DBAS32100 – Final Project

Group 5

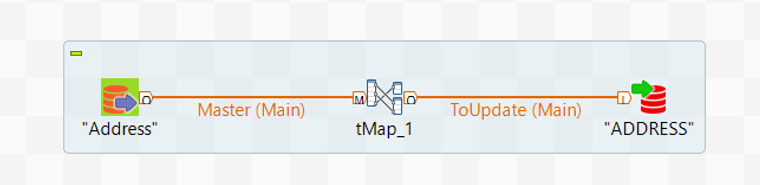
Jordan Way, Omar Kanawati, Erin Evite, Fernando Dominguez

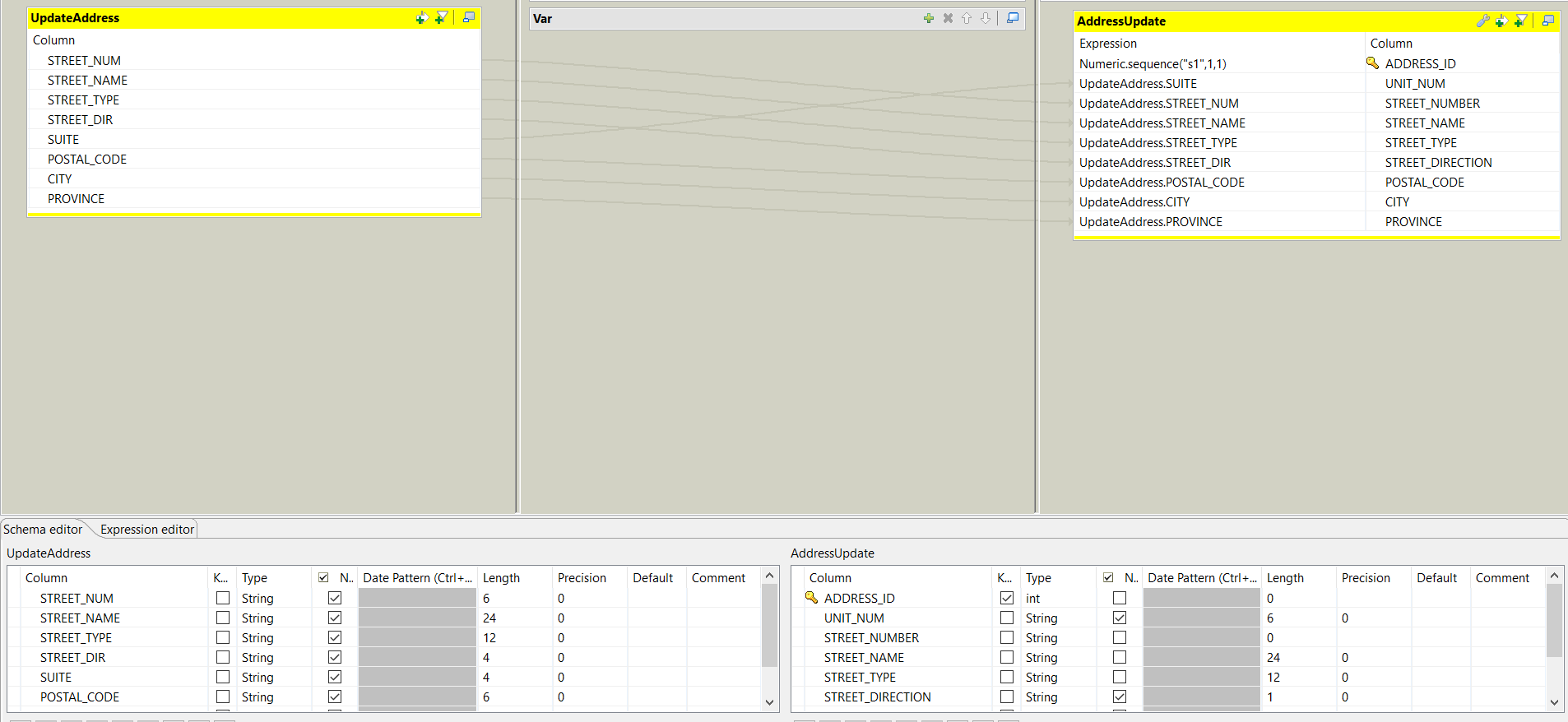
**Table Of contents**

|  |  |
| --- | --- |
| Part 1 - Refresh the address table in the oracle database with the addresses in the master table | Page 2 |
| Part 2 - Create a process that loads the donations list to the central donations table | Page 3 – 6 |
| Part 3 - Star Schema Documentation | Page 7 – 9 |
| Part 4 - Process to load the data to the star schema from the central donation repository | Page 10 - 12 |
| Part 5 - Views | Page 13 – 18 |
| Part 6 - BASIC SECURITY | Page 19 - 24 |

**Part 1 - Refresh the address table in the oracle database with the addresses in the master table**

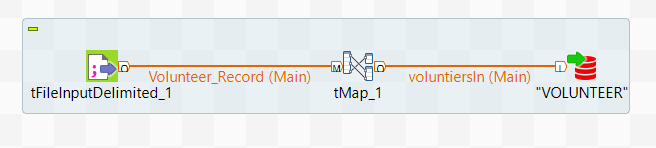
Update the community address table (SQL Developer) based on the records found in the master databases address table. Changes to ID values were made from bigDecimal to Integer.



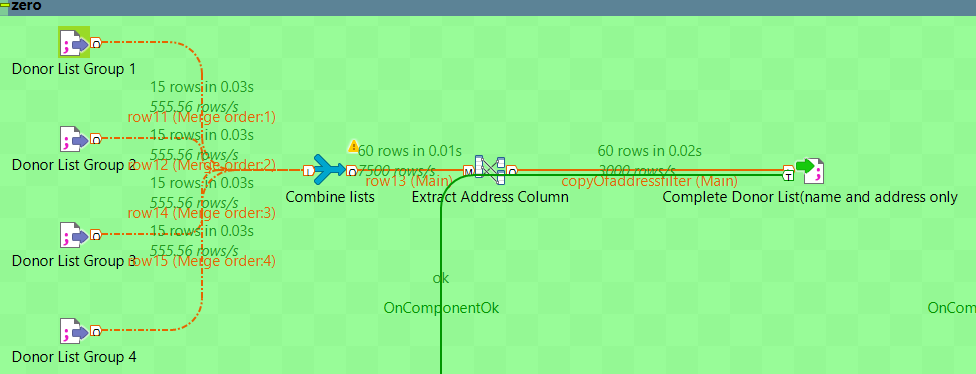


**Part 2 - Create a process that loads the donations list to the central donations table**

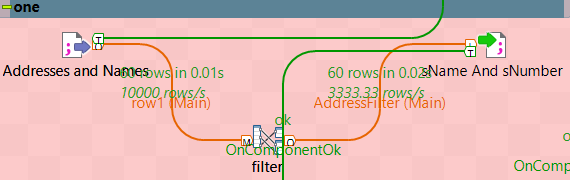
Here the volunteer list is integrated to the community’s database in the volunteer table. Once again BigDecimal on the GROUP\_LEADER column was changed to integer and the ID column was generated by an increment of one, starting at one when uploading records to the database.



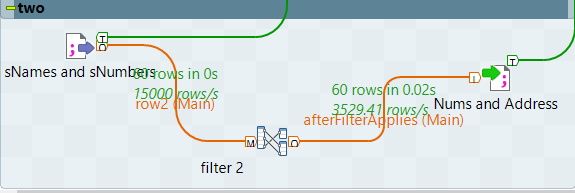
Now that volunteer table has been updated, we proceed to find the address ID’s with help from the master database.



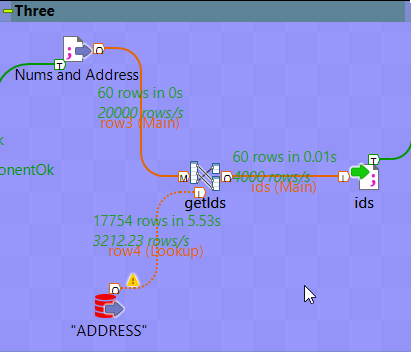
Using a combined list, we gather all the donors list together isolating the addresses.



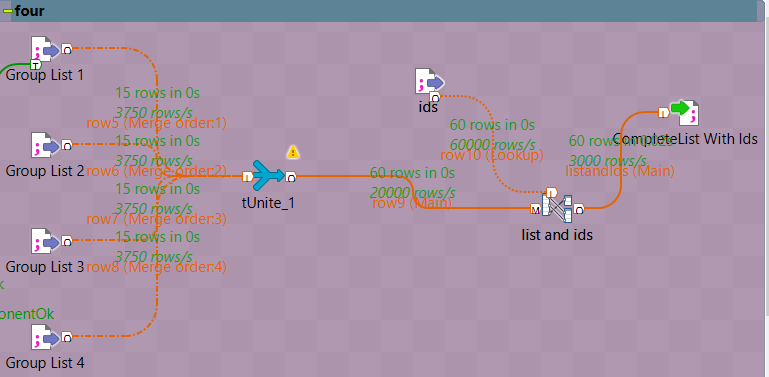
Here, the number and the street name are filtered out.



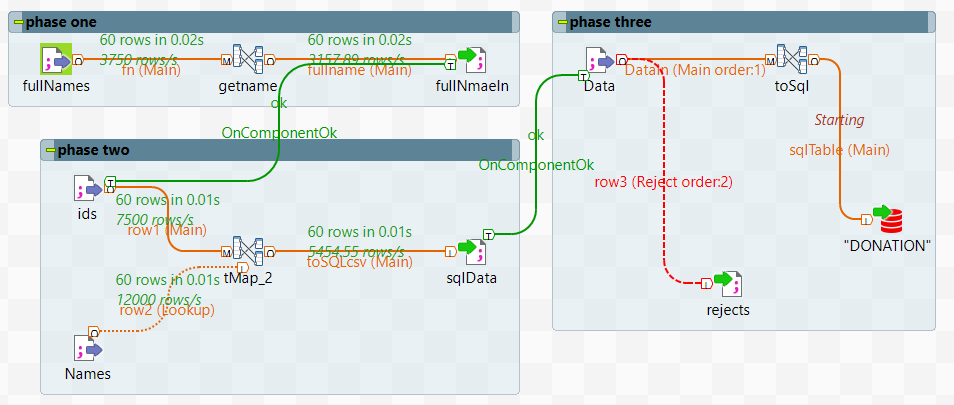
After extracting the street number and street name to a file (numsandsName.csv) we proceed to (give) them their own columns for further examination.



By giving the street number and street name their own columns, we can now use them to find their ids by comparing every entry value to the address table steet\_Num and steet\_name.



Now that we have their ID’s, all we must do is combine them with compleatedonationList.csv forming one single file call completeListplusIDs.csv for further procedures.



**Phase one**: takes the full name from the list of donors.

**Phase two**: takes the data from the donors list (the one with the address id’s) plus the file names.csv made by phase one and now we have all the data needed for our SQL entries.

**Phase three**: using the SQL entry file sqlEntries.csv and creating a rejects file call rejects.csv. Data is ready to go into the SQL table, sending records with null fields or data not matching our database records based on null entries to a rejects.csv which will later be sent back to the office for proper corrections.

**Part 3 - Star Schema Documentation**

**fact\_don table**

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Description** |
| d\_id | number | Foreign key to dim\_date table |
| a\_id | number | Foreign key to dim\_address table |
| v\_id | number | Foreign key to dim\_volunteer table |
| donation\_amount | number (10,2) | Dollar amount that was donated |

**dim\_date table**

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Description** |
| d\_id | number | Primary key of date dimension table. Foreign key to fact\_don table |
| Donation\_date | date | Date that donation was made |

**dim\_address table**

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Description** |
| a\_id | number | Primary key of address dimension table. Foreign key to fact\_don table |
| address\_id | number | Foreign key to address table. |
| street\_number | number | Street number of donor’s residence |
| street\_name | varchar2(24 byte) | Street name of donor’s residence. |
| postal\_code | char (7 byte) | Postal code of donor’s residence |
| city | varchar2(16 byte) | City of donor’s residence |
| province | char (2 byte) | Province of donor’s residence |

**dim\_volunteer table**

|  |  |  |
| --- | --- | --- |
| **Columns** | **Data Type** | **Description** |
| v\_id | number | Primary key of volunteer dimension table. Foreign key to fact\_don table |
| volunteer\_id | number | Foreign key to volunteer table |
| first\_name | varchar2(16 byte) | First name of volunteer |
| last\_name | varchar2(16 byte) | Last name of volunteer |
| group\_leader | number | ID number of the group leader. Foreign key to volunteer table. |

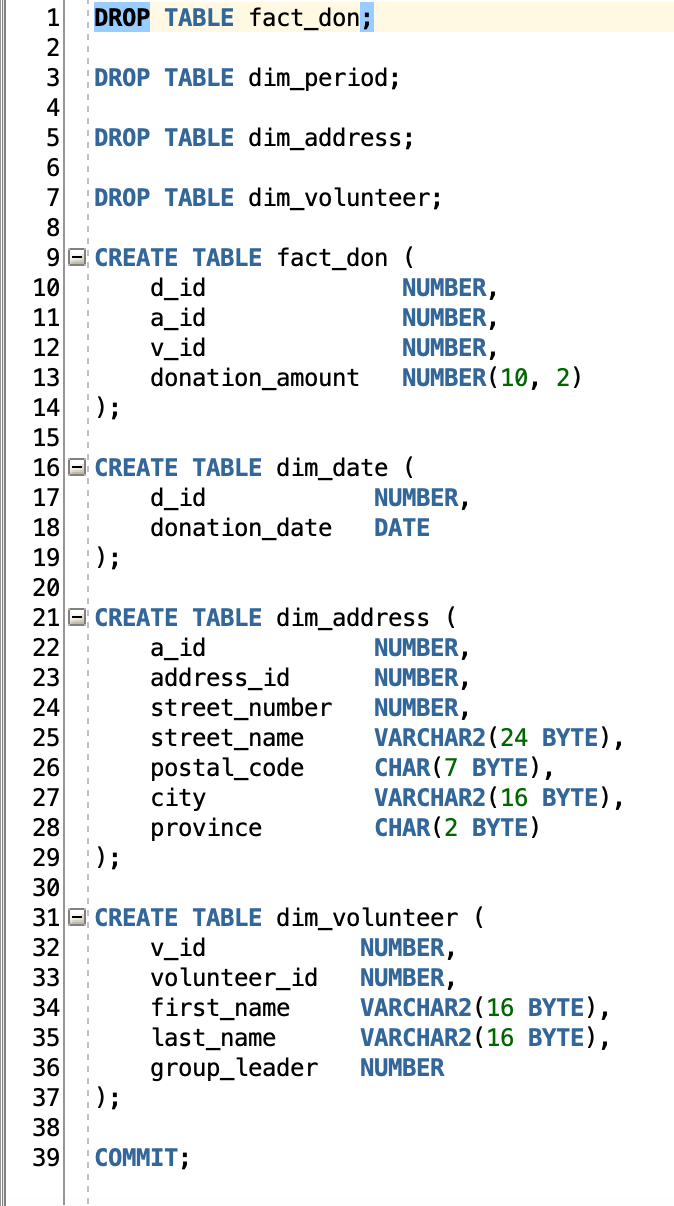
The fact and dimension tables are constructed based on the definition of the grain. In this case, the grain is a combination of the day the donation was made, the address of the donor, and which volunteer is assigned to the donation area. We can determine through the grain that the dimensions will be “dim\_date”, “dim\_address”, and “dim\_volunteer”. The fact table contains the quantitative value that connects these three dimensions, which is the donation amount. The donation amount is stored in “fact\_don”, along with the foreign keys of the three dimension tables.

The level of detail in the dimensions are based on both the granularity of the star schema, and what information is needed for the different views. The first view displays the average and sum of donations based on day, month, and year. Therefore, the “dim\_date” table will need the entire date value from the donation table.

The second view displays the average and sum of donations based on the address and postal code. Because the entire address is needed, the level of detail required is significant, resulting in including the entire address table (except for the street direction, which is not recorded) in the address dimension table.

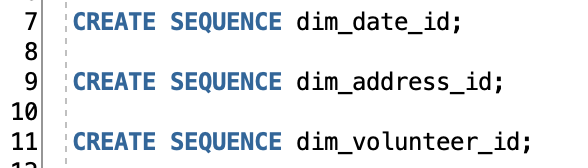
The third view displays the average and sum of donations based on volunteer and volunteer group leader. Again, the amount of information about the volunteer is not specified, so all information from the volunteer table is included in the volunteer dimension table.

Source code of the star schema

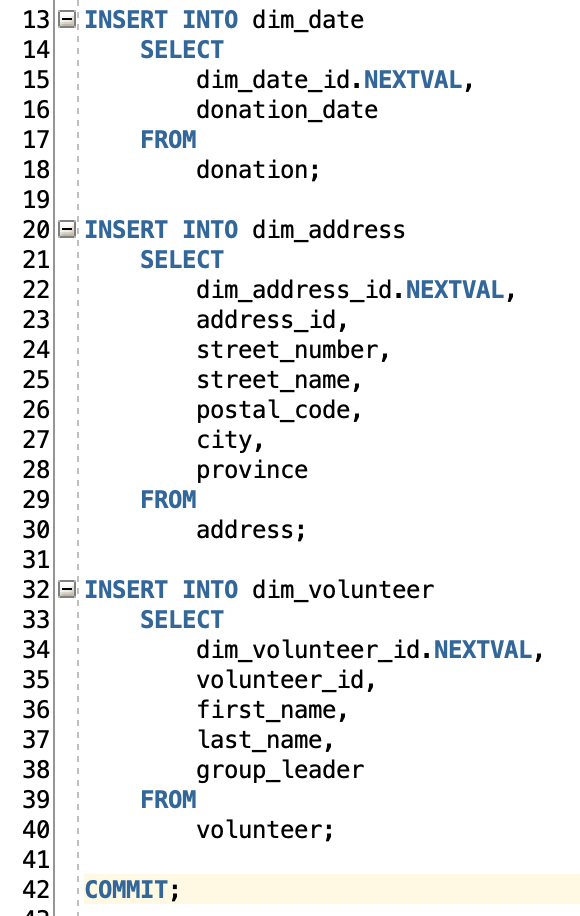


**Part 4 - Process to load the data to the star schema from the central donation repository**

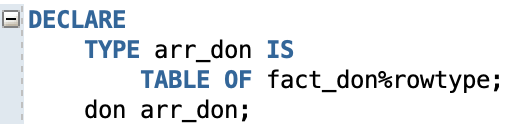
The star schema needs to be loaded with the data collected and filtered into the central repository table. Sequences need to be created for all three dimension tables in order to create auto incrementing integers that will serve as the table’s primary keys.



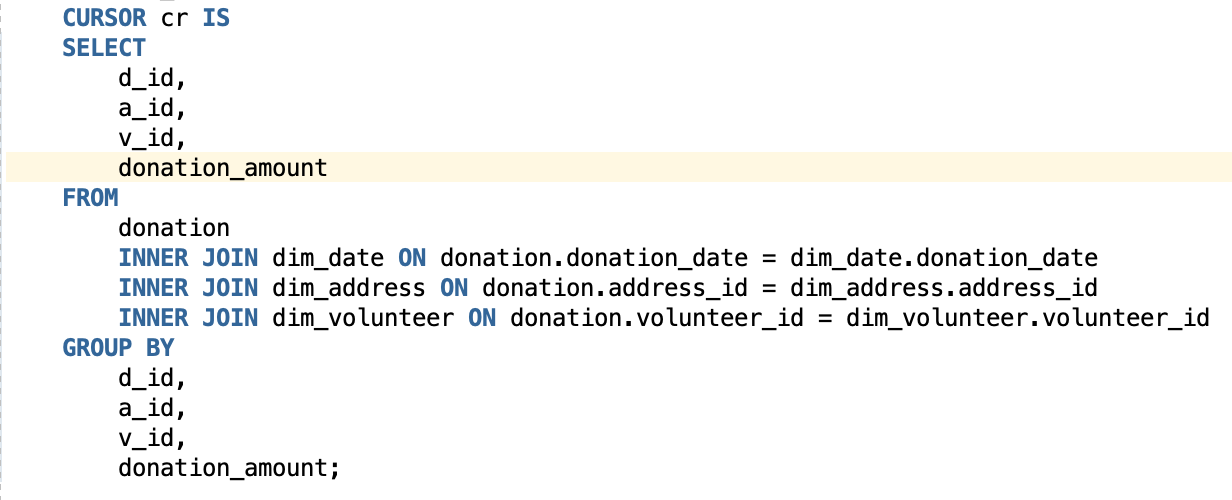
Once the sequences are created, they are used to create the surrogate keys of the dimension tables every time they are called using “.NEXTVAL”. This will be used within a SELECT statement to populate the dimension tables and assign them with unique ID values.



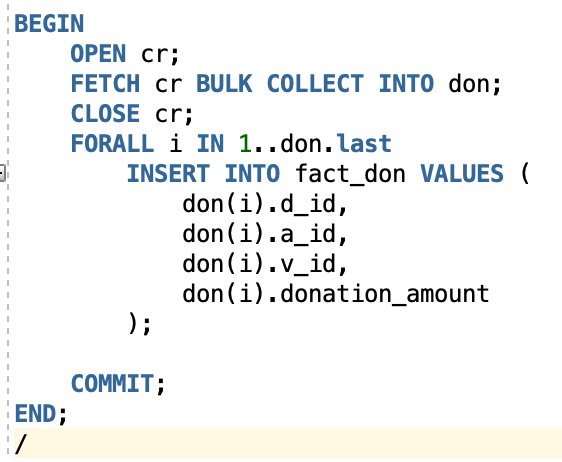
The following PL/SQL script is used to populate the fact\_don table. An array of records found in the fact\_don table is created, with the attribute %rowtype to store each row as they are represented in the database.



A cursor is used to traverse the array, selecting every column in the fact\_don table and retrieving the data needed to populate it from the main donation table and the different dimension tables through consecutive INNER JOINs.



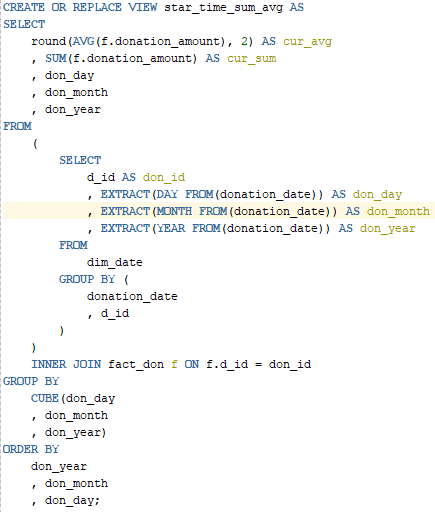
The cursor is opened, and a BULK COLLECT is performed to traverse the array and insert all the values into the fact\_don table. Once the process is complete, the transaction is committed, and the block has ended.



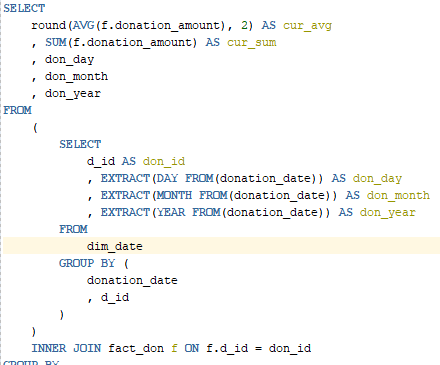
**Part 5 - Views**

**5.1 - View that shows the average and sum of the donation by day, month, year**

Source code of the view

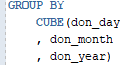


Fields Required



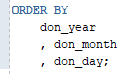
The sum and average of the donation amount is included as required. A subquery is made to get the day, month and year from the donation\_date field which included all those values in one field, these fields are separated so that the query results can be calculated by daily, monthly and yearly totals. d\_id is included so that the subquery results (date results) can be connected to the fact\_don table allowing access to the donation\_amount

Group by Cube



Cube was used to calculate the subtotals by the donation day, month and year. The order of the cube sets the query to get subtotals first by the day, then calculate a monthly total and finally calculate a yearly total.

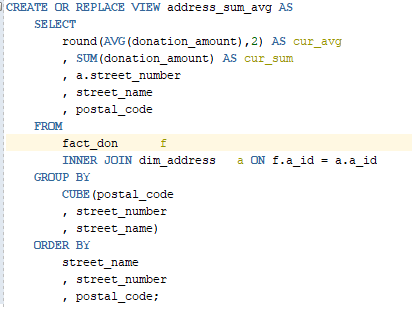
Order by



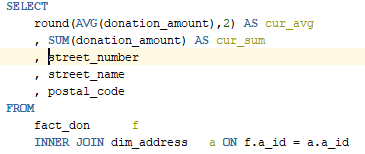
The order by formats the output to sort by day, month and year; all ascending (ex. 1, 1, 1990 – 12,2,2020)

**5.2 View that shows the average and sum of the donations by address, postal code**

Source code of the view

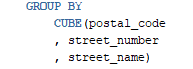


Fields Needed



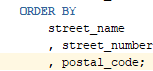
The average and sum of the donation amount is required for the view. Fact\_don is joined to dim\_address so that the fields of street\_number, street\_name and postal\_code can be accessed.

Group by Cube



Cube was used to calculate the subtotals by postal codes, street numbers, and street names. The order of the cube sets the query to get subtotals by postal code, by a street name and then the individual street houses / numbers.

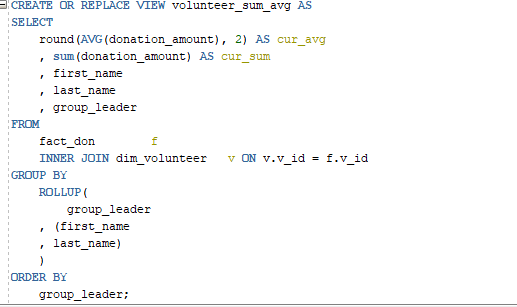
Order by



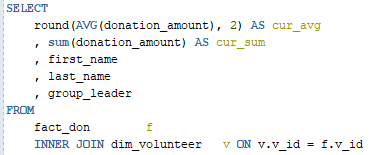
The order by is used to format the output, it will display divided by the street name, list the houses in order and then order them by postal codes

**5.3 View that shows the average and sum of the donations by volunteer and volunteer group leader**

Source code of the view

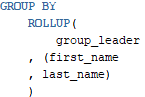


Fields needed



The sum and average of the donation amount is required, this is gathered from the fact\_don table. The volunteer table is joined to fact\_don so that first and last names and the group leader ID can be included

Group by Rollup

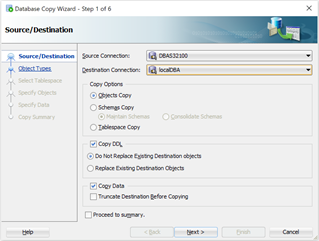


Rollup was used to calculate the subtotal donations received by individuals (i.e. John Smith) and the subtotal of their group. Cube wasn’t used here since only the individual and group subtotals were required whereas cube would’ve produced tabulated data that isn’t needed here.

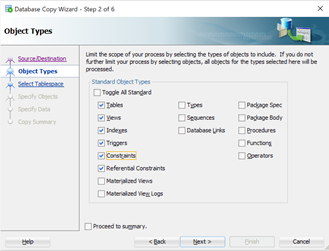
The order of the rollup sets the query to divide by volunteer groups (gets the group subtotal), then divide by volunteer name (individual subtotals)

**Part 6 - BASIC SECURITY**

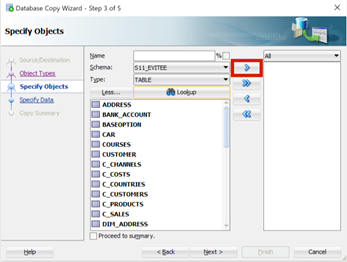
Due to insufficient privileges on the class database, the security portion of the project was done on the local database created on week 11. The tables and views from the project do not exist in the local database so it must be copied from the class database. In order to do this, go to Tools and select ‘Database copy…’.



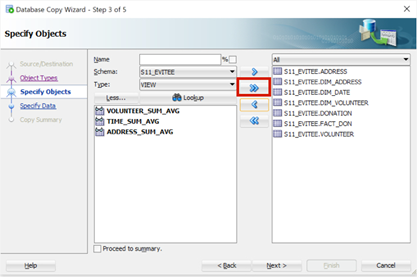
Set the Source Connection to the class database and Destination Connection to the local database. A database version warning window will pop up and click “Yes” to continue.



In the Object Types window, only select Tables, Views, Indexes, Triggers, Constraints, and Referential Constraints as shown above to limit the scope. In the next window, click on the “More…” button. Select “Table” from the drop-down field of “Type” to filter the selection as shown in the image below.



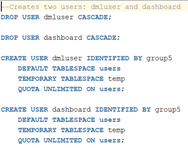
Select ADDRESS and click on the first arrow button (with red border) to move it to the adjacent text field. Do this again for DIM\_ADDRESS, DIM\_DATE, DIM\_VOLUNTEER, DONATION, FACT\_DON, and VOLUNTEER.



To transfer the views, set the Type drop-down field to “View” and click on the Lookup button. The three views will show up; and to move them, click on the second arrow button (with the red border) and then hit the “Next >” button.

In the next window (Specify Data), just click “Next >” and select “Finish” in the Copy Summary window to begin the process.

**Create users source code**



The first two lines of script drops any existing users named “dmluser” and “dashboard”. The word CASCADE was included to ensure even all the objects are dropped. The next two statements create new users: “dmluser” and “dashboard” with a password of “group5” for simplicity. A default tablespace for users and a temporary tablespace was created as well.

**Grant privileges to DMLUser and Dashboard source code**



Revoke statements



These statements were included in the script to ensure any existing privileges were revoked before running the next statements.

Grant create session



This system privilege must be granted to the new users so that they can connect to the database.

Grant DML privileges to DMLUser



The object privileges were specified instead of using “all” to make sure only DML operations are granted to the address, donation, and volunteer tables.

Grant read permissions for Dashboard



By granting the select object privileges on the views, Dashboard user can only read the views.