**CSCI 4125/5125 Course Project**

**Data Models and Database Systems**

**Fall 2023**

**Course Project**

**Phase 4: Introductory SQL, DDL & Inserts (9/15)**

**Due: Sunday, 9/24 @ 11:59pm**

**Reading:** Silberschatz Chapters 3.1 – 3.4, 3.6 – 3.9, 4.3 – 4.5

**Submission Guidelines:**

1. This assignment is worth 50 points for all students.

2. It is your responsibility to make sure all files are readable and submitted on time.

**Submission:**

- Task 1 requires you to submit a single .sql file worth 30 points.

- Task 2 requires you submit five .sql files and another file in the format of your choice (e.g., Word, PDF, text, etc.) worth 20 points.

**Task 1. SQL DDL (30 points)**

Write a SQL DDL script to create your tables from Phase 3. Use the following guidelines and relational schema:

* At the top of your script include a drop table command for each of your tables. Note that you must pay attention to referential integrity when considering the order to drop tables.
* Column names should match the attributes in the relational schema. This will allow for consistency in our queries later.
* Columns must use reasonable domains based on the data in the included .txt files.
* All primary keys must be declared.
* All foreign keys must be declared.
* Run your SQL script and debug any errors.
* **Submit:** a single .sql file named store\_schema.sql.

A diagram of a product

Description automatically generated

**Task 2. Populating the database (20 points)**

Using your Java program from Phase 3, generate INSERT statements for the six .txt files included with this document. You should generate a total of six SQL scripts containing INSERT statements. Use the following guidelines to submit your work:

* Data types must be properly formatted, e.g., strings must use single quotes, dates must use the correct format.
* Some strings might contain a single quote. Make sure the single quote appears in the value. This can be easily addressed with the replace() method.
* Each script should include a commit (i.e., “commit;”) at the end of the file. You can modify your Java program to simply write that before you close the outfile.
* Name each file [table name].sql.
* Run each script in SQL Developer. Remember that you can run SQL scripts using **@[path]\[file].sql**
* You will need to run the INSERT’s in the proper order that does not violate referential integrity.
* Read the output that is generated. If there are any errors, it is up to you to fix them. Errors to watch out for include, improper syntax, violating domain constraints, violating primary key constraints, and violating referential integrity.
* If you want to verify that all records were populated, you can run “SELECT COUNT(\*) FROM [table name]” in SQL Developer. This should return 20 for Customer, 15 for Address, 20 for Product, 12 for Review, 50 for Orders, and 100 for Lineitem.
* **Submit:** Six .sql files containing your insert statements. Name them [table].sql. Also submit in any file format (e.g., txt, doc, pdf, etc) you want that explains the order in which you ran your scripts (that doesn’t violate referential integrity).

/\*

KEY

CSCI 4125 Fall 2023

Phase4

\*/

DROP TABLE REVIEW CASCADE CONSTRAINTS;

DROP TABLE LINEITEM CASCADE CONSTRAINTS;

DROP TABLE ORDERS CASCADE CONSTRAINTS;

DROP TABLE ADDRESS CASCADE CONSTRAINTS;

DROP TABLE CUSTOMER CASCADE CONSTRAINTS;

DROP TABLE PRODUCT CASCADE CONSTRAINTS;

CREATE TABLE PRODUCT (

P\_ID CHAR(3),

P\_Name VARCHAR(50) NOT NULL,

P\_Price DECIMAL(5, 2) NOT NULL,

CONSTRAINT ProductPK PRIMARY KEY (P\_ID)

);

CREATE TABLE CUSTOMER (

C\_ID CHAR(3),

C\_Name VARCHAR(50) NOT NULL,

C\_DateOfBirth DATE NOT NULL,

C\_ReferrerID CHAR(3),

CONSTRAINT CustomerPK PRIMARY KEY (C\_ID),

CONSTRAINT CustomerFK FOREIGN KEY (C\_ReferrerID) REFERENCES CUSTOMER(C\_ID)

);

CREATE TABLE ORDERS (

O\_OrderNumber CHAR(5),

O\_OrderDate DATE NOT NULL,

O\_CustID CHAR(3),

CONSTRAINT OrdersPK PRIMARY KEY (O\_OrderNumber),

CONSTRAINT OrdersFK FOREIGN KEY (O\_CustID) REFERENCES CUSTOMER(C\_ID)

);

CREATE TABLE LINEITEM (

L\_OrderNumber CHAR(5),

L\_ProductID CHAR(3),

L\_Quantity INT NOT NULL,

CONSTRAINT LineItemPK PRIMARY KEY (L\_OrderNumber, L\_ProductID),

CONSTRAINT LineItemOrderFK FOREIGN KEY (L\_OrderNumber) REFERENCES ORDERS(O\_OrderNumber),

CONSTRAINT LineItemProductFK FOREIGN KEY (L\_ProductID) REFERENCES PRODUCT(P\_ID)

);

CREATE TABLE REVIEW (

R\_ProductID CHAR(3),

R\_Number INT,

R\_Rating INT NOT NULL,

R\_Text VARCHAR(250),

CONSTRAINT ReviewPK PRIMARY KEY (R\_ProductID, R\_Number),

CONSTRAINT ReviewProductFK FOREIGN KEY (R\_ProductID) REFERENCES PRODUCT(P\_ID)

);

CREATE TABLE ADDRESS (

A\_CustID CHAR(3),

A\_Address VARCHAR(300) NOT NULL,

CONSTRAINT AddressPK PRIMARY KEY (A\_CustID, A\_Address),

CONSTRAINT AddressCustomerFK FOREIGN KEY (A\_CustID) REFERENCES CUSTOMER(C\_ID)

);

**Explains The Order In Which I Ran My Scripts**

**A diagram of a product

Description automatically generatedPRODUCT:** This table has no foreign keys, which means it does not depend on any data in other tables. So logically I can import this table first as it won’t cause errors due to foreign key constraints.

**CUSTOMER:** This table has a foreign key ReferrerID, which refers to an ID in the same table.

**ORDERS:** This table has a foreign key CustID, which refers to the CUSTOMER table. This means that every order is associated with a customer. Before importing the ORDERS table, I must ensure that all relevant CustIDs already exist in the CUSTOMER table, otherwise foreign key constraints will be violated.

**ADDRESS:** This table also has a foreign key CustID, which refers to the CUSTOMER table. I must ensure that all CustIDs already exist in the CUSTOMER table.

**LINEITEM:** This table has two foreign keys, OrderNumber and ProductID, which reference the **ORDERS** and **PRODUCT** tables respectively. This means that each line item is associated with an order and a product. Before importing data, I must ensure that relevant records in both tables already exist.

**REVIEW:** This table has a foreign key ProductID, which refers to the PRODUCT table. This means that each review is associated with a product. Therefore, before importing the REVIEW table, I must ensure that all ProductIDs already exist in the PRODUCT table.