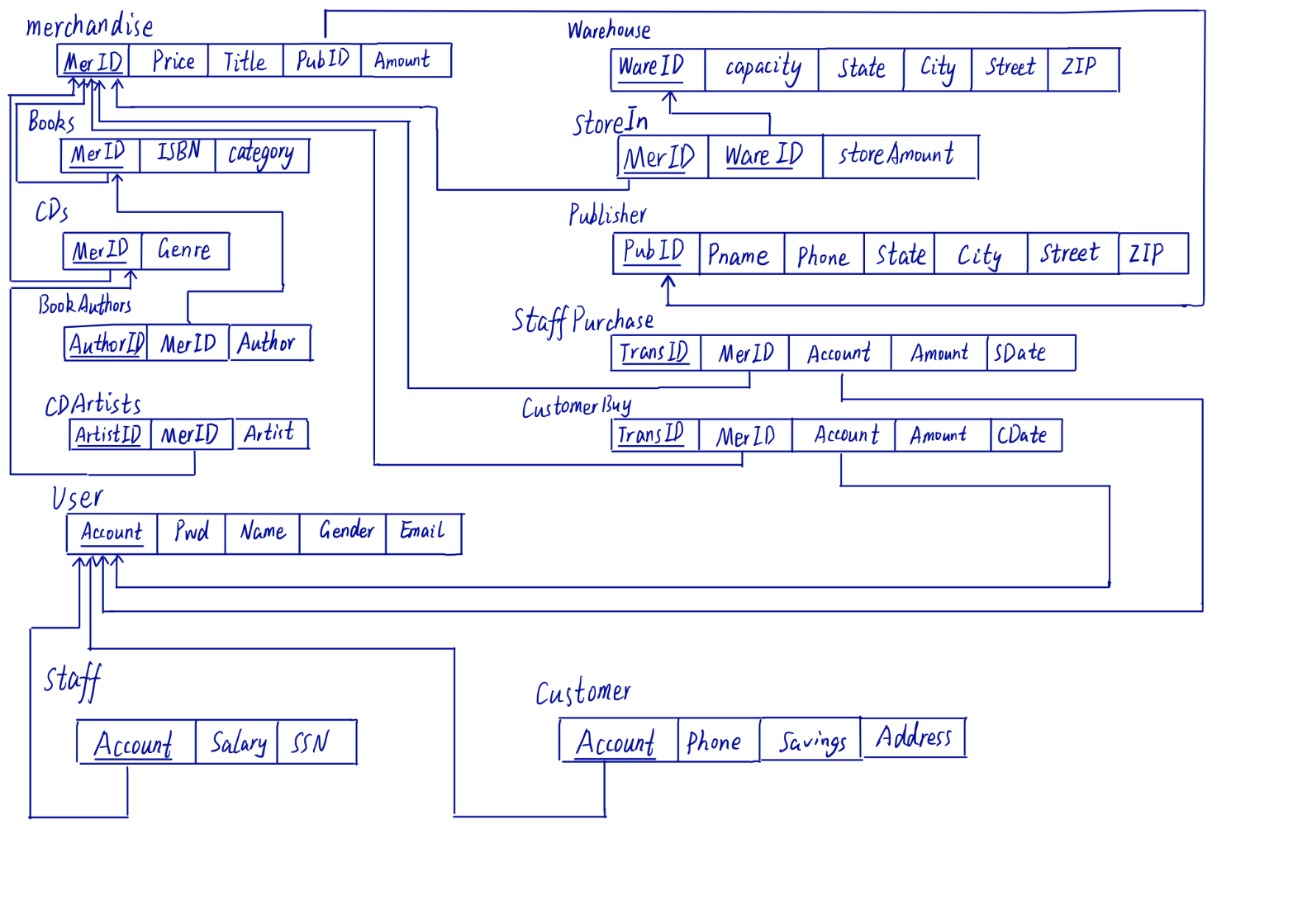
**CSE 3241 Project Checkpoint 04 – Functional Dependencies and Normal Forms**

Names Date

In a **NEATLY TYPED** document, provide the following:

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 03. **If you were instructed to change the model for Project Checkpoint 03, make sure you use the revised versions of your models.**

According to advice from feedbacks, we assign AuthorID and ArtistID. User Schema is also split into two separate schemas: Customer and Staff.



1. For each relation schema in your model, indicate the functional dependencies. Think carefully about what you are modeling here - make sure you consider all the possible dependencies in each relation and not just the ones from your primary keys. For example, a customer’s credit card number is unique, and so will uniquely identify a customer even if you have another key in the same table (in fact, if the customer can have multiple credit card numbers, the dependencies can get even more involved).

MerID->(Price, title, PubID, Amount)

MerID->(ISBN, category)

MerID->(Genre)

AuthorID->(MerID, Author)

AuthorID->(MerID, Artist)

WareID->(Capacity, State, City, Street, ZIP)

(MerID, WareID)->storeAmount

PubID->(Pname, Phone, State,City, Street, ZIP)

TransID->(MerID, Account, Amount, SDate)

TransID-> (MerID, Account, Amount, CDate)

Account->(Pwd, Name, Gender, Email)

Account->(Salary, SSN)

Account->(Savings, Phone, Address)

1. For each relation schema in your model, determine the highest normal form of the relation. If the relation is not in 3NF, rewrite your relation schema so that it is in at least 3NF.

The whole relations are 3NF.

1. For each relation schema in your model that is in 3NF but not in BCNF, either rewrite the relation schema to BCNF or provide a short justification for why this relation should be an exception to the rule of putting relations into BCNF.

The whole relations are BCNF because the primary key is not dependent on other attributes.

1. For your database, propose at least two interesting views that can be built from your relations. These views must involve joining at least two tables together each and must include some kind of aggregation in the view. Each view must also be able to be described by a one or two sentence description in plain English. Provide the code for constructing your views along with the English language description of what the view is supposed to be providing.
2. First view:

This view is to display the total money which each customer spends on books.

The SQL is:

CREATE VIEW ConsumptionOnBooks(Customer, Consumption)

AS SELECT U.Name, Round(sum(M.Price), 2) AS Money\_Spent\_on\_Books

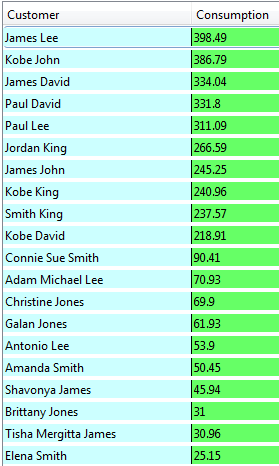
FROM User U, Merchandise M, Books B, CustomerBuy C

WHERE U.Account = C.Account and C.MerID = B.MerID and B.MerID = M.MerID

Group by U.account

ORDER BY 2 DESC;

The result is:



1. Second view:

Motivation: Represent the each customer’s preference for books.

SQL:

CREATE VIEW CustomerPreference(Name, Category)

AS SELECT U.Name, F.Category

FROM (SELECT A.Account, A.Category, Max(A.CategoryNum)

FROM (SELECT C.Account, B.Category, Count(\*) as CategoryNum

FROM Books as B, CustomerBuy as CB, Customer as C

WHERE B.MerID = CB.MerID AND CB.Account = C.Account

GROUP BY C.Account, B.Category) As A

GROUP BY A.Account) As F, User As U

WHERE F.account = U.Account;

Result:

(Continued on the next page)

