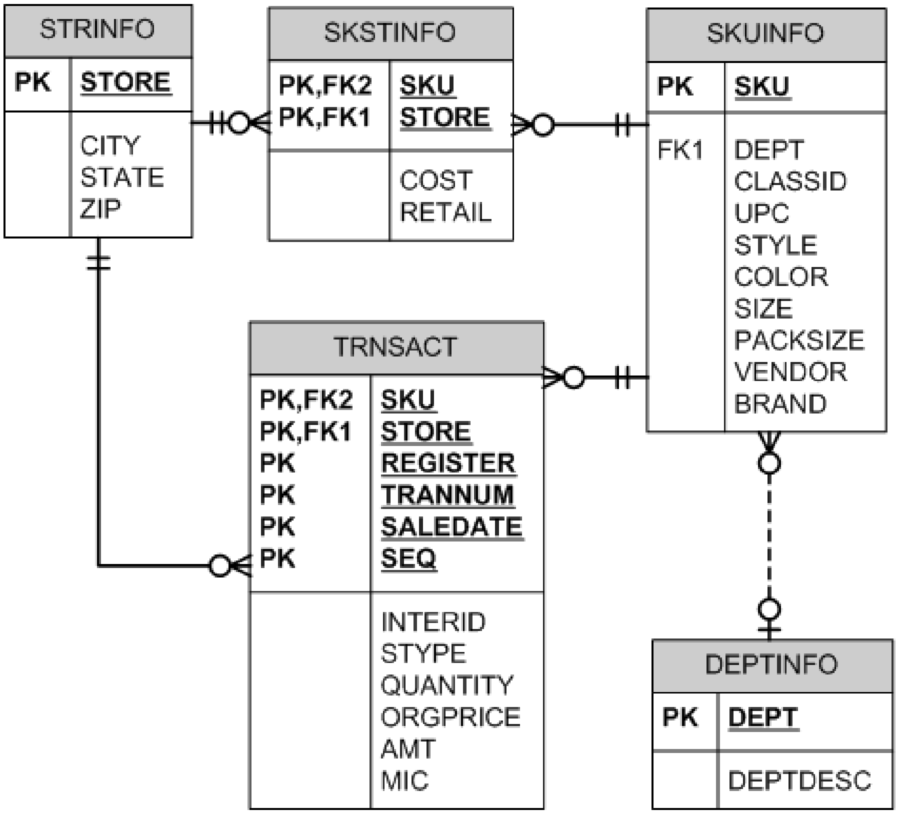
Teradata University Network

A Premier Resource for
Data Warehousing, DSS/BI and Database

UNIVERSITY OF ARKANSAS
SAM M. WALTON
COLLEGE OF BUSINESS

An Introduction to Teradata OLAP Capabilities

The Teradata SQL commands, using Teradata SQL Assistant, used for illustrating Teradata OLAP capabilities are based on the following data structure.



Dillard's Department Stores Sales Transactions

Academic Units, faculty and students, that are members of the Teradata University Network -- free membership at http://www.teradata.com/t/page/137474/index.html, have access to a year of retail sales data from Dillard's. Sensitive data has been removed but the data is very realistic and a rich environment for students and faculty for learning data base, data warehousing and OLAP concepts. Practice with larger more realistic datasets provides enriched learning opportunities not otherwise available. The ua_dillards database consists of 5 tables with more than 120 million rows already populated in the TRNSACT table for your use. The data was provided by Dillard's Department Stores, Inc. and contains the sales transaction data for August 2004 through July 2005 for 453 Dillard's stores.

Use this link http://enterprise.waltoncollege.uark.edu/membership.asp?show=tunregistration **to get a University of Arkansas Account.** From the drop down under Teradata University Network, click Member Information and complete the request forms. You may also refer to the "How to become a TUN member…" documentation up in the University of Arkansas TUN website for more information and Teradata basics. Link specified above.

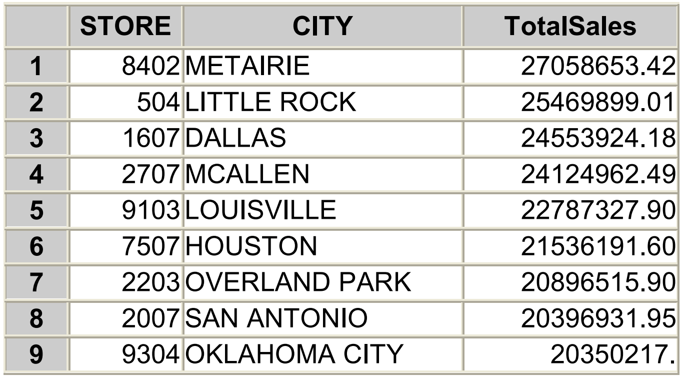
Once you receive your University of Arkansas Teradata account, access will be via remote desktop connection. Remote access documentation is at the following link: http://enterprise.waltoncollege.uark.edu/Remote_Desktop_TUN.pdf

Example 1 - (Examples 1-3 do not use any special OLAP features)

Dillard's management wishes to know the best performing stores (by city) in terms of total sales for the period of August 2004 through July 2005 (note that the Dillard's covers these dates)

```
SELECT s.store, s.city, sum(amt) as TotalSales
FROM  ua_dillards.trnsact t
  INNER JOIN ua_dillards.strinfo s
  ON t.store = s.store

GROUP BY city, s.store
ORDER BY 3 DESC;
```



	STORE	CITY	TotalSales
1	8402	METAIRIE	27058653.42
2	504	LITTLE ROCK	25469899.01
3	1607	DALLAS	24553924.18
4	2707	MCALLEN	24124962.49
5	9103	LOUISVILLE	22787327.90
6	7507	HOUSTON	21536191.60
7	2203	OVERLAND PARK	20896515.90
8	2007	SAN ANTONIO	20396931.95
9	9304	OKLAHOMA CITY	20350217.

Example 2 -- using a WHERE clause to join instead of INNER JOIN

```
SELECT s.store, s.city, sum(amt) as TotalSales
FROM  trnsact t, strinfo s
WHERE t.store = s.store

GROUP BY city, s.store
ORDER BY 3 DESC;
```

Note that in both cases, the database name was used to qualify the table. In Teradata, one can specify the database which then allows creation of the SQL statements without the database name qualification. The following SQL illustrates this capability.

```
DATABASE ua_dillards;
SELECT s.store, s.city, sum(amt) as TotalSales
FROM  trnsact t, strinfo s
WHERE t.store = s.store

GROUP BY city, s.store
ORDER BY 3 DESC;
```

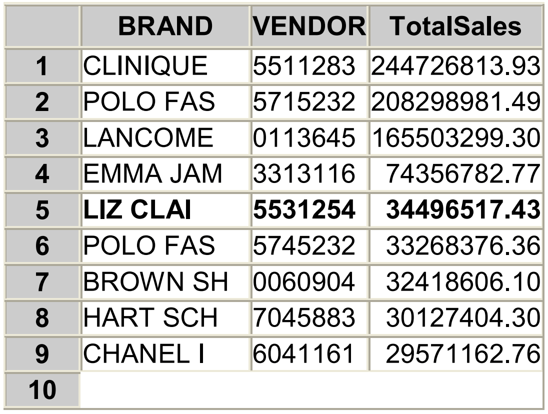
The DATABASE statement specifies the current database and remains in effect until another DATABASE statement is executed.

Example 3

Dillard's management wishes to know the vendors and associated dollar amount of sales for the brand "Liz Clairborne". The results should be from largest sales to smallest sales.

```
SELECT k.brand, k.vendor, sum(amt) as TotalSales
FROM     ua_dillards.trnsact t
   INNER JOIN ua_dillards.skuinfo k ON
   t.sku = k.sku

GROUP BY k.brand, k.vendor
ORDER BY 3 DESC;
```



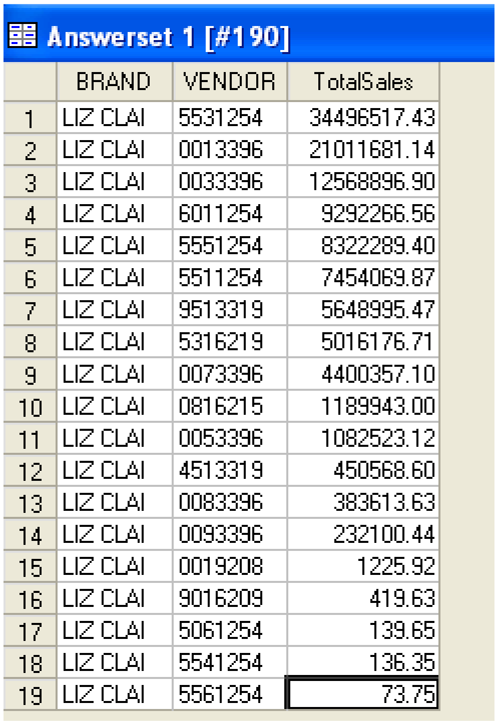
	BRAND	VENDOR	TotalSales
1	CLINIQUE	5511283	244726813.93
2	POLO FAS	5715232	208298981.49
3	LANCOME	0113645	165503299.30
4	EMMA JAM	3313116	74356782.77
5	**LIZ CLAI**	**5531254**	**34496517.43**
6	POLO FAS	5745232	33268376.36
7	BROWN SH	0060904	32418606.10
8	HART SCH	7045883	30127404.30
9	CHANEL I	6041161	29571162.76
10			

OR

```
SELECT k.brand, k.vendor, sum(amt) as TotalSales
FROM     ua_dillards.trnsact t
   INNER JOIN ua_dillards.skuinfo k
   ON     t.sku = k.sku

WHERE k.brand LIKE  '%LIZ CLAI%'      (to retrieve only Liz Claiborne)

GROUP BY k.brand, k.vendor
ORDER BY 3 DESC;
```



	BRAND	VENDOR	TotalSales
1	LIZ CLAI	5531254	34496517.43
2	LIZ CLAI	0013396	21011681.14
3	LIZ CLAI	0033396	12568896.90
4	LIZ CLAI	6011254	9292266.56
5	LIZ CLAI	5551254	8322289.40
6	LIZ CLAI	5511254	7454069.87
7	LIZ CLAI	9513319	5648995.47
8	LIZ CLAI	5316219	5016176.71
9	LIZ CLAI	0073396	4400357.10
10	LIZ CLAI	0816215	1189943.00
11	LIZ CLAI	0053396	1082523.12
12	LIZ CLAI	4513319	450568.60
13	LIZ CLAI	0083396	383613.63
14	LIZ CLAI	0093396	232100.44
15	LIZ CLAI	0019208	1225.92
16	LIZ CLAI	9016209	419.63
17	LIZ CLAI	5061254	139.65
18	LIZ CLAI	5541254	136.35
19	LIZ CLAI	5561254	73.75

OLAP – On-Line Analytical Processing Functions

Using OLAP to Analyze Data

- On-Line Transactions Processing (OLTP) for recorded transactions from terminals
- On-Line Complex Processing (OLCP) for very complex queries
- On-Line Analytical Processing (OLAP) provide the ability to analyze large amounts of data (historical, transactions, trends, …) and provide data mining capabilities

Similar to Functions, but more…

- Like aggregate functions, OLAP operates on groups of rows and permit qualification and filtering.
- Unlike aggregate functions, OLAP return the individual row detail data and not just a final aggregate value.

Basic Teradata OLAP Functions:

CSUM – (Cumulation)
MAVG – (Moving Average)
MSUM – (Moving Sum)
MDIFF – (Moving Differences)
RANK – (Rankings)
QUANTILE – (Quantiles)
SAMPLE – (Sampling)

MLINREG – (Moving Linear Regression)
ROLLUP –subtotaling groups
CUBE – provides data warehouse type capabilities

One purpose of the Teradata OLAP functions is to allow data mining on a database using SQL. Note that

- OLAP functions are similar to aggregate functions
  - Operate on groups of rows (like GROUP BY clause)
  - Allow filtering groups using QUALIFY (like HAVING clause)
- OLAP functions are unlike aggregate functions
  - Return data value for each qualifying now—not group
  - May not be performed within subqueries
- OLAP functions may be performed on
  - Tables
  - Views
  - INSERT/SELECT populations

OLAP Examples*

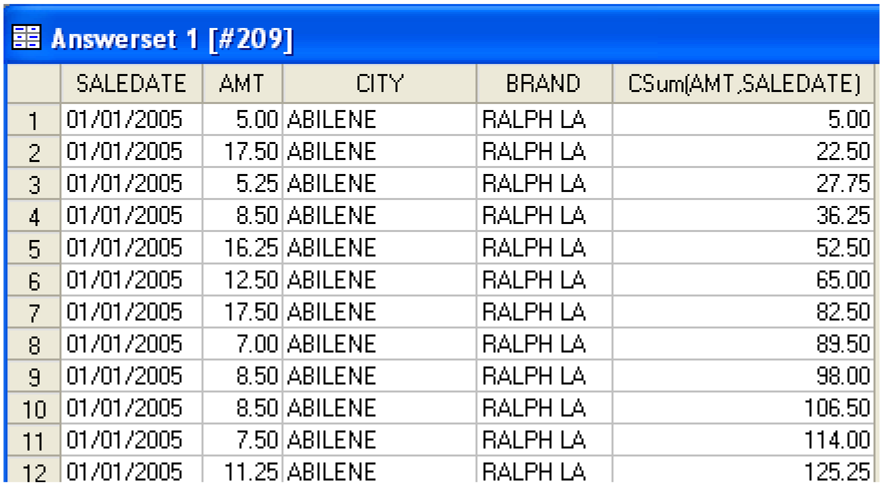
Cumulative SUM - Cumulative Sum of Sales for Store 5203 before January 1

General Form: **CSUM (colname, sort_item1, sort_item2…)**

Example 4

Obtain the sale date, store, department and cumulative sales for Dillard's department stores for January 1, 2005 for department 1704 in Abilene.

```
SELECT saledate, amt, city, brand, CSUM(amt, saledate)
FROM ua_dillards.trnsact t, ua_dillards.skuinfo k, ua_dillards.strinfo s
WHERE t.sku = k.sku
AND t.store = s.store
AND t.saledate BETWEEN '2005-01-01' AND '2005-01-02'
AND k.dept=1704
AND s.city='ABILENE';
```



Answerset 1 [#209]

	SALEDATE	AMT	CITY	BRAND	CSum(AMT,SALEDATE)
1	01/01/2005	5.00	ABILENE	RALPH LA	5.00
2	01/01/2005	17.50	ABILENE	RALPH LA	22.50
3	01/01/2005	5.25	ABILENE	RALPH LA	27.75
4	01/01/2005	8.50	ABILENE	RALPH LA	36.25
5	01/01/2005	16.25	ABILENE	RALPH LA	52.50
6	01/01/2005	12.50	ABILENE	RALPH LA	65.00
7	01/01/2005	17.50	ABILENE	RALPH LA	82.50
8	01/01/2005	7.00	ABILENE	RALPH LA	89.50
9	01/01/2005	8.50	ABILENE	RALPH LA	98.00
10	01/01/2005	8.50	ABILENE	RALPH LA	106.50
11	01/01/2005	7.50	ABILENE	RALPH LA	114.00
12	01/01/2005	11.25	ABILENE	RALPH LA	125.25

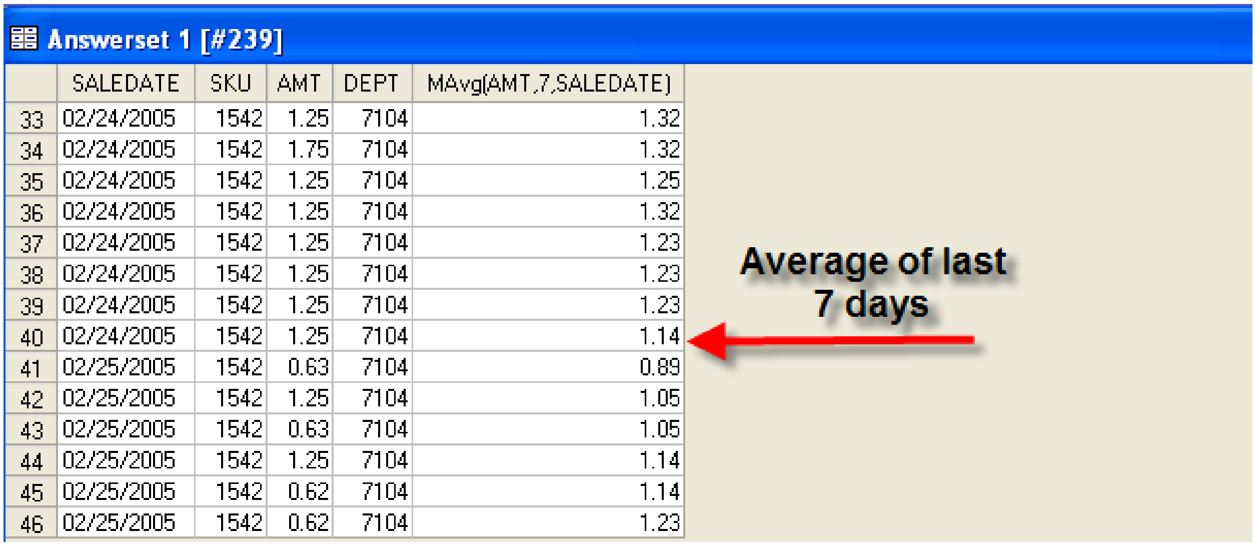
Moving Averages

General form: **MAVG(colname, n, sort_item1, sort_item2, etc)**

Example 5

This simple example shows a moving average on a 7 day window for sales for SKU of ‘0000180’ which is in department ‘1704’ which happens to be a Ralph Lauren product.

```
SELECT saledate, k.sku, amt, dept, MAVG(amt,7,saledate)
FROM ua_dillards.trnsact t, ua_dillards.skuinfo k
WHERE t.sku = k.sku
AND t.sku <2000
AND DEPT=7104
AND EXTRACT(MONTH FROM saledate) IN(1,2)
AND EXTRACT(DAY FROM saledate) IN(21,22,23,24,25)
ORDER BY saledate, t.sku
```



Answerset 1 [#239]

	SALEDATE	SKU	AMT	DEPT	MAvg(AMT,7,SALEDATE)
33	02/24/2005	1542	1.25	7104	1.32
34	02/24/2005	1542	1.75	7104	1.32
35	02/24/2005	1542	1.25	7104	1.25
36	02/24/2005	1542	1.25	7104	1.32
37	02/24/2005	1542	1.25	7104	1.23
38	02/24/2005	1542	1.25	7104	1.23
39	02/24/2005	1542	1.25	7104	1.23
40	02/24/2005	1542	1.25	7104	1.14
41	02/25/2005	1542	0.63	7104	0.89
42	02/25/2005	1542	1.25	7104	1.05
43	02/25/2005	1542	0.63	7104	1.05
44	02/25/2005	1542	1.25	7104	1.14
45	02/25/2005	1542	0.62	7104	1.14
46	02/25/2005	1542	0.62	7104	1.23

Average of last 7 days

Moving sum and moving difference – replace MAVG with MSUM or MDIFF.

Simple and Qualified Rankings

The **Ranking** function permits a column to be ranked, either based on high or low order, against other rows in the answer set. By default, the output will be sorted in descending sequence of the ranking column. The Rank function syntax is:

RANK(colname)

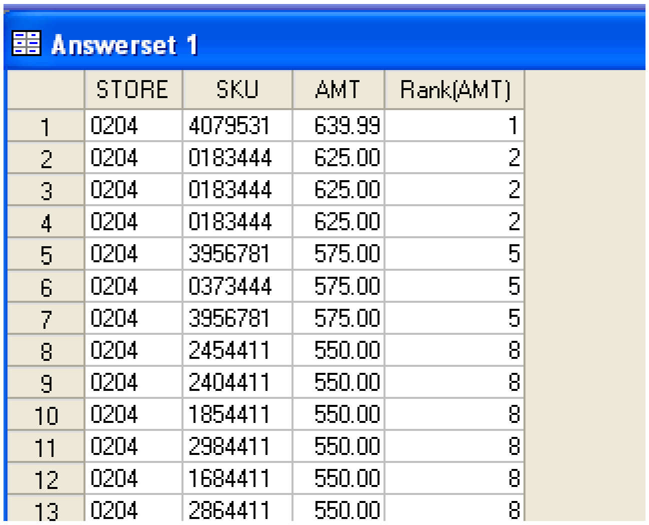
The **QUALIFY** clause allows restriction of the output in the final result.

QUALIFY RANK(amt) <=7

Example 6

Determine the highest selling products for store 204

```
SELECT Store, sku, amt, RANK(amt)
FROM ua_dillards.trnsact
WHERE store= 204
```



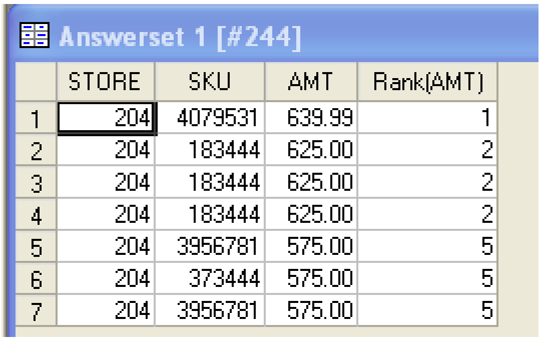
Answerset 1

	STORE	SKU	AMT	Rank(AMT)
1	0204	4079531	639.99	1
2	0204	0183444	625.00	2
3	0204	0183444	625.00	2
4	0204	0183444	625.00	2
5	0204	3956781	575.00	5
6	0204	0373444	575.00	5
7	0204	3956781	575.00	5
8	0204	2454411	550.00	8
9	0204	2404411	550.00	8
10	0204	1854411	550.00	8
11	0204	2984411	550.00	8
12	0204	1684411	550.00	8
13	0204	2864411	550.00	8

Example 7

Get the top five selling products for store 204

```
SELECT Store, sku, amt, RANK(amt)
FROm ua_dillards.trnsact
WHERE store= 204
QUALIFY RANK(amt) <= 5
```



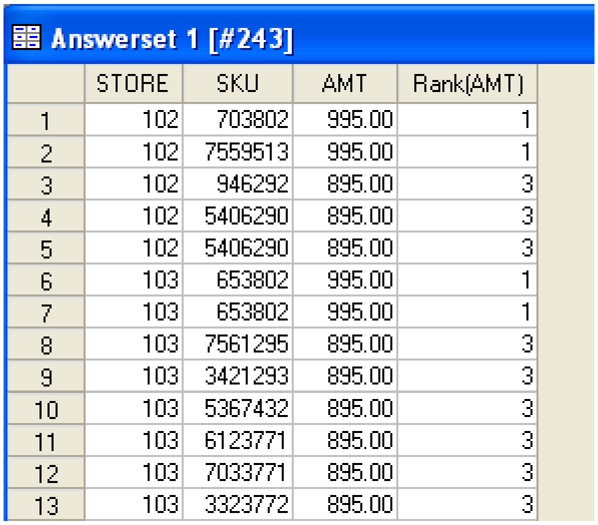
Answerset 1 [#244]

	STORE	SKU	AMT	Rank(AMT)
1	204	4079531	639.99	1
2	204	183444	625.00	2
3	204	183444	625.00	2
4	204	183444	625.00	2
5	204	3956781	575.00	5
6	204	373444	575.00	5
7	204	3956781	575.00	5

Example 8

Get the top three selling products for each store

```
SELECT Store, sku, amt, RANK(amt)
FROm ua_dillards.trnsact
GROUP BY store
QUALIFY RANK(amt) <= 3
```



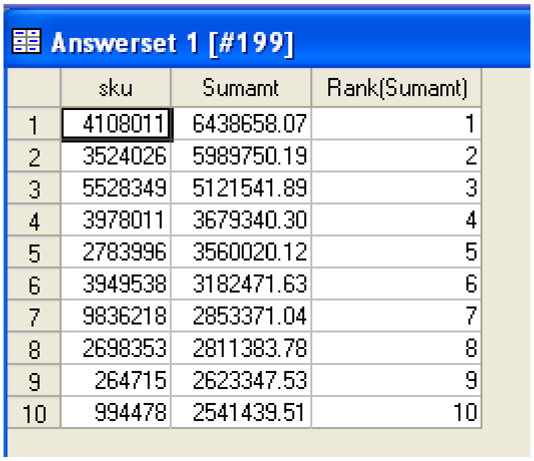
Answerset 1 [#243]

	STORE	SKU	AMT	Rank(AMT)
1	102	703802	995.00	1
2	102	7559513	995.00	1
3	102	946292	895.00	3
4	102	5406290	895.00	3
5	102	5406290	895.00	3
6	103	653802	995.00	1
7	103	653802	995.00	1
8	103	7561295	895.00	3
9	103	3421293	895.00	3
10	103	5367432	895.00	3
11	103	6123771	895.00	3
12	103	7033771	895.00	3
13	103	3323772	895.00	3

Example 8

Get the top 10 selling products across all stores

```
SELECT tt.sku, tt.Sumamt, RANK(tt.Sumamt)
FROM      (SELECT t.sku, SUM(t.amt)
         FROM ua_dillards.trnsact t
         GROUP BY 1)
         AS tt(sku, Sumamt)
   QUALIFY RANK(Sumamt) <= 10
```

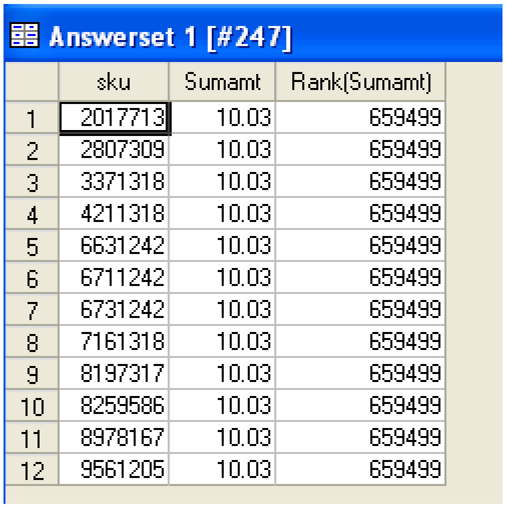


	sku	Sumamt	Rank(Sumamt)
1	4108011	6438658.07	1
2	3524026	5989750.19	2
3	5528349	5121541.89	3
4	3978011	3679340.30	4
5	2783996	3560020.12	5
6	3949538	3182471.63	6
7	9836218	2853371.04	7
8	2698353	2811383.78	8
9	264715	2623347.53	9
10	994478	2541439.51	10

Example 9

Get the ten poorest selling items (greater than $10) across all stores and order by product id -- The only syntax difference between the two queries is the ASC (sort order ascending) on the RANK function.

```
SELECT tt.sku, tt.Sumamt, RANK(tt.Sumamt)
FROM          (SELECT t.sku, SUM(t.amt)
              FROM ua_dillards.trnsact t
              WHERE amt >10
              GROUP BY 1)
              AS tt(sku, Sumamt)
QUALIFY RANK(Sumamt ASC) <= 10
ORDER BY 1
```



	sku	Sumamt	Rank(Sumamt)
1	2017713	10.03	659499
2	2807309	10.03	659499
3	3371318	10.03	659499
4	4211318	10.03	659499
5	6631242	10.03	659499
6	6711242	10.03	659499
7	6731242	10.03	659499
8	7161318	10.03	659499
9	8197317	10.03	659499
10	8259586	10.03	659499
11	8978167	10.03	659499
12	9561205	10.03	659499

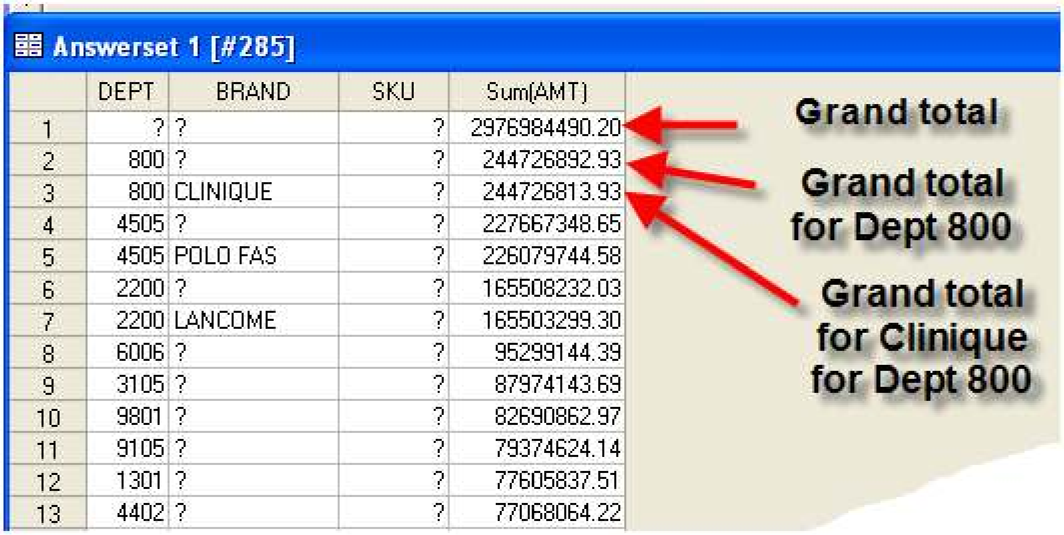
Rollup and Cube OLAP features

General Form: **rollup(col1, col2..)**

Example 10 -- rollup

Dillard's management wishes to know the total sales by department, brand and sku number with the highest sales first. After reviewing this information it will be easy to rerun the query for any combination of single or multiple departments, brands, and skus. One can also limit the output by selecting only a predefined number—SELECT TOP 100, for example.

```
DATABASE ua_dillards;
SELECT k.dept, k.brand, k.sku, sum(t.amt)
FROM  skuinfo k, trnsact t
WHERE k.sku=t.sku
GROUP BY ROLLUP(k.dept, k.brand, k.sku)
ORDER BY 4 DESC, k.dept, k.brand, k.sku
```



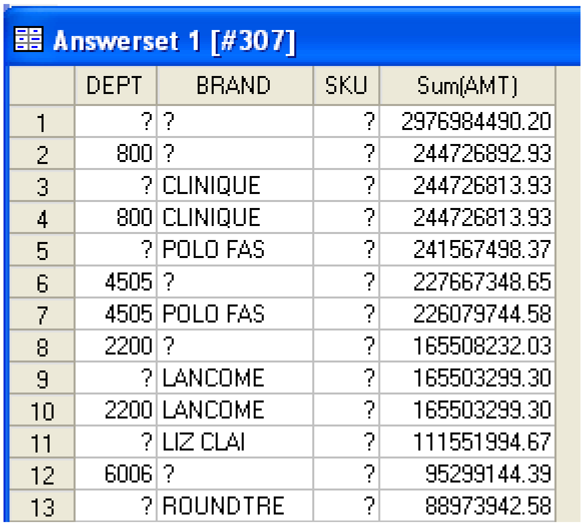
Answerset 1 [#285]

	DEPT	BRAND	SKU	Sum(AMT)
1	?	?	?	2976984490.20
2	800	?	?	244726892.93
3	800	CLINIQUE	?	244726813.93
4	4505	?	?	227667348.65
5	4505	POLO FAS	?	226079744.58
6	2200	?	?	165508232.03
7	2200	LANCOME	?	165503299.30
8	6006	?	?	95299144.39
9	3105	?	?	87974143.69
10	9801	?	?	82690862.97
11	9105	?	?	79374624.14
12	1301	?	?	77605837.51
13	4402	?	?	77068064.22

Example 11 -- Cube

Use the cube capability to obtain the top 100 items in terms of total sales for each department and brand.   The ? entries means the total for that column—thus, row one is the grand total for all items.

```
DATABASE ua_dillards;
SELECT TOP 100 k.dept, k.brand, k.sku, sum(t.amt)
FROM skuinfo k, trnsact t
WHERE k.sku=t.sku
GROUP BY CUBE(k.dept, k.brand, k.sku)
Order by 4 DESC, k.dept, k.brand, k.sku;
```



	DEPT	BRAND	SKU	Sum(AMT)
1	?	?	?	2976984490.20
2	800	?	?	244726892.93
3	?	CLINIQUE	?	244726813.93
4	800	CLINIQUE	?	244726813.93
5	?	POLO FAS	?	241567498.37
6	4505	?	?	227667348.65
7	4505	POLO FAS	?	226079744.58
8	2200	?	?	165508232.03
9	?	LANCOME	?	165503299.30
10	2200	LANCOME	?	165503299.30
11	?	LIZ CLAI	?	111551994.67
12	6006	?	?	95299144.39
13	?	ROUNDTRE	?	88973942.58