A picture containing text, queen, clipart

Description automatically generated

**RIGA TECHNICAL UNIVERSITY  
Faculty of Computer Science and Information Technology**

**Institute of Applied Computer Systems**

**DSP797 – Data Models in Database Systems**

**Practical Work 1: Redesigning an operational database into a data warehouse**Task Variant: Customer Orders(CO)

**Student Name:** Balasuriyage Aritha Dewnith

KUMARASINGHE **Student Card No:** 203AEB014

2022./2023.

Table of Contents

[Introduction 2](#_Toc117797457)

[1. Design of the warehouse 2](#_Toc117797458)

[Creation of ER diagram for the operational database based 2](#_Toc117797459)

[Creation of ER diagram for data warehouse 3](#_Toc117797460)

[2. Implementation of the warehouse 4](#_Toc117797461)

[Queries that were used to create and populate the data tables. 4](#_Toc117797462)

[Results of data warehouse implementation 7](#_Toc117797463)

[3. Retrieving information using CUBE, ROLLUP, and OVER construction 8](#_Toc117797464)

[Simple rollup query 8](#_Toc117797465)

[Simple cube query 9](#_Toc117797466)

[Simple over query 9](#_Toc117797467)

[Additional queries 10](#_Toc117797468)

[Query 1 10](#_Toc117797469)

[Query 2 10](#_Toc117797470)

[Query 3 11](#_Toc117797471)

[4. Working with Model constructions 12](#_Toc117797472)

[Model #1 12](#_Toc117797473)

[Model #2 13](#_Toc117797474)

[5. Creating materialized views 14](#_Toc117797475)

[6. Conclusions 14](#_Toc117797476)

[Appendix 15](#_Toc117797477)

[Queries. 15](#_Toc117797478)

[References 18](#_Toc117797479)

# Introduction

The goal of this practical work is to take the data tables that are present in a given database schema and restructure the data tables in order to make a data warehouse and implement the warehouse. Then work with this implementation using advanced queries.

Link to script within Live SQL profile:

<https://livesql.oracle.com/apex/livesql/s/n9ccam0jzb5ymlqpd0xowid3m>

Link to google drive containing demonstration video:

<https://drive.google.com/file/d/1T7VuGKW_xen0a0YFgP0599FKO2mxKVaD/view?usp=sharing>

# Design of the warehouse

## Creation of ER diagram for the operational database based

Timeline

Description automatically generated

Diagram 1: ERD for the operational database

The schema is definitely not a star schema but when converting this this to a star schema I would consider the tables **CUSTOMER, ORDER\_ITEMS, PRODUCT, ORDERS, STORE** to be my dimension tables and the materialized view MV\_ORDER\_DETAILS to be my fact table because it has an amalgamation of all the data from all the table within the database except for the PRODUCT table since I consider it more of a sub- table for the order items table.

The next step is take this database schema and denormalized some of the data tables such as the order items and products, or to get rid of some tables entirely so that the final product is star schema. With a star schema being a being only interconnected(dimension tables cannot be interconnected through foreign keys) using a singular fact table(a table that consist of only foreign keys as well a measure(s)).

## Creation of ER diagram for data warehouse

A picture containing diagram

Description automatically generated

Diagram 2: ERD for the database star schema

Dimension tables:

1.**CUSTOMERS**– Table that contains data about customers.

2.**STORE** – Table that contains details about the stores and their locations, although the implementation doesn’t explicitly represent it the physical address for a store is considered to be unique.

3.**PRODUCT** – Contains details about products that are sold by the organization as a whole.

4.**ORDER\_DETAILS** – Contains details about customer orders. Every time a customer places an order an entry is made with this data table containing the a unique order identifier (ORDER\_ID) as well as a timestamp and what the order status is. The timestamp is generated automatically when an insert statement happens, however it does not change every time the order status changes.

Fact table:

Product\_sales – This table is meant to track product sales with relation to orders, which in turn includes which customer made the order and which store did said order go to.

Measures:

Product\_quantity – this is an attribute that is unique to the fact table, and it represents how much of each ordered product present in each order.

Unit price – unit price is a value taken from the product table and represents the price of a product.

# Implementation of the warehouse

## Queries that were used to create and populate the data tables.

DROP TABLE PRODUCT\_SALES;

DROP TABLE CUSTOMERS;

DROP TABLE STORES;

DROP TABLE PRODUCTS;

DROP TABLE ORDER\_DETAILS;

DROP MATERIALIZED VIEW EXTENDED\_ORDER\_DETAILS;

create table CUSTOMERS(

CUSTOMER\_ID number GENERATED BY DEFAULT AS IDENTITY (START WITH 1) NOT NULL,

EMAIL VARCHAR2(150) NOT NULL,

FULL\_NAME VARCHAR2(150) UNIQUE NOT NULL,

CONSTRAINT PK\_CUSTOMER\_ID\_int PRIMARY KEY ( CUSTOMER\_ID ));

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Christopher Reynolds','ChristopherReynolds@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Meagan Alexander','MeaganAlexander@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Robert Phillips','RobertPhillips@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Erin Fernandez','ErinFernandez@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Briana Byrd','BrianaByrd@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Jennifer Love','JenniferLove@gmail.com');

create table PRODUCTS(

PRODUCT\_ID number NOT NULL,

PRODUCT\_NAME VARCHAR2(100) NOT NULL,

UNIT\_PRICE NUMBER(10,2) NOT NULL,

PRODUCT\_DETAILS VARCHAR2(500) NOT NULL,

CONSTRAINT PK\_PRODUCT\_ID\_int PRIMARY KEY ( PRODUCT\_ID ));

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 3615140 ,'Coffe Table',2000.0,'Sleek coffee table made of maple wood.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 4568925,'Dining Table Set',15000,'Everything your life needs including a bulter.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 2544678 ,'Chair',3300,'Its a chair.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 2678011 ,'Sofa',2900,'Blue sofa that will make family time even better.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 4456324 ,'Table',10009,'Table made of gondorian wood.');

create table STORES(

STORE\_ID number NOT NULL,

STORE\_NAME VARCHAR2(100) NOT NULL,

WEB\_ADDRESS VARCHAR2(500) NOT NULL,

PHYSICAL\_ADDRESS VARCHAR2(500) NOT NULL,

CONSTRAINT PK\_STORE\_ID\_int PRIMARY KEY ( STORE\_ID ));

INSERT INTO stores(STORE\_ID,STORE\_NAME,WEB\_ADDRESS,PHYSICAL\_ADDRESS) VALUES (13000,'hardwood furniture shop Kashyyyk', 'www.hfskash.com','Kashyyyk');

INSERT INTO stores(STORE\_ID,STORE\_NAME,WEB\_ADDRESS,PHYSICAL\_ADDRESS) VALUES (10000,'hardwood furniture shop Tatooine', 'www.hfstato.com','Tatooine');

INSERT INTO stores(STORE\_ID,STORE\_NAME,WEB\_ADDRESS,PHYSICAL\_ADDRESS) VALUES (50000,'hardwood furniture shop Coruscant', 'www.hfscor.com','Coruscant');

CREATE TABLE ORDER\_DETAILS(

ORDER\_ID number GENERATED BY DEFAULT AS IDENTITY (START WITH 1) NOT NULL,

ORDER\_DATETIME timestamp with local time zone default on null localtimestamp,

ORDER\_STATUS VARCHAR(10),

CONSTRAINT PK\_ORDER\_ID\_INT PRIMARY KEY ( ORDER\_ID ),

CONSTRAINT CHK\_ORDER\_STATUS CHECK (ORDER\_STATUS IN ('OPEN','PAID','SHIPPED','COMPLETE','CANCELED','REFUNDED'))

);

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('OPEN');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('PAID');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('SHIPPED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('CANCELED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('REFUNDED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('PAID');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('SHIPPED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('SHIPPED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('COMPLETE');

CREATE TABLE PRODUCT\_SALES(

PRODUCT\_ID NUMBER NOT NULL,

PRODUCT\_QUANTITY NUMBER NOT NULL,

UNIT\_PRICE NUMBER(10,2) NOT NULL,

ORDER\_ID NUMBER NOT NULL,

CUSTOMER\_ID NUMBER NOT NULL,

STORE\_ID NUMBER NOT NULL,

CONSTRAINT FK\_ORDER\_PRODUCT\_ID\_int FOREIGN KEY ( PRODUCT\_ID ) REFERENCES products (PRODUCT\_ID),

CONSTRAINT FK\_ORDER\_ID\_int FOREIGN KEY ( ORDER\_ID ) REFERENCES ORDER\_DETAILS (ORDER\_ID),

CONSTRAINT FK\_CUSTOMER\_ID\_int FOREIGN KEY ( CUSTOMER\_ID ) REFERENCES CUSTOMERS (CUSTOMER\_ID),

CONSTRAINT FK\_STORE\_ID\_int FOREIGN KEY ( STORE\_ID ) REFERENCES STORES (STORE\_ID)

);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=1),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=3),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=2),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=5),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=50000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=2544678),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=2),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=5),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=50000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=2544678),3);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=3),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=6),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=4),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=2),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=5),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),2);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=6),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=4),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4456324),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=7),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=2),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4456324),4);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=7),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=2),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=8),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=4),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=8),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=4),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),3);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=2678011),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=9),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=2678011),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=2544678),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=9),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=2544678),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=9),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),2);

/\*The distinct select is to avoid returning duplicate values due to the fact that some orders have multiple products which tends to create duplicate rows if each product type is not(product\_id) CONSIDERED\*/

CREATE MATERIALIZED VIEW EXTENDED\_ORDER\_DETAILS

REFRESH FORCE

ON DEMAND

AS

SELECT

ORDER\_DETAILS.ORDER\_ID,ORDER\_DETAILS.CUSTOMER\_ID,ORDER\_DETAILS.STORE\_ID,ORDER\_DETAILS.PHYSICAL\_ADDRESS STORE\_ADDRESS,ORDER\_DETAILS.ORDER\_STATUS, ORDER\_METRIC\_DETAILS.TOTAL\_PRO\_QUANTITY, ORDER\_METRIC\_DETAILS.TOTAL\_VALUE

FROM

(SELECT DISTINCT PS.ORDER\_ID,PS.CUSTOMER\_ID,PS.STORE\_ID,S.PHYSICAL\_ADDRESS,OD.ORDER\_STATUS FROM PRODUCT\_SALES PS, STORES S, ORDER\_DETAILS OD WHERE PS.ORDER\_ID=OD.ORDER\_ID AND PS.STORE\_ID= S.STORE\_ID ) ORDER\_DETAILS

JOIN

(SELECT PS.ORDER\_ID, SUM(PS.PRODUCT\_QUANTITY) TOTAL\_PRO\_QUANTITY,SUM(PS.PRODUCT\_QUANTITY\*PS.UNIT\_PRICE) TOTAL\_VALUE FROM PRODUCT\_SALES PS GROUP BY PS.ORDER\_ID) ORDER\_METRIC\_DETAILS

ON

ORDER\_DETAILS.ORDER\_ID=ORDER\_METRIC\_DETAILS.ORDER\_ID;

Note: A materialized view called Extended\_Order\_Details was included in this implementation of the schema. This materialized view is meant to be the counterpart to the materialized view Order\_details(MV\_ORDER\_DETAILS) in the operational data base. This materialized view is meant to a basic dash board of order details which gives a summary of the order includes but is not limited to the order id, order status, total product quantity (which is a count of the number of products within the order), total\_value(which is the total order value). This materialized view will also act of a corner stone for this data warehouse since it is included in most of queries for the next parts of this assignment.

## Results of data warehouse implementation

Graphical user interface, text, application

Description automatically generated

Graphical user interface, table

Description automatically generated

# Retrieving information using CUBE, ROLLUP, and OVER construction

The aim of all the queries written below is to evaluate the performance of each of the stores using different KPIs.

## Simple rollup query

SELECT STORE\_ADDRESS,ORDER\_STATUS,COUNT(ORDER\_STATUS),SUM(TOTAL\_VALUE) FROM EXTENDED\_ORDER\_DETAILS GROUP BY ROLLUP (STORE\_ADDRESS, ORDER\_STATUS);

Query description: This query returns the count and the sum of total values for each type of order statuses for each of the stores. Additionally, the rollup clause adds a row that contains the total values of all the orders for each of the stores.

Query results:

Table

Description automatically generated

## Simple cube query

SELECT CUSTOMER\_ID, STORE\_ADDRESS,COUNT(ORDER\_ID) ORDER\_COUNT,SUM(TOTAL\_VALUE) ORDER\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY CUBE(CUSTOMER\_ID,STORE\_ADDRESS);

Query description: This query is responsible for the showing the total order count and order value for different combinations of stores and order statuses.

Results of query:

Graphical user interface

Description automatically generated

## Simple over query

SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS;

Query description: The query above is equivalent to a simple group by query but with added partitioning. Which returns the sum of the values of all the orders that are in a particular staus at each of the stores.

Results of query:

Graphical user interface, text, application

Description automatically generated

## Additional queries

### Query 1

SELECT STORE\_ADDRESS,SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_GAINED FROM (SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('COMPLETE','OPEN','SHIPPED','PAID') GROUP BY ROLLUP(STORE\_ADDRESS);

Query description: This query builds upon the previous over query and sums up all the total value of orders that are considered to be gains or potential gains ('COMPLETE','OPEN','SHIPPED','PAID') as well as the overall gains for all the stores combined.

Results of query:

Graphical user interface, text, application

Description automatically generated

### Query 2

SELECT STORE\_ADDRESS, SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_LOST FROM (SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('CANCELLED','REFUNDED') GROUP BY ROLLUP(STORE\_ADDRESS);

Query description: this a slight alteration of the previous query in order to see the losses of each of the stores as well a from all the stores combined.

Graphical user interface, text, application

Description automatically generated

### Query 3

SELECT GAIN.STORE\_ADDRESS,SUM(COALESCE (GAIN.VALUE\_GAINED,0)) GAIN,SUM(COALESCE(LOSS.VALUE\_LOST,0)) LOSS ,SUM(COALESCE (GAIN.VALUE\_GAINED,0)-COALESCE(LOSS.VALUE\_LOST,0)) DIFF

FROM

(SELECT \* FROM (SELECT STORE\_ADDRESS,SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_GAINED

FROM

(SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('COMPLETE','OPEN','SHIPPED','PAID') GROUP BY STORE\_ADDRESS)) GAIN

FULL OUTER JOIN

(SELECT STORE\_ADDRESS,SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_LOST FROM (SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('CANCELLED','REFUNDED') GROUP BY STORE\_ADDRESS) LOSS

ON GAIN.STORE\_ADDRESS=LOSS.STORE\_ADDRESS GROUP BY ROLLUP(GAIN.STORE\_ADDRESS);

Query description: This query is an amalgamation of the previous quires which shows not only profit and loss but also their difference profits.

Graphical user interface, text, application, email

Description automatically generated

# Working with Model constructions

## Model #1

Query:

SELECT STORE\_ADDRESS,PRODUCT\_NAME,PROFIT\_PER\_PRODUCT,CPROFIT FROM

(SELECT EOD.STORE\_ADDRESS,PRO.PRODUCT\_NAME,SUM(PS.UNIT\_PRICE\*PS.PRODUCT\_QUANTITY) AS PROFIT\_PER\_PRODUCT FROM PRODUCT\_SALES PS, PRODUCTS PRO, EXTENDED\_ORDER\_DETAILS EOD

WHERE PS.PRODUCT\_ID=PRO.PRODUCT\_ID AND PS.ORDER\_ID=EOD.ORDER\_ID GROUP BY(PRO.PRODUCT\_NAME,EOD.STORE\_ADDRESS))

MODEL

DIMENSION BY(STORE\_ADDRESS,PRODUCT\_NAME)

MEASURES (PROFIT\_PER\_PRODUCT, 0 CPROFIT)

RULES(CPROFIT[ANY,ANY] =SUM(PROFIT\_PER\_PRODUCT) OVER (PARTITION BY STORE\_ADDRESS ORDER BY PRODUCT\_NAME))

ORDER BY STORE\_ADDRESS,PRODUCT\_NAME;

Query description: this model query gives the profit per product for each of the stores as well as the cumulative profits form each of the products for each of the stores. The dimension query itself is quite straight forward although the rule is quite unorthodox. To further elaborate on the rule what happens is that simply gives the sum of the profit per product partition by each store.

Query results:

Graphical user interface, text, application

Description automatically generated

## Model #2

Query:

SELECT PRODUCT\_NAME,UNIT\_PRICE FROM PRODUCTS;

SELECT PRODUCT\_NAME,UNIT\_PRICE,PROFIT\_MARGIN,INCOME FROM PRODUCTS

MODEL

DIMENSION BY(PRODUCT\_NAME)

MEASURES (UNIT\_PRICE,0 PROFIT\_MARGIN,0 INCOME)

IGNORE NAV

UNIQUE DIMENSION

RULES UPSERT SEQUENTIAL ORDER(

PROFIT\_MARGIN['Chair']=0.1,

INCOME['Chair']=PROFIT\_MARGIN['Chair']\*UNIT\_PRICE['Chair'],

PROFIT\_MARGIN['Coffe Table']=0.3,

INCOME['Coffe Table']=PROFIT\_MARGIN['Coffe Table']\*UNIT\_PRICE['Coffe Table'],

PROFIT\_MARGIN['Dining Table Set']=0.5,

INCOME['Dining Table Set']=PROFIT\_MARGIN['Dining Table Set']\*UNIT\_PRICE['Dining Table Set'],

PROFIT\_MARGIN['Sofa']=0.8,

INCOME['Sofa']=PROFIT\_MARGIN['Sofa']\*UNIT\_PRICE['Sofa'],

PROFIT\_MARGIN['Table']=0.6,

INCOME['Table']=PROFIT\_MARGIN['Table']\*UNIT\_PRICE['Table']

)

ORDER BY PRODUCT\_NAME;

Query description: This is a more conventional model that simply defines a rules based on a combination of column values. In this case it defines the profit margin as decimal values and what the resulting income for the profit margin is for each of the products.

Graphical user interface, text, application

Description automatically generated

# Creating materialized views

Query:

CREATE MATERIALIZED VIEW MOST\_RECENT\_CUSTOMER\_ORDER

REFRESH FORCE

ON DEMAND

AS

SELECT EOD.CUSTOMER\_ID, MAX(OD.ORDER\_DATETIME) FROM ORDER\_DETAILS OD, EXTENDED\_ORDER\_DETAILS EOD WHERE OD.ORDER\_ID=EOD.ORDER\_ID GROUP BY CUSTOMER\_ID;

Query description: This query creates a materialized view that shows a customer’s most recent order. The parameters of forced refresh is selected because the is I acts as the best of both words for a simple select query like this with unpredictable return values(due to not knowing when a customer might order).

Results from query:

Graphical user interface, text, application

Description automatically generated

# Conclusions

Personally, the designing of the warehouse was the hardest part. Given that I did the entirety of the assignment assuming that the operational database model was what was expected from us. But on the day that the submission was supposed to happen I got to know about the various constraints of the creating a data warehouse which conforms to a star schema. These constrains included the fact that the dimension tables cannot have any foreign keys that interconnect them and the fact tables cannot have a primary key that isn’t a compound key consisting of dimension table foreign keys. Once these new constraints were placed I had to redesign the entire schema and by extension adjust all the quires that I had previously written. Beyond data warehouse design I gained a more in depth understanding of materialized view parameters such as on demand and on commit. I was also introduced to models which are to me the most valuable part of this experience due to their versatility although as of now they are not the easiest to works with.

# Appendix

## Queries.

DROP TABLE PRODUCT\_SALES;

DROP TABLE CUSTOMERS;

DROP TABLE STORES;

DROP TABLE PRODUCTS;

DROP TABLE ORDER\_DETAILS;

DROP MATERIALIZED VIEW EXTENDED\_ORDER\_DETAILS;

create table CUSTOMERS(

CUSTOMER\_ID number GENERATED BY DEFAULT AS IDENTITY (START WITH 1) NOT NULL,

EMAIL VARCHAR2(150) NOT NULL,

FULL\_NAME VARCHAR2(150) UNIQUE NOT NULL,

CONSTRAINT PK\_CUSTOMER\_ID\_int PRIMARY KEY ( CUSTOMER\_ID ));

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Christopher Reynolds','ChristopherReynolds@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Meagan Alexander','MeaganAlexander@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Robert Phillips','RobertPhillips@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Erin Fernandez','ErinFernandez@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Briana Byrd','BrianaByrd@gmail.com');

INSERT INTO customers(FULL\_NAME,EMAIL) VALUES ('Jennifer Love','JenniferLove@gmail.com');

create table PRODUCTS(

PRODUCT\_ID number NOT NULL,

PRODUCT\_NAME VARCHAR2(100) NOT NULL,

UNIT\_PRICE NUMBER(10,2) NOT NULL,

PRODUCT\_DETAILS VARCHAR2(500) NOT NULL,

CONSTRAINT PK\_PRODUCT\_ID\_int PRIMARY KEY ( PRODUCT\_ID ));

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 3615140 ,'Coffe Table',2000.0,'Sleek coffee table made of maple wood.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 4568925,'Dining Table Set',15000,'Everything your life needs including a bulter.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 2544678 ,'Chair',3300,'Its a chair.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 2678011 ,'Sofa',2900,'Blue sofa that will make family time even better.');

INSERT INTO PRODUCTS(PRODUCT\_ID,PRODUCT\_NAME,UNIT\_PRICE,PRODUCT\_DETAILS) VALUES ( 4456324 ,'Table',10009,'Table made of gondorian wood.');

create table STORES(

STORE\_ID number NOT NULL,

STORE\_NAME VARCHAR2(100) NOT NULL,

WEB\_ADDRESS VARCHAR2(500) NOT NULL,

PHYSICAL\_ADDRESS VARCHAR2(500) NOT NULL,

CONSTRAINT PK\_STORE\_ID\_int PRIMARY KEY ( STORE\_ID ));

INSERT INTO stores(STORE\_ID,STORE\_NAME,WEB\_ADDRESS,PHYSICAL\_ADDRESS) VALUES (13000,'hardwood furniture shop Kashyyyk', 'www.hfskash.com','Kashyyyk');

INSERT INTO stores(STORE\_ID,STORE\_NAME,WEB\_ADDRESS,PHYSICAL\_ADDRESS) VALUES (10000,'hardwood furniture shop Tatooine', 'www.hfstato.com','Tatooine');

INSERT INTO stores(STORE\_ID,STORE\_NAME,WEB\_ADDRESS,PHYSICAL\_ADDRESS) VALUES (50000,'hardwood furniture shop Coruscant', 'www.hfscor.com','Coruscant');

CREATE TABLE ORDER\_DETAILS(

ORDER\_ID number GENERATED BY DEFAULT AS IDENTITY (START WITH 1) NOT NULL,

ORDER\_DATETIME timestamp with local time zone default on null localtimestamp,

ORDER\_STATUS VARCHAR(10),

CONSTRAINT PK\_ORDER\_ID\_INT PRIMARY KEY ( ORDER\_ID ),

CONSTRAINT CHK\_ORDER\_STATUS CHECK (ORDER\_STATUS IN ('OPEN','PAID','SHIPPED','COMPLETE','CANCELED','REFUNDED'))

);

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('OPEN');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('PAID');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('SHIPPED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('CANCELED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('REFUNDED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('PAID');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('SHIPPED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('SHIPPED');

INSERT INTO ORDER\_DETAILS(ORDER\_STATUS) VALUES ('COMPLETE');

CREATE TABLE PRODUCT\_SALES(

PRODUCT\_ID NUMBER NOT NULL,

PRODUCT\_QUANTITY NUMBER NOT NULL,

UNIT\_PRICE NUMBER(10,2) NOT NULL,

ORDER\_ID NUMBER NOT NULL,

CUSTOMER\_ID NUMBER NOT NULL,

STORE\_ID NUMBER NOT NULL,

CONSTRAINT FK\_ORDER\_PRODUCT\_ID\_int FOREIGN KEY ( PRODUCT\_ID ) REFERENCES products (PRODUCT\_ID),

CONSTRAINT FK\_ORDER\_ID\_int FOREIGN KEY ( ORDER\_ID ) REFERENCES ORDER\_DETAILS (ORDER\_ID),

CONSTRAINT FK\_CUSTOMER\_ID\_int FOREIGN KEY ( CUSTOMER\_ID ) REFERENCES CUSTOMERS (CUSTOMER\_ID),

CONSTRAINT FK\_STORE\_ID\_int FOREIGN KEY ( STORE\_ID ) REFERENCES STORES (STORE\_ID)

);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=1),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=3),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=2),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=5),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=50000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=2544678),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=2),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=5),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=50000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=2544678),3);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=3),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=6),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=4),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=2),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=5),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),2);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=6),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=4),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4456324),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=7),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=2),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4456324),4);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=7),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=2),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=8),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=4),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=4568925),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=8),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=4),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=13000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=4568925),3);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=2678011),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=9),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=2678011),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=2544678),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=9),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=2544678),1);

INSERT INTO PRODUCT\_SALES(PRODUCT\_ID,ORDER\_ID,CUSTOMER\_ID,STORE\_ID,UNIT\_PRICE,PRODUCT\_QUANTITY) VALUES ((SELECT PRODUCT\_ID FROM PRODUCTS WHERE PRODUCT\_ID=3615140),(SELECT ORDER\_ID FROM ORDER\_DETAILS WHERE ORDER\_ID=9),(SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID=1),(SELECT STORE\_ID FROM STORES WHERE STORE\_ID=10000),(SELECT UNIT\_PRICE FROM PRODUCTS WHERE PRODUCT\_ID=3615140),2);

/\*The distinct select is to avoid returning duplicate values due to the fact that some orders have multiple products which tends to create duplicate rows if each product type is not(product\_id) CONSIDERED\*/

CREATE MATERIALIZED VIEW EXTENDED\_ORDER\_DETAILS

REFRESH FORCE

ON DEMAND

AS

SELECT ORDER\_DETAILS.ORDER\_ID,ORDER\_DETAILS.CUSTOMER\_ID,ORDER\_DETAILS.STORE\_ID,ORDER\_DETAILS.PHYSICAL\_ADDRESS STORE\_ADDRESS,ORDER\_DETAILS.ORDER\_STATUS, ORDER\_METRIC\_DETAILS.TOTAL\_PRO\_QUANTITY, ORDER\_METRIC\_DETAILS.TOTAL\_VALUE

FROM

(SELECT DISTINCT PS.ORDER\_ID,PS.CUSTOMER\_ID,PS.STORE\_ID,S.PHYSICAL\_ADDRESS,OD.ORDER\_STATUS FROM PRODUCT\_SALES PS, STORES S, ORDER\_DETAILS OD WHERE PS.ORDER\_ID=OD.ORDER\_ID AND PS.STORE\_ID= S.STORE\_ID ) ORDER\_DETAILS

JOIN

(SELECT PS.ORDER\_ID, SUM(PS.PRODUCT\_QUANTITY) TOTAL\_PRO\_QUANTITY,SUM(PS.PRODUCT\_QUANTITY\*PS.UNIT\_PRICE) TOTAL\_VALUE FROM PRODUCT\_SALES PS GROUP BY PS.ORDER\_ID) ORDER\_METRIC\_DETAILS

ON

ORDER\_DETAILS.ORDER\_ID=ORDER\_METRIC\_DETAILS.ORDER\_ID;

SELECT \* FROM EXTENDED\_ORDER\_DETAILS;

/\*Materilized view creation\*/

CREATE MATERIALIZED VIEW MOST\_RECENT\_CUSTOMER\_ORDER

REFRESH FORCE

ON DEMAND

AS

SELECT EOD.CUSTOMER\_ID, MAX(OD.ORDER\_DATETIME) FROM ORDER\_DETAILS OD, EXTENDED\_ORDER\_DETAILS EOD WHERE OD.ORDER\_ID=EOD.ORDER\_ID GROUP BY CUSTOMER\_ID;

/\*Model Quries\*/

SELECT PRODUCT\_NAME,UNIT\_PRICE FROM PRODUCTS;

SELECT PRODUCT\_NAME,UNIT\_PRICE,PROFIT\_MARGIN,INCOME FROM PRODUCTS

MODEL

DIMENSION BY(PRODUCT\_NAME)

MEASURES (UNIT\_PRICE,0 PROFIT\_MARGIN,0 INCOME)

IGNORE NAV

UNIQUE DIMENSION

RULES UPSERT SEQUENTIAL ORDER(

PROFIT\_MARGIN['Chair']=0.1,

INCOME['Chair']=PROFIT\_MARGIN['Chair']\*UNIT\_PRICE['Chair'],

PROFIT\_MARGIN['Coffe Table']=0.3,

INCOME['Coffe Table']=PROFIT\_MARGIN['Coffe Table']\*UNIT\_PRICE['Coffe Table'],

PROFIT\_MARGIN['Dining Table Set']=0.5,

INCOME['Dining Table Set']=PROFIT\_MARGIN['Dining Table Set']\*UNIT\_PRICE['Dining Table Set'],

PROFIT\_MARGIN['Sofa']=0.8,

INCOME['Sofa']=PROFIT\_MARGIN['Sofa']\*UNIT\_PRICE['Sofa'],

PROFIT\_MARGIN['Table']=0.6,

INCOME['Table']=PROFIT\_MARGIN['Table']\*UNIT\_PRICE['Table']

)

ORDER BY PRODUCT\_NAME;

/\*Model used to see the cumilative profit from each of the products sold in each of the stores.\*/

SELECT STORE\_ADDRESS,PRODUCT\_NAME,PROFIT\_PER\_PRODUCT,CPROFIT FROM

(SELECT EOD.STORE\_ADDRESS,PRO.PRODUCT\_NAME,SUM(PS.UNIT\_PRICE\*PS.PRODUCT\_QUANTITY) AS PROFIT\_PER\_PRODUCT FROM PRODUCT\_SALES PS, PRODUCTS PRO, EXTENDED\_ORDER\_DETAILS EOD

WHERE PS.PRODUCT\_ID=PRO.PRODUCT\_ID AND PS.ORDER\_ID=EOD.ORDER\_ID GROUP BY(PRO.PRODUCT\_NAME,EOD.STORE\_ADDRESS))

MODEL

DIMENSION BY(STORE\_ADDRESS,PRODUCT\_NAME)

MEASURES (PROFIT\_PER\_PRODUCT, 0 CPROFIT)

RULES(CPROFIT[ANY,ANY] =SUM(PROFIT\_PER\_PRODUCT) OVER (PARTITION BY STORE\_ADDRESS ORDER BY PRODUCT\_NAME))

ORDER BY STORE\_ADDRESS,PRODUCT\_NAME;

/\*ADVANCE QUERIES\*/

/\*QUERY 1.1: SHOW THE VALUE GAINED FROM OPEN, COMPLETE,PAID OR SHIPPED ORDERS IN EACH OF THE STORES,OR TO PUT BLUNTLY SOTRE PROFITS\*/

SELECT STORE\_ADDRESS,SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_GAINED FROM (SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('COMPLETE','OPEN','SHIPPED','PAID') GROUP BY ROLLUP(STORE\_ADDRESS);

/\*QUERY 1.2: Alter query 1.1 to shows the total order values that have been canceled or refunded for each of the stores, or the store losses\*/

SELECT STORE\_ADDRESS,SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_LOST FROM (SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('CANCELLED','REFUNDED') GROUP BY ROLLUP(STORE\_ADDRESS);

/\*QUERY 1.3: Evolution of the queries that were written above to show what the profit vs loss for each of the stores are as well as the overall difference for all the stores combiened. \*/

SELECT GAIN.STORE\_ADDRESS,SUM(COALESCE (GAIN.VALUE\_GAINED,0)) GAIN,SUM(COALESCE(LOSS.VALUE\_LOST,0)) LOSS ,SUM(COALESCE (GAIN.VALUE\_GAINED,0)-COALESCE(LOSS.VALUE\_LOST,0)) DIFF

FROM

(SELECT \* FROM (SELECT STORE\_ADDRESS,SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_GAINED

FROM

(SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('COMPLETE','OPEN','SHIPPED','PAID') GROUP BY STORE\_ADDRESS)) GAIN

FULL OUTER JOIN

(SELECT STORE\_ADDRESS,SUM(EXTENDED\_TOTAL\_VALUE) VALUE\_LOST FROM (SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) EXTENDED\_TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS)

WHERE ORDER\_STATUS IN ('CANCELLED','REFUNDED') GROUP BY STORE\_ADDRESS) LOSS

ON GAIN.STORE\_ADDRESS=LOSS.STORE\_ADDRESS GROUP BY ROLLUP(GAIN.STORE\_ADDRESS);

/\*OVER QUERY 1: This query returns the total value of orders that are currently in a particular status for each of the stores \*/

SELECT STORE\_ADDRESS,ORDER\_STATUS,SUM(SUM(TOTAL\_VALUE)) OVER(PARTITION BY ORDER\_STATUS, STORE\_ADDRESS) TOTAL\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY STORE\_ADDRESS,ORDER\_STATUS;

/\*CUBE QUERY 1: QUERY RETURNS THE TOTAL NUMBER OF ORDERS FOR DIFFERENT COMBINATIONS OF STORES(SOTRE ADDRESSES SINCE IT IS UNIQUE) AND CUSTOMERS\*/

SELECT CUSTOMER\_ID, STORE\_ADDRESS,COUNT(ORDER\_ID) ORDER\_COUNT,SUM(TOTAL\_VALUE) ORDER\_VALUE FROM EXTENDED\_ORDER\_DETAILS GROUP BY CUBE(CUSTOMER\_ID,STORE\_ADDRESS);

/\*ROLLUP QUERY 1:This query shows the count of the each order statuses as well as the cumulative order\_value for each store using rollup\*/

SELECT STORE\_ADDRESS,ORDER\_STATUS,COUNT(ORDER\_STATUS),SUM(TOTAL\_VALUE) FROM EXTENDED\_ORDER\_DETAILS GROUP BY ROLLUP (STORE\_ADDRESS, ORDER\_STATUS);

# References