**You will get different CPU limits (not spool space) at different times. CPU abort setup based on time.**

**1) 0:00 UTC - 10:59 UTC:  30K CPU maximum per query**

**2) 11:00 UTC - 15:59 UTC:  60K CPU maximum per query**

**3) 16:00 UTC - 23:59 UTC: 125K CPU maximum per query**

**In order to efficiently run your queries you need to optimize you query .**

**The below are recommendations which will help you to review your code for better performance. Incase after reviewing your code based on the below and still running into any challenges we can check and help you on that.**

**1) Leverage partition columns on the table to reduce the dataset.**

**2) Only select the required columns as per requirement to reduce the spool usage.**

**3) Collect statistics on user/sandbox tables for joining columns and where predicates.**

http://www.teradatawiki.net/2014/03/collect-statistics-in-teradata.html

**4) Define proper datatypes for any sandbox tables.**

**5) Avoid using functions like cast, trunc, coalesce in the joining function.**

**6) Always check is there a real need to go for a left join/full outer joins.**

**The below query is running for long and stuck in the merge step due to bad choice of Primary Index.**

**Please create the table as MULTISET table since you are already using group by in the query which eliminates duplicates and also please change the PI for better distribution of records.**

**Choosing better Primary Index**

**Make sure you define a Proper Primary index on your sandbox table. To find better Primary Index make use of below query which gives you the data distribution details across all AMP. Make sure data is distributed evenly across all AMP.**

**SEL HASHAMP(HASHBUCKET(HASHROW(column1))) as AMP# , COUNT(\*) Record\_count FROM databasename .tablename GROUP BY 1;**

**(or)**

**SEL HASHAMP(HASHBUCKET(HASHROW( column1, coulmn2 ,....))) as AMP# , COUNT(\*) FROM databasename .tablename GROUP BY 1;**

**Record\_count: Total number of records distributed in each AMP.**

**Note: Try to define Primary Index on joining column if its doesn't hold more duplicated records**

<http://developer.teradata.com/database/articles/what-you-need-to-know-before-creating-a-table-in-teradata>

**To find the last alter timestamps of table**

sel LastAlterTimeStamp from dbc.tables where databasename ='sandbox' and tablename = 'viv\_cc\_features';

**Grant statement**:

GRANT EXECUTE, SELECT, STATISTICS, SHOW ON "DBC" TO "DBADMIN" WITH GRANT OPTION;

**CASE statment:**

1. SELECT column1,
2. CASE column2
3. WHEN value1 THEN result1
4. WHEN value2 THEN result2
5. END
6. FROM table

sel month\_, case when char\_length(cast(month\_ as varchar(2))) = 1 then '0'||cast(month\_ as varchar(2)) else cast(month\_ as varchar(2)) end from user\_gp.user\_billing\_records sample 10;

sel month\_,year\_,

(case

when char\_length(cast(month\_ as varchar(2))) = 1

then '0'||cast(month\_ as varchar(2))

else cast(month\_ as varchar(2)) end) ||'-'||cast(year\_ as varchar(4)),

cast(((case

when char\_length(cast(month\_ as varchar(2))) = 1

then '0'||cast(month\_ as varchar(2))

else cast(month\_ as varchar(2)) end) ||'-'||cast(year\_ as varchar(4))) as date format 'mm-yyyy')

from user\_gp.user\_billing\_records sample 10;

**Minus and Union**

sel count(distinct tokenized\_pan) from

(((sel distinct tokenized\_pan from user\_gp.user\_billing\_records where year\_ > 2016)

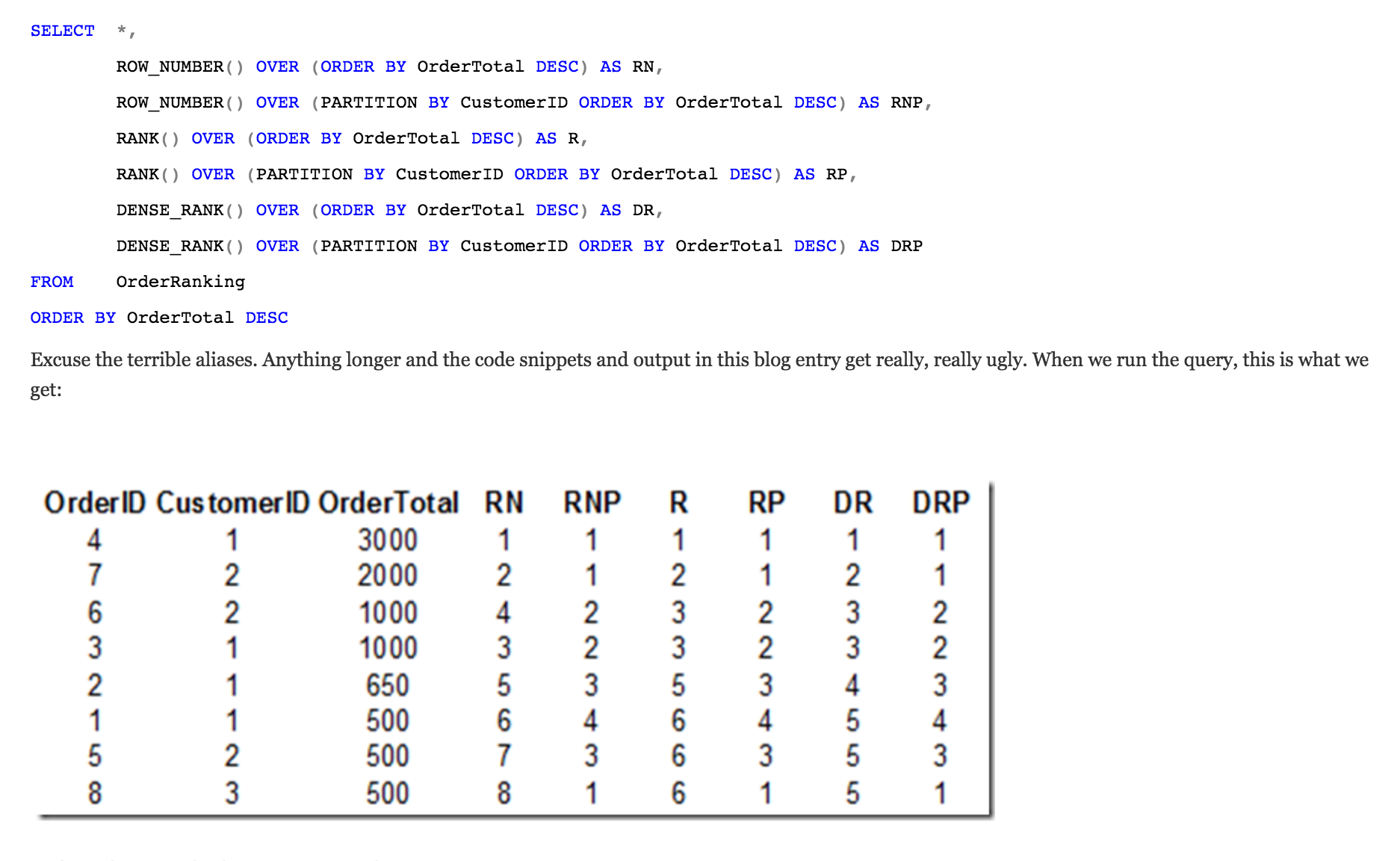
union (sel distinct tokenized\_pan from user\_gp.user\_billing\_records where year\_ = 2016 and month\_ >= 9))

minus

((sel distinct tokenized\_pan from sandbox.viv\_cc\_deletion\_base where expiry\_year > 2016)

union

(sel distinct tokenized\_pan from sandbox.viv\_cc\_deletion\_base where expiry\_year = 2016 and expiry\_month >= 9))) as c;



* ROW\_NUMBER() assigns sequential numbers to each partition in a result set (an unpartitioned result set simply has a single partition), based upon the order of the results as specified in the ORDER BY clause. If you look carefully, you’ll see that the values in column RN are based upon a simple sort of TotalDue, while the values in Column RNP (Row\_Number partitioned) are first partitioned or “grouped” by CustomerID, and then numbered by TotalDue, with the row number resetting on change of customer.
* Contrary to popular belief, RANK() does not sort rows based upon how bad they smell. RANK() does much the same thing as ROW\_NUMBER(), only it acknowledges ties in the columns specified in the ORDER BY clause, and assigns them the same rank. Where a tie occurs (as was the case for orders 6/3, and 1/5/8), the numbers that would otherwise have been “used up” are skipped, and numbering resumes at the next available number. As you can see, RANK() leaves a gap whenever there is a tie.
* DENSE\_RANK() doesn’t like gaps. It’s more of an Abercrombie & Fitch kind of function (ba-dum-ching!). Ohhhhh…that was terrible. My sense of humour may give me up for lent. You might follow it. Anyway….DENSE\_RANK() “fills in the gaps”. It starts from the next number after a tie occurs, so instead of 1, 2, 3, 3, 5 you get 1, 2, 3, 3, 4.

**Evaluation Order of WHERE, GROUP BY, and QUALIFY Clauses**

When the WHERE, GROUP BY, and QUALIFY clauses are used together in a SELECT request, the order of evaluation is as follows:

**1**WHERE clause

**2**GROUP BY clause

**3**QUALIFY clause

In detail, the steps are these.

**1**Teradata Database evaluates the WHERE clause conditions on the FROM clause tables.

**2**The system groups the resulting rows using the GROUP BY columns.

**3**Teradata Database evaluates the ordered analytical functions on the grouped table.

**4**The system applies the QUALIFY clause to the resulting set.

Teradata Database-specific functions such as CSUM and MAVG that are invoked in both the select list and in the search condition of the QUALIFY clause are computed on the grouped rows without eliminating any rows and then the system applies the search condition of the QUALIFY clause.

**Aggregation without Group by : using Over**

100 \* reasonCount/sum(reasonCount) over() as percentage

<http://stackoverflow.com/questions/6218902/the-sql-over-clause-when-and-why-is-it-useful>

**Stratified sampling in Teradata:**

<http://teradatafaqs.blogspot.in/2013/05/teradata-sampling-stratified-sampling.html>

Select employeeid,departmentno,sampleid

From employee2

SAMPLE

WHEN departmentno=100 then .40 --ceil(0.4\*7), 7 is the total rows in this departmentno

WHEN departmentno=200 then .25

WHEN departmentno=300 then .05

WHEN departmentno=400 then .35

END ;

|  |  |  |
| --- | --- | --- |
| **Employeeid** | **DepartmentNo** | **SampleId** |
| 5 | 100 | 1 |
| 1 | 100 | 1 |
| 9 | 100 | 1 |
| 140 | 200 | 2 |
| 139 | 200 | 2 |
| 144 | 400 | 4 |
| 134 | 400 | 4 |
| 137 | 400 | 4 |

select week\_of\_year,calendar\_date,

EXTRACT(YEAR from calendar\_date) as calendarYear,

cast(week\_of\_year as varchar(5))|| '-'||CAST(EXTRACT(YEAR from calendar\_date) as varchar(5)) as weekYearLabel

from sys\_calendar.calendar

where calendar\_date > '2015-08-31'

and calendar\_date <= current\_date