# **COMP-1701 - Transferring Data to Databases**

## **Data Science & Machine Learning (DSML) - RRC Polytech**

Competency - ERD – Applying Changes With ALTER and Create People JOIN to Emulate the Front of an Envelope

This assignment will connect the ERD diagram entities to the tables you’ve created and give you some understandings of how the data has been applied so you can create a last JOIN for this competency.

Part 1 – ERD (submit file: fl\_01234567\_boxstore.drawio)

Finalize you ERD drawing and save the file with the same naming convention as you master boxstore file, instead of file extension .sql you use .drawio. File name would be like **fl\_0123456\_boxstore.drawio**

And should be like the diagram on the following page. Noting below:

* **TINT**/**SINT**/**MINT** below mean **TINYINT**/**SMALLINT**/**MEDIUMINT**  
  phone numbers will be North American (without the 1)
* all PKs should be **UNSIGNED AUTO\_INCREMENT** (marked as **US AI**)
* first\_name/last\_name VARCHAR(**35**)
* phone numbers will be **CHAR(12)** just North American format: ###-###-####

A computer screen shot of a diagram

Description automatically generated

Part 2 – SQL (submit file: fl\_01234567\_boxstore.sql)

Your existing fl\_0123456\_boxstore.sql should contain the following sections in the following order:

1. DROP/CREATE/USE fl\_0123456\_boxstore DATABASE section
2. TABLE geo\_address\_type (from COMP-1701\_SQL\_E.3)
3. TABLE geo\_country (from COMP-1701\_SQL\_E.3)
   1. DROP/CREATE & TRUNCATE/INSERT/SELECT block
4. TABLE geo\_region (from COMP-1701\_SQL\_E.3)
   1. DROP/CREATE & TRUNCATE/INSERT/SELECT block
   2. geo\_region JOIN to geo\_country   
      (from: COMP-1701\_SQL\_E.4)
5. TABLE geo\_towncity (from COMP-1701\_SQL\_E.3)
   1. DROP/CREATE & TRUNCATE/INSERT/SELECT block
   2. geo\_towncity JOIN to geo\_region JOIN to geo\_country   
      (from: COMP-1701\_SQL\_E.4)
6. TABLE people (from COMP-1701\_SQL\_D.2)
   1. DROP/CREATE & TRUNCATE/INSERT/SELECT block
   2. INSERT consists of the 2 people records (the instructor record p\_id=1 and p\_id=2) ~~and the SELECT to verify~~
   3. LOAD DATA INFILE to load your 10000 people records and SELECT to verify
   4. ALTER TABLE people that adds first\_name and last\_name
   5. UPDATE to update the instructor record and your record that splits the first\_name and last\_name out of the full\_name column (SELECT that verifies those 2 records)
   6. The UPDATE using MID, INSTR, LENGTH functions to UPDATE all the remaining 10000 records. (SELECT to verify).
   7. REMOVE the ALTER TABLE that drops full\_name, and replace it with:

ALTER TABLE people (  
 DROP COLUMN full\_name  
 , MODIFY COLUMN first\_name VARCHAR(35) NOT NULL  
 , MODIFY COLUMN last\_name VARCHAR(35)  
 , ADD COLUMN email\_addr VARCHAR(50)  
 , ADD COLUMN password CHAR(32)  
 , ADD COLUMN phone\_pri CHAR(12) -- NOT NULL later switch  
 , ADD COLUMN phone\_sec CHAR(12)   
 , ADD COLUMN phone\_fax CHAR(12)   
 , ADD COLUMN addr\_prefix VARCHAR(10)   
 , ADD COLUMN addr VARCHAR(60)   
 , ADD COLUMN addr\_code CHAR(7)   
 , ADD COLUMN addr\_info VARCHAR(191)   
 , ADD COLUMN addr\_delivery VARCHAR(191)   
 , ADD COLUMN addr\_type\_id TINYINT(4) -- FK geo\_address\_type  
 , ADD COLUMN tc\_id INT(11) -- FK geo\_towncity  
 , ADD COLUMN user\_mod INT(11) DEFAULT 2  
 , ADD COLUMN date\_mod DATETIME DEFAULT CURRENT\_TIMESTAMP  
 , ADD COLUMN user\_act INT(11) DEFAULT 1  
 , ADD COLUMN date\_act DATETIME DEFAULT CURRENT\_TIMESTAMP  
 , ADD COLUMN active BIT(1) DEFAULT 1  
);

* 1. Create 2 UPDATEs for the Instructor and Your record, by checking the columns in the geo\_address\_type and geo\_towncity tables and using the following information
     + **email\_addr** for each employee will be lowercase and include the **first\_name**, a **dot (period)**, the **last\_name**, and then **@boxstore.com**
     + **password** will be created using the MD5 function, create a difficult password
     + **phone\_** records will be 204 area code, with different phone numbers for each record in the following format: 204-255-1989
     + You are in an Apartment, so add your Room Number to **addr\_prefix** , for Instructor just make it =NULL
     + Choose an different address (**addr**) for each record (ie: 123 Main St) but make your own.
     + Use 2 different Canadian Postal Codes for **addr\_code** (irony will not be lost on the Instructor’s city being used).
     + **addr\_info** use this for listing a PO Box record, look up how they are formatted and add this only to the Instructor’s record, just use = NULL for your record
     + **addr\_delivery**, knock on outside door is the message in the instructor record and check with concierge on main level is the message in your record
     + Instructor lives in a House, You live in an Apartment, check within **geo\_address\_type** for the corresponding Primary Key to use for each record. if the record does not exist, INSERT the value to **geo\_address\_type** maintaining its alphabetic order after reload (~~leave Other option at the bottom~~).
     + Instructor lives in Redmond, Washington, US and You live in Winnipeg, Manitoba, Canada, so checking the **geo\_towncity** table add the corresponding Primary Key
     + Update **usermod**=2 and **datemod**=NOW() as your p\_id is updating this record, and NOW() just gets the datetime right now.
     + All updated data, in any table, should use **Proper Title Case**, just like in the 4 geo tables, uppercase first letters and lowercase rest of word. Any Updates or Inserts done previously all in lowercase or all in UPPERCASE, need to be adjusted. The only 2 fields that should have uppercase values only, are the addr\_code (which is the postal code) and the rg\_abbr, co\_abbr (which is the region/country code).
     + Only string values and dates would require single quote (') characters around them, any numeric fields DO NOT REQUIRE THEM.
  2. Copy and Edit the last SELECT with:

SELECT p.p\_id, p.first\_name, p.last\_name  
FROM people  
WHERE p.active=1;

then add a table alias of p after people and p. and add the rest of the new columns explicitly, try to group similar columns on the same line, keep each line within the 70 characters per line.

In the end, only 2 rows should have data in it, instructor record (p\_id=1) and your record (p\_id=2).

1. **Creating what appears on the front of an envelope:**  
     
   Create one query, that has multiple JOINs to all the tables we have, from **people** to **geo\_address\_type**\_to **geo\_towncity** to **geo\_region** to **geo\_country**. You have 3 queries now that you could use to construct this large JOIN query. This must be done in one query.  
     
   The front of an envelope you are either sending from, in the top left corner envelope, and sending to, centered in the middle of the envelope, are in a specific order.  
     
   You will replicate this, by calling 2 rows of data and creating the final JOINs from the people table to JOIN to the **geo\_address\_type** table (you should have the basic SELECT query you can use from within the Module E.3 file)  
     
   From within the Module E.4 file, you should have the JOIN from **geo\_towncity** thru **geo\_country** so take that final JOIN and add it to this query.  
     
   **Exclude** all **the geo\_table active** columns for this exercise, only do check the **p.active=1** for the final JOIN here.  
     
   Only the fields that are address related fields apply (the **addr\_info** field with the PO Box value, though you can show the **addr\_delivery** field too, but do it last), along with the first/last names being the first 2 columns you pull. For apartments, sometimes on the envelope it will say the word apartment or condo (etc) so as mentioned show the address type.  
     
   ~~Skip all of the people columns that have nothing to do with this exercise (ie: user\*, date\*).~~  
     
   When writing any JOINs p\_id the PK for people, should be displayed first, the FKs within the people table should be displayed, and the PKs in the corresponding tables should be displayed, with their “reason for existing” VARCHAR fields in those secondary tables.  
     
   Construct this JOIN to keep the FK values close to their PK counter parts. So have a line for:  
     
   **, p.addr\_type\_id, p.tc\_id**

Then the next line should be the:

**, gat.addr\_type\_id** *… and its unique column*

And after should be the

**, gtc.tc\_id** *… and its unique column*

**Advice:** as soon as you have listed an FK in a table, type out the JOIN below the DROP/CREATE/TRUNCATE/INSERT/SELECT block of the table that contains the FK then JOIN to the PK of the table. If the FK appears in 2 different tables, then the JOIN should include the other tables (ie: geo\_towncity, geo\_region, geo\_country).  
  
After you just need to create the geo\_address\_type JOIN and the 3 table geo\_\* JOIN to the people table. After its explicit column’d SELECT.

**PLEASE SUBMIT ALL FILES AS IS, files are to be in a lowercase naming format, 3 files are to be submitted are the drawio and boxstore sql file and updated cheatsheet SQL file.**

**DO NOT ZIP UP THE FILES! Thank you!!**

Next Up:

Round up on what we’ve learnt about JOINs and UNIQUEness/Elimination of Duplicates.

## 