

# INFO 579 Week 16 Final Project Report

Student's Full Name: Utkarsha Ajay Patil

Course Title: INFO 579: SQL/NoSQL Databases for Data and Information Sciences

Term name and year: Spring 2024

Submission Week: Week of May 1, 2024 [Week 16 Final Project Report]

Instructor's Name: Nayem Rahman

Date of Submission: 8<sup>th</sup> May 2024

## Topic: Week 16 Final Project Report

### Week 11 Final Project Update 1:

#### Pharmacy Store Data Management System

##### **Project Description:**

The Pharmacy Store Data Management System aims to develop a database system to efficiently manage various aspects of a pharmacy store. This system will facilitate the organization of customer information, medicine inventory, supplier and manufacturer details, as well as sales data. By implementing this system, the pharmacy store can streamline its operations, enhance customer service, and optimize inventory management processes.

Through the implementation of SQL queries, the project enables users to perform essential database operations such as data retrieval, insertion, deletion, and updating. This project serves as a practical exercise for learning SQL concepts and applying them to real-world scenarios within a pharmacy setting.

##### **The data utilized for the Pharmacy Store Data Management System project includes:**

1. Customer Data: Customer Name, Phone number, Address, Birthdate, Prescription
2. Medicines Data: Name, Generic Name, Supplier, Manufacturer, Category, Expiry Date, Purchase Price, Selling Price, Quantity in Stock
3. Supplier Data: Supplier Name
4. Manufacturer Data: Manufacturer Name
5. Sales Data: Number, Customer Name, Date, Medicines, Quantity Sold

##### **Data Sources:**

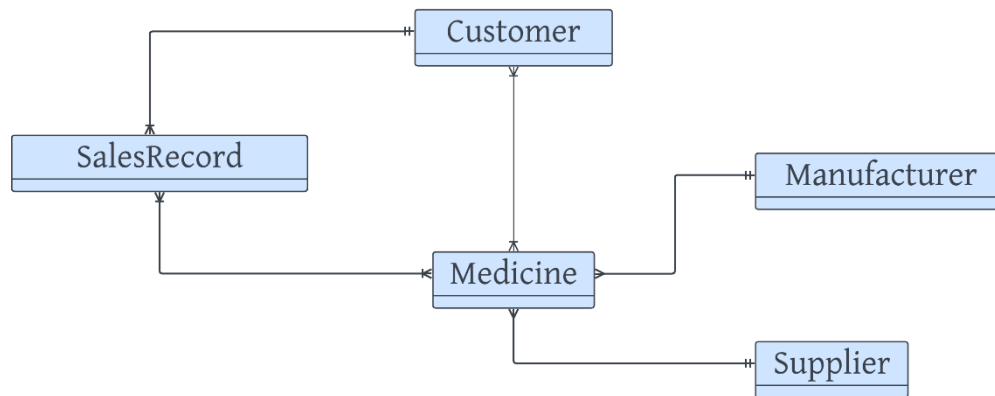
The data for the Pharmacy Store Data Management System was formulated by me. To enrich the dataset with accurate and relevant information, I referred to [https://www.drugs.com/drug\\_information.html](https://www.drugs.com/drug_information.html) as a reference source. This website provided valuable insights into crucial aspects of medicines such as manufacturer details, generic names, and categories.

I ensured that all tables and data within the database are related to each other, establishing connections between entities to maintain data integrity and facilitate efficient querying and reporting.

In conclusion, the Pharmacy Store Data Management System not only facilitates effective pharmacy management but also serves as a platform for me to learn and apply SQL query operations. Through this project, I can enhance my skills in database management and query optimization, contributing to a more proficient and informed approach to data-driven decision-making.

## Week 13 Final Project Update 2:

### Conceptual Model:



### Relationships:

**Customer to Medicine:** The relationship between medicine and customer can be conceptualized as a many-to-many relationship. This is because a single customer can purchase multiple medicines over time, and a single type of medicine can be purchased by multiple customers.

We see that each customer has a prescription that may consist of one or more medicines. For example:

- Jackson Wang has a prescription for Sudafed PE and Advil.
- Aditi Whagire has a prescription for Auvi-Q.
- Namjun has a prescription for Benadryl.

**SalesRecord to Medicine:** Each sale can involve multiple medicines, and each medicine can be involved in multiple sales transactions. This necessitates a many-to-many relationship between the sales and medicine tables.

Consider the sale transaction with ID 1:

Customer: Ashley Brown

Date: 07-Mar-23

Medicines: Banophen, Motrin

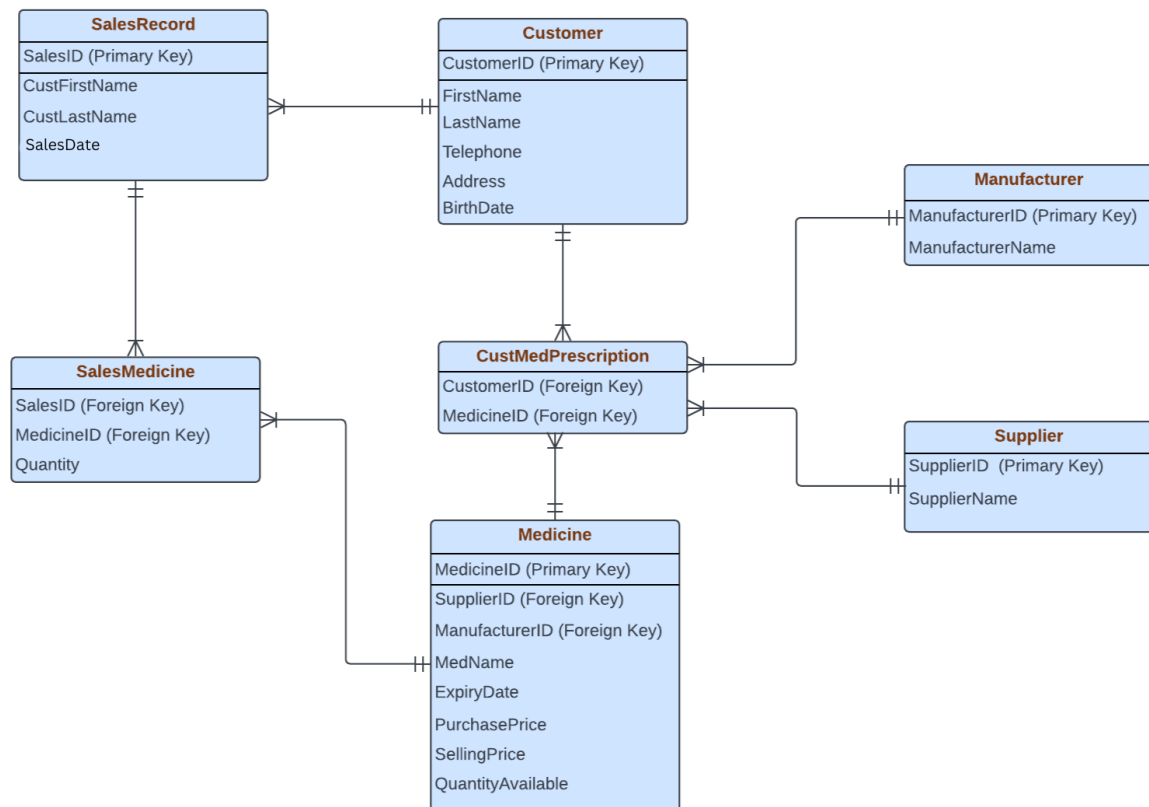
From this example, you can see that each sale transaction involves multiple medicines.

**Manufacturer to Medicine:** This is a one-to-many relationship. Each manufacturer can produce multiple medicines, but each medicine is produced by only one manufacturer.

**Medicine to Supplier:** This is a many-to-one relationship. Each medicine can be supplied by the one supplier in my dataset, and each supplier can supply multiple medicines.

**Customer to SalesRecord:** Each row in the sales table represents a single sale transaction, and each sale is associated with a single customer. However, a customer can have multiple sales transactions, indicating a one-to-many relationship between the customer and sales tables.

### Logical Model:

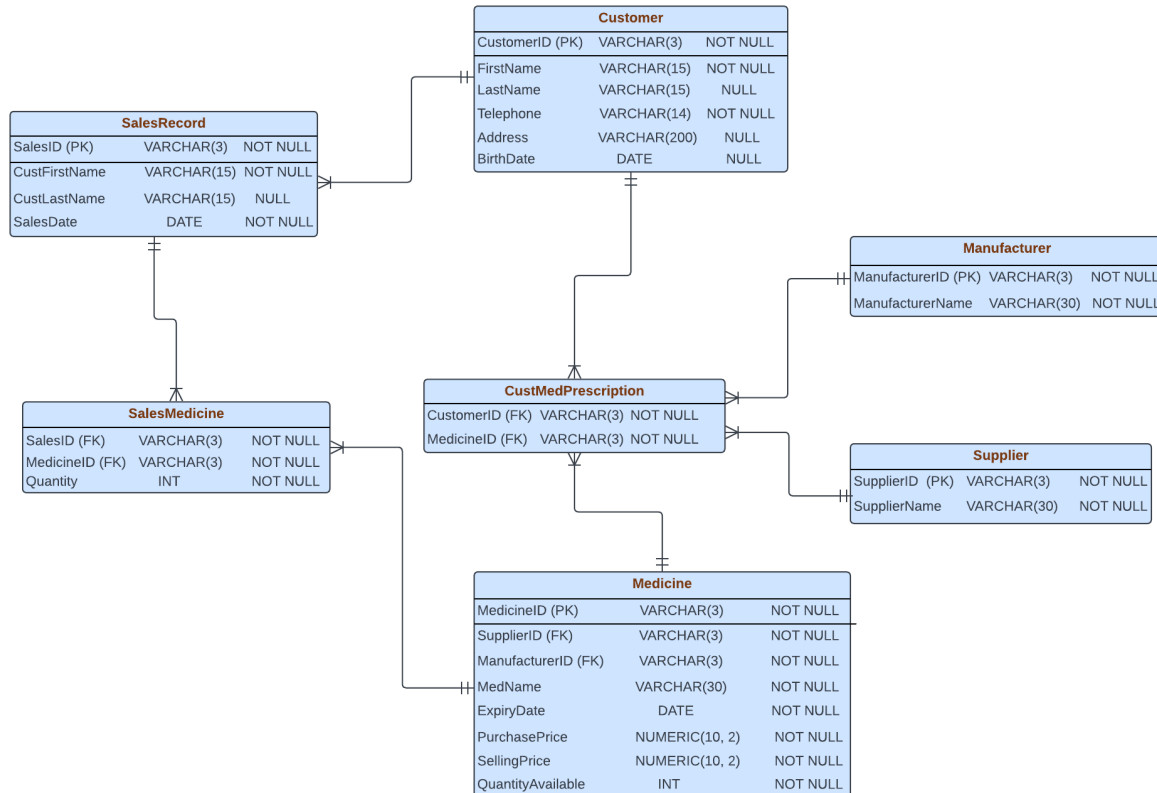


To address the many-to-many relationship between customers and medicines, a junction entity called CustMedPrescription has been created. This entity facilitates the association between customers and the medicines they are prescribed.

Additionally, to resolve the many-to-many relationship between sales records and medicines, a table named SalesMedicine has been introduced. This table enables the linkage between sales records and the medicines involved in each transaction.

## Week 16 Final Project Report:

### Physical Model:



The provided physical model represents a pharmacy database system that tracks customer information, sales transactions, medicines, manufacturers, and suppliers. The model includes several tables:

1. **Customer**: This table stores essential details about pharmacy customers such as CustomerID (primary key), first and last names, telephone number, address, and birthdate, facilitating personalized customer service and relationship management.
2. **SalesRecord**: Contains records of sales transactions, documenting SalesID (primary key), customer's first and last name directly associated with each sale, and the sales date, allowing for tracking of purchase history and sales trends.
3. **Medicine**: Holds information on medicines, including MedicineID (primary key), supplier and manufacturer IDs, medicine name, expiry date, purchase and selling prices, and quantity available. This table is crucial for managing pharmacy inventory and ensuring medicines are adequately stocked and priced.
4. **SalesMedicine**: Serves as a junction table that links SalesRecord to Medicine, detailing which medicines were sold in each transaction and in what quantity, supporting many-to-many relationships between sales and medicines.
5. **CustMedPrescription**: Another junction table that connects Customer to Medicine, recording which medicines have been prescribed to which customers, thereby illustrating the medical needs and history of each customer.
6. **Manufacturer**: Stores details about the medicine manufacturers, including ManufacturerID (primary key) and the manufacturer's name. This table is vital for tracing medicine origins, ensuring quality compliance, and facilitating manufacturer-related operations and communications.

7. **Supplier:** Maintains information on medicine suppliers, capturing SupplierID and the supplier's name, which is essential for managing supply chains and ordering processes.

## Creation of database & tables and insertion of data:

### 1. Creating database for Pharmacy Store:

Here I created database to save all the tables required for the pharmacy store.

```
CREATE DATABASE Pharmacy;  
SHOW DATABASES;  
USE Pharmacy;
```

```
1 • CREATE DATABASE Pharmacy;  
2 • SHOW DATABASES;  
3 • USE Pharmacy;  
.
```

### 2. Creating all the required tables, inserting and displaying the data:

1. Creating table for customer to include all their demographic data.

```
CREATE TABLE Pharmacy.Customer (  
    CustomerID VARCHAR(3) PRIMARY KEY,  
    FirstName VARCHAR(15) NOT NULL,  
    LastName VARCHAR(15) NULL,  
    Telephone VARCHAR(14) NOT NULL,  
    Address VARCHAR(200) NULL,  
    BirthDate DATE NULL  
);
```

```
13 -- Customer table  
14 • CREATE TABLE Pharmacy.Customer (  
15     CustomerID VARCHAR(3) PRIMARY KEY,  
16     FirstName VARCHAR(15) NOT NULL,  
17     LastName VARCHAR(15) NULL,  
18     Telephone VARCHAR(14) NOT NULL,  
19     Address VARCHAR(200) NULL,  
20     BirthDate DATE NULL  
21 );
```

```
INSERT INTO Pharmacy.Customer (CustomerID, FirstName, LastName, Telephone,  
Address, BirthDate) VALUES  
(1, 'Jackson', 'Wang', '(520) 599-5123', '345 South Street', '1967-03-02'),  
(2, 'Yoonki', 'Min', '(580) 599-8760', '780 Avenue Town', '2002-01-23'),  
(3, 'Milind', 'Ahire', '(334) 509-1990', '947 Down Town', '1987-09-05'),  
(4, 'Aditi', 'Whagire', '(990) 665-1657', '486 Campus Crossing', '1997-09-01'),
```

```
(5, 'Ashley', 'Brown', '(230) 695-1123', '204 U at Park', '1987-09-05'),
(6, 'Peter', 'Johnson', '(590) 655-1008', '106 A Mid Town', '2006-11-01'),
(7, 'Namjun', NULL, '(483) 249-3420', '15 Long Ave', '1994-11-11'),
(8, 'John', NULL, '(920) 243-4530', '24 J M Road', '1975-05-03'),
(9, 'Sohail', 'Khan', '(492) 249-3950', '12 Marine Drive', '1976-02-12'),
(10, 'Rohit', 'Verma', '(478) 555-8822', '15 Highland market', '2001-12-31');
```

```
23 • INSERT INTO Pharmacy.Customer (CustomerID, FirstName, LastName, Telephone, Address, BirthDate) VALUES
24 (1, 'Jackson', 'Wang', '(520) 599-5123', '345 South Street', '1967-03-02'),
25 (2, 'Yoonki', 'Min', '(580) 599-8760', '780 Avenue Town', '2002-01-23'),
26 (3, 'Milind', 'Ahire', '(334) 509-1990', '947 Down Town', '1987-09-05'),
27 (4, 'Aditi', 'Whagire', '(990) 665-1657', '486 Campus Crossing', '1997-09-01'),
28 (5, 'Ashley', 'Brown', '(230) 695-1123', '204 U at Park', '1987-09-05'),
29 (6, 'Peter', 'Johnson', '(590) 655-1008', '106 A Mid Town', '2006-11-01'),
30 (7, 'Namjun', NULL, '(483) 249-3420', '15 Long Ave', '1994-11-11'),
31 (8, 'John', NULL, '(920) 243-4530', '24 J M Road', '1975-05-03'),
32 (9, 'Sohail', 'Khan', '(492) 249-3950', '12 Marine Drive', '1976-02-12'),
33 (10, 'Rohit', 'Verma', '(478) 555-8822', '15 Highland market', '2001-12-31');
```

SELECT \* FROM Pharmacy.Customer;

```
34
35 • SELECT * FROM Pharmacy.Customer;
~
```

Result:

	CustomerID	FirstName	LastName	Telephone	Address	BirthDate
▶	1	Jackson	Wang	(520) 599-5123	345 South Street	1967-03-02
	10	Rohit	Verma	(478) 555-8822	15 Highland market	2001-12-31
	2	Yoonki	Min	(580) 599-8760	780 Avenue Town	2002-01-23
	3	Milind	Ahire	(334) 509-1990	947 Down Town	1987-09-05
	4	Aditi	Whagire	(990) 665-1657	486 Campus Crossing	1997-09-01
	5	Ashley	Brown	(230) 695-1123	204 U at Park	1987-09-05
	6	Peter	Johnson	(590) 655-1008	106 A Mid Town	2006-11-01
	7	Namjun	NULL	(483) 249-3420	15 Long Ave	1994-11-11
	8	John	NULL	(920) 243-4530	24 J M Road	1975-05-03
	9	Sohail	Khan	(492) 249-3950	12 Marine Drive	1976-02-12

2. Creating table for supplier to include all suppliers that provide medicines to the store.

```
CREATE TABLE Pharmacy.Supplier (
    SupplierID VARCHAR(3) PRIMARY KEY,
    SupplierName VARCHAR(30) NOT NULL
);
```

```
37 -- Supplier table
38 • CREATE TABLE Pharmacy.Supplier (
39     SupplierID VARCHAR(3) PRIMARY KEY,
40     SupplierName VARCHAR(30) NOT NULL
41 );
```



```
INSERT INTO Pharmacy.Supplier (SupplierID, SupplierName) VALUES
('S1', 'Medicare'),
('S2', 'Medline'),
('S3', 'Cardinal Health'),
('S4', 'Lexicon'),
('S5', 'US Pharma'),
('S6', 'DSS Health');
```

```
43 • INSERT INTO Pharmacy.Supplier (SupplierID, SupplierName) VALUES
44     ('S1', 'Medicare'),
45     ('S2', 'Medline'),
46     ('S3', 'Cardinal Health'),
47     ('S4', 'Lexicon'),
48     ('S5', 'US Pharma'),
49     ('S6', 'DSS Health');
```

```
SELECT * FROM Pharmacy.Supplier;
```

```
51 • SELECT * FROM Pharmacy.Supplier;
--
```

Result:

Result Grid   Filter Rows:		
	SupplierID	SupplierName
▶	S1	Medicare
	S2	Medline
	S3	Cardinal Health
	S4	Lexicon
	S5	US Pharma
	S6	DSS Health

3. Creating table for Manufacturer to include all manufacturers that produce medicines needed by the store.

```
CREATE TABLE Pharmacy.Manufacturer (
    ManufacturerID VARCHAR(3) PRIMARY KEY,
    ManufacturerName VARCHAR(30) NOT NULL
);
```

```
54 • CREATE TABLE Pharmacy.Manufacturer (
55     ManufacturerID VARCHAR(3) PRIMARY KEY,
56     ManufacturerName VARCHAR(30) NOT NULL
57 );
```

```
INSERT INTO Pharmacy.Manufacturer (ManufacturerID, ManufacturerName) VALUES
('M1', 'Kaleo'),
('M2', 'Pfizer'),
('M3', 'Johnson & Johnson'),
('M4', 'Major Pharmaceuticals'),
('M5', 'Amphastar Pharmaceuticals'),
('M6', 'Amgen');
```




('M7', 'Biovista');

```
59 • INSERT INTO Pharmacy.Manufacturer (ManufacturerID, ManufacturerName) VALUES
60     ('M1', 'Kaleo'),
61     ('M2', 'Pfizer'),
62     ('M3', 'Johnson & Johnson'),
63     ('M4', 'Major Pharmaceuticals'),
64     ('M5', 'Amphastar Pharmaceuticals'),
65     ('M6', 'Amgen'),
66     ('M7', 'Biovista');
```

SELECT \* FROM Pharmacy.Manufacturer;

```
68 • SELECT * FROM Pharmacy.Manufacturer;
```

Result:

Result Grid    Filter Rows: <input type="text"/>		
	ManufacturerID	ManufacturerName
▶	M1	Kaleo
	M2	Pfizer
	M3	Johnson & Johnson
	M4	Major Pharmaceuticals
	M5	Amphastar Pharmaceuticals
	M6	Amgen
	M7	Biovista

4. Creating table for medicines to include all medicines that store sells.

```
CREATE TABLE Pharmacy.Medicine (
    MedicineID VARCHAR(3) PRIMARY KEY,
    SupplierID VARCHAR(3) NOT NULL,
    ManufacturerID VARCHAR(3) NOT NULL,
    MedName VARCHAR(30) NOT NULL,
    ExpiryDate DATE NOT NULL,
    PurchasePrice NUMERIC(10, 2) NOT NULL,
    SellingPrice NUMERIC(10, 2) NOT NULL,
    QuantityAvailable INT NOT NULL,
    FOREIGN KEY (SupplierID) REFERENCES Pharmacy.Supplier(SupplierID),
    FOREIGN KEY (ManufacturerID) REFERENCES
Pharmacy.Manufacturer(ManufacturerID));
```

```
71 • CREATE TABLE Pharmacy.Medicine (
72     MedicineID VARCHAR(3) PRIMARY KEY,
73     SupplierID VARCHAR(3) NOT NULL,
74     ManufacturerID VARCHAR(3) NOT NULL,
75     MedName VARCHAR(30) NOT NULL,
76     ExpiryDate DATE NOT NULL,
77     PurchasePrice NUMERIC(10, 2) NOT NULL,
78     SellingPrice NUMERIC(10, 2) NOT NULL,
79     QuantityAvailable INT NOT NULL,
80     FOREIGN KEY (SupplierID) REFERENCES Pharmacy.Supplier(SupplierID),
81     FOREIGN KEY (ManufacturerID) REFERENCES Pharmacy.Manufacturer(ManufacturerID)
82 );
```



```

INSERT INTO Pharmacy.Medicine (MedicineID, MedName, ExpiryDate, PurchasePrice,
SellingPrice, QuantityAvailable, SupplierID, ManufacturerID) VALUES
('Q1', 'Auvi-Q', '2025-03-07', 30, 35, 13, 'S1', 'M1'),
('Q2', 'Advil', '2025-05-17', 13, 15, 50, 'S2', 'M2'),
('Q3', 'Banophen', '2024-08-15', 20, 22, 3, 'S1', 'M4'),
('Q4', 'Sudafed PE', '2025-03-12', 10, 12, 30, 'S3', 'M3'),
('Q5', 'Motrin', '2024-06-07', 11, 12, 25, 'S2', 'M3'),
('Q6', 'Arthrotec', '2027-02-20', 35, 38, 7, 'S4', 'M2'),
('Q7', 'Primatene Mist', '2025-08-15', 40, 42, 13, 'S3', 'M5'),
('Q8', 'Benadryl', '2024-12-11', 8, 11, 58, 'S2', 'M3');

```

```

84 • INSERT INTO Pharmacy.Medicine (MedicineID, MedName, ExpiryDate, PurchasePrice, SellingPrice, QuantityAvailable
85   , SupplierID, ManufacturerID) VALUES
86   ('Q1', 'Auvi-Q', '2025-03-07', 30, 35, 13, 'S1', 'M1'),
87   ('Q2', 'Advil', '2025-05-17', 13, 15, 50, 'S2', 'M2'),
88   ('Q3', 'Banophen', '2024-08-15', 20, 22, 3, 'S1', 'M4'),
89   ('Q4', 'Sudafed PE', '2025-03-12', 10, 12, 30, 'S3', 'M3'),
90   ('Q5', 'Motrin', '2024-06-07', 11, 12, 25, 'S2', 'M3'),
91   ('Q6', 'Arthrotec', '2027-02-20', 35, 38, 7, 'S4', 'M2'),
92   ('Q7', 'Primatene Mist', '2025-08-15', 40, 42, 13, 'S3', 'M5'),
93   ('Q8', 'Benadryl', '2024-12-11', 8, 11, 58, 'S2', 'M3');

```

SELECT \* FROM Pharmacy.Medicine;

```

95 • SELECT * FROM Pharmacy.Medicine;

```

Result:

	MedicineID	SupplierID	ManufacturerID	MedName	ExpiryDate	PurchasePrice	SellingPrice	QuantityAvailable
►	Q1	S1	M1	Auvi-Q	2025-03-07	30.00	35.00	13
	Q2	S2	M2	Advil	2025-05-17	13.00	15.00	50
	Q3	S1	M4	Banophen	2024-08-15	20.00	22.00	3
	Q4	S3	M3	Sudafed PE	2025-03-12	10.00	12.00	30
	Q5	S2	M3	Motrin	2024-06-07	11.00	12.00	25
	Q6	S4	M2	Arthrotec	2027-02-20	35.00	38.00	7
	Q7	S3	M5	Primatene Mist	2025-08-15	40.00	42.00	13
	Q8	S2	M3	Benadryl	2024-12-11	8.00	11.00	58

5. Creating connecting table for prescription to include all medicines that each customer has been prescribed.

```

CREATE TABLE Pharmacy.CustMedPrescription (
    CustomerID VARCHAR(3) NOT NULL,
    MedicineID VARCHAR(3) NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES Pharmacy.Customer(CustomerID),
    FOREIGN KEY (MedicineID) REFERENCES Pharmacy.Medicine(MedicineID));

```

```

97   -- Prescription table
98 • CREATE TABLE Pharmacy.CustMedPrescription (
99     CustomerID VARCHAR(3) NOT NULL,
100     MedicineID VARCHAR(3) NOT NULL,
101     FOREIGN KEY (CustomerID) REFERENCES Pharmacy.Customer(CustomerID),
102     FOREIGN KEY (MedicineID) REFERENCES Pharmacy.Medicine(MedicineID)
103   );

```

```

INSERT INTO Pharmacy.CustMedPrescription (CustomerID, MedicineID) VALUES
(1, 'Q4'),
(1, 'Q2'),
(2, 'Q3'),
(2, 'Q6'),
(3, 'Q2'),
(4, 'Q1'),
(5, 'Q4'),
(5, 'Q3'),
(5, 'Q5'),
(6, 'Q6'),
(6, 'Q2'),
(7, 'Q8'),
(8, 'Q5'),
(9, 'Q8'),
(9, 'Q3'),
(10, 'Q4'),
(10, 'Q5');

```

```

105 • INSERT INTO Pharmacy.CustMedPrescription (CustomerID, MedicineID) VALUES
106     (1, 'Q4'),
107     (1, 'Q2'),
108     (2, 'Q3'),
109     (2, 'Q6'),
110     (3, 'Q2'),
111     (4, 'Q1'),
112     (5, 'Q4'),
113     (5, 'Q3'),
114     (5, 'Q5'),
115     (6, 'Q6'),
116     (6, 'Q2'),
117     (7, 'Q8'),
118     (8, 'Q5'),
119     (9, 'Q8'),
120     (9, 'Q3'),
121     (10, 'Q4'),
122     (10, 'Q5');

```

```

SELECT * FROM Pharmacy.CustMedPrescription;

```

```

124 • SELECT * FROM Pharmacy.CustMedPrescription;

```

Result:

	CustomerID	MedicineID
▶	1	Q4
	1	Q2
	2	Q3
	2	Q6
	3	Q2
	4	Q1
	5	Q4
	5	Q3
	5	Q5
	6	Q6
	6	Q2
	7	Q8
	8	Q5
	9	Q8
	9	Q3
	10	Q4
	10	Q5

6. Creating table for sales data to include sales records.

```
CREATE TABLE Pharmacy.SalesRecord (
    SalesID VARCHAR(3) PRIMARY KEY,
    CustFirstName VARCHAR(15) NOT NULL,
    CustLastName VARCHAR(15) NULL,
    SalesDate DATE NOT NULL
);
```

127 • CREATE TABLE Pharmacy.SalesRecord (  
128       SalesID VARCHAR(3) PRIMARY KEY,  
129       CustFirstName VARCHAR(15) NOT NULL,  
130       CustLastName VARCHAR(15) NULL,  
131       SalesDate DATE NOT NULL  
132     );

```
INSERT INTO Pharmacy.SalesRecord (SalesID, CustFirstName, CustLastName,
SalesDate) VALUES
```

```
('1', 'Ashley', 'Brown', '2023-03-07'),
('2', 'Jackson', 'Wang', '2024-01-12'),
('3', 'Peter', 'Johnson', '2022-07-10'),
('4', 'Aditi', 'Whagire', '2022-07-10'),
('5', 'Yoonki', 'Min', '2023-05-17'),
('6', 'Sohail', 'Khan', '2022-01-25'),
('7', 'John', NULL, '2023-04-13'),
('8', 'Rohit', 'Verma', '2024-02-20'),
('9', 'Jackson', 'Wang', '2022-12-11'),
('10', 'Sohail', 'Khan', '2023-09-26'),
('11', 'Namjun', NULL, '2023-09-26'),
('12', 'Yoonki', 'Min', '2024-01-01'),
('13', 'Ashley', 'Brown', '2022-06-13'),
('14', 'Peter', 'Johnson', '2024-02-22'),
('15', 'Rohit', 'Verma', '2022-07-13'),
('16', 'Ashley', 'Brown', '2023-08-13'),
('17', 'Jackson', 'Wang', '2023-01-25');
```

```

134 • INSERT INTO Pharmacy.SalesRecord (SalesID, CustFirstName, CustLastName, SalesDate) VALUES
135 ('1', 'Ashley', 'Brown', '2023-03-07'),
136 ('2', 'Jackson', 'Wang', '2024-01-12'),
137 ('3', 'Peter', 'Johnson', '2022-07-10'),
138 ('4', 'Aditi', 'Whagire', '2022-07-10'),
139 ('5', 'Yoonki', 'Min', '2023-05-17'),
140 ('6', 'Sohail', 'Khan', '2022-01-25'),
141 ('7', 'John', NULL, '2023-04-13'),
142 ('8', 'Rohit', 'Verma', '2024-02-20'),
143 ('9', 'Jackson', 'Wang', '2022-12-11'),
144 ('10', 'Sohail', 'Khan', '2023-09-26'),
145 ('11', 'Namjun', NULL, '2023-09-26'),
146 ('12', 'Yoonki', 'Min', '2024-01-01'),
147 ('13', 'Ashley', 'Brown', '2022-06-13'),
148 ('14', 'Peter', 'Johnson', '2024-02-22'),
149 ('15', 'Rohit', 'Verma', '2022-07-13'),
150 ('16', 'Ashley', 'Brown', '2023-08-13'),
151 ('17', 'Jackson', 'Wang', '2023-01-25');

```

SELECT \* FROM Pharmacy.SalesRecord;

```

153 • SELECT * FROM Pharmacy.SalesRecord;
---
```

Result:

	SalesID	CustFirstName	CustLastName	SalesDate
▶	1	Ashley	Brown	2023-03-07
	10	Sohail	Khan	2023-09-26
	11	Namjun	NULL	2023-09-26
	12	Yoonki	Min	2024-01-01
	13	Ashley	Brown	2022-06-13
	14	Peter	Johnson	2024-02-22
	15	Rohit	Verma	2022-07-13
	16	Ashley	Brown	2023-08-13
	17	Jackson	Wang	2023-01-25
	2	Jackson	Wang	2024-01-12
	3	Peter	Johnson	2022-07-10
	4	Aditi	Whagire	2022-07-10
	5	Yoonki	Min	2023-05-17
	6	Sohail	Khan	2022-01-25
	7	John	NULL	2023-04-13
	8	Rohit	Verma	2024-02-20
	9	Jackson	Wang	2022-12-11

7. Creating connecting table for medicines and sales records .

```

CREATE TABLE Pharmacy.SalesMedicine (
    SalesID VARCHAR(3) NOT NULL,
    MedicineID VARCHAR(3) NOT NULL,
    Quantity INT NOT NULL,
    FOREIGN KEY (SalesID) REFERENCES Pharmacy.SalesRecord(SalesID),
    FOREIGN KEY (MedicineID) REFERENCES Pharmacy.Medicine(MedicineID)
);

```

```

156 ● CREATE TABLE Pharmacy.SalesMedicine (
157     SalesID VARCHAR(3) NOT NULL,
158     MedicineID VARCHAR(3) NOT NULL,
159     Quantity INT NOT NULL,
160     FOREIGN KEY (SalesID) REFERENCES Pharmacy.SalesRecord(SalesID),
161     FOREIGN KEY (MedicineID) REFERENCES Pharmacy.Medicine(MedicineID)
162 );

```

INSERT INTO Pharmacy.SalesMedicine (SalesID, MedicineID, Quantity) VALUES

```

('1', 'Q3', 2),
('1', 'Q5', 4),
('2', 'Q2', 2),
('3', 'Q2', 5),
('4', 'Q1', 3),
('5', 'Q6', 1),
('6', 'Q8', 1),
('7', 'Q5', 5),
('8', 'Q5', 2),
('9', 'Q4', 1),
('9', 'Q2', 3),
('10', 'Q3', 2),
('11', 'Q8', 1),
('12', 'Q3', 3),
('12', 'Q6', 1),
('13', 'Q4', 1),
('14', 'Q6', 5),
('15', 'Q4', 1),
('15', 'Q5', 1),
('16', 'Q4', 1),
('16', 'Q3', 1),
('16', 'Q5', 1),
('17', 'Q2', 2);

```

```

164 ● INSERT INTO Pharmacy.SalesMedicine (SalesID, MedicineID, Quantity) VALUES
165     ('1', 'Q3', 2),
166     ('1', 'Q5', 4),
167     ('2', 'Q2', 2),
168     ('3', 'Q2', 5),
169     ('4', 'Q1', 3),
170     ('5', 'Q6', 1),
171     ('6', 'Q8', 1),
172     ('7', 'Q5', 5),
173     ('8', 'Q5', 2),
174     ('9', 'Q4', 1),
175     ('9', 'Q2', 3),
176     ('10', 'Q3', 2),
177     ('11', 'Q8', 1),
178     ('12', 'Q3', 3),
179     ('12', 'Q6', 1),
180     ('13', 'Q4', 1),
181     ('14', 'Q6', 5),

```

---

```

182      ('15', 'Q4', 1),
183      ('15', 'Q5', 1),
184      ('16', 'Q4', 1),
185      ('16', 'Q3', 1),
186      ('16', 'Q5', 1),
187      ('17', 'Q2', 2);
188
189 • SELECT * FROM Pharmacy.SalesMedicine;

```

Result:

	SalesID	MedicineID	Quantity
▶	1	Q3	2
	1	Q5	4
	2	Q2	2
	3	Q2	5
	4	Q1	3
	5	Q6	1
	6	Q8	1
	7	Q5	5
	8	Q5	2
	9	Q4	1
	9	Q2	3
	10	Q3	2
	11	Q8	1
	12	Q3	3
	12	Q6	1
	13	Q4	1
	14	Q6	5
	15	Q4	1
	15	Q5	1
	16	Q4	1
	16	Q3	1
	16	Q5	1
	17	Q2	2

7. Create a variety of SQL queries to retrieve data from one or many tables:

1. Retrieve the data from each table by using the SELECT \* statement and order by PK column(s).

Show the output. Make sure you show the print screen of the complete set of rows and columns.

The rows must be ordered by PK column(s).

### **Retrieving all records using SELECT clause and ordered by Primary Key:**

1. Customer Table:

Here I am using CAST as I using using VARCHAR for all the ID's so I can use alphabets as well in other records.

```

SELECT * FROM Pharmacy.Customer
ORDER BY CAST(CustomerID AS UNSIGNED);

```

```

1      -- 1
2      -- For the Customer Table
3  ●    SELECT * FROM Pharmacy.Customer
4      ORDER BY CAST(CustomerID AS UNSIGNED);

```

Result:

	CustomerID	FirstName	LastName	Telephone	Address	BirthDate
▶	1	Jackson	Wang	(520) 599-5123	345 South Street	1967-03-02
	2	Yoonki	Min	(580) 599-8760	780 Avenue Town	2002-01-23
	3	Milind	Ahire	(334) 509-1990	947 Down Town	1987-09-05
	4	Aditi	Whagire	(990) 665-1657	486 Campus Crossing	1997-09-01
	5	Ashley	Brown	(230) 695-1123	204 U at Park	1987-09-05
	6	Peter	Johnson	(590) 655-1008	106 A Mid Town	2006-11-01
	7	Namjun	NULL	(483) 249-3420	15 Long Ave	1994-11-11
	8	John	NULL	(920) 243-4530	24 J M Road	1975-05-03
	9	Sohail	Khan	(492) 249-3950	12 Marine Drive	1976-02-12
	10	Rohit	Verma	(478) 555-8822	15 Highland market	2001-12-31

## 2. Supplier Table:

```

SELECT * FROM Pharmacy.Supplier
ORDER BY SupplierID;

```

```

6      -- For the Supplier Table
7  ●    SELECT * FROM Pharmacy.Supplier
8      ORDER BY SupplierID;

```

Result:

	SupplierID	SupplierName
▶	S1	Medicare
	S2	Medline
	S3	Cardinal Health
	S4	Lexicon
	S5	US Pharma
	S6	DSS Health

## 3. Manufacturer Table:

```

SELECT * FROM Pharmacy.Manufacturer
ORDER BY ManufacturerID;

```

```

10     -- For the Manufacturer Table
11  ●    SELECT * FROM Pharmacy.Manufacturer
12     ORDER BY ManufacturerID;

```

Result:



	ManufacturerID	ManufacturerName
►	M1	Kaleo
	M2	Pfizer
	M3	Johnson & Johnson
	M4	Major Pharmaceuticals
	M5	Amphastar Pharmaceuticals
	M6	Amgen
	M7	Biovista

#### 4. Medicine Table:

```
SELECT * FROM Pharmacy.Medicine
ORDER BY MedicineID;
```

```
14      -- For the Medicine Table
15 •    SELECT * FROM Pharmacy.Medicine
16      ORDER BY MedicineID;
17
```

Result:

	MedicineID	SupplierID	ManufacturerID	MedName	ExpiryDate	PurchasePrice	SellingPrice	QuantityAvailable
►	Q1	S1	M1	Auvi-Q	2025-03-07	30.00	35.00	13
	Q2	S2	M2	Advil	2025-05-17	13.00	15.00	50
	Q3	S1	M4	Banophen	2024-08-15	20.00	22.00	3
	Q4	S3	M3	Sudafed PE	2025-03-12	10.00	12.00	30
	Q5	S2	M3	Motrin	2024-06-07	11.00	12.00	25
	Q6	S4	M2	Arthrotec	2027-02-20	35.00	38.00	7
	Q7	S3	M5	Primatene Mist	2025-08-15	40.00	42.00	13
	Q8	S2	M3	Benadryl	2024-12-11	8.00	11.00	58

#### 5. SalesRecord Table:

```
SELECT * FROM Pharmacy.SalesRecord
ORDER BY CAST(SalesID AS UNSIGNED);
```

```
18      -- For the SalesRecord Table
19 •    SELECT * FROM Pharmacy.SalesRecord
20      ORDER BY CAST(SalesID AS UNSIGNED);
```

Result:

	SalesID	CustFirstName	CustLastName	SalesDate
▶	1	Ashley	Brown	2023-03-07
	2	Jackson	Wang	2024-01-12
	3	Peter	Johnson	2022-07-10
	4	Aditi	Whagire	2022-07-10
	5	Yoonki	Min	2023-05-17
	6	Sohail	Khan	2022-01-25
	7	John	NULL	2023-04-13
	8	Rohit	Verma	2024-02-20
	9	Jackson	Wang	2022-12-11
	10	Sohail	Khan	2023-09-26
	11	Namjun	NULL	2023-09-26
	12	Yoonki	Min	2024-01-01
	13	Ashley	Brown	2022-06-13
	14	Peter	Johnson	2024-02-22
	15	Rohit	Verma	2022-07-13
	16	Ashley	Brown	2023-08-13
	17	Jackson	Wang	2023-01-25

#### 6. SalesMedicine Table:

```

SELECT * FROM Pharmacy.SalesMedicine
ORDER BY CAST(SalesID AS UNSIGNED), MedicineID;

22      -- For the SalesMedicine Table
23 •     SELECT * FROM Pharmacy.SalesMedicine
24      ORDER BY CAST(SalesID AS UNSIGNED), MedicineID;

```

Result:

	SalesID	MedicineID	Quantity
▶	1	Q3	2
	1	Q5	4
	2	Q2	2
	3	Q2	5
	4	Q1	3
	5	Q6	1
	6	Q8	1
	7	Q5	5
	8	Q5	2
	9	Q2	3
	9	Q4	1
	10	Q3	2
	11	Q8	1
	12	Q3	3
	12	Q6	1
	13	Q4	1
	14	Q6	5
	15	Q4	1
	15	Q5	1
	16	Q3	1
	16	Q4	1
	16	Q5	1
	17	Q2	2

7. CustMedPrescription Table:

```
SELECT * FROM Pharmacy.CustMedPrescription
ORDER BY CAST(CustomerID AS UNSIGNED), MedicineID;
```

```
26      -- For the CustMedPrescription Table
27 •    SELECT * FROM Pharmacy.CustMedPrescription
28      ORDER BY CAST(CustomerID AS UNSIGNED), MedicineID;
```

Result:

	CustomerID	MedicineID
▶	1	Q2
	1	Q4
	2	Q3
	2	Q6
	3	Q2
	4	Q1
	5	Q3
	5	Q4
	5	Q5
	6	Q2
	6	Q6
	7	Q8
	8	Q5
	9	Q3
	9	Q8
	10	Q4
	10	Q5

2. Write an SQL involving the junction table and two other related tables. You must use the INNER JOIN to connect with all three tables. The database that you created must be included in your SQL queries.

**SQL Query Involving Junction Table with INNER JOINS:**

SELECT

sr.SalesID,  
sr.CustFirstName,  
sr.CustLastName,  
sm.MedicineID,  
m.MedName,  
sm.Quantity,  
sr.SalesDate

FROM

Pharmacy.SalesMedicine sm

INNER JOIN

Pharmacy.SalesRecord sr ON sm.SalesID = sr.SalesID

INNER JOIN

Pharmacy.Medicine m ON sm.MedicineID = m.MedicineID

ORDER BY

CAST(sr.SalesID AS UNSIGNED), sm.MedicineID;

```
30      -- 2
31      -- SQL Query Involving Junction Table with INNER JOINS
32  •   SELECT
33          sr.SalesID,
34          sr.CustFirstName,
35          sr.CustLastName,
36          sm.MedicineID,
37          m.MedName,
38          sm.Quantity,
39          sr.SalesDate
40  FROM
41      Pharmacy.SalesMedicine sm
42  INNER JOIN
43      Pharmacy.SalesRecord sr ON sm.SalesID = sr.SalesID
44  INNER JOIN
45      Pharmacy.Medicine m ON sm.MedicineID = m.MedicineID
46  ORDER BY
47      CAST(sr.SalesID AS UNSIGNED), sm.MedicineID;
```

Result:

	SalesID	CustFirstName	CustLastName	MedicineID	MedName	Quantity	SalesDate
▶	1	Ashley	Brown	Q3	Banophen	2	2023-03-07
	1	Ashley	Brown	Q5	Motrin	4	2023-03-07
	2	Jackson	Wang	Q2	Advil	2	2024-01-12
	3	Peter	Johnson	Q2	Advil	5	2022-07-10
	4	Aditi	Whagire	Q1	Auvi-Q	3	2022-07-10
	5	Yoonki	Min	Q6	Arthrotec	1	2023-05-17
	6	Sohail	Khan	Q8	Benadryl	1	2022-01-25
	7	John	NULL	Q5	Motrin	5	2023-04-13
	8	Rohit	Verma	Q5	Motrin	2	2024-02-20
	9	Jackson	Wang	Q2	Advil	3	2022-12-11
	9	Jackson	Wang	Q4	Sudafed PE	1	2022-12-11
	10	Sohail	Khan	Q3	Banophen	2	2023-09-26
	11	Namjun	NULL	Q8	Benadryl	1	2023-09-26
	12	Yoonki	Min	Q3	Banophen	3	2024-01-01
	12	Yoonki	Min	Q6	Arthrotec	1	2024-01-01
	13	Ashley	Brown	Q4	Sudafed PE	1	2022-06-13
	14	Peter	Johnson	Q6	Arthrotec	5	2024-02-22
	15	Rohit	Verma	Q4	Sudafed PE	1	2022-07-13
	15	Rohit	Verma	Q5	Motrin	1	2022-07-13
	16	Ashley	Brown	Q3	Banophen	1	2023-08-13
	16	Ashley	Brown	Q4	Sudafed PE	1	2023-08-13
	16	Ashley	Brown	Q5	Motrin	1	2023-08-13
	17	Jackson	Wang	Q2	Advil	2	2023-01-25

This query would be very useful in a retail scenario to report or analyze the details of what products (medicines) were bought by which customer on specific dates, which can help in inventory management, customer purchase behavior analysis, and sales tracking.

3. Write an SQL by including two or more tables and using the LEFT OUTER JOIN. Show the results and sort the results by key field(s). Interpret the results compared to what an INNER JOIN does.

#### **SQL Query Using LEFT OUTER JOIN:**

SELECT

sr.SalesID,

sr.CustFirstName,

sr.CustLastName,

sr.SalesDate,

sm.MedicineID,

sm.Quantity

FROM

Pharmacy.SalesRecord sr

LEFT OUTER JOIN

Pharmacy.SalesMedicine sm ON sr.SalesID = sm.SalesID

ORDER BY CAST(sr.SalesID AS UNSIGNED);

```

50      -- 3
51      -- SQL Query Using LEFT OUTER JOIN
52      • SELECT
53          sr.SalesID,
54          sr.CustFirstName,
55          sr.CustLastName,
56          sr.SalesDate,
57          sm.MedicineID,
58          sm.Quantity
59      FROM
60          Pharmacy.SalesRecord sr
61      LEFT OUTER JOIN
62          Pharmacy.SalesMedicine sm ON sr.SalesID = sm.SalesID
63      ORDER BY CAST(sr.SalesID AS UNSIGNED);
64

```

Result:

	SalesID	CustFirstName	CustLastName	SalesDate	MedicineID	Quantity
▶	1	Ashley	Brown	2023-03-07	Q3	2
	1	Ashley	Brown	2023-03-07	Q5	4
	2	Jackson	Wang	2024-01-12	Q2	2
	3	Peter	Johnson	2022-07-10	Q2	5
	4	Aditi	Whagire	2022-07-10	Q1	3
	5	Yoonki	Min	2023-05-17	Q6	1
	6	Sohail	Khan	2022-01-25	Q8	1
	7	John	NULL	2023-04-13	Q5	5
	8	Rohit	Verma	2024-02-20	Q5	2
	9	Jackson	Wang	2022-12-11	Q4	1
	9	Jackson	Wang	2022-12-11	Q2	3
	10	Sohail	Khan	2023-09-26	Q3	2
	11	Namjun	NULL	2023-09-26	Q8	1
	12	Yoonki	Min	2024-01-01	Q3	3
	12	Yoonki	Min	2024-01-01	Q6	1
	13	Ashley	Brown	2022-06-13	Q4	1
	14	Peter	Johnson	2024-02-22	Q6	5
	15	Rohit	Verma	2022-07-13	Q4	1
	15	Rohit	Verma	2022-07-13	Q5	1
	16	Ashley	Brown	2023-08-13	Q4	1
	16	Ashley	Brown	2023-08-13	Q3	1
	16	Ashley	Brown	2023-08-13	Q5	1
	17	Jackson	Wang	2023-01-25	Q2	2

This join includes all records from the left table (SalesRecord), regardless of whether there is a corresponding entry in the right table (SalesMedicine). This means the result will include sales that might not have any medicines associated with them, reflecting NULL in the MedicineID and Quantity columns for such cases.

But, an INNER JOIN between these tables would only include records where there is a match between the SalesRecord and SalesMedicine tables, i.e., only sales that have at least one associated medicine. Sales records without any associated medicine details would not appear in the results.

4. Write a single-row subquery. Show the results and sort the results by key field(s). Interpret the output.

**SQL Query with Single-Row Subquery:**

```
SELECT
    SalesID,
    CustFirstName,
    CustLastName,
    SalesDate
FROM
    Pharmacy.SalesRecord
WHERE
    SalesDate = (SELECT MIN(SalesDate) FROM Pharmacy.SalesRecord)
ORDER BY CAST(SalesID AS UNSIGNED);
```

```
1      -- 4
2      -- SQL Query with Single-Row Subquery
3 •    SELECT
4          SalesID,
5          CustFirstName,
6          CustLastName,
7          SalesDate
8      FROM
9          Pharmacy.SalesRecord
10     WHERE
11         SalesDate = (SELECT MIN(SalesDate) FROM Pharmacy.SalesRecord)
12     ORDER BY CAST(SalesID AS UNSIGNED);
```

Result:

	SalesID	CustFirstName	CustLastName	SalesDate
►	6	Sohail	Khan	2022-01-25



Suppose we want to find all the sales records that occurred on the same date as the earliest recorded sale in the database. This will help us understand the initial sales performance on the first day that data was recorded.

This kind of query is helpful for historical analyses, where understanding the launch or first day of operation is critical. For example, that only one transaction happened on the first day.

5. Write a multiple-row subquery. Show the results and sort the results by key field(s). Interpret the output.

**SQL Query with Multiple-Row Subquery:**

Identify all sales records that involve medicines prescribed to multiple customers.

```
SELECT
    sr.SalesID,
    sr.CustFirstName,
    sr.CustLastName,
    sr.SalesDate,
    sm.MedicineID,
    sm.Quantity
FROM
    Pharmacy.SalesMedicine sm
JOIN
    Pharmacy.SalesRecord sr ON sm.SalesID = sr.SalesID
WHERE
    sm.MedicineID IN (SELECT MedicineID
                      FROM Pharmacy.CustMedPrescription
                      GROUP BY MedicineID
                      HAVING COUNT(DISTINCT CustomerID) > 1)
ORDER BY sm.MedicineID, CAST(sr.SalesID AS UNSIGNED);
```

```

17 • SELECT
18     sr.SalesID,
19     sr.CustFirstName,
20     sr.CustLastName,
21     sr.SalesDate,
22     sm.MedicineID,
23     sm.Quantity
24 FROM
25     Pharmacy.SalesMedicine sm
26 JOIN
27     Pharmacy.SalesRecord sr ON sm.SalesID = sr.SalesID
28 WHERE
29     sm.MedicineID IN (SELECT MedicineID
30                       FROM Pharmacy.CustMedPrescription
31                       GROUP BY MedicineID
32                       HAVING COUNT(DISTINCT CustomerID) > 1)
33 ORDER BY sm.MedicineID, CAST(sr.SalesID AS UNSIGNED);

```

Result:

Result Grid		Filter Rows:		Export:		Wrap Cell Content:	
	SalesID	CustFirstName	CustLastName	SalesDate	MedicineID	Quantity	
▶	2	Jackson	Wang	2024-01-12	Q2	2	
	3	Peter	Johnson	2022-07-10	Q2	5	
	9	Jackson	Wang	2022-12-11	Q2	3	
	17	Jackson	Wang	2023-01-25	Q2	2	
	1	Ashley	Brown	2023-03-07	Q3	2	
	10	Sohail	Khan	2023-09-26	Q3	2	
	12	Yoonki	Min	2024-01-01	Q3	3	
	16	Ashley	Brown	2023-08-13	Q3	1	
	9	Jackson	Wang	2022-12-11	Q4	1	
	13	Ashley	Brown	2022-06-13	Q4	1	
	15	Rohit	Verma	2022-07-13	Q4	1	
	16	Ashley	Brown	2023-08-13	Q4	1	
	1	Ashley	Brown	2023-03-07	Q5	4	
	7	John	NULL	2023-04-13	Q5	5	
	8	Rohit	Verma	2024-02-20	Q5	2	
	15	Rohit	Verma	2022-07-13	Q5	1	
	16	Ashley	Brown	2023-08-13	Q5	1	
	5	Yoonki	Min	2023-05-17	Q6	1	
	12	Yoonki	Min	2024-01-01	Q6	1	
	14	Peter	Johnson	2024-02-22	Q6	5	
	6	Sohail	Khan	2022-01-25	Q8	1	
	11	Namjun	NULL	2023-09-26	Q8	1	

This SQL statement would effectively display all relevant sales for medicines that are frequently prescribed, organized by medicine and transaction order.

6. Write an SQL to aggregate the results by using multiple columns in the SELECT clause. Interpret the output.

**SQL Query with Aggregation:**

```
SELECT
    m.MedName,
    YEAR(sr.SalesDate) AS SalesYear,
    SUM(sm.Quantity) AS TotalQuantitySold
FROM
    Pharmacy.SalesMedicine sm
JOIN
    Pharmacy.SalesRecord sr ON sm.SalesID = sr.SalesID
JOIN
    Pharmacy.Medicine m ON sm.MedicineID = m.MedicineID
GROUP BY
    m.MedName, YEAR(sr.SalesDate)
ORDER BY m.MedName, SalesYear;
```

```
35      -- 6
36      -- SQL Query with Aggregation
37 •    SELECT
38          m.MedName,
39          YEAR(sr.SalesDate) AS SalesYear,
40          SUM(sm.Quantity) AS TotalQuantitySold
41      FROM
42          Pharmacy.SalesMedicine sm
43      JOIN
44          Pharmacy.SalesRecord sr ON sm.SalesID = sr.SalesID
45      JOIN
46          Pharmacy.Medicine m ON sm.MedicineID = m.MedicineID
47      GROUP BY
48          m.MedName, YEAR(sr.SalesDate)
49      ORDER BY m.MedName, SalesYear;
```

Result:

	MedName	SalesYear	TotalQuantitySold
►	Advil	2022	8
	Advil	2023	2
	Advil	2024	2
	Arthrotec	2023	1
	Arthrotec	2024	6
	Auvi-Q	2022	3
	Banophen	2023	5
	Banophen	2024	3
	Benadryl	2022	1
	Benadryl	2023	1
	Motrin	2022	1
	Motrin	2023	10
	Motrin	2024	2
	Sudafed PE	2022	3
	Sudafed PE	2023	1

This query provides a clear picture of the total annual sales volume for each medicine, which is useful for assessing yearly performance and planning for future stock needs based on past demand. Understanding sales trends on an annual basis can help in making decisions about which medicines to prioritize or promote, negotiations with suppliers, and potentially discontinuing low-performing products.

7. Write a subquery using the NOT IN operator. Show the results and sort the results by key field(s). Interpret the output.

#### **SQL Query Using NOT IN:**

SELECT

m.MedicineID,  
m.MedName,  
m.ExpiryDate,  
m.PurchasePrice,  
m.SellingPrice,  
m.QuantityAvailable

FROM

Pharmacy.Medicine m

WHERE

m.MedicineID NOT IN (SELECT DISTINCT sm.MedicineID  
FROM Pharmacy.SalesMedicine sm)

ORDER BY m.MedicineID;

```

1      -- 7
2      -- SQL Query Using NOT IN
3      •  SELECT
4          m.MedicineID,
5          m.MedName,
6          m.ExpiryDate,
7          m.PurchasePrice,
8          m.SellingPrice,
9          m.QuantityAvailable
10     FROM
11         Pharmacy.Medicine m
12     WHERE
13         m.MedicineID NOT IN (SELECT DISTINCT sm.MedicineID
14                               FROM Pharmacy.SalesMedicine sm)
15     ORDER BY m.MedicineID;

```

Result:

	MedicineID	MedName	ExpiryDate	PurchasePrice	SellingPrice	QuantityAvailable
►	Q7	Primatene Mist	2025-08-15	40.00	42.00	13

Selected only those medicines from the Medicine table whose IDs do not appear in the list obtained from the SalesMedicine table.

The output lists all medicines that have never been sold according to the sales records. This can be particularly useful for inventory management, to identify products that may need promotional efforts or need to be discontinued. This insight can help in managing stock levels more effectively, avoiding overstocking items that do not sell.

8. Write a query using a CASE statement. Show the results and sort the results by key field(s). Interpret the output.

### **SQL Query with CASE Statement:**

```

SELECT
    MedicineID,
    MedName,
    ExpiryDate,
    PurchasePrice,
    SellingPrice,
    QuantityAvailable,
CASE
    WHEN SellingPrice < 10 THEN 'Low Price'

```

```

        WHEN SellingPrice BETWEEN 10 AND 20 THEN 'Moderate Price'

        WHEN SellingPrice > 20 THEN 'High Price'

        ELSE 'Undefined'

    END AS PriceTier

FROM

    Pharmacy.Medicine

ORDER BY MedicineID;

```

```

17      -- 8
18      -- SQL Query with CASE Statement
19  •    SELECT
20          MedicineID,
21          MedName,
22          ExpiryDate,
23          PurchasePrice,
24          SellingPrice,
25          QuantityAvailable,
26          CASE
27              WHEN SellingPrice < 10 THEN 'Low Price'
28              WHEN SellingPrice BETWEEN 10 AND 20 THEN 'Moderate Price'
29              WHEN SellingPrice > 20 THEN 'High Price'
30              ELSE 'Undefined'
31          END AS PriceTier
32  FROM
33      Pharmacy.Medicine
34  ORDER BY MedicineID;

```

Result:

	MedicineID	MedName	ExpiryDate	PurchasePrice	SellingPrice	QuantityAvailable	PriceTier
►	Q1	Auvi-Q	2025-03-07	30.00	35.00	13	High Price
	Q2	Advil	2025-05-17	13.00	15.00	50	Moderate Price
	Q3	Banophen	2024-08-15	20.00	22.00	3	High Price
	Q4	Sudafed PE	2025-03-12	10.00	12.00	30	Moderate Price
	Q5	Motrin	2024-06-07	11.00	12.00	25	Moderate Price
	Q6	Arthrotec	2027-02-20	35.00	38.00	7	High Price
	Q7	Primatene Mist	2025-08-15	40.00	42.00	13	High Price
	Q8	Benadryl	2024-12-11	8.00	11.00	58	Moderate Price

CASE Statement categorizes medicines into different pricing tiers:

Medicines with a selling price below \$10 are labeled as 'Low Price'.

Medicines priced between \$10 and \$20 are considered 'Moderate Price'.

Medicines selling for more than \$20 are categorized as 'High Price'.

This query provides a clear overview of how medicines are distributed across different price tiers, which can help management understand the current pricing strategy. Understanding the distribution of medicine prices can also help in analyzing customer buying behaviors and preferences. If most sales occur in a particular price tier, it might indicate the price sensitivity or purchasing power of the customer base.

9. Write a query using the NOT EXISTS operator. Show the results and sort the results by key field(s). Interpret the output.

**SQL Query Using NOT EXISTS:**

Find all medicines from the Medicine table that have never been prescribed to any customer, as indicated by the absence of their records in the CustMedPrescription table.

```
SELECT
    m.MedicineID,
    m.MedName,
    m.ExpiryDate,
    m.PurchasePrice,
    m.SellingPrice,
    m.QuantityAvailable
FROM
    Pharmacy.Medicine m
WHERE NOT EXISTS (
    SELECT 1
    FROM Pharmacy.CustMedPrescription cmp
    WHERE cmp.MedicineID = m.MedicineID
)
ORDER BY m.MedicineID;
```



```

36      -- 9
37      -- SQL Query Using NOT EXISTS
38      • SELECT
39          m.MedicineID,
40          m.MedName,
41          m.ExpiryDate,
42          m.PurchasePrice,
43          m.SellingPrice,
44          m.QuantityAvailable
45      FROM
46          Pharmacy.Medicine m
47      WHERE NOT EXISTS (
48          SELECT 1
49          FROM Pharmacy.CustMedPrescription cmp
50          WHERE cmp.MedicineID = m.MedicineID
51      )
52      ORDER BY m.MedicineID;

```

Result:

	MedicineID	MedName	ExpiryDate	PurchasePrice	SellingPrice	QuantityAvailable
►	Q7	Primatene Mist	2025-08-15	40.00	42.00	13

NOT EXISTS Clause is used to check for the non-existence of any related entries in the CustMedPrescription table. Identifying medicines that have never been prescribed can help the pharmacy manage its inventory more efficiently by not stocking items with no demand.

10. Write a subquery using the NOT NULL operator in the inner query. Show the results and sort the results by key field(s). Interpret the output.

### **SQL Query with a NOT NULL Subquery:**

Retrieve all customers who have a last name recorded and who have made at least one purchase (i.e., their ID appears in the SalesRecord table).

SELECT

c.CustomerID,  
 c.FirstName,  
 c.LastName,  
 c.Telephone,  
 c.Address,  
 c.BirthDate

FROM

Pharmacy.Customer c

WHERE

```

c.LastName IS NOT NULL
AND CONCAT(c.FirstName, ' ', c.LastName) IN (
    SELECT DISTINCT CONCAT(sr.CustFirstName, ' ', sr.CustLastName)
    FROM Pharmacy.SalesRecord sr
)
ORDER BY c.LastName, c.CustomerID;

```

```

54      -- 10
55      -- SQL Query with a NOT NULL Subquery
56  •   SELECT
57          c.CustomerID,
58          c.FirstName,
59          c.LastName,
60          c.Telephone,
61          c.Address,
62          c.BirthDate
63  FROM
64      Pharmacy.Customer c
65  WHERE
66      c.LastName IS NOT NULL
67      AND CONCAT(c.FirstName, ' ', c.LastName) IN (
68          SELECT DISTINCT CONCAT(sr.CustFirstName, ' ', sr.CustLastName)
69          FROM Pharmacy.SalesRecord sr
70      )
71  ORDER BY c.LastName, c.CustomerID;
72

```

Result:

	CustomerID	FirstName	LastName	Telephone	Address	BirthDate
▶	5	Ashley	Brown	(230) 695-1123	204 U at Park	1987-09-05
	6	Peter	Johnson	(590) 655-1008	106 A Mid Town	2006-11-01
	9	Sohail	Khan	(492) 249-3950	12 Marine Drive	1976-02-12
	2	Yoonki	Min	(580) 599-8760	780 Avenue Town	2002-01-23
	10	Rohit	Verma	(478) 555-8822	15 Highland market	2001-12-31
	1	Jackson	Wang	(520) 599-5123	345 South Street	1967-03-02
	4	Aditi	Whagire	(990) 665-1657	486 Campus Crossing	1997-09-01

The query filters Customer entries where the last name is not null and the concatenated full name matches any of the concatenated names from the SalesRecord table.

## **Summary:**

The database is structured to efficiently manage a pharmacy's operations, encompassing various essential components such as customer information, sales transactions, medicine details, prescriptions, and supplier and manufacturer data. This comprehensive setup enables the pharmacy to track all aspects of its business operations, from inventory management to customer service.

## **Functionality and Real-Life Application:**

### **1. Customer and Sales Management:**

The database includes detailed records of customers (e.g., names, contact information, addresses, birth dates) and sales transactions. These records help in personalizing customer service, such as sending birthday greetings or tailored health tips, enhancing customer relationships. Furthermore, the ability to track every sale by customer allows the pharmacy to analyze purchasing behaviors, identify sales trends, and develop targeted marketing strategies. For example, if a customer regularly purchases medication for chronic conditions, the pharmacy can send reminders when it's time to refill prescriptions or provide information about relevant wellness products.

### **2. Inventory and Prescription Tracking:**

Medicines are central to the database, with detailed records on each medicine, including its supplier, manufacturer, prices, and stock levels. This setup aids in meticulous inventory management, ensuring that the pharmacy is well-stocked with high-demand medicines and can quickly respond to supplier issues or recalls. Additionally, the link between medicines and prescriptions (through the `CustMedPrescription` table) ensures that pharmacists can quickly verify prescriptions, check for potential drug interactions, and maintain accurate dispensing records. This is crucial for patient safety and compliance with health regulations.

### **3. Supplier and Manufacturer Coordination:**

By maintaining information on medicine suppliers and manufacturers, the pharmacy can manage its supply chain more effectively. This information helps in negotiating prices, ensuring timely deliveries, and maintaining quality standards. In cases of drug recalls or quality issues, having direct links to the manufacturers and suppliers allows for rapid response actions to protect patient health and comply with regulatory requirements.

## **Overall Impact:**

In practice, this database not only streamlines day-to-day operations but also supports strategic business decisions and enhances customer care in the pharmacy sector. By integrating various data points from sales to stock levels and customer demographics the pharmacy can optimize its operations, reduce costs, and improve service delivery.