COMP810 Data Warehousing and Big Data

Lab – Week 5 (Data Warehousing) Due: ---

On successful completion of this paper students will be able to:

(a) Perform OLAP operations with SQL,
(b) Perform drill-down and roll-up operations,
(c) Self-Join Tables by using self-join.

Task 1. In this task we will use tables from the victor's schema. To query these table you'll need to prefix the table name with 'victor.', e.g.,

Select * from victor.TABLENAME

These tables are for demo purposes only and contain a minimal set of columns and constraints.

Table	To query:	Columns
TimeDim	victor.TimeDim	TimeID, TDATE, DAYNAME, MONTH, YEAR
ProductDim	victor.ProductDim	ProductID, PNAME, PCATEGORY, PGROUP
LocationDim	victor.LocationDim	LocID, REGION, COUNTRY
ProducerDim	victor.ProducerDim	CompanyID, CompanyName
SalesFact	Victor.SalesFact	TimeID, PRODUCTID, LOCID, COMPANYID, SALEAMOUNT

Table 1. Tables in 'victor''s schema

Task 1. Familiarize yourself with these tables. Suggested commands:

- Desc
- Select * From

We will work through some commands from the lecture in this lab.
NOTE: You will need to edit the sql queries form lecture slides to match column names and table names.

Continued…

Task 2. Exercises

1. (A) The UNION operator and aggregations. Run file 'Table_Sales0506_example.sql' to create the 'Sales0506' table. **Using UNION and aggregate functions,** write SQL code to display:
   - 'turnover per year and region',
   - 'turnover per year', and
   - 'total turnover' (see Figure below).

Product	Year	Region	Turnover per year and region	Turnover per year	Turnover
Fridge	2005	Northland	135		
		Auckland	120		
				255	
	2006	Northland	140		
		Auckland	135		
				275	
					530

Product	Year	Region	Turnover
Fridge	2005	Northland	135
Fridge	2005	Auckland	120
Fridge	2005		255
Fridge	2006	Northland	140
Fridge	2006	Auckland	135
Fridge	2005		275
Fridge			530

SAMPLE CODE – SOLUTION For 1(A) is in file '1a_w11_sol.sql'. Understand this code and answer 1(B).

1. (B) Now, using the similar command and the tables from the victor's schema, write SWL code to display:
   a. Total turnover,
   b. Turnover per year,
   c. turnover per year and region, and
   d. their union to display this information as follows:

	PCATEGORY	NULL	NULL_1	TURNOVER
1	Fridge	2008	Auckland	80
2	Fridge	2008	Northland	100
3	Fridge	2008	(null)	180
4	Fridge	2009	Auckland	135
5	Fridge	2009	Northland	135
6	Fridge	2009	(null)	270
7	Fridge	2010	Auckland	40
8	Fridge	2010	Northland	40
9	Fridge	2010	(null)	80
10	Fridge	(null)	(null)	530

Here, 'turnover' is given in the column 'saleamount' from the table victor.salesfact.

Continued…

2. Consider the example on Slides 37-40 of lecture — ROLLUP. Run the query to produce results *similar* to that shown below. Experiment by changing the order of the columns in the query:

COMPANYNA	YEAR	SUM
Nokia	2008	1000
Nokia	2009	1500
Nokia	2010	2000
Nokia		4500
Siemens	2008	2000
Siemens	2009	3000
Siemens	2010	3500
Siemens		8500
Motorola	2008	1000
Motorola	2009	1000
Motorola	2010	1500

3. CUBE Operator slides 40-44. Run the query on slide 45 to produce the result *similar* to that shown below, or on the next page.

COMPANYNA	YEAR	TOTAL
		16500
	2008	4000
	2009	5500
	2010	7000
Nokia		4500
Nokia	2008	1000
Nokia	2009	1500
Nokia	2010	2000
Siemens		8500
Siemens	2008	2000
Siemens	2009	3000

COMPANYNA	YEAR	TOTAL
Siemens	2010	3500
Motorola		3500
Motorola	2008	1000
Motorola	2009	1000
Motorola	2010	1500

OUTPUT (next page):

COMPANYNAME	YEAR	TOTAL
(null)	(null	1703(
(null)	200[	25[
(null)	200[	27[
(null)	200[	400(
(null)	200[	550(
(null)	201(	700(
Nokia	(null	466(
Nokia	200[	8(
Nokia	200[	8(
Nokia	200[	100(
Nokia	200[	150(
Nokia	201(	200(
Siemens	(null	873[
Siemens	200[	14(
Siemens	200[	9[
Siemens	200[	200(
Siemens	200[	300(
Siemens	201(	350(
Motorola	(null	363[
Motorola	200[	3[
Motorola	200[	10(
Motorola	200[	100(
Motorola	200[	100(
Motorola	201(	150(

4. **SLIDING WINDOW with MOVING (SLIDING, GLIDING) AVERAGE example**

Run the following sql query. What's the effect of the nested operation Avg(Sum(…)) ? Discuss.

```
SELECT Pr.CompanyName, t.Year, SUM(saleAmount),
    Avg(SUM(saleAmount)) OVER (PARTITION BY Pr.CompanyName
          ORDER BY t.Year
          ROWS 1 PRECEDING) AS Moving_Avg
FROM victor.SalesFact s, victor.ProducerDim pr,
        victor .TimeDim t
WHERE s.companyID = pr.companyID
AND s.timeID = t.timeID
AND t.Year BETWEEN 2008 AND 2010
Group BY Pr.CompanyName, t.Year;
```

OUTPUT:

	COMPANYNAME	YEAR	SUM(SALEAMOUNT)	MOVING_AVG
1	Motorola	2008	600	600
2	Motorola	2009	1135	867.5
3	Motorola	2010	1900	1517.5
4	Nokia	2008	780	780
5	Nokia	2009	1500	1140
6	Nokia	2010	2380	1940
7	Siemens	2008	1300	1300
8	Siemens	2009	3135	2217.5
9	Siemens	2010	4300	3717.5

Continued…

5. **RANK OPERATORS example.**

Run the sql query below. Experiment with ODER BY descending

```
SELECT Pr.CompanyName, t.Year, SUM(saleAmount),
     RANK() OVER (ORDER BY SUM(saleAmount) DESC) AS Rank,
     DENSE_RANK() OVER (ORDER BY SUM(saleAmount) DESC) AS Drank
FROM victor.SalesFact s, victor.ProducerDim pr,
victor.TimeDim t
WHERE s.companyID = pr.companyID
AND s.timeID = t.timeID
AND t.Year BETWEEN 2008 AND 2010
Group BY Pr.CompanyName, t.Year;
```

OUTPUT:

	COMPANYNAME	YEAR	SUM(SALEAMOUNT)	RANK	DRANK
1	Siemens	2010	4300	1	1
2	Siemens	2009	3135	2	2
3	Nokia	2010	2380	3	3
4	Motorola	2010	1900	4	4
5	Nokia	2009	1500	5	5
6	Siemens	2008	1300	6	6
7	Motorola	2009	1135	7	7
8	Nokia	2008	780	8	8
9	Motorola	2008	600	9	9

END OF LAB WEEK 5 DW