**CSC 355 Database Systems**

**Fall Quarter 2018**

**Assignment 1**

**Due Sunday, September 16th @ 11:59pm.**

Supplemental Reading: Elmasri and Navathe (6th edition), Chapters 1.1 – 1.9, 2.1 - 2.2, 7.1, 3.1 - 3.3, 4.1 – 4.4, 5.4.

Oracle Database 12c SQL (available at https://library.depaul.edu/Pages/default.aspx): Chapter1: “Using SQL Developer”, “Adding, Modifying, and Removing Rows”, Chapter 11: “Creating Tables”

**Part 1. SQL Developer Connection (20 points)**

If you are not working in a CDM lab, download the CDMOracleTutorial.pdf from D2L and make sure you can access CDM’s Oracle database.

A. Create a new connection to the CDM Oracle instance. Name it **YOURLASTNAME355**.

B. Run the following SQL:

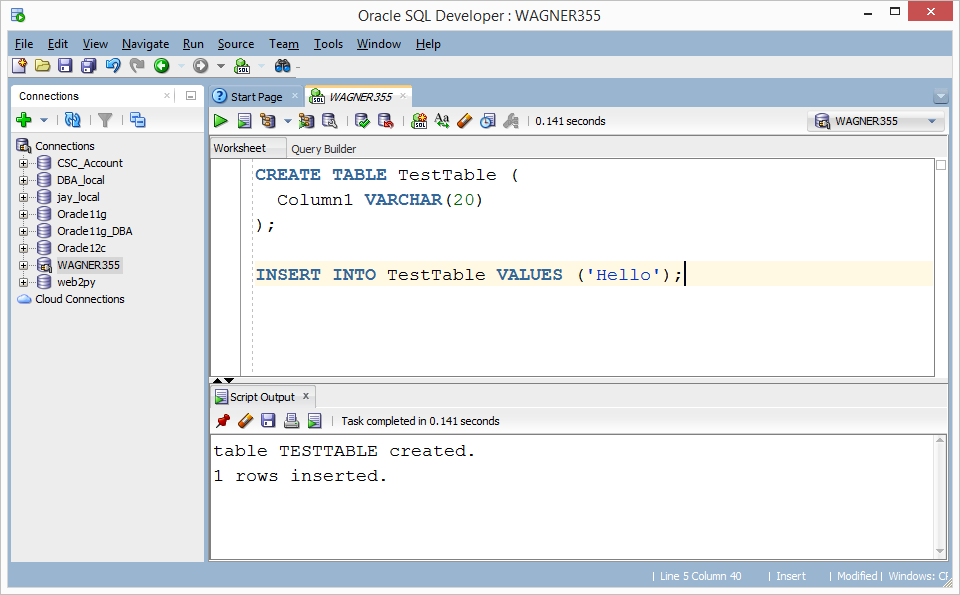
CREATE TABLE TestTable (

Column1 VARCHAR(20)

);

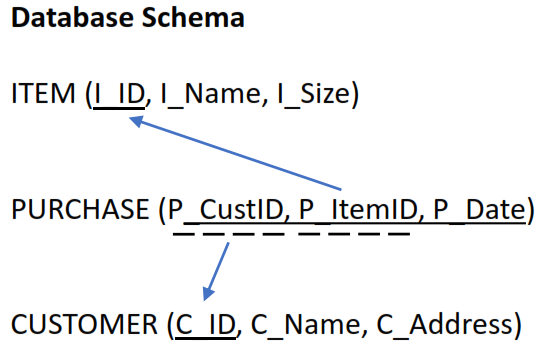
INSERT INTO TestTable VALUES ('Hello');

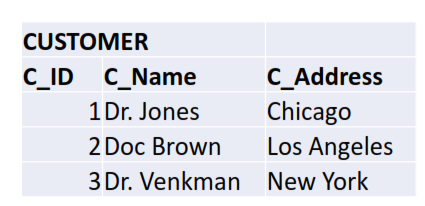
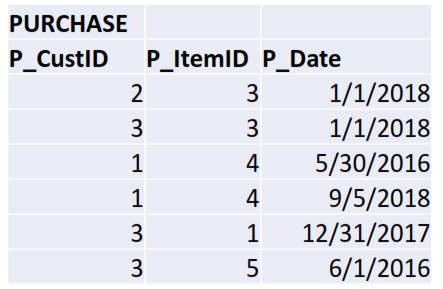
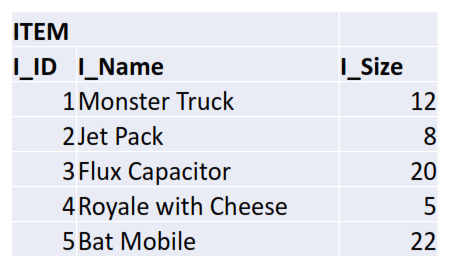
Submit a single screen shot that shows your open connection, and the output from the SQL. Example:



**Part 2. Relational Model (32 points)**

Answer all four questions given the following database schema and relational database instance containing three relations:





A. List the primary keys for each table.

ITEM: I\_ID

PURCHASE: P\_CustID, P\_ItemID, P\_Date

CUSTOMER: C\_ID

B. List the foreign keys for each table.

ITEM: None

PURCHASE: P\_CustID, P\_ItemID

CUSTOMER: None

C. For each table, construct a tuple that does not violate any constraints.

ITEM VALUES (6, “Computer”, 32);

PURCHASE VALUES (3, 3, 1/21/2017);

CUSTOMER VALUES (4, “Dr. Robotnik”, “Miami”);

D. Which records can be removed from ITEM without violating any constraints?

**Part 3. SQL DDL (48 points)**

Write a script to do the following. (Do this one part at a time, testing the partial results by inspecting the tables in SQLDeveloper, and not going on the next part until you have the previous part working.)

**A.** Define the following database, representing hotels, travelers who may stay in the hotels, and reservations that the travelers make at hotels.

  TRAVELER(ID, FirstName, LastName, Phone)

HOTEL(ID, HotelName, City)

RESERVATION(TravelerID, HotelID, StartDate, EndDate, Confirmation)

Traveler IDs should be exactly nine characters long, hotel IDs should be exactly four characters long, and phone numbers should be exactly ten characters long; reservation confirmation codes should be at most ten characters long. You may decide on appropriate maximum lengths for names of customers, hotels, and cities.

Define the primary keys as indicated in the schemas above. Also define TravelerID in RESERVATION to be a foreign key referencing ID in TRAVELER, and HotelID in RESERVATION to be a foreign key referencing ID in HOTEL. (Thus you will have to create TRAVELER and HOTEL before RESERVATION.) We can’t assume that the confirmation codes are unique, since they are supplied by the hotels when reservations are made we can’t be sure that different hotels won’t use the same codes. (All of the confirmation codes supplied by a single hotel will be distinct, but verifying that would require a more complicated constraint than we can implement right now…)

In order to avoid conflicts, start your script with DROP TABLE commands for all three tables (since RESERVATION contains foreign keys, you have to drop it first). Run your script and look at the columns and constraints of each table to verify that they have been created correctly.

**B.** Populate the HOTEL table with names and cities of at least four actual hotels anywhere in the world (you can make up the IDs, as they are internal to the database). Then populate the TRAVELER table with information for yourself and at least two of your friends. (For privacy reasons, IDs and phone numbers should be made up.) Look at the data in each table to verify that they have been populated correctly.

**C.** Next, insert at least five records into the RESERVATION table. There should be at least one record for each traveler, and at least one traveler should have two or more records in the RESERVATION table. Be sure that each traveler is staying at only one hotel at a time – that is, a traveler’s reservation at a second hotel can’t begin before their previous reservation at the first hotel ends. (This particular constraint – insuring that reservations do not overlap – is also too complicated for us to implement right now.) Every record in RESERVATION must have a starting date, but the ending date can be NULL for the traveler’s last reservation in the table. Look at the data in the table to verify that it has been populated correctly.