National University
of computer and emerging sciences

Data WareHousing

Project Report

Zahid Iqbal 17I-0035 Section B

10th Dec, 2020

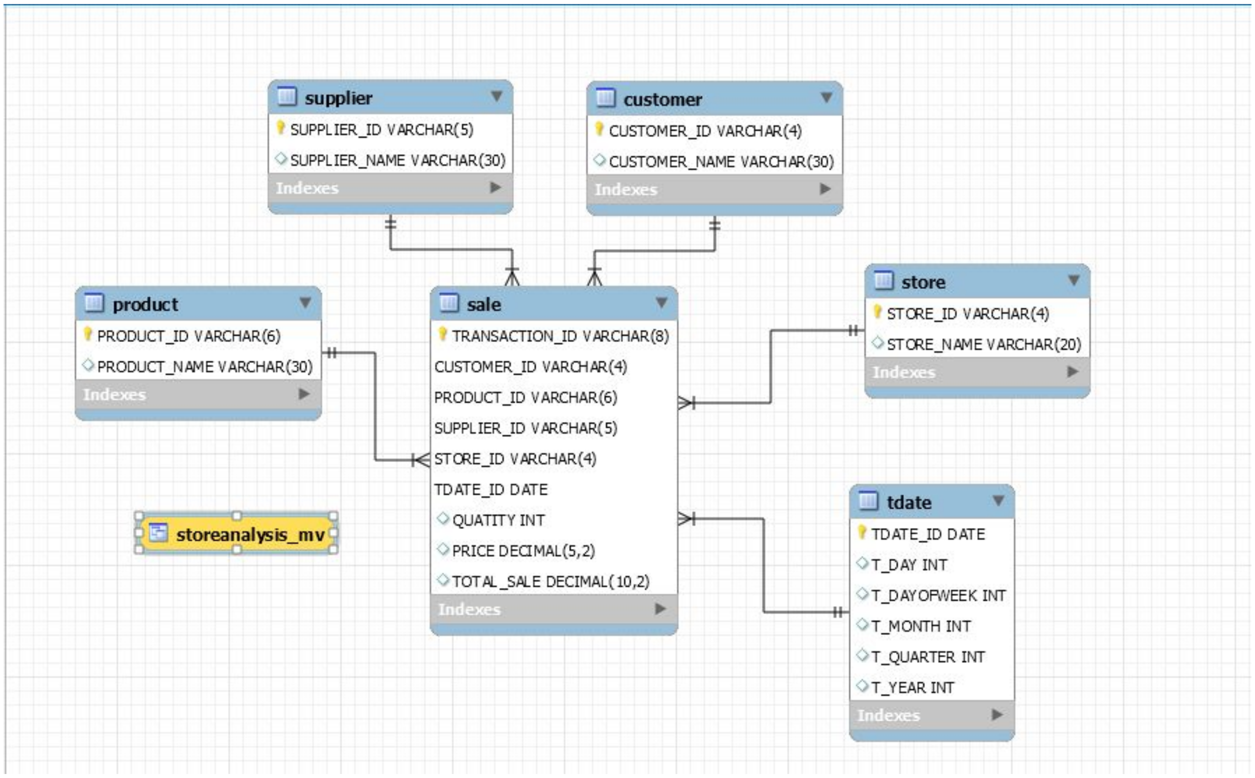
Introduction and Project Overview:

METRO is one of the biggest superstores chains in Pakistan. The stores has thousands of customers and therefore it is important for the store to online analyse the shopping behaviour of their customers. Based on that the store can optimise their selling techniques e.g. giving of promotions on different products.

Now, to make this analysis of shopping behaviour practical there is a need of building a near-real-time DW and customers' transactions from Data Sources (DSs) are required to reflect into DW as soon as they appear in DSs. To build a near-real-time DW we need to implement a near-real-time ETL (Extraction, Transformation, and Loading) tool. Since the data generated by customers is not in the format required by DW therefore, it needs to process in the transformation layer of ETL. For example enriching of some information e.g. attributes in colour red from disk-based Master Data (MD).

To implement this enrichment feature in the transformation phase of ETL we need a join operator. There are a number of algorithms available to implement this join operation however, the most A popular one is HYBRIDJOIN (Hybrid Join) which is explained in the next section and we will implement it in this project using Java Eclipse.

Schema of DWH:



Hyper Join Algorithm:

I'll explain also with code

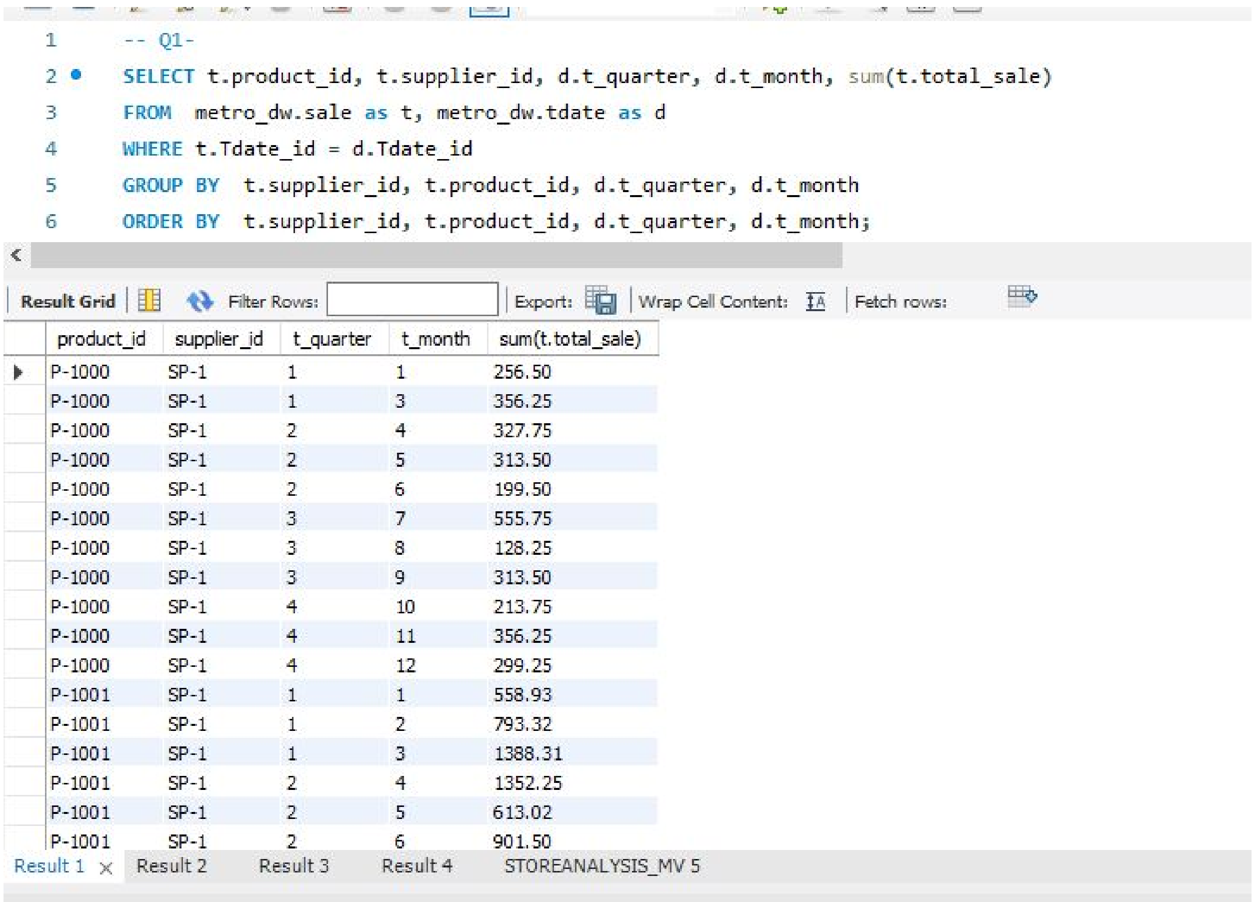
```
class StagingArea {

   class HashMapEntry { // used for hashmap entry

            TransactionData td = null;
            Node pointer_to_queue = null;
   }

   // data members

   DLQ queue = new DLQ(); // dubbely queue
```

```
//product id maps to HashMapEntry object

 MultiValuedMap<String, HashMapEntry> hashmaps = new ArrayListValuedHashMap<>();
ArrayList<MasterData> master_data = new ArrayList<MasterData>(); // master data buffer
int max_hashmaps_entries = 1000;  // max entries on a hash maps
 int max_masterdata_buffer_size = 20; // max entries in masterdata buffer
 int starting_index = 0; //

    //member functions

    public void read_transaction_data(Statement stmt, String table){
             // this function loads transaction data into hashmaps and product_id in a queue.
  }

   public int read_master_data(Statement stmt, String table, String p_id){
             // this function loads master data into master data buffer
    }

    public List<Object>  setup_database_conn(String database){
             // this function setups db conn and return Statement and Connection objects
             return Arrays.asList(stmt, con);
    }

public void send_data_to_dwh(TransactionData td, MasterData masterData, Statement
dwh_statement) {
            // this function reciives the materData tupple (from masterData buffer) and transaction
            data //tuples (from hashtables) and loads it into DWH
    }
Main function {

            // set up data_base conn
             setup_database_conn("metro_DB");
             // set up DWH conn
             setup_database_conn("metro_DW");

             // read and store 1000 transactions into hashtable and product_ids into queue
             read_transaction_data(db_statement, "TRANSACTIONS");
```

```
        // loop untill all data is loaded into dwh
        while(queue is not empty){

                product_id = queue.dequeue();
                // read masterData into masterData buffer based on product_id

                read_master_data(db_statement, "MASTERDATA", product_id);

                //1
                //iter over all master data values
                // iter over all hashmap values if matched
                // then attach/join master data and hashmap values and send it to
                // send_data_to_dwh() function.
                //after that remove the entry from queue too.



                //2
                //now iter over remaining  hashmaps values
                //match it with master data buffer values and if found
                //join and sent it to send_data_to_dwh() function.
                //after that remove the entry from queue too.

                // now read and store transactions into hashtable to complete 1000 entries in
               //hashpmap and product_ids into queue

               read_transaction_data(db_statement, "TRANSACTIONS");


        }

        // close db conn
        // close dwh conn



  }

}
```

OLAP Queries output:

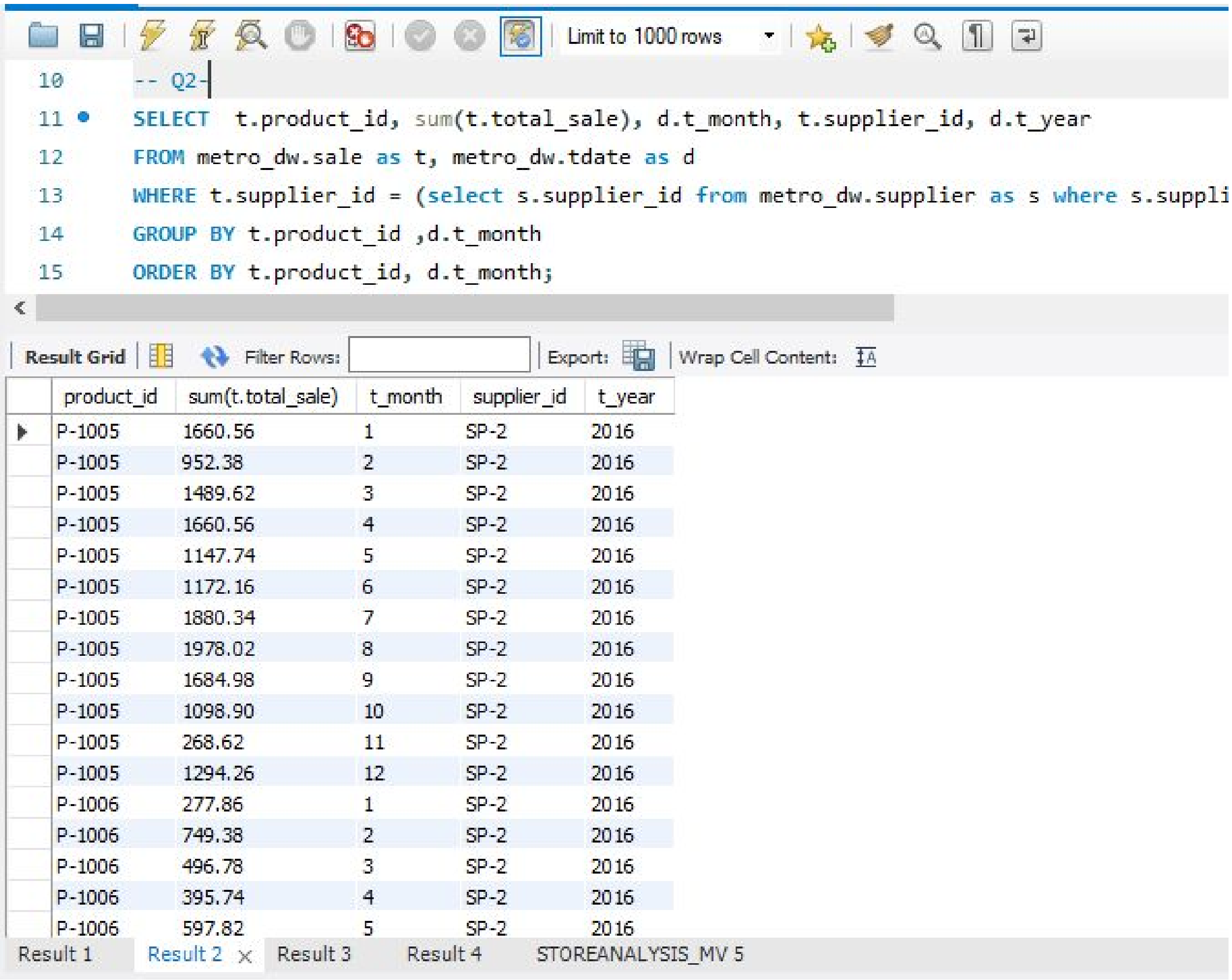
Q1



```
-- Q1-
SELECT t.product_id, t.supplier_id, d.t_quarter, d.t_month, sum(t.total_sale)
FROM  metro_dw.sale as t, metro_dw.tdate as d
WHERE t.Tdate_id = d.Tdate_id
GROUP BY  t.supplier_id, t.product_id, d.t_quarter, d.t_month
ORDER BY  t.supplier_id, t.product_id, d.t_quarter, d.t_month;
```

product_id	supplier_id	t_quarter	t_month	sum(t.total_sale)
P-1000	SP-1	1	1	256.50
P-1000	SP-1	1	3	356.25
P-1000	SP-1	2	4	327.75
P-1000	SP-1	2	5	313.50
P-1000	SP-1	2	6	199.50
P-1000	SP-1	3	7	555.75
P-1000	SP-1	3	8	128.25
P-1000	SP-1	3	9	313.50
P-1000	SP-1	4	10	213.75
P-1000	SP-1	4	11	356.25
P-1000	SP-1	4	12	299.25
P-1001	SP-1	1	1	558.93
P-1001	SP-1	1	2	793.32
P-1001	SP-1	1	3	1388.31
P-1001	SP-1	2	4	1352.25
P-1001	SP-1	2	5	613.02
P-1001	SP-1	2	6	901.50

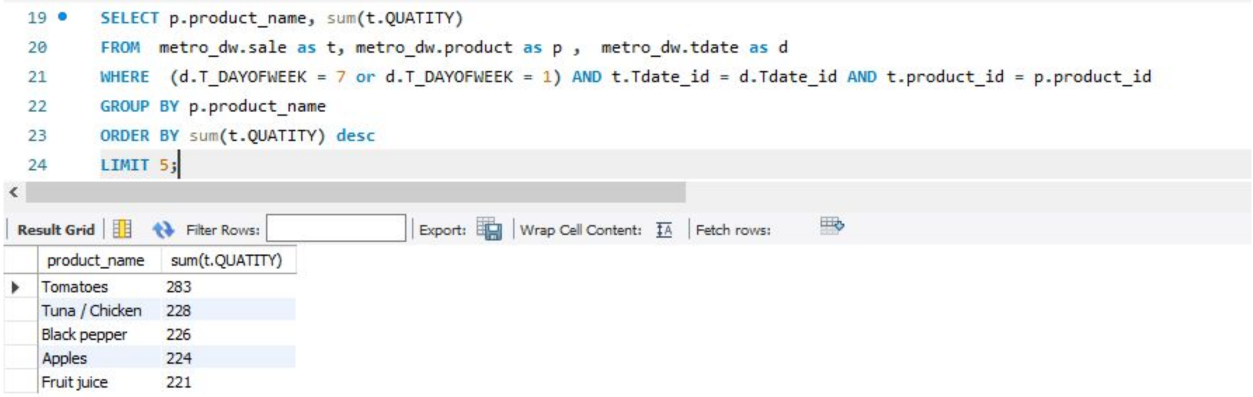
Q2



```
-- Q2
SELECT  t.product_id, sum(t.total_sale), d.t_month, t.supplier_id, d.t_year
FROM metro_dw.sale as t, metro_dw.tdate as d
WHERE t.supplier_id = (select s.supplier_id from metro_dw.supplier as s where s.suppli
GROUP BY t.product_id ,d.t_month
ORDER BY t.product_id, d.t_month;
```

product_id	sum(t.total_sale)	t_month	supplier_id	t_year
P-1005	1660.56	1	SP-2	2016
P-1005	952.38	2	SP-2	2016
P-1005	1489.62	3	SP-2	2016
P-1005	1660.56	4	SP-2	2016
P-1005	1147.74	5	SP-2	2016
P-1005	1172.16	6	SP-2	2016
P-1005	1880.34	7	SP-2	2016
P-1005	1978.02	8	SP-2	2016
P-1005	1684.98	9	SP-2	2016
P-1005	1098.90	10	SP-2	2016
P-1005	268.62	11	SP-2	2016
P-1005	1294.26	12	SP-2	2016
P-1006	277.86	1	SP-2	2016
P-1006	749.38	2	SP-2	2016
P-1006	496.78	3	SP-2	2016
P-1006	395.74	4	SP-2	2016
P-1006	597.82	5	SP-2	2016

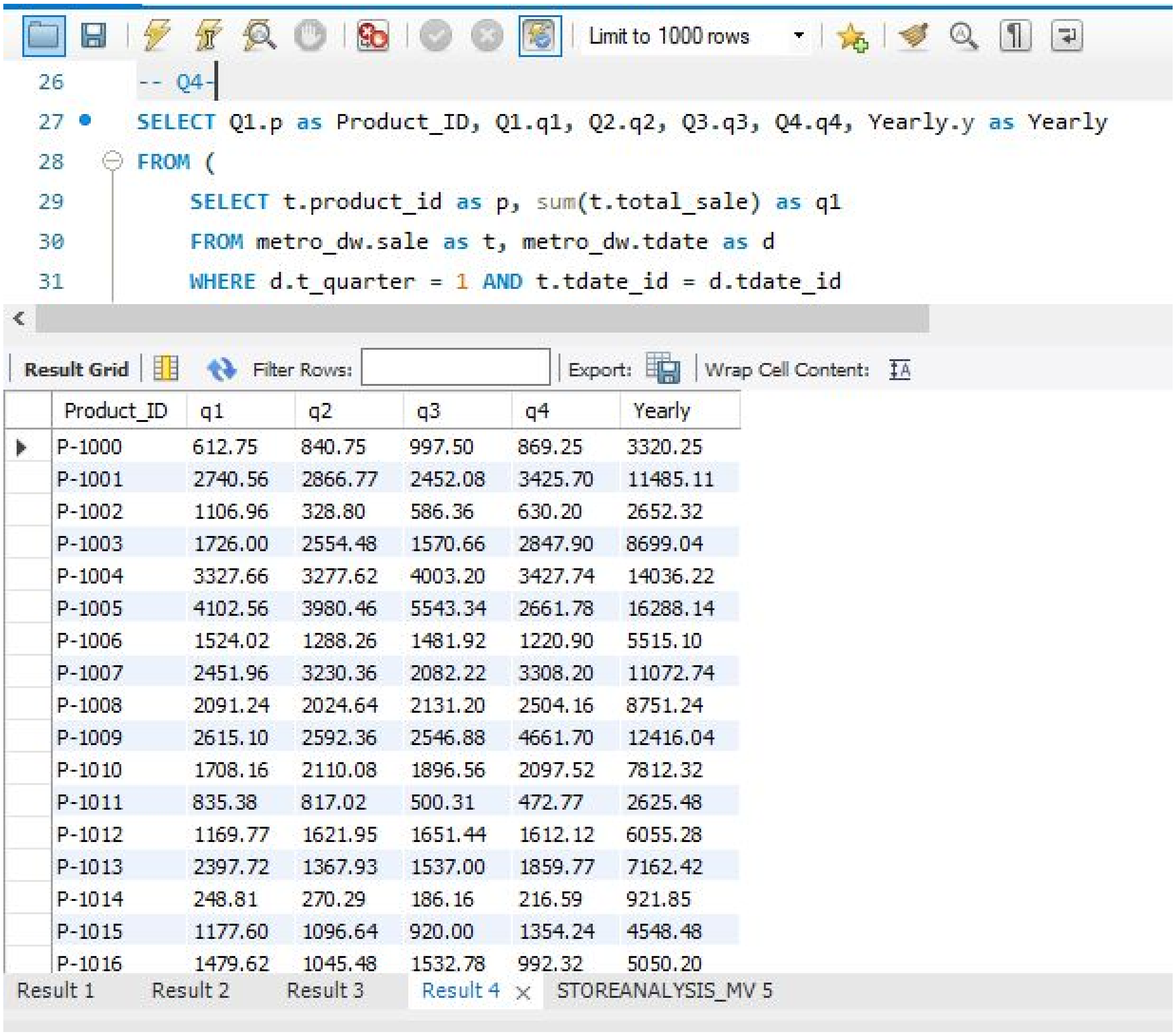
Q3



```sql
SELECT p.product_name, sum(t.QUATITY)
FROM  metro_dw.sale as t, metro_dw.product as p ,  metro_dw.tdate as d
WHERE  (d.T_DAYOFWEEK = 7 or d.T_DAYOFWEEK = 1) AND t.Tdate_id = d.Tdate_id AND t.product_id = p.product_id
GROUP BY p.product_name
ORDER BY sum(t.QUATITY) desc
LIMIT 5;
```

product_name	sum(t.QUATITY)
Tomatoes	283
Tuna / Chicken	228
Black pepper	226
Apples	224
Fruit juice	221

Q4

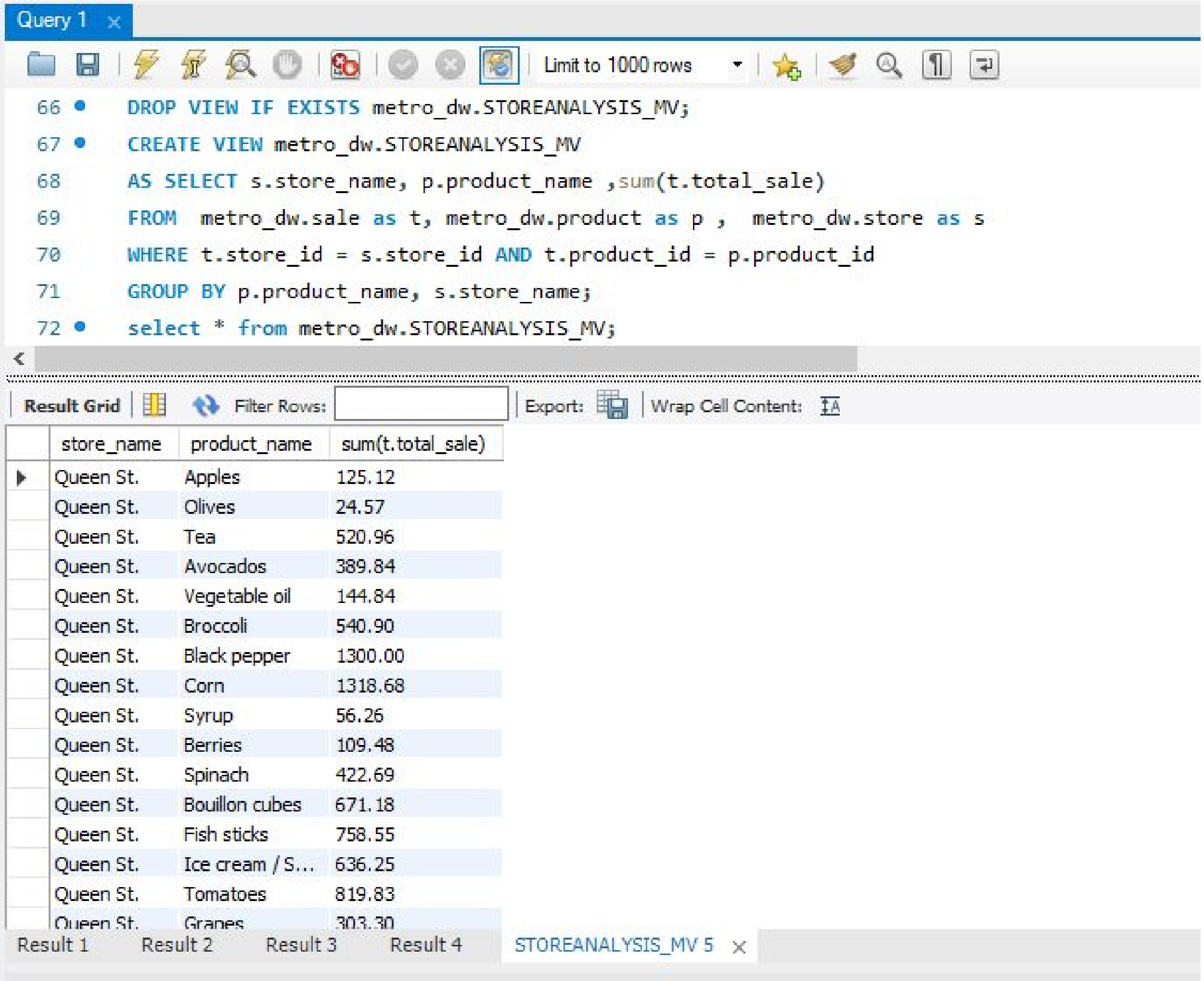


```sql
-- Q4-
SELECT Q1.p as Product_ID, Q1.q1, Q2.q2, Q3.q3, Q4.q4, Yearly.y as Yearly
FROM (
    SELECT t.product_id as p, sum(t.total_sale) as q1
    FROM metro_dw.sale as t, metro_dw.tdate as d
    WHERE d.t_quarter = 1 AND t.tdate_id = d.tdate_id
```

Product_ID	q1	q2	q3	q4	Yearly
P-1000	612.75	840.75	997.50	869.25	3320.25
P-1001	2740.56	2866.77	2452.08	3425.70	11485.11
P-1002	1106.96	328.80	586.36	630.20	2652.32
P-1003	1726.00	2554.48	1570.66	2847.90	8699.04
P-1004	3327.66	3277.62	4003.20	3427.74	14036.22
P-1005	4102.56	3980.46	5543.34	2661.78	16288.14
P-1006	1524.02	1288.26	1481.92	1220.90	5515.10
P-1007	2451.96	3230.36	2082.22	3308.20	11072.74
P-1008	2091.24	2024.64	2131.20	2504.16	8751.24
P-1009	2615.10	2592.36	2546.88	4661.70	12416.04
P-1010	1708.16	2110.08	1896.56	2097.52	7812.32
P-1011	835.38	817.02	500.31	472.77	2625.48
P-1012	1169.77	1621.95	1651.44	1612.12	6055.28
P-1013	2397.72	1367.93	1537.00	1859.77	7162.42
P-1014	248.81	270.29	186.16	216.59	921.85
P-1015	1177.60	1096.64	920.00	1354.24	4548.48
P-1016	1479.62	1045.48	1532.78	992.32	5050.20

Q5 Anomaly : in this data warehouse dataset is that in transaction table if we make the a primary key by combining all forign keys the dataset still have some records are recurring e.g if a person buys the same product on the same day from the same store, supplied by same supplier. The Anomaly is occurring because dwh doesn’t store time of the sale. And to solve this problem I’m using Transaction_ID with all other forign keys to make the primary key of the sales table.

Q6



```
DROP VIEW IF EXISTS metro_dw.STOREANALYSIS_MV;
CREATE VIEW metro_dw.STOREANALYSIS_MV
AS SELECT s.store_name, p.product_name ,sum(t.total_sale)
FROM  metro_dw.sale as t, metro_dw.product as p ,  metro_dw.store as s
WHERE t.store_id = s.store_id AND t.product_id = p.product_id
GROUP BY p.product_name, s.store_name;
select * from metro_dw.STOREANALYSIS_MV;
```

store_name	product_name	sum(t.total_sale)
Queen St.	Apples	125.12
Queen St.	Olives	24.57
Queen St.	Tea	520.96
Queen St.	Avocados	389.84
Queen St.	Vegetable oil	144.84
Queen St.	Broccoli	540.90
Queen St.	Black pepper	1300.00
Queen St.	Corn	1318.68
Queen St.	Syrup	56.26
Queen St.	Berries	109.48
Queen St.	Spinach	422.69
Queen St.	Bouillon cubes	671.18
Queen St.	Fish sticks	758.55
Queen St.	Ice cream / S...	636.25
Queen St.	Tomatoes	819.83
Queen St.	Grapes	303.30

Two shortcomings of hybridJoin

1 - In case of frequent data e.g products like milk,etc hybridJoin doesn’t have a system to store its masterData tuples in cache so that it doesn’t have to read the same product again and again.

2 -

What I have Learned

This project was very balanced wrt to what we have learned in the class. It was a good opportunity for us to get hands on experience for creating DWH by applying all the concepts we have learned in class.