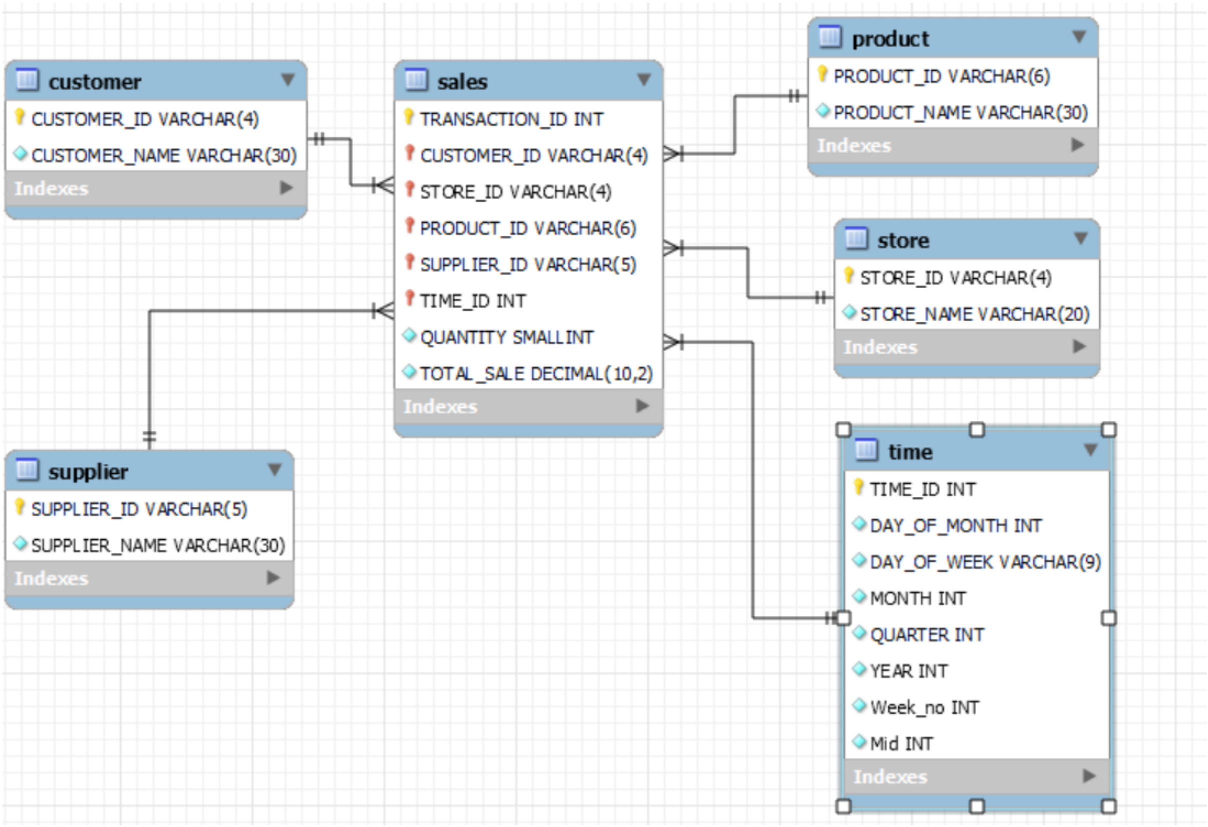
Project Overview

This project is to design and create such a data warehouse system Metro Pakistan | Biggest Wholesale & Supermarket Pakistan. The store has thousands of customers and therefore it is important for the store to online analyze the shopping behavior of their customers. Based on that the store can optimize their selling techniques e.g. giving promotions on different products. Due to the data from one-year transaction data.Transaction table 10000 records and master table 100 products. Program write for Extract Transform Load (ETL) in JAVA. When joining the transaction data and master data in the Meshjoin algorithm. The Start schema is used to model the data structure in the data warehouse first design the star schema. The start schema has five dimensions customer, store, Time, Product, Supplier, and Fact table Sale have two measures Total Sold and Total_Sale on the basis of the transaction. After Extracting and Transforming the data then loaded the data in the Data warehouse. Perform the Online Analytical Processing (OLAP) query.

Schema for DW

The Start schema is used to model the data structure in the data warehouse. Star schema has five dimensions customer, Store, Product, Time, and Supplier. Fact sale table has two-measure Total sold and Total Sale.



MESHJOIN Algorithm

The program dynamically loads the partition size and the stream partition's records size. The partition size of Meshjoin-partition is 10 and the stream partition size is 50. Add transaction records in an input queue. Enqueue the partition when it gets full. Add all new records into the record list. Load the next master partition data. Compare every record product id with master partition product id. Put a new record into the output queue. Last dequeue the last.

Shortcomings in Mesh Join

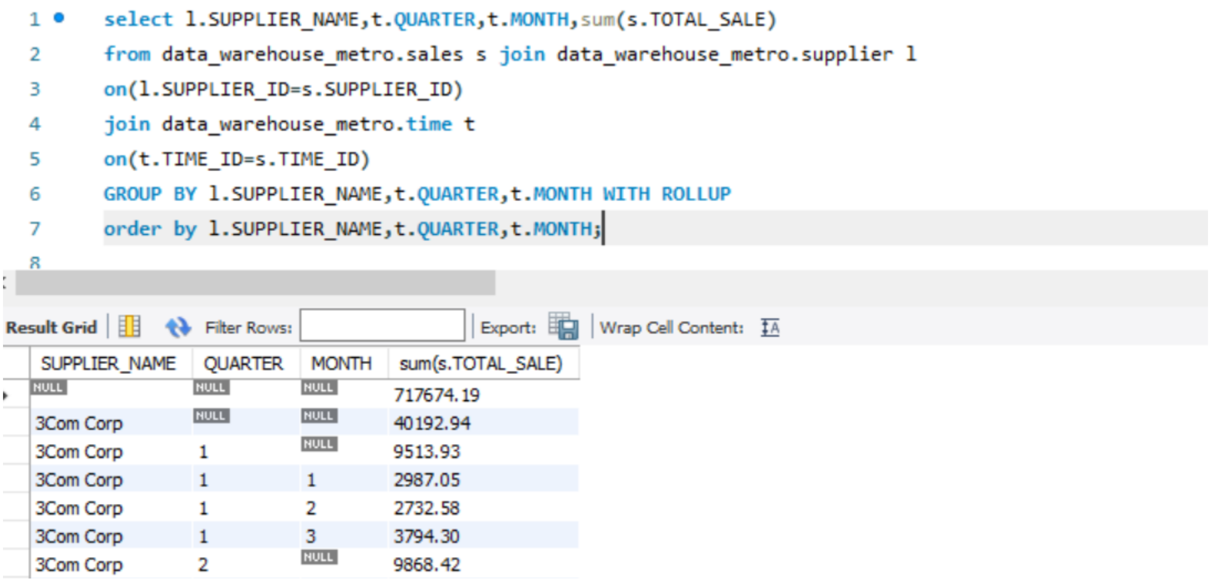
1. Meshjoin depends on the partitions for the internal queue for The stream data and the number of changes needed for the disk based relation In memory . This dependency hampers the optimal distribution of memory between the Join components. Normally the size of the disk-buffer differs with the size of the disk-Based relation and its not necessary.
2. Stream-based algorithm Mesh Join (meshjoin) has been proposed to amortize disk Access over the fast stream. Meshjoin makes no assumptions about the data distribution. In Actual world applications, however, skewed distributions can be found, e.g, some products are sold more often than the remaining products. The question that comes to mind is, how much does the performance of mesh join loose by not adapting to data Skew.
3 The main problem in stream-relation joins is the different nature of inputs; stream data is Fast and busty, where the disk-based relation is not as fast because of the high disk I/O cost.

Lessons Learned

- I learned java language programming and Logic builds Extraction Transform and Load (ETL) import data into Data warehouse.
- A real-time ETL program is to be implemented in reality.
- I learned OLAP querys Roll up , drill down

DW analysis

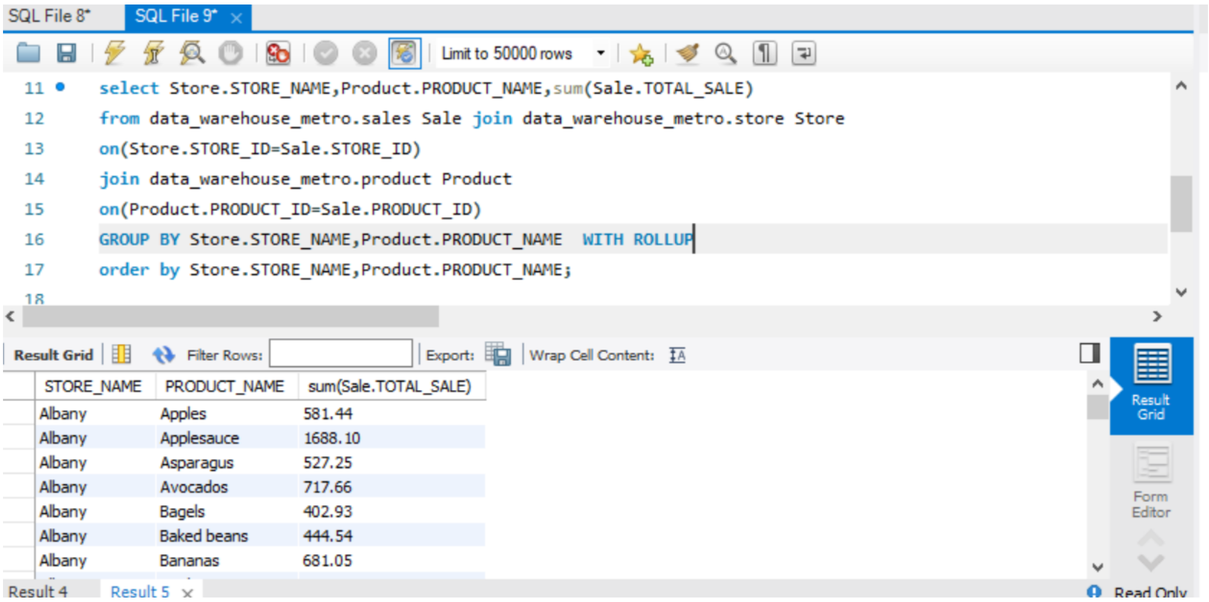
Q1 Present total sales of all products supplied by each supplier with respect to quarter and month



```
select l.SUPPLIER_NAME,t.QUARTER,t.MONTH,sum(s.TOTAL_SALE)
from data_warehouse_metro.sales s join data_warehouse_metro.supplier l
on(l.SUPPLIER_ID=s.SUPPLIER_ID)
join data_warehouse_metro.time t
on(t.TIME_ID=s.TIME_ID)
GROUP BY l.SUPPLIER_NAME,t.QUARTER,t.MONTH WITH ROLLUP
order by l.SUPPLIER_NAME,t.QUARTER,t.MONTH;
```

SUPPLIER_NAME	QUARTER	MONTH	sum(s.TOTAL_SALE)
NULL	NULL	NULL	717674.19
3Com Corp	NULL	NULL	40192.94
3Com Corp	1	NULL	9513.93
3Com Corp	1	1	2987.05
3Com Corp	1	2	2732.58
3Com Corp	1	3	3794.30
3Com Corp	2	NULL	9868.42

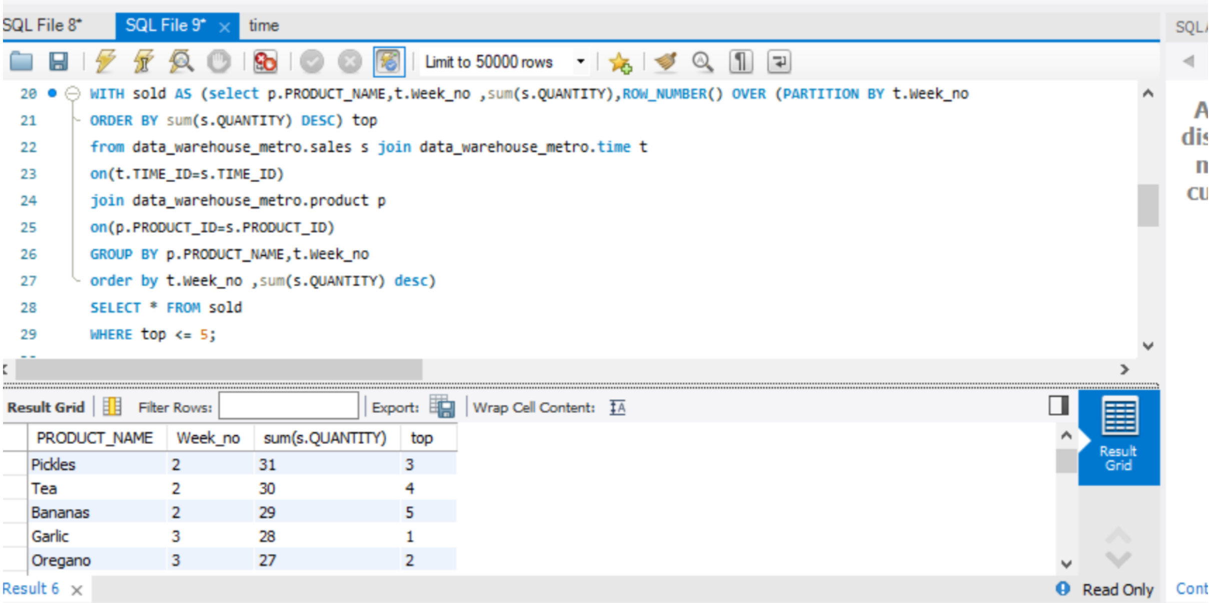
Q2 Present total sales of each product sold by each store. The output should be organised store wise and then product wise under each store.



```
select Store.STORE_NAME,Product.PRODUCT_NAME,sum(Sale.TOTAL_SALE)
from data_warehouse_metro.sales Sale join data_warehouse_metro.store Store
on(Store.STORE_ID=Sale.STORE_ID)
join data_warehouse_metro.product Product
on(Product.PRODUCT_ID=Sale.PRODUCT_ID)
GROUP BY Store.STORE_NAME,Product.PRODUCT_NAME  WITH ROLLUP
order by Store.STORE_NAME,Product.PRODUCT_NAME;
```

STORE_NAME	PRODUCT_NAME	sum(Sale.TOTAL_SALE)
Albany	Apples	581.44
Albany	Applesauce	1688.10
Albany	Asparagus	527.25
Albany	Avocados	717.66
Albany	Bagels	402.93
Albany	Baked beans	444.54
Albany	Bananas	681.05

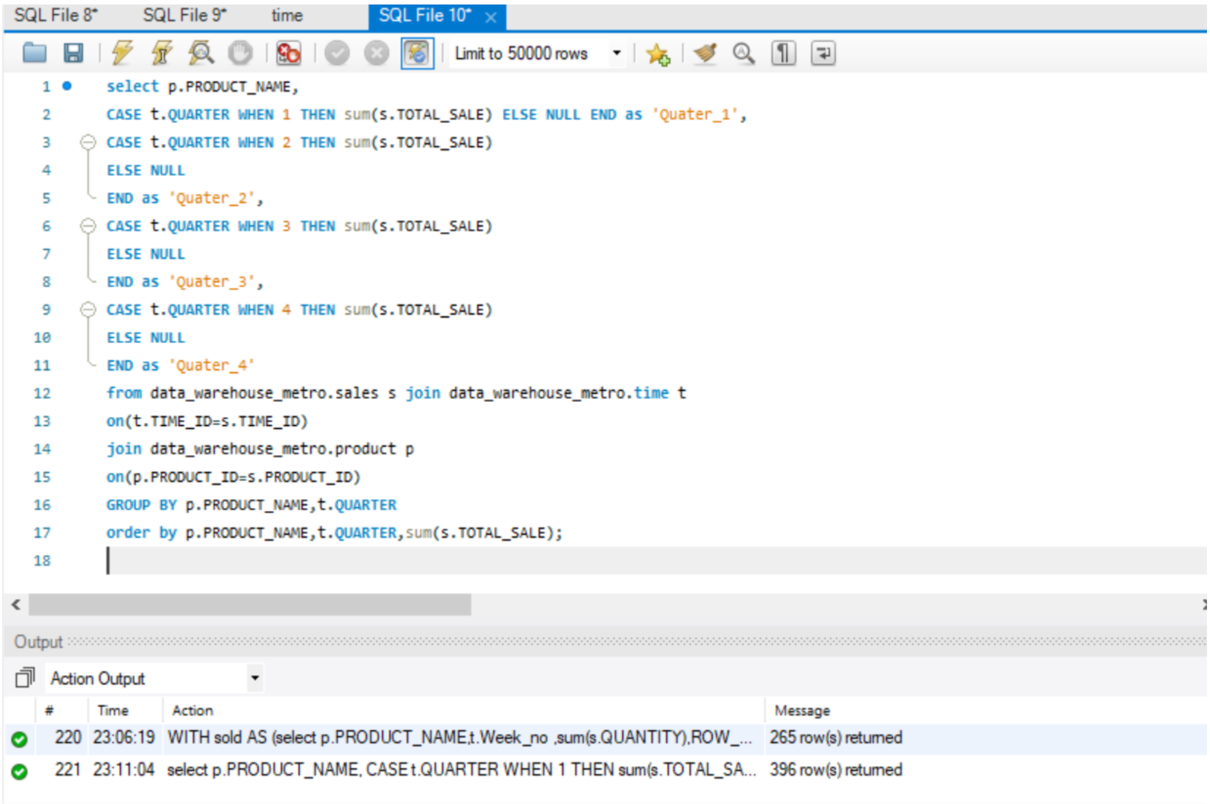
Q3 Find the 5 most popular products sold over the weekends.



```sql
WITH sold AS (select p.PRODUCT_NAME,t.Week_no ,sum(s.QUANTITY),ROW_NUMBER() OVER (PARTITION BY t.Week_no
ORDER BY sum(s.QUANTITY) DESC) top
from data_warehouse_metro.sales s join data_warehouse_metro.time t
on(t.TIME_ID=s.TIME_ID)
join data_warehouse_metro.product p
on(p.PRODUCT_ID=s.PRODUCT_ID)
GROUP BY p.PRODUCT_NAME,t.Week_no
order by t.Week_no ,sum(s.QUANTITY) desc)
SELECT * FROM sold
WHERE top <= 5;
```

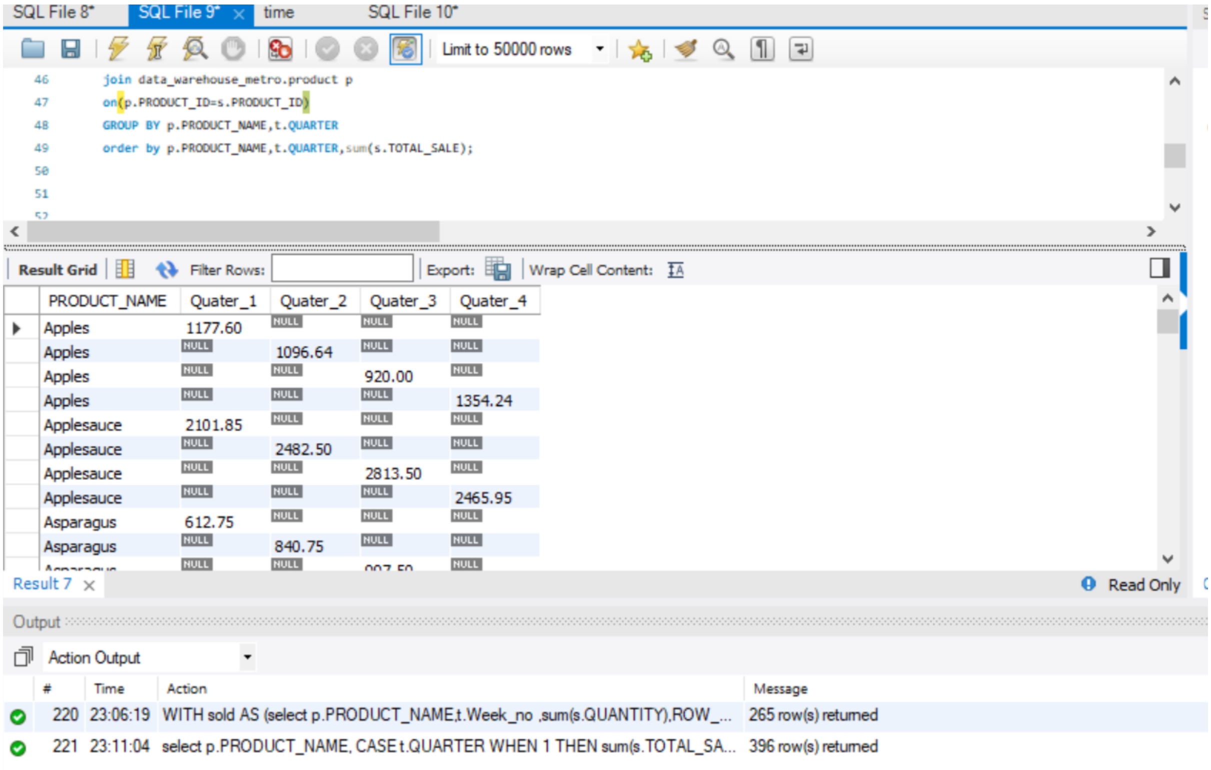
PRODUCT_NAME	Week_no	sum(s.QUANTITY)	top
Pickles	2	31	3
Tea	2	30	4
Bananas	2	29	5
Garlic	3	28	1
Oregano	3	27	2

Q4 Present the quarterly sales of each product for year 2016 using drill down query concept. Note: each quarter sale must be a column.



```sql
select p.PRODUCT_NAME,
CASE t.QUARTER WHEN 1 THEN sum(s.TOTAL_SALE) ELSE NULL END as 'Quater_1',
CASE t.QUARTER WHEN 2 THEN sum(s.TOTAL_SALE)
ELSE NULL
END as 'Quater_2',
CASE t.QUARTER WHEN 3 THEN sum(s.TOTAL_SALE)
ELSE NULL
END as 'Quater_3',
CASE t.QUARTER WHEN 4 THEN sum(s.TOTAL_SALE)
ELSE NULL
END as 'Quater_4'
from data_warehouse_metro.sales s join data_warehouse_metro.time t
on(t.TIME_ID=s.TIME_ID)
join data_warehouse_metro.product p
on(p.PRODUCT_ID=s.PRODUCT_ID)
GROUP BY p.PRODUCT_NAME,t.QUARTER
order by p.PRODUCT_NAME,t.QUARTER,sum(s.TOTAL_SALE);
```

	#	Time	Action	Message
✓	220	23:06:19	WITH sold AS (select p.PRODUCT_NAME,t.Week_no ,sum(s.QUANTITY),ROW_...	265 row(s) returned
✓	221	23:11:04	select p.PRODUCT_NAME, CASE t.QUARTER WHEN 1 THEN sum(s.TOTAL_SA...	396 row(s) returned



Limit to 50000 rows

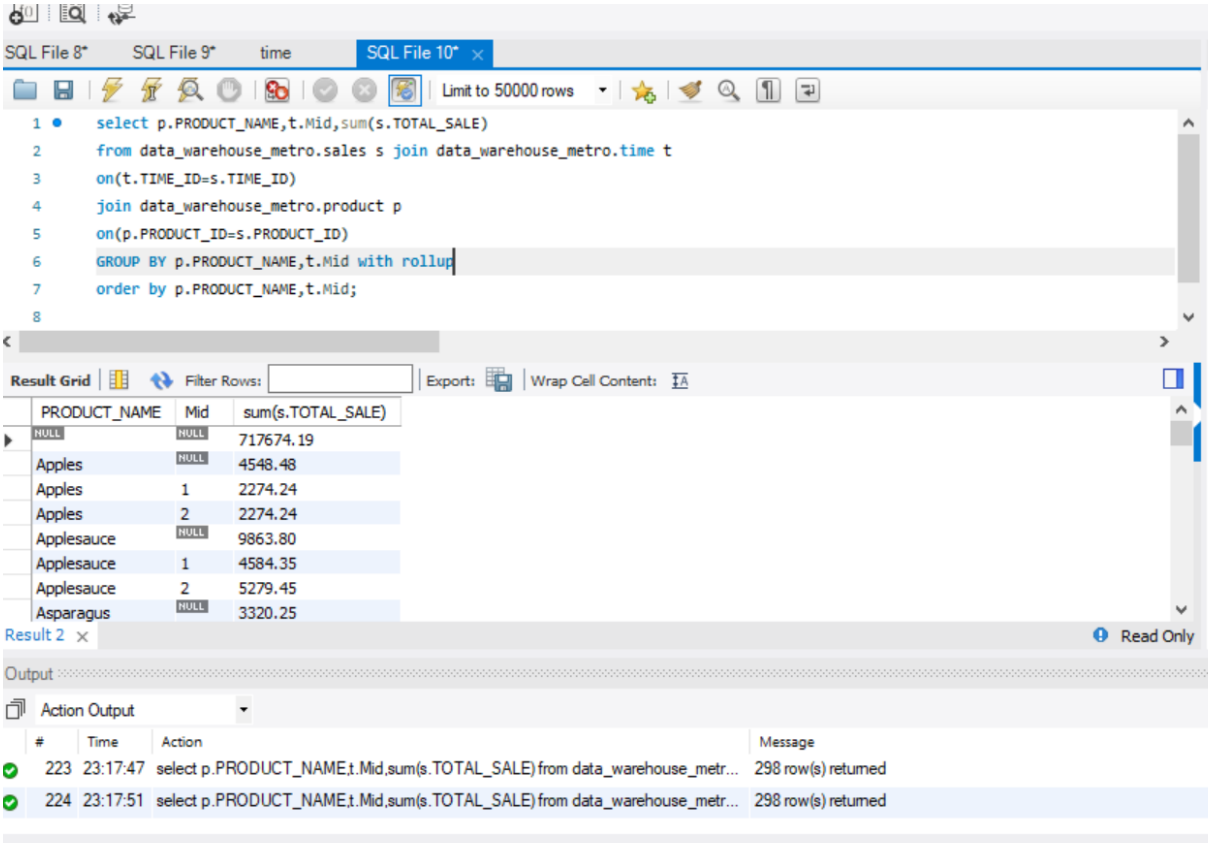
```sql
join data_warehouse_metro.product p
on(p.PRODUCT_ID=s.PRODUCT_ID)
GROUP BY p.PRODUCT_NAME,t.QUARTER
order by p.PRODUCT_NAME,t.QUARTER,sum(s.TOTAL_SALE);
```

PRODUCT_NAME	Quater_1	Quater_2	Quater_3	Quater_4
Apples	1177.60	NULL	NULL	NULL
Apples	NULL	1096.64	NULL	NULL
Apples	NULL	NULL	920.00	NULL
Apples	NULL	NULL	NULL	1354.24
Applesauce	2101.85	NULL	NULL	NULL
Applesauce	NULL	2482.50	NULL	NULL
Applesauce	NULL	NULL	2813.50	NULL
Applesauce	NULL	NULL	NULL	2465.95
Asparagus	612.75	NULL	NULL	NULL
Asparagus	NULL	840.75	NULL	NULL

Action Output

```
#    Time      Action                                                              Message
220  23:06:19  WITH sold AS (select p.PRODUCT_NAME,t.Week_no ,sum(s.QUANTITY),ROW_...  265 row(s) returned
221  23:11:04  select p.PRODUCT_NAME, CASE t.QUARTER WHEN 1 THEN sum(s.TOTAL_SA...   396 row(s) returned
```

Q5 Extract total sales of each product for the first and second half of year 2016 along with its total yearly sales.

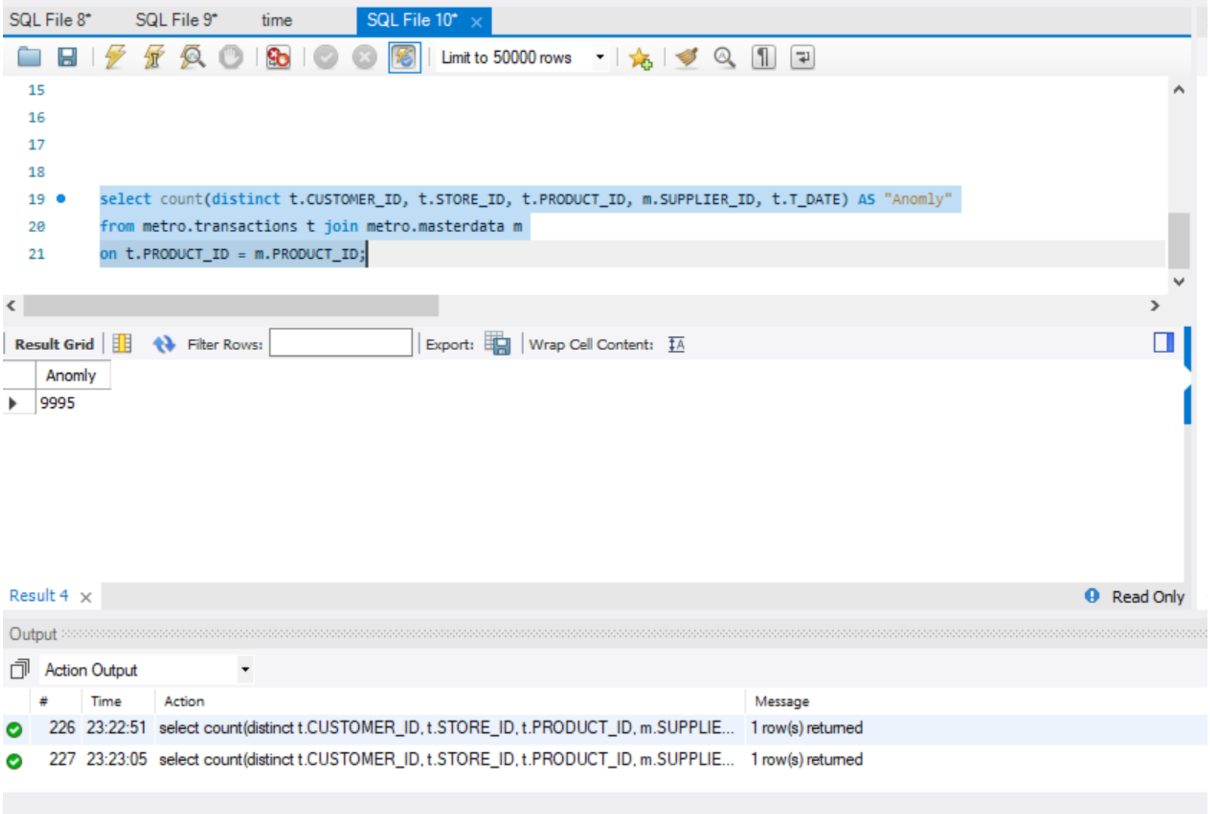


```sql
select p.PRODUCT_NAME,t.Mid,sum(s.TOTAL_SALE)
from data_warehouse_metro.sales s join data_warehouse_metro.time t
on(t.TIME_ID=s.TIME_ID)
join data_warehouse_metro.product p
on(p.PRODUCT_ID=s.PRODUCT_ID)
GROUP BY p.PRODUCT_NAME,t.Mid with rollup
order by p.PRODUCT_NAME,t.Mid;
```

PRODUCT_NAME	Mid	sum(s.TOTAL_SALE)
NULL	NULL	717674.19
Apples	NULL	4548.48
Apples	1	2274.24
Apples	2	2274.24
Applesauce	NULL	9863.80
Applesauce	1	4584.35
Applesauce	2	5279.45
Asparagus	NULL	3320.25

#	Time	Action	Message
223	23:17:47	select p.PRODUCT_NAME,t.Mid,sum(s.TOTAL_SALE) from data_warehouse_metr...	298 row(s) returned
224	23:17:51	select p.PRODUCT_NAME,t.Mid,sum(s.TOTAL_SALE) from data_warehouse_metr...	298 row(s) returned

<u>Q6 Find an anomaly in the data warehouse dataset. write a query to show the anomaly and explain the anomaly in your project report.</u>



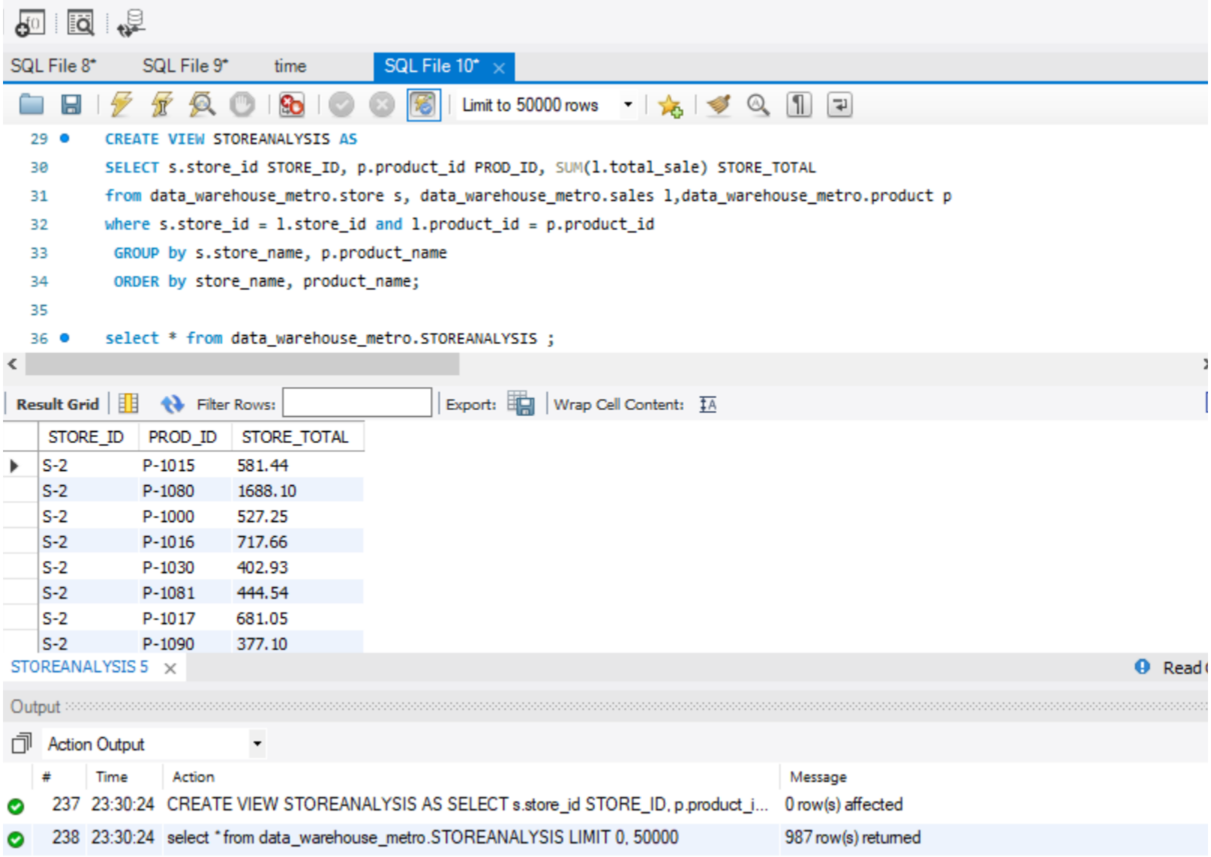
Limit to 50000 rows

```
select count(distinct t.CUSTOMER_ID, t.STORE_ID, t.PRODUCT_ID, m.SUPPLIER_ID, t.T_DATE) AS "Anomly"
from metro.transactions t join metro.masterdata m
on t.PRODUCT_ID = m.PRODUCT_ID;
```

Anomly
9995

```
    #    Time      Action                                                              Message
✓   226  23:22:51  select count(distinct t.CUSTOMER_ID, t.STORE_ID, t.PRODUCT_ID, m.SUPPLIE...  1 row(s) returned
✓   227  23:23:05  select count(distinct t.CUSTOMER_ID, t.STORE_ID, t.PRODUCT_ID, m.SUPPLIE...  1 row(s) returned
```

Q7 Create a materialised view with the name “STOREANALYSIS_MV” that presents the product- wise sales analysis for each store.



```
CREATE VIEW STOREANALYSIS AS
SELECT s.store_id STORE_ID, p.product_id PROD_ID, SUM(l.total_sale) STORE_TOTAL
from data_warehouse_metro.store s, data_warehouse_metro.sales l,data_warehouse_metro.product p
where s.store_id = l.store_id and l.product_id = p.product_id
 GROUP by s.store_name, p.product_name
 ORDER by store_name, product_name;

select * from data_warehouse_metro.STOREANALYSIS ;
```

STORE_ID	PROD_ID	STORE_TOTAL
S-2	P-1015	581.44
S-2	P-1080	1688.10
S-2	P-1000	527.25
S-2	P-1016	717.66
S-2	P-1030	402.93
S-2	P-1081	444.54
S-2	P-1017	681.05
S-2	P-1090	377.10

#	Time	Action	Message
237	23:30:24	CREATE VIEW STOREANALYSIS AS SELECT s.store_id STORE_ID, p.product_i...	0 row(s) affected
238	23:30:24	select * from data_warehouse_metro.STOREANALYSIS LIMIT 0, 50000	987 row(s) returned