# ASSIGNMENT-2

# DATA WAREHOUSING AND BIG DATA

## Overview:

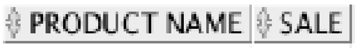
This project is regarding the largest supermarket chain “New World” in New Zealand. Till today there are 140 stores located around north and South Island of New Zealand. We know that it is important for any big organization to keep up-to-date information of their customers purchasing pattern and records of products in their database. More than thousands of customers purchase different product from the store. Therefore, like other companies by analysing data from the database this organization can improve their productivity by attracting customers through special offers on products. Thus, this project is about creating data warehouse from the existing database of the transactions. Thereafter solving OLAP queries to analyse the data warehouse.

## Pseudocode:

```
First put set serveroutput on size 10000.
Write DECLARE to state the variable.
Initialise the variables initial_1, initial_2, initial_3.
Initialise the characters sl1, sl2, sl3, sl4, sl5, sl6.
Pass CURSOR ab and parameterized cursor int1 and int2 for Transactions table to fetch 100 tuples till 10000.
Pass CURSOR cd and parameterized cursor int3 to join the columns from the following table.
Pass CURSORS ef, kl, gh, mn, ij, op along with respective parameterized cursor int4, int7, int5, int8, int6, int9 to product_id, store_id, customer_id, time_id, supplier_id, transaction_id respectively.
Give the datatype for all columns.
Pass the loop for the three variables (initial_1, initial_2, initial_3) declared before.
Open all the CURSORS to insert data into the warehouse.
```

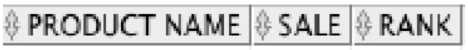
## OLAP queries:

1. Which product generated maximum sales in September, 2017?

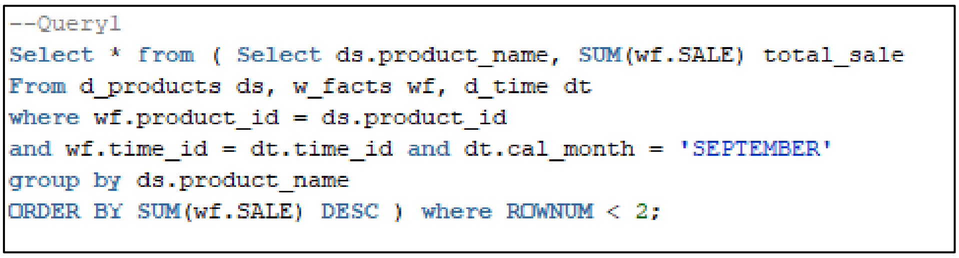


| PRODUCT NAME | SALE |
|---|---|

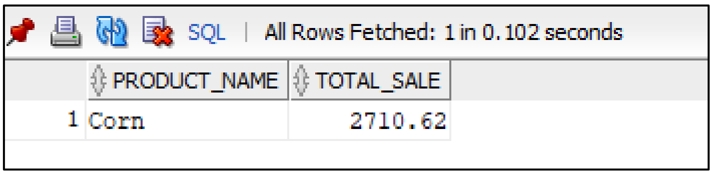
OR



| PRODUCT NAME | SALE | RANK |
|---|---|---|



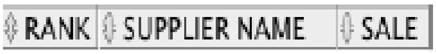
```
--Query1
Select * from ( Select ds.product_name, SUM(wf.SALE) total_sale
From d_products ds, w_facts wf, d_time dt
where wf.product_id = ds.product_id
and wf.time_id = dt.time_id and dt.cal_month = 'SEPTEMBER'
group by ds.product_name
ORDER BY SUM(wf.SALE) DESC ) where ROWNUM < 2;
```



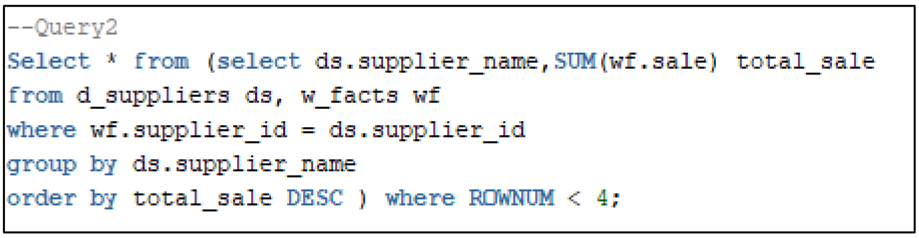
All Rows Fetched: 1 in 0.102 seconds

|   | PRODUCT_NAME | TOTAL_SALE |
|---|---|---|
| 1 | Corn | 2710.62 |

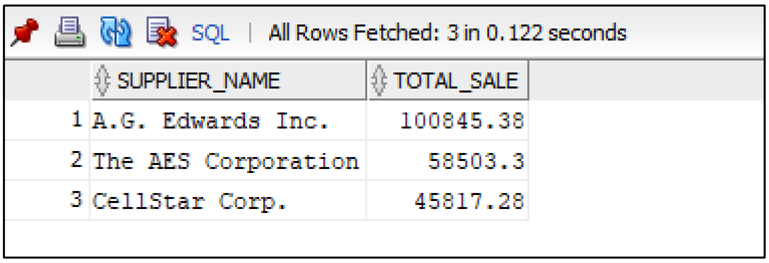
2. Determine top three supplier names based on highest sales of their products.



| RANK | SUPPLIER NAME | SALE |
|---|---|---|



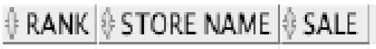
```
--Query2
Select * from (select ds.supplier_name,SUM(wf.sale) total_sale
from d_suppliers ds, w_facts wf
where wf.supplier_id = ds.supplier_id
group by ds.supplier_name
order by total_sale DESC ) where ROWNUM < 4;
```



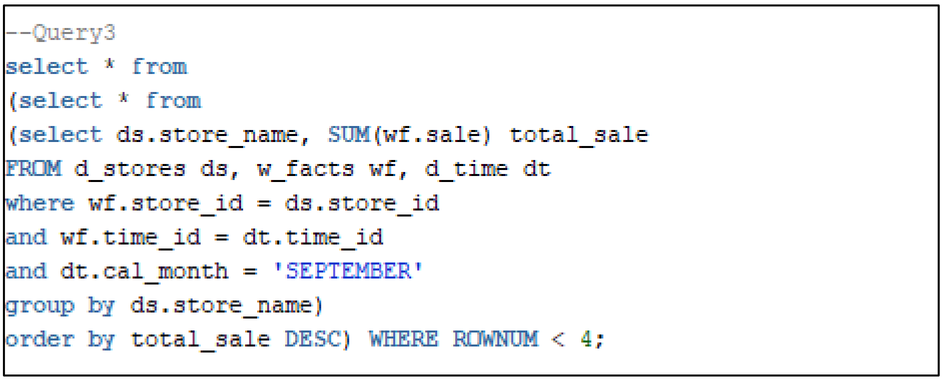
SQL | All Rows Fetched: 3 in 0.122 seconds

|   | SUPPLIER_NAME | TOTAL_SALE |
|---|---|---|
| 1 | A.G. Edwards Inc. | 100845.38 |
| 2 | The AES Corporation | 58503.3 |
| 3 | CellStar Corp. | 45817.28 |

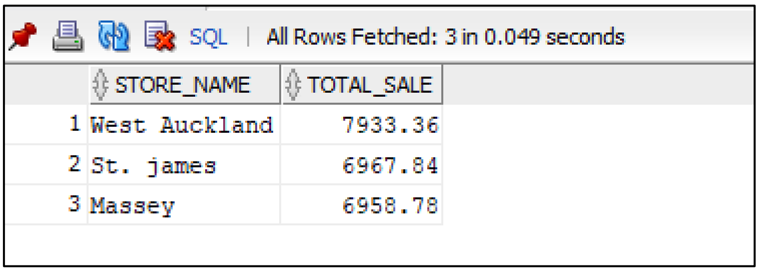
3. Determine the top 3 store names who generated highest sales in September, 2017.



| RANK | STORE NAME | SALE |
|---|---|---|



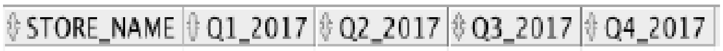
```
--Query3
select * from
(select * from
(select ds.store_name, SUM(wf.sale) total_sale
FROM d_stores ds, w_facts wf, d_time dt
where wf.store_id = ds.store_id
and wf.time_id = dt.time_id
and dt.cal_month = 'SEPTEMBER'
group by ds.store_name)
order by total_sale DESC) WHERE ROWNUM < 4;
```



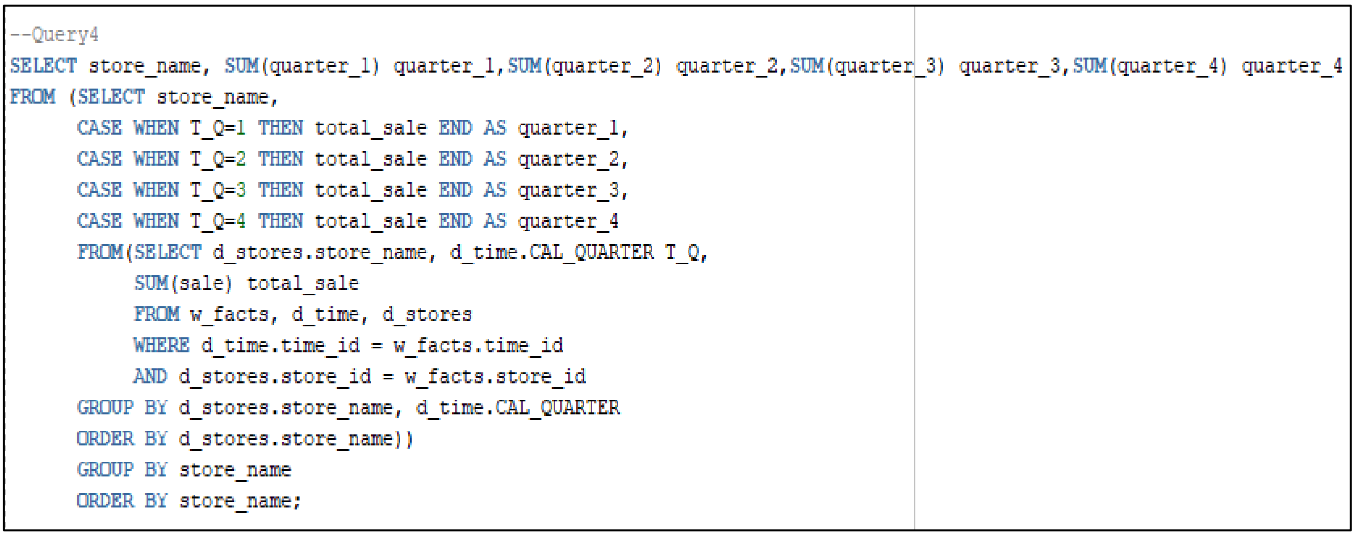
SQL | All Rows Fetched: 3 in 0.049 seconds

|   | STORE_NAME | TOTAL_SALE |
|---|---|---|
| 1 | West Auckland | 7933.36 |
| 2 | St. james | 6967.84 |
| 3 | Massey | 6958.78 |

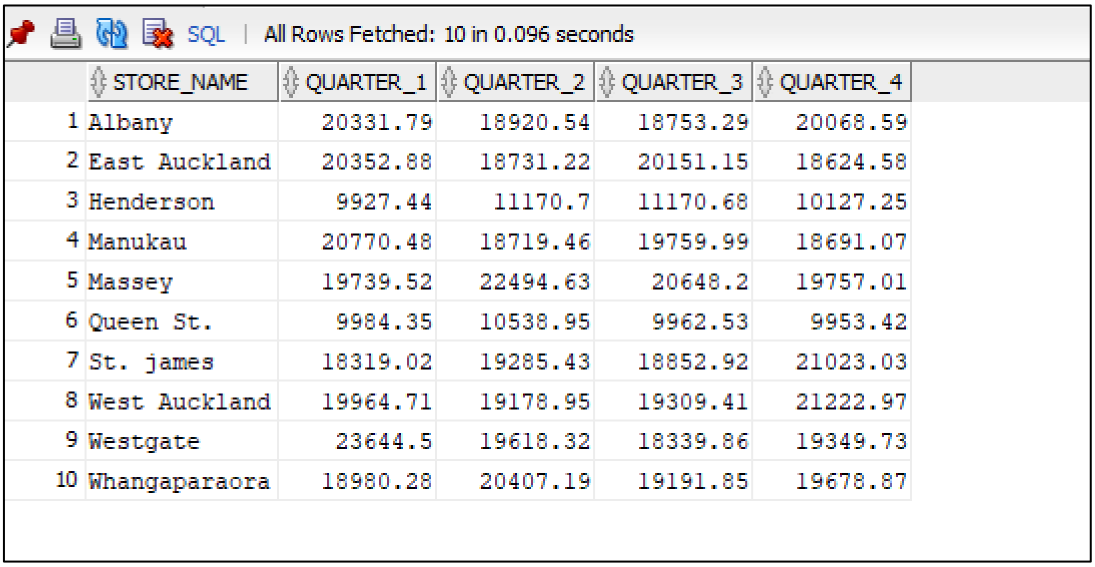
4. Presents the quarterly sales analysis for all stores using drill down query concepts.



| STORE_NAME | Q1_2017 | Q2_2017 | Q3_2017 | Q4_2017 |
|---|---|---|---|---|



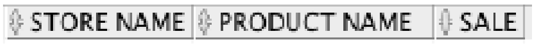
```
--Query4
SELECT store_name, SUM(quarter_1) quarter_1,SUM(quarter_2) quarter_2,SUM(quarter_3) quarter_3,SUM(quarter_4) quarter_4
FROM (SELECT store_name,
      CASE WHEN T_Q=1 THEN total_sale END AS quarter_1,
      CASE WHEN T_Q=2 THEN total_sale END AS quarter_2,
      CASE WHEN T_Q=3 THEN total_sale END AS quarter_3,
      CASE WHEN T_Q=4 THEN total_sale END AS quarter_4
      FROM(SELECT d_stores.store_name, d_time.CAL_QUARTER T_Q,
           SUM(sale) total_sale
           FROM w_facts, d_time, d_stores
           WHERE d_time.time_id = w_facts.time_id
           AND d_stores.store_id = w_facts.store_id
      GROUP BY d_stores.store_name, d_time.CAL_QUARTER
      ORDER BY d_stores.store_name))
      GROUP BY store_name
      ORDER BY store_name;
```



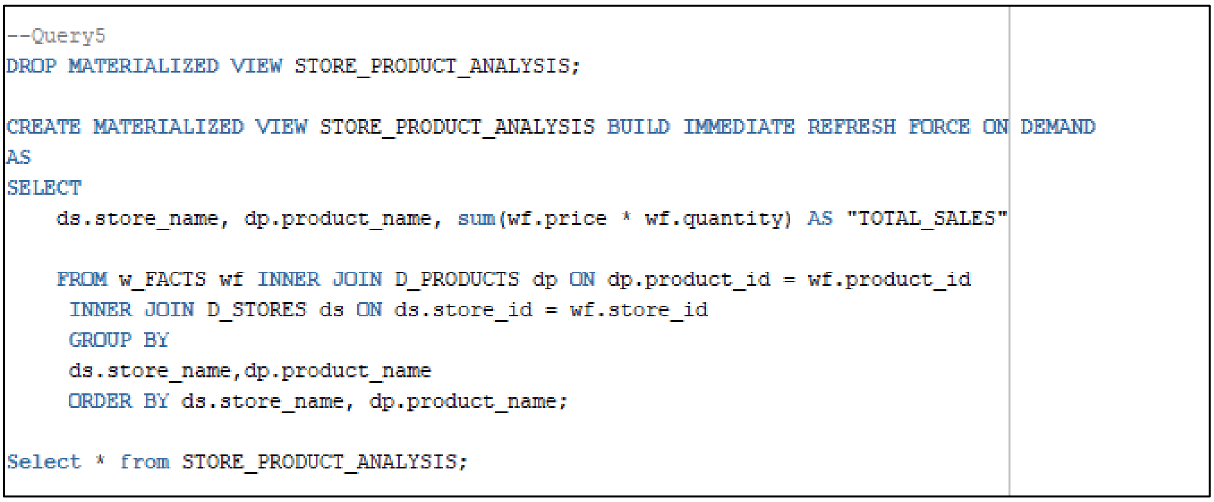
All Rows Fetched: 10 in 0.096 seconds

|  | STORE_NAME | QUARTER_1 | QUARTER_2 | QUARTER_3 | QUARTER_4 |
|---|---|---|---|---|---|
| 1 | Albany | 20331.79 | 18920.54 | 18753.29 | 20068.59 |
| 2 | East Auckland | 20352.88 | 18731.22 | 20151.15 | 18624.58 |
| 3 | Henderson | 9927.44 | 11170.7 | 11170.68 | 10127.25 |
| 4 | Manukau | 20770.48 | 18719.46 | 19759.99 | 18691.07 |
| 5 | Massey | 19739.52 | 22494.63 | 20648.2 | 19757.01 |
| 6 | Queen St. | 9984.35 | 10538.95 | 9962.53 | 9953.42 |
| 7 | St. james | 18319.02 | 19285.43 | 18852.92 | 21023.03 |
| 8 | West Auckland | 19964.71 | 19178.95 | 19309.41 | 21222.97 |
| 9 | Westgate | 23644.5 | 19618.32 | 18339.86 | 19349.73 |
| 10 | Whangaparaora | 18980.28 | 20407.19 | 19191.85 | 19678.87 |

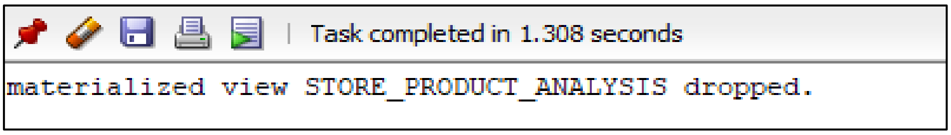
5. Create a materialised view with name “STORE_PRODUCT_ANALYSIS” that presents store and product wise sales. The results should be ordered by store name and then product name.



| STORE NAME | PRODUCT NAME | SALE |
|---|---|---|

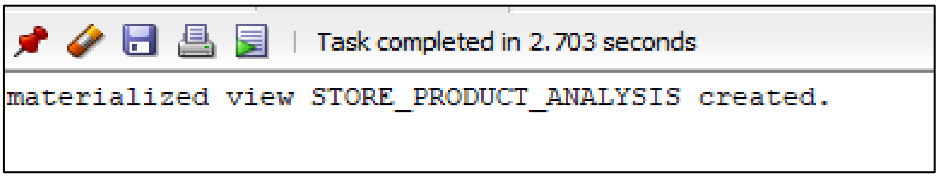


```
--Query5
DROP MATERIALIZED VIEW STORE_PRODUCT_ANALYSIS;

CREATE MATERIALIZED VIEW STORE_PRODUCT_ANALYSIS BUILD IMMEDIATE REFRESH FORCE ON DEMAND
AS
SELECT
    ds.store_name, dp.product_name, sum(wf.price * wf.quantity) AS "TOTAL_SALES"

    FROM w_FACTS wf INNER JOIN D_PRODUCTS dp ON dp.product_id = wf.product_id
     INNER JOIN D_STORES ds ON ds.store_id = wf.store_id
     GROUP BY
     ds.store_name,dp.product_name
     ORDER BY ds.store_name, dp.product_name;

Select * from STORE_PRODUCT_ANALYSIS;
```



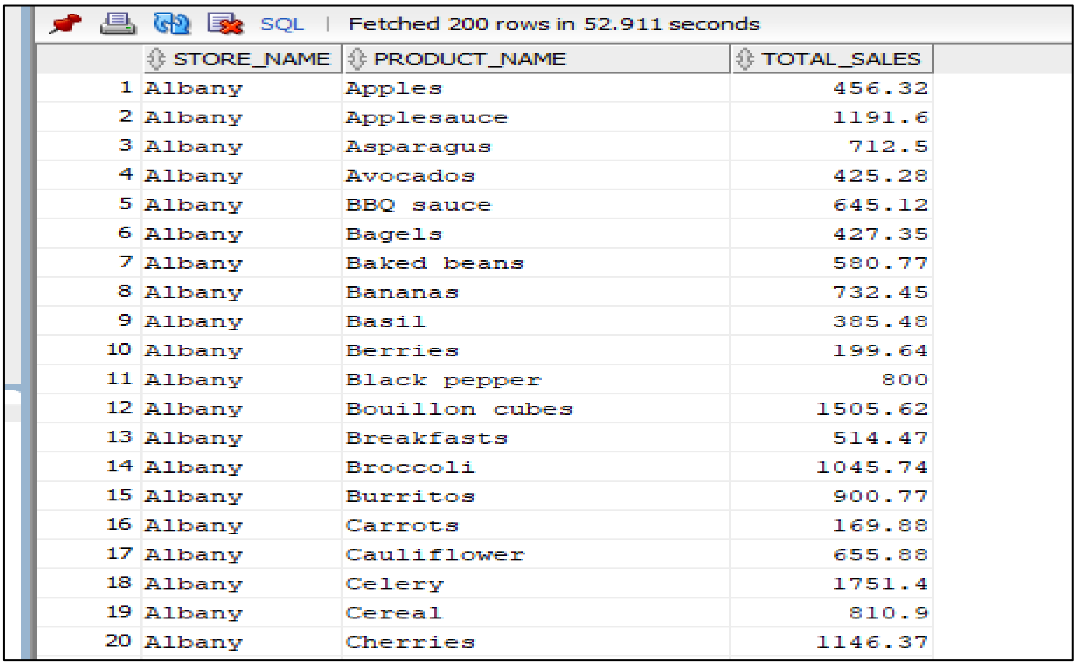
Task completed in 1.308 seconds

```
materialized view STORE_PRODUCT_ANALYSIS dropped.
```



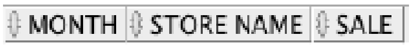
Task completed in 2.703 seconds

```
materialized view STORE_PRODUCT_ANALYSIS created.
```

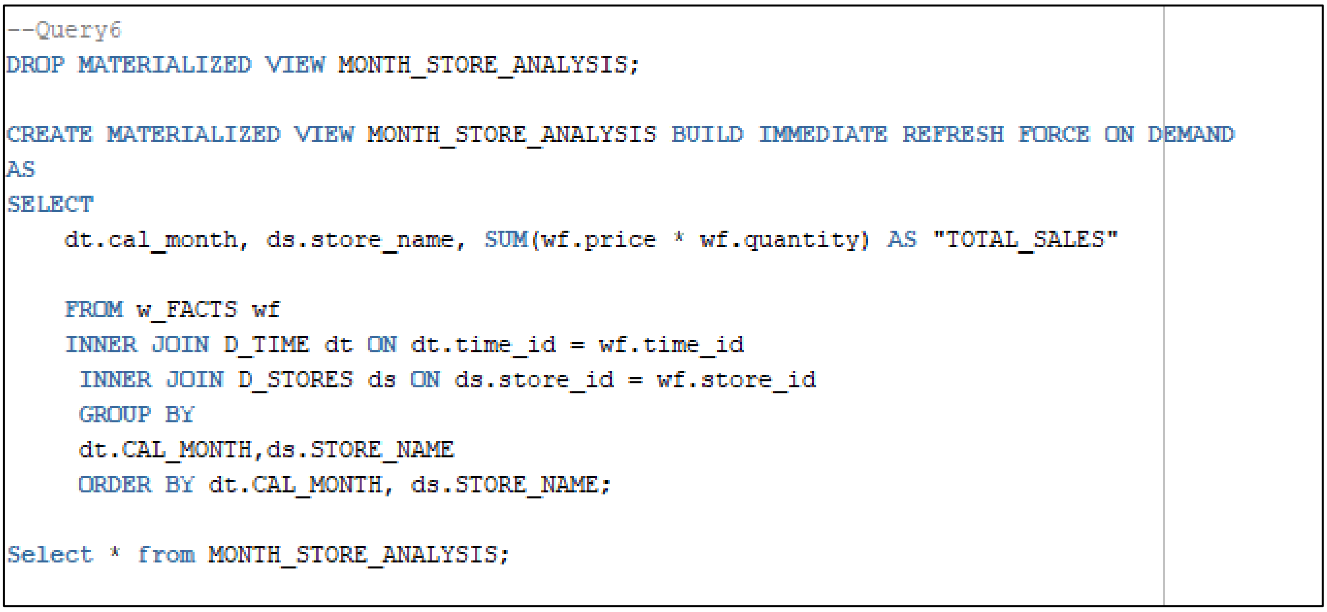


| | STORE_NAME | PRODUCT_NAME | TOTAL_SALES |
|---|---|---|---|
| 1 | Albany | Apples | 456.32 |
| 2 | Albany | Applesauce | 1191.6 |
| 3 | Albany | Asparagus | 712.5 |
| 4 | Albany | Avocados | 425.28 |
| 5 | Albany | BBQ sauce | 645.12 |
| 6 | Albany | Bagels | 427.35 |
| 7 | Albany | Baked beans | 580.77 |
| 8 | Albany | Bananas | 732.45 |
| 9 | Albany | Basil | 385.48 |
| 10 | Albany | Berries | 199.64 |
| 11 | Albany | Black pepper | 800 |
| 12 | Albany | Bouillon cubes | 1505.62 |
| 13 | Albany | Breakfasts | 514.47 |
| 14 | Albany | Broccoli | 1045.74 |
| 15 | Albany | Burritos | 900.77 |
| 16 | Albany | Carrots | 169.88 |
| 17 | Albany | Cauliflower | 655.88 |
| 18 | Albany | Celery | 1751.4 |
| 19 | Albany | Cereal | 810.9 |
| 20 | Albany | Cherries | 1146.37 |

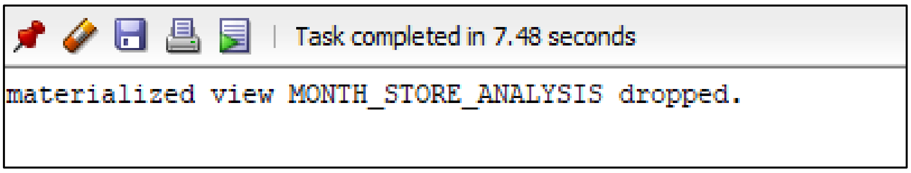
6. Create a materialised view with name “MONTH_STORE_ANALYSIS” that presents month and store wise sales. The results should be ordered by month name and then store name.



| MONTH | STORE NAME | SALE |
|---|---|---|

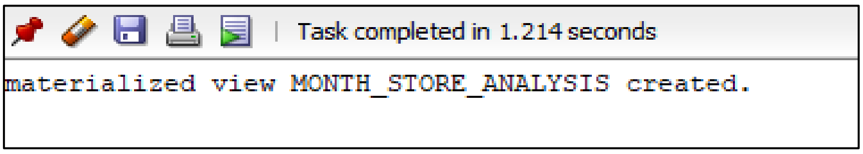


```
--Query6
DROP MATERIALIZED VIEW MONTH_STORE_ANALYSIS;

CREATE MATERIALIZED VIEW MONTH_STORE_ANALYSIS BUILD IMMEDIATE REFRESH FORCE ON DEMAND
AS
SELECT
    dt.cal_month, ds.store_name, SUM(wf.price * wf.quantity) AS "TOTAL_SALES"

    FROM w_FACTS wf
    INNER JOIN D_TIME dt ON dt.time_id = wf.time_id
     INNER JOIN D_STORES ds ON ds.store_id = wf.store_id
     GROUP BY
     dt.CAL_MONTH,ds.STORE_NAME
     ORDER BY dt.CAL_MONTH, ds.STORE_NAME;

Select * from MONTH_STORE_ANALYSIS;
```



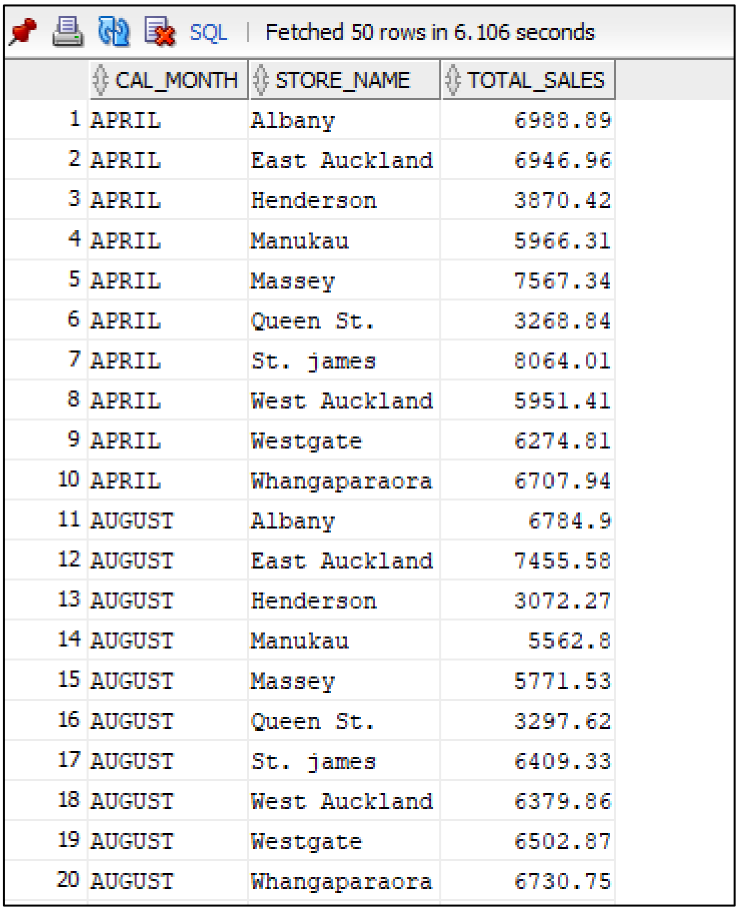
Task completed in 7.48 seconds

```
materialized view MONTH_STORE_ANALYSIS dropped.
```



Task completed in 1.214 seconds

```
materialized view MONTH_STORE_ANALYSIS created.
```



Fetched 50 rows in 6.106 seconds

| | CAL_MONTH | STORE_NAME | TOTAL_SALES |
|---|---|---|---|
| 1 | APRIL | Albany | 6988.89 |
| 2 | APRIL | East Auckland | 6946.96 |
| 3 | APRIL | Henderson | 3870.42 |
| 4 | APRIL | Manukau | 5966.31 |
| 5 | APRIL | Massey | 7567.34 |
| 6 | APRIL | Queen St. | 3268.84 |
| 7 | APRIL | St. james | 8064.01 |
| 8 | APRIL | West Auckland | 5951.41 |
| 9 | APRIL | Westgate | 6274.81 |
| 10 | APRIL | Whangaparaora | 6707.94 |
| 11 | AUGUST | Albany | 6784.9 |
| 12 | AUGUST | East Auckland | 7455.58 |
| 13 | AUGUST | Henderson | 3072.27 |
| 14 | AUGUST | Manukau | 5562.8 |
| 15 | AUGUST | Massey | 5771.53 |
| 16 | AUGUST | Queen St. | 3297.62 |
| 17 | AUGUST | St. james | 6409.33 |
| 18 | AUGUST | West Auckland | 6379.86 |
| 19 | AUGUST | Westgate | 6502.87 |
| 20 | AUGUST | Whangaparaora | 6730.75 |

## Summary

This project of New World was wonderful and taught me to analyse the data by creating the data warehouse in the relational database system. Initially, I learnt fully to write SQL queries to get answers to the question about the customer's product purchasing behaviour. Anyone using standard query language (SQL) can easily access information stored in relational database. For instance, an employee working for an ecommerce, retail or telecom etc, can pull the information to gain insight about customers selling techniques. Further, I learnt to write a PL/SQL block, that includes the procedure to set conditions followed by looping to open the cursors, to fetch the data from data source into data warehouse. PL/SQL is an excellent procedural language that creates and define a PLSQL block. Moreover, I studied how to transfer the customers data from data source into data warehouse by performing index nested loop join (INLJ). INLJ is efficient algorithm to transfer the data from master data. The best part in this project was to write OLAP (online analytical operations) queries, OLAP queries is part of business intelligence that solves the multidimensional analytical queries in relational database system. There are three basic operations in OLAP roll-up, dicing and slicing. It aggregates the data so that it can be evaluated in dimensions. For instance, sales in the office can be accumulated to the respective sales department by using roll up drill down function. Secondly the drill down function can help user to impart details. By implementing slicing or dicing user can look the information from different angles. OLAP can execute critical and complicated queries with minimum processing time. The significant aspects of online analytical processing is the OLAP cube which is also called a hypercube. It contains the numerical details which is segregated by several dimensions. The metadata from the cube is formed from star schema which is also called snowflake schema. The dimensions of the schema are called as label, then the variation of stare schema is defined by the multidimensional structure, it tells the relationship tables and manages the data efficiently. Decision with respective these aggregations to calculate is called view selection. The part of the total dataset can be decided by using view selection. Foremost feature of OLAP because of which the OLAP can perform so well is the aggregation of the data within the dimensions. These each possible dimension is called as granularities. Further there are several forms of OLAP queries those are MOLAP, ROLAP and HOLAP. In short, I learned to create a stare schema and to create a data warehouse in relational database system followed by OLAP queries to get the details of the customers in Oracle SQL developer.