WolfWR Database Management System

CSC 540 Project Report 2

Matt Farver, Nick Garner, Jonathan Nguyen, Wyatt Plaga

Project Assumptions and Modifications

1. Products have a single, unique supplier - e.g. all bananas come from the same supplier.
2. Products can only have one Discount active at a time.
3. All *quantity* of a given Merchandise from a given Transaction must be returned together. Returns of one out of *x* of the same Merchandise are not allowed.
4. *transactionID*, *staffID*, and *memberID* are unique across all stores
5. A single Warehouse processes all Supplier shipments for the entire chain, and is represented as a subclass of Store (storeID=1).
6. *rewardAmount* and *amountOwed* are additional attributes attached to Member and Supplier entities, respectively, to keep track of the money to be paid to each as shipments and transactions occur.
7. Members can decline to give *phone* or *email* when signing up at a store, hence why these fields are allowed to be NULL.
8. Every Staff member must have a *phone* and *email*, hence why these fields are NOT NULL.
9. *products* has been changed to *productID* and added to the primary key for Transaction, such that each Transaction will be split across multiple tuples with one product each.
10. *storeID* has been added to the primary key for Merchandise, such that each Store's inventory of an item is stored on a separate tuple. This allows us to track shipments from Warehouse to Store and from Store to Store.
11. Not all products are perishable, thus *expiration* is allowed to be NULL in Merchandise.

1. Global Database Schema

Staff(staffID, name, age, address, title, phone, email, employmentTime)

- Functionally Dependencies
    - staffID → name, age, address, title, phone, email, employmentTime
        - In 3NF since the left side of the relation (staffID) contains a key.

Member(memberID,level, email, firstName, lastName, phone, address, rewardAmount, activeStatus)

Normalized forms:
Member(memberID, activeStatus, email, firstName, lastName, phone, address, rewardAmount)
Reward(rewardAmount, level)

- Functional Dependencies
    - memberID → activeStatus, level, email, firstName, lastName, phone, address, rewardAmount
        - In 3NF since the left side of the relation (memberID) contains a key.
    - rewardAmount → level
        - In 3NF since the left side of the relation (rewardAmount) now contains a key.

Store(storeID, managerID, phone, address)

- Functional Dependencies
    - storeID → managerID, phone, address
        - In 3NF since the left side of the relation (storeID) contains a key.

Supplier(supplierID, name, phone, location, amountOwed, email)

- Functional Dependencies
    - supplierID → name, phone, email, location, amountOwed
        - In 3NF since the left side of the relation (supplierID) contains a key.

Merchandise(productID, supplierID, storeID, name, quantity, buyPrice, marketPrice, expiration, productionDate)

- Functionally Dependencies
    - productID, supplierID, storeID → name, quantity, buyPrice, marketPrice, expiration, productionDate
        - In 3NF since the left side of the relation (productID) contains a key.

Discount(productID, startDate, endDate, priceReduction)

- Functional Dependencies
    - productID, startDate, endDate → priceReduction
        - In 3NF since the left side of the relation (productID, startDate, endData) contains a key.

Transaction(transactionID, productID, total, date, cashierID, memberID, storeID, quantity)

- Functional Dependencies
    - transactionID, productID → total, date, cashierID, memberID, storeID, quantity
        - In 3NF since the left side of the relation (transactionID) contains a key.

Sign-up(memberID, storeID, staffID, signUpDate)

- Functional Dependencies
    - memberID, storeID, staffID → signUpDate
        - In 3NF since the left side of the relation (memberID, storeID, staffID) contains a key.

2. Design Decisions For Global Database Schema

Staff(staffID, name, age, address, title, phone, email, employmentTime)

- Keys
    - staffID
        - Each staff member will have a unique staffID identifier.
- NULL
    - none
- NOT NULL
    - staffID, name, age, address, title, phone, email, employmentTime
        - Each staff member will have name, address, title, phone, and email when hired.
        - A staff member should never have an employmentTime less than 1 so when a staff member is hired it will be set to a default of 1.
- Referential Integrity
    - none

Member(memberID, level, email, firstName, lastName, phone, address, rewardAmount, activeStatus)

- Keys
    - memberID

        - Each member will have a unique memberID identifier.
- NULL
    - phone, email
        - Phone and email could be NULL if the customer declines to provide them. Means that the customer does not have a phone or email associated with their account.
- NOT NULL
    - memberID, activeStatus, level, firstName, lastName, address, rewardAmount
        - Every member will have a firstName, lastName, phone, address, and level when entered into the database.
        - activeStatus will be set to true if active and false if inactive
        - A member cannot have a reward amount less than 0 so this is set to a default value of 0.
- Referential Integrity
    - memberID
        - memberID refers to the memberID of the SignUp Table.

Store(<u>storeID</u>, managerID, phone, address)

- Keys
    - storeID
        - Each store will have a unique storeID identifier.
    - managerID
        - Each manager will have a unique managerID identifier.
- NULL
    - none
- NOT NULL
    - storeID, managerID, phone, address
        - Each store will have a storeID, managerID, phone, and address assigned.
- Referential Integrity
    - managerID
        - managerID refers to the managerID of the Staff Table.

Supplier(<u>supplierID</u>, name, phone, location, amountOwed, email)

- Keys
    - supplierID
        - Each supplier will have a unique supplierID identifier.
- NULL
    - none
- NOT NULL
    - supplierID, name, phone, email, location, amountOwed

            - Every supplier will have a supplierID, name, phone, email, and location when entered into the database.
            - A supplier cannot have an amount owed less than 0 so it is set to a default value of 0 when a supplier is entered into the database for the first time.
- Referential Integrity
    - none

Merchandise(<u>productID</u>, <u>supplierID</u>, <u>storeID</u>, name, quantity, buyPrice, marketPrice, expiration, productionDate)

- Keys
    - productID
        - Each product will have a unique productID identifier.
    - supplierID
        - Each supplier will have a unique supplierID identifier
    - storeID
        - Each store will have a unique storeID identifier
- NULL
    - expiration
        - A product could not have an expiration date.
- NOT NULL
    - productID, supplierID, storeID, name, quantity, buyPrice, marketPrice, productionDate
        - Each merchandise will have productID, supplierID, storeID, name, quantity, buyPrice, marketPrice, and productionDate when entered into the database.
- Referential Integrity
    - supplierID
        - supplierID refers to the supplierID of the Supplier Table.
    - storeID
        - storeID refers to the storeID of the Store Table.

Discount(<u>productID</u>, <u>startDate</u>, <u>endDate</u>, priceReduction)

- Keys
    - productID
        - Each discount will have a unique productID identifier
    - startDate
        - Each discount will have a unique startDate identifier
    - endDate
        - Each discount will have a unique endDate identifier.
- NULL
    - none

- NOT NULL
    - productID, startDate, endDate, priceReduction
        - Each discount will have a productID, start date, end date, and how much the product is reduced by.
- Referential Integrity
    - productID
        - productID refers to the productID of the Merchandise Table

Transaction(<u>transactionID</u>, <u>productID</u>, total, date, cashierID, memberID, storeID, quantity)

- Keys
    - transactionID
        - Each transaction will have a unique transactionID identifier.
    - productID
        - Each product will have a unique productID identifier.
- NULL
    - none
- NOT NULL
    - transactionID, productID, total, date, cashierID, memberID, storeID, quantity
        - Each transaction record will have transactionID, productID, total cost, quantity purchased, and date of purchase.
        - The quantity for a purchase cannot be less than 1 so it will be defaulted to 1 when entered into the system.
        - Along with each transaction record, it will contain the identifier of cashier, member, and store where the transaction took place.
- Referential Integrity
    - cashierID, memberID, storeID, productID
        - cashierID refers to the staffID in the Staff Table.
        - memberID refers to the memberID in the Member Table.
        - storeID refers to the storeID in the Store Table.
        - productID refers to the productID in the Merchandise Table.

Sign-up(<u>memberID</u>, <u>storeID</u>, <u>staffID</u>, signUpDate)

- Keys
    - memberID, storeID, staffID
        - Each member will have a unique memberID identifier.
        - Each store will have a unique storeID identifier.
        - Each staff member will have a unique staffID identifier.
- NULL
    - none
- NOT NULL

    - memberID,storeID, staffID, signUpDate
        - Each sign up will contain a memberID, storeID, and signUpDate
        - A signUpDate will always exist when a member signs up to the wholesale chain.
- Referential Integrity
    - storeID, staffID
        - storeID refers to the storeID in the Store Table
        - staffID refers to the staffID in the Staff Table

3. Create Table Statements

CREATE TABLE Staff

```
MariaDB [mfarver]> CREATE TABLE Staff (
      -> staffID INT (9) NOT NULL,
      -> name VARCHAR(128) NOT NULL,
      -> age INT NOT NULL,
      -> address VARCHAR(255) NOT NULL,
      -> title VARCHAR(128) NOT NULL,
      -> phone VARCHAR(16) NOT NULL,
      -> email VARCHAR(128) NOT NULL,
      -> employmentTime INT(2) DEFAULT 1 NOT NULL,
      -> UNIQUE(email),
      -> PRIMARY KEY(staffID),
      -> CHECK (age<=100));
Query OK, 0 rows affected (0.01 sec)
```

SELECT * FROM Staff Query Results:

```
+---------+-----------------+-----+-----------------------------------------------------+--------------------+-----------+-----------------------+----------------+
| staffID | name            | age | address                                             | title              | phone     | email                 | employmentTime |
+---------+-----------------+-----+-----------------------------------------------------+--------------------+-----------+-----------------------+----------------+
|       1 | John Smith      |  34 | 13 West 75th street apt 51, New York, New York 10024 | Manager            | 736492735 | jsmith@club.com       |             20 |
|       2 | Eddie Johnson   |  22 | 4 40th street, New York, New York 10004             | Assistant Manager  | 182635487 | ejohnson@club.com     |             15 |
|       3 | Cathy Marks     |  43 | 734 West 108th street apt 8, New York, New York 10001 | Warehouse Checker  | 847638465 | cmarks@club.com       |             10 |
|       4 | June Macky      |  53 | 16 East 70th street, New York, New York 10834       | Billing Staff      | 893764936 | jmacky@club.com       |              5 |
|       5 | Carlia Williams |  43 | 90 West 23th street apt 5, New York, New York 10001 | Cashier            | 837047626 | cwilliams@club.com    |              2 |
|       6 | Jennifer Biden  |  22 | 18 East 44th street, New York, New York 10987       | Cashier            | 873645255 | jbiden@club.com       |              2 |
|       7 | Nick Shoeman    |  21 | 19 East 50th street, New York, New York 109873      | Cashier            | 987374653 | nshoeman@club.com     |              2 |
|       8 | Conner Vestit   |  65 | 20 West 75th street apt 5, New York, New York 10024 | Cashier            | 625384616 | cvestit@club.com      |              1 |
+---------+-----------------+-----+-----------------------------------------------------+--------------------+-----------+-----------------------+----------------+
8 rows in set (0.00 sec)
```

CREATE TABLE Member

```
MariaDB [mfarver]> CREATE TABLE Member (
      -> memberID INT (9) NOT NULL,
      -> level VARCHAR(16) NOT NULL,
      -> email VARCHAR(128),
      -> firstName VARCHAR(64) NOT NULL,
      -> lastName VARCHAR(64) NOT NULL,
```

```
      -> phone VARCHAR(16),
      -> address VARCHAR(255) NOT NULL,
      -> rewardAmount DOUBLE(9, 2) DEFAULT 0 NOT NULL,
      -> activeStatus BOOL NOT NULL,
      -> UNIQUE(email),
      -> FOREIGN KEY(memberID) REFERENCES SignUp(memberID),
      -> PRIMARY KEY(memberID));
Query OK, 0 rows affected (0.01 sec)
```

SELECT * FROM Member Query Results:

```
MariaDB [mfarver]> SELECT * FROM Member;
+----------+----------+-----------------+-----------+----------+-----------+-----------------------------------+--------------+--------------+
| memberID | level    | email           | firstName | lastName | phone     | address                           | rewardAmount | activeStatus |
+----------+----------+-----------------+-----------+----------+-----------+-----------------------------------+--------------+--------------+
|        1 | gold     | joe@hotmail.com | Joe       | Smith    | 837462983 | 1340 Rakes Road, Balitmore MD, 12321 |         0.00 |            1 |
|        2 | gold     | pat@gmail.com   | Pat       | Car      | 725384652 | 8374 HYW 123, New York, NY 98345  |         0.00 |            0 |
|        3 | platinum | chadw@gmail.com | Chad      | William  | 837087364 | 78 Duke Way, Jersey City NY, 77345 |         8.89 |            1 |
|        4 | platinum | 3948@gmail.com  | Jen       | Studdy   | 273625292 | 257 Sunshine Way, Raleigh NC, 12343 |        12.45 |            1 |
+----------+----------+-----------------+-----------+----------+-----------+-----------------------------------+--------------+--------------+
4 rows in set (0.00 sec)
```

CREATE TABLE Store

```
MariaDB [mfarver]> CREATE TABLE Store (
    -> storeID INT (9) NOT NULL,
    -> managerID INT NOT NULL,
    -> phone VARCHAR(16) NOT NULL,
    -> address VARCHAR(128) NOT NULL,
    -> FOREIGN KEY (managerID) REFERENCES Staff(staffID),
    -> PRIMARY KEY(storeID, managerID));
Query OK, 0 rows affected (0.01 sec)
```

SELECT * FROM Store Query Results:

```
MariaDB [mfarver]> SELECT * FROM Store;
+---------+-----------+------------+-----------------------------------------+
| storeID | managerID | phone      | address                                 |
+---------+-----------+------------+-----------------------------------------+
|       1 |         1 | 8736475593 | 22 Jonny Road, Westminster MD 21157     |
|       2 |         9 | 128376492  | 87 Rough Way, New York, NY 12321        |
|       3 |        10 | 276391234  | 977 West Road, Raleigh, NC 12212        |
|       4 |        11 | 783652876  | 8823 Happy Way, Chapel Hill NC, 12111   |
+---------+-----------+------------+-----------------------------------------+
4 rows in set (0.00 sec)
```

CREATE TABLE Supplier

```
MariaDB [mfarver]> CREATE TABLE Supplier (
    -> supplierID INT (9) NOT NULL,
    -> name VARCHAR(128) NOT NULL,
```

```
    -> phone VARCHAR(16) NOT NULL,
    -> location VARCHAR(128) NOT NULL,
    -> amountOwed DOUBLE(12,2) DEFAULT 0 NOT NULL,
    -> email VARCHAR(128) NOT NULL,
    -> PRIMARY KEY(supplierID));
Query OK, 0 rows affected (0.01 sec)
```

SELECT * FROM Supplier Query Results:

```
MariaDB [mfarver]> select * from Supplier;
+------------+-------------------+-----------+---------------+------------+----------------------+
| supplierID | name              | phone     | location      | amountOwed | email                |
+------------+-------------------+-----------+---------------+------------+----------------------+
|          1 | Wigets-R-Us       | 837402749 | Baltimore, MD |       0.00 | widgets@gmail.com    |
|          2 | Bulk Food Inc.    | 928374629 | New York, NY  |   59908.87 | bulk-food@gmail.com  |
|          3 | Lots of Junk LLC. | 827364927 | Newark, NY    |   10984.00 | junk-guys@yahoo.com  |
|          4 | Tech-Gagets Inc   | 928374921 | Raleigh, NC   |  873679.88 | tech.gadgets@aol.com |
+------------+-------------------+-----------+---------------+------------+----------------------+
4 rows in set (0.00 sec)
```

CREATE TABLE Merchandise

```
MariaDB [mfarver]> CREATE TABLE Merchandise (
    -> productID INT (9) NOT NULL,
    -> storeID INT (9) NOT NULL,
    -> name VARCHAR(128) NOT NULL,
    -> quantity INT(9) NOT NULL,
    -> buyPrice DOUBLE(9,2) NOT NULL,
    -> marketPrice DOUBLE(9,2) NOT NULL,
    -> supplierID INT NOT NULL,
    -> expiration DATE,
    -> productionDate DATE,
    -> PRIMARY KEY(productID, supplierID, storeID),
    -> FOREIGN KEY(supplierID) REFERENCES Supplier(supplierID),
    -> FOREIGN KEY(storeID) REFERENCES Store(storeID),
    -> CHECK (quantity>=0),
    -> CHECK (buyPrice>=0.00),
    -> CHECK (marketPrice>=0.00));
Query OK, 0 rows affected (0.01 sec)
```

SELECT * FROM Merchandise Query Results:

```
MariaDB [mfarver]> SELECT * FROM Merchandise;
+-----------+---------+------------------+----------+----------+-------------+------------+------------+----------------+
| productID | storeID | name             | quantity | buyPrice | marketPrice | supplierID | expiration | productionDate |
+-----------+---------+------------------+----------+----------+-------------+------------+------------+----------------+
|         1 |       1 | All Beef Patties |       50 |     6.50 |       12.99 |          2 | 2021-07-01 | 2021-01-01     |
|         2 |       1 | Plastic Bottle   |     1000 |     1.00 |        2.59 |          3 | NULL       | 2021-01-01     |
|         3 |       1 | Mens Watch       |      203 |    25.00 |       75.99 |          4 | NULL       | 2021-01-01     |
|         3 |       2 | Mens Watch       |      190 |    25.00 |       75.99 |          4 | NULL       | 2021-01-01     |
|         3 |       3 | Mens Watch       |       60 |    25.00 |       75.99 |          4 | NULL       | 2021-01-01     |
|         4 |       1 | Wine Opener      |      100 |     6.00 |       12.99 |          1 | NULL       | 2021-01-01     |
|    199283 |       1 | 24# Letterstock  |    10000 |     0.02 |        0.05 |          3 | NULL       | 2021-01-01     |
+-----------+---------+------------------+----------+----------+-------------+------------+------------+----------------+
7 rows in set (0.00 sec)
```

CREATE TABLE Discount

```
MariaDB [mfarver]> CREATE TABLE Discount(
    -> productID INT (9) NOT NULL,
    -> startDate DATE NOT NULL,
    -> endDate DATE NOT NULL,
    -> priceReduction DOUBLE(9,2) NOT NULL,
    -> FOREIGN KEY(productID) REFERENCES Merchandise(productID),
    -> PRIMARY KEY(productID, startDate, endDate),
    -> CHECK (priceReduction>=0.0),
    -> CHECK (endDate>=startDate));
Query OK, 0 rows affected (0.01 sec)
```

SELECT * FROM Discount Query Results:

```
MariaDB [mfarver]> SELECT * FROM Discount;
+------------+------------+------------+-----------------+
| productID  | startDate  | endDate    | priceReduction  |
+------------+------------+------------+-----------------+
|          1 | 2021-02-10 | 2021-07-01 |            1.00 |
|          2 | 2021-02-10 | 2021-07-01 |            0.25 |
|          3 | 2021-02-10 | 2021-07-01 |           10.00 |
|          4 | 2021-02-10 | 2021-07-01 |            2.00 |
+------------+------------+------------+-----------------+
4 rows in set (0.01 sec)
```

CREATE TABLE Transaction

```
CREATE TABLE Transaction (
    -> transactionID INT (9) NOT NULL,
    -> total DOUBLE(9,2) NOT NULL,
    -> productID INT (9) NOT NULL,
    -> date DATE NOT NULL,
    -> cashierID INT (9) NOT NULL,
    -> memberID INT (9) NOT NULL,
    -> storeID INT (9) NOT NULL,
    -> quantity INT (9) NOT NULL DEFAULT 1,
```

```
    -> FOREIGN KEY(cashierID) REFERENCES Staff(staffID),
    -> FOREIGN KEY(memberID) REFERENCES Member(memberID),
    -> FOREIGN KEY(storeID) REFERENCES Store(storeID),
    -> FOREIGN KEY(productID) REFERENCES Merchandise(productID),
    -> PRIMARY KEY(transactionID, productID));
```

SELECT * FROM Transaction Query Results:

```
MariaDB [mfarver]> SELECT * FROM Transaction;
+---------------+-------+-----------+------------+-----------+----------+---------+----------+
| transactionID | total | productID | date       | cashierID | memberID | storeID | quantity |
+---------------+-------+-----------+------------+-----------+----------+---------+----------+
|             1 | 12.99 |         1 | 2021-03-23 |         2 |        3 |       2 |        1 |
|             2 | 53.89 |         3 | 2021-03-18 |         3 |        1 |       2 |        1 |
|             3 | 24.98 |         4 | 2021-01-17 |         3 |        2 |       3 |        1 |
|             4 | 43.99 |         3 | 2021-02-20 |         4 |        4 |       3 |        1 |
+---------------+-------+-----------+------------+-----------+----------+---------+----------+
4 rows in set (0.00 sec)
```

CREATE TABLE SignUp

```
MariaDB [mfarver]> CREATE TABLE SignUp (
    -> memberID INT (9) NOT NULL,
    -> storeID INT (9)NOT NULL,
    -> signUpDate DATE NOT NULL,
    -> staffID INT (9) NOT NULL,
    -> FOREIGN KEY(storeID) REFERENCES Store(storeID),
    -> FOREIGN KEY(staffID) REFERENCES Staff(staffID),
    -> PRIMARY KEY(memberID, storeID, staffID));
```

SELECT * FROM SignUp Query Results:

```
MariaDB [mfarver]> SELECT * FROM SignUp;
+----------+---------+------------+---------+
| memberID | storeID | signUpDate | staffID |
+----------+---------+------------+---------+
|        1 |       1 | 2021-03-13 |       5 |
|        2 |       1 | 2021-03-14 |       5 |
|        3 |       2 | 2021-03-15 |       6 |
|        4 |       2 | 2021-03-16 |       6 |
+----------+---------+------------+---------+
4 rows in set (0.00 sec)
```

4.1 SQL Queries for Narrative Operations

1. "*Information processing.* Enter/update/delete basic information about stores, customers, staff, and suppliers. Manage promotion or sale information for products."
    a. INSERT INTO Store
       VALUES (27017, 3002, 9845553217, '327 Capital Blvd, Raleigh NC');

b. UPDATE Store
   SET phone=9845553218
   WHERE storeID=27017;
c. DELETE FROM Store
   WHERE storeID=27017;

```
MariaDB [mfarver]> INSERT INTO Store VALUES (27017, 3002, 9845553217, '327 Capital Blvd, Raleigh NC');
Query OK, 1 row affected (0.01 sec)

MariaDB [mfarver]> UPDATE Store SET phone=9845553218 WHERE storeID=27017;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> DELETE FROM Store WHERE storeID=27017;
Query OK, 1 row affected (0.00 sec)
```

d. INSERT INTO SignUp
   VALUES(302893, 3, 2, '2021-03-22');

   INSERT INTO Member
   VALUES ('platinum', 'kevinmalone@dundermifflin.com', 'Kevin', 'Malone', 4105553829, '443 Main St, Scranton PA', 0, 1, 302893);
   Assumptions: This is a new member, therefore we have to create a SignUp tuple before creating a referencing Member tuple.
e. UPDATE Member
   SET level=2
   WHERE memberID=302893;
f. DELETE FROM Member
   WHERE memberID=302893;

```
MariaDB [mfarver]> INSERT INTO SignUp VALUES(302893, 3, 2, '2021-03-22');
Query OK, 1 row affected (0.00 sec)

MariaDB [mfarver]> INSERT INTO Member VALUES('platinum', 'kevinmalone@dundermifflin.com', 'Kevin', 'Malone', 4105553829, '44
3 Main St, Scranton PA', 0, 1, 302893);
Query OK, 1 row affected (0.00 sec)

MariaDB [mfarver]> UPDATE Member SET phone='4105553830' WHERE memberID=302893;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> DELETE FROM Member WHERE memberID=302893;
Query OK, 1 row affected (0.01 sec)
```

g. INSERT INTO Staff
   VALUES (99189, 'Dwight Schrute', 39, '123 Scranton St, Scranton PA', 'Assistant Regional Manager', '3335558888', 'dwightkschrute@dundermifflin.com', 17);
h. UPDATE Staff
   SET title='Assistant to the Regional Manager'
   WHERE staffID=99189;
i. DELETE FROM Staff
   WHERE storeID=99189;

```
MariaDB [mfarver]> INSERT INTO Staff VALUES (99189, 'Dwight Schrute', 39, '123 Scranton St, Scranton P
A', 'Assistant Regional Manager', '3335558888', 'dwightkschrute@dundermifflin.com', 17);
Query OK, 1 row affected (0.00 sec)

MariaDB [mfarver]> UPDATE Staff SET title='Assistant to the Regional Manager' WHERE staffID=99189;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> DELETE FROM Staff WHERE staffID=99189;
Query OK, 1 row affected (0.00 sec)
```

j. INSERT INTO Supplier
   VALUES (18192, 'Dunder Mifflin', 3825553892, '1029 Jones Rd, Scranton PA', 0, 'customersupport@dundermifflin.com');
k. UPDATE Supplier
   SET phone=3825553893
   WHERE supplierID=18192;
l. DELETE FROM Supplier
   WHERE supplierID=18192;

```
MariaDB [mfarver]> INSERT INTO Supplier VALUES (18192, 'Dunder Mifflin', 3825553892, '1029 Jones Rd, S
cranton PA', 0, 'customersupport@dundermifflin.com');
Query OK, 1 row affected (0.00 sec)

MariaDB [mfarver]> UPDATE Supplier SET phone=3825553893 WHERE supplierID=18192;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> DELETE FROM Supplier WHERE supplierID=18192;
Query OK, 1 row affected (0.00 sec)
```

m. INSERT INTO Discount
   VALUES(3, ‘2021-03-22’, ‘2021-04-22’, 20.03);

   UPDATE Merchandise
   SET marketPrice=marketPrice - 20.03
   WHERE productID=3;
   Assumptions: Discount is currently active, so we are updating product price.

```
MariaDB [mfarver]> INSERT INTO Discount VALUES (3, '2021-03-22', '2021-04-22', 20.03);
Query OK, 1 row affected (0.01 sec)

MariaDB [mfarver]> UPDATE Merchandise SET marketPrice=marketPrice - 20.03 WHERE productID=3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

n. UPDATE Merchandise
   SET marketPrice=marketPrice + 20.03
   WHERE productID=3;

   UPDATE Discount

SET priceReduction=22.10
WHERE productID=3
AND startDate='2021-03-22'
AND endDate='2021-04-22';

UPDATE Merchandise
SET marketPrice=marketPrice - 22.10
WHERE productID=3;
Assumptions: Discount is currently active. We are adjusting price to first remove old discount before applying new discount.

```
MariaDB [mfarver]> UPDATE Merchandise SET marketPrice=marketPrice + 20.03 WHERE productID=3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> UPDATE Discount SET priceReduction=22.10 WHERE productID=3 AND startDate='2021-03-2
2' AND endDate='2021-04-22';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> UPDATE Merchandise SET marketPrice=marketPrice - 22.10 WHERE productID=3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

o. DELETE FROM Discount
WHERE productID=3
AND startDate='2021-03-22'
AND endDate='2021-04-22';

UPDATE Merchandise
SET marketPrice=marketPrice + 22.10
WHERE productID=3;
Assumptions: Discount has ended or is removed early, we are adjusting price back to original value.

```
MariaDB [mfarver]> DELETE FROM Discount WHERE productID=3 AND startDate='2021-03-22' AND endDate='2021
-04-22';
Query OK, 1 row affected (0.00 sec)

MariaDB [mfarver]> UPDATE Merchandise SET marketPrice=marketPrice + 22.10 WHERE productID=3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

2. "*Maintaining inventory records.* Create inventory for newly arrived products. Update inventory with returns. Manage product transfers between stores in the chain."
    a. Newly arrived products:
    INSERT INTO Merchandise
    VALUES (199283, 1, '24# Letterstock', 10000, 0.02, 0.05, 3, NULL,

'2021-01-01')
ON DUPLICATE KEY UPDATE quantity=quantity + 10000;

UPDATE Supplier
SET amountOwed=amountOwed + (0.02 * 10000)
WHERE supplierID=3;
Assumptions: Product is arriving at central warehouse, hence storeID=1.

```
MariaDB [mfarver]> INSERT INTO Merchandise VALUES (199283,1,'24# Letterstock', 10000, 0.02, 0.05, 3, Null, '202
1-01-01') ON DUPLICATE KEY UPDATE quantity=quantity+10000;
Query OK, 1 row affected (0.01 sec)

MariaDB [mfarver]> UPDATE Supplier SET amountOwed=amountOwed + (0.02 * 10000) WHERE supplierID=3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

b. Returns:
UPDATE Merchandise
SET quantity=quantity + 1
WHERE productID=3
AND supplierID=4
AND storeID=1;

UPDATE Member
SET rewardAmount=rewardAmount - (
    SELECT (total * 0.02)
    FROM Transactions
    WHERE transactionID=93820
    AND productID=199283)
WHERE memberID=22893;

DELETE FROM Transactions
WHERE transactionID=93820;
Assumptions: Single-item transaction, returned to the Warehouse.

```
MariaDB [mfarver]> UPDATE Merchandise SET quantity=quantity + 1 WHERE productID=3 AND supplierID=4 AND storeID=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> UPDATE Member SET rewardAmount=rewardAmount - (SELECT(total * 0.02) FROM Transaction WHERE transactionID=
4 AND productID=3) WHERE memberID=4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> DELETE FROM Transaction WHERE transactionID=4;
Query OK, 1 row affected (0.00 sec)
```

c. Product transfer between stores:
UPDATE Merchandise
SET quantity=quantity - 10
WHERE productID=199283

AND supplierID=18192
AND storeID=13328;

INSERT INTO Merchandise
VALUES (199283, 14389, '24# Letterstock', 10, 0.02, 0.05, 18192, NULL, 'Scranton PA')
ON DUPLICATE KEY UPDATE quantity=quantity + 10;

```
MariaDB [mfarver]> UPDATE Merchandise SET quantity=quantity - 10 WHERE productID=3 AND supplierID=4 AND storeID=2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mfarver]> INSERT INTO Merchandise VALUES(3, 3, 'Mens Watch', 10, 25.00, 75.99, 4, NULL, NULL) ON DUPLICATE KEY UPDA
TE quantity=quantity + 10;
Query OK, 2 rows affected (0.00 sec)
```

3. "*Maintaining billing and transaction records.* Create or generate bills that are to be paid to a specific supplier. Generate reward checks for platinum customers that are due at the end of the year. For each transaction, calculate the total price, check if any item is on sale or not and, if it is, apply discounts according to the discount information."
    a. Create or generate bills that are to be paid to a specific supplier:
    SELECT amountOwed
    FROM Supplier
    WHERE supplierID=3;
    Assumptions: We created the amountOwed field to keep an updated tally of the money owed to each supplier, specifically so that this report could be generated by a simple SELECT query.

```
MariaDB [mfarver]> SELECT amountOwed FROM Supplier WHERE supplierID=3;
+------------+
| amountOwed |
+------------+
|   10984.00 |
+------------+
1 row in set (0.00 sec)
```

    b. Generate reward checks for platinum customers that are due at the end of the year:
    SELECT rewardAmount
    FROM Member
    WHERE MemberID=3;
    Assumptions: Similar to suppliers, we created the rewardAmount field to keep an updated tally of the reward money owed to each platinum member so that this report could be generated by a simple SELECT query.

```
MariaDB [mfarver]> SELECT rewardAmount FROM Member WHERE MemberID=3;
+--------------+
| rewardAmount |
+--------------+
|         8.89 |
+--------------+
1 row in set (0.00 sec)
```

c. For each transaction, calculate the total price:
SELECT SUM(total) FROM Transaction WHERE transactionID=2;

```
MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE transactionID=2;
+------------+
| SUM(total) |
+------------+
|      53.89 |
+------------+
1 row in set (0.00 sec)
```

d. Check if any item is on sale or not:
SELECT Transaction.transactionID, Transaction.productID,
Discount.priceReduction
FROM Transaction
INNER JOIN Discount
ON Transaction.productID=Discount.productID
WHERE Transaction.transactionID=4
AND Discount.startDate <= Transaction.date
AND Discount.endDate >= Transaction.date;

```
MariaDB [mfarver]> SELECT Transaction.transactionID, Transaction.productID, Discount.priceReduction
    -> FROM Transaction
    -> INNER JOIN Discount
    -> ON Transaction.productID=Discount.productID
    -> WHERE Transaction.transactionID=4
    -> AND Discount.startDate <= Transaction.date
    -> AND Discount.endDate >= Transaction.date;
+--------------+-----------+---------------+
| transactionID | productID | priceReduction |
+--------------+-----------+---------------+
|            4 |         3 |         10.00 |
+--------------+-----------+---------------+
1 row in set (0.00 sec)
```

e. If it is, apply discounts according to the discount information.
UPDATE Transaction
SET total=total - quantity * 10.00
WHERE transactionID=4
AND productID=3;

```
MariaDB [mfarver]> UPDATE Transaction
    -> SET total=total - quantity * 10.00
    -> WHERE transactionID=4
    -> AND productID=3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

4. *Reports: Generate all of the following reports.* General reports such as total sales report by day, by month, or by year. Sales growth report for a specific store for a given time period. Merchandise stock report for each store or for a certain product. Customer growth report by month or by year. Customer activity report such as total purchase amount for a given time period.

    a. General reports such as total sales report by day, by month, or by year:
    SELECT SUM(total)
    FROM Transaction
    WHERE date=(SELECT CURDATE());

    SELECT SUM(total)
    FROM Transaction
    WHERE MONTH(date)=MONTH((SELECT CURDATE()))
    AND YEAR(date)=YEAR((SELECT CURDATE()));

    SELECT SUM(total)
    FROM Transaction
    WHERE YEAR(date)=YEAR((SELECT CURDATE()));

    Assumptions: This assumes we are querying for the current day, month, and year. Otherwise, a date would have to be used in place of 'SELECT CURDATE()'.

```
MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE date=(SELECT CURDATE());
+------------+
| SUM(total) |
+------------+
|      12.99 |
+------------+
1 row in set (0.01 sec)

MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE MONTH(date)=MONTH((SELECT CURDATE())) AND YEAR(date)=YEAR((SELE
CT CURDATE()));
+------------+
| SUM(total) |
+------------+
|      66.88 |
+------------+
1 row in set (0.00 sec)

MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE YEAR(date)=YEAR((SELECT CURDATE()));
+------------+
| SUM(total) |
+------------+
|     145.85 |
+------------+
1 row in set (0.00 sec)
```

b. Sales growth report for a specific store for a given time period:

```
SELECT SUM(total)
FROM Transaction
WHERE date=(SELECT CURDATE())
AND storeID=3;

SELECT SUM(total)
FROM Transaction
WHERE MONTH(date)=MONTH((SELECT CURDATE()))
AND YEAR(date)=YEAR((SELECT CURDATE()))
AND storeID=3;

SELECT SUM(total)
FROM Transaction
WHERE YEAR(date)=YEAR((SELECT CURDATE()))
AND storeID=3;
```

Assumptions: This assumes we are querying for the current day, month, and year. Otherwise, a date would have to be used in place of 'SELECT CURDATE()'.

```
MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE date=(SELECT CURDATE()) AND storeID=3;
+------------+
| SUM(total) |
+------------+
|      24.98 |
+------------+
1 row in set (0.00 sec)

MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE MONTH(date)=MONTH((SELECT CURDATE())) AND YEAR(da
te)=YEAR((SELECT CURDATE())) AND storeID=3;
+------------+
| SUM(total) |
+------------+
|      24.98 |
+------------+
1 row in set (0.00 sec)

MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE YEAR(date)=YEAR((SELECT CURDATE())) AND storeID=3
;
+------------+
| SUM(total) |
+------------+
|      68.97 |
+------------+
1 row in set (0.00 sec)
```

c. Merchandise stock report for each store or for a certain product:

```
SELECT *
FROM Merchandise
WHERE storeID=1;

SELECT *
FROM Merchandise
WHERE productID=3;
```

Assumptions: Query is for current inventory details, as we are not tracking inventory history.

```
MariaDB [mfarver]> SELECT * FROM Merchandise WHERE storeID=1;
+-----------+---------+------------------+----------+----------+-------------+------------+------------+----------------+
| productID | storeID | name             | quantity | buyPrice | marketPrice | supplierID | expiration | productionDate |
+-----------+---------+------------------+----------+----------+-------------+------------+------------+----------------+
|         1 |       1 | All Beef Patties |       50 |     6.50 |       12.99 |          2 | 2021-07-01 | 2021-01-01     |
|         2 |       1 | Plastic Bottle   |     1000 |     1.00 |        2.59 |          3 | NULL       | 2021-01-01     |
|         3 |       1 | Mens Watch       |      203 |    25.00 |       75.99 |          4 | NULL       | 2021-01-01     |
|         4 |       1 | Wine Opener      |      100 |     6.00 |       12.99 |          1 | NULL       | 2021-01-01     |
|    199283 |       1 | 24# Letterstock  |    10000 |     0.02 |        0.05 |          3 | NULL       | 2021-01-01     |
+-----------+---------+------------------+----------+----------+-------------+------------+------------+----------------+
5 rows in set (0.00 sec)

MariaDB [mfarver]> SELECT * FROM Merchandise WHERE productID=3;
+-----------+---------+------------+----------+----------+-------------+------------+------------+----------------+
| productID | storeID | name       | quantity | buyPrice | marketPrice | supplierID | expiration | productionDate |
+-----------+---------+------------+----------+----------+-------------+------------+------------+----------------+
|         3 |       1 | Mens Watch |      203 |    25.00 |       75.99 |          4 | NULL       | 2021-01-01     |
|         3 |       2 | Mens Watch |      190 |    25.00 |       75.99 |          4 | NULL       | 2021-01-01     |
|         3 |       3 | Mens Watch |       60 |    25.00 |       75.99 |          4 | NULL       | 2021-01-01     |
+-----------+---------+------------+----------+----------+-------------+------------+------------+----------------+
3 rows in set (0.01 sec)
```

d. Customer growth report by month or by year:

```
SELECT COUNT(memberID)
FROM SignUp
WHERE MONTH(signUpDate)=MONTH((SELECT CURDATE()))
```

AND YEAR(signUpDate)=YEAR((SELECT CURDATE()));

SELECT COUNT(memberID)
FROM SignUp
WHERE YEAR(signUpDate)=YEAR((SELECT CURDATE()));

Assumptions: This assumes we are querying for the current month and year. Otherwise, a date would have to be used in place of 'SELECT CURDATE()'.

```
MariaDB [mfarver]> SELECT COUNT(memberID) FROM SignUp WHERE MONTH(signUpDate)=MONTH((SELECT CURDATE())) AND YEAR(signUpDate
)=YEAR((SELECT CURDATE()));
+-----------------+
| COUNT(memberID) |
+-----------------+
|               4 |
+-----------------+
1 row in set (0.00 sec)

MariaDB [mfarver]> SELECT COUNT(memberID) FROM SignUp WHERE YEAR(signUpDate)=YEAR((SELECT CURDATE()));
+-----------------+
| COUNT(memberID) |
+-----------------+
|               4 |
+-----------------+
1 row in set (0.00 sec)
```

e. Customer activity report such as total purchase amount for a given time period:
SELECT SUM(total)
FROM Transaction
WHERE memberID=3
AND YEAR(date)=YEAR((SELECT CURDATE));

Assumptions: Query to get the total amount spent by a customer for the current year-to-date.

```
MariaDB [mfarver]> SELECT SUM(total) FROM Transaction WHERE memberID=3 AND YEAR(date)=YEAR((SELECT CURDATE()));
+------------+
| SUM(total) |
+------------+
|      12.99 |
+------------+
1 row in set (0.00 sec)
```

4.2 EXPLAIN Directives

Get Staff Member

1. SELECT * FROM Staff WHERE title='Manager';

```
MariaDB [mfarver]> SELECT * FROM Staff WHERE title='Manager';
+---------+--------------+-----+-----------------------------------------------------+---------+-----------+-------------------+----------------+
| staffID | name         | age | address                                             | title   | phone     | email             | employmentTime |
+---------+--------------+-----+-----------------------------------------------------+---------+-----------+-------------------+----------------+
|       1 | John Smith   |  34 | 13 West 75th street apt 51, New York, New York 10024 | Manager | 736492735 | jsmith@club.com   |             20 |
|       9 | Jim Halpert  |  55 | 23 W. 88th Street New York, NY, 10089               | Manager | 879476394 | jhalpert@club.com |             23 |
|      10 | Michael Scott |  58 | 3 W. 76th Street New York, NY, 10089                | Manager | 879421394 | mscott@club.com   |             30 |
|      11 | Pam Beasly   |  45 | 22 Dunder Street New York, NY, 10065                | Manager | 786421394 | pbeasly@club.com  |             19 |
+---------+--------------+-----+-----------------------------------------------------+---------+-----------+-------------------+----------------+
4 rows in set (0.00 sec)
```

2. EXPLAIN SELECT * FROM Staff WHERE title=’Manager’;

```
MariaDB [mfarver]> EXPLAIN SELECT * FROM Staff WHERE title='Manager';
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | Staff | ALL  | NULL          | NULL | NULL    | NULL |   11 | Using where |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.01 sec)
```

3. CREATE INDEX titleIndex ON Staff(title);

4. EXPLAIN SELECT * FROM Staff FORCE INDEX (titleIndex) WHERE title=’Manager’;

```
MariaDB [mfarver]> EXPLAIN SELECT * FROM Staff FORCE INDEX (titleIndex) WHERE title='Manager';
+------+-------------+-------+------+---------------+------------+---------+-------+------+-----------------------+
| id   | select_type | table | type | possible_keys | key        | key_len | ref   | rows | Extra                 |
+------+-------------+-------+------+---------------+------------+---------+-------+------+-----------------------+
|    1 | SIMPLE      | Staff | ref  | titleIndex    | titleIndex | 130     | const |    4 | Using index condition |
+------+-------------+-------+------+---------------+------------+---------+-------+------+-----------------------+
1 row in set (0.00 sec)
```

Get Member Level

1. SELECT * FROM Member WHERE level=’gold’;

```
MariaDB [mfarver]> SELECT * FROM Member WHERE level='gold';
+-------+-----------------+-----------+----------+----------+-----------------------------------+--------------+--------------+----------+
| level | email           | firstName | lastName | phone    | address                           | rewardAmount | activeStatus | memberID |
+-------+-----------------+-----------+----------+----------+-----------------------------------+--------------+--------------+----------+
| gold  | joe@hotmail.com | Joe       | Smith    | 837462983 | 1340 Rakes Road, Balitmore MD, 12321 |         0.00 |            1 |        1 |
| gold  | pat@gmail.com   | Pat       | Car      | 725384652 | 8374 HYW 123, New York, NY 98345  |         0.00 |            0 |        2 |
+-------+-----------------+-----------+----------+----------+-----------------------------------+--------------+--------------+----------+
2 rows in set (0.00 sec)
```

2. EXPLAIN SELECT * FROM Member WHERE level=’gold’;

```
MariaDB [mfarver]> EXPLAIN SELECT * FROM Member WHERE level='gold';
+------+-------------+--------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table  | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+--------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | Member | ALL  | NULL          | NULL | NULL    | NULL |    4 | Using where |
+------+-------------+--------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

3. CREATE INDEX levelIndex ON Member(level);

4. EXPLAIN SELECT * FROM Member FORCE INDEX (levelIndex) WHERE level=’gold’;

```
MariaDB [mfarver]> EXPLAIN SELECT * FROM Member FORCE INDEX (levelIndex) WHERE level='gold';
+------+-------------+--------+------+---------------+------------+---------+-------+------+-----------------------+
| id   | select_type | table  | type | possible_keys | key        | key_len | ref   | rows | Extra                 |
+------+-------------+--------+------+---------------+------------+---------+-------+------+-----------------------+
|    1 | SIMPLE      | Member | ref  | levelIndex    | levelIndex | 18      | const |    2 | Using index condition |
+------+-------------+--------+------+---------------+------------+---------+-------+------+-----------------------+
1 row in set (0.00 sec)
```

4.3 JOIN Queries Writeup

1. Check if any products in a transaction have a discount:
    a. Query:
    ```
    SELECT Transaction.transactionID, Transaction.productID,
    Discount.priceReduction
    FROM Transaction
    INNER JOIN Discount
    ON Transaction.productID=Discount.productID
    WHERE Transaction.transactionID=4
    AND Discount.startDate <= Transaction.date
    AND Discount.endDate >= Transaction.date;
    ```

    b. Relational Algebra:

$$\Pi_{transactionID,Transaction.productID,priceReduction}(\sigma_{transactionID=4,startDate\leq date,endDate\geq date}(Transaction \bowtie Discount))$$

    c. Correctness Proof:
    Suppose that *x* is a tuple in the Transaction relation and *y* is a tuple in the Discount relation, such that the value of *x.productID* is equal to *y.productID*. Each combination of x and y (as determined by the natural join of Transaction and Discount) gives us products in a Transaction that have a matching Discount in the database. By using the constraints startDate<=date and endDate>=date, we ensure that the Discount is currently active. Finally, by limiting the data to the transactionID in question, we can see items with a discount in the current transaction only.

2. Generate a report on the amount of Transaction revenue generated by Platinum members for the current year:
    a. Query:
    ```
    SELECT SUM(Transaction.total)
    FROM Transaction
    INNER JOIN Member
    ON Transaction.memberID=Member.memberID
    WHERE Transaction.level='platinum'
    AND YEAR(Transaction.date)=YEAR((SELECT CURDATE()));
    ```

    b. Relational Algebra:

$$\gamma_{SUM(Transaction.total)}($$
$$\sigma_{Member.level=\text{platinum},YEAR(Transaction.date)=YEAR(\sigma_{CURDATE})}(Transaction \bowtie Member))$$

c. Correctness Proof:
   Suppose that *x* is a tuple in the Member relation and *y* is a tuple in the Transaction relation such that *x.memberID* is equal to *y.memberID*. Each combination of *x* and *y* (as determined by the natural join of Member and Transaction) gives us all Transaction records for a particular Member. By constraining the output to only tuples where memberID=’platinum’ and taking the SUM of the Transaction.total, we are able to calculate the total revenue spent in all stores by platinum members.