RDMS Design

John McDonald, [redacted]

{jamcdon3,[redacted]@ncsu.edu

0. Assumptions **3**

1. Problem Statement **3**

2. Intended Users **5**

3. Five Main Entities **5**

4. Task and Operations - Realistic Situations **5**

5. Application Program Interfaces **6**

6. Description of Views **10**

7. Local E/R Diagrams **11**

Admin View 11
Cashier View 12
Registration View 13
Billing View 14
Warehouse View 15

8. Description of Local E/R Diagrams **16**

9. Local Relational Schemas **17**

Manager View 17
Cashier View 17
Registration View 18
Billing View 18
Warehouse View 18

10. Local Schema Documentation **19**

11. Assumptions: **21**

12. Global Relational Database Schema **22**

13. Design for Global Schema **25**

Design Decisions for Global Schema: 25
Integrity Constraints 25

14. Base Relations **28**

Create Statements 28
Select Statements 33

15. SQL Queries **37**

15.1: Interactive SQL Queries 37
Information Processing 37
Maintaining Inventory Records 39
Maintaining Billing and Transaction Records 39
Reports 41
15.2 Explain Directive 44
15.3: Query Correctness 45

0. Assumptions

1. Memberships are assumed to be free such that there is no cost associated with them.
2. The same product can be available at multiple stores for different prices.
3. Discounts on products are only given in percentages.
4. There may be at most one promotion or discount active on a product at a specific store at any given time.
5. We assume no taxes for purchases, and total PricePaid is tracked in transaction contains, such that no extra calculations are needed.
6. A product is assumed to be only available from one supplier.
7. Transfer of products between stores is assumed to be automatic and immediate.
8. Returned inventory is handled by warehouse staff, while the monetary transaction is handled by cashiers.
9. A product can be returned at any store, and is assumed to be instantly transferred back to the store it was purchased from.
10. We do not track the location of a return, only the location of the initial purchase.

1. Problem Statement

We are tasked with designing a database system for a wholesale-store chain, WolfWR, to manage inventory, supply chain information, customer memberships, monetary transactions, and staff information. Primary tasks to be performed by users of the database system are information processing, maintaining inventory records, maintaining billing and transaction records, and generating reports that pertain to sales, inventory, and growth. The full lifecycle of a product should be captured in the database from the supplier to the warehouse to the customer that purchases it.

Databases offer benefits over using a set of files where a large number of users might want to alter data concurrently. Filesystems are inherently not atomic and performing thread-safe operations is the responsibility of the programmer. Filesystem locking will also prevent more than 1 user from accessing and modifying the data at once, which would prevent any sort of efficiency in the chain. Database management systems also inherently allow for data to be structured and related to each other much easier than a set of files would. Lastly, a database has the benefit of allowing data to be updated atomically so that data in the database accurately reflects its real-life counterpart rather than going out of sync.

2. Intended Users

1. (staff) **Cashiers:** who run register and transactions for customers
2. (staff) **Registration Staff:** who manage memberships for customers
3. (staff) **Billing Staff:** who generate bills for orders from suppliers, manage rewards for memberships, oversee transactions from cashiers, and have visibility to all billing related reports.
4. (staff) **Warehouse Staff:** who manage physical transfer of product from supplier shipments, manage supply of returns, and manage transfer between stores
5. (staff) **Managers:** who have access to all information in the stores. They can view and edit all information across all staff members. They have complete control of the system.

3. Five Main Entities

1. **Staff:** staff ID, name, age, phone, email, tenure, title, department, store
2. **Customers:** customer ID, first name, last name, phone, email, address, membership information
3. **Products:** store ID, product ID, name, quantity, price, cost, expiration date, production data, discount information
4. **Transactions:** transaction ID, customer ID, cashier ID, transaction date, transaction items information
5. **Shipments:** shipment ID, shipment date, shipment supplier information, shipment items information

4. Task and Operations - Realistic Situations

Situation 1: A customer enters the store and asks a sales representative to sign up for a membership. He enters her information into their system and creates a new membership. He repeats the information back to her to confirm that it's correct. The phone number he entered was wrong, so he corrected it.

Situation 2: The customer from the previous scenario had signed up for a platinum membership. The total amount of purchases she made throughout the year added up to about $5,000. At the end of the year, the billing clerk created and sent out a reward check of about $100 to her.

5. Application Program Interfaces

Information Processing

1. enterCustomerInfo(CustID, FirstName, LastName, Email, Phone, Address, MemID)
    a. return confirmation
2. updateCustomerInfo(CustID, FirstName, LastName, Email, Phone, Address, MemID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
3. deleteCustomerInfo(CustID, FirstName, LastName, Email, Phone, Address, MemID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
4. enterStaffInfo(StaffID, Tenure, Address, Email, Name, Phone, Age, Title, StoreID)
    a. return confirmation
5. updateStaffInfo(StaffID, Tenure, Address, Email, Name, Phone, Age, Title, StoreID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
6. deleteStaffInfo(StaffID, Tenure, Address, Email, Name, Phone, Age, Title, StoreID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
7. enterSupplierInfo(SupplierID, Name, Location, Phone, Email)
    a. return confirmation
8. updateSupplierInfo(SupplierID, Name, Location, Phone, Email)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
9. deleteSupplierInfo(SupplierID, Name, Location, Phone, Email)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
10. enterStoreInfo(StoreID, MgrID, Address, Phone)
    a. return confirmation
11. updateStoreInfo(StoreID, MgrID, Address, Phone)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
12. deleteStoreInfo(StoreID, MgrID, Address, Phone)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
13. enterProductInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID, SupplierID, Cost)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
14. updateProductInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID, SupplierID, Cost)

    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
15. deleteProductInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID, SupplierID, Cost)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
16. enterTransactionInfo(TransID, Date, StoreID, CustID, StaffID)
    a. return confirmation
17. updateTransactionInfo(TransID, Date, StoreID, CustID, StaffID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
18. deleteTransactionInfo(TransID, Date, StoreID, CustID, StaffID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
19. enterDiscountInfo(DiscountID, Percent, StartDt, EndDt, ProductID, StoreID)
    a. return confirmation
20. updateDiscountInfo(DiscountID, Percent, StartDt, EndDt, ProductID, StoreID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
21. deleteDiscountInfo(DiscountID, Percent, StartDt, EndDt, ProductID, StoreID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
22. enterMembershipInfo(MemID, StoreID, StaffID, SignUpDt, LevelID, Length)
    a. return confirmation
23. updateMembershipInfo(MemID, StoreID, StaffID, SignUpDt, LevelID, Length)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
24. deleteMembershipInfo(MemID, StoreID, StaffID, SignUpDt, LevelID, Length)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed

Maintaining Inventory Records

1. getInventoryRecords(ProductID)
    a. returns list of store products in the inventory (ProductID, Name, Quantity, Price, Production, ExpireDt)
2. addInventoryRecords(ProductID, Name, Quantity, Price, Production, ExpireDt)
    a. returns True if added successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
3. updateInventoryRecords(ProductID, Name, Quantity, Price, Production, ExpireDt)
    a. returns True if updated successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
4. deleteInventoryRecords(ProductID, Name, Quantity, Price, Production, ExpireDt)
    a. returns True if deleted successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed

5. getReturnInfo(ProductID)
    a. returns list of store products that have been returned
6. enterReturnInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
7. updateReturnInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. returns True if updated successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
8. deleteReturnInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. returns True if deleted successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
9. getShipmentInfo(ProductID)
    a. returns list of store products that have been shipped
10. enterShipmentInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
11. updateShipmentInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. returns True if updated successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
12. deleteShipmentInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. returns True if deleted successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
13. enterProductTransferInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. return confirmation
    b. If NULL for any of the field attributes then they will not be changed
14. updateProductTransferInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. returns True if updated successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
15. deleteProductTransferInfo(ProductID, Name, Quantity, Price, Production, ExpireDt, StoreID)
    a. returns True if deleted successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed

Maintaining Billing and Transaction Records

1. checkAvailabililty(StoreID, ProductID)
    a. returns True if available
2. createSupplierBills(SupplierID, Name, Location, Phone, Email)
    a. returns True if the Supplier’s bills are successfully added and False if not

    b. If NULL for any of the field attributes then they will not be changed
3. maintainSupplierBills(SupplierID, Name, Location, Phone, Email)
    a. returns True if the Supplier's billing account is successfully edited correctly
    b. If NULL for any of the field attributes then they will not be changed
4. getBillingAccount(SupplierID)
    a. returns supplier billing information
5. getTransaction(TransID)
    a. returns list of transactions(TransID, Date, Amount)
6. calculateTransactionPrice(TransID)
    a. returns total price for each transaction
7. getItemDiscountStatus(StoreID, ProductID)
    a. returns True if an item is on sale and False if not
    b. If NULL for any of the field attributes then they will not be changed
8. getDiscount()
    a. returns list of possible discounts (Percent, StartDt, EndDt, ProductID, StoreID)
    b. If NULL for any of the field attributes then they will not be changed
9. addDiscount(Percent, StartDt, EndDt, ProductID, StoreID)
    a. returns True if the discount is added successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
10. updateDiscount(Percent, StartDt, EndDt, ProductID, StoreID)
    a. returns True if the discount is updated successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed
11. getRewardChecks(CustID)
    a. returns list of reward checks (CustID, Year, RewardAmt)
    b. If NULL for any of the field attributes then they will not be changed
12. addRewardChecks(CustID, Year, RewardAmt)
    a. returns True if added successfully and False if not
    b. If NULL for any of the field attributes then they will not be changed

Reports

1. getHistory(MembershipID, SignUpDt)
    a. returns records for members whose date is after SignUpDt
2. getCustomers(StoreID)
    a. returns information on the store's customers (FirstName, LastName, Email, Phone, CustID, Address, MemID)
3. getStaffInfo(StoreID)
    a. returns employee information for each staff member, grouped by position
4. getSupplier()
    a. returns supplier information (SupplierID, Name, Location, Phone, Email)
5. getStocks(StoreID)
    a. returns merchandise stock report for each product at each store (Cost, Quantity, WarehouseID, ShipID, ProductID)
6. getSales(StoreID)
    a. returns sales growth report for a specific store for specific time period
7. getCustomerGrowth(CustID, StartDt, EndDt)

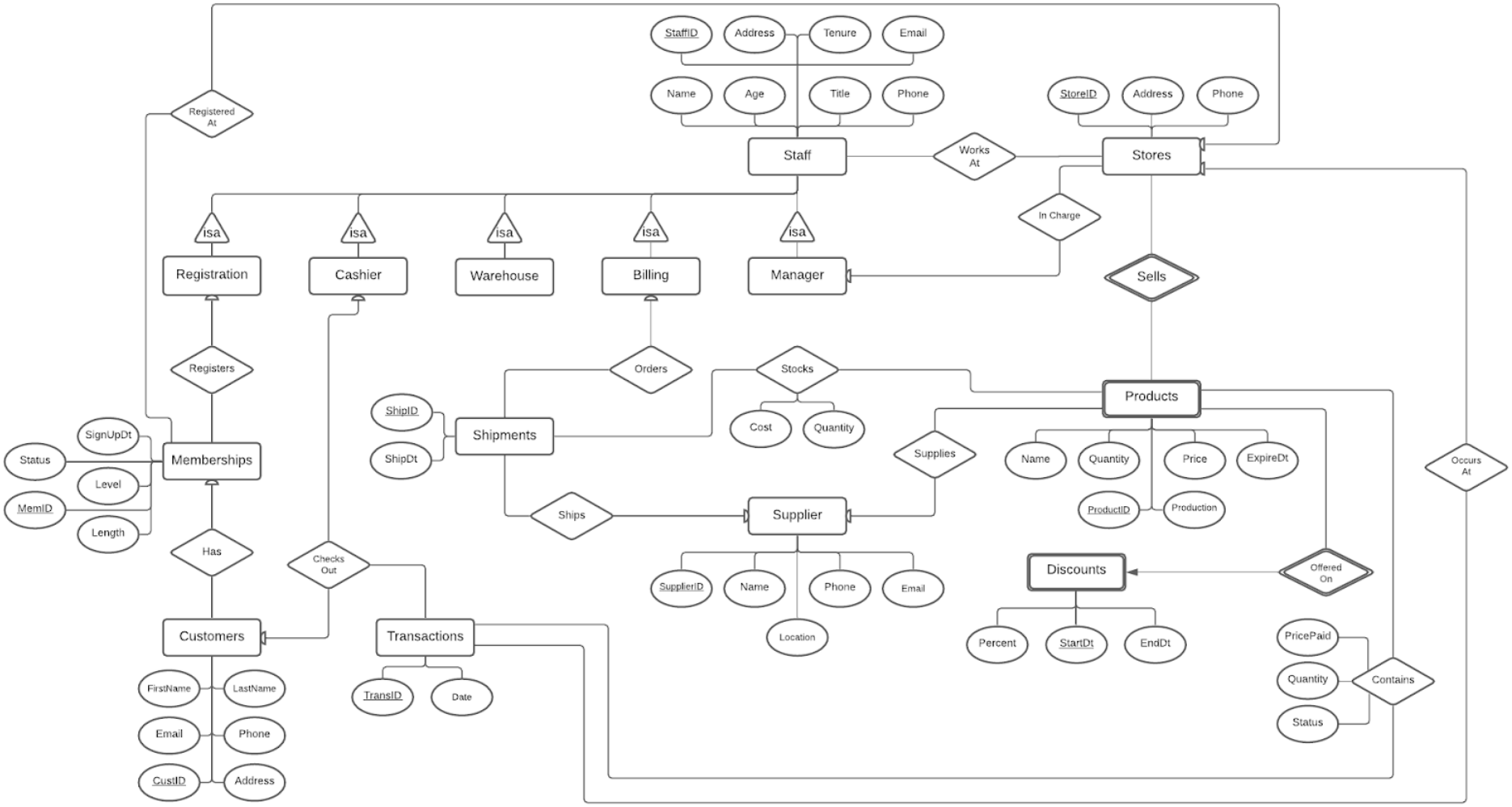
    a. returns customer growth by a specific time
8. getCustomerActivity(CustID, StartDt, EndDt)
    a. returns customer activity for a specific time period
9. getTotalSales(TransID, ProductID, StoreID, PricePaid, Quantity, Status, TransID, Date, CustID, StaffID)
    a. returns total sales for each product at each store over a specific time period

6. Description of Views

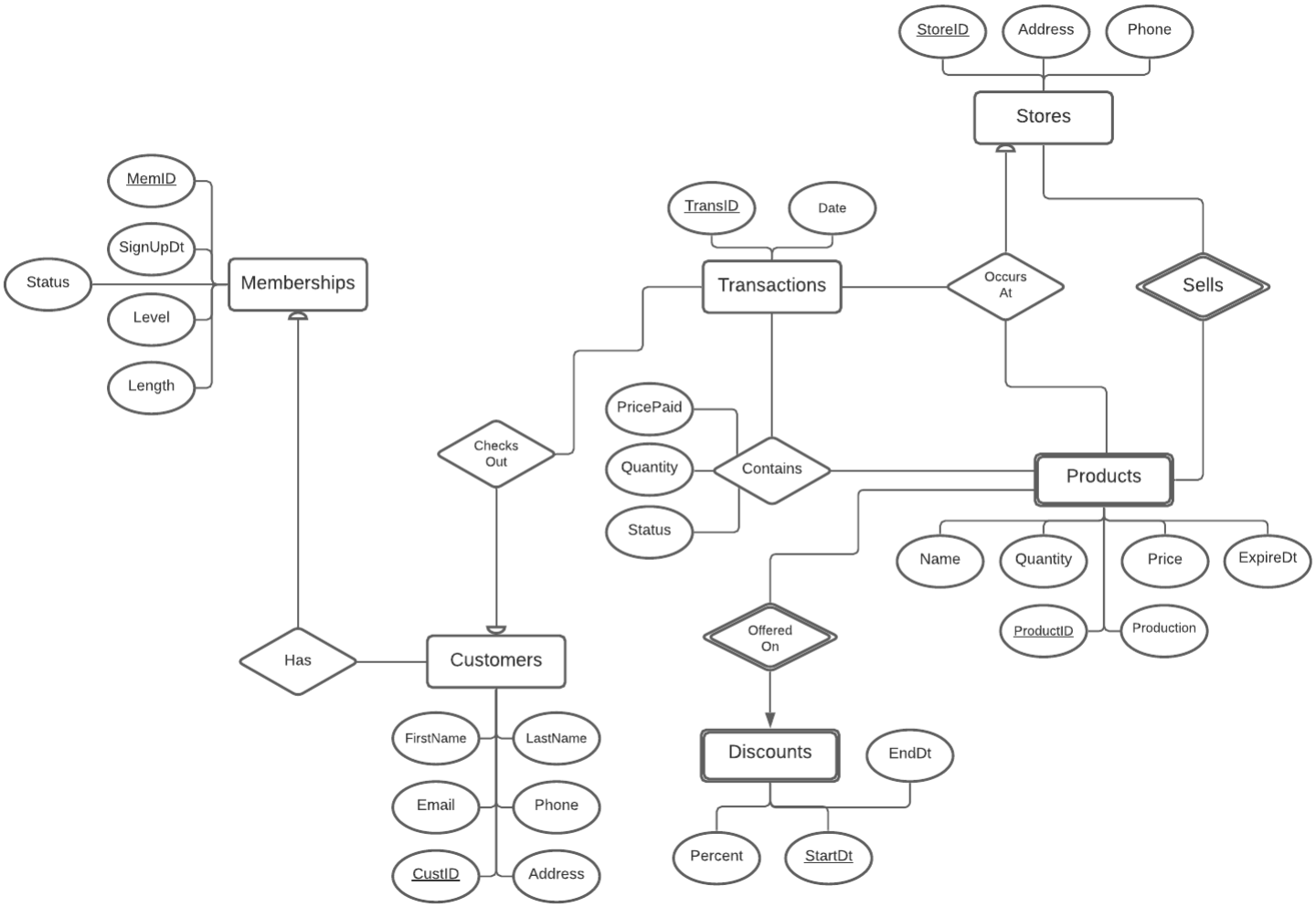
1. **Cashiers:** can view customer information and their associated membership levels. They have access to all front store operations and history. This allows them to manage purchases and returns, as well as notify customers if their memberships are expiring soon on checkout.
2. **Billing Staff:** Has high visibility into the overall E/R diagram as they need to keep track of purchases of both front and back store operations. They order the shipments as they are responsible for generating bills to the suppliers. They also keep track of reward checks that are sent out to members.
3. **Warehouse Staff:** has access to information regarding back store operations. They stock products from supplier shipments, and can edit product quantities for each store. They do not keep track of their own work via relationships to the ‘warehouse’ entity.
4. **Managers:** has access to all facets of the database. Can view all information about employees, customers, products, stores, and suppliers.
5. **Registration Staff:** has access to customer and membership information. Does not require information about other employees to perform their duties.

7. Local E/R Diagrams

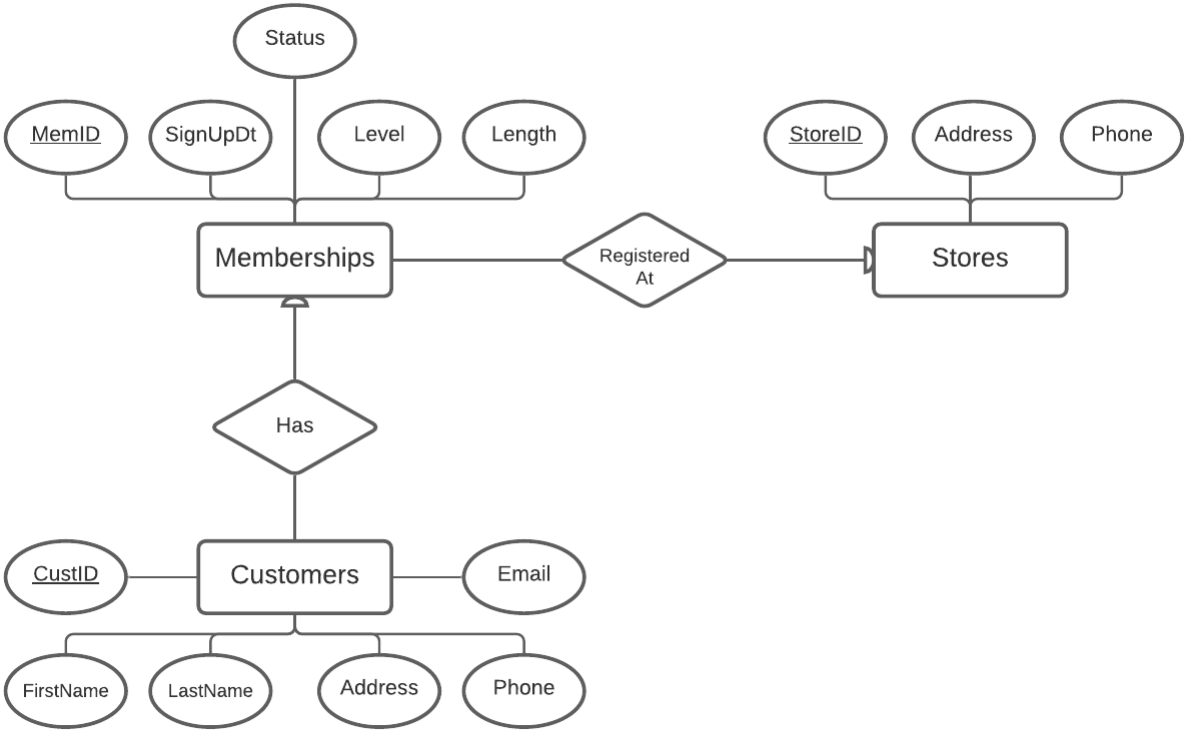
Admin View



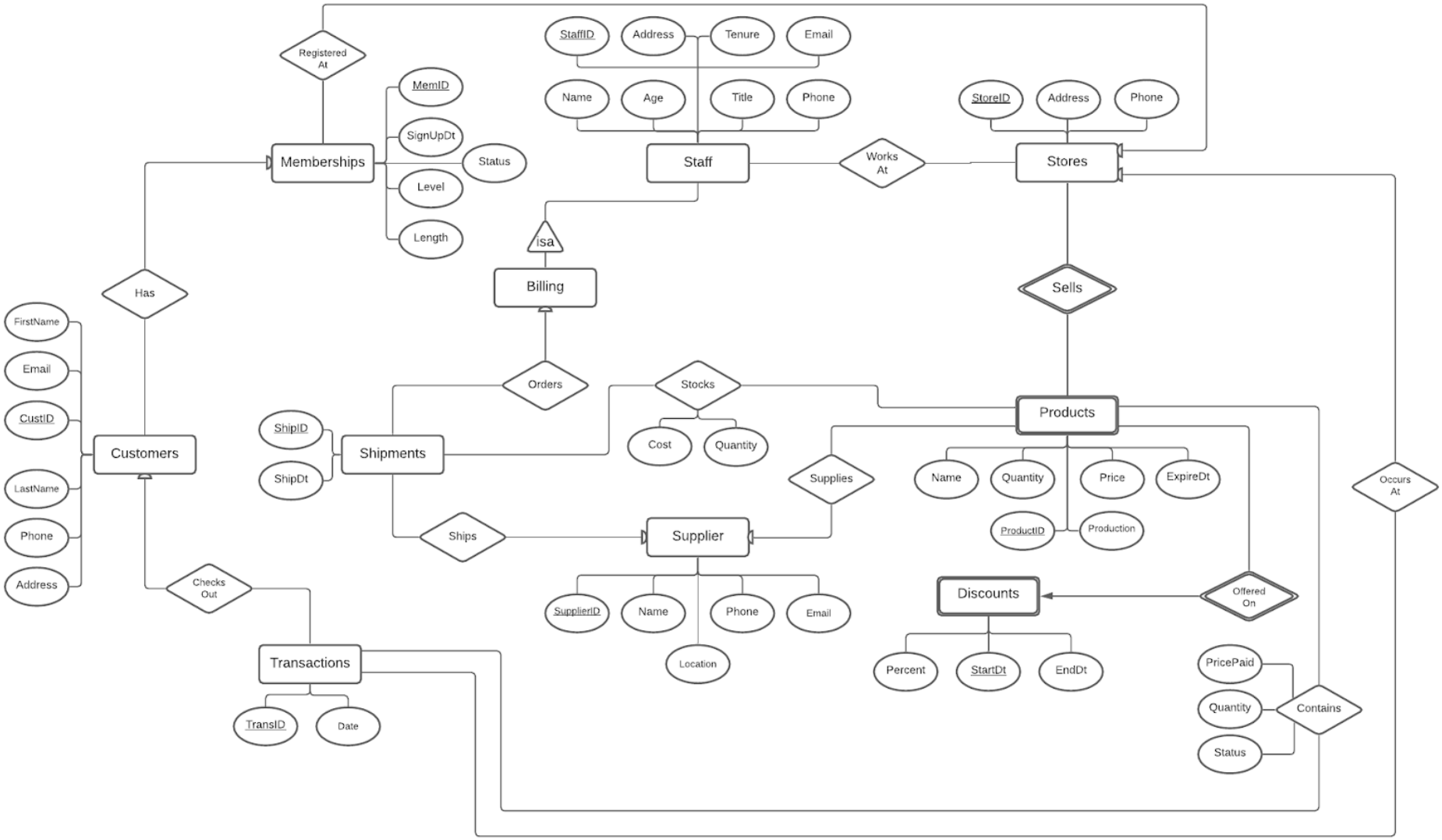
Cashier View



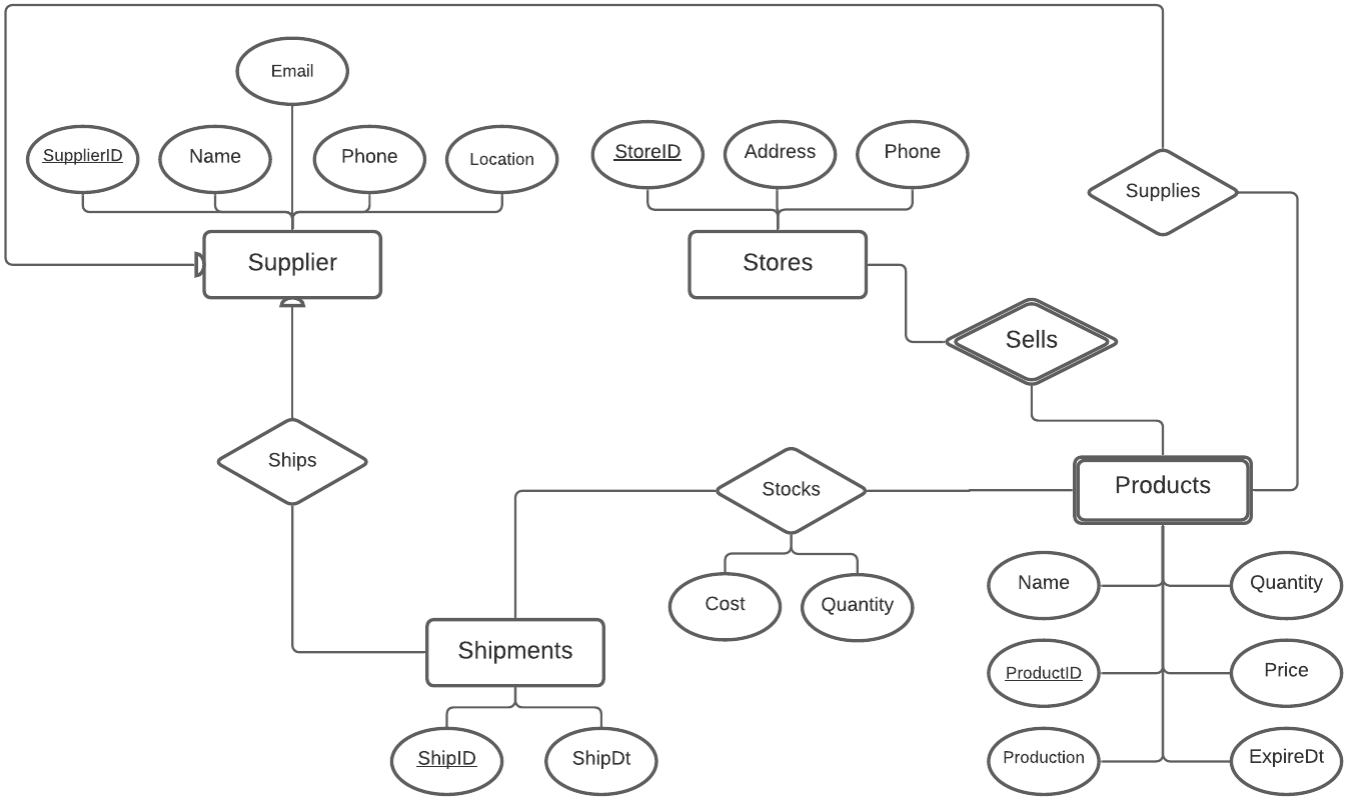
Registration View



Billing View



Warehouse View



8. Description of Local E/R Diagrams

1. Registration doesn't need to have access to staff membership
2. Registration only has visibility into current customers and memberships
3. Billing staff has access to staff information so they can raise inquiries about transactions
4. Billing staff have access to in store transactions as well as supply chain shipment information
5. Billing staff orders shipments, so that they can generate necessary bills for suppliers
6. Cashiers have visibility into in store operations to help with customer service and inquiries
7. Warehouse does not have access to other staff information, but does have access to warehouse entities. This is so they can read, write, and edit staff information relevant to the 'stocks' relationship.
8. Warehouse staff has access to back store operations, allowing them to handle incoming shipments, edit quantity information for products at stores, and thus transfer products between stores and update the database to reflect the same
9. For store information, title does not track the 'isa' relationship, but instead the level of employment, for example new hire, senior, mid-level, etc.
10. For products, a customer's buy price is tracked with the 'contains' relationship to transaction
11. For products, supplier ID is tracked through a relationship to supplier
12. For customers, club membership level and active status is tracked through the 'has' relationship.
13. Sign up information is managed with a membership entity that is related to registration staff id, customer id, and store id.
14. Rewards information for platinum members is not stored in the database, but is calculated at the application level, to account for transactions for members that occurred each year.
15. Transactions entity does not have amount, as PricePaid is tracked in the Contains relationship
16. Transactions keeps track of purchases vs returns using the 'Status' attribute
17. Transactions keeps track of different products in each purchase through the Contains relationship, as opposed to using product list which violates NF1
18. Supplier information is the same as prompt recommendations
19. Discount information only tracks percent discount and time frame, with a weak relationship to the product ID it applies for.

9. Local Relational Schemas

Manager View

Staff(StaffID, Address, Tenure, Email, Name, Age, Title, Phone, StoreID)
Registration(StaffID)
Cashier(StaffID)
Manager(StaffID)
Billing(StaffID)
Warehouse(StaffID)

Customers(FirstName, LastName, Email, Phone, CustID, Address, MemID)
Memberships(MemID, SignUpDt, Status, Level, Length, StoreID, StaffID)

Transactions(TransID, Date, StoreID, CustID, StaffID)
TransactionContains(TransID, ProductID, StoreID, PricePaid, Quantity, Status)

Stores(StoreID, Address, Phone, StaffID)

Products(Name, Quantity, ProductID, Price, Production, ExpireDt, StoreID, SupplierID)
Discounts(Percent, StartDt, EndDt, ProductID, StoreID)

Shipments(ShipID, ShipDt, SupplierID, StaffID)
Supplier(SupplierID, Name, Location, Phone, Email)
ShipmentStocks(Cost, Quantity, ShipID, ProductID, StoreID)

Cashier View

Stores(StoreID, Address, Phone, StaffID)

Products(Name, Quantity, ProductID, Price, Production, ExpireDt, StoreID, SupplierID)
Discounts(Percent, StartDt, EndDt, ProductID, StoreID)

Transactions(TransID, Date, StoreID, CustID, StaffID)
TransactionContains(TransID, ProductID, StoreID, PricePaid, Quantity, Status)

Customers(FirstName, LastName, Email, Phone, CustID, Address, MemID)
Memberships(MemID, SignUpDt, Status, Level, Length, StoreID, StaffID)

Registration View

Customers(FirstName, LastName, Email, Phone, CustID, Address, MemID)
Memberships(MemID, SignUpDt, Status, Level, Length, StoreID, StaffID)
Stores(StoreID, Address, Phone, StaffID)

Billing View

Staff(StaffID, Address, Tenure, Email, Name, Age, Title, Phone, StoreID)
Billing(StaffID)

Customers(FirstName, LastName, Email, Phone, CustID, Address, MemID)
Memberships(MemID, SignUpDt, Status, Level, Length, StoreID, StaffID)

Transactions(TransID, Date, StoreID, CustID, StaffID)
TransactionContains(TransID, ProductID, StoreID, PricePaid, Quantity, Status)

Stores(StoreID, Address, Phone, StaffID)

Products(Name, Quantity, ProductID, Price, Production, ExpireDt, StoreID, SupplierID)
Discounts(Percent, StartDt, EndDt, ProductID, StoreID)

Shipments(ShipID, ShipDt, SupplierID, StaffID)
Supplier(SupplierID, Name, Location, Phone, Email)
ShipmentStocks(Cost, Quantity, ShipID, ProductID, StoreID)

Warehouse View

Shipments(ShipID, ShipDt, SupplierID, StaffID)
Supplier(SupplierID, Name, Location, Phone, Email)
ShipmentStocks(Cost, Quantity, ShipID, ProductID, StoreID)

Stores(StoreID, Address, Phone, StaffID)
Products(Name, Quantity, ProductID, Price, Production, ExpireDt, StoreID, SupplierID)

10. Local Schema Documentation

Entity Sets

- Relations were created for the entity sets in our diagram with the attributes for Staff, Stores, Products, Discounts, Customers, Memberships, Shipments and Supplier.
- Relations were made for the Staff entity and its subsets on the diagrams to avoid redundancy.

Many-One Relationships

- In the following entities, an attribute was combined by using a many-to-one relationship. This reduces redundancy, and it makes queries run faster.
    - A relationship was made between the Transactions entity set and the Stores entity set. The Transactions entity set receives the StoreID attribute from the Stores entity. The inclusion of the StoreID attribute represents the “Occurs At” relationship between the two entities. Many transactions can occur at one store.
    - A relationship was made between the Memberships entity set and the Stores entity set. The Memberships entity set receives the StoreID attribute from the Stores entity. The inclusion of StoreID represents the “RegisteredAt” relationship between the two entities. Many memberships can be registered at one store and each membership can only exist in one store.
    - A relationship was made between the Supplier entity set and the Products entity set. The Product entity receives the SupplierID attribute. The inclusion of SupplierID represents the “Supplies” relationship between the two entities. One supplier can supply many products.

One-to-One Relationships

- In the following entities, an attribute was combined by using a one-to-one relationship.
    - A relationship was made between the entity set Cashier and the Customer entity set. The inclusion of the CustomerID and StaffID represents the “ChecksOut” relationship between the two entities. A cashier can only check out one customer at a time.
    - A relationship was made between the entity set Customers and the entity set Memberships. The inclusion of the MemID represents the “Has” relationship between the two entities. Only one customer can have only one membership.

Weak Entities

- The following entities are considered weak and derive their full key through a supporting relationship to another entity, which is represented as multiple keys in the relation.
    - Discounts are keyed by not only their own StartDt, but also by the ProductID and StoreID to which they apply to.
    - Products are keyed not only by their own ProductID, but also by the StoreID to which they belong.
- In the following entities, an attribute was combined by using a many-to-one relationship.

Action Relationships

- The actions from the list below defines how the entities and their attributes are connected in our schema. Their attributes are used as keys for the entities.
    - WorksAt()
        - This connects attributes StaffID and StoreID to Staff and Stores.

- ChecksOut()
  - This connects attributes CashierID, CustID, MembershipID and TransID to Cashier, Customers, Memberships and Transactions.
- Manages()
  - This connects RegistrationStaffID and MemID to Memberships and Registration Staff.
- Ships()
  - This connects ShipID and SupplierID to Shipments and Supplier.
- Supplies()
  - This connects SupplierID and ProductID to Supplier and Products.
- Sells()
  - This connects StoreID and ProductID to Stores and Products.
- Orders()
  - This connects StaffID and ShipID to Billing and Shipments.
- Contains()
  - This connects TransID, ProductID and its own attributes (PricePaid, Quantity, Status) and to Transaction and Products
- Stocks()
  - This connects Cost, Quantity, ShipID and ProductID to Warehouse, Shipments and Products.
- RegisteredAt()
  - This connects MemID and StoreID to Memberships and Stores.
- InCharge()
  - This connects StaffID and StoreID to Manager and Store.

11. Assumptions

- Multiple managers can work in a single store (area managers, etc.), and the store a staff member works at is tracked in the Staff relation. StaffID is also attached to Stores, to make sure at least one Manager is associated with a store.
- Memberships are either active or inactive, and we do not distinguish between expired and cancelled memberships. Active means the membership will automatically renew. Inactive means the membership is expired, or renewal is cancelled but there is still time left on the current membership. This can be determined by registration date and length.
- Only the most recent membership for a customer is tracked in the memberships entity, as there is no need to track past memberships.
- Each membership level can have its own reward rate, but currently only the Platinum membership offers a rewards rate above 0%.
- Membership levels and rewards rates are defined across the chain as a whole and not per store.
- A cashier who transacted a purchase at one store can transfer to another store in the future, so the store where a transaction occurred has to be tracked independently from the cashier.
- When a purchase is refunded (handled at the application level), the full quantity of the product purchased must be refunded.
- Rewards history is maintained for all customers who have earned rewards in a particular year.
- Shipments can come from different suppliers to different stores supplying the same product.
- Cost is not tracked in ShipmentStocks. Instead, the cost of a product from a vendor is held in the Products relation. The cost of a shipment can be derived from ShipmentStocks.quantity * Product.Cost

12. Global Relational Database Schema

Registration(<u>StaffID</u>)
StaffID → StaffID satisfies 3NF requirements as it is a trivial functional dependency with no other attributes existing in the relation.

Cashier(<u>StaffID</u>)
StaffID → StaffID satisfies 3NF requirements as it is a trivial functional dependency with no other attributes existing in the relation.

Billing(<u>StaffID</u>)
StaffID → StaffID satisfies 3NF requirements as it is a trivial functional dependency with no other attributes existing in the relation.

Warehouse(<u>StaffID</u>)
StaffID → StaffID satisfies 3NF requirements as it is a trivial functional dependency with no other attributes existing in the relation.

Manager(<u>StaffID</u>)
StaffID → StaffID satisfies 3NF requirements as it is a trivial functional dependency with no other attributes existing in the relation.

Staff(<u>StaffID</u>, Address, Tenure, Email, Name, Age, Title, Phone, StoreID)
StaffID → StaffID, Name, Age, Address, Tenure, Email, Phone, Title, StoreID holds because the StaffID attribute uniquely identifies staff members. Two employees can theoretically share an email address, phone number, physical address, or name, and as such a combination of these can't be used to derive other attributes. As StaffID is a superkey for the only valid functional dependency, this relation is in 3NF.

Customers(<u>CustID</u>, FirstName, LastName, Email, Phone, Address, MemID)
CustID → CustID, FirstName, LastName, Email, Phone, Address, MemID holds because each customer with all of their respective attributes has a unique customer ID (CustID). Because a customer also only has one membership at a time, a membership ID (MemID) is also derived from a customer ID.
MemID → MemID, CustID holds because each membership is unique to a customer as a customer can only hold one membership at a time, and a membership can only be held by one customer at a time.
No other combination of attributes can be used to uniquely identify a customer as various individuals can share names and contact information. Thus, given that both CustID and MemID are superkeys, the relation is in 3NF.

Memberships(MemID, SignUpDt, Status, LevelID, Length, StoreID, StaffID)
MemID → MemID, Status, LevelID, Length, StoreID, StaffID, SignUpDt holds because the details about a membership including where it was registered, by whom, etc. are uniquely identified by MemID. Since MemID is a superkey for the relation, the relation is in 3NF.

MembershipLevels(LevelID, LevelName, RewardRate)
LevelID → LevelID, LevelName, RewardRate holds as each level has an associated name and rate, and these levels are distinguished by a unique LevelID. Since LevelID is a superkey for the relation, the relation is in 3NF.

Transactions(TransID, Date, StoreID, CustID, StaffID)
TransID → TransID, Date, StoreID, CustID, StaffID holds as a transaction ID (TransID) uniquely identifies all aspects of a transaction. No other combination of attributes allows for the derivation of the rest since multiple transactions can take place on a particular date, in a particular store, by a particular customer, or transacted by a particular cashier.
StaffID → StoreID cannot hold as a staff member can change the store they work at (via internal transfer, etc.) and so a transaction that happened in the past at a different store would not have the correct corresponding information.
Since the only valid functional dependency on the relation has a superkey on its left hand side, the relation is in 3NF.

TransactionContains(TransID, ProductID, StoreID, PricePaid, Quantity, Status, Price)
TransID, ProductID, StoreID → TransID, ProductID, StoreID, PricePaid, Quantity, Status, Price holds as a transaction can contain multiple products, and the only attributes that can be derived are the quantity, price, price paid, and status of an item in a particular transaction. This means that a specific item in a particular transaction is uniquely identified by which transaction it was, what product it was, and what store it took place at. Thus the combination of attributes TransID, ProductID, and StoreID are a superkey of the relation on the left hand side of the functional dependency and the relation is in 3NF.

Stores(StoreID, Address, Phone, StaffID)
StoreID → StoreID, Address, Phone, StaffID holds because an address can only host one store, and any store only has one phone number to reach it at. This means that an address and a phone number are unique to a store (identified by StoreID). Since StoreID is thus a superkey, the relation is in 3NF.

Products(ProductID, StoreID, Name, Quantity, Price, Production, ExpireDt, SupplierID, Cost)
ProductID, StoreID → ProductID, StoreID, Name, Quantity, Price, Production, ExpireDt, SupplierID, Cost holds as a particular product and its respective attributes only exist within a specific store as each store can order its own shipments, set its own price, set its own cost, have its own supplier for the product, etc. Even though the product may be the same, the details of the products can vary so it is the combination of a product ID and a store that uniquely identifies a product. The left hand side of the above functional dependency thus becomes a superkey, and so the relation is in 3NF.

Discounts(DiscountID, ProductID, StoreID, StartDt, Percent, EndDt)
DiscountID, ProductID, StoreID → DiscountID, ProductID, StoreID, StartDt, Percent, EndDt holds because not only does a discount have to exist on a particular product in a particular store (it is on a per store basis), the aforementioned product can also have multiple discounts so a discount ID for the discount is needed to identify which one is being talked about. Thus, a discount ID, product ID, and store uniquely identifies the discount and is a superkey for the relation (on the left side of the functional dependency) as each product can only have one active discount at a time. The relation is therefore in 3NF.

Shipments(ShipID, ShipDt, SupplierID)
ShipID → ShipID, ShipDt, SupplierID holds as a shipment ID uniquely identifies a particular shipment from a supplier on a specific date. This makes ShipID a superkey for the relation, and the relation is in 3NF.

Supplier(SupplierID, Name, Location, Phone, Email)
SupplierID → SupplierID, Name, Location, Phone, Email holds as many suppliers can share a business name, a shipping warehouse, and (oddly enough) contact information - this is not up to our chain to decide. However, a supplier ID will uniquely identify any combination of these attributes, making it a superkey for the relation on the left hand side of the valid functional dependency, and thus the relation is in 3NF.

ShipmentStocks(ShipID, ProductID, StoreID, Quantity)
ShipID, ProductID, StoreID -> ShipID, ProductID, StoreID,Quantity holds as a particular shipment can contain many products going to one or more stores, and the only way to identify what quantity is going to which store or which store was sold the product at what cost is to use the combination of those attributes. Thus ShipID, ProductID, and StoreID are a superkey for the only valid functional dependency that holds on the relation, and the relation is in 3NF.

Rewards(CustID, Year, RewardAmt)
CustID, Year → RewardAmt holds as a customer can receive a reward check across multiple years of membership. Multiple customers can receive a reward check for any given year, and that check amount can also be the same for multiple customers, so no other functional dependencies are valid. Since the left hand side of the above functional dependency is a superkey, the relation is in 3NF.

13. Design for Global Schema

Design Decisions for Global Schema:

Each entity set is Entity steps made into a relation. For the most part the translations are intuitive, but we elaborate on some design decisions as follows. Each ‘isa’ relationship for staff members holds only the staffID.

- There is a chain of weak relationships between store, product, and discounts. This is justified as the same product can be at different stores, and different discounts can be applied at different stores, so each ID is needed for their relative primary keys.
- Each transaction has a transactionContains relationship with products. This is to keep track of different products and relative quantities that are included in each transaction, while following the first normal form.
- We created a new entity for membershipLevels, as to keep track of membership details that apply to all memberships, thus removing redundancy.

All of the many-one relationships were converted into attributes on the many entities, thus reducing redundancy. This applied to most relationships and was effective in simplifying our design.

Other relationships were converted into relations in our schema. These include TransactionContains and ShipmentStocks. These relations hold key attributes from all related entities, and are effective in keeping track of accounting and stocking information. The primary keys are created using inherited attributes.

Integrity Constraints

Registration(<u>StaffID</u>)
StaffID is a primary key and a foreign key to Staff relation

Cashier(<u>StaffID</u>)
StaffID is a primary key and a foreign key to Staff relation

Billing(<u>StaffID</u>)
StaffID is a primary key and a foreign key to Staff relation

Warehouse(<u>StaffID</u>)
StaffID is a primary key and a foreign key to Staff relation

Manager(<u>StaffID</u>, StoreID)
StaffID is a primary key and a foreign key to Staff relation

Staff(<u>StaffID</u>, Address, Tenure, Email, Name, Age, Title, Phone, StoreID)
StaffID is a primary key
StoreID is a foreign key to Stores relation

Address, Tenure, Email, Name, Age, Title, Phone, and StoreID are not allowed to be null as all contact and employee information is required to be maintained by the for legal purposes
Staff is the only relation with Phone required to be not null, as staff need to be reachable at all times.

Customers(<u>CustID</u>, FirstName, LastName, Email, Phone, Address, MemID)
CustID is a primary key
MemID is a foreign key to Memberships relation
FirstName, LastName, Email, Address are not allowed to be null as a customer's contact information is required to register for membership and contact them about rewards.
MemID cannot be null as we do not register Customers to database unless they have a membership.
Phone is allowed to be Null, in case the customer does not want to disclose this information. This field is VARCHAR(15) to allow for international phone numbers, which support up to 15 digits.

Memberships(<u>MemID</u>, SignUpDt, Status, Level, Length, StoreID, StaffID)
MemID is a primary key
StoreID is a foreign key to Stores relation
StaffID is a foreign key to Registration relation
LevelID is a foreign key to MembershipLevels relation
SignUpDt, Status, Level, Length, StoreID, StaffID are not allowed to be null as these attributes are required to track validity of a customer's membership as well as audit who registered them and where they registered

MembershipLevels(<u>LevelID</u>, LevelName, RewardRate)
LevelID is a primary key
LevelName, RewardRate are not allowed to be null as these hold important information regarding membership level details.

Transactions(<u>TransID</u>, Date, StoreID, CustID, StaffID)
TransID is a primary key
StoreID is a foreign key to Stores relation
CustID is a foreign key to Customers relation
StaffID is a foreign key to Cashier relation
Date, StoreID, CustomID, and StaffID are not allowed to be null as the store needs to register who was involved in the transaction and when it took place for accounting purposes

TransactionContains(<u>TransID</u>, <u>ProductID</u>, <u>StoreID</u>, PricePaid, Quantity, Status, Price)
TransID, ProductID, StoreID are the three primary keys.
TransID is a foreign key to Transactions relation
ProductID and StoreID are foreign keys to Products relation
ProductID, StoreID, PricePaid, Quantity, Status, Price are not allowed to be null as this information must be tracked for accounting purposes as well as ensuring the customer is able to refund a transaction or receive rewards based on their purchases.

Stores(<u>StoreID</u>, Address, Phone, StaffID)

StoreID is a primary key
Address, Phone, StaffID are not allowed to be null

Products(Name, Quantity, ProductID, Price, Production, ExpireDt, StoreID, SupplierID, Cost)
ProductID, StoreID are both primary keys.
StoreID is a foreign key to Stores relation
SupplierID is a foreign key to Supplier relation
Name, Quantity, Price, Production, SupplierID are not allowed to be Null.
ExpireDt is allowed to be Null, in case it is a product that does not have an expiration date, i.e. furniture.

Discounts(DiscountID, Percent, StartDt, EndDt, ProductID, StoreID)
DiscountID, Product ID, and Store ID are the three primary keys.
ProductID and StoreID are both foreign keys to Products relationPercent and StartDt are not allowed to be Null. Percent is also described as data type decimal(3,2). This treats 1 as 100% and .5 as 50%. The total 3 decimal places force only whole number percentage discounts (0% to 100%).
EndDt is allowed to be Null, in case the discount is for an indefinite amount of time.

Shipments(ShipID, ShipDt, SupplierID)
ShipID is the primary key.
SupplierID is a foreign key to Supplier relation
ShipDt, SupplierID are not allowed to be null.

Supplier(SupplierID, Name, Location, Phone, Email)
SupplierID is the primary key.
Name, Location, Email are not allowed to be Null. These fields are all VARCHAR(128) and any data validation would be done at application level.
Phone is allowed to be Null, in case the supplier does not want to disclose this information. This field is VARCHAR(15) to allow for international phone numbers, which support up to 15 digits.

ShipmentStocks(Quantity, ShipID, ProductID, StoreID)
ShipID, ProductID, StoreID are the threef three primary keys.
ShipID is a foreign key to Shipments relation
ProductID, StoreID are foreign keys to Products relation
Quantity is not allowed to be Null as it holds important info for billing staff. (Note, ShipmentStocks.quantity * Products.cost gives the total cost of shipment).

Rewards(CustID, Year, RewardAmt)
CustID is a primary key and a foreign key to Customers relation
Year is a primary key
RewardAmt is allowed to be Null, as this attribute is updated in the application layer, and may not be done until the end of the year.

14. Base Relations

Create Statements

```
CREATE TABLE Registration (
        StaffID                 INT,
        PRIMARY KEY(StaffID),
        FOREIGN KEY(StaffID) REFERENCES Staff(StaffID)
        ON UPDATE CASCADE
);

CREATE TABLE Cashier (
        StaffID                 INT,
        PRIMARY KEY(StaffID),
        FOREIGN KEY(StaffID) REFERENCES Staff(StaffID)
        ON UPDATE CASCADE
);

CREATE TABLE Billing (
        StaffID                 INT,
        PRIMARY KEY(StaffID),
        FOREIGN KEY(StaffID) REFERENCES Staff(StaffID)
        ON UPDATE CASCADE
);

CREATE TABLE Warehouse (
        StaffID                 INT,
        PRIMARY KEY(StaffID),
        FOREIGN KEY(StaffID) REFERENCES Staff(StaffID)
        ON UPDATE CASCADE
);

CREATE TABLE Manager (
        StaffID                 INT,
        PRIMARY KEY(StaffID),
        FOREIGN KEY(StaffID) REFERENCES Staff(StaffID)
        ON UPDATE CASCADE
);
```

```
CREATE TABLE Staff (
        StaffID         INT,
        Address         VARCHAR(256) NOT NULL,
        Tenure          DATE NOT NULL,
        Email           VARCHAR(64) NOT NULL,
        Name            VARCHAR(128) NOT NULL,
        Age             INT NOT NULL,
        Title           VARCHAR(64) NOT NULL,
        Phone           VARCHAR(15) NOT NULL,
        StoreID         INT NOT NULL,
        PRIMARY KEY(StaffID),
        FOREIGN KEY(StoreID) REFERENCES Stores(StoreID)
        ON UPDATE CASCADE
);

CREATE TABLE Customers (
        CustID          INT,
        FirstName       VARCHAR(64) NOT NULL,
        LastName        VARCHAR(64) NOT NULL,
        Address         VARCHAR(256) NOT NULL,
        Email           VARCHAR(64) NOT NULL,
        Phone           VARCHAR(15),
        MemID           INT NOT NULL,
        PRIMARY KEY(CustID),
        FOREIGN KEY(MemID) REFERENCES Memberships(MemID)
        ON UPDATE CASCADE
);

CREATE TABLE Memberships (
        MemID           INT,
        StoreID         INT NOT NULL,
        StaffID         INT NOT NULL,
        SignUpDt        DATE NOT NULL,
        Status          BIT(1) NOT NULL,
        LevelID         INT NOT NULL,
        Length          INT NOT NULL,
        PRIMARY KEY(MemID),
        FOREIGN KEY(StaffID) REFERENCES Registration(StaffID)
        ON UPDATE CASCADE,
        FOREIGN KEY(StoreID) REFERENCES Stores(StoreID)
        ON UPDATE CASCADE,
        FOREIGN KEY(LevelID) REFERENCES MembershipLevels(LevelID)
        ON UPDATE CASCADE
);
```

```
CREATE TABLE MembershipLevels (
        LevelID         INT,
        LevelName       VARCHAR(16) NOT NULL,
        RewardRate      DECIMAL(3,2)
        PRIMARY KEY(LevelID)
);

CREATE TABLE Transactions (
        TransID         INT NOT NULL,
        Date            DATE NOT NULL,
        StoreID         INT NOT NULL,
        CustID          INT NOT NULL,
        StaffID         INT NOT NULL,
        PRIMARY KEY(TransID),
        FOREIGN KEY(StaffID) REFERENCES Cashier(StaffID)
        ON UPDATE CASCADE,
        FOREIGN KEY(CustID) REFERENCES Customers(CustID)
        ON UPDATE CASCADE,
        FOREIGN KEY(StoreID) REFERENCES Stores(StoreID)
        ON UPDATE CASCADE
);

CREATE TABLE TransactionContains (
        TransID         INT,
        ProductID       INT NOT NULL,
        StoreID         INT NOT NULL,
        PricePaid       DECIMAL(9,2) NOT NULL,
        Quantity        INT NOT NULL,
        Status          BIT(1) NOT NULL,
        Price           DECIMAL(9,2) NOT NULL,
        PRIMARY KEY(TransID, ProductID, StoreID),
        FOREIGN KEY(TransID) REFERENCES Transactions(TransID)
        ON UPDATE CASCADE,
        FOREIGN KEY(ProductID, StoreID) REFERENCES Products(ProductID, StoreID)
        ON UPDATE CASCADE
);

CREATE TABLE Stores (
        StoreID         INT,
        Address         VARCHAR(256) NOT NULL,
        Phone           VARCHAR(32) NOT NULL,
        StaffID         INT,
        PRIMARY KEY(StoreID)
        FOREIGN KEY(StaffID) REFERENCES Staff(StaffID)
        ON UPDATE CASCADE
```

```
);



CREATE TABLE Products (
        Name        VARCHAR(128) NOT NULL,
        Quantity    INT NOT NULL,
        ProductID   INT,
        Price       DECIMAL(9,2) NOT NULL,
        Production  DATE NOT NULL,
        ExpireDt    DATE,
        StoreID     INT,
        SupplierID  INT NOT NULL,
        Cost        DECIMAL(9,2) NOT NULL,
        PRIMARY KEY (ProductID, StoreID),
        FOREIGN KEY (SupplierID)
        REFERENCES Supplier(SupplierID)
        ON UPDATE CASCADE,
        FOREIGN KEY (StoreID)
        REFERENCES Stores(StoreID)
        ON UPDATE CASCADE
);

CREATE TABLE Discounts (
        Percent     DECIMAL(3,2) NOT NULL,
        StartDt     DATE NOT NULL,
        EndDt       DATE,
        ProductID   INT,
        StoreID     INT,
        DiscountID  INT,
        PRIMARY KEY (DiscountID, ProductID, StoreID),
        FOREIGN KEY (ProductID, StoreID)
        REFERENCES Products (ProductID, StoreID)
        ON UPDATE CASCADE
);

CREATE TABLE Shipments (
        ShipID      INT,
        ShipDt      DATE NOT NULL,
        SupplierID  INT NOT NULL,
        PRIMARY KEY (ShipID),
        FOREIGN KEY (SupplierID)
        REFERENCES Supplier(SupplierID)
        ON UPDATE CASCADE
);
```

```
CREATE TABLE Supplier (
        SupplierID    INT,
        Name          VARCHAR(128) NOT NULL,
        Location      VARCHAR(128) NOT NULL,
        Phone VARCHAR(15),
        Email         VARCHAR(128) NOT NULL,
        PRIMARY KEY (SupplierID)
);

CREATE TABLE ShipmentStocks (
        Quantity      INT NOT NULL,
        ShipID        INT,
        ProductID     INT,
        StoreID       INT,
        PRIMARY KEY (ShipID, ProductID, StoreID),
        FOREIGN KEY (ShipID)
        REFERENCES Shipments(ShipID)
        ON UPDATE CASCADE,
        FOREIGN KEY (ProductID, StoreID)
        REFERENCES Products(ProductID, StoreID)
        ON UPDATE CASCADE
);

CREATE TABLE Rewards (
        CustID        INT,
        Year          INT,
        RewardAmt     DECIMAL(9,2),
        PRIMARY KEY(CustID, Year),
        FOREIGN KEY(CustID)
        REFERENCES Customers(CustID)
        ON UPDATE CASCADE
);
```

Select Statements

Note: Data Subject to change as per testing and demo, but schemas will be the same

SELECT * FROM Registration ;

StaffID
7
17
18
19
20
1003

SELECT * FROM Cashier ;

StaffID
13
14
15
16
1003

SELECT * FROM Billing ;

StaffID
5
6
7
8

SELECT * FROM Warehouse ;

StaffID
9
10
11
12

SELECT * FROM Manager ;

StaffID
1
2
3
4
1001
1002

SELECT * FROM Staff ;

StaffID	Address	Tenure	Email	Name	Age	Title	Phone	StoreID
1	123 8th Ave. Manahawkin, NJ 08050	24	johndoe@gmail.com	John Doe	56	Senior Mgr	3065551952	1
2	436 Brighton Avenue Cleveland, TN 37312	26	mikedoe@gmail.com	Mike Doe	53	Senior Mgr	3065551029	2
3	123 9th Street Oak Forest, IL 60452	28	jakedoe@gmail.com	Jake Doe	51	Senior Mgr	306552052	3
4	1523 Brick St North Bergen, NJ 07047	22	chaddoe@gmail.com	Chad Doe	47	Senior Mgr	306551237	4
5	998 6th Ave. Manahawkin, NJ 08050	18	gracesmith@gmail.com	Grace Smith	54	Analyst	3065558792	1
6	5431 Geek Avenue Cleveland, TN 37312	20	lilysmith@gmail.com	Lily Smith	44	Analyst	3065558427	2
7	22 Fake Street Oak Forest, IL 60452	15	annesmith@gmail.com	Anne Smith	43	Analyst	3065557315	3
8	111 North Ave North Bergen, NJ 07047	13	ritasmith@gmail.com	Rita Smith	39	Analyst	3065550538	4
9	1234 2nd St Manahawkin, NJ 08050	4	johnbyrd@gmail.com	John Byrd	32	Associate	3065551025	1
10	7089 Grey Road Cleveland, TN 37312	12	brianbyrd@gmail.com	Brian Byrd	45	Associate	3065558792	2
11	131 Concrete Street Oak Forest, IL 60452	18	brodybyrd@gmail.com	Brody Byrd	36	Associate	3065555030	3
12	121 Dakota Circle North Bergen, NJ 07047	22	deetzbyrd@gmail.com	Deetz Byrd	48	Associate	3065559653	4
13	1212 32nd Ave Manahawkin, NJ 08050	2	chrisnguyen@gmail.com	Chris Nguyen	19	Associate	3065550000	1
14	12 Clevland Avenue Cleveland, TN 37312	1	willnguyen@gmail.com	Will Nguyen	23	Associate	3065551111	2
15	542 GoodWood Street Oak Forest, IL 60452	3	jessicanguyen@gmail.com	Jessica Nguyen	25	Associate	3065552222	3
16	22 Cali Circle North Bergen, NJ 07047	2	gracenguyen@gmail.com	Grace Nguyen	28	Associate	3065555401	4
17	999 8th Ave. Manahawkin, NJ 08050	1	andreapowers@gmail.com	Andrea Powers	23	Associate	3065552143	1
18	101 Earl Avenue Cleveland, TN 37312	2	jakepowers@gmail.com	Jake Powers	19	Associate	3065551023	2
19	32 Brighton Street Oak Forest, IL 60452	3	marshalpowers@gmail.com	Marshal Powers	23	Associate	3065558327	3
20	59 Oregon Blvd North Bergen, NJ 07047	2	iavorpowers@gmail.com	Iavor Powers	26	Associate	3065555813	4
1001	1101, S Street, NC	2018-10-10	john01@gmail.com	John	32	Senior Mgr	919-1111-123	2001
1002	1102, T Street, NC	2015-07-19	alex12@gmail.com	Alex	42	Senior Mgr	919-1111-456	2002
1003	1103, U Street, NC	2019-07-19	mary34@gmail.com	Mary	28	Associate	919-1111-789	2001

SELECT * FROM Customers ;

CustID	FirstName	LastName	Address	Email	Phone	MemID
1	John	McDonald	7660 SE. Pearl Court Tewksbury, MA 01876	johnmcdonald@gmail.com	8185559432	1
2	Shreeram	Shorey	5432 Fancy St Tewksbury, MA 01876	shreeramshorey@gmail.com	8185550231	2
3	Colleen	Britt	12 Palm St Tewksbury, MA 01876	shreeramshorey@gmail.com	8185554321	3
4	Abhi	Joshi	345 Bay Circle Tewksbury, MA 01876	abhi13joshi@gmail.com	8185553921	4
5001	James	Smith	5500, E Street, NC	James5001@gmail.com	919-5555-123	5
5002	David	Smith	5501 F Street, NC	David5002@gmail.com	919-5555-456	6
5003	tee	grizzly	654 trail road	teegrizzly@aol.com	0900900900	7

SELECT * FROM Memberships ;

MemID	StoreID	StaffID	SignUpDt	MembershipStatus	LevelID	Length
1	1	17	2021-03-15	1	4	365
2	3	18	2021-03-10	1	2	365
3	2	18	2021-03-05	1	3	365
4	4	20	2021-03-01	1	1	365
5	2001	1003	2019-08-01	1	2	365
6	2002	1003	2018-01-01	1	2	365
7	3	7	0000-00-00	1	2	120

SELECT * FROM MembershipLevels ;

LevelID	LevelName	Rate
1	Platinum	0.02
2	Gold	0.00
3	Silver	0.00
4	Bronze	0.00

SELECT * FROM Transactions ;

TransID	Date	StoreID	CustID	StaffID
1	2021-02-20	4	1	13
2	2021-02-20	2	2	15
3	2021-02-20	3	3	16
4	2020-01-21	1	4	14
6001	2020-05-01	2001	5002	1003
6002	2020-06-01	2001	5002	1003
6003	2020-07-01	2001	5001	1003

SELECT * FROM Stores ;

StoreID	Address	Phone	StaffID
1	998 6th Ave. Manahawkin, NJ 08050	7035559428	4
2	7089 Kingston Avenue Cleveland, TN 37312	7035552957	2
3	373 E. Buttonwood Street Oak Forest, IL 60452	7035551048	3
4	569 Virginia Circle North Bergen, NJ 07047	703555739	1
2001	2221, B Street, NC	919-2222-123	1001
2002	2222, C Street, NC	919-2222-456	1002

SELECT * FROM TransactionContains ;

TransID	ProductID	StoreID	PricePaid	Quantity	Status	Price
1	1	1	20.00	4		5.00
2	2	2	16.00	4		4.00
3	3	3	6.00	6		1.00
4	4	4	8.00	2		4.00
6001	3001	2001	80.00	5		20.00
6001	3002	2001	20.00	2		10.00
6002	3002	2001	100.00	10		10.00
6003	3001	2001	160.00	10		20.00

SELECT * FROM Products ;

Name	Quantity	ProductID	Price	Production	ExpireDt	StoreID	SupplierID	Cost
Organic Avacado	3	1	2.00	2021-03-19	2021-04-19	1	4	1
Organic Rice	7	2	10.00	2021-03-19	2022-02-19	2	4	5
Frozen Pizza	12	3	8.00	2021-03-19	2022-01-19	3	1	4
Natural Cheese Puffs	3	4	2.00	2021-03-19	2021-09-19	4	3	1
Overpriced Red Peppers	5	5	6.00	2021-03-19	2021-04-19	4	2	3
Skittles	100	6	4.00	2021-03-21	2022-03-21	1	2	2
Snickers	14	7	2.00	2021-03-21	2022-03-21	3	3	1
Twizzlers	67	8	4.00	2021-03-21	2022-03-21	1	1	2
Pepsi	25	9	2.00	2021-03-21	2022-03-21	1	2	1
Lemons	100	20	2.50	2021-04-15	2021-04-30	1	2	1
Limes	50	21	2.50	2021-04-16	2021-04-25	2	2	1
Oranges	130	22	5.00	2021-04-18	2021-04-25	2	3	3
AAA Paper Towels	100	3001	20.00	2020-01-01	2025-01-01	2001	4001	10
AAA Paper Towels	150	3001	20.00	2020-01-01	2025-01-01	2002	4001	10
BBB Hand soap	200	3002	10.00	2020-01-01	2022-01-01	2001	4002	5
BBB Hand soap	0	3002	10.00	2020-01-01	2022-01-01	2002	4002	5
CCC Red Wine	100	3003	30.00	2021-01-01	2022-01-01	2001	4002	15

SELECT * FROM Discounts ;

Percent	StartDt	EndDt	ProductID	StoreID	DiscountID
0.20	2020-01-01	2021-05-01	3001	2001	7001
0.20	2020-01-01	2021-05-01	3001	2002	7001
0.20	2021-01-01	2021-05-01	3003	2001	7002

SELECT * FROM Shipments ;

ShipID	ShipDt	SupplierID
1	2021-03-01	3
2	2021-03-10	1
3	2021-03-07	4
4	2021-03-16	2

SELECT * FROM Supplier ;

SupplierID	Name	Location	Phone	Email
1	Food Lion	1414 Glenwood Rd, Raleigh, NC, 27604	9195559195	foodlion@ncsu.edu
2	Harris Teeter	504 Cameron Rd, Raleigh, NC, 27604	9195558584	harristeeter@ncsu.edu
3	Trader Joes	432 Cary Dr, Raleigh, NC, 27604	9195554209	traderjoes@ncsu.edu
4	Whole Foods	5420 Wade Blvd, Raleigh, NC, 27604	9195559041	wholefoods@ncsu.edu

SELECT * FROM ShipmentStocks ;

Quantity	ShipID	ProductID	StoreID
6	1	3	3
10	2	2	2
20	3	1	1
34	4	4	4

SELECT * FROM Rewards ;

CustID	Year	RewardAmt
1	2021	120.32
2	2020	98.32
3	2019	52.25
4	2021	120.32

15. SQL Queries

15.1: Interactive SQL Queries

- **Information Processing**

Enter Store Information

```
SQL>INSERT INTO Stores VALUES(5, "123 Main St", "9191234567",
"1002");

Query OK, 1 row affected (0.00 sec)
```

Update Store Information

```
SQL>UPDATE Stores SET Phone="9191232342" WHERE StoreID=5;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Store Information

```
SQL>DELETE FROM Stores WHERE StoreID=5;

Query OK, 1 row affected (0.00 sec)
```

Enter Customer Information

```
SQL>INSERT INTO Memberships VALUES(5, 4, 20, "2020-12-24", 1, 2,
365);

Query OK, 1 row affected (0.01 sec)

SQL>INSERT INTO Customers VALUES(5, "Tiger", "Woods", "2000
University Blvd", "wolfwrcustomer@gmail.com", "9191311323", 5);

Query OK, 1 row affected (0.00 sec)
```

Update Customer Information

```
SQL>UPDATE Memberships SET Status=0 WHERE MemID=5;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

SQL>UPDATE Customers SET Phone="9192223333" WHERE CustID=5;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Customer Information

```
SQL>DELETE FROM Customers WHERE CustID=5;

Query OK, 1 row affected (0.00 sec)
```

```
SQL>DELETE FROM Memberships WHERE MemID=5;

Query OK, 1 row affected (0.00 sec)
```

Enter Staff Information

```
SQL>INSERT INTO Staff VALUES(21, "112 Garden Ave", "1990-02-04",
"adamsandler@gmail.com", "Adam Sandler", 45, "Associate",
"9191233149", 1);

Query OK, 1 row affected (0.01 sec)
```

Update Staff Information

```
SQL>UPDATE Staff SET Phone="9101233149" WHERE StaffID=21;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Staff Information

```
SQL>DELETE FROM Staff WHERE StaffID=21;

Query OK, 1 row affected (0.00 sec)
```

Enter Supplier Information

```
SQL>INSERT INTO Supplier VALUES(5, "Lowes Foods", "Raleigh",
"9195551234", "lowesfoods@gmail.com");

Query OK, 1 row affected (0.00 sec)
```

Update Supplier Information

```
SQL>UPDATE Supplier SET Phone="9105551234" WHERE SupplierID=5;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Supplier Information

```
SQL>DELETE FROM Supplier WHERE SupplierID=5;

Query OK, 1 row affected (0.00 sec)
```

Manage Promotion Information for Products

```
SQL>UPDATE Discounts SET Percent=0.35 WHERE ProductID=2;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Manage Sale Information for Products

```
SQL>UPDATE Products SET Price=3.99 WHERE ProductID=5;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- **Maintaining Inventory Records**

Create inventory for newly arrived products

```
SQL>INSERT INTO Products(Name, Quantity, ProductID, Price,
Production, ExpireDt, StoreID, SupplierID)
VALUES("Pepsi", 25, 9, 1.99, '2021-03-21', '2022-03-21', 1, 2);

Query OK, 1 row affected (0.01 sec)
```

Update inventory with returns

```
SQL>INSERT INTO Products(Name, Quantity, ProductID, Price,
Production, ExpireDt, StoreID, SupplierID, Cost)
VALUES("Snickers", 1, 7, 1.99, '2021-03-21', '2022-03-21', 3, 3, 1)
ON DUPLICATE KEY
UPDATE Quantity=Quantity+1;

Query OK, 1 row affected (0.02 sec)
```

Manage product transfers between stores in the chain

```
SQL>INSERT INTO Products(Name, Quantity, ProductID, Price,
Production, ExpireDt, StoreID, SupplierID, Cost)
VALUES("Twizzlers", 10, 8, 3.49, '2021-03-21', '2022-03-21', 3, 1, 1)
ON DUPLICATE KEY
UPDATE Quantity=Quantity+1, StoreID=3;

Query OK, 1 row affected (0.00 sec)

DELETE FROM Products WHERE ProductID=8 AND StoreID=1;

Query OK, 1 row affected (0.00 sec)
```

- **Maintaining Billing and Transaction Records**

Create or generate bills that are to be paid to a specific supplier

```
SQL>SELECT Cost * Quantity
FROM ShipmentStocks as stocks
JOIN Shipments as shipments
ON shipments.ShipID = stocks.ShipID
JOIN Supplier as supplier
ON supplier.SupplierID  = shipments.SupplierID
WHERE supplier.SupplierID=1;

+-----------------+
| Cost * Quantity |
+-----------------+
|         9203.20 |
+-----------------+
1 row in set (0.01 sec)
```

Generate reward checks for platinum customers that are due at the end of the year

```
SQL>SELECT RewardAmt
FROM Rewards as rewards
JOIN Customers as customers
ON customers.CustID = rewards.CustID
JOIN Memberships as memberships
ON memberships.MemID = customers.MemID
JOIN MembershipLevels as membershipLevel
ON membershipLevel.LevelID = memberships.LevelID
WHERE membershipLevel.LevelName = 'Platinum'
AND rewards.Year = 2021;

+-----------+
| RewardAmt |
+-----------+
|    120.32 |
+-----------+
1 row in set (0.00 sec)
```

For each transaction, calculate the total price, check if any item is on sale, and apply discounts if available

```
SQL>SELECT SUM(PricePaid - (Percent * PricePaid))
FROM Discounts as discounts
JOIN TransactionContains as transactionContains
ON transactionContains.ProductID = discounts.ProductID
JOIN Transactions as transactions
ON transactions.TransID = transactionContains.TransID
WHERE transactions.TransID=1;

+-----------------------------------+
| PricePaid - (Percent * PricePaid) |
+-----------------------------------+
|                           14.2240 |
+-----------------------------------+
1 row in set (0.00 sec)
```

- **Reports**

General reports such as total sales report by day, by month, or by year

```
SQL>SELECT *
FROM TransactionContains as transactionContains
JOIN Transactions as transactions ON transactions.TransID =
transactionContains.TransID
WHERE YEAR(Date)=2020 AND MONTH(Date)=01 AND DAY(Date)=21;
```

```
+---------+-----------+---------+-----------+----------+--------+-------+---------+------------+---------+--------+---------+
| TransID | ProductID | StoreID | PricePaid | Quantity | Status | Price | TransID | Date       | StoreID | CustID | StaffID |
+---------+-----------+---------+-----------+----------+--------+-------+---------+------------+---------+--------+---------+
|       4 |         4 |       4 |      8.00 |        2 |        |  4.00 |       4 | 2020-01-21 |       1 |      4 |      14 |
+---------+-----------+---------+-----------+----------+--------+-------+---------+------------+---------+--------+---------+
1 row in set (0.00 sec)
```

```
SQL>SELECT SUM(PricePaid)
FROM TransactionContains as transactionContains
INNER JOIN Transactions as transactions ON transactions.TransID =
transactionContains.TransID
WHERE YEAR(Date)=2020 AND MONTH(Date)=01 AND DAY(Date)=21;

+-----------------+
| SUM(PricePaid) |
+-----------------+
|          8.43   |
+-----------------+
1 row in set (0.00 sec)
```

```
SQL>SELECT SUM(PricePaid)
FROM TransactionContains as transactionContains
INNER JOIN Transactions as transactions ON transactions.TransID =
transactionContains.TransID
WHERE YEAR(Date)=2020 AND MONTH(Date)=01;

+-----------------+
| SUM(PricePaid) |
+-----------------+
|          8.43   |
+-----------------+
1 row in set (0.00 sec)
```

```
SQL>SELECT SUM(PricePaid)
FROM TransactionContains as transactionContains
INNER JOIN Transactions as transactions ON transactions.TransID =
transactionContains.TransID
WHERE YEAR(Date)=2020;

+-----------------+
| SUM(PricePaid) |
+-----------------+
|          8.43   |
+-----------------+
1 row in set (0.00 sec)
```

Sales growth report for a specific store for a given time period

```
SQL>SELECT SUM(PricePaid)
FROM TransactionContains as transactionContains
JOIN Transactions as transactions ON transactions.TransID =
transactionContains.TransID
WHERE transactionContains.StoreID=1 AND transactionContains.Status=1
AND Date BETWEEN '2021-02-20' AND '2021-02-20';

+----------------+
| SUM(PricePaid) |
+----------------+
|         20.32  |
+----------------+
1 row in set (0.00 sec)
```

Merchandise stock report for each store or for a certain product

```
select ShipmentStocks.*, ShipmentStocks.Quantity * Products.Cost as
ShipmentCost
from ShipmentStocks
inner join Products
     on ShipmentStocks.ProductID = Products.ProductID and
     ShipmentStocks.StoreID = Products.StoreID
where ShipmentStocks.StoreID = 4;

+----------+--------+-----------+---------+--------------+
| Quantity | ShipID | ProductID | StoreID | ShipmentCost |
+----------+--------+-----------+---------+--------------+
|       34 |      4 |         4 |       4 |           34 |
+----------+--------+-----------+---------+--------------+
1 row in set (0.01 sec)


select ShipmentStocks.*, ShipmentStocks.Quantity * Products.Cost as
ShipmentCost
from ShipmentStocks
inner join Products
     on ShipmentStocks.ProductID = Products.ProductID and
     ShipmentStocks.StoreID = Products.StoreID
where ShipmentStocks.ProductID = 3;

+----------+--------+-----------+---------+--------------+
| Quantity | ShipID | ProductID | StoreID | ShipmentCost |
+----------+--------+-----------+---------+--------------+
|        6 |      1 |         3 |       3 |           24 |
+----------+--------+-----------+---------+--------------+
1 row in set (0.00 sec)
```

Customer growth report by month or by year

```
SQL>SELECT COUNT(CustID) FROM Transactions WHERE YEAR(Date)=2021 AND
MONTH(Date)=02;

+---------------+
| COUNT(CustID) |
+---------------+
|             3 |
+---------------+
1 row in set (0.00 sec)

SQL>SELECT COUNT(CustID) FROM Transactions WHERE YEAR(Date)=2020;

+---------------+
| COUNT(CustID) |
+---------------+
|             4 |
+---------------+
1 row in set (0.00 sec)
```

Customer activity report such as total purchase amount for a given time period

```
SQL>SELECT SUM(PricePaid)
FROM TransactionContains as transactionContains
JOIN Transactions as transactions
ON transactions.TransID = transactionContains.TransID
JOIN Customers as customers
ON customers.CustID = transactions.CustID
WHERE customers.CustID=2 AND Date BETWEEN '2021-02-20' AND
'2021-03-20';

+----------------+
| SUM(PricePaid) |
+----------------+
|         16.43  |
+----------------+
1 row in set (0.00 sec)
```

15.2 Explain Directive

Merchandise stock report for a certain product

1. EXPLAIN SELECT * FROM ShipmentStocks WHERE ProductID=1;
2.

```
+------+-------------+----------------+------+---------------+-----------+---------+-------+------+-------+
| id   | select_type | table          | type | possible_keys | key       | key_len | ref   | rows | Extra |
+------+-------------+----------------+------+---------------+-----------+---------+-------+------+-------+
|    1 | SIMPLE      | ShipmentStocks | ref  | ProductID     | ProductID | 4       | const |  1   |       |
+------+-------------+----------------+------+---------------+-----------+---------+-------+------+-------+
```

3. CREATE INDEX productIDIndex ON ShipmentStocks(ProductID);
4.

```
+------+-------------+----------------+------+--------------------------+-----------+---------+-------+------+-------+
| id   | select_type | table          | type | possible_keys            | key       | key_len | ref   | rows | Extra |
+------+-------------+----------------+------+--------------------------+-----------+---------+-------+------+-------+
|    1 | SIMPLE      | ShipmentStocks | ref  | ProductID,productIDIndex | ProductID |       4 | const |    1 |       |
+------+-------------+----------------+------+--------------------------+-----------+---------+-------+------+-------+
```

Merchandise stock report for each store

1. EXPLAIN SELECT * FROM ShipmentStocks WHERE StoreID=4;
2.

```
+------+-------------+----------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table          | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+----------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | ShipmentStocks | ALL  | NULL          | NULL | NULL    | NULL |    4 | Using where |
+------+-------------+----------------+------+---------------+------+---------+------+------+-------------+
```

3. CREATE INDEX storeIDIndex ON ShipmentStocks(StoreID);
4.

```
+------+-------------+----------------+------+---------------+--------------+---------+-------+------+-------+
| id   | select_type | table          | type | possible_keys | key          | key_len | ref   | rows | Extra |
+------+-------------+----------------+------+---------------+--------------+---------+-------+------+-------+
|    1 | SIMPLE      | ShipmentStocks | ref  | storeIDIndex  | storeIDIndex | 4       | const |  1   |       |
+------+-------------+----------------+------+---------------+--------------+---------+-------+------+-------+
```

15.3: Query Correctness

1. **Sales growth report for a specific store for a given time period**

```
SQL>SELECT SUM(PricePaid)
FROM TransactionContains as transactionContains
JOIN Transactions as transactions ON transactions.TransID =
transactionContains.TransID
WHERE transactionContains.StoreID=1 AND transactionContains.Status=1
AND Date BETWEEN '2021-02-20' AND '2021-02-20';
```

$$\Pi_{\text{Sum(PricePaid)}}\left(\sigma_{\text{TransactionContains.StoreID =1 and TransactionContains.Status=1 AND Date BETWEEN '2021-02-20' AND '2021-02-20'}}(\text{TransactionContains} \bowtie \text{Transaction})\right)$$

Suppose tc is any tuple in the TransactionContains relation, and t is any tuple in the Transactions relation, such that the value of tc.TransID and t.TransID are the same. The combination of the tuples tc and t will provide information about which of the transactions belong to a specific store at a certain status and which of those transactions occurred within a specified time period. Because of this combination, the sum of the transaction costs that occurred at a store within a time frame will be returned. This is exactly what we want from this query.

2. **Generate reward checks for platinum customers that are due at the end of the year**

```
SQL>SELECT RewardAmt
FROM Rewards as rewards
JOIN Customers as customers
ON customers.CustID = rewards.CustID
JOIN Memberships as memberships
ON memberships.MemID = customers.MemID
JOIN MembershipLevels as membershipLevel
ON membershipLevel.LevelID = memberships.LevelID
WHERE membershipLevel.LevelName = 'Platinum'
AND rewards.Year = 2021;
```

$$\Pi_{\text{RewardAmt}}\left(\sigma_{\text{membershipLevel.LevelName = 'Platinum' AND rewards.Year = 2021}}(\text{Rewards} \bowtie (\text{Customers} \bowtie (\text{MembershipLevel} \bowtie \text{Memberships}))))\right)$$

Suppose r is any tuple in the Rewards relation, c is any tuple in the Customers relation, m is any tuple in the Memberships relation, and L is any tuple in the MembershipLevel relation. This is such that the value of c.CustID is equal to r.CustID, m.MemID is equal to c.MemID, and L.LevelID is equal to m.LevelID. The combination of tuples c and r will determine if the customer will earn rewards. The combinations of tuples m and c will determine if the customer has a membership. The tuples L and m will determine what level of membership the customer has. If the customer's membership is platinum and due at the end of the current year which is 2021 then it will return the reward amount. This is exactly what we want from this query.