--- WEEK 7 TASK FOR 3SIGNET BY OKORIGWE CLINTON EGWOLOGHENE

---Data Cleaning and Validation steps

--- 1. Checking for NULL values in target columns

```
SELECT *

FROM Pharm_sales

WHERE Distributor IS NULL

  OR "Customer Name" IS NULL

  OR City IS NULL

  OR Country IS NULL

  OR Latitude IS NULL

  OR Longitude IS NULL

  OR Channel IS NULL

  OR "Product Name" IS NULL

  OR Quantity IS NULL

  OR Price IS NULL

  OR Sales IS NULL

  OR Month IS NULL

  OR Year IS NULL;
```

---2. Checking for duplication in the relevant column

```
  SELECT Distributor, "Customer Name", City, Country, Channel, "Product Name", Quantity, Month, Year, COUNT(*)

FROM Pharm_sales

GROUP BY Distributor, "Customer Name", City, Country, Channel, "Product Name", Quantity, Month, Year

HAVING COUNT(*) > 1;
```

---3. Removal of resulting duplicate columns fron the previous Query which resulted in 122 rows affected

```
DELETE FROM Pharm_sales

WHERE ROWID NOT IN (

   SELECT MIN(ROWID)

   FROM Pharm_sales

   GROUP BY Distributor, "Customer Name", City, Country, Channel, "Product Name", Quantity, Month, Year

);
```

---4. Standardize channel and product class values(I had already used the tools provided by sqlite browser to standardize the values for example, changing 'Price' and 'Sales' columns to REAL values)

```
UPDATE Pharm_sales

SET

   Channel = UPPER(SUBSTR(Channel, 1, 1)) || LOWER(SUBSTR(Channel, 2)),

   "Sub-channel" = UPPER(SUBSTR("Sub-channel", 1, 1)) || LOWER(SUBSTR("Sub-channel", 2)),

   "Product Class" = UPPER(SUBSTR("Product Class", 1, 1)) || LOWER(SUBSTR("Product Class", 2));
```

---5. Check and correction of Geolocation data

```
          SELECT *

FROM Pharm_sales

WHERE Latitude NOT BETWEEN -90 AND 90

  OR Longitude NOT BETWEEN -180 AND 180;
```

---6. Check that Quantity and Price contain only Numeric values

```
  SELECT *

FROM Pharm_sales

WHERE NOT Quantity GLOB '[0-9]*'

  OR NOT Price GLOB '[0-9.]*';
```

---7. Consistency checks for Month and Year

```
            SELECT *
FROM Pharm_sales
WHERE Month NOT BETWEEN 1 AND 12
   OR Year NOT BETWEEN 2017 AND 2020;
```

---8. Correct the spelling 'Alfa' to 'Alpha'

```
  UPDATE Pharm_sales
SET "Sales Team" = 'Alpha'
WHERE "Sales Team" = 'Alfa';
```

---9. Validate aggregate data to check for outliers

```
SELECT MIN(Sales), MAX(Sales), AVG(Sales)
FROM Pharm_sales;
```

---10. View table to confirm changes

```
select * from Pharm_sales
```

CREATION OF RELATIONSHIP TABLES FOR THE ER DIAGRAM AND UPDATING WITH RELEVANT VALUES

-- Create a table for Distributors

```
CREATE TABLE Distributor (
    DistributorID INTEGER PRIMARY KEY,
    DistributorName TEXT UNIQUE NOT NULL
);
INSERT INTO Distributor (DistributorName)
SELECT DISTINCT Distributor FROM Pharm_sales;
```

-- Create a table for Customers

```
CREATE TABLE Customer (
   CustomerID INTEGER PRIMARY KEY,
   CustomerName TEXT UNIQUE NOT NULL
);
INSERT INTO Customer (CustomerName)
SELECT DISTINCT "Customer Name" FROM Pharm_sales;
```

-- Create a table for Products

```
CREATE TABLE Product (
   ProductID INTEGER PRIMARY KEY,
   ProductName TEXT UNIQUE NOT NULL,
   ProductClass TEXT
);
INSERT INTO Product (ProductName, ProductClass)
SELECT DISTINCT "Product Name", "Product Class" FROM Pharm_sales;
```

-- Create a table for Sales Representatives

```
CREATE TABLE SalesRep (
   SalesRepID INTEGER PRIMARY KEY,
   Name TEXT UNIQUE NOT NULL,
   Manager TEXT,
   SalesTeam TEXT
);
INSERT INTO SalesRep (Name, Manager, SalesTeam)
SELECT DISTINCT "Name of Sales Rep", Manager, "Sales Team" FROM Pharm_sales;
```

```
-- Create a table for Channels
CREATE TABLE Channel (
    ChannelID INTEGER PRIMARY KEY,
    ChannelName TEXT,
    SubChannel TEXT
);
INSERT INTO Channel (ChannelName, SubChannel)
SELECT DISTINCT Channel, "Sub-channel" FROM Pharm_sales;


-- Create a table for Location data
CREATE TABLE Location (
    LocationID INTEGER PRIMARY KEY,
    City TEXT,
    Country TEXT,
    Latitude REAL,
    Longitude REAL
);
INSERT INTO Location (City, Country, Latitude, Longitude)
SELECT DISTINCT City, Country, Latitude, Longitude FROM Pharm_sales;


-- Create a new Sales table to store transactional data with foreign keys
CREATE TABLE Sales (
    SaleID INTEGER PRIMARY KEY,
    DistributorID INTEGER,
    CustomerID INTEGER,
    ProductID INTEGER,

    SalesRepID INTEGER,

    ChannelID INTEGER,

    LocationID INTEGER,

    Quantity INTEGER,

    Price REAL,

    Sales REAL,

    Month INTEGER,

    Year INTEGER,

    FOREIGN KEY (DistributorID) REFERENCES Distributor(DistributorID),

    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

    FOREIGN KEY (SalesRepID) REFERENCES SalesRep(SalesRepID),

    FOREIGN KEY (ChannelID) REFERENCES Channel(ChannelID),

    FOREIGN KEY (LocationID) REFERENCES Location(LocationID)

);
```

-- Insert data from the parent table into the Sales table

```
INSERT INTO Sales (DistributorID, CustomerID, ProductID, SalesRepID, ChannelID, LocationID, Quantity,
Price, Sales, Month, Year)

SELECT

    (SELECT DistributorID FROM Distributor WHERE DistributorName = Pharm_sales.Distributor),

    (SELECT CustomerID FROM Customer WHERE CustomerName = Pharm_sales."Customer Name"),

    (SELECT ProductID FROM Product WHERE ProductName = Pharm_sales."Product Name"),

    (SELECT SalesRepID FROM SalesRep WHERE Name = Pharm_sales."Name of Sales Rep"),
    (SELECT ChannelID FROM Channel WHERE ChannelName = Pharm_sales.Channel AND SubChannel = Pharm_sales."Sub-channel"),
    (SELECT LocationID FROM Location WHERE City = Pharm_sales.City AND Country = Pharm_sales.Country),
    Quantity, Price, Sales, Month, Year
FROM Pharm_sales;
```

--- Conduct data integrity check

```
PRAGMA integrity_check;
```

Integrity check returned “OK”

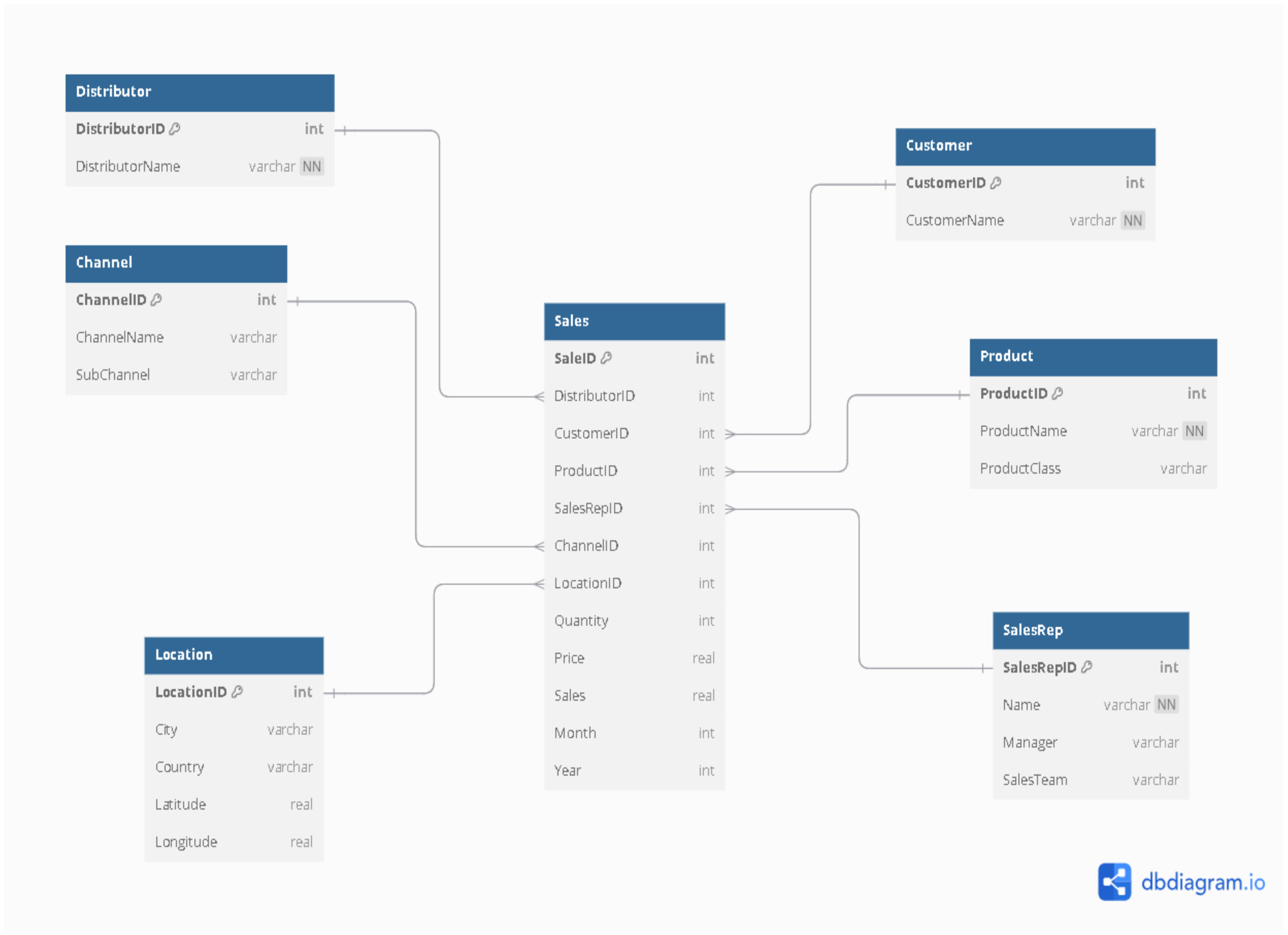
ER DIAGRAM

DBML Code for the creation of ER Diagram

```
// Distributor Table
Table Distributor {
   DistributorID int [pk, increment]
   DistributorName varchar [unique, not null]
}
// Customer Table
Table Customer {
   CustomerID int [pk, increment]
   CustomerName varchar [unique, not null]
}
// Product Table
Table Product {
   ProductID int [pk, increment]
   ProductName varchar [unique, not null]
```

```
  ProductClass varchar
}
// SalesRep Table
Table SalesRep {
  SalesRepID int [pk, increment]
  Name varchar [unique, not null]
  Manager varchar
  SalesTeam varchar
}

// Channel Table
Table Channel {
  ChannelID int [pk, increment]
  ChannelName varchar
  SubChannel varchar
}

// Location Table
Table Location {
  LocationID int [pk, increment]
  City varchar
  Country varchar
  Latitude real
  Longitude real
```

```
}


// Sales Table

Table Sales {

    SaleID int [pk, increment]

    DistributorID int [ref: > Distributor.DistributorID]

    CustomerID int [ref: > Customer.CustomerID]

    ProductID int [ref: > Product.ProductID]

    SalesRepID int [ref: > SalesRep.SalesRepID]

    ChannelID int [ref: > Channel.ChannelID]

    LocationID int [ref: > Location.LocationID]

    Quantity int

    Price real

    Sales real

    Month int

    Year int

}
```


Distributor
DistributorID int
DistributorName varchar NN
Channel
ChannelID int
ChannelName varchar
SubChannel varchar
Customer
CustomerID int
CustomerName varchar NN
Sales
SaleID int
DistributorID int
CustomerID int
ProductID int
SalesRepID int
ChannelID int
LocationID int
Quantity int
Price real
Sales real
Month int
Year int
Product
ProductID int
ProductName varchar NN
ProductClass varchar
SalesRep
SalesRepID int
Name varchar NN
Manager varchar
SalesTeam varchar
Location
LocationID int
City varchar
Country varchar
Latitude real
Longitude real
dbdiagram.io