

# Database Management & Database Design

## Final Project Report

**Name:** Divya Taneja

**NUID:** 001494818

**Section:** 05

### **1. Topic**

Pharmacy Management System

### **2. Problem Statement**

The project aims to develop backend of the pharmacy retail chain for effective and efficient management of drugs, inventory, sale and customer base. It makes the life easier for a salesperson/pharmacist to dispense the correct drug efficiently by reducing the effort in manually order/invoice filing, keeping track of expiration dates of drugs and handling the re-order of drugs once those are nearing out of stock situation.

A pharmacy management system helps improving the patient care by providing a broad view about the patient's current condition, currently used drug details and allergies which can occurs due to usage of alternative drugs or different doses. The scope of the project is to design the backend database to keep track of all the activities aforementioned in a pharmacy retail chain using MySQL.

The Database created will store the pharmacy customer base with the proper history of every purchase. It will store the different attributes of customer like address, the order details, inventory detail, drug details etc.

The database will also have some other procedural routines to help out keeping track of total inventory and refill details of patient. To do all such operation we have the following in place

- Stored Procedures
- Triggers
- Access Control
- Views

Database Development Tool: MySQL Workbench

### 3. Tables Involved in the Project

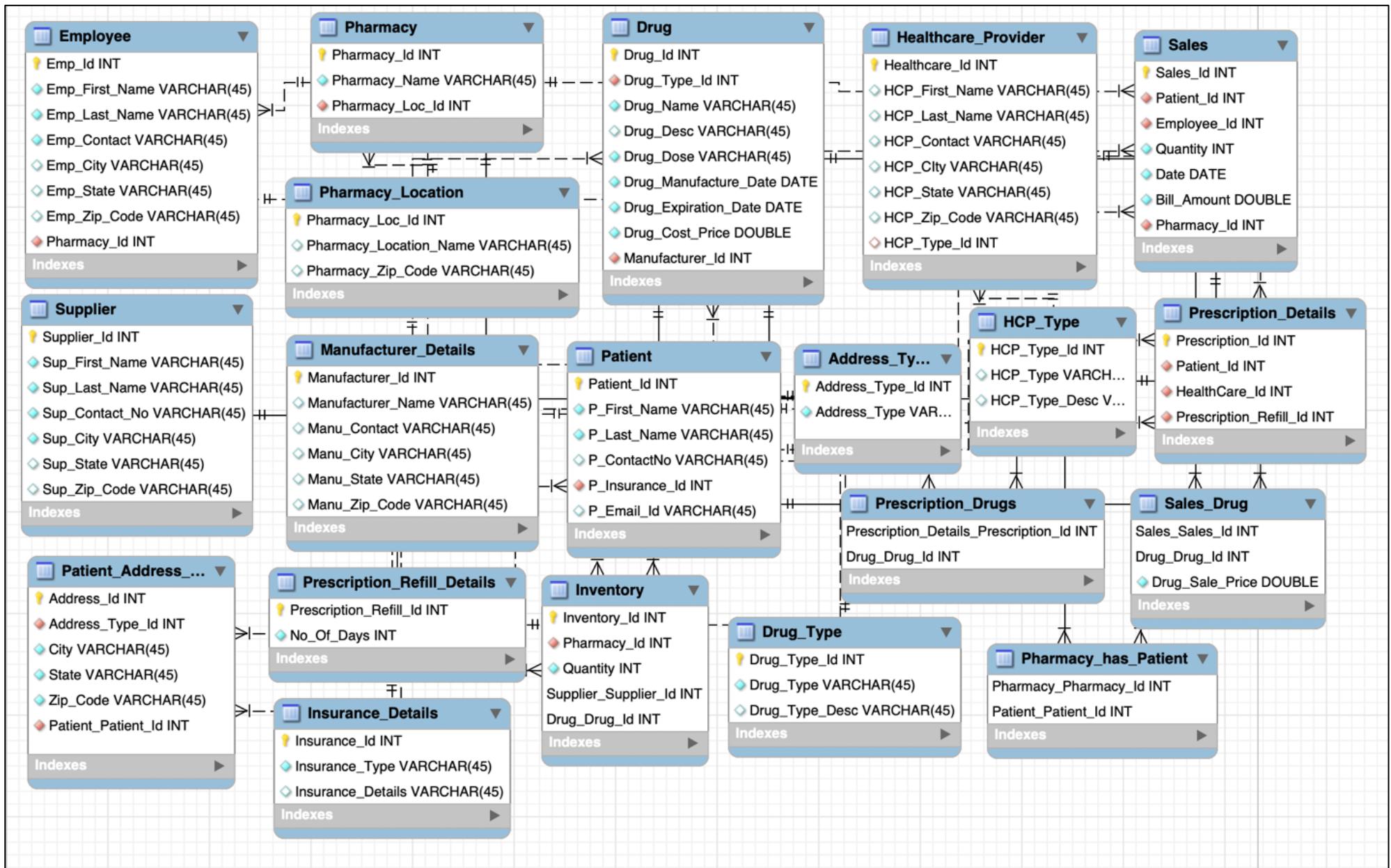
Below is the list of tables which have been created as a part of the complete Project

- Patient (Patient\_Id (PK), P\_First\_Name, P\_Last\_Name, P\_ContactNo, P\_Insurance\_Id, P\_Email\_Id)
- Patient\_Address\_Details (Address\_Id (PK) , Address\_Type\_Id(FK), City, State, Zip\_Code )
- Address\_Type (Address\_Type\_Id (PK) , Address\_Type\_Desc)
- Prescription (Prescription\_Id (PK) , Patient\_Id(FK) , Healthcare\_Provider\_Id(FK), Prescription\_Refill\_Id(FK))
- Prescription\_Refill\_Details (Prescription\_Refill\_Id (PK), No\_Of\_Days)
- Insurance\_Details (Insurance\_Id (PK), Insurance\_Type, Insurance\_Description)
- Pharmacy (Pharmacy\_Id(PK), Pharmacy\_Name, Pharmacy\_Loc\_Id(FK))
- Pharmacy\_Location( Pharmacy\_Loc\_Id(PK), Pharmacy\_Loc\_Name, Pharmacy\_Zip\_Code)
- Employee (Employee\_Id(PK), Employee\_Fname, Employee\_Lname, Employee\_Contact, Employee\_City, Employee\_State, Pharmacy\_Id(FK))
- Inventory (Inventory\_Id(PK), Pharmacy\_Id(FK), QuantitY, Drug\_Drug\_Id (FK), Supplier\_ID(FK))
- Supplier(Supplier\_ID(PK), Sup\_First\_Name, Sup\_Last\_Name, Sup\_Contact\_No, Sup\_City, Sup\_State, Sup\_Zip\_Code)
- Drug (Drug\_Id(PK), Drug\_Type\_Id(FK), Drug\_Name, Drug Desc, Drug Dose, Drug\_Man\_Date, Drug\_Exp\_Date, Drug\_Cost\_Price, Manufacturer\_Id(FK))
- Drug\_Type (Drug\_Type\_Id(PK), Drug\_Type, Drug\_Type\_Description)
- Manufacture\_Details (Manufacturer\_Id(PK), Manufacturer\_Name, Manufacturer\_Contact, Manufacturer\_City, Manufacturer\_State, Manufacturer\_Zip\_Code)
- Healthcare Provider (Healthcare\_provider\_id(PK), HCP\_TYPE\_ID(FK), HCP\_FName, HCP\_Lname, HCP\_Contact, HCP\_City, HCP\_State, HCP\_Zip\_code)
- HCP\_TYPE (HCP\_TYPE\_ID(PK), HCP\_TYPE, HCP\_TYPE\_desc)
- Prescription\_Drug(Prescription\_Id(PK), Drug\_Id(PK))
- Sales (Sales\_Id(PK), Patient\_Id(FK), Employee\_Id(FK), Sale\_Date, Bill\_Amount, Pharmacy\_Id(FK), Prescription\_Id(FK))
- Sales\_Drug (Sales\_Id(PK), Drug\_Id(PK), Drug\_Sale\_Price, Drug\_Quantity)
- Pharmacy\_Patient\_History(Pharmacy\_Id(PK), Patient\_Id(PK), Sale\_Id(PK), Sale\_Date)

#### **4. Relationships Details**

- A patient can have multiple drug prescriptions.
- A patient can purchase drug from more than 1 pharmacy.
- A pharmacy has multiple employees.
- A prescription can have more than one drug.
- A Supplier can supply more than one drug to Pharmacy.
- A prescription is meant to be for only one Patient.
- A manufacturer can produce more than one drugs.
- A drug can be there in multiple prescriptions
- A prescription has only one healthcare provider.

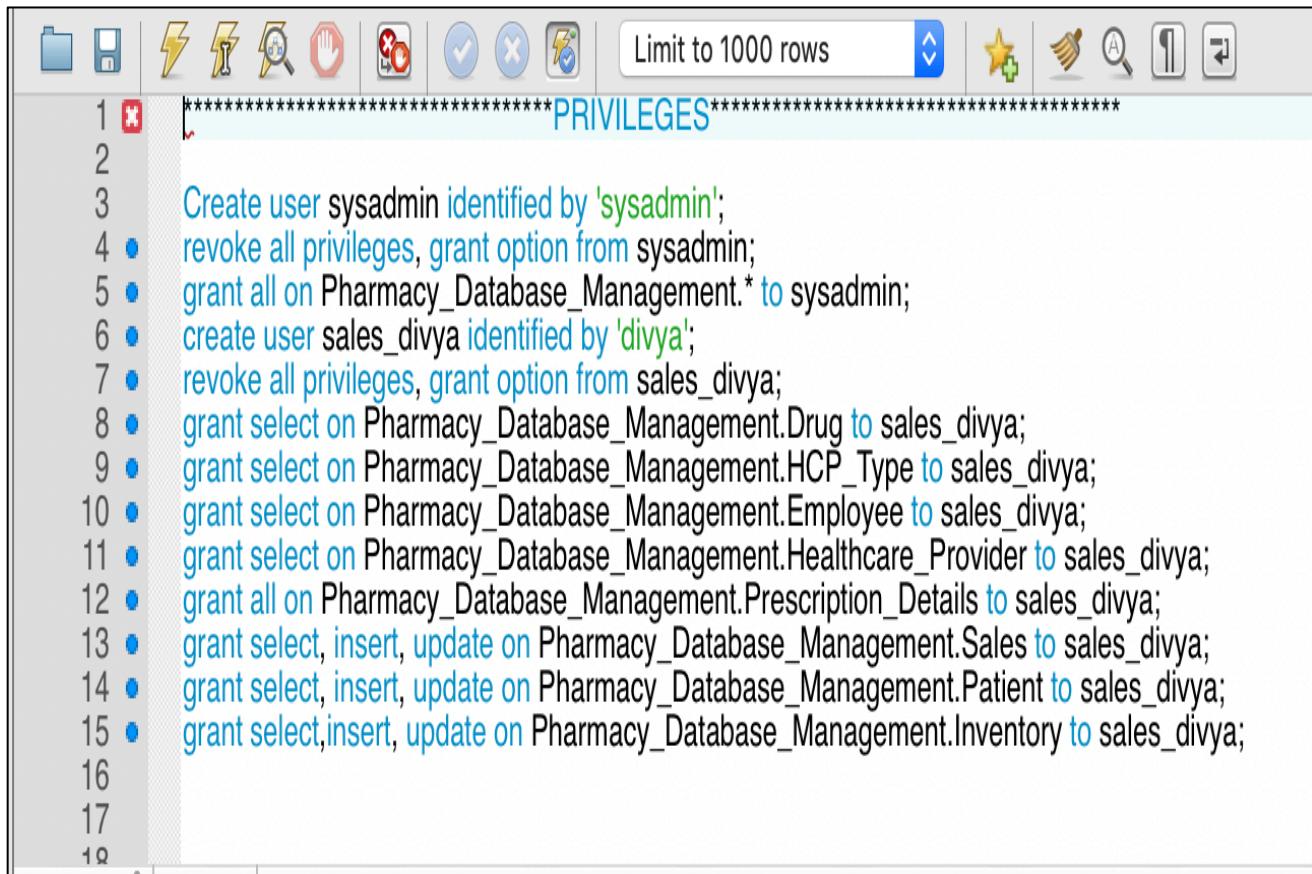
## 5. Entity Relationship Diagram



## 6. Privileges

1. A Sales person has limited access to the Pharmacy Database.
2. Sales person can select on Drug, Healthcare Provider details, Employee details tables.
3. A sales person has insert, update and select on inventory, sales , Prescription\_Details, and patient tables.
4. A salesperson can't access insurance details of a patient.
5. A systemadmin has access to the complete database;

PFB the details of the access provided to systemadmin and salesperson.



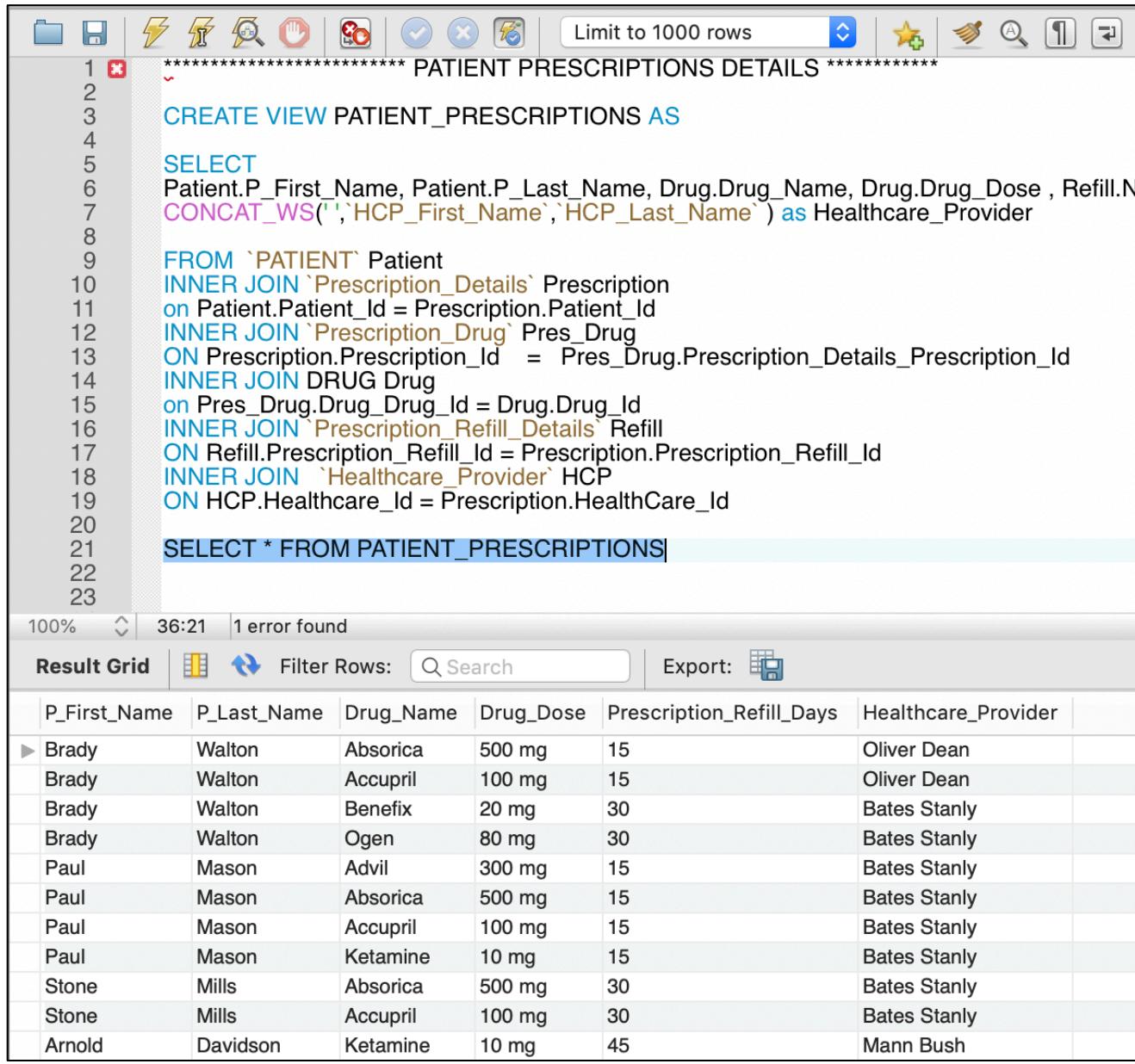
The screenshot shows a SQL query editor with a toolbar at the top containing icons for file, save, cut, copy, paste, find, search, and other database management functions. Below the toolbar is a table with 19 numbered rows. The table has two columns: a row number column and a code column. The code grants various privileges to the users 'sysadmin' and 'sales\_divya' on the 'Pharmacy\_Database\_Management' schema. The grants include SELECT, INSERT, UPDATE, and EXECUTE permissions on multiple tables and views within the schema.

	PRIVILEGES
1	Create user sysadmin identified by 'sysadmin';
2	revoke all privileges, grant option from sysadmin;
3	grant all on Pharmacy_Database_Management.* to sysadmin;
4	create user sales_divya identified by 'divya';
5	revoke all privileges, grant option from sales_divya;
6	grant select on Pharmacy_Database_Management.Drug to sales_divya;
7	grant select on Pharmacy_Database_Management.HCP_Type to sales_divya;
8	grant select on Pharmacy_Database_Management.Employee to sales_divya;
9	grant select on Pharmacy_Database_Management.Healthcare_Provider to sales_divya;
10	grant all on Pharmacy_Database_Management.Prescription_Details to sales_divya;
11	grant select, insert, update on Pharmacy_Database_Management.Sales to sales_divya;
12	grant select, insert, update on Pharmacy_Database_Management.Patient to sales_divya;
13	grant select,insert, update on Pharmacy_Database_Management.Inventory to sales_divya;
14	
15	
16	
17	
18	
19	

## 7. Views

- **Patient\_Prescriptions:** A view where all the drug prescription details can be seen for patient.

**SQL Query:** Select \* from Patient\_Prescriptions.



```
***** PATIENT PRESCRIPTIONS DETAILS *****  
CREATE VIEW PATIENT_PRESCRIPTIONS AS  
SELECT  
Patient.P_First_Name, Patient.P_Last_Name, Drug.Drug_Name, Drug.Drug_Dose , Refill.N  
CONCAT_WS(' ',HCP.First_Name,HCP.Last_Name) as Healthcare_Provider  
FROM `PATIENT` Patient  
INNER JOIN `Prescription_Details` Prescription  
on Patient.Patient_Id = Prescription.Patient_Id  
INNER JOIN `Prescription_Drug` Pres_Drug  
ON Prescription.Prescription_Id = Pres_Drug.Prescription_Details_Prescription_Id  
INNER JOIN DRUG Drug  
on Pres_Drug.Drug_Drug_Id = Drug.Drug_Id  
INNER JOIN `Prescription_Refill_Details` Refill  
ON Refill.Prescription_Refill_Id = Prescription.Prescription_Refill_Id  
INNER JOIN `Healthcare_Provider` HCP  
ON HCP.Healthcare_Id = Prescription.HealthCare_Id  
SELECT * FROM PATIENT_PRESCRIPTIONS
```

Result Grid | Filter Rows:  Search | Export:

P_First_Name	P_Last_Name	Drug_Name	Drug_Dose	Prescription_Refill_Days	Healthcare_Provider
Brady	Walton	Absorica	500 mg	15	Oliver Dean
Brady	Walton	Accupril	100 mg	15	Oliver Dean
Brady	Walton	Benefix	20 mg	30	Bates Stanly
Brady	Walton	Ogen	80 mg	30	Bates Stanly
Paul	Mason	Advil	300 mg	15	Bates Stanly
Paul	Mason	Absorica	500 mg	15	Bates Stanly
Paul	Mason	Accupril	100 mg	15	Bates Stanly
Paul	Mason	Ketamine	10 mg	15	Bates Stanly
Stone	Mills	Absorica	500 mg	30	Bates Stanly
Stone	Mills	Accupril	100 mg	30	Bates Stanly
Arnold	Davidson	Ketamine	10 mg	45	Mann Bush

- **Billing\_Invoice** : Billing information for patients with sales\_Id

**SQL :** SELECT \* FROM Billing\_Invoice;

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a query editor window displays the following SQL code:

```

1 **** Billing Information for Patient ****
2
3 CREATE VIEW Billing_Invoice AS
4
5     SELECT
6         SALES_ID,
7         CONCAT_WS(' ',Patient.P_First_Name, Patient.P_Last_Name) AS PATIENT,
8         CONCAT_WS(' ',EMPLOYEE.Emp_First_Name, EMPLOYEE.Emp_Last_Name) AS EMPLOYEE,
9         SALES.SALE_DATE AS PURCHASE_DATE, PHARMA.PHARMACY_NAME, SALES.BILL_AMOUNT
10
11     FROM `PATIENT` Patient
12     INNER JOIN SALES
13     ON SALES.Patient_id = PATIENT.Patient_Id
14     INNER JOIN EMPLOYEE
15     ON EMPLOYEE.Emp_Id = SALES.Employee_Id
16     INNER JOIN PHARMACY PHARMA
17     ON PHARMA.Pharmacy_Id = SALES.Pharmacy_Id
18
19     SELECT * FROM Billing_Invoice;
20

```

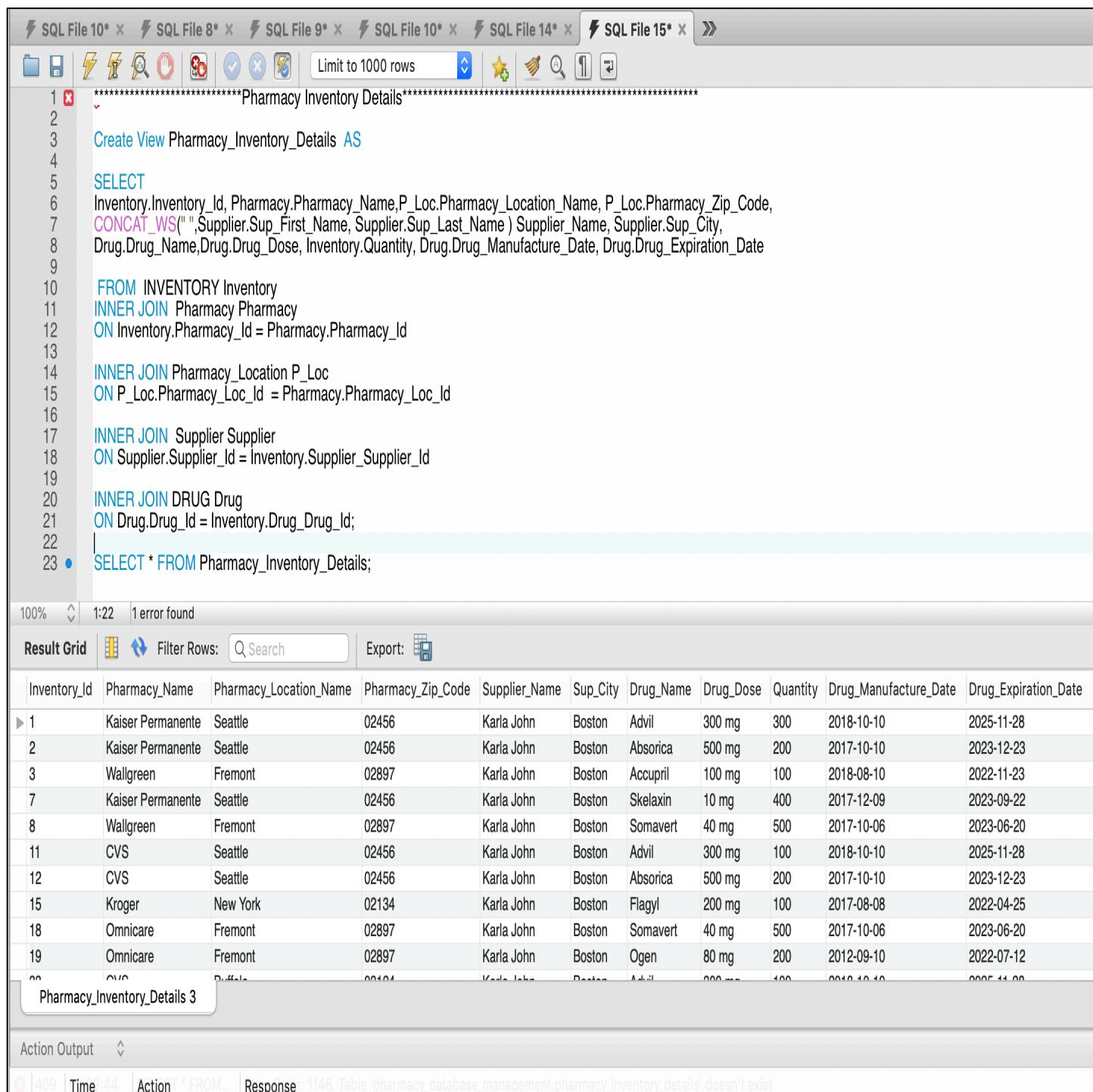
Below the code, the status bar shows "100%" and "31:20". A message "1 error found" is displayed. The "Result Grid" tab is selected, showing the results of the query:

	SALES_ID	PATIENT	EMPLOYEE	PURCHASE_DATE	PHARMACY_NAME	BILL_AMOUNT
▶ 1	Brady Walton	Jones Miller	2018-12-01	CVS	140	
2	Brady Walton	Gracia Wilson	2018-11-12	Kroger	414	
3	Brady Walton	Jackson Martin	2018-11-21	Omnicare	529	
4	Brady Walton	Jackson Martin	2016-11-12	Omnicare	500	
5	Paul Mason	Harris Clark	2018-09-12	Wallgreen	230	
6	Paul Mason	Green Baker	2018-10-12	Wallgreen	600	
7	Paul Mason	Parkar Turner	2018-11-12	Walmart	300	
8	Paul Mason	Parkar Turner	2016-12-01	Walmart	100	
22	Paul Mason	Parkar Turner	2018-07-09	Walmart	180	
9	Stone Mills	Cook Roger	2018-07-10	Kaiser Permanente	50	
10	Stone Mills	Peter Dinklage	2018-11-12	CVS	135	
23	Stone Mills	Ryan Patel	2017-12-12	Cosco Pharmacy	100	
11	Arnold David...	Salene Gomez	2018-12-12	Superfula	100	
12	Arnold David...	Fisher Barner	2017-12-12	Wallgreen	180	
24	Arnold David...	Fisher Barner	2018-11-09	Wallgreen	120	
13	Peter Hawkins	Alice Palmer	2018-12-03	Cosco Pharmacy	126	
14	Peter Hawkins	Alice Palmer	2017-11-01	Cosco Pharmacy	200	

The bottom left corner of the result grid shows the label "Billing\_Invoice 1".

- **Pharmacy\_Inventory\_Details:** View to show all the drugs and quantity available for a pharmacy.

**SQL Query:** SELECT \* FROM Pharmacy\_Inventory\_Details;



```

1 x ****Pharmacy Inventory Details*****
2
3 Create View Pharmacy_Inventory_Details AS
4
5 SELECT
6     Inventory.Inventory_Id, Pharmacy.Pharmacy_Name, P_Loc.Pharmacy_Location_Name, P_Loc.Pharmacy_Zip_Code,
7     CONCAT_WS(" ",Supplier.Sup_First_Name,Supplier.Sup_Last_Name) Supplier_Name, Supplier.Sup_City,
8     Drug.Drug_Name, Drug.Dose, Inventory.Quantity, Drug.Manufacture_Date, Drug.Expiration_Date
9
10    FROM INVENTORY Inventory
11    INNER JOIN Pharmacy Pharmacy
12      ON Inventory.Pharmacy_Id = Pharmacy.Pharmacy_Id
13
14    INNER JOIN Pharmacy_Location P_Loc
15      ON P_Loc.Pharmacy_Loc_Id = Pharmacy.Pharmacy_Loc_Id
16
17    INNER JOIN Supplier Supplier
18      ON Supplier.Supplier_Id = Inventory.Supplier_Supplier_Id
19
20    INNER JOIN DRUG Drug
21      ON Drug.Drug_Id = Inventory.Drug_Drug_Id;
22
23 • SELECT * FROM Pharmacy_Inventory_Details;

```

Result Grid | Filter Rows:  Search | Export:

Inventory_Id	Pharmacy_Name	Pharmacy_Location_Name	Pharmacy_Zip_Code	Supplier_Name	Sup_City	Drug_Name	Drug_Dose	Quantity	Drug_Manufacture_Date	Drug_Expiration_Date
1	Kaiser Permanente	Seattle	02456	Karla John	Boston	Advil	300 mg	300	2018-10-10	2025-11-28
2	Kaiser Permanente	Seattle	02456	Karla John	Boston	Absorica	500 mg	200	2017-10-10	2023-12-23
3	Wallgreen	Fremont	02897	Karla John	Boston	Accupril	100 mg	100	2018-08-10	2022-11-23
7	Kaiser Permanente	Seattle	02456	Karla John	Boston	Skelaxin	10 mg	400	2017-12-09	2023-09-22
8	Wallgreen	Fremont	02897	Karla John	Boston	Somavert	40 mg	500	2017-10-06	2023-06-20
11	CVS	Seattle	02456	Karla John	Boston	Advil	300 mg	100	2018-10-10	2025-11-28
12	CVS	Seattle	02456	Karla John	Boston	Absorica	500 mg	200	2017-10-10	2023-12-23
15	Kroger	New York	02134	Karla John	Boston	Flagyl	200 mg	100	2017-08-08	2022-04-25
18	Omnicare	Fremont	02897	Karla John	Boston	Somavert	40 mg	500	2017-10-06	2023-06-20
19	Omnicare	Fremont	02897	Karla John	Boston	Ogen	80 mg	200	2012-09-10	2022-07-12
20	CVS	Buffalo	02451	Karla John	Boston	Advil	300 mg	100	2018-10-10	2025-11-28

Action Output ▾

409 Time 44 Action \* FROM... Response: 1146. Table 'pharmacy\_database\_management.pharmacy\_inventory\_details' doesn't exist.

## 8. Procedures

- **Revenue\_of\_Year** : This procedure return the revenue for different pharmacy in year. We are passing year as the input parameter to this procedure.

The screenshot shows the SQL Developer interface with several tabs at the top labeled "SQL File 10\*", "SQL File 8\*", "SQL File 9\*", "SQL File 10\*", "SQL File 14\*", "SQL File 15\*", and "Limit to 1000 rows". Below the tabs is a toolbar with various icons. The main area contains the following SQL code:

```
1  **** Calculate the Revenue ****
2
3
4  delimiter &&
5  • create procedure Revenue_of_Year (in year_1 DATE)
6  Begin
7  □ SELECT Pharmacy_Name, (Selling - Purchase) AS Revenue FROM (
8  □ SELECT Pharmacy_Name, SUM(Cost_Price) Purchase, SUM(Selling_Price) Selling from (
9  □ SELECT Pharmacy_Name, Drug_Id, SUM(Drug_Quantity * Drug_Cost_Price) as Cost_Price,
10 | SUM(Drug_Quantity * Drug_Sale_Price) as Selling_Price
11 | from
12 | (
13 | | SELECT distinct Pharmacy.Pharmacy_Name, Pharmacy.Pharmacy_Id, s.sales_id,
14 | | d.Drug_Id, Drug_Sale_Price, d.Drug_Quantity, Drug_Cost_Price
15 | | FROM sales s
16 | | inner join Sales_Drug d on s.Sales_Id = d.Sales_Id
17 | | inner join drug on d.drug_id = drug.drug_id
18 | | Inner join Inventory Inv on Inv.Pharmacy_Id = s.Pharmacy_Id
19 | | Inner join Pharmacy ON Pharmacy.Pharmacy_Id = s.Pharmacy_Id
20 | | where year(s.Sale_Date) = YEAR(year_1)
21 | | )T1 group by Pharmacy_Name, Drug_Id
22 | | )T2 group by Pharmacy_Name
23 | | )T3;
24 | | END;
25 | &&
26
27 • CALL Revenue_of_Year("2016-12-12");
```

The code defines a stored procedure named "Revenue\_of\_Year" that takes a date parameter "year\_1". It calculates the revenue for each pharmacy by summing the difference between selling and purchase prices. The query uses multiple joins to link sales, drugs, and inventory tables. The result of the procedure call is displayed in a "Result Grid" table:

Pharmacy_Name	Revenue
Omnicare	265
Walmart	70
Wallgreen	200

The table has three rows with data: Omnicare (Revenue 265), Walmart (Revenue 70), and Wallgreen (Revenue 200). The bottom left corner of the interface shows the text "Result 4".

- **Refill\_Reminder** : Generate the patient report for whom the refill is due according to their prescription information.

The screenshot shows a MySQL Workbench interface with multiple tabs at the top labeled "SQL File 10\*", "SQL File 8\*", "SQL File 9\*", "SQL File 10\*", "SQL File 14\*", and "SQL File 15\*". The main area displays a stored procedure named "Refill\_Reminder". The code is as follows:

```

1  ****Drug Refill Reminder ****
2
3
4  delimiter &&
5  create procedure Refill_Reminder ()
6  Begin
7  SELECT * FROM (
8  SELECT pres.prescription_id,
9  concat_ws(" ",p.p_first_name,p.p_last_name) Patient, p.P_ContactNo as Patient_Contact,
10 p.P_Email_id as Patient_Email, latest_fill, Refill.No_Of_Days,
11 ADDDATE(latest_fill,No_of_Days) AS Refill_Due FROM (
12 SELECT prescription_id, max(sale_date) as latest_fill FROM SALES
13 group by prescription_id)
14 inner join prescription_details pres on pres.prescription_id = T. prescription_id
15 INNER JOIN `Prescription_Refill_Details` Refill
16 on pres.prescription_refill_id = Refill.Prescription_Refill_Id
17 inner join patient p on pres.patient_id = p.patient_id)T2 WHERE curdate() > Refill_Due;
18 END;
19 &&
20
21
22 • CALL Refill_Reminder();

```

The "Result Grid" tab is selected at the bottom, showing the following data:

prescription_id	Patient	Patient_Contact	Patient_Email	latest_fill	No_of_Days	Refill_Due
3	Paul Mason	5634253674	p.mason@gmail.com	2018-11-12	15	2018-11-27
4	Stone Mills	2378934256	stone.m@gmail.com	2018-11-12	30	2018-12-12
7	Ray Lynn	7878123415	ray.lynn@gmail.com	2018-10-09	15	2018-10-24
9	Jacob John	2343452345	jacob@gmail.com	2018-01-05	30	2018-02-04

The status bar at the bottom left indicates "Result 5".

## 9. Triggers

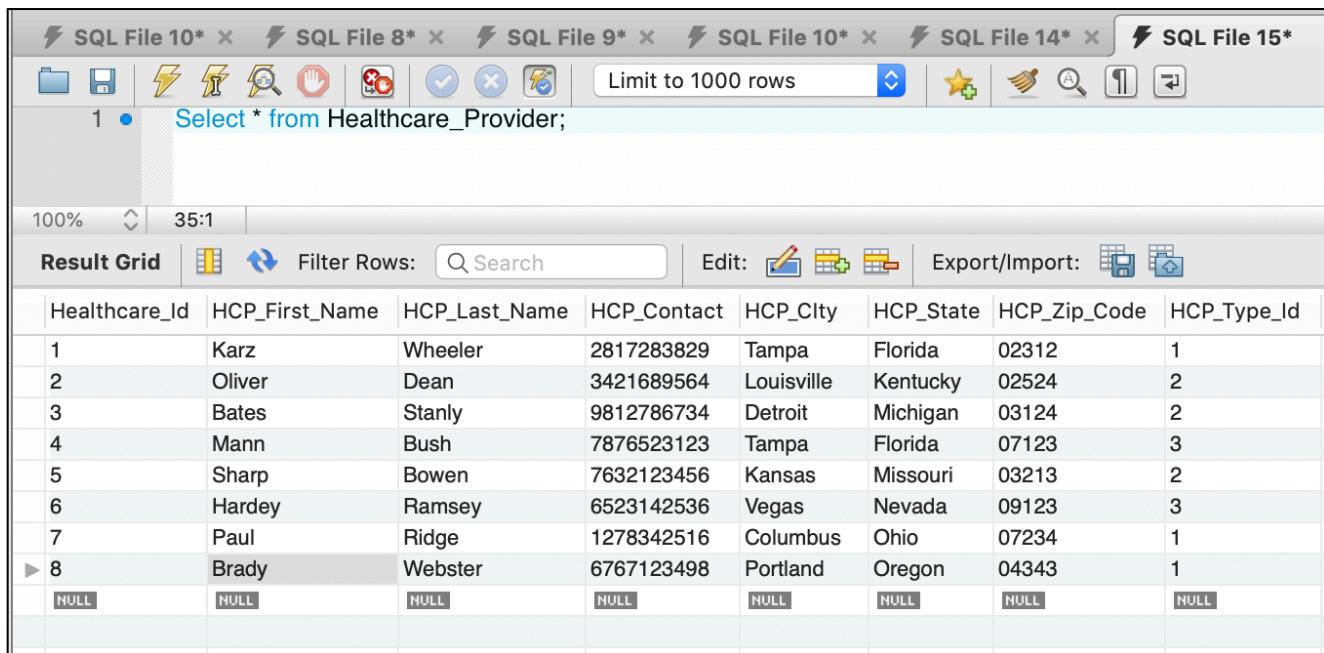
- **Calc\_Bill\_Amt** : Calculate the total bill amount for each purchase based on the quantity sold \* selling price of drug and update in table SALES for each sale.

- Before inserting into **Sales\_Drug** table, the Bill amount is not populated.

- After inserting into **Sales\_Drug** table, the Bill amount is populated due to trigger.

- **Doctor\_Screen:** Whenever a record is inserted into Prescription\_details table, the healthcare provider information is checked. If the Healthcare provider exist in the system, the prescription is added and drug is dispensed and if the Healthcare provider doesn't exist, an error message is thrown and record is discarded.

Valid Healthcare Provider List in System:



The screenshot shows a SQL Server Management Studio (SSMS) interface. At the top, there are tabs for various SQL files. Below the tabs is a toolbar with icons for file operations like New Query, Save, and Open, along with other tools like Find, Replace, and Refresh. A dropdown menu says "Limit to 1000 rows". To the right of the toolbar are several small icons for navigating between queries and managing sessions. The main area contains a query window with the following text:

```
1 • Select * from Healthcare_Provider;
```

Below the query window is a status bar showing "100%" and "35:1". The main pane displays the results of the query in a "Result Grid". The grid has the following columns:

Healthcare_Id	HCP_First_Name	HCP_Last_Name	HCP_Contact	HCP_City	HCP_State	HCP_Zip_Code	HCP_Type_Id
1	Karz	Wheeler	2817283829	Tampa	Florida	02312	1
2	Oliver	Dean	3421689564	Louisville	Kentucky	02524	2
3	Bates	Stanly	9812786734	Detroit	Michigan	03124	2
4	Mann	Bush	7876523123	Tampa	Florida	07123	3
5	Sharp	Bowen	7632123456	Kansas	Missouri	03213	2
6	Hardey	Ramsey	6523142536	Vegas	Nevada	09123	3
7	Paul	Ridge	1278342516	Columbus	Ohio	07234	1
► 8	Brady	Webster	6767123498	Portland	Oregon	04343	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Adding a prescription where the Healthcare provider is Invalid.

Healthcare\_Provider\_Id = 12

The screenshot shows the MySQL Workbench interface. In the main pane, a trigger named 'Doctor\_Screen' is defined:

```
1 ****TRIGGER FOR DOCTOR SCREENING ****
2
3 DELIMITER //
4 • CREATE TRIGGER Doctor_Screen
5   BEFORE INSERT ON Prescription_Details
6   FOR EACH ROW
7   BEGIN
8
9     IF NOT EXISTS
10    (SELECT * FROM `Healthcare_Provider` HCP
11     WHERE HCP.Healthcare_Id = NEW.HealthCare_Id )
12    THEN
13      SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'The Healthcare Provider is Invalid !';
14    end if;
15
16    END;
17  //
18
19 • INSERT INTO Prescription_Details (Patient_Id,HealthCare_Id,Prescription_Refill_Id)
20   VALUES (12,50,2);
```

In the 'Action Output' pane, a log entry is shown:

Action	Time	Response
INSERT INTO Prescri...	01:21:32	Error Code: 1643. The Healthcare Provider is Invalid !

Check if the prescription is added with invalid Healthcare Provider Id = 12.

The screenshot shows the MySQL Workbench interface. A query is run to select prescriptions from the 'Prescription\_Details' table where the 'HealthCare\_Id' is 12:

```
21
22 SELECT * FROM Prescription_Details WHERE HealthCare_Id = 12;
```

The results grid shows the following data:

Prescription_Id	Patient_Id	HealthCare_Id	Prescription_Refill_Id
NULL	NULL	NULL	NULL

In the 'Action Output' pane, two log entries are shown:

Action	Time	Response
INSERT INTO Prescri...	01:21:32	Error Code: 1643. The Healthcare Provider is Invalid !
SELECT * FROM Pre...	01:27:59	0 row(s) returned

- **DRUG\_Nearing\_Expiration** : Whenever a drug is added to the drug table, its expiration date is checked. If the drug expiration date is less than a year from current date, the drug details are not entered and returned to supplier.

Inserting a record into drug table where the expiration date of the drug is less a month from now.

```

1
2
3 *****DRUG EXPIRATION DATE < 1 YEAR *****
4
5 DELIMITER //
6 • CREATE TRIGGER DRUG_Nearing_Expiration
7 BEFORE INSERT ON DRUG
8 FOR EACH ROW
9 BEGIN
10
11 IF DATEDIFF(NOW(), NEW.`Drug_Expiration_Date`) < 365
12 THEN
13 SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'The Expiration Date Margin is too low';
14 end if;
15 END;
16
17
18
19 • INSERT INTO DRUG (Drug_Type_Id, Drug_Name, Drug_Desc, Drug_Dose,
20 Drug_Manufacture_Date, Drug_Expiration_Date, Drug_Cost_Price, Manufacturer_Id)
21 VALUES
22 (1,'ABC','ABC','300 MG','2018-09-01','2018-12-31',2,2);
23

```

Action Output

	Time	Action	Response
✗ 1	01:32:26	INSERT INTO DRUG...	Error Code: 1643. The Expiration Date Margin is too low

Check if the drug is added to the drug table.

```

23
24 SELECT * FROM DRUG where Drug_Name like '%ABC%';
25

```

Result Grid

Drug_Id	Drug_Type_Id	Drug_Name	Drug_Desc	Drug_Dose	Drug_Manufacture_Date	Drug_Expiration_Date	Drug_Cost_Price	Manufacturer_Id
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

DRUG 20

Action Output

	Time	Action	Response
✗ 1	01:32:26	INSERT INTO DRUG...	Error Code: 1643. The Expiration Date Margin is too low
✓ 2	01:34:52	SELECT * FROM DRUG...	0 row(s) returned

## 10. Data Analysis

**Most Selling Drug:** The most selling drug across all pharmacy based on the total quantity sold.

The screenshot shows a database query editor interface. At the top, there are various toolbar icons. Below the toolbar, the SQL query is displayed in a code editor with line numbers from 1 to 18. The query uses multiple joins and subqueries to calculate the total sold quantity for each drug and then select the top one. The result grid at the bottom shows a single row for 'Absorica'.

```
1
2 •  SELECT * FROM (
3   SELECT Drug_Name as Most_Selling_Drug , Drug_Desc, Drug_Dose, Drug_Type,
4   Drug_Type, Sold_Quantity, manu.Manufacturer_Name,
5   manu.manu_Contact as Manufacturer_Contact
6   from
7   (
8     SELECT Drug_id, SUM(drug_quantity) Sold_Quantity FROM Sales s
9     inner join sales_drug d
10    on s.sales_id = d.sales_id
11    GROUP BY drug_id
12  ) T
13  inner join Drug on Drug.Drug_Id = T.Drug_id
14  inner join Drug_Type on Drug.Drug_Type_Id = Drug_Type.Drug_Type_Id
15  inner join Manufacturer_Details manu on manu.Manufacturer_Id = Drug.Manufacturer_Id
16  )T1
17  ORDER BY Sold_Quantity DESC
18  LIMIT 1;
```

Result Grid

Most_Selling_Drug	Drug_Desc	Drug_Dose	Drug_Type	Sold_Quantity	Manufacturer_Name	Manufacturer_Contact
Absorica	Isotretion New Tab	500 mg	Generic	79	Trish Co	2315987123

**Most Successful Manufacturer:** Based on the total sale of its drugs.

The screenshot shows a SQL editor interface with multiple tabs at the top labeled "SQL File 10\*", "SQL File 8\*", "SQL File 9\*", "SQL File 10\*", "SQL File 14\*", and "SQL File 15\*". The main window displays a SQL query for finding the most successful manufacturer based on total sales. The code uses common table expressions (CTEs) T1 and T2 to calculate the total sales per manufacturer and drug combination, then groups by manufacturer and sorts by total sales in descending order, limiting the result to one row. The results are shown in a grid below, indicating that Pfizer is the most successful manufacturer with a total sale of 86400.

```
1  ****Most Successfull Manufacturer*****
2
3  Select manufacturer_name, SUM(base_sum) Total_Sale from (
4      Select manufacturer_name, drug_name, SUM(quantity * Drug_Cost_Price) base_sum from (
5          Select m.manufacturer_name, drug_name, quantity,Drug_Cost_Price FROM Manufacturer_Details m
6              inner join drug d
7                  on d.manufacturer_id = m.manufacturer_id
8              inner join inventory inv
9                  on inv.Drug_Drug_Id = d.drug_id
10             )T1
11             Group by manufacturer_name,drug_name
12         )T2
13             group by manufacturer_name
14             order by Total_Sale desc
15             Limit 1;
16
17
```

100% 4:10 1 error found

Result Grid | Filter Rows: Search Export: Fetch rows:

manufacturer_name	Total_Sale
Pfizer	86400