**DBW624 Assignment 3 – Rohil Khakhar – 109270173**

**Step 1 – EXTRACT**

1. **Create staging tables for each of the reference tables are to be dealt with**

**Names (Female and Male Baby Names)**

create table male\_names(

year\_ number not null,

name\_ varchar2(30) not null,

frequency number not null

);

create table female\_names(

year\_ number not null,

name\_ varchar2(30) not null,

frequency number not null

);

**Life Span**

create table life\_span\_stage(

ref\_date varchar2(20) not null,

geo varchar2(50) not null,

dguid varchar2(50) not null,

sex varchar2(30) not null,

life\_expectancy varchar2(20) not null,

uom varchar2(30) not null,

uom\_id varchar2(30) not null,

scalar\_factor varchar2(25) not null,

scalar\_id varchar2(30) not null,

vector varchar2(40) not null,

coordinates varchar2(30) not null,

value varchar2(30) not null,

status varchar2(20) not null,

symbol varchar2(20) not null,

terminated\_ varchar2(20),

decimals varchar2(20)

);

**Population**

create table city\_population\_stage(

ref\_date varchar(20) not null,

geography varchar2(50) not null,

dguid varchar2(50) not null,

sex varchar2(50) not null,

age\_group varchar2(50) not null,

uom varchar2(50) not null,

uom\_id varchar2(50) not null,

scalar\_factor varchar2(50) not null,

scalar\_id varchar2(50) not null,

vector varchar2(50) not null,

coordinate varchar2(50) not null,

population varchar2(50) not null,

status varchar2(50),

symbol varchar2(50),

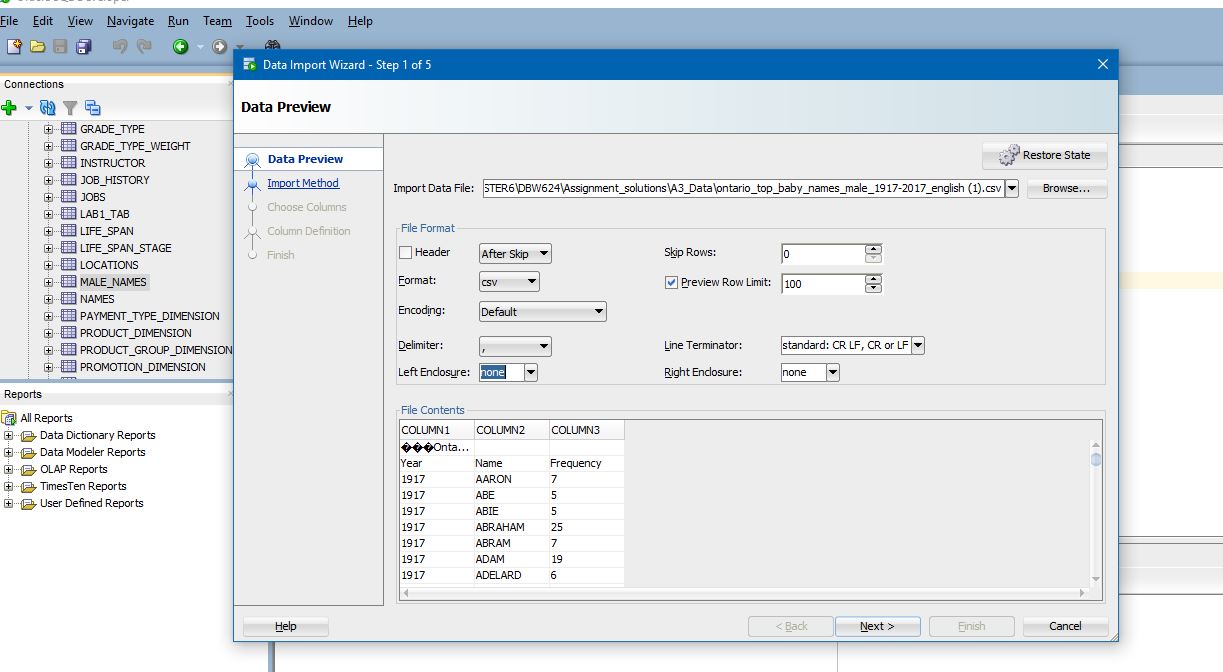
terminated\_ varchar2(50),

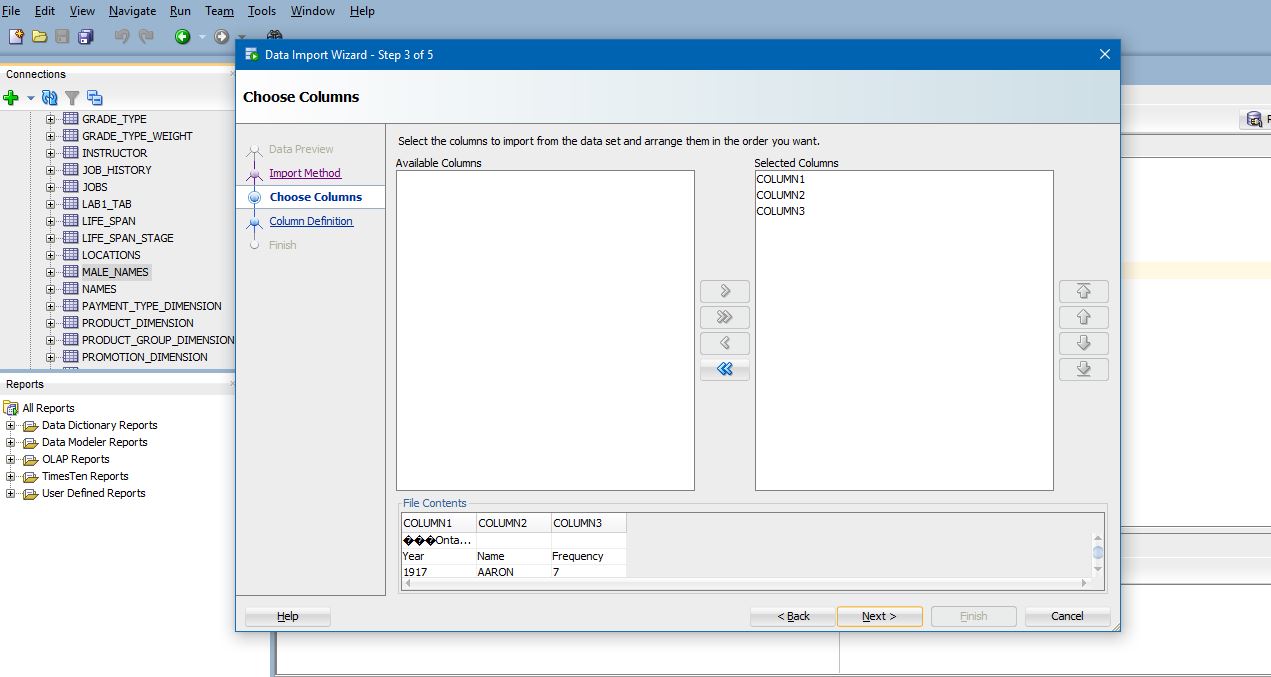
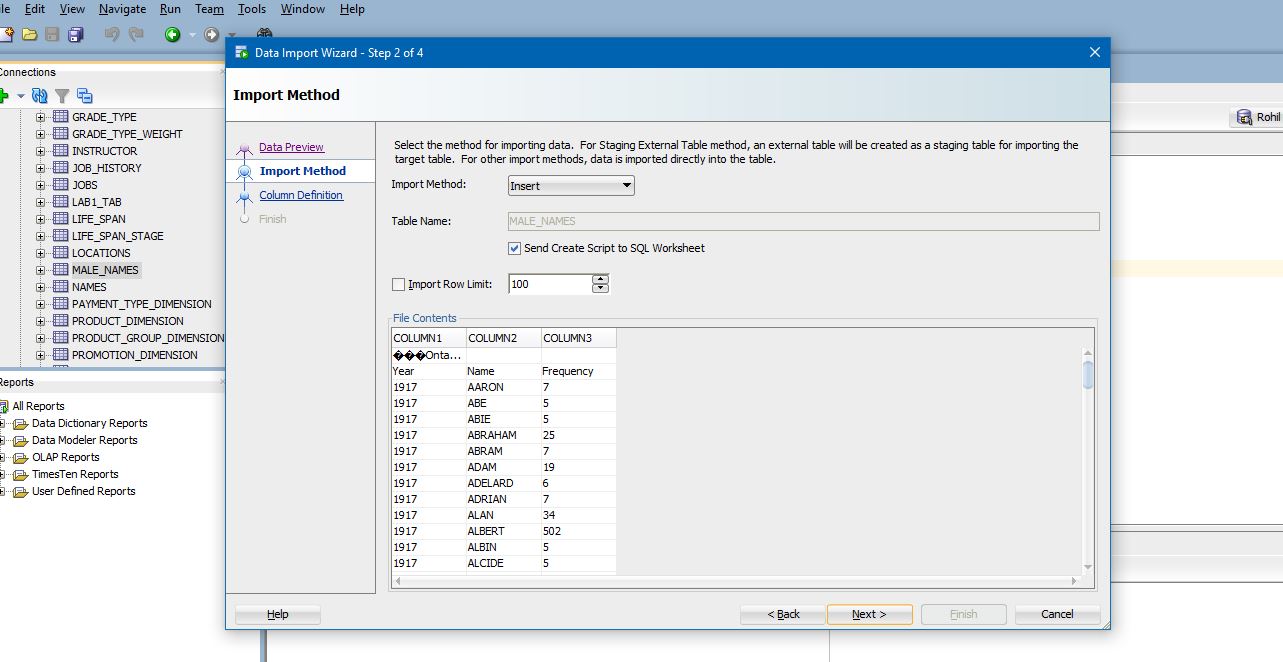
decimals varchar2(50) not null

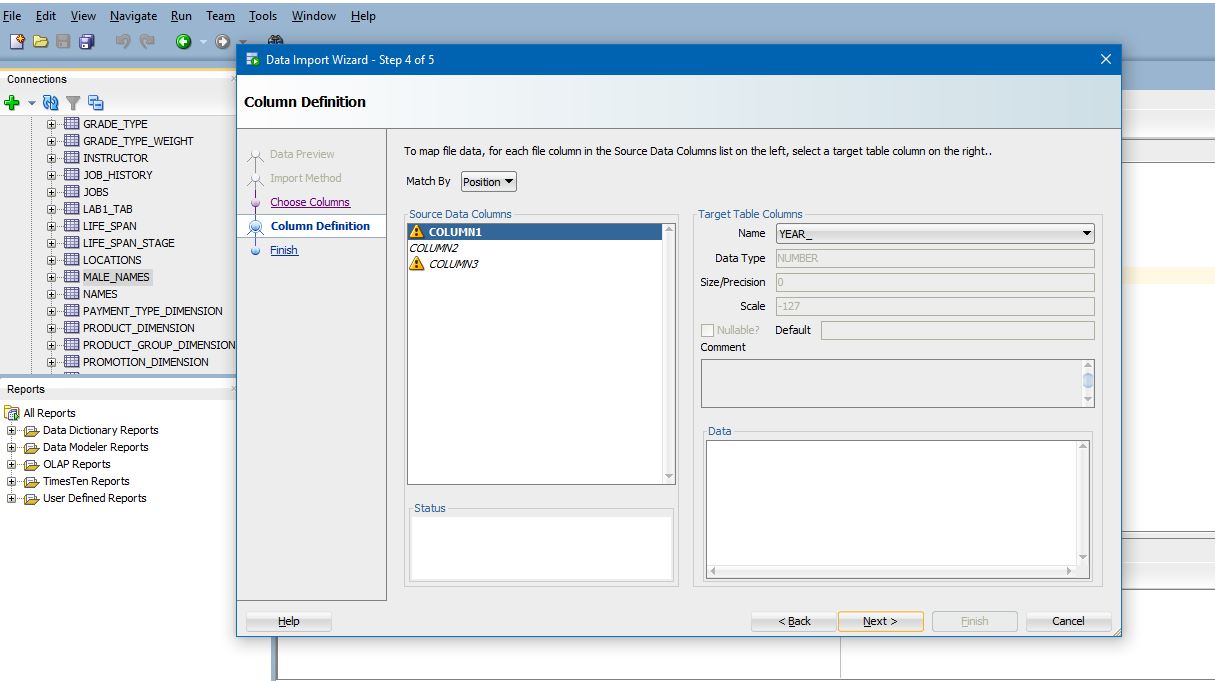
);

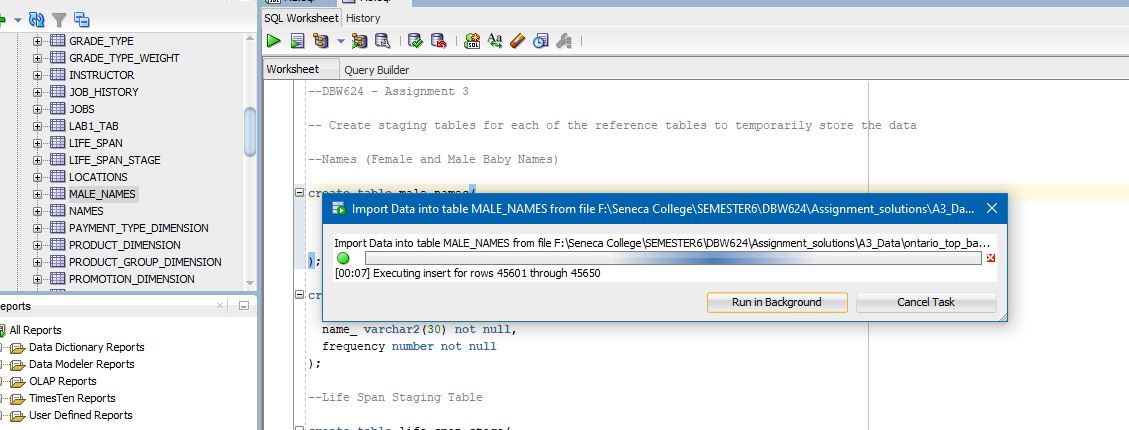
1. **Now that we have the skeleton established, we will right click on each of these tables on SQL developer and “Import Data” from the corresponding csv files as follows:**

**\*\*REPEAT THIS PROCEDURE FOR FEMALE\_NAMES, LIFE\_SPAN\_STAGE & CITY\_POPULATION\_STAGE Tables**

****

****

****

****

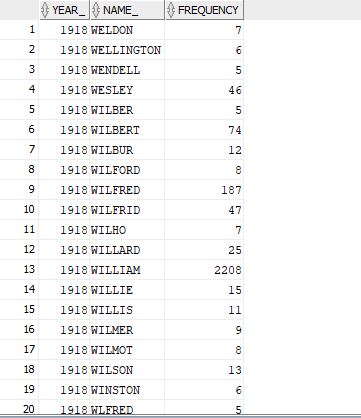
1. **Verify the presence of data in EACH of these staging tables**

**MALE NAMES**

**F:\Seneca College\SEMESTER6\DBW624\Assignment_solutions\A3_steps\7.JPG**

select \* from male\_names;

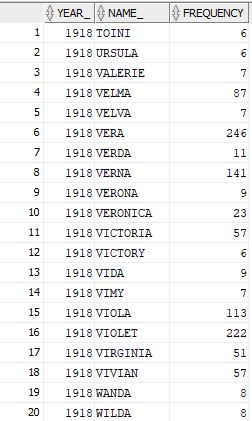
select count(\*) from male\_names;

****

**FEMALE NAMES**

select \* from female\_names;

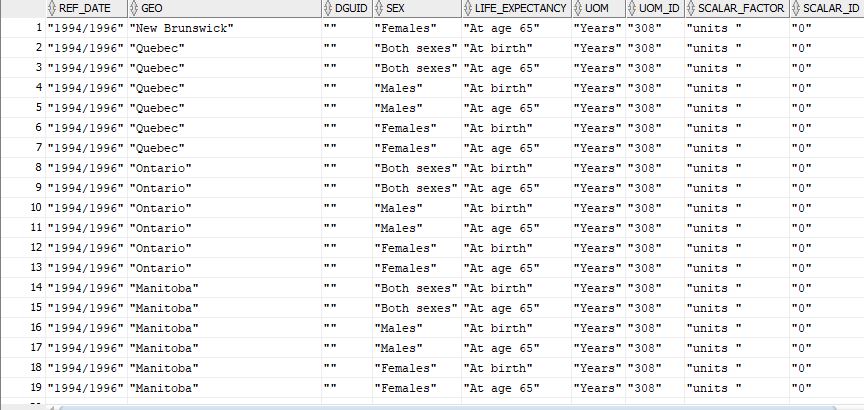
**F:\Seneca College\SEMESTER6\DBW624\Assignment_solutions\A3_steps\9.JPG**select count(\*) from female\_names;

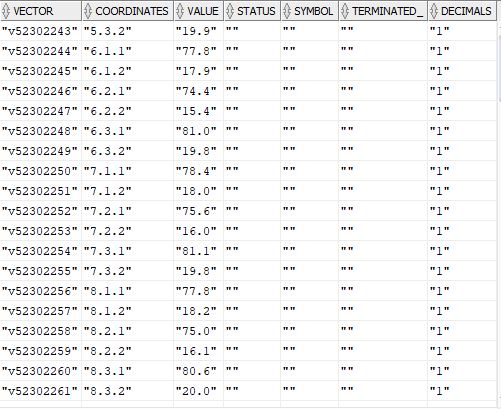
****

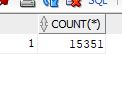
**F:\Seneca College\SEMESTER6\DBW624\Assignment_solutions\A3_steps\11.JPGLIFE SPAN**

select \* from life\_span\_stage;

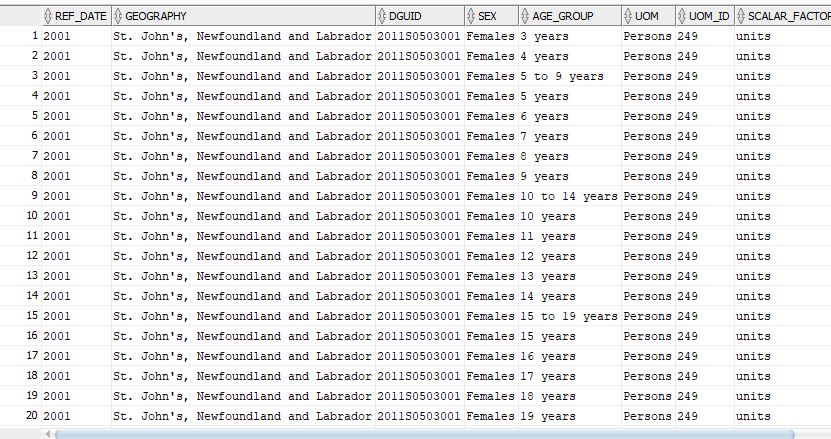
select count(\*) from life\_span\_stage;

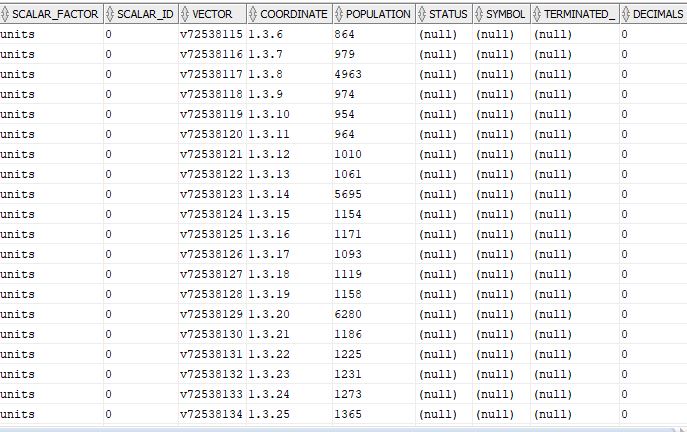
****

****

**CITY POPULATION**

select \* from city\_population\_stage;

select count(\*) from city\_population\_stage;



**Step 2 – TRANSFORM & CLEANSE**

**This is the ideal recipe for each of these tables we will be dealing with**

**Using common table expressions (cte) will help us scope a particular select statement and eventually reference if when the time comes for deletion.**

1. **Remove Null Values**
2. **Remove Unnecessary fields**
3. **Transform columns into meaningful data where required**
4. **Match the columns to the ones in the warehouse**

**MALE NAMES**

delete from male\_names where

name\_ is null or year\_ is null or frequency is null;

**- 0 rows deleted (Hence proved that there are no null values)**

**--Add a gender field to correspond with the original table**

alter table male\_names add gender varchar2(1) not null;

alter table male\_names

modify gender default 'M';

**OR**

alter table male\_names modify (gender varchar2(1) not null default 'M');

**--drop this limitation afterwards (for some reason it didn’t work using sql so I had to do it manually)**

**FEMALE NAMES**

delete from female\_names where

name\_ is null or year\_ is null or frequency is null;

**- 0 rows deleted (Hence proved that there are no null values)**

**--Add a gender field to correspond with the original table**

alter table female\_names add gender varchar2(1) not null;

alter table female\_names

modify gender default 'F';

**OR**

alter table female\_names modify (gender varchar2(1) not null default 'M');

**CITY POPULATION**

**Remove all the unnecessary columns**

**--This is to ensure that the remaining columns correspond with the ones in the warehouse**

alter table city\_population\_stage

drop (dguid, scalar\_factor, scalar\_id, uom, uom\_id, vector, coordinate,

status, symbol, terminated\_, decimals)

**--Memory preservation - Reduce the length of/ modify the values so that they take up relatively less space – good practice**

update city\_population\_stage

set sex = 'M'

where sex = 'Males';

update city\_population\_stage

set sex = 'F'

where sex = 'Females';

update city\_population\_stage

set sex = 'B'

where sex = 'Both sexes';

**--Remove any null values**

delete from city\_population\_stage where

geography is null or sex is null or ref\_date is null or age\_group is null or population is null;

**- 0 rows deleted (Hence proved that there are no null values)**

**--Delete rows with an age range to make life simpler**

delete from city\_population\_stage where

age\_group like '%to%';

**LIFE SPAN**

**--Drop unnecessary columns**

alter table life\_span\_stage

drop (dguid, scalar\_factor, scalar\_id, uom, uom\_id, vector, coordinates,

status, symbol, terminated\_, decimals)

**--Drop unnecessary symbols – those annoying quotation marks won’t exist after these queries have been run**

update life\_span\_stage

set ref\_date = replace(ref\_date,'"', '');

update life\_span\_stage

set geo = replace(geo,'"', '');

update life\_span\_stage

set sex = replace(sex,'"', '');

update life\_span\_stage

set life\_expectancy = replace(life\_expectancy,'"', '');

update life\_span\_stage

set value = replace(value,'"', '');

**--Memory preservation - Reduce the length of/ modify the values so that they take up relatively less space**

update life\_span\_stage

set sex = 'M'

where sex = 'Males';

update life\_span\_stage

set sex = 'F'

where sex = 'Females';

update life\_span\_stage

set sex = 'B'

where sex = 'Both sexes';

**--Remove any null values**

delete from life\_span\_stage where

geo is null or sex is null or ref\_date is null or life\_expectancy is null or value is null;

**- 0 rows deleted (Hence proved that there are no null values)**

**Step 3 – LOAD**

**Now that everything’s cleaned up, Load the data into their designated tables**

**(See next page)**

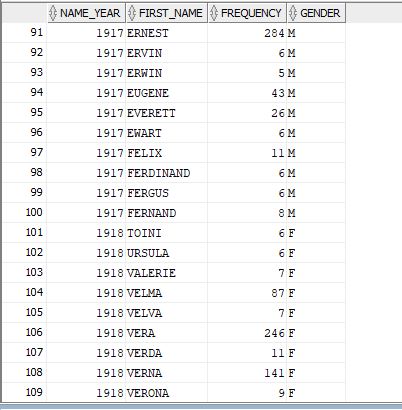
**NAMES DIMENSION**

insert into names(name\_year, first\_name, frequency, gender)

select \* from male\_names union select \* from female\_names;

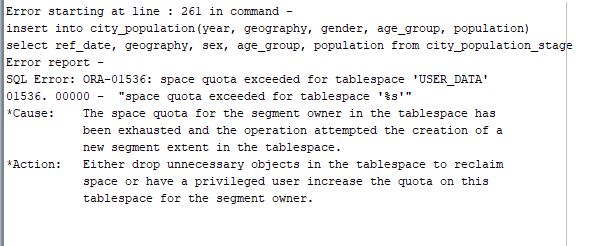
F:\Seneca College\SEMESTER6\DBW624\Assignment_solutions\A3_steps\14.JPGselect count(\*) from names

select \* from names



**For some reason, when I run the insert statement, it throws a quota error saying I have exceeded the tablespace storage for ‘USER DATA. Since I am not a sys admin, regulating it is not in my hands haha.**

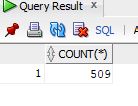
**Hence I had to run a few inserts manually.**

**CITY POPULATION**

insert into city\_population(year, geography, gender, age\_group, population)

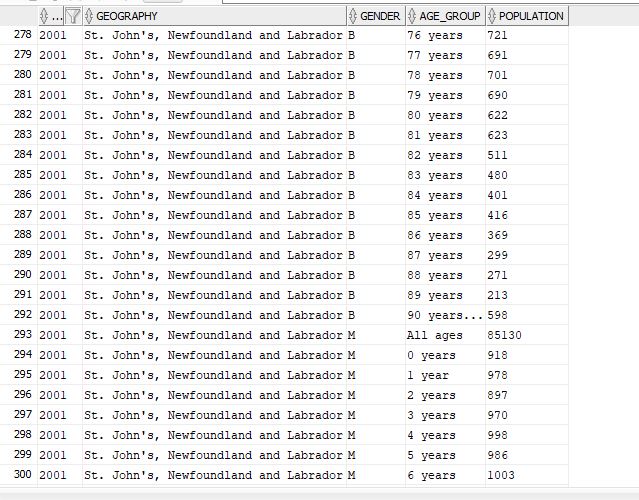
select ref\_date, geography, sex, age\_group, population from city\_population\_stage

* **Same quota error persists**
* **Hence it had to be done manually**



select count(\*) from city\_population

select \* from city\_population



**LIFE SPAN**

insert into life\_span(year\_range, geo, gender, life\_expectancy, value)

select \* from life\_span\_stage  **-**

**\*\*This didn’t generate any errors (strangely) – probably because only 400 rows were extracted in this staging table as opposed to the 1000-10000 extracted in the other 2 tables in Step 1**

**Future Solution - I think I’ll need to enable a limit on the number of rows to be imported for the extract procedure for future use**

F:\Seneca College\SEMESTER6\DBW624\Assignment_solutions\A3_steps\18.JPGselect count(\*) from life\_span

select \* from life\_span\_stage

