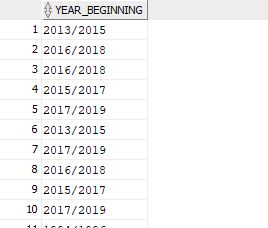
**DBW624 Assignment 5 – Rohil Khakhar – 109270173**

**Here are the 5 performance enhancements I’ve decided to implement inside my warehouse to help refine it**

1. **Column Split** 
   1. Split the year range column for readability. This will make it easier to query and help in the partitioning process. Segregating such columns helps refine the warehouse to an extent.
   2. In my case we’ll be implementing this in LIFE\_SPAN

**select year\_beginning from life\_span; - initial orientation (we will be splitting this)**

****

**alter table life\_span rename column year\_beginning to year\_start; -** Successfully altered

Now add a new column – year end

**alter table life\_span add year\_end varchar(20);**

Split the table using **select** to get an idea of what’s expected

**select substr(year\_start, 0, 4) as left\_date,**

**substr(year\_start, 6, 4) as right\_date from life\_span;**

Now we use substr() to do the required extraction and update the right sided date to the new column and the left sided date to the original column (year\_beginning)

**update life\_span set year\_end = substr(year\_start, 6, 4);**

**update life\_span set year\_start = substr(year\_start, 0, 4);**

Now the columns will be much more readable and easier to query

1. **Indexing**
   1. Creating indexes will help me get quicker access to the table rows.
   2. This will ultimately help with the warehouse performance.

**create index product\_idx on sales\_fact(product\_key);**

**create index promotion\_idx on sales\_fact(promotion\_key);**

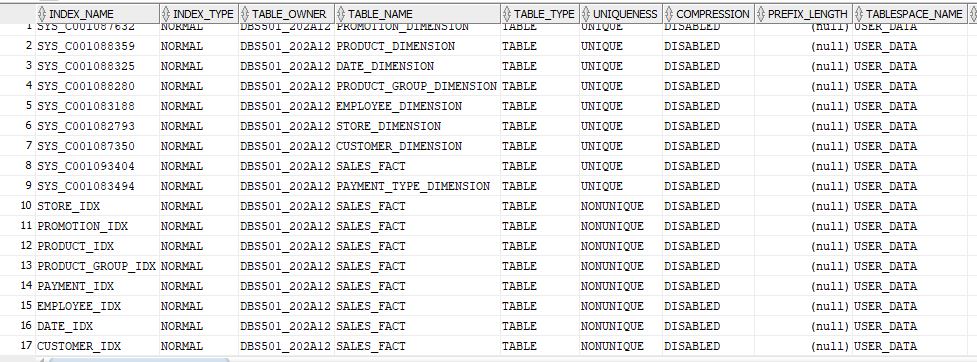
**create index date\_idx on sales\_fact(date\_key);**

**create index payment\_idx on sales\_fact(payment\_type\_key);**

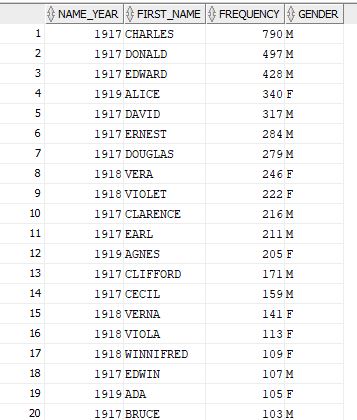
**create index store\_idx on sales\_fact(store\_key);**

**create index employee\_idx on sales\_fact(employee\_key);**

**create index customer\_idx on sales\_fact(customer\_key);**

**create index product\_group\_idx on sales\_fact(product\_group\_key);**

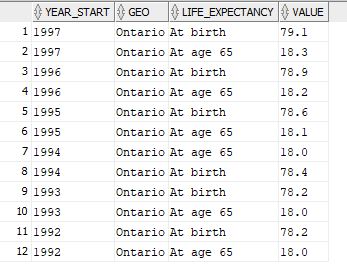
**select \* from user\_indexes – gives us the following output**

1. **Summary Tables** 
   1. Summary tables are useful for a handy reference to obtaining statistics about data in a specific table. Generally applicable for tables with quite a lot of rows.
   2. Its biggest benefit is that it helps in the readability of data and something we can consult at any point.
   3. **In my case, these will be conducted for tables with quite a lot of data.. aka staging tables.**

**create table names\_summary as (select first\_name, gender, frequency from names);**

**select \* from names summary**

**order by frequency desc;**

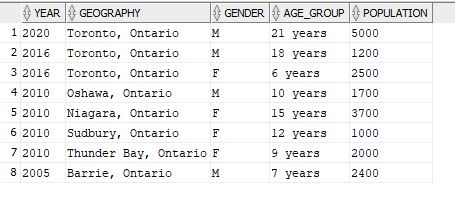
****Repeat the same procedure for population and lifespan. Be intelligent about the grouping choices

**create table life\_span\_summary as (select year\_start, geo, life\_expectancy, value from life\_span**

**where gender = 'B' and geo = 'Ontario') –** specific to Ontario since that’s where the warehouse is based

**select \* from life\_span\_summary**

**order by year\_start desc;**

**create table ontario\_population as (select \* from city\_population**

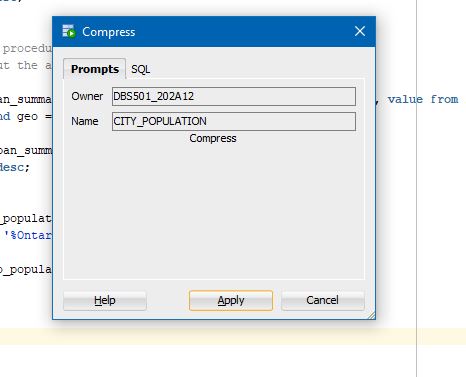
**where geography like '%Ontario%');** - specific to Ontario

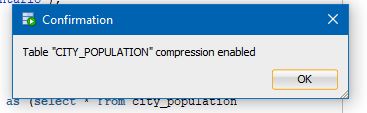
**select \* from ontario\_population**

**order by year desc;**

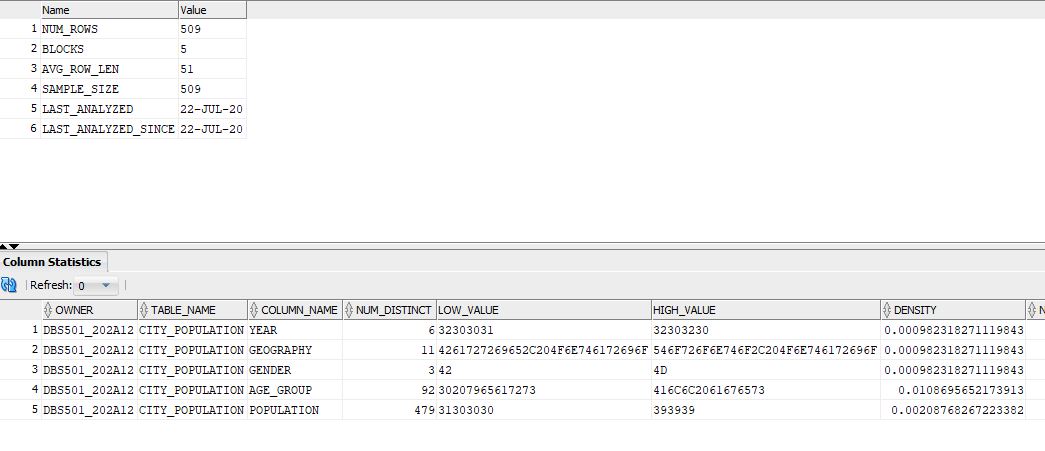
1. **Compression**
   1. Significantly takes the load off the warehouse and helps speed up querying and database performance.
   2. Particularly useful in my case since some of my staging tables have quite a lot of data
   3. From a business standpoint, this will also help reduce storage costs in the future

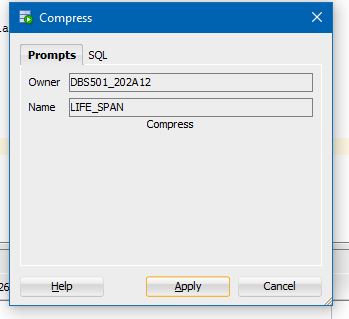
**Steps for performing compression for staging tables**

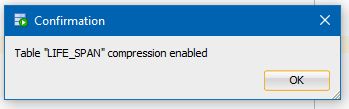
1. **Right click on the desired table**
2. **Select the “Storage Option”**
3. **Inside “Storage”, look for the “Compress” Option**
4. **A pop up box will be generated asking you to confirm the compression**
5. **An alert box showing a success message will show up**

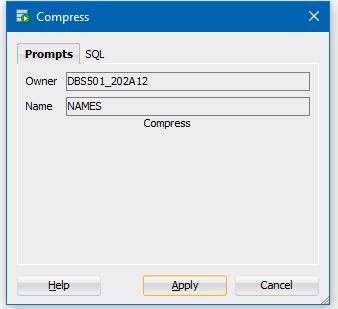
****

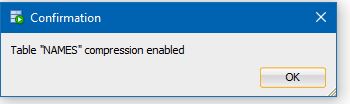
**Statistics can be subsequently seen by right clicking on the table ad hovering over the statistics tab.**

****

****

****

****

****

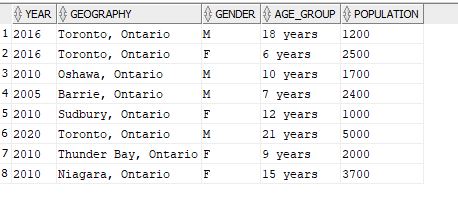
1. **Clean up unnecessary data**
   1. This is another enhancement that can take place to help preserve storage and only keep the data that we need
   2. Ideally, we would have performed **ARCHIVING** by creating a new database but my tablespace **USER\_DATA** unfortunately does not have that much space in its quota.
   3. As always, we will be dealing with staging tables since they’re taking up most of the data

As far as the CITY\_POPULATION is concerned, we are only concerned with data related to Ontario since that’s where our warehouse will operate.

Hence we will clean up the table and get rid of the unrelated provinces

**delete from city\_population where geography not like '%Ontario%';**

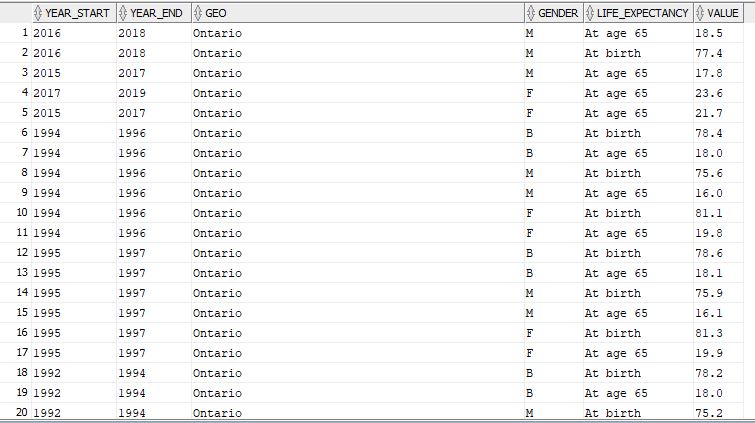
**select \* from city\_population;**



For LIFE\_SPAN, we will incorporate the same approach where we will only be concerned with Ontario

**delete from life\_span where geo not like '%Ontario%';**

**select \* from life\_span;**

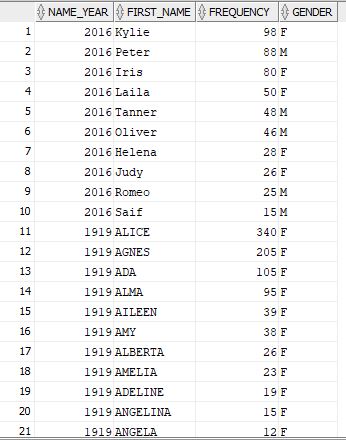
****

As for Names, we wouldn’t want to be concerned with those names which are not that frequently used. So we can get rid of those names which have a frequency of less than 10.

**delete from names where frequency < 10;**

**select \* from names**

**order by name\_year desc, frequency desc;**

****