1. **Relational Schema**

Create a relational database schema consisting of the four relation schemas representing various entities recorded by a furniture company.

Write CREATE TABLE statements for the following four relation schemas. Define all necessary attributes, domains, and primary and foreign keys.

Customer(CustomerID, Name, Address)

FullOrder(OrderID, OrderDate, CustomerID)

Request(OrderID, ProductID, Quantity)

Product(ProductID, Description, Finish, Price)

You should assume the following:  Each CustomerID is a number with at most three digits, each OrderID is a number with at most five digits, and each ProductID is a number with at most two digits.  Each Name is at most 25 characters long, and each Address is at most 50 characters long. Each Description is at most 28 characters long, and each Finish is at most 10 characters long. Quantity and Price are integer and float respectively.

In this schema, primary and foreign keys are not marked. Identify primary and foreign keys and other custom constraints. Declare PK and FK in your CREATE TABLE statement and custom constraints as follows:

* Each Price is a non-negative value (zero is allowed) representing a price in dollars and cents that cannot exceed 999.99.
* Each Quantity is a positive whole number between 1 and 100, inclusive.

Populate the database as follows:

**CUSTOMER:**

**CUSTOMERID NAME ADDRESS**

**---------------------- ------------------------------ --------------------**

**2 CASUAL FURNITURE PLANO, TX**

**6 MOUNTAIN GALLERY BOULDER, CO**

**FULLORDER:**

**ORDERID ORDERDATE CUSTOMERID**

**---------------------- ------------------------- ----------------------**

**1006 24-MAR-10 2**

**1007 25-MAR-10 6**

**1008 25-MAR-10 6**

**1009 26-MAR-10 2**

**PRODUCT:**

**PRODUCTID DESCRIPTION FINISH PRICE**

**---------------------- ------------------------------ ---------- ----------**

**10 WRITING DESK OAK 425**

**30 DINING TABLE ASH 600**

**40 ENTERTAINMENT CENTER MAPLE 650**

**70 CHILDRENS DRESSER PINE 300**

**REQUEST:**

**ORDERID PRODUCTID QUANTITY**

**---------------------- ---------------------- ----------------------**

**1006 10 4**

**1006 30 2**

**1006 40 1**

**1007 40 3**

**1007 70 2**

**1008 70 1**

**1009 10 2**

**1009 40 1**

Remember to insert the records containing primary key values before inserting the records containing the foreign keys that reference those values (or your insert statements will fail).

Display the contents of each table by adding four statements of the form SELECT \* FROM *TABLE\_NAME* ; to the end of your script, where TABLE\_NAME is substituted by the four table names above.

In order to avoid conflicts, include DROP TABLE commands for all four tables before your CREATE TABLE statements.  Either drop tables containing foreign keys before the tables containing the referenced primary keys, or use CASCADE CONSTRAINTS.  Verify that the tables have been defined correctly before going on to the next step.

Run the complete script.  Be sure that that the contents of the four tables are displayed correctly by your script.

Submit a Q1.sql file.

**Note:**

You must code this schema using ‘Create Table’ statements . You should not use any other tool except Oracle SQLDeveloper. **No points will be given for auto-generated schemas using other 3rd party tools or web-based editors.**

1. **Inserting Real Data**

The objective of this assignment is to give you a feel about inserting real data into a database.

Download the Loan.csv file from D2L. Data in Loan.csv is acquired from a real Insurance company.

1. Identify the schema from this file. Write a ‘Create Table’ statement for loans. You must name the table as ‘Loan’. Attribute names must match the column names in the .csv file. Declare appropriate datatypes based on data and add a PK constraint.

Save the `Create Table` statement in a Q2.sql file.

2. Now, upload data in Loan.csv into Loan table. There are 149 rows to upload. Inserting by hand is a waste of time, unless you have extreme patience. Write a function in your favorite programming language to open and read this CSV file line by line and generate statements of the form

Insert Into Loan Values (Val1, ‘Val2’, ‘Val3’, Val4,…);

Where Val1 and Val4 are numeric and Val2 and Val3 are strings.

3. Copy/paste ‘Insert Into’ statements generated from your program to Q2.sql file and upload data into the table.

Note: You may encounter errors and will have to fix some errors.

4. After inserting all data, copy/paste the following query into Q2.sql and run it:

select count(\*) from Loan;

5. Submit (i) Q2.sql, (ii) a Q2.pdf consisting of screenshot of your query and result.

**Bonus:** The Loan.csv considers all data values as strings. However, some values are indeed numbers and dates. Map a string to a NUMBER or DATE data type in your program and insert into a Loan table declared accordingly.

Note:

1. For this example, it is strongly recommended to go from a working-to-working condition. First, write the create table statement. Then use the first line of data values to manually write an `Insert Into ` statement in SQLDeveloper. From that determine a pattern, based on which you should write your program to generate several `Insert Into` statements.
2. Copy and paste the output from your program in SQLDeveloper to see if your `Insert Into` statements are correct. If correct, data will be inserted.
3. No attribute is missing a value, so your program need not check for NULLs.
4. For more information on screenshots read Resources FAQ on the website. Failure to comply with Screenshot instructions will result in loss of points.