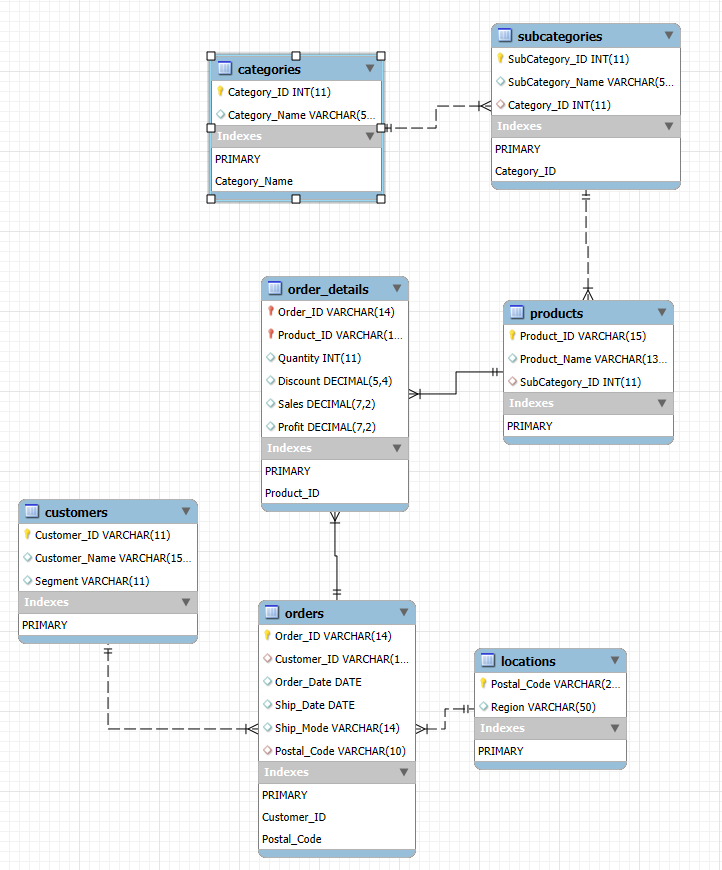
**Milestone 2**

Team: Owen Randolph, Gabe Tharp, Marcos Fernandez

**1. Conceptual Schema & Diagram:**



After normalization we have 7 different tables:

1. Orders: Holds records for unique orders.

Relationships/Cardinalities:

With Customers – Many-To-One (Many orders to a single customer, 1 customer per order)

With Locations – Many-To-One (Many orders to a single location, 1 location per order)

With Order\_Details -One-To-Many (One order to a single order\_details, many order\_details per order)

1. Customers: Holds records for unique customers.

Relationships/Cardinalities:

With Orders – One-To-Many (One customer to a single order, many orders per customer)

1. Locations: Holds records for unique locations.

Relationships/Cardinalities:

With Orders – One-To-Many (One location to a single order, many orders per location)

1. Order\_Details: Holds the records for the details of each order.

Relationships/Cardinalities:

With Orders - Many-To-One (Many order\_details to a single order, 1 order per order\_detail)

With Products – Many-To-One (Many order\_details to a single product, 1 product per order\_detail)

1. Products: Holds records for unique products.

Relationships/Cardinalities:

With Order\_details – One-To-Many (One product to a single order\_detail, many order\_detail per customer)

With Subcategories – Many-To-One (Many products to a single subcategory, 1 subcategory per product)

1. Subcategories: Holds records for unique product subcategories.

Relationships/Cardinalities:

With Products - One-To-Many (One subcategory to a single product, many products per subcategory)

With Categories – Many-To-One (Many subcategories to a single category, 1 category per subcategories)

1. Categories: Holds records for unique product categories.

Relationships/Cardinalities:

With Subcategories – One-To-Many (One category to a single subcategory, many subcategories per category)

Here are the steps taken to reach these tables and the reason behind the steps taken:

1. Attempted to change the date format. The date format in CSV file from MM/DD/YYYY to YYYY-MM-DD in Order Date and Ship Date columns doesn’t work. Windows reverts to the MM/DD/YYYY date format even after saving. This has to be done using Python code in a Jupyter notebook and pushed directly into MySQL Workbench using a connector (see Jupyter notebook)
2. Create Schema for new database “ecommerce” in My SQL Workbench

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create Pre-normalized Table “ecommerce” in MySQL Workbench

A screenshot of a computer

AI-generated content may be incorrect.

1. Data Cleaning using Python Pandas library (see Jupyter notebook code)

* Removal of “$” before the numbers in the Sales, Profit, and Discount values
* Round those values to 2 significant figures to account for cents in the dollar amount
* Change the dates in Order Date and Ship Date columns to YYYY-MM-DD format for MySQL readability

1. Load data into MySQL workbench table and test to make sure it worked smoothly by checking the number of rows and trying a test query.

A screenshot of a computer

AI-generated content may be incorrect.

1. Normalize the Database
   1. 1NF : a table is normalized to the first form if each column has individual values, each row is distinct (no duplicates), and each column has only values of the same type. These were satisfied when we created the database. To check for these we use a Python for-loop that checks all values for commas or colons.

**The data table was already in first normalized form from the beginning, no adjustments necessary.**

* 1. 2NF: a table is normalized to the second form if it satisfies 1NF and has no partial dependencies. This is satisfied when the non-key columns are completely dependent on the entire primary key.

To get this data table into the second normalized form, we need to break three new tables:

* Customers: Customer\_Id (PK), Customer\_Name, Segment
* Orders: Order\_Id (PK), Customer\_Id (FK), Order\_Date, Ship\_Date, Ship\_Mode, Postal\_Code, Region
* Products: Product\_Id (PK), Product\_Name, Category, Sub-Category
* Order Details: Order\_Id (PK, FK), Product\_Id (PK, FK), Quantity, Discount, Sales, Profit

We use CRUD methods to create the new tables in MySQL Workbench and insert the correct data into each new table. See Appendix (CRUD methods)

* 1. 3NF: a table is normalized to the third form if there are no transitive dependencies, meaning there are no non-primary key fields what are reliant on only the primary key. Foreign keys have to also be added for the new tables:
* In Products table, Sub-Category is dependent on Category, so we break it into a new table and add a new ID for it (SubCategory\_ID)
* We create a new table called Locations because Postal code is dependent on region

**2. Data Constraints:**

1. Orders:

- Customer\_ID is a foreign key that references the Customers table and therefore cannot be null.

- Postal\_Code is a foreign key that references the Locations table and therefore cannot be null.

1. Customers:

- Customer\_Name must be Unique and Not Null as this represents real customers which we must be able to identify so it must not be null and must be unique.

1. Locations:

- Region must be Unique and Not Null as every zip code must be associated with a region.

1. Order\_Details:

- Order\_ID & Product\_ID make a combination primary key so each can be duplicated but the combination must be unique.

- Order\_ID is a foreign key that references the Orders table and therefore cannot be null.

- Product\_ID is a foreign key that references the Products table and therefore cannot be null.

-Quantity must not be Null as we need to know how much of the product was sold.

1. Products:

- Product\_Name must be Unique and Not Null as this represents the real product and therefore it must be identified with a name.

- SubCategory\_ID is a foreign key that references the Subcategories table and therefore cannot be null.

1. Subcategories:

- SubCategory\_Name must be Unique and Not Null as this represents the unique subcategory that exists and therefore the tie to the category that the product is sorted into.

- Category\_ID is a foreign key that references the Categories table and therefore cannot be null.

1. Categories:

- Category\_Name must be Unique and Not Null as this represents a unique category that exists and must have a name so it can be attached to products.

**3. Database Creation & Queries:**

Here we provide the queries used (Note that the queries also exist in the uploaded “Milestone 2 Script.sql”

Create your database, tables (or collections), keys/indexes, and initial data:

-- Create Inital Table schema from column headers - Gabe Tharp

CREATE TABLE ecommerce (

Row\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Order\_ID VARCHAR(14),

Order\_Date DATE,

Ship\_Date DATE,

Ship\_Mode VARCHAR(14),

Customer\_ID VARCHAR(11),

Customer\_Name VARCHAR(150),

Segment VARCHAR(11),

Country VARCHAR(13),

City VARCHAR(17),

State VARCHAR(20),

Postal\_Code VARCHAR(10),

Region VARCHAR(7),

Product\_ID VARCHAR(15),

Category VARCHAR(15),

SubCategory VARCHAR(12),

Product\_Name VARCHAR(130),

Sales DECIMAL(7,2),

Quantity INT,

Discount DECIMAL(3,2),

Profit DECIMAL(7,2)

);

-- Note: at this point we uploaded the data with the Python code (images in the appendix)

-- Test that it uploaded - Gabe Tharp

SELECT \* FROM ecommerce LIMIT 5;

-- Create 2NF tables for normalization -- Owen Randolph

CREATE TABLE Customers (

Customer\_ID VARCHAR(11) PRIMARY KEY,

Customer\_Name VARCHAR(150) NOT NULL UNIQUE,

Segment VARCHAR(11)

);

CREATE TABLE Orders (

Order\_ID VARCHAR(14) PRIMARY KEY,

Customer\_ID VARCHAR(11),

Order\_Date DATE,

Ship\_Date DATE,

Ship\_Mode VARCHAR(14),

Postal\_Code VARCHAR(10),

Region VARCHAR(7),

FOREIGN KEY (Customer\_ID) REFERENCES Customers(Customer\_ID)

);

CREATE TABLE Products (

Product\_ID VARCHAR(15) PRIMARY KEY,

Product\_Name VARCHAR(130) NOT NULL UNIQUE,

Category VARCHAR(15),

SubCategory VARCHAR(12)

);

CREATE TABLE Order\_Details (

Order\_ID VARCHAR(14),

Product\_ID VARCHAR(15),

Quantity INT NOT NULL,

Discount DECIMAL(3,2),

Sales DECIMAL(7,2),

Profit DECIMAL(7,2),

PRIMARY KEY (Order\_ID, Product\_ID),

FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID),

FOREIGN KEY (Product\_ID) REFERENCES Products(Product\_ID)

);

-- Add data to the new tables - Owen Randolph

INSERT INTO Customers (Customer\_ID, Customer\_Name, Segment)

SELECT DISTINCT Customer\_ID, Customer\_Name, Segment

FROM ecommerce;

INSERT INTO Orders (Order\_ID, Customer\_ID, Order\_Date, Ship\_Date, Ship\_Mode, Postal\_Code, Region)

SELECT DISTINCT Order\_ID, Customer\_ID, Order\_Date, Ship\_Date, Ship\_Mode, Postal\_Code, Region

FROM ecommerce;

INSERT INTO Products (Product\_ID, Product\_Name, Category, SubCategory)

SELECT DISTINCT Product\_ID, Product\_Name, Category, SubCategory

FROM ecommerce;

INSERT INTO Order\_Details (Order\_ID, Product\_ID, Quantity, Discount, Sales, Profit)

SELECT DISTINCT Order\_ID, Product\_ID, Quantity, Discount, Sales, Profit

FROM ecommerce;

-- Test query to see if DDL has worked

SELECT \* FROM order\_details LIMIT 5;

-- Create more tables for 3NF normalization - Marcos Fernandez

CREATE TABLE Categories (

Category\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Category\_Name VARCHAR(50) NOT NULL UNIQUE

);

CREATE TABLE Subcategories (

SubCategory\_ID INT AUTO\_INCREMENT PRIMARY KEY,

SubCategory\_Name VARCHAR(12) NOT NULL UNIQUE,

Category\_ID INT,

FOREIGN KEY (Category\_ID) REFERENCES Categories(Category\_ID)

);

CREATE TABLE Locations (

Postal\_Code VARCHAR(10) PRIMARY KEY,

Region VARCHAR(7) NOT NULL UNIQUE

);

-- Insert data into new tables - Marcos Fernandez

INSERT INTO Categories (Category\_Name)

SELECT DISTINCT Category FROM ecommerce;

INSERT INTO Subcategories (SubCategory\_Name, Category\_ID)

SELECT DISTINCT

e.SubCategory,

c.Category\_ID

FROM ecommerce e

JOIN Categories c ON e.Category = c.Category\_Name;

INSERT INTO Locations (Postal\_Code, Region)

SELECT DISTINCT Postal\_Code, Region FROM ecommerce;

-- Add foreign keys for new tables - Owen Randolph

ALTER TABLE products

ADD FOREIGN KEY (SubCategory\_ID) REFERENCES subcategories(SubCategory\_ID);

ALTER TABLE orders

ADD FOREIGN KEY (Postal\_Code) REFERENCES locations(Postal\_Code);

ALTER TABLE products

DROP COLUMN Category,

DROP COLUMN SubCategory;

ALTER TABLE orders

DROP COLUMN Region;

--Perform all CRUD operations required by your application scope:

--Note these are generic sample queries that will be used in the application to add/remove and update records.

-- The read queries will change depending on the page so we will add these later.

-- All 3 team members contributed to creating these.

INSERT INTO Customers (Customer\_ID, Customer\_Name, Segment)

VALUES ('TEST', 'TEST', 'TEST')

DELETE FROM Customers

WHERE Customer\_ID = 'TEST'

UPDATE Customers

SET Segment = 'TEST'

WHERE Customer\_ID = 'TEST'

INSERT INTO Locations (Postal\_Code, Region)

VALUES ('TEST', 'TEST')

DELETE FROM Locations

WHERE Postal\_Code = 'TEST'

UPDATE Locations

SET Region = 'TEST'

WHERE Postal\_Code = 'TEST'

INSERT INTO Orders (Order\_ID, Customer\_ID, Order\_Date, Ship\_Date, Ship\_Mode, Postal\_Code)

VALUES ('TEST', 'TEST', 'TEST', 'TEST', 'TEST', 'TEST')

DELETE FROM Orders

WHERE Order\_ID = 'TEST'

UPDATE Orders

SET Ship\_Mode = 'TEST'

WHERE Order\_ID = 'TEST'

INSERT INTO Order\_Details (Order\_ID, Product\_ID, Quantity, Discount, Sales, Profit)

VALUES ('TEST', 'TEST', 'TEST', 'TEST', 'TEST', 'TEST')

DELETE FROM Order\_Details

WHERE Order\_ID = 'TEST' AND Product\_ID = 'TEST'

UPDATE Order\_Details

SET Quantity = 'TEST'

WHERE Order\_ID = 'TEST' AND Product\_ID = 'TEST'

INSERT INTO Products (Product\_ID, Product\_Name, Category, SubCategory)

VALUES ('TEST', 'TEST', 'TEST', 'TEST')

DELETE FROM Products

WHERE Product\_ID = 'TEST'

UPDATE Products

SET Product\_Name = 'TEST'

WHERE Product\_ID = 'TEST'

INSERT INTO Subcategories (SubCategory\_Name, Category\_ID)

VALUES ('TEST', 'TEST', 'TEST')

DELETE FROM Subcategories

WHERE SubCategory\_ID = 'TEST'

UPDATE Subcategories

SET SubCategory\_Name = 'TEST'

WHERE SubCategory\_ID = 'TEST'

INSERT INTO Categories (Category\_Name)

VALUES ('TEST')

DELETE FROM Categories

WHERE Category\_ID = 'TEST'

UPDATE Categories

SET Category\_Name = 'TEST'

WHERE Category\_ID = 'TEST'

**Reference**

Chat GPT used for writing python code to prove 1NF to check for non-atomic values, duplicate rows, and consistent data types

**4. Overall Contribution Summary/Assessment:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Task** | **Contribution Details** | **Hours** |
| Gabriel Tharp | Create MySQL DB | Create db and empty table in MySQL workbench with appropriate data types, constraints - SQL | 0.5 |
| Gabriel Tharp | Clean/Transform | Remove duplicates, handle NULLs, clean data - Python | 1.5 |
| Gabriel Tharp | Load Data | Load CSV into Jupyter Notebook - Python | 0.1 |
| Gabriel Tharp | ER Diagram | Reverse engineer ER diagram - MySQL Workbench | 0.1 |
| Gabriel Tharp | Load Data | Make a connection into MySQL Workbench - Python | 0.25 |
| Owen Randolph | Reporting | Create structured draft of report | 3 |
| Owen Randolph | Normalization | Check for 1NF Normalization - Python via ChatGPT | 0.5 |
| Owen Randolph | Normalization | Create 2NF tables for normalization - SQL script | 1 |
| Owen Randolph | Load Data | Add data to new tables - SQL script | 0.25 |
| Owen Randolph | Normalization | Add foreign keys for 3NF - SQL Script | 0.5 |
| Marcos Fernandez | Load Data | Loaded data from jupyter Notebook to MySQL Workbench | 0.25 |
| Marcos Fernandez | Check for dulplicate values | After data transfer from csv in Jupyter directory to MySQL workbench - Python | 0.25 |
| Marcos Fernandez | Normalization | Create 3NF tables for normalization - SQL script | 1 |
| Marcos Fernandez | Reporting | Refine Report Draft | 3 |

**Appendix**

