ALGONQUIN COLLEGE

School of
Advanced Technology

Assignment 1 - Weight - 5%

NOTE: This assignment has related information located in the Workbook under Assignment 1. Though similar, there are differences between the two assignments - *it is this assignment which you must follow and submit*.

Objectives:

1. Extend, enhance and populate existing database.
2. Enforce constraints to maintain referential integrity.
3. Write queries using JOIN, subquery.
4. Write VIEWs.

Team Rules

One model per team. You may work alone or work in a group of two. Select a team member from the *same* lab section.

Submission

Every student must upload their work, regardless of working alone or in a group, by the due date, to the assignment designation located in the Brightspace course site (Activities>Assignments>Assignment #1).

IMPORTANT: When submitting your assignment in Brightspace, your assignment (consisting of five files), must begin with your class section number, followed by your last name, then your first name and finally the assignment number. For example:

301_Smith_John_A1 or 303_Roy_Mary_A1

Five files must be submitted for the assignment:

- **File #1**: A DDL .sql file (Ensure file commands can be executed.)
- **File #2**: A DML .sql file (Ensure file commands can be executed.)
- **File #3**: CREATE VIEW statements go in a separate .sql file (even though it is a DDL file)
- **File #4**: Delete errors explained in a .txt file
- **File #5**: All JOIN questions should be in a separate DML .sql file.

Background

In this assignment you will create a new database based on the existing databases from Lab #3 (World) and Lab #7 (Inventory) by modifying the DDL statements, enforcing the constraints, testing and verifying the database.

Requirement

The trading company sells its products to customers in other countries, it wants to identify its customers by country. The company also purchases products from different countries and wants to keep track of country of origin of the product.

Extend the Database You need to extend the Inventory database and implement additional requirements. The new database is titled **InventoryII**. You are required to modify the two original Inventory scripts (DDL and DML) from Lab #7, run and test them. The scripts will create the database and the tables.

- **NOTE: This is File #1 to be submitted (Parts 1, 2, 3 and 4 below)**

1. **Country Table** Add a country table, named **COUNTRY_T**, similar to the country table in the world database (from Lab #3). Add the following attributes to the table:
Code, character 3, Primary Key
Cntry_Name, variable length 30, Mandatory data
Cntry_Population, INTEGER
Add a suitable constraint, use the same naming convention for constraints as in the Inventory database, begin the name with PK .

2. **City Table** Create a city table, named **CITY_T**, similar to the city table in the world database (from Lab #3). Add the following attributes to the table:

City_ID INTEGER, Primary Key
City_Name variable length character, 30
Cntry_Code fixed length character, 3
City_Population INTEGER
Add a foreign key constraint on the Cntry_Code attribute. Reference it to Code in the COUNTRY_T table.

3. **Modify Customer_T table** (from the Inventory database - Lab #7).

   a. Add a column to the Customer_T table. Call it **Cust_Country**, using an appropriate type to accommodate the CountryCode (CntryCode) from the Country T table. Add a foreign key constraint to the Customer T table, Cust Country references CntryCode in the Country T table. Use the same naming conventions for this constraint.

4. **Modify Product_T table** (from the Inventory database - Lab #7).

   a. Add a column to the Product_T table, call it **Cntry_Origin**. This attribute indicates the country of origin of the product. Add a foreign key constraint to the Product_T table to reference CntryCode in the Country_T table.

5. Populate COUNTRY_T and CITY_T tables. Add about 10 countries and 20 cities. You may use the existing data from the world database. Add at least the following six countries in the **COUNTRY_T** table. The sample code should work in the table you create. All INSERT statements you write should have the names of attributes in it, as shown in the sample below. Modify all INSERT statements in the Inventory-DML.sql file, add the attributes.

- **NOTE: This is part of File #2 to be submitted**

```
INSERT INTO COUNTRY_T ( Cntry_Code, Cntry_Name, Cntry_Population ) VALUES ( 'RUS', 'Russian Federation', 144192450 );
INSERT INTO COUNTRY_T ( Cntry_Code, Cntry_Name, Cntry_Population ) VALUES ( 'MEX', 'Mexico', 119530753 );
INSERT INTO COUNTRY_T ( Cntry_Code, Cntry_Name, Cntry_Population ) VALUES ( 'DZA', 'Algeria', 40400000 );
INSERT INTO COUNTRY_T ( Cntry_Code, Cntry_Name, Cntry_Population ) VALUES ( 'CHN', 'China', 1376049000 );
INSERT INTO COUNTRY_T ( Cntry_Code, Cntry_Name, Cntry_Population ) VALUES ( 'CHL', 'Chile', 18006407 );
INSERT INTO COUNTRY_T ( Cntry_Code, Cntry_Name, Cntry_Population ) VALUES ( 'CAN', 'Canada', 36155487 );
```

6. Populate Customer_T and Product_T tables. Modify the insert statements from the Inventory database (Lab #7) to accommodate the new fields. Add at least one customer from Canada. Make sure that at least one product is from the Russian Federation and another product from Chile, make sure that these products already appear in Invoice_Line_T. Ensure to update the Inventory-DML.sql file

a. NOTE: This is part of File #2 to be submitted

7. Add the CITY_T and COUNTRY_T tables **to the DROP TABLE list (**This will enable you to run the DDL file multiple times.)

a. NOTE: This is part of File #1 to be submitted

b. NOTE: **COUNTRY_T** table is the last one to be dropped. **CITY_T** table should be dropped before the **COUNTRY_T** table. Change the order in the DROP TABLE list, observe the error message, determine the cause of the message. *Aside:* Tables need to be created in a certain order, foreign key constraints determine this. If the order of creation is changed, use ALTER TABLE to add constraints.

8. Views Write two views of your choice. Use the naming convention as ⟨ViewName_V⟩

a. Save the views in a separate file called **Inventory_Views.sql**

b. NOTE: This is File #3 to be submitted

9. DELETE Statements After you have populated the Customer_T and Product_T tables, run the following DELETE statements (Indicate the error number and error message. Explain in your own words the reason for failure):

a. DELETE FROM Customer_T WHERE Cust_Country = 'CAN';

b. DELETE FROM Product T WHERE Cntry_Origin = 'RUS';

c. NOTE: This is File #4 to be submitted (Put your comments in a separate text file called **delete_err.txt**)

10. JOINS Write two joins. You may use RIGHT JOIN and/or LEFT JOIN

a. Write a SQL statement to list countries that do not have any Customers.

b. Write an SQL statement to list countries from which no products are bought.

c. NOTE: This is File #5 to be submitted