بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ



Data warehousing and Big Data

ASSIGNMENT TWO

**Semester 2, 2018**

**Submitted by**

**Muhammad Kundi**

**Student Id: 15924072**

Table of Contents

[Project Overview 3](#_Toc528096197)

[Data and structure specification 3](#_Toc528096198)

[Pseudocode for INLJ 3](#_Toc528096199)

[OLAP queries 4](#_Toc528096200)

[Output-1 5](#_Toc528096201)

[Output-2 5](#_Toc528096202)

[Output-3 6](#_Toc528096203)

[Output-4 6](#_Toc528096204)

[Output-5 6](#_Toc528096205)

[Output-6 7](#_Toc528096206)

[Summary 8](#_Toc528096207)

## Project Overview

New World is one of the largest supermarket chains in NZ. The stores are placed all over the

country. New World has thousands of customers and thus it's important for the organisation

to analyse the behaviour of their customers. As a result of such analysis ,NewWorld can

optimise their selling strategies e.g. by having relevant promotions on different product.

This task can be achieved by a Data Warehouse where customer’s transactions from different Data Sources(DSs) are stored in the DW on daily basis. The Data Integration in this process can be maintained by ETL (Extraction,Transformation and Loading). But the data generated from DSs is not in the format required by DW. Therefore, data needs to be processed in the transformation layer of ETL using the information from Master Data as shown in Figure-1



Figure 1 An example of enriching transactional data with information from MD

## Data and structure specification

There are 10,000 records in DS. This data will be generated randomly based on 100 products, 50 customers, 10 stores, and one year time period as a date - from 01-Jan-17 to 31-Dec-17. The values for the quantity attribute will be random between 1 and 10. The other two tables named PRODUCTS and CUSTOMERS in MD with 100 and 50 records respectively

## Pseudocode for INLJ

Following steps are taken for this task

1. Making a procedure called INLJ
2. Make new variables to facilitate the curser run in batch form

|  |  |
| --- | --- |
| Serial No | Variable Names |
|  | variable\_total\_records |
|  | variable\_records\_per\_batch |
|  | variable\_total\_batch |
|  | variable\_count\_batch |
|  | variable\_from |
|  | variable\_to |

1. Declaring those variables which are holding values for customer and product from Master Data at the moment when the transaction records are joined.

|  |  |
| --- | --- |
| Serial No | Variable Names |
|  | variable\_customer\_name |
|  | variable\_product\_name |
|  | variable\_supplier\_id |
|  | variable\_supplier\_name |
|  | variable\_price |

1. Making a new variable called : var\_count
2. Get all transactions for transaction table and assigning it to var\_total\_records;
3. Using a while to get records in batches of 100
4. Print the information to user also
5. Declaring a new cursor :cur\_transaction.
6. The For loop is actually reading the data tuple by tuple using the cursor
7. Assign customer name in var\_customer\_name
8. At the same time doing pre existing check of data.
9. Find the product information from Master Data and then set these product name, supplier id, supplier name, and price attributes value in variable\_product\_name, variable\_supplier\_id , variable\_supplier\_name , variable\_price variables respectively.
10. Inserting new record if the customer is not existing in the d\_customers dimention table
11. Adding new record for product if product is not existing in the d\_products dimension table. Similar operation is doen for d\_stores,d\_suppliers,d\_time dimension tables.
12. Fact record is inserted in w\_facts fact table in DW If the same fact record id is not already existing in dimension table. Inserting record on w\_facts table if the same record is not already existing
13. Assigning the variables value to NULL
14. Variable\_count value is reset to 1.
15. The For loop is ended
16. Commit
17. Again we are re assigning the variables to take the next 100 records .
18. And finally ending procedure INLJ

## **OLAP queries**

--------------------------------------------------------------------------------------

--Question No-1. Which product generated maximum sales in September, 2017?

--------------------------------------------------------------------------------------

SELECT d\_products.product\_name,

SUM ( w\_facts.sale ) sale,

DENSE\_RANK ( ) OVER (ORDER BY SUM ( w\_facts.sale ) DESC NULLS LAST)

RANK

FROM w\_facts,

d\_products,

(SELECT d\_time.time\_id

FROM d\_time

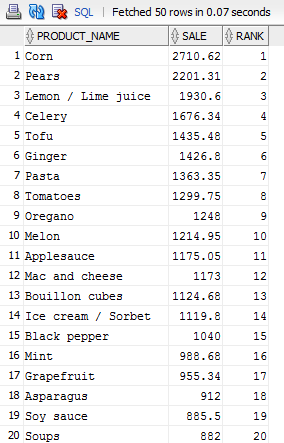
WHERE cal\_month = 'September' AND cal\_year = 2017) v\_dt

WHERE w\_facts.product\_id = d\_products.product\_id

AND w\_facts.time\_id = v\_dt.time\_id

GROUP BY d\_products.product\_name;

### Output-1



--------------------------------------------------------------------------------------

--Question No-2. Determine top three supplier names based on highest sales of their products.

--------------------------------------------------------------------------------------

SELECT \*

FROM (SELECT DENSE\_RANK ( )

OVER (ORDER BY SUM ( w\_facts.sale ) DESC NULLS LAST)

RANK,

d\_suppliers.supplier\_name,

SUM ( w\_facts.sale ) sale

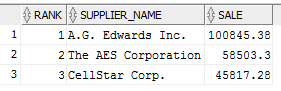
FROM w\_facts, d\_suppliers

WHERE w\_facts.supplier\_id = d\_suppliers.supplier\_id

GROUP BY d\_suppliers.supplier\_name)

WHERE RANK < 4;

### Output-2



--------------------------------------------------------------------------------------

--Question No-3. Determine the top 3 store names who generated highest sales in September, 2017.

--------------------------------------------------------------------------------------

SELECT \*

FROM (SELECT DENSE\_RANK ( )

OVER (ORDER BY SUM ( w\_facts.sale ) DESC NULLS LAST)

RANK,

d\_stores.store\_name,

SUM ( w\_facts.sale ) sale

FROM w\_facts,

d\_stores,

(SELECT d\_time.time\_id

FROM d\_time

WHERE cal\_month = 'September' AND cal\_year = 2017) v\_dt

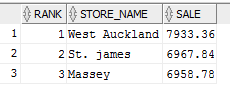
WHERE w\_facts.time\_id = v\_dt.time\_id

AND w\_facts.store\_id = d\_stores.store\_id

GROUP BY store\_name)

WHERE RANK < 4;

### Output-3



--------------------------------------------------------------------------------------

--Question No-4. Presents the quarterly sales analysis for all stores using drill down query concepts.

--------------------------------------------------------------------------------------

SELECT d\_stores.store\_name,

SUM ( DECODE ( d\_time.cal\_quarter, 1, w\_facts.sale, 0 ) ) q1\_2017,

SUM ( DECODE ( d\_time.cal\_quarter, 2, w\_facts.sale, 0 ) ) q2\_2017,

SUM ( DECODE ( d\_time.cal\_quarter, 3, w\_facts.sale, 0 ) ) q3\_2017,

SUM ( DECODE ( d\_time.cal\_quarter, 4, w\_facts.sale, 0 ) ) q4\_2017

FROM w\_facts, d\_stores, d\_time

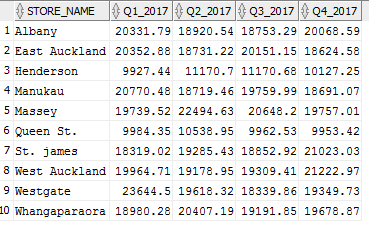
WHERE d\_stores.store\_id = w\_facts.store\_id

AND d\_time.time\_id = w\_facts.time\_id

GROUP BY d\_stores.store\_name

ORDER BY d\_stores.store\_name;

### Output-4



--------------------------------------------------------------------------------------

------ Question No-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

CREATE MATERIALIZED VIEW STORE\_PRODUCT\_ANALYSIS

AS

SELECT s.store\_name "Store Name", dp.product\_name "Product Name", SUM(wf.sale) "Sale"

FROM w\_facts wf

INNER JOIN d\_products dp ON wf.product\_id = dp.product\_id

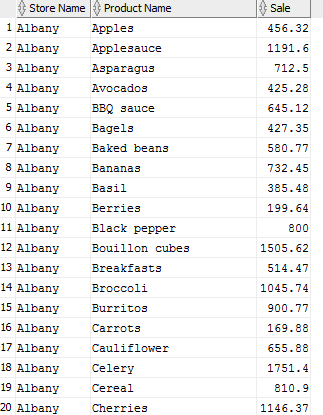
INNER JOIN d\_stores ds ON wf.store\_id = s.store\_id

GROUP BY ROLLUP(ds.store\_name, dp.product\_name)

ORDER BY ds.store\_name, dp.product\_name;

SELECT \* FROM store\_product\_analysis;

### Output-5



--------------------------------------------------------------------------------------

------Question No-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.

--------------------------------------------------------------------------------------

CREATE MATERIALIZED VIEW MONTH\_STORE\_ANALYSIS

AS

SELECT dt.cal\_month "Month", ds.store\_name "Store Name", SUM(wf.sale) "Sale"

FROM w\_facts f

INNER JOIN d\_time dt ON wf.time\_id = dt.time\_id

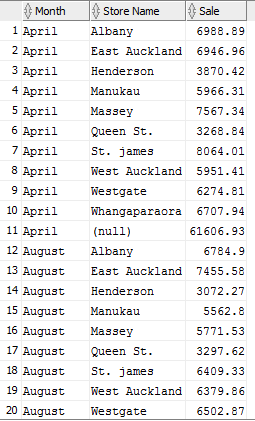
INNER JOIN d\_stores ds ON wf.store\_id = ds.store\_id

GROUP BY ROLLUP(dt.cal\_month, ds.store\_name)

ORDER BY dt.cal\_month, ds.store\_name;

SELECT \* FROM month\_store\_analysis;

## Output-6



## **Summary**

In the above task, we can clearly find out the differences between normal databses and DW.We can perform daily routine operations on the normal data bases which are well known as 'Relational Data Bases'. We also call them as OLTP i.e Online Transaction Processing database because of the continuous transactions involved.But in RDBMS we do not apply analytical methods as it can be very costly for the database operation.

For the analytics, we need DW where we can easily execute OLAP (online analytical processing). Data warehouse is structured to access historical data and make analytics fast .

In Normal databases we use relational models but In this project we have used star schema for data modelling.Star Schema is normally used for DW. This technique is used to map multidimensional decision support into a relational database. The star schema represents aggregated data for specific business activities. Using this schema, one can store aggregated data from multiple sources that will represent different aspects of business operations.

We also noticed that in order to convert the transactional data to the format required by DW we need to enrich data according to DW specifications. The join operator used for this enrichment of data will be Index Nested Loop Join (INLJ).This is a join operator between DS and MD.In INLJ ,DS is scanned in batches of tuples and based on each of the tuple in a batch, the disk-based MD is accessed using an index on the join attribute. This is shown in Figure-2 where tuple from the batch of DS’s is combined with the tuple from MD.



Figure 2 Execution architecture of INLJ

Finally using INLJ after we have transferred all the data in DW using batches of 100 members each, we applied the OLAP queries to analyse the data.