060576

Note. I expressly left screenshots to see sys time, so you can see rows **updated** in 3 minutes **automatically**

<u>Required points:</u>

- Use Standard CREATE MATERIALIZED VIEW CLAUSE
- Test Refreshing at definitive Time moment.

Task Results:

Create required objects:

- Put objects script to Git.
- Prepare Document with Screenshot of Tests Data
- Prepare Document with Screenshot of Refreshing MView scripts

Laboratory Work Summary

At this Laboratory Work we practised different typed **Materialized Views** from previously created **Reports**. Saw the difference between them and their special features. Tested received results.