Distinction Level Project

Datastore Implementation Design Report

Project-Pendro Pharma: Pharmacy Inventory & Sales Database

Prepared by

Abdul Mahi

Table of Contents

1. Introduction

1.1 Abstract

The purpose of this report is to outline the design, development, and implementation of a Pharmaceutical Inventory Database Management System in MySQL. The system is implemented by creating a database which contains data about the stored medicine in the inventory of the trading company, employees, medicine suppliers, payments, various customer's transaction with the trading company that owns the inventory etc. An Entity relationship diagram of the implemented database has also been shown which outlines how various entities of the system interact with each other. Additionally, the report also outlines various queries for common use cases such as "Accessing information about the medicines that are available in the inventory" or "Updating available stock quantity". MySQL Workbench 8.0 CE was used for implementing this database on a windows 10 operating system. This system can be used to facilitate smooth workflow of sale and purchase operations with minimal effort and high efficiency.

1.2 Background Information

Pendro Pharma is a pharmaceutical retail company that purchases drugs from suppliers and sells them to customers through its stores. To manage operations efficiently, the company requires a database to track inventory, including stock levels, batch details, and expiry dates. The system must also support employees in recording supplier purchase invoices and customer sales invoices, ensuring accurate tracking of drug movements, payments, and returns.

1.3 Main Uses and Objectives of the Datastore

The proposed design enables the database to support the following key functions:

- Maintain employee records, including joining year, address, and contact information.
- Manage customer and supplier details for accurate tracking of transactions.
- Store and update medicine information such as drug ID, name, and manufacturer.
- Monitor inventory levels, including batch number, quantity, manufacturing, and expiry dates.
- Generate and manage purchase and sales invoices.
- Record and track customer payments.
- Identify and flag expired drugs for disposal.
- Handle returned or damaged medicines efficiently.

1.4 Reason for choosing MySQL Database

MySQL was chosen because the project involves highly structured data that fits naturally into relational tables. SQL databases excel at enforcing data integrity, supporting transactions, and running complex joins—capabilities essential for managing inventory, invoices, and payments. In contrast, NoSQL solutions are better suited for unstructured or semi-structured data.

From a practical perspective, MySQL is also widely adopted in industry, open-source, cost-effective, and supported by strong community resources. Its ease of deployment and compatibility with tools like MySQL Workbench make it a reliable and maintainable choice for this project.

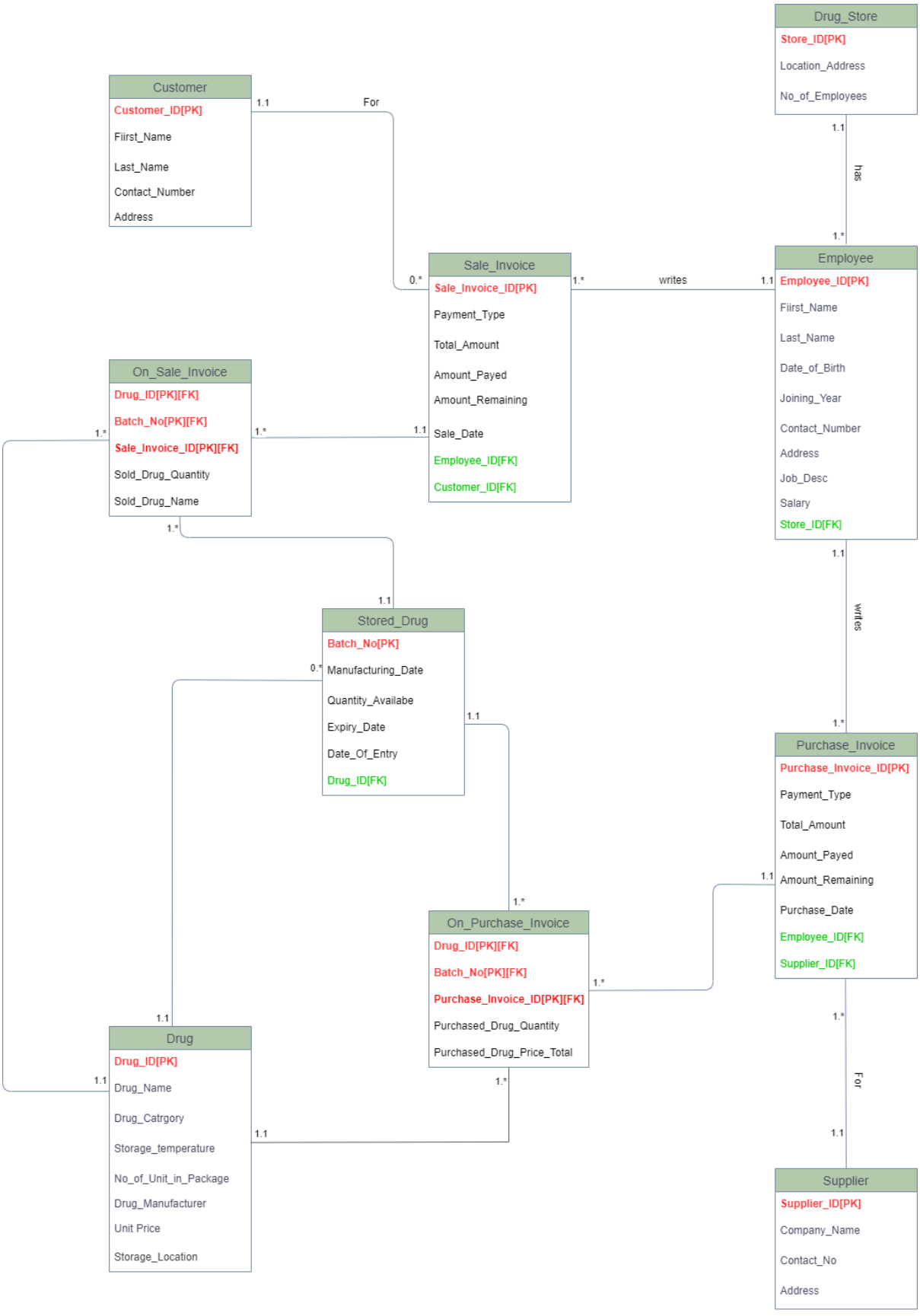
1.5 Normalized ERD design

The Entity–Relationship Diagram (ERD) for the Pendro Pharma inventory database models the core entities of the system, including employees, customers, suppliers, drugs, invoices, and stored drug batches. The relationships between these entities capture key business processes such as purchasing, selling, and stock management.

Each table has been normalized to Third Normal Form (3NF) to eliminate redundancy and ensure data integrity. This means:

- Each attribute contains only atomic values (1NF).
- All non-key attributes depend on the full primary key (2NF).
- No transitive dependencies exist between non-key attributes (3NF).

Normalization reduces data anomalies, improves query efficiency, and ensures consistency across the database. The ERD diagram below illustrates the final schema and the relationships between all major entities.


Drug_Store
Store_ID[PK]
Location_Address
No_of_Employees
Customer
Customer_ID[PK]
Fiirst_Name
Last_Name
Contact_Number
Address
1.1
For
1.1
has
1.*
0.*
Sale_Invoice
Sale_Invoice_ID[PK]
Payment_Type
Total_Amount
Amount_Payed
Amount_Remaining
Sale_Date
Employee_ID[FK]
Customer_ID[FK]
1.*
writes
1.1
Employee
Employee_ID[PK]
Fiirst_Name
Last_Name
Date_of_Birth
Joining_Year
Contact_Number
Address
Job_Desc
Salary
Store_ID[FK]
On_Sale_Invoice
Drug_ID[PK][FK]
Batch_No[PK][FK]
Sale_Invoice_ID[PK][FK]
Sold_Drug_Quantity
Sold_Drug_Name
1.*
1.*
1.1
1.*
1.1
Stored_Drug
Batch_No[PK]
Manufacturing_Date
Quantity_Availabe
Expiry_Date
Date_Of_Entry
Drug_ID[FK]
0.*
1.1
1.1
writes
1.*
Purchase_Invoice
Purchase_Invoice_ID[PK]
Payment_Type
Total_Amount
Amount_Payed
Amount_Remaining
Purchase_Date
Employee_ID[FK]
Supplier_ID[FK]
1.1
1.*
On_Purchase_Invoice
Drug_ID[PK][FK]
Batch_No[PK][FK]
Purchase_Invoice_ID[PK][FK]
Purchased_Drug_Quantity
Purchased_Drug_Price_Total
1.*
1.1
1.*
1.*
For
1.1
Drug
Drug_ID[PK]
Drug_Name
Drug_Catrgory
Storage_temperature
No_of_Unit_in_Package
Drug_Manufacturer
Unit Price
Storage_Location
1.1
1.1
Supplier
Supplier_ID[PK]
Company_Name
Contact_No
Address

1.6 Attribute Data Types

The tables shown below includes all the data types that has been used in the tables of the pendropharma database.

Drug_Store Table

Attribute	Data Type	Justification
Store_ID	INTEGER(4)	Each store is assumed to have a 4-digit unique number.
Location_Address	VARCHAR(100)	A store's address is expected to contain both numeric and string values and also the length can vary so it's datatype is varchar and character length is within 100 characters.
No_of_Employees	INTEGER(10)	Number of total employees is stored in numeric format

Primary Key: Store_ID,

Stored_ID can uniquely identify all the tuples.

Employee Table

Attribute	Data Type	Justification
Employee_ID	INTEGER(6)	Each Employee_ID is assumed to have a 6-digit unique number.
First_Name	VARCHAR(50)	Length of first name can vary so varchar type is used.
Last_Name	VARCHAR(50)	Length of Last name can vary so varchar type is used.
Date_of_Birth	Date	Date of birth is expected to be in a date format.
Joining_Year	YEAR	Cause all accepted year format is in YYYY or YY format.
Contact_Number	VARCHAR(20)	Contact numbers may contain special characters and can be of variable length and it is expected to be within 20 characters.
Address	VARCHAR(100)	An employee's address is expected to contain both numeric and string values so it's datatype is varchar and character length is within 100 characters.

Job_Desc	VARCHAR(100)	A Job description shouldn't logically cross 100 characters and it will vary in length so using a "varchar" will be more memory efficient than that of "char".
Salary	INTEGER(10)	Salary of employees is stored in numeric format and is in salary per year format.
Store_ID	INTEGER(4)	Each store is assumed to have a 4-digit unique number.

Primary Key: Employee_ID,

Employee_ID can uniquely identify all the tuples.

Foreign Key: Store_ID **References:** Drug_Store

Customer Table

Attribute	Data Type	Justification
Customer_ID	INTEGER (AUTO_INCREMENT)	Auto-incrementing unique ID Number.
First_Name	VARCHAR(50)	Length of first name can vary so varchar type is used.
Last_Name	VARCHAR(50)	Length of Last name can vary so varchar type is used.
Contact_Number	VARCHAR(20)	Contact numbers may contain special characters and can be of variable length and it is expected to be within 20 characters.
Address	VARCHAR(100)	An employee's address is expected to contain both numeric and string values so it's datatype is varchar and character length is within 100 characters.

Primary Key: Customer_ID,

An auto_incrementing surrogate key which acts as primary key and can uniquely identify all the tuples of customer table.

Supplier Table

Attribute	Data Type	Justification
Supplier_ID	INTEGER(6)	Each supplier is given an unique 6 digit Supplier_ID
Company_Name	VARCHAR(50)	A supplier's company name can vary and so varchar is preferred over char.
Contact_No	VARCHAR(20)	Contact numbers may contain special characters and can be of variable length and it is expected to be within 20 characters.
Address	VARCHAR(100)	A supplier's address is expected to contain both numeric and string values so it's datatype is varchar and character length is within 100 characters.

Primary Key: Supplier_ID,

Supplier_ID can uniquely identify all the tuples.

Drug Table

Attribute	Data Type	Justification
Drug_ID	INTEGER(10)	Each type of drug is assumed to have a 9-digit unique number, 10 is kept as length here so that overflow doesn't occur.
Drug_Name	VARCHAR(50)	Drug names will vary in length, so varchar is used to sure less memory usage.
Drug_Category	VARCHAR(50)	Drug category might contain both numeric and alphabetic values and its length may also vary.
Storage_temperature	DOUBLE(3,2)	We are taking upto 2 digits after decimal point and temperature won't be more than 3 digits.
No_of_Unit_in_Package	INTEGER(6)	Units in package is assumed to not cross 6 digits.
Drug_Manufacturer	VARCHAR(50)	Drug Manufacturer's names will vary in length, so varchar is used to sure less memory usage
Unit_Price	INTEGER(6)	Each unit price is assumed to not cross 6 digits in length.

Storage_Location	VARCHAR(50)	Location will contain both numeric and string values and will also vary in size.

Primary Key: Drug_ID,

Drug_ID can uniquely identify all the tuples.

Stored_Drug Table

Attribute	Data Type	Justification
Batch_No	Varchar(20)	Batch no is assumed to have both numeric and alphabetic values.
Manufacturing_Date	DATE	Only Date value is inserted here
Quantity_Avaiable	Interger(10)	Quantity available is assumed to be within 10-digit unique number.
Expiry_Date	DATE	Only Date value is inserted here
Date_Of_Entry	Date	The date on which entry happens in storage is a date type
Drug_ID	INTEGER(10)	Each type of drug is assumed to have a 10-digit unique number.

Primary Key: Batch_No,

Batch_No can uniquely identify all the tuples because different batches of drugs have different entry dates into storage and it also doesn't share any other common values with other entities of Drug_table.

Foreign Key: Drug_ID **References:** Drug

Sale_Invoice Table

Attribute	Data Type	Justification
Sale_Invoice_ID	INTEGER (5)	All Sale invoice ID is assumed to be of 5 digits
Payment_Type	Varchar (50)	Payment_type is assumed to have both numeric and alphabetic values.
Total_Amount	INTEGER (10)	It is assumed the amount won't exceed 10 digits length (All amount is in dollars)

Amount_Payed	INTEGER (10)	It is assumed the amount won't exceed 10 digits length
Amount_Remaining	INTEGER (10)	It is assumed the amount won't exceed 10 digits length
Sale_Date	DATE	Only Date value is inserted here
Employee_ID	INTEGER(6)	Each Employee_ID is assumed to have a 6-digit unique number.
Customer_ID	INTEGER (AUTO_INCREMENT)	Auto-incrementing unique ID Number.

Primary Key: Sale_Invoice_ID,

Uniquely identifies all other tuples of this table

Foreign Key 1: Employee_ID **References:** Employee

Foreign Key 2: Customer_ID **References:** Customer

On_Sale_Invoice Table

Attribute	Data Type	Justification
Drug_ID	INTEGER(10)	Each type of drug is assumed to have a 10-digit unique number.
Batch_No	Varchar(20)	Batch no is assumed to have both numeric and alphabetic values.
Sale_Invoice_ID	INTEGER(5)	All Sale invoice ID is assumed to be of 5 digits
Sold_Drug_Quantity	INTEGER(6)	Drug quantity is assumed to be numeric.
Sold_Drug_Name	Varchar(30)	Drug names vary in size

Primary Key: Batch_No, Drug_ID, Sale_Invoice_ID

Weak entity that uses the primary keys of other strong entities for unique identification of its tuples.

Foreign Key 1: Drug_ID **References:** Drug

Foreign Key 2: Batch_No **References:** Stored_Drug

Foreign Key 3: Sale_Invoice_ID **References:** Sale_Invoice

Purchase_Invoice Table

Attribute	Data Type	Justification
Purchase_Invoice_ID	INTEGER (5)	All Purchase invoice ID is assumed to be of 5 digits
Payment_Type	Varchar(50)	Payment_type is assumed to have both numeric and alphabetic values.
Total Amount	INTEGER (10)	It is assumed the amount won't exceed 10 digits length
Amount_Payed	INTEGER (10)	It is assumed the amount won't exceed 10 digits length
Amount_Remaining	INTEGER (10)	It is assumed the amount won't exceed 10 digits length
Purchase_Date	DATE	Only Date value is inserted here
Employee_ID	INTEGER(6)	Each Employee_ID is assumed to have a 6-digit unique number.
Supplier_ID	INTEGER(6)	Each supplier is given an unique 6 digit Supplier_ID

Primary Key: Purchase_Invoice_ID,

Uniquely identifies all other tuples of this table.

Foreign Key 1: Employee_ID **References:** Employee

Foreign Key 2: Supplier_ID **References:** Supplier

On_Purchase_Invoice Table

Attribute	Data Type	Justification
Drug_ID	INTEGER(10)	Each type of drug is assumed to have a 10-digit unique number.
Batch_No	Varchar(20)	Batch no is assumed to have both numeric and alphabetic values.
Purchase_Invoice_ID	INTEGER (5)	All Purchase invoice ID is assumed to be of 5 digits
Purchased_Drug_Quantity	INTEGER(6)	Drug quantity is assumed to be numeric.
Purchased_Drug_Total_Price	INTEGER(8)	Drug price is assumed to be no more than 8 digits in length

Primary Key: Batch_No, Drug_ID, Purchase_Invoice_ID

Weak entity that uses the primary keys of other strong entities for unique identification of its tuples.

Foreign Key 1: Drug_ID **References:** Drug

Foreign Key 2: Batch_No **References:** Stored_Drug

Foreign Key 3: Purchase_Invoice_ID **References:** Purchase_Invoice

2. Database Development

2.1 Scripts Necessary for Design

In MYSQL the datastore has been designed and then it is tested on a local sever utilizing MySQL workbench 8.0 CE in windows 10 OS.

The Scripts shown below involves the construction of proper references for each table and their relationship with each other. Furthermore, it will also include the datatypes of each of the attributes.

2.2 Constructing the Database

The following script constructs a database and its necessary tables using **SQL DDL statements** and it is also refined to better accept relational data.

```
create schema PendroPharma;
use PendroPharma;

Set AUTOCOMMIT = false;

create table Drug_Store(

    Store_ID int(100) not null,
    Location_Address varchar(100) not null,
    No_of_Employees int(10) not null,

    PRIMARY KEY (Store_ID)
);

create table Employee(

    Employee_ID int(6) not null,
    First_Name varchar(50) not null,
    Last_Name varchar(50) not null,
    Date_of_Birth Date not null,
    Joining_Year year not null,
    Contact_Number varchar(20) not null,
    Address varchar(100) not null,
    Job_Desc varchar(100) not null,
    Salary int(10) not null,
    Store_ID int(4) not null,

    PRIMARY KEY (Employee_ID),
    FOREIGN KEY (Store_ID) References Drug_Store(Store_ID)
);
```

```
set AUTOCOMMIT = false;
use pendropharma;

CREATE TABLE Customer (
    Customer_ID int NOT NULL AUTO_INCREMENT,
    First_Name varchar(50) NOT NULL,
    Last_Name varchar(50) NOT NULL,
    Contact_Number varchar(20) NOT NULL,
    Address varchar(100) NOT NULL,

    PRIMARY KEY (Customer_ID)
);

CREATE TABLE Supplier (
    Supplier_ID int(6) NOT NULL,
    Comapany_Name varchar(50) NOT NULL,
    Contact_Number varchar(20) NOT NULL,
    Address varchar(100) NOT NULL,

    PRIMARY KEY (Supplier_ID)
);

CREATE TABLE Drug (
    Drug_ID int(10) NOT NULL,
    Drug_Name varchar(50) NOT NULL,
    Drug_Category varchar(50) NOT NULL,
    Storage_Temp double(3,2) NOT NULL,
    No_of_Unit_in_Package int(6) NOT NULL,
   Storage_Location varchar(50) NOT NULL,

    PRIMARY KEY (Drug_ID)
);

commit;
```

Query 1

```
set AUTOCOMMIT = false;
use pendropharma;

CREATE TABLE Stored_Drug (
    Batch_No varchar(20) NOT NULL,
    Manufacturing_Date DATE NOT NULL,
    Quantity_Available int(10) NOT NULL,
    Expiry_Date DATE NOT NULL,
    Date_Of_Entry DATE NOT NULL,
    Drug_ID int(10) NOT NULL,


    PRIMARY KEY (Batch_No),
    FOREIGN KEY (Drug_ID) References Drug(Drug_ID)
);

CREATE TABLE Sale_Invoice (
    Sale_Invoice_ID int(5) NOT NULL,
    Payment_Type varchar(50) NOT NULL,
    Total_Amount int(10) NOT NULL,
    Amount_Payed int(10) NOT NULL,
    Amount_Remaining int(10) NOT NULL,
    Sale_Date Date NOT NULL,
    Employee_ID int(6) NOT NULL,
    Customer_ID int NOT NULL AUTO_INCREMENT,

    PRIMARY KEY (Sale_Invoice_ID),
    FOREIGN KEY (Employee_ID) References Employee(Employee_ID),
    FOREIGN KEY (Customer_ID) References Customer(Customer_ID)
);
commit;
```

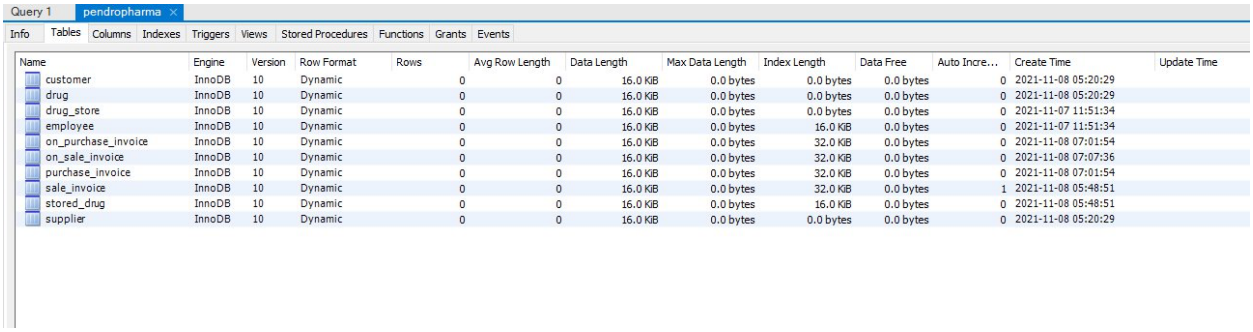
```sql
set AUTOCOMMIT = false;
use pendropharma;

CREATE TABLE Purchase_Invoice (
    Purchase_Invoice_ID int(5) NOT NULL,
    Payment_Type varchar(50) NOT NULL,
    Total_Amount int(10) NOT NULL,
    Amount_Payed int(10) NOT NULL,
    Amount_Remaining int(10) NOT NULL,
    Purchase_Date Date NOT NULL,
    Employee_ID int(6) NOT NULL,
    Supplier_ID int(6) NOT NULL,

    PRIMARY KEY (Purchase_Invoice_ID),
    FOREIGN KEY (Employee_ID) References Employee(Employee_ID),
    FOREIGN KEY (Supplier_ID) References Supplier(Supplier_ID)
);

CREATE TABLE On_Purchase_Invoice (
    Drug_ID int(10) NOT NULL,
    Batch_No varchar(20) NOT NULL,
    Purchase_Invoice_ID int(5) NOT NULL,
    Purchased_Drug_Qunatity int(6) NOT NULL,
    Purchased_Drug_Total_Price int(8) NOT NULL,

    PRIMARY KEY (Drug_ID, Batch_No, Purchase_Invoice_ID),
    FOREIGN KEY (Drug_ID) References Drug(Drug_ID),
    FOREIGN KEY (Batch_No) References Stored_Drug(Batch_No),
    FOREIGN KEY (Purchase_Invoice_ID) References Purchase_Invoice(Purchase_Invoice_ID)
);

commit;
```

Query 1

Limit to 1000 row

```sql
set AUTOCOMMIT = false;
use pendropharma;

CREATE TABLE On_Sale_Invoice (
    Drug_ID int(10) NOT NULL,
    Batch_No varchar(20) NOT NULL,
    Sale_Invoice_ID int(5) NOT NULL,
    Sold_Drug_Qunatity int(6) NOT NULL,
    Sold_Drug_Name varchar(30) NOT NULL,

    PRIMARY KEY (Drug_ID, Batch_No, Sale_Invoice_ID),
    FOREIGN KEY (Drug_ID) References Drug(Drug_ID),
    FOREIGN KEY (Batch_No) References Stored_Drug(Batch_No),
    FOREIGN KEY (Sale_Invoice_ID) References Sale_invoice(Sale_Invoice_ID)
);

commit;
```

These are the tables in pendropharma's inventory design database

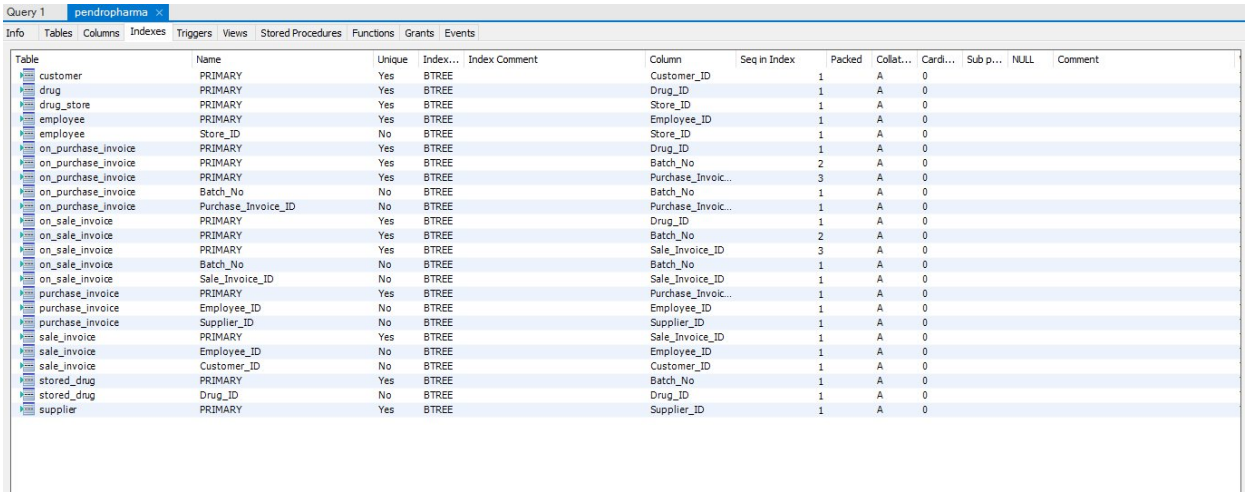


Query 1 | pendropharma

Info | Tables | Columns | Indexes | Triggers | Views | Stored Procedures | Functions | Grants | Events

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free	Auto Incre...	Create Time	Update Time
customer	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-11-08 05:20:29	
drug	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-11-08 05:20:29	
drug_store	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-11-07 11:51:34	
employee	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0	2021-11-07 11:51:34	
on_purchase_invoice	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	0	2021-11-08 07:01:54	
on_sale_invoice	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	0	2021-11-08 07:07:36	
purchase_invoice	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	0	2021-11-08 07:01:54	
sale_invoice	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	1	2021-11-08 05:48:51	
stored_drug	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0	2021-11-08 05:48:51	
supplier	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-11-08 05:20:29	

These are the list of indexes used



Query 1 | pendropharma

Info | Tables | Columns | Indexes | Triggers | Views | Stored Procedures | Functions | Grants | Events

Table	Name	Unique	Index...	Index Comment	Column	Seq in Index	Packed	Collat...	Cardi...	Sub p...	NULL	Comment
customer	PRIMARY	Yes	BTREE		Customer_ID	1		A	0			
drug	PRIMARY	Yes	BTREE		Drug_ID	1		A	0			
drug_store	PRIMARY	Yes	BTREE		Store_ID	1		A	0			
employee	PRIMARY	Yes	BTREE		Employee_ID	1		A	0			
employee	Store_ID	No	BTREE		Store_ID	1		A	0			
on_purchase_invoice	PRIMARY	Yes	BTREE		Drug_ID	1		A	0			
on_purchase_invoice	PRIMARY	Yes	BTREE		Batch_No	2		A	0			
on_purchase_invoice	PRIMARY	Yes	BTREE		Purchase_Invoic...	3		A	0			
on_purchase_invoice	Batch_No	No	BTREE		Batch_No	1		A	0			
on_purchase_invoice	Purchase_Invoice_ID	No	BTREE		Purchase_Invoic...	1		A	0			
on_sale_invoice	PRIMARY	Yes	BTREE		Drug_ID	1		A	0			
on_sale_invoice	PRIMARY	Yes	BTREE		Batch_No	2		A	0			
on_sale_invoice	PRIMARY	Yes	BTREE		Sale_Invoice_ID	3		A	0			
on_sale_invoice	Batch_No	No	BTREE		Batch_No	1		A	0			
on_sale_invoice	Sale_Invoice_ID	No	BTREE		Sale_Invoice_ID	1		A	0			
purchase_invoice	PRIMARY	Yes	BTREE		Purchase_Invoic...	1		A	0			
purchase_invoice	Employee_ID	No	BTREE		Employee_ID	1		A	0			
purchase_invoice	Supplier_ID	No	BTREE		Supplier_ID	1		A	0			
sale_invoice	PRIMARY	Yes	BTREE		Sale_Invoice_ID	1		A	0			
sale_invoice	Employee_ID	No	BTREE		Employee_ID	1		A	0			
sale_invoice	Customer_ID	No	BTREE		Customer_ID	1		A	0			
stored_drug	PRIMARY	Yes	BTREE		Batch_No	1		A	0			
stored_drug	Drug_ID	No	BTREE		Drug_ID	1		A	0			
supplier	PRIMARY	Yes	BTREE		Supplier_ID	1		A	0			

2.3 Inserting Relevant Data into the Database

Names of the drugs and other related necessary information for the drugs used in this database were collected from this website: https://www.beximcopharma.com/products . Many data insertion has been done in the table some notable ones to showcase the SQL statements used are:

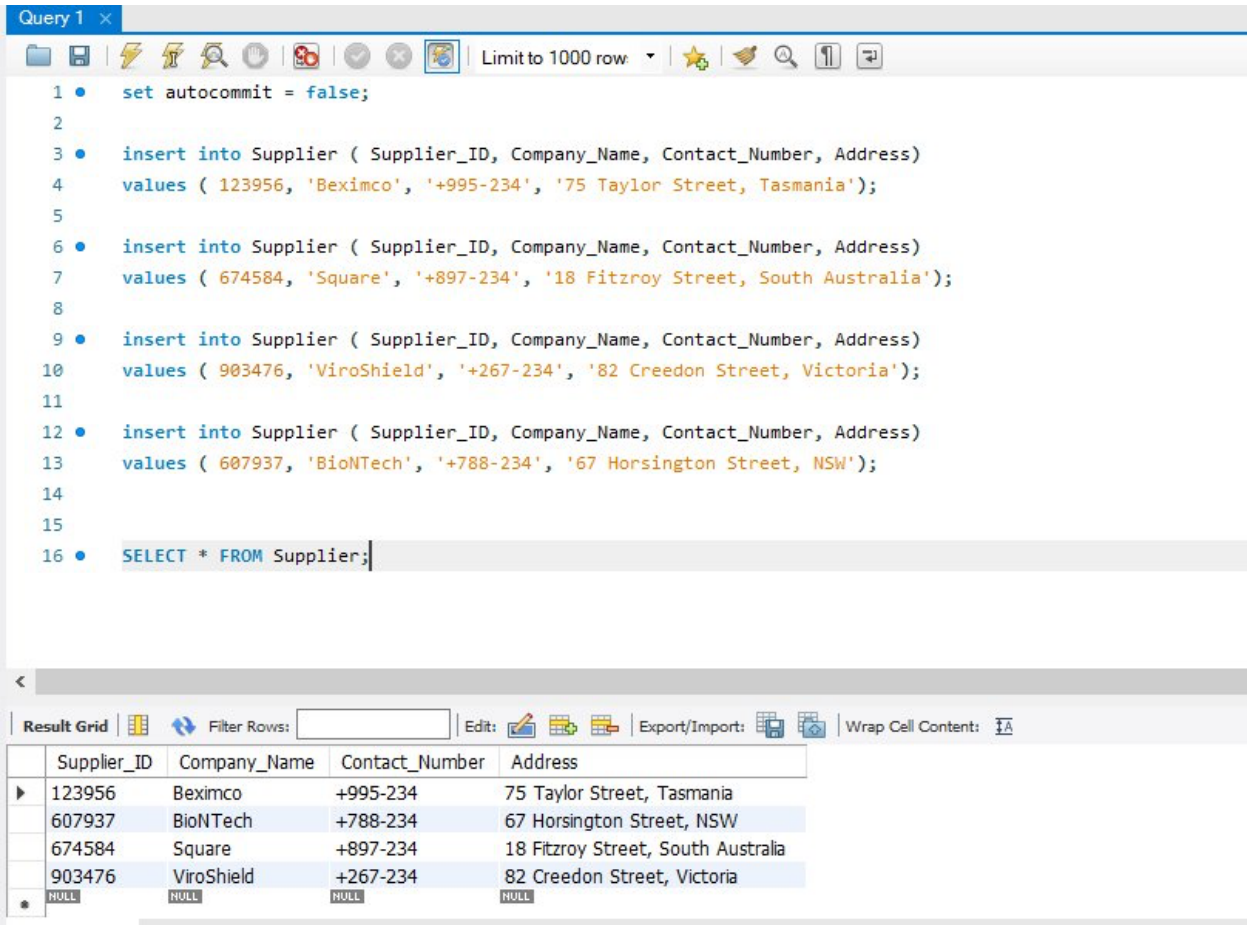
Drugs Table Insertion: These SQL statements were used for the insertion of values into the Drug table.

```
set autocommit = false;

INSERT INTO Drug( Drug_ID, Drug_Name, Drug_Category, Storage_Temp, No_of_Unit_in_Package, Storage_Location )
Values ( '123455678', 'Barri 4', 'Musculo Skeletal', 2.75, '1200', 'Warehouse-1, wartina street');

INSERT INTO Drug( Drug_ID, Drug_Name, Drug_Category, Storage_Temp, No_of_Unit_in_Package, Storage_Location )
Values ( '980765432', 'Apixa', 'Cardio Vascular', -5.00, '50', 'Warehouse-3, roger road');

INSERT INTO Drug( Drug_ID, Drug_Name, Drug_Category, Storage_Temp, No_of_Unit_in_Package, Storage_Location )
Values ( '453627189', 'Bexitrol F Maxhaler', 'Respiratory', -7.00, '80', 'Warehouse-4, roger road');

INSERT INTO Drug( Drug_ID, Drug_Name, Drug_Category, Storage_Temp, No_of_Unit_in_Package, Storage_Location )
Values ( '883627189', 'Odycin D', 'Eye Care', 3.7, '120', 'Warehouse-13, 20/21 Upton Ave');

INSERT INTO Drug( Drug_ID, Drug_Name, Drug_Category, Storage_Temp, No_of_Unit_in_Package, Storage_Location )
Values ( '536271669', 'Bemsivir', 'Anti Viral', 2.75, '500', 'Warehouse-1, wartina street');

INSERT INTO Drug( Drug_ID, Drug_Name, Drug_Category, Storage_Temp, No_of_Unit_in_Package, Storage_Location )
Values ( '772455678', 'Glyriva Bexicap', 'Respiratory', '-5.00', '120', 'Warehouse-4, roger road');

INSERT INTO Drug( Drug_ID, Drug_Name, Drug_Category, Storage_Temp, No_of_Unit_in_Package, Storage_Location )
Values ( '981455678', 'Billi', 'Respiratory', '-5.00', '120', 'Warehouse-4, roger road');

commit;
SELECT * FROM drug
```

Sale_Invoice Table Insertion: These SQL statements were used for the insertion of values into the Sale_Invoice Table.

```
set autocommit = false;

INSERT INTO Sale_Invoice( Sale_Invoice_ID, Payment_Type, Total_Amount, Amount_Payed, Amount_Remaining, Sale_Date, Employee_ID, Customer_ID)
Values ( 23567, 'Debit Card', 23, 30, (Amount_Payed - Total_Amount), '2021-10-28', 123456, 1);

INSERT INTO Sale_Invoice( Sale_Invoice_ID, Payment_Type, Total_Amount, Amount_Payed, Amount_Remaining, Sale_Date, Employee_ID, Customer_ID)
Values ( 12456, 'Cash', 13, 20, (Amount_Payed - Total_Amount), '2021-06-08', 405678, 2);

INSERT INTO Sale_Invoice( Sale_Invoice_ID, Payment_Type, Total_Amount, Amount_Payed, Amount_Remaining, Sale_Date, Employee_ID, Customer_ID)
Values ( 45678, 'Cash', 78, 100, (Amount_Payed - Total_Amount), '2021-10-18', 667889, 3);

INSERT INTO Sale_Invoice( Sale_Invoice_ID, Payment_Type, Total_Amount, Amount_Payed, Amount_Remaining, Sale_Date, Employee_ID, Customer_ID)
Values ( 78543, 'Debit Card', 334, 500, (Amount_Payed - Total_Amount), '2021-10-21', 123456, 4);

Select * FROM Sale_Invoice;

commit;
```

Supplier Table Insertion: These SQL statements were used for the insertion of values into the Supplier Table.



```
Query 1
Limit to 1000 row

1  set autocommit = false;
2
3  insert into Supplier ( Supplier_ID, Company_Name, Contact_Number, Address)
4  values ( 123956, 'Beximco', '+995-234', '75 Taylor Street, Tasmania');
5
6  insert into Supplier ( Supplier_ID, Company_Name, Contact_Number, Address)
7  values ( 674584, 'Square', '+897-234', '18 Fitzroy Street, South Australia');
8
9  insert into Supplier ( Supplier_ID, Company_Name, Contact_Number, Address)
10 values ( 903476, 'ViroShield', '+267-234', '82 Creedon Street, Victoria');
11
12 insert into Supplier ( Supplier_ID, Company_Name, Contact_Number, Address)
13 values ( 607937, 'BioNTech', '+788-234', '67 Horsington Street, NSW');
14
15
16 SELECT * FROM Supplier;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

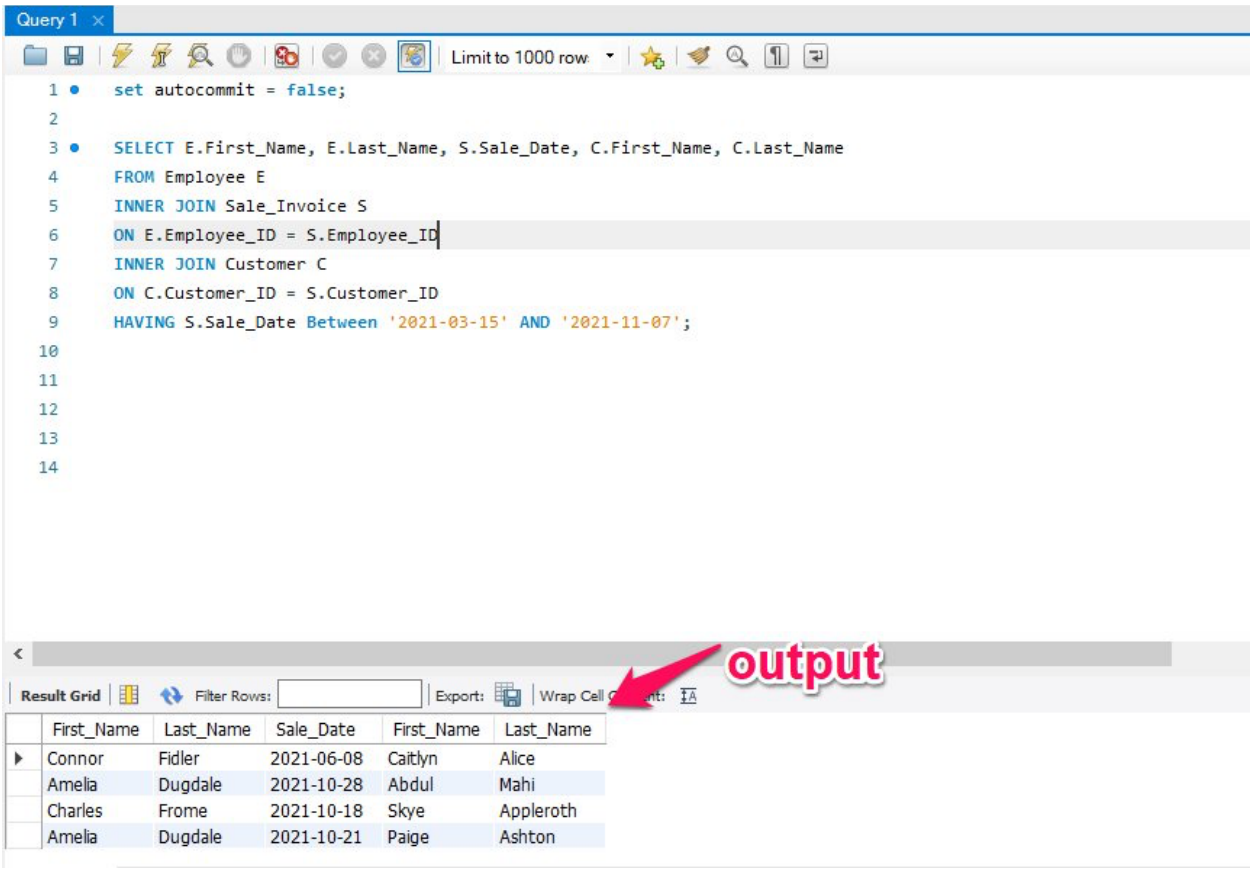
Supplier_ID	Company_Name	Contact_Number	Address
123956	Beximco	+995-234	75 Taylor Street, Tasmania
607937	BioNTech	+788-234	67 Horsington Street, NSW
674584	Square	+897-234	18 Fitzroy Street, South Australia
903476	ViroShield	+267-234	82 Creedon Street, Victoria
NULL	NULL	NULL	NULL

3. Statements and Queries for typical use cases

These are some of the main things that our database would allow a user to do:

(i) Identifying the names of certain employees and the names of the customers they have served on a certain day or between a certain period.

This can be implemented by using the following query:



(ii) The name of the stored drugs their details and their remaining quantity

```
Query 1
Limit to 1000 row
1  set autocommit = false;
2
3  SELECT D.Drug_Name, D.Drug_Category, D.Storage_Temp, D.Storage_Location, D.No_of_Unit_in_Package
4  FROM Drug D
5  INNER JOIN Stored_Drug S
6  ON D.Drug_ID = S.Drug_ID
7
8
9
10
11
12
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Drug_Name	Drug_Category	Storage_Temp	Storage_Location	No_of_Unit_in_Package
Barri 4	Musculo Skeletal	2.75	Warehouse-1, wartina street	1200
Barri 4	Musculo Skeletal	2.75	Warehouse-1, wartina street	1200
Barri 4	Musculo Skeletal	2.75	Warehouse-1, wartina street	1200
Bexitrol F Maxhaler	Respiratory	-7.00	Warehouse-4, roger road	80
Billi	Respiratory	-5.00	Warehouse-4, roger road	120

(iii) Figuring out whether there is enough drug of a certain type available or not after also considering their expiry date

```
Query 1
Limit to 1000 row
1  set autocommit = false;
2
3  SELECT D.Drug_Name, D.Drug_Category, D.Storage_Location, S.Expiry_Date, S.Quantity_Available
4  FROM Drug D
5  INNER JOIN Stored_Drug S
6  ON D.Drug_ID = S.Drug_ID
7  ORDER BY S.Quantity_Available ASC
8
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

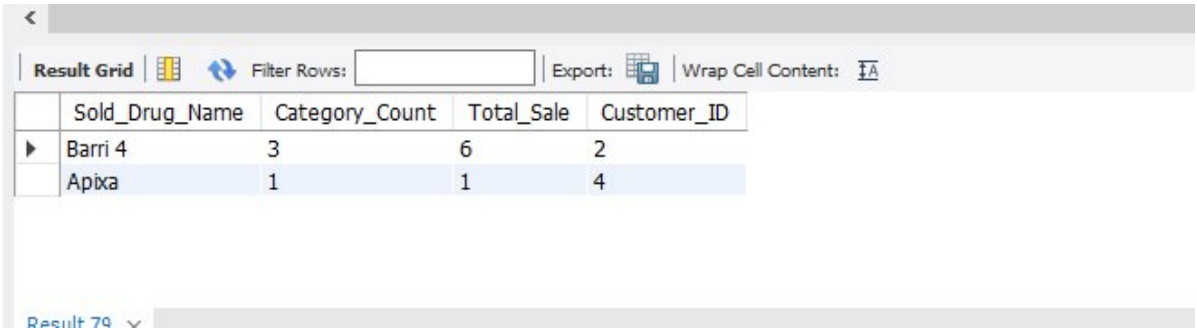
Drug_Name	Drug_Category	Storage_Location	Expiry_Date	Quantity_Available
Billi	Respiratory	Warehouse-4, roger road	2022-01-29	0
Barri 4	Musculo Skeletal	Warehouse-1, wartina street	2024-03-19	0
Bexitrol F Maxhaler	Respiratory	Warehouse-4, roger road	2022-07-09	500
Barri 4	Musculo Skeletal	Warehouse-1, wartina street	2024-09-09	24500
Barri 4	Musculo Skeletal	Warehouse-1, wartina street	2024-09-09	74500

output

(iv) Whether selling a certain type of drug is profitable or not

```
set autocommit = false;

SELECT F.Sold_Drug_Name, Count(D.Drug_Category) as Category_Count, (F.Sold_Drug_Qunatity * Count(D.Drug_Category)) as Total_Sale, S.Customer_ID
FROM Drug D
INNER JOIN On_Sale_Invoice F
ON D.Drug_ID = F.Drug_ID
INNER JOIN Sale_Invoice S
ON S.Sale_Invoice_ID = F.Sale_Invoice_ID
GROUP BY Drug_Category
```

Output:



Sold_Drug_Name	Category_Count	Total_Sale	Customer_ID
Barri 4	3	6	2
Apixa	1	1	4

(v) Profit Calculation

```
set autocommit = false;

SELECT P.Payment_Type as Purchase_Medium, (P.Total_Amount - P.Amount_Remaining) as Expenditure, S.Payment_Type as Sale_Medium, (S.Total_Amount - S.Amount_Remaining) as Profit
FROM Purchase_Invoice P
INNER JOIN Employee E
ON E.Employee_ID = P.Employee_ID
INNER JOIN Sale_Invoice S
ON S.Employee_ID = E.Employee_ID
HAVING Purchase_Medium = 'Check' AND Sale_Medium = 'Cash'
```

Output:

Purchase_Medium	Expenditure	Sale_Medium	Profit
Check	11668	Cash	56
Check	17300	Cash	6

4. Some Extra DML Statements used

(i) Spelling fixes with ALTER statement:

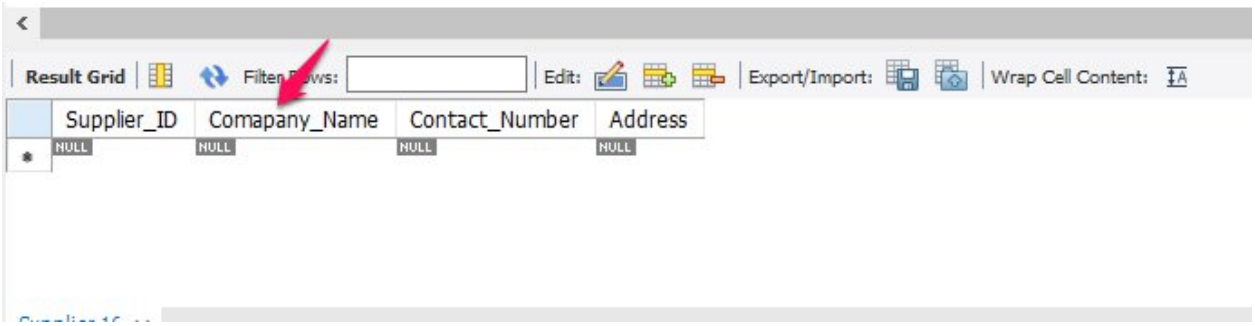


Fig 1: before update

Fig 2: After update

The table naming mistake of “comapany name” have been Altered to “Company Name” using DML statements.

(ii)Updating table data:

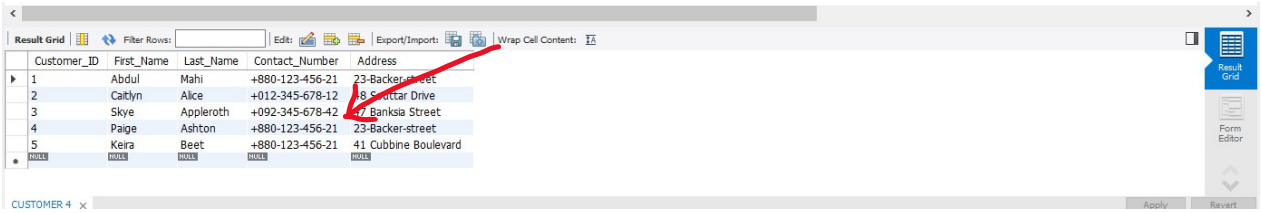


Fig 3: Before update

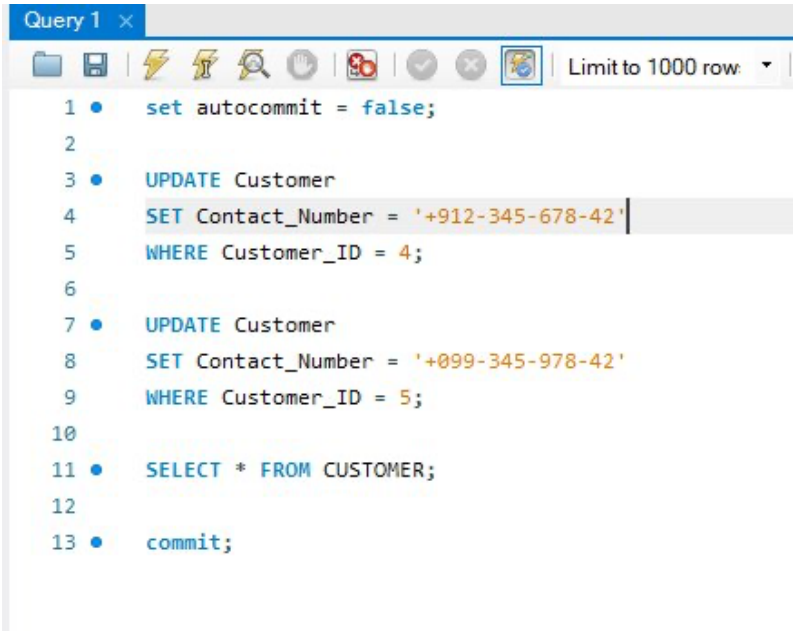


Fig 4: Executing SQL statement

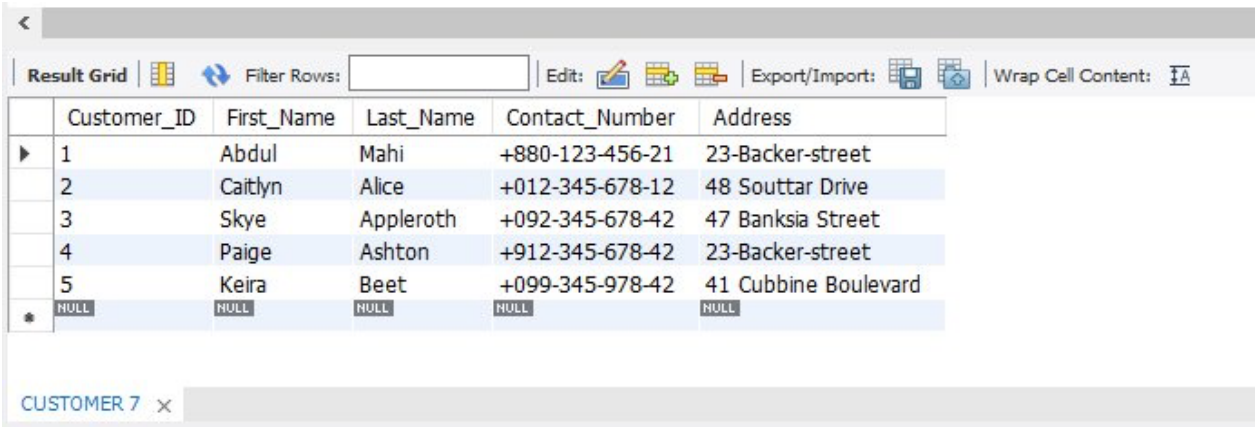


Fig 5: After update

5. References

SQL Data Types for MySQL, SQL Server, and MS Access 2021, W3schools.com, viewed 6 November 2021, <https://www.w3schools.com/sql/sql_datatypes.asp>.

Pharmacy Management System UML Diagram | FreeProjectz 2018, Freeprojectz.com, viewed 6 November 2021, <https://www.freeprojectz.com/uml-diagram/pharmacy-management-system-uml-diagram>.

Mikanwolfe 2021, *fundamentals-of-data-management/Design Report D.pdf at master · Mikanwolfe/fundamentals-of-data-management*, GitHub, viewed 6 November 2021, <https://github.com/Mikanwolfe/fundamentals-of-data-management/blob/master/d-task/Design%20Report%20D.pdf>.

What is Entity-Relationship Diagram | Pharmacy Illustrations | UML Class Diagram Example - Medical Shop | Entity Relationship Model Example For A Pharmacy 2021, https://www.conceptdraw.com, viewed 6 November 2021, <https://www.conceptdraw.com/examples/entity-relationship-model-example-for-a-pharmacy>.

Sheldon, R 2021, *How to choose between SQL and NoSQL databases*, Simple Talk, Simple Talk, viewed 7 November 2021, <https://www.red-gate.com/simple-talk/databases/nosql/how-to-choose-between-sql-and-nosql-databases/>.

Sama Samaan 2017, *Design and Implementation of a Pharmaceutical Inventory Database Management System*, ResearchGate, Al-Khwarizmi Engineering Journal, viewed 7 November 2021, <https://www.researchgate.net/publication/332348491_Design_and_Implementation_of_a_Pharmaceutical_Inventory_Database_Management_System>.