NAME: Chilukuri ,Nitin Chakravarthy
ADDRESS: 4-68-4, LB Colony
Flat-101, Vishnu's Rama Chaya Residency
Visakhapatnam, AP 530017
Social Security Number: ***-**-
Student Number: 11601099

UNIVERSITY OF NORTH TEXAS
DENTON, TEXAS 76203
OFFICE OF THE REGISTRAR

OFFICIAL ACADEMIC RECORD

Birthdate: 08/09/****

*** Graduate Transcript ***

2022 Fall

COURSE NUMBER	TITLE	EARN HRS	GRADE	GRADE PTS
CSCE 5210	FUND ARTIFICIAL INTELL	3.0	A	12.0
CSCE 5150	COMP ALGORITHMS	3.0	A	12.0
CSCE 5300	BIG DATA, DATA SCIENCE	3.0	A	12.0

Academic Status Good Standing

Graduate	EARN HRS	P/NP HRS	ATMP HRS	GRADE PNTS	GPA
Current	9.0	0.0	9.0	36.0	4.000
Cumulative	9.0	0.0	9.0	36.0	4.000
TRF Hrs Included	9.0				

2023 Spring

COURSE NUMBER	TITLE	EARN HRS	GRADE	GRADE PTS
CSCE 5430	SOFTWARE ENGINEER	3.0	B	9.0
CSCE 5310	EMPIRICAL ANALYSIS	3.0	A	12.0
CSCE 5350	FUNDAMENTALS OF DB	3.0	B	9.0

Academic Status Good Standing

Graduate	EARN HRS	P/NP HRS	ATMP HRS	GRADE PNTS	GPA
Current	9.0	0.0	9.0	30.0	3.333
Cumulative	18.0	0.0	18.0	66.0	3.666
TRF Hrs Included	18.0				

2023 Fall

Academic Status Good Standing

Graduate	EARN HRS	P/NP HRS	ATMP HRS	GRADE PNTS	GPA
Current	0.0	0.0	0.0	0.0	0.000
Cumulative	18.0	0.0	18.0	66.0	3.666
TRF Hrs Included	18.0				

Status 08/28/2023 Withdrawn

*** End of Graduate Transcript ***

Sheri Schwartz, Registrar

This officially sealed and signed transcript is printed on green security paper with the signature printed in blue. IF SIGNATURE AND SEAL ARE NOT VISIBLE OR ARE DISCOLORED, DO NOT ACCEPT AS AN OFFICIAL

This transcript processed and delivered by Parchment

CSCE 5350 004 FUNDAMENTALS OF DATABSE SYSTEMS

GROUP-8

PROJECT DESCRIPTION:

In this project we are creating a database for a National Pharmacy Company. This Company buys drugs from manufacturers and sells them in their stores.

For such a company which operates on a national scale we require to store so much information about various things. We need to store the information about their stores, warehouses, the employees that work there, the patients that visit the stores, the various drugs that are stored at warehouses and sold at the stores. All this information is crucial to the working of the Company. This data can be used to gain information about the sales of a particular drug, keep an eye on the inventory, manage the employees etc.

To create and store such information we need to know what types and how the information should be stored. Hence, we require a database with a good design. To design a database, we need to know about every piece of information that we will be storing in the database, how they are related to each other and how many types there are. So, we need to do a requirements analysis.

After researching on what information that such a database should contain our group decided to include the following:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager, Building ID

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN, Insurance Num

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Import/Export, Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

Two or more Entities that we included above might have a relation between them which might create additional attributes in an entity while creating the tables. Examples of such relations are:

An Employee works in a Store (One to Many)

Store/Warehouse belongs to a Region (One to Many)

Drugs stored in a Warehouse (Many to Many)

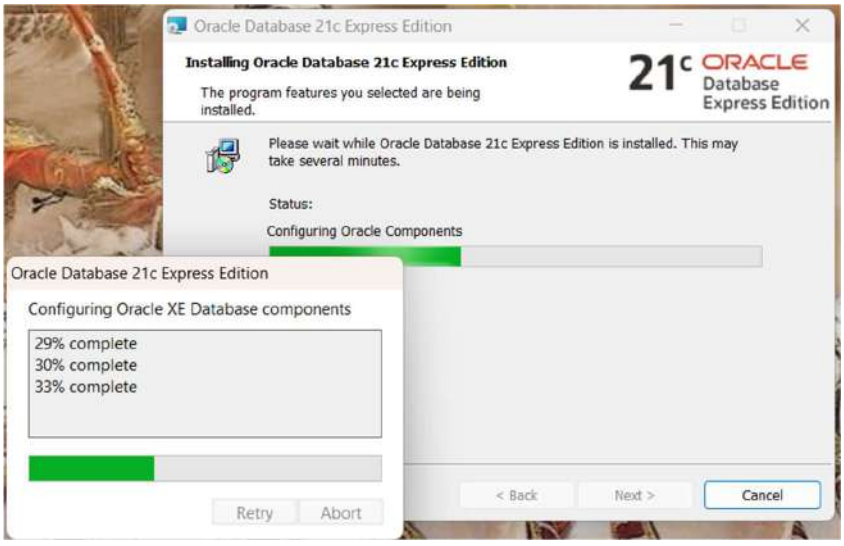
Warehouse supplies drugs to Store (Many to Many)

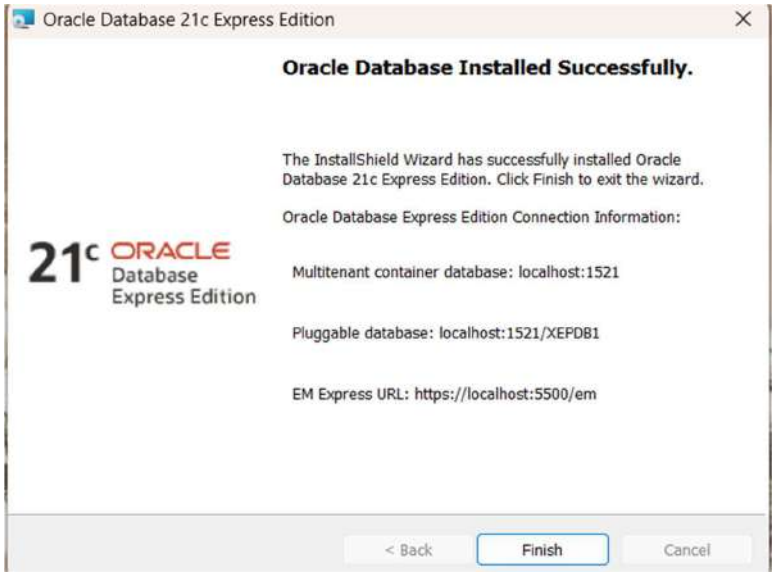
Description about the Entities:

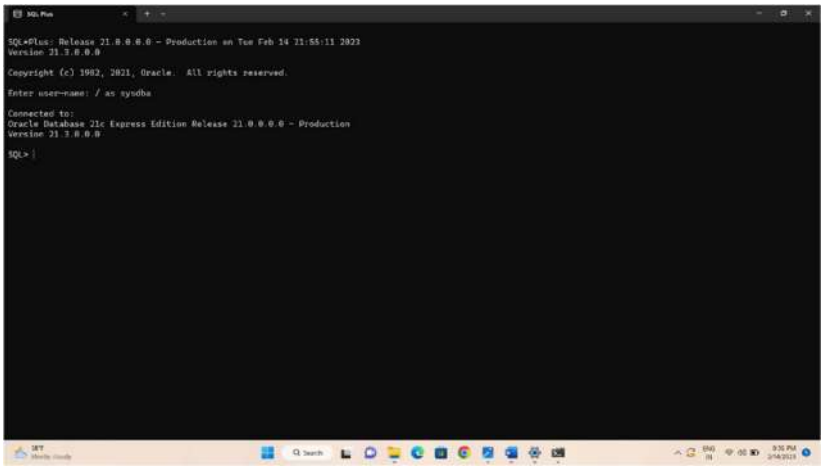
- Store: This entity contains all the information about the various stores that sell the medicine. Their location, their assigned manager and pharmacist, and the region it belongs to. The Store Id is the Primary Key for this.
- Warehouse: This contains the details about a particular warehouse of the company. Its location, capacity, current stock quantity, the region, and the manager. Warehouse Id is the Primary Key.
- Region: This entity is used to ease the management of various stores and warehouses. It contains all the IDs of the stores and the warehouses that belong to a particular region and about the manager. In this Building ID is a multi-valued attribute which contains the IDs of the stores and warehouses. The Region Code is the Primary Key.

- Employees: This entity consists of all the information about the people who work in the company. Their basic information, contact details, position, wage, and banking information. Employee Id (EID) is the Primary Key.
- Patients: It contains the details about the customers of the company. Their basic information, contact information and their insurance details if any. Patient ID (PID) is the Primary Key.
- Insurance: This entity contains information about a patient's insurance so that it can be used to settle the claims. It contains the insurance number, name and amount claimed with date. Insurance Number (INM) is the Primary Key.
- Drug: This is the most important entity. This contains all the information about the various drugs that are dealt by the company like drug name, price, manufacture company etc. Drug ID is the Primary Key.
- Inventory: This is used to keep track of the stock of the medicines that are dealt with by the company. This stores how much quantity of a particular drug is left at a particular location. The Building ID can be either Store ID or Warehouse ID. Drug ID together with Building ID will be the Primary Key.
- Logistics: This keeps information about the various movements of goods in the company. This is used to know what order a particular warehouse placed (import) or what drug a warehouse is sending to a store (export).
- Sales: This is used to get data about how well a particular drug is performing (selling). This reveals to us the information about the market and can be used to design market strategies.

ORACLE DATABASE INSTALLATION:







Individual Contribution:

In this phase everyone had their own ideas and so we decided that each member must come up with two entities on their own and about their attributes such that it satisfies the theme of the project and the requirements mentioned. I came up with the Entities: Drug and Logistics and the relation Drug stored in Warehouse mentioned above.

Each person has typed the description of entities that they came up with on their own. So, the description about Drugs and Logistics are given by me. Lastly this document was prepared by me.

Nitin Chakravarthy Chilukuri

11601099

CSCE 5350 004 FUNDAMENTALS OF DATABSE SYSTEMS

GROUP-8

Nitin Chakravarthy - 11601099

Surya Vamsi - 11645442

Lohitha Sai Bonthu - 11611601

Yamini Gollamudi - 11642723

Prathyusharani Dumpala - 11656342

PROJECT DESCRIPTION:

In this project we are creating a database for a National Pharmacy Company. This Company buys drugs from manufacturers and sells them in their stores.

For such a company which operates on a national scale we require to store so much information about various things. We need to store the information about their stores, warehouses, the employees that work there, the patients that visit the stores, the various drugs that are stored at warehouses and sold at the stores. All this information is crucial to the working of the Company. This data can be used to gain information about the sales of a particular drug, keep an eye on the inventory, manage the employees etc.

To create and store such information we need to know what types and how the information should be stored. Hence, we require a database with a good design. To design a database, we

need to know about every piece of information that we will be storing in the database, how they are related to each other and how many types there are. So, we need to do a requirements analysis.

After researching on what information that such a database should contain our group decided to include the following:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN, Insurance Num

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Import/Export, Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

Two or more Entities that we included above might have a relation between them which might create additional attributes in an entity while creating the tables. Examples of such relations are:

An Employee works in a Store (One to Many)

Store/Warehouse belongs to a Region (One to Many)

Drugs stored in a Warehouse (Many to Many)

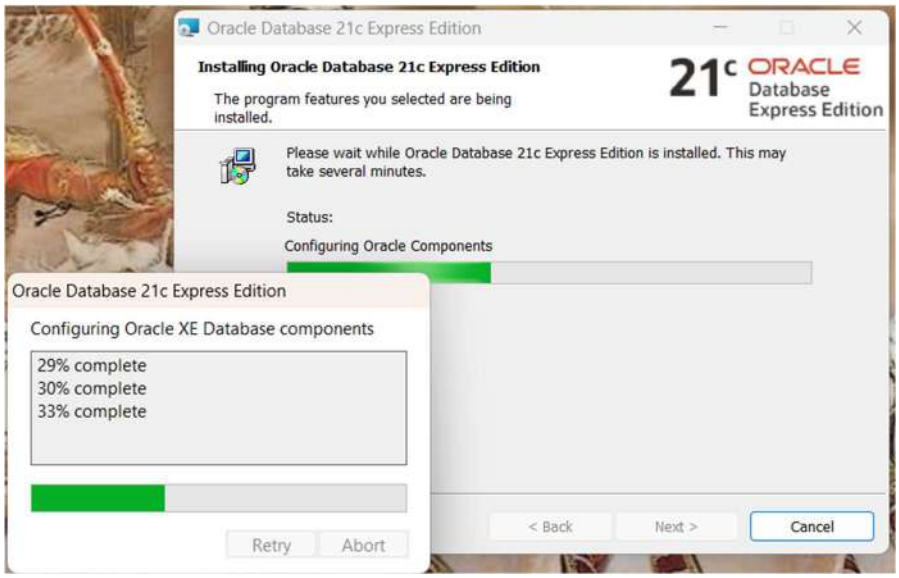
Warehouse supplies drugs to Store (Many to Many)

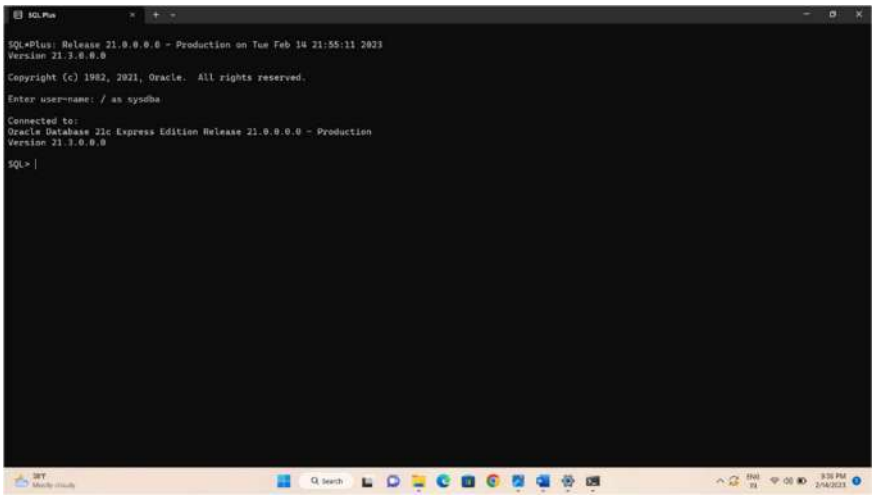
Description about the Entities:

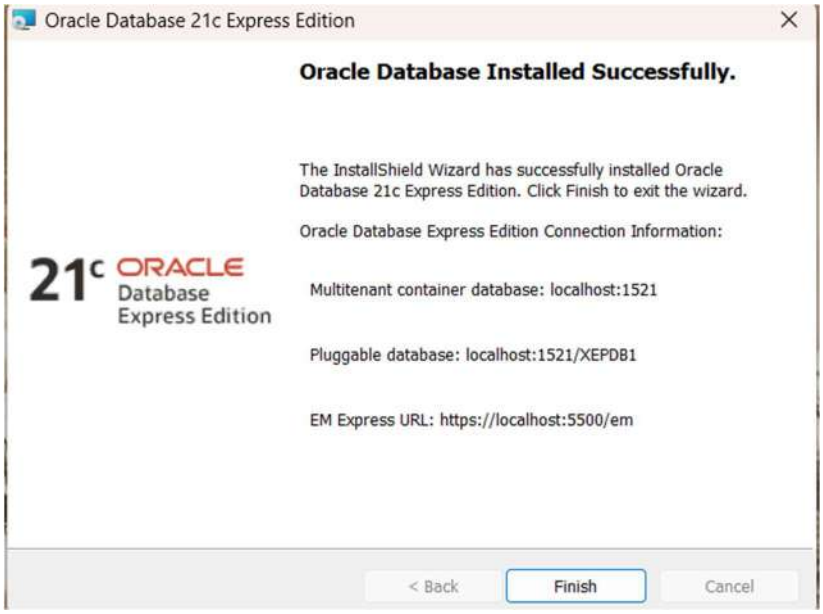
- Store: This entity contains all the information about the various stores that sell the medicine. Their location, their assigned manager and pharmacist, and the region it belongs to. The Store Id is the Primary Key for this.
- Warehouse: This contains the details about a particular warehouse of the company. Its location, capacity, current stock quantity, the region, and the manager. Warehouse Id is the Primary Key.

- Region: This entity is used to ease the management of various stores and warehouses. It contains all the IDs of the stores and the warehouses that belong to a particular region and about the manager. In this Building ID is a multi-valued attribute which contains the IDs of the stores and warehouses. The Region Code is the Primary Key.
- Employees: This entity consists of all the information about the people who work in the company. Their basic information, contact details, position, wage, and banking information. Employee Id (EID) is the Primary Key.
- Patients: It contains the details about the customers of the company. Their basic information, contact information and their insurance details if any. Patient ID (PID) is the Primary Key.
- Insurance: This entity contains information about a patient's insurance so that it can be used to settle the claims. It contains the insurance number, name and amount claimed with date. Insurance Number (INM) is the Primary Key.
- Drug: This is the most important entity. This contains all the information about the various drugs that are dealt by the company like drug name, price, manufacture company etc. Drug ID is the Primary Key.
- Inventory: This is used to keep track of the stock of the medicines that are dealt with by the company. This stores how much quantity of a particular drug is left at a particular location. The Building ID can be either Store ID or Warehouse ID. Drug ID together with Building ID will be the Primary Key.
- Logistics: This keeps information about the various movements of goods in the company. This is used to know what order a particular warehouse placed (import) or what drug a warehouse is sending to a store (export).
- Sales: This is used to get data about how well a particular drug is performing (selling). This reveals to us the information about the market and can be used to design market strategies.

ORACLE DATABASE INSTALLATION:







Individual Contribution:

In this phase everyone had their own ideas and so we decided that each member must come up with two entities on their own and about their attributes such that it satisfies the theme of the project and the requirements mentioned. I came up with the Entities: Drug and Logistics and the relation Drug stored in Warehouse mentioned above.

Each person has typed the description of entities that they came up with on their own. So, the description about Drugs and Logistics are given by me. Lastly this document was prepared by me.

CSCE 5350 004 FUNDAMENTALS OF DATABSE SYSTEMS

GROUP-8

Project Group Details

SNO	Name	ID
1.	**Surya Vamsi Chintapalli**	**11645442**
2.	**Nitin Chakravarthy Chilukuri**	**11601099**
3.	**Prathyusharani Dumpala**	**11656342**
4.	**Yamini Gollamudi**	**11642723**
5.	**Lohitha Sai Bonthu**	**11611601**

Creation of Tables:

Initial Entities and their Attributes:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager, BuildingID

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN, Insurance Num

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Import/Export, Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

ASSUMPTIONS:

Before creating the required tables, we made the following assumptions and changes:

1.) There will only be 4 Types of Employees: Normal, Pharmacist, Manager, Doctor.

2.) There are only two Genders: Male and Female.

3.) There are only three statuses for a transaction using insurance: Pending, Approved and Failed.

4.) There are three types of Drugs: Tablet, Syrup, Injection (Shot).

5.) There are 3 statuses for logistic order: Pending, In Transit and Delivered.

6.) An SSN can only be associated to only one person.

7.) We removed the Import/Export attribute from the Logistics relation, Building ID attribute from Region entity and Insurance Number attribute from Patients entity as we found they were unnecessary when we created the ER Diagram.

8.) We made a change in Inventory relation such that Building ID now refers to warehouse only and not stores. That is this relation now shows which drug is stored in which warehouse and how much quantity.

Updated Entities and their Attributes:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

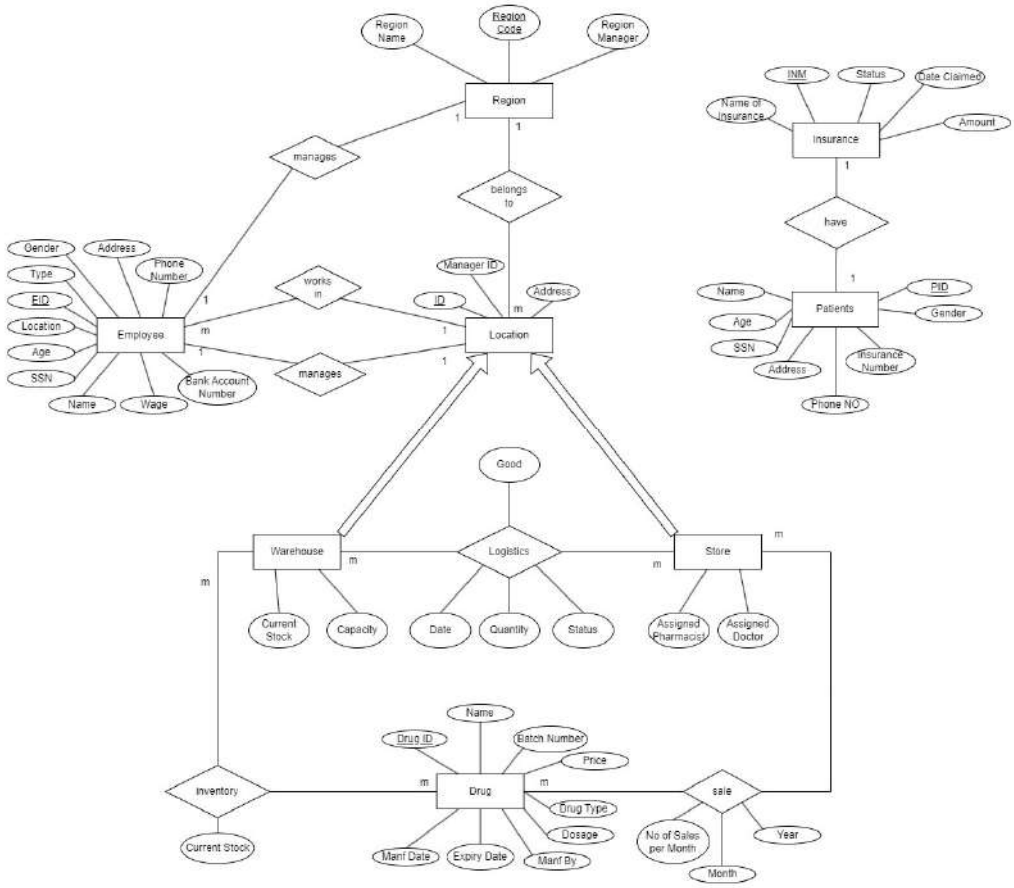
Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

The E-R Diagram for the Updated Relations is:



Creating Tables:

1.) Employee:

```
create table employee(
    EID varchar2(20) primary key,
    name varchar2(20),
    ssn varchar2(10),
    unique(ssn),
    age int check(age>18),
    gender varchar2(10) check(gender in ('Male','Female')),
    address varchar2(20),
    PhoneNO varchar2(20),
    wage varchar2(20),
    Type varchar2(20) check(Type in ('Normal','Pharmacist','Doctor','Manager')),
```

```
    Location varchar2(20),
    BankAccount varchar2(20)
);
```

```
SQL> create table employee(
  2      EID varchar2(20) primary key,
  3      name varchar2(20),
  4      ssn varchar2(10),
  5      unique(ssn),
  6      age int check(age>18),
  7      gender varchar2(10) check(gender in ('Male','Female')),
  8      address varchar2(20),
  9      PhoneNO varchar2(20),
 10      wage varchar2(20),
 11      Type varchar2(20) check(Type in ('Normal','Pharmacist','Doctor','Manager')),
 12      Location varchar2(20),
 13      BankAccount varchar2(20));

Table created.
```

2.) Region:

```
create table Region(
    RegionCode varchar2(20) primary key,
    RegionName varchar2(20),
    RegionManager varchar2(20),
    foreign key(RegionManager) references employee(EID)
);
```

```
SQL> create table Region(
  2      RegionCode varchar2(20) primary key,
  3      RegionName varchar2(20),
  4      RegionManager varchar2(20),
  5      foreign key(RegionManager) references employee(EID)
  6  );

Table created.
```

3.) Store:

```
create table store(
    StoreID varchar2(20) primary key,
    address varchar2(20),
    ManagerID varchar2(20),
    PharmacistID varchar2(20),
    foreign key(PharmacistID) references employee(EID),
    DoctorID varchar2(20),
    foreign key(DoctorID) references employee(EID),
    Region varchar2(20),
    foreign key(Region) references region(RegionCode)
```

);

```
SQL> create table store(
  2      StoreID varchar2(20) primary key,
  3      address varchar2(20),
  4      ManagerID varchar2(20),
  5      PharmacistID varchar2(20),
  6      foreign key(PharmacistID) references employee(EID),
  7      DoctorID varchar2(20),
  8      foreign key(DoctorID) references employee(EID),
  9      Region varchar2(20),
 10      foreign key(Region) references region(RegionCode)
 11  );

Table created.
```

4.) Warehouse:

```
create table warehouse(
    WarehouseID varchar2(20) primary key,
    address varchar2(20),
    ManagerID varchar2(20),
    foreign key(ManagerID) references employee(EID),
    CurrentStock varchar2(10),
    Capacity varchar2(10),
    Region varchar2(20),
    foreign key(Region) references region(RegionCode)
);
```

```
SQL> create table warehouse(
  2      WarehouseID varchar2(20) primary key,
  3      address varchar2(20),
  4      ManagerID varchar2(20),
  5      foreign key(ManagerID) references employee(EID),
  6      CurrentStock varchar2(10),
  7      Capacity varchar2(10),
  8      Region varchar2(20),
  9      foreign key(Region) references region(RegionCode)
 10  );

Table created.
```

5.) Patients:

```
create table Patients(
    PID varchar2(10) primary key,
    Name varchar2(20),
    SSN varchar2(20),
    Age int,
    Gender varchar2(10) check(Gender in ('Male','Female')),
    PhoneNO varchar2(20),
    Address varchar2(20)
```

);

```
SQL> create table Patients(
  2      PID varchar2(10) primary key,
  3      Name varchar2(20),
  4      SSN varchar2(20),
  5      Age int,
  6      Gender varchar2(10) check(Gender in ('Male','Female')),
  7      PhoneNO varchar2(20),
  8      Address varchar2(20)
  9  );

Table created.
```

6.) Insurance:

```
create table insurance(
    InsuranceNUM varchar2(20),
    Name varchar2(20),
    PID varchar2(20),
    foreign key(PID) references patients(PID),
    Amount varchar2(10),
    DateClaimed varchar2(20),
    Status varchar2(20) check(Status in ('Pending','Approved','Failed')),
    CONSTRAINT PK_INSURANCE PRIMARY KEY(InsuranceNUM, PID)
);
```

```
SQL> create table insurance(
  2      InsuranceNUM varchar2(20),
  3      Name varchar2(20),
  4      PID varchar2(20),
  5      foreign key(PID) references patients(PID),
  6      Amount varchar2(10),
  7      DateClaimed varchar2(20),
  8      Status varchar2(20) check(Status in ('Pending','Approved','Failed')),
  9      CONSTRAINT PK_INSURANCE PRIMARY KEY(InsuranceNUM, PID)
 10  );

Table created.
```

7.) Drug:

```
create table drug(
    DrugID varchar2(20) primary key,
    Name varchar2(20),
    Price varchar2(10),
    DrugType varchar2(20) check(DrugType in ('Tablet','Syrup','Injection')),
    Dosage varchar2(10),
    ManfBY varchar2(20),
    ManfDate varchar2(20),
```

```
    BatchNO varchar2(20),
    ExpiryDate varchar2(20)
);
```

```
SQL> create table drug(
  2      DrugID varchar2(20) primary key,
  3      Name varchar2(20),
  4      Price varchar2(10),
  5      DrugType varchar2(20) check(DrugType in ('Tablet','Syrup','Injection')),
  6      Dosage varchar2(10),
  7      ManfBY varchar2(20),
  8      ManfDate varchar2(20),
  9      BatchNO varchar2(20),
 10      ExpiryDate varchar2(20)
 11  );

Table created.
```

8.) Inventory:

```
create table inventory(
    DrugID varchar2(20),
    BuildingID varchar2(20),
    foreign key(DrugID) references drug(DrugID),
    foreign key(BuildingID) references warehouse(WarehouseID),
    CurrentStock varchar2(20),
    CONSTRAINT PK_INVENTORY PRIMARY KEY(DrugID, BuildingID)
);
```

```
SQL> create table inventory(
  2      DrugID varchar2(20),
  3      BuildingID varchar2(20),
  4      foreign key(DrugID) references drug(DrugID),
  5      foreign key(BuildingID) references warehouse(WarehouseID),
  6      CurrentStock varchar2(20),
  7      CONSTRAINT PK_INVENTORY PRIMARY KEY(DrugID, BuildingID)
  8  );

Table created.
```

9.) Logistics:

```
create table logistics(
    Good varchar2(20),
    WarehouseID varchar2(20),
    StoreID varchar2(20),
    foreign key(Good) references drug(DrugID),
    foreign key(WarehouseID) references warehouse(WarehouseID),
    foreign key(StoreID) references store(StoreID),
    DateofOrder varchar2(20),
```

Quantity varchar2(20),

Status varchar2(20) check(Status in ('Pending','In Transit','Delivered')),

CONSTRAINT PK_LOGISTICS PRIMARY KEY(Good, WarehouseID, StoreID, DateofOrder)

);

```
SQL> create table logistics(
  2      Good varchar2(20),
  3      WarehouseID varchar2(20),
  4      StoreID varchar2(20),
  5      foreign key(Good) references drug(DrugID),
  6      foreign key(WarehouseID) references warehouse(WarehouseID),
  7      foreign key(StoreID) references store(StoreID),
  8      DateofOrder varchar2(20),
  9      Quantity varchar2(20),
 10      Status varchar2(20) check(Status in ('Pending','In Transit','Delivered')),
 11      CONSTRAINT PK_LOGISTICS PRIMARY KEY(Good, WarehouseID, StoreID, DateofOrder)
 12  );

Table created.
```

10.) Sales:

create table sales(

Number_Of_Sales varchar2(20),

Month varchar2(20),

Year varchar2(20) check (Year>2010),

DrugID varchar2(20),

StoreID varchar2(20),

foreign key(DrugID) references drug(DrugID),

foreign key(StoreID) references store(StoreID),

CONSTRAINT PK_SALES PRIMARY KEY(DrugID, StoreID, Month, Year)

);

```
SQL> create table sales(
  2      Number_Of_Sales varchar2(20),
  3      Month varchar2(20),
  4      Year varchar2(20) check (Year>2010),
  5      DrugID varchar2(20),
  6      StoreID varchar2(20),
  7      foreign key(DrugID) references drug(DrugID),
  8      foreign key(StoreID) references store(StoreID),
  9      CONSTRAINT PK_SALES PRIMARY KEY(DrugID, StoreID, Month, Year)
 10  );

Table created.
```

Inserting Tuples into Database:

1.) Employee Table:

insert into employee values
('E1','NA1','SSN1',19,'Male','AD1','PH1','15','Normal','S1','BAC1');

```
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL>
SQL> insert into employee values ('E1','NA1','SSN1',19,'Male','AD1','PH1','15','Normal','S1','BAC1');

1 row created.

SQL>
SQL> insert into employee values ('E2','NA2','SSN2',20,'Female','AD2','PH2','15','Normal','S2','BAC2');

1 row created.

SQL>
SQL> insert into employee values ('E3','NA3','SSN3',21,'Male','AD3','PH3','15','Normal','S3','BAC3');

1 row created.

SQL>
SQL> insert into employee values ('E4','NA4','SSN4',22,'Female','AD4','PH4','15','Normal','S4','BAC4');

1 row created.

SQL>
SQL> insert into employee values ('E5','NA5','SSN5',23,'Male','AD5','PH5','15','Normal','S5','BAC5');

1 row created.

SQL>
SQL> insert into employee values ('E6','NA6','SSN6',24,'Female','AD6','PH6','15','Normal','W1','BAC6');

1 row created.

SQL>
SQL> insert into employee values ('E7','NA7','SSN7',25,'Male','AD7','PH7','15','Normal','W2','BAC7');

1 row created.

SQL>
SQL> insert into employee values ('E8','NA8','SSN8',26,'Female','AD8','PH8','15','Normal','W3','BAC8');
```

```
SQL> select * from employee;

EID                 NAME                SSN             AGE GENDER     ADDRESS             PHONENO             WAGE                TYPE
   LOCATION         BANKACCOUNT
------------------- ------------------- ---------- ---------- ---------- ------------------- ------------------- ------------------- -------------------
-- -------------------- --------------------
E1                   NA1                 SSN1             19 Male       AD1                 PH1                 15                  Normal
   S1                BAC1
E2                   NA2                 SSN2             20 Female     AD2                 PH2                 15                  Normal
   S2                BAC2
E3                   NA3                 SSN3             21 Male       AD3                 PH3                 15                  Normal
   S3                BAC3
E4                   NA4                 SSN4             22 Female     AD4                 PH4                 15                  Normal
   S4                BAC4
E5                   NA5                 SSN5             23 Male       AD5                 PH5                 15                  Normal
   S5                BAC5
E6                   NA6                 SSN6             24 Female     AD6                 PH6                 15                  Normal
   W1                BAC6
E7                   NA7                 SSN7             25 Male       AD7                 PH7                 15                  Normal
   W2                BAC7
E8                   NA8                 SSN8             26 Female     AD8                 PH8                 15                  Normal
   W3                BAC8
E9                   NA9                 SSN9             27 Male       AD9                 PH9                 15                  Normal
   W4                BAC9
E10                  NA10                SSN10            28 Female     AD10                PH10                15                  Normal
   W5                BAC10
P1                   NA11                SSN11            29 Male       AD11                PH11                25                  Pharmacist
   S1                BAC11

EID                 NAME                SSN             AGE GENDER     ADDRESS             PHONENO             WAGE                TYPE
   LOCATION         BANKACCOUNT
------------------- ------------------- ---------- ---------- ---------- ------------------- ------------------- ------------------- -------------------
-- -------------------- --------------------
P2                   NA12                SSN12            20 Female     AD12                PH12                25                  Pharmacist
   S2                BAC12
P3                   NA13                SSN13            21 Male       AD13                PH13                25                  Pharmacist
   S3                BAC13
P4                   NA14                SSN14            22 Female     AD14                PH14                25                  Pharmacist
   S4                BAC14
```

This table contains 60 entries so all could not be shown in the above screenshot.

2.) Region Table:

insert into region values(’R1’,’North’,’M1’);

```
SQL> insert into region values('R1','North','M1');

1 row created.

SQL>
SQL> insert into region values('R2','South','M2');

1 row created.

SQL>
SQL> insert into region values('R3','East','M3');

1 row created.

SQL>
SQL> insert into region values('R4','West','M4');

1 row created.

SQL>
SQL> insert into region values('R5','Cental','M5');

1 row created.

SQL>
SQL> insert into region values('R6','ECoast','M6');

1 row created.

SQL>
SQL> insert into region values('R7','WCoast','M7');

1 row created.

SQL>
SQL> insert into region values('R8','MidWest','M8');

1 row created.
```

```
SQL> select * from region;

REGIONCODE           REGIONNAME           REGIONMANAGER
-------------------- -------------------- --------------------
R1                   North                M1
R2                   South                M2
R3                   East                 M3
R4                   West                 M4
R5                   Cental               M5
R6                   ECoast               M6
R7                   WCoast               M7
R8                   MidWest              M8
R9                   Islands              M9
R10                  Alaska               M10

10 rows selected.
```

3.) Store Table:

insert into store values(‘S1’,’SAD1’,’M11’,’P1’,’D1’,’R1’);

```
SQL> insert into store values('S1','SAD1','M11','P1','D1','R1');

1 row created.

SQL>
SQL> insert into store values('S2','SAD2','M12','P2','D2','R2');

1 row created.

SQL>
SQL> insert into store values('S3','SAD3','M13','P3','D3','R3');

1 row created.

SQL>
SQL> insert into store values('S4','SAD4','M14','P4','D4','R4');

1 row created.

SQL>
SQL> insert into store values('S5','SAD5','M15','P5','D5','R5');

1 row created.

SQL>
SQL> insert into store values('S6','SAD6','M16','P6','D6','R6');

1 row created.

SQL>
SQL> insert into store values('S7','SAD7','M17','P7','D7','R7');

1 row created.

SQL>
SQL> insert into store values('S8','SAD8','M18','P8','D8','R8');

1 row created.
```

```
SQL> select * from store;

STOREID            ADDRESS            MANAGERID          PHARMACISTID       DOCTORID           REGION
------------------ ------------------ ------------------ ------------------ ------------------ ------------------
S1                 SAD1               M11                P1                 D1                 R1
S2                 SAD2               M12                P2                 D2                 R2
S3                 SAD3               M13                P3                 D3                 R3
S4                 SAD4               M14                P4                 D4                 R4
S5                 SAD5               M15                P5                 D5                 R5
S6                 SAD6               M16                P6                 D6                 R6
S7                 SAD7               M17                P7                 D7                 R7
S8                 SAD8               M18                P8                 D8                 R8
S9                 SAD9               M11                P9                 D9                 R9
S10                SAD10              M20                P10                D10                R10

10 rows selected.
```

4.) Warehouse Table:

insert into warehouse values (‘W1’,’WAD1’,’M21’,’1000’,’5000’,’R1’);

```
SQL> insert into warehouse values('W1','WAD1','M21','1000','5000','R1');

1 row created.

SQL>
SQL> insert into warehouse values('W2','WAD2','M22','1500','5000','R2');

1 row created.

SQL>
SQL> insert into warehouse values('W3','WAD3','M23','2000','5000','R3');

1 row created.

SQL>
SQL> insert into warehouse values('W4','WAD4','M24','2500','5000','R4');

1 row created.

SQL>
SQL> insert into warehouse values('W5','WAD5','M25','3000','5000','R5');

1 row created.

SQL>
SQL> insert into warehouse values('W6','WAD6','M26','4000','8000','R6');

1 row created.

SQL>
SQL> insert into warehouse values('W7','WAD7','M27','1000','8000','R7');

1 row created.

SQL>
SQL> insert into warehouse values('W8','WAD8','M28','2000','8000','R8');

1 row created.
```

```
SQL> select * from warehouse;

WAREHOUSEID          ADDRESS              MANAGERID            CURRENTSTO CAPACITY   REGION
-------------------- -------------------- -------------------- ---------- ---------- --------------------
W1                   WAD1                 M21                  1000       5000       R1
W2                   WAD2                 M22                  1500       5000       R2
W3                   WAD3                 M23                  2000       5000       R3
W4                   WAD4                 M24                  2500       5000       R4
W5                   WAD5                 M25                  3000       5000       R5
W6                   WAD6                 M26                  4000       8000       R6
W7                   WAD7                 M27                  1000       8000       R7
W8                   WAD8                 M28                  2000       8000       R8
W9                   WAD9                 M29                  2500       8000       R9
W10                  WAD10                M30                  4500       8000       R10

10 rows selected.
```

5.) Patients Table:

insert into patients (‘Pa1’, ‘PNAM1’,’PSSN1’,20,’Male’,’PPHNO1’,’PAD1’);

```
SQL> insert into patients values ('Pa1','PNAM1','PSSN1',20,'Male','PPHNO1','PAD1');

1 row created.

SQL>
SQL> insert into patients values ('Pa2','PNAM2','PSSN2',21,'Female','PPHNO2','PAD2');

1 row created.

SQL>
SQL> insert into patients values ('Pa3','PNAM3','PSSN3',22,'Male','PPHNO3','PAD3');

1 row created.

SQL>
SQL> insert into patients values ('Pa4','PNAM4','PSSN4',23,'Female','PPHNO4','PAD4');

1 row created.

SQL>
SQL> insert into patients values ('Pa5','PNAM5','PSSN5',24,'Male','PPHNO5','PAD5');

1 row created.

SQL>
SQL> insert into patients values ('Pa6','PNAM6','PSSN6',25,'Female','PPHNO6','PAD6');

1 row created.

SQL>
SQL> insert into patients values ('Pa7','PNAM7','PSSN7',26,'Male','PPHNO7','PAD7');

1 row created.

SQL>
SQL> insert into patients values ('Pa8','PNAM8','PSSN8',27,'Female','PPHNO8','PAD8');

1 row created.
```

```
SQL> select * from patients;

PID        NAME                SSN                       AGE GENDER     PHONENO             ADDRESS
---------- ------------------- ------------------- ---------- ---------- ------------------- --------------------
Pa1        PNAM1               PSSN1                       20 Male       PPHNO1              PAD1
Pa2        PNAM2               PSSN2                       21 Female     PPHNO2              PAD2
Pa3        PNAM3               PSSN3                       22 Male       PPHNO3              PAD3
Pa4        PNAM4               PSSN4                       23 Female     PPHNO4              PAD4
Pa5        PNAM5               PSSN5                       24 Male       PPHNO5              PAD5
Pa6        PNAM6               PSSN6                       25 Female     PPHNO6              PAD6
Pa7        PNAM7               PSSN7                       26 Male       PPHNO7              PAD7
Pa8        PNAM8               PSSN8                       27 Female     PPHNO8              PAD8
Pa9        PNAM9               PSSN9                       28 Male       PPHNO9              PAD9
Pa10       PNAM10              PSSN10                      29 Female     PPHNO10             PAD10

10 rows selected.
```

6.) Insurance Table:

insert into insurance values (‘INSNUM1’,’INSCOM1’,’Pa1’,’100$’,’03-01-2023’,’Pending’);

```
SQL> insert into insurance values('INSNUM1','INSCOM1','Pa1','100$','03-01-2023','Pending');

1 row created.

SQL>
SQL> insert into insurance values('INSNUM2','INSCOM2','Pa2','200$','03-02-2023','Approved');

1 row created.

SQL>
SQL> insert into insurance values('INSNUM3','INSCOM3','Pa3','300$','03-03-2023','Approved');

1 row created.

SQL>
SQL> insert into insurance values('INSNUM4','INSCOM4','Pa4','400$','03-04-2023','Approved');

1 row created.

SQL>
SQL> insert into insurance values('INSNUM5','INSCOM5','Pa5','250$','03-05-2023','Approved');

1 row created.

SQL>
SQL> insert into insurance values('INSNUM6','INSCOM6','Pa6','200$','03-06-2023','Approved');

1 row created.

SQL>
SQL> insert into insurance values('INSNUM7','INSCOM7','Pa7','300$','03-07-2023','Pending');

1 row created.

SQL>
SQL> insert into insurance values('INSNUM8','INSCOM8','Pa8','150$','03-08-2023','Pending');

1 row created.
```

```
SQL> select * from insurance;

INSURANCENUM         NAME                PID                 AMOUNT     DATECLAIMED          STATUS
-------------------- ------------------- ------------------- ---------- -------------------- --------------------
INSNUM1              INSCOM1             Pa1                 100$       03-01-2023           Pending
INSNUM2              INSCOM2             Pa2                 200$       03-02-2023           Approved
INSNUM3              INSCOM3             Pa3                 300$       03-03-2023           Approved
INSNUM4              INSCOM4             Pa4                 400$       03-04-2023           Approved
INSNUM5              INSCOM5             Pa5                 250$       03-05-2023           Approved
INSNUM6              INSCOM6             Pa6                 200$       03-06-2023           Approved
INSNUM7              INSCOM7             Pa7                 300$       03-07-2023           Pending
INSNUM8              INSCOM8             Pa8                 150$       03-08-2023           Pending
INSNUM9              INSCOM9             Pa9                 200$       03-09-2023           Approved
INSNUM10             INSCOM10            Pa10                200$       03-10-2023           Failed

10 rows selected.
```

7.) Drug Table:

insert into drug values (‘Dg1’,’DNAM1’,’5$’,’Tablet’,’10mg’,’ManfCon1’,’03-01-2023’,’BAT01’,’03-01-2024’);

```
SQL> insert into drug values('Dg1','DNAM1','5$','Tablet','10mg','ManfCom1','03-01-2023','BAT01','03-01-2024');

1 row created.

SQL>
SQL> insert into drug values('Dg2','DNAM2','10$','Syrup','20mg','ManfCom2','03-02-2023','BAT02','03-02-2024');

1 row created.

SQL>
SQL> insert into drug values('Dg3','DNAM3','15$','Injection','30mg','ManfCom3','03-03-2023','BAT03','03-03-2024');

1 row created.

SQL>
SQL> insert into drug values('Dg4','DNAM4','20$','Tablet','5mg','ManfCom4','03-04-2023','BAT04','03-04-2024');

1 row created.

SQL>
SQL> insert into drug values('Dg5','DNAM5','25$','Syrup','10mg','ManfCom5','03-05-2023','BAT05','03-05-2024');

1 row created.

SQL>
SQL> insert into drug values('Dg6','DNAM6','5$','Tablet','10mg','ManfCom6','03-06-2023','BAT06','03-06-2024');

1 row created.

SQL>
SQL> insert into drug values('Dg7','DNAM7','15$','Injection','5mg','ManfCom7','03-07-2023','BAT07','03-07-2024');

1 row created.

SQL>
SQL> insert into drug values('Dg8','DNAM8','25$','Syrup','15mg','ManfCom8','03-08-2023','BAT08','03-08-2024');

1 row created.
```

```
SQL> select * from drug;

DRUGID              NAME                PRICE      DRUGTYPE            DOSAGE     MANFBY              MANFDATE            BATCHNO             EXPIRYDA
TE
------------------- ------------------- ---------- ------------------- ---------- ------------------- ------------------- ------------------- --------
-------------
Dg1                 DNAM1               5$         Tablet              10mg       ManfCom1            03-01-2023          BAT01               03-01-20
24
Dg2                 DNAM2               10$        Syrup               20mg       ManfCom2            03-02-2023          BAT02               03-02-20
24
Dg3                 DNAM3               15$        Injection           30mg       ManfCom3            03-03-2023          BAT03               03-03-20
24
Dg4                 DNAM4               20$        Tablet              5mg        ManfCom4            03-04-2023          BAT04               03-04-20
24
Dg5                 DNAM5               25$        Syrup               10mg       ManfCom5            03-05-2023          BAT05               03-05-20
24
Dg6                 DNAM6               5$         Tablet              10mg       ManfCom6            03-06-2023          BAT06               03-06-20
24
Dg7                 DNAM7               15$        Injection           5mg        ManfCom7            03-07-2023          BAT07               03-07-20
24
Dg8                 DNAM8               25$        Syrup               15mg       ManfCom8            03-08-2023          BAT08               03-08-20
24
Dg9                 DNAM9               35$        Tablet              10mg       ManfCom9            03-09-2023          BAT09               03-09-20
24
Dg10                DNAM10              45$        Injection           25mg       ManfCom10           03-10-2023          BAT10               03-10-20
24

10 rows selected.
```

8.) Inventory Table:

insert into inventory values (‘Dg1’,’W1’,’200’);

```
SQL> insert into inventory values('Dg1','W1','200');

1 row created.

SQL>
SQL> insert into inventory values('Dg2','W2','300');

1 row created.

SQL>
SQL> insert into inventory values('Dg3','W3','400');

1 row created.

SQL>
SQL> insert into inventory values('Dg4','W4','500');

1 row created.

SQL>
SQL> insert into inventory values('Dg5','W5','200');

1 row created.
```

```
SQL> select * from inventory;

DRUGID               BUILDINGID           CURRENTSTOCK
-------------------- -------------------- --------------------
Dg1                  W1                   200
Dg2                  W2                   300
Dg3                  W3                   400
Dg4                  W4                   500
Dg5                  W5                   200
Dg6                  W6                   100
Dg7                  W7                   150
Dg8                  W8                   250
Dg9                  W9                   200
Dg10                 W10                  300

10 rows selected.
```

9.) Logistics Table:

insert into inventory values (‘Dg1’,’W1’,’S1’,’03-01-2023’,’50’,’Pending’);

```
SQL> insert into logistics values('Dg1','W1','S1','03-01-2023','50','Pending');

1 row created.

SQL>
SQL> insert into logistics values('Dg2','W2','S2','03-02-2023','25','In Transit');

1 row created.

SQL>
SQL> insert into logistics values('Dg3','W3','S3','03-03-2023','30','Delivered');

1 row created.

SQL>
SQL> insert into logistics values('Dg4','W4','S4','03-04-2023','40','In Transit');

1 row created.

SQL>
SQL> insert into logistics values('Dg5','W5','S5','03-05-2023','50','Pending');

1 row created.

SQL>
SQL> insert into logistics values('Dg6','W6','S6','03-06-2023','50','Pending');

1 row created.

SQL>
SQL> insert into logistics values('Dg7','W7','S7','03-07-2023','10','Pending');

1 row created.

SQL>
SQL> insert into logistics values('Dg8','W8','S8','03-08-2023','20','Delivered');

1 row created.
```

```
SQL> select * from logistics;

GOOD               WAREHOUSEID        STOREID            DATEOFORDER        QUANTITY           STATUS
------------------ ------------------ ------------------ ------------------ ------------------ --------------------
Dg1                W1                 S1                 03-01-2023         50                 Pending
Dg2                W2                 S2                 03-02-2023         25                 In Transit
Dg3                W3                 S3                 03-03-2023         30                 Delivered
Dg4                W4                 S4                 03-04-2023         40                 In Transit
Dg5                W5                 S5                 03-05-2023         50                 Pending
Dg6                W6                 S6                 03-06-2023         50                 Pending
Dg7                W7                 S7                 03-07-2023         10                 Pending
Dg8                W8                 S8                 03-08-2023         20                 Delivered
Dg9                W9                 S9                 03-09-2023         25                 Pending
Dg10               W10                S10                03-10-2023         30                 In Transit

10 rows selected.
```

10.) Sales Table:

insert into sales values (‘40’,’January’,’2023’,’Dg1’,’S1’);

```
SQL> insert into sales values('40','January','2023','Dg1','S1');

1 row created.

SQL>
SQL> insert into sales values('30','Febuary','2023','Dg2','S2');

1 row created.

SQL>
SQL> insert into sales values('50','March','2023','Dg3','S3');

1 row created.

SQL>
SQL> insert into sales values('10','April','2022','Dg4','S4');

1 row created.

SQL>
SQL> insert into sales values('20','May','2022','Dg5','S5');

1 row created.

SQL>
SQL> insert into sales values('25','June','2022','Dg6','S6');

1 row created.

SQL>
SQL> insert into sales values('45','July','2022','Dg7','S7');

1 row created.

SQL>
SQL> insert into sales values('15','August','2022','Dg8','S8');

1 row created.
```

```
SQL> select * from sales;

NUMBER_OF_SALES      MONTH                YEAR                 DRUGID               STOREID
-------------------- -------------------- -------------------- -------------------- --------------------
40                   January              2023                 Dg1                  S1
30                   Febuary              2023                 Dg2                  S2
50                   March                2023                 Dg3                  S3
10                   April                2022                 Dg4                  S4
20                   May                  2022                 Dg5                  S5
25                   June                 2022                 Dg6                  S6
45                   July                 2022                 Dg7                  S7
15                   August               2022                 Dg8                  S8
10                   September            2021                 Dg9                  S9
10                   October              2021                 Dg10                 S10

10 rows selected.
```

Nitin Chakravarthy Chilukuri

11601099

CSCE 5350 004

FUNDAMENTALS OF DATABASE SYSTEMS

PROJECT - PART 3

GROUP-8

Name	ID
Nitin Chakravarthy Chilukuri	11601099
Surya Vamsi Chintapalli	11645442
Prathyusharani Dumpala	11656342
Yamini Gollamudi	11642723
Lohitha Sai Bonthu	11611601

INITIAL ENTITIES & ATTRIBUTES:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

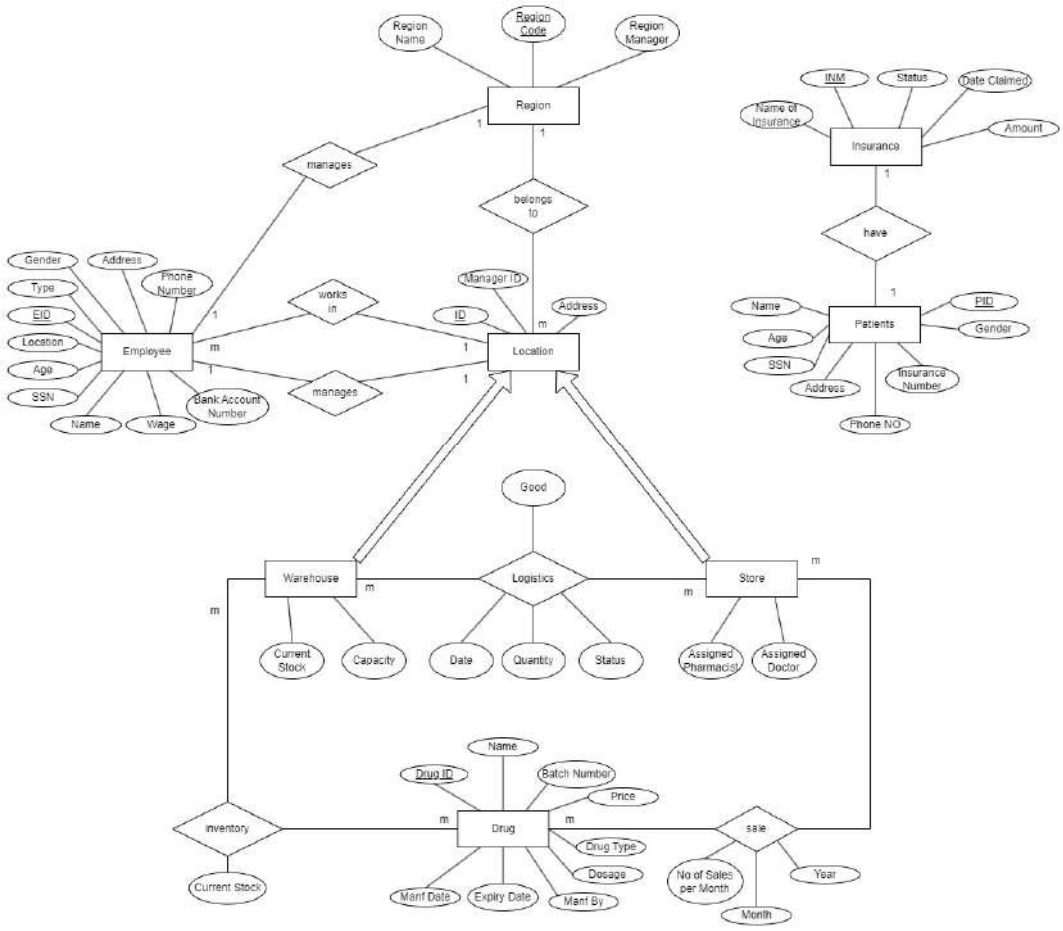
Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

ER DIAGRAM:



ASSUMPTIONS:

To solve the given Queries in the project we do not have the necessary tables and hence it will be impossible to solve the Queries.

So, we decided to add two new tables, Prescription and Payroll, to the existing entities.

As a result, the entities change as follows.

UPDATED ENTITIES & ATTRIBUTES:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Payroll: EID, Date, Hours Worked

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

Prescription: PrescriptionID, DoctorID, PatientID, Date_Prescribed

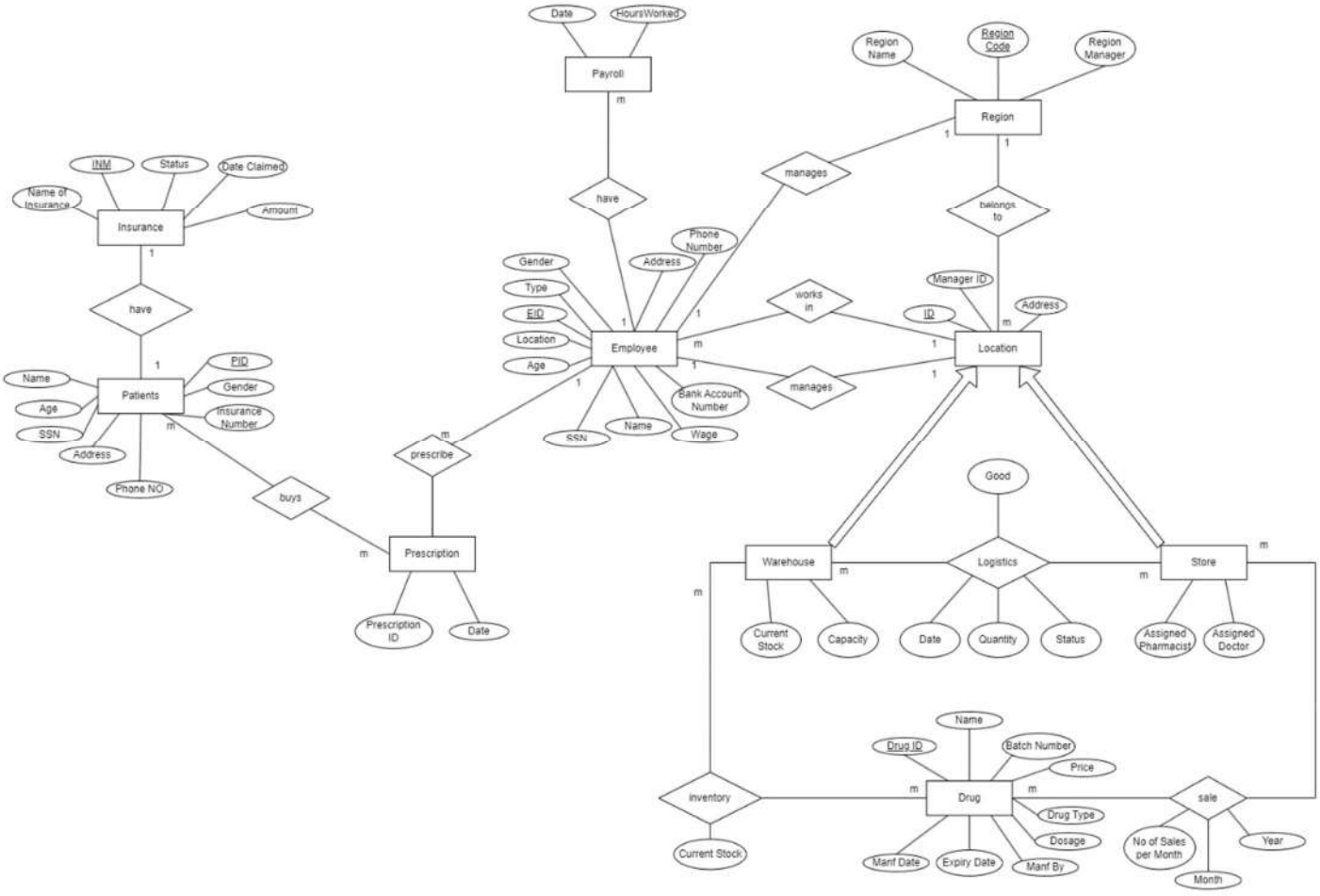
Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

ER DIAGRAM OF UPDATED ENTITIES:



CREATION OF THE TWO NEW TABLES:

Payroll:

```
create table Payroll(
    EID varchar2(10),
    foreign key(EID) references employee(EID) on DELETE CASCADE,
    Work_Date Date,
    Hours_Worked int check(Hours_Worked<24),
    PRIMARY KEY(EID, Work_Date)
);
```

Prescription:

```
create table Prescription(
    PrescriptionID varchar2(10),
    DoctorID varchar2(10),
    foreign key(DoctorID) references employee(EID) on DELETE CASCADE,
    PatientID varchar2(10),
    foreign key(PatientID) references Patients(PID) on DELETE CASCADE,
    Date_Prescribed Date,
    CONSTRAINT PK_PRESCRIPTION PRIMARY KEY(PrescriptionID, DoctorID)
);
```

TUPLES IN THE NEW TABLES:

Payroll:

```
SQL Plus

EID        WORK_DATE HOURS_WORKED
---------- --------- ------------
E10        06-MAR-23            2
E10        04-MAR-23            2
E9         09-MAR-23            4
E8         08-MAR-23            4
E7         06-MAR-23            3
E6         09-MAR-23            4
E6         06-MAR-23            4
E6         04-MAR-23            6
E5         09-MAR-23            8
E5         07-MAR-23            7
E5         04-MAR-23            6
E4         07-MAR-23            8
E4         06-MAR-23            8
E4         05-MAR-23            8
E4         04-MAR-23            8
E3         05-MAR-23            2
E3         04-MAR-23            6
E2         07-MAR-23            9
E2         06-MAR-23            9
E2         05-MAR-23            9
E2         04-MAR-23            9
E1         06-MAR-23            7
E1         05-MAR-23            8
E1         04-MAR-23            9
E10        06-MAR-22            2
E10        04-MAR-22            2
E9         09-MAR-22            4
E8         08-MAR-22            4
E7         06-MAR-22            3
E6         09-MAR-22            4
E6         06-MAR-22            4
E6         04-MAR-22            6
E5         09-MAR-22            8
E5         07-MAR-22            7
E5         04-MAR-22            6
E4         07-MAR-22            8
E4         06-MAR-22            8
```

Prescription:

```
SQL> select * from prescription;

PRESCRIPTI DOCTORID   PATIENTID  DATE_PRES
---------- ---------- ---------- ---------
PR1        D1         Pa1        02-JUN-21
PR2        D2         Pa2        02-JUN-21
PR3        D3         Pa3        02-JUN-21
PR4        D4         Pa4        02-JUN-21
PR5        D5         Pa5        02-JUN-21
PR6        D1         Pa6        02-JUN-21
PR7        D1         Pa7        02-JUN-21
PR8        D1         Pa8        02-JUN-21
PR9        D2         Pa9        02-JUN-21
PR10       D2         Pa1        02-JUN-21
PR11       D3         Pa2        02-JUN-21
PR12       D3         Pa3        02-JUN-21
PR13       D4         Pa4        02-JUN-21
PR14       D5         Pa5        02-JUN-21
PR15       D5         Pa6        02-JUN-21

15 rows selected.
```

GIVEN QUERIES:

1) List the total number of prescriptions group by doctors and pharmacy location issued on June 2nd, 2021.

SELECT e.name AS doctor_name, s.address AS pharmacy_location, COUNT(*) AS num_prescriptions

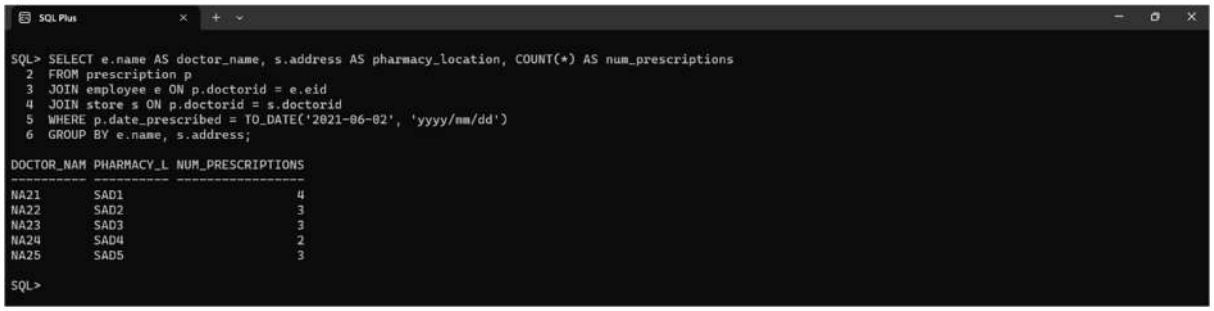
FROM prescription p

JOIN employee e ON p.doctorid = e.eid

JOIN store s ON p.doctorid = s.doctorid

WHERE p.date_prescribed = TO_DATE('2021-06-02', 'yyyy/mm/dd')

GROUP BY e.name, s.address;



2) Find locations with inventories that list at least one missing product (a product that has quantity of zero in the inventory).

select Distinct BUILDINGID from inventory where currentstock = 0;

3) Find the name of the employee(s) that had worked the most hours on November 3, 2022

select e.NAME

from employee e where e.EID in (

select p.EID from payroll p

where p.HOURS_WORKED=(select max(HOURS_WORKED) from payroll) and p.WORK_DATE=TO_DATE('2022-11-03','yyyy/mm/dd')

);

```
SQL> select e.NAME
  2  from employee e
  3  where e.EID in (
  4     select p.EID
  5     from payroll p
  6      where p.HOURS_WORKED=(select max(HOURS_WORKED) from payroll) and p.WORK_DATE=TO_DATE('2022-11-03','yyyy/mm/dd')
  7      );

NAME
----------
NA9
NA3

SQL>
```

4)List the items that currently have the least quantity on inventory.

```
select DNAME, Stock
from(
SELECT drug.Name as DNAME, SUM(inventory.CurrentStock) AS Stock
FROM drug
INNER JOIN inventory ON drug.DrugID = inventory.DrugID
GROUP BY drug.Name
)
where Stock=(
select min(Stock)
from(
SELECT drug.Name as DNAME, SUM(inventory.CurrentStock) AS Stock
FROM drug
INNER JOIN inventory ON drug.DrugID = inventory.DrugID
GROUP BY drug.Name
)
);
```

```
SQL> select DNAME, Stock
  2  from(
  3  SELECT drug.Name as DNAME, SUM(inventory.CurrentStock) AS Stock
  4  FROM drug
  5  INNER JOIN inventory ON drug.DrugID = inventory.DrugID
  6  GROUP BY drug.Name
  7  )
  8  where Stock=(
  9  select min(Stock)
 10  from(
 11  SELECT drug.Name as DNAME, SUM(inventory.CurrentStock) AS Stock
 12  FROM drug
 13  INNER JOIN inventory ON drug.DrugID = inventory.DrugID
 14  GROUP BY drug.Name
 15  )
 16  );

DNAME           STOCK
---------- ----------
DNAM5             250
DNAM6             250
DNAM8             250

SQL>
```

5) Print the payroll from March 4, 2022, to March 10, 2022 displaying employee name, hours worked and total salary for all employees

```
select ENAME, SUM(HOURS) as TOTAL_HOURS, SUM(Salary) as TOTAL_SALARY

from(

SELECT employee.name as ENAME, Payroll.Hours_Worked as HOURS, Payroll.Hours_Worked * employee.wage AS Salary

FROM employee

INNER JOIN Payroll ON employee.EID = Payroll.EID

WHERE Payroll.Work_Date>= TO_DATE('2022-03-04','yyyy/mm/dd') AND Payroll.Work_Date<=TO_DATE('2022-03-10', 'yyyy/mm/dd')

)

GROUP BY ENAME;
```

```
SQL> select ENAME, SUM(HOURS) as TOTAL_HOURS, SUM(Salary) as TOTAL_SALARY
  2  from(
  3  SELECT employee.name as ENAME, Payroll.Hours_Worked as HOURS, Payroll.Hours_Worked * employee.wage AS Salary
  4  FROM employee
  5  INNER JOIN Payroll ON employee.EID = Payroll.EID
  6  WHERE Payroll.Work_Date>= TO_DATE('2022-03-04','yyyy/mm/dd') AND Payroll.Work_Date<=TO_DATE('2022-03-10', 'yyyy/mm/dd')
  7  )
  8  GROUP BY ENAME;

ENAME      TOTAL_HOURS TOTAL_SALARY
---------- ----------- ------------
NA1                24          360
NA2                36          540
NA3                 8          120
NA4                32          480
NA5                21          315
NA6                14          210
NA7                 3           45
NA8                 4           60
NA9                 4           60
NA10                4           60

10 rows selected.

SQL>
```

6) Design a delete statement to delete employees working less than 5 hours from March 4, 2023, to March 10, 2023.

```
delete from Employee where Employee.EID in(

select EID

from payroll p

where WORK_DATE>= TO_DATE('2022-03-04','yyyy/mm/dd') and WORK_DATE<= TO_DATE('2022-03-10','yyyy/mm/dd')

GROUP by EID

having SUM(HOURS_WORKED)<5);
```

```
SQL> delete from Employee where Employee.EID in(
  2  select EID
  3  from payroll p
  4  where WORK_DATE>= TO_DATE('2022-03-04','yyyy/mm/dd') and WORK_DATE<= TO_DATE('2022-03-10','yyyy/mm/dd')
  5  GROUP by EID
  6  having SUM(HOURS_WORKED)<5
  7  );

4 rows deleted.

SQL>
```

7) Design an update statement to give a 23% salary raise to employees working more than 5 hours from March 4, 2023, to March 10, 2023.

update Employee

set wage=wage*1.23

where Employee.EID in(

select EID

from payroll p

where WORK_DATE>= TO_DATE('2022-03-04','yyyy/mm/dd') and WORK_DATE<= TO_DATE('2022-03-10','yyyy/mm/dd')

GROUP by EID

having SUM(HOURS_WORKED)>=5

);

ADDITIONAL QUERIES:

1) List the StoreId of all the Stores in a Region R6

select StoreID from Store where Region='R6';

2) List the warehouseID of all warehouses in Region R1

select WarehouseID from Warehouse where Region='R1';

3) List the name and phone number of all employees working in stores with StoreID S1, S2

select Name, PhoneNO from Employee where(Location='S1' or Location='S2');

```
SQL> select Name, PhoneNO from Employee where(Location='S1' or Location='S2');

NAME       PHONENO
---------- ----------
NA1        PH1
NA2        PH2
NA11       PH11
NA12       PH12
NA21       PH21
NA22       PH22
NA31       PH31
NA32       PH32

8 rows selected.

SQL>
```

4) Get the name(s) of employee who worked for the maximum hours in a day in the month of March 2023

select Name

from Employee

where Employee.EID in(

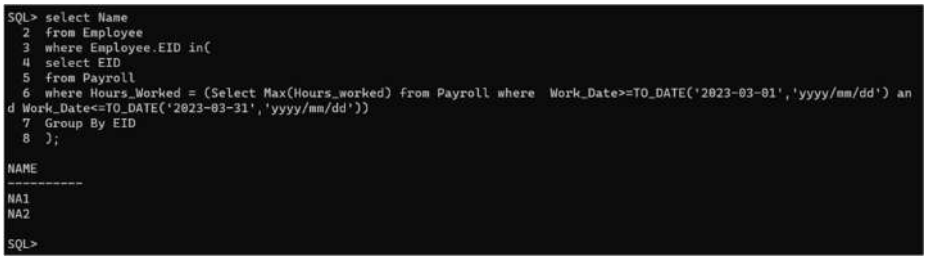
select EID

from Payroll

where Hours_Worked = (Select Max(Hours_worked) from Payroll where Work_Date>=TO_DATE('2023-03-01','yyyy/mm/dd') and Work_Date<=TO_DATE('2023-03-31','yyyy/mm/dd'))

Group By EID

);



```
SQL> select Name
  2  from Employee
  3  where Employee.EID in(
  4  select EID
  5  from Payroll
  6  where Hours_Worked = (Select Max(Hours_worked) from Payroll where  Work_Date>=TO_DATE('2023-03-01','yyyy/mm/dd') an
d Work_Date<=TO_DATE('2023-03-31','yyyy/mm/dd'))
  7  Group By EID
  8  );

NAME
----------
NA1
NA2

SQL>
```

5) List all the PrescriptionIDs and the name of Patients who bought them issued on June 2nd 2021

select pr.PrescriptionID, pa.Name

from Prescription pr

join Patients pa on pr.PATIENTID=pa.PID

where pr.Date_Prescribed = TO_DATE('2021-06-02','yyyy/mm/dd');

```
SQL> select pr.PrescriptionID, pa.Name
  2  from Prescription pr
  3  join Patients pa on pr.PATIENTID=pa.PID
  4  where pr.Date_Prescribed = TO_DATE('2021-06-02','yyyy/mm/dd')
  5  ;

PRESCRIPTI NAME
---------- ----------
PR1        PNAM1
PR2        PNAM2
PR3        PNAM3
PR4        PNAM4
PR5        PNAM5
PR6        PNAM6
PR7        PNAM7
PR8        PNAM8
PR9        PNAM9
PR10       PNAM1
PR11       PNAM2

PRESCRIPTI NAME
---------- ----------
PR12       PNAM3
PR13       PNAM4
PR14       PNAM5
PR15       PNAM6

15 rows selected.

SQL>
```

6) List all the drugs names and type of drugs manufactured by ManfCom2

select Name, Drugtype from Drug where Manfby='ManfCom2';

```
SQL> select Name, Drugtype from Drug where Manfby='ManfCom2';

NAME       DRUGTYPE
---------- ----------
DNAM2      Syrup

SQL>
```

7) List all the item names currently in stock in warehouse W7

select Name

from Drug

where DrugID in(

select DrugID

from Inventory

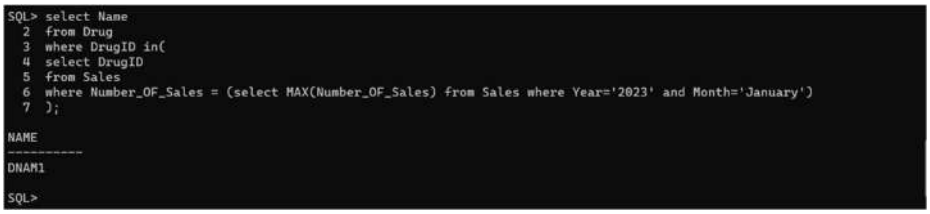
where Buildingid='W7' and Currentstock>0

);

```
SQL> select Name
  2  from Drug
  3  where DrugID in(
  4  select DrugID
  5  from Inventory
  6  where Buildingid='W7' and Currentstock>0
  7  );

NAME
----------
DNAM7

SQL>
```

8) Get the name(s) of Drug with the highest No of Sales per Month in Jan 2023

```
select Name
from Drug
where DrugID in(
select DrugID
from Sales
where Number_OF_Sales = (select MAX(Number_OF_Sales) from Sales where Year='2023' and Month='January')
);
```

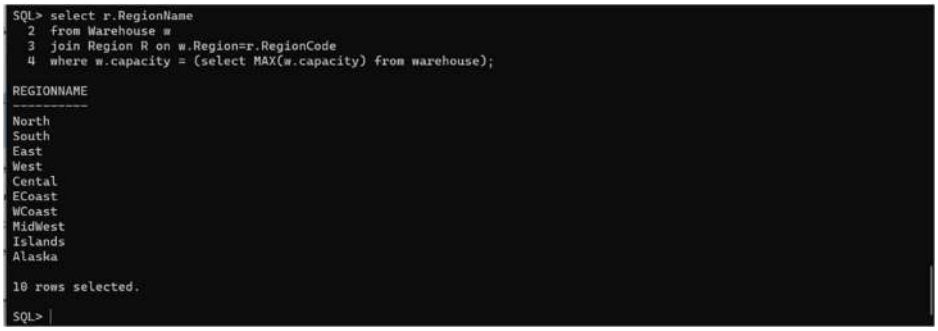


9) Get the list of logistics order that are in 'In Transit' stage

```
select * from Logistics where Status='In Transit';
```

10) Get the name of the Region in which the Warehouse with highest storage capacity is in

```
select r.RegionName
from Warehouse w
join Region R on w.Region=r.RegionCode
where w.capacity = (select MAX(w.capacity) from warehouse);
```



ADDITIONAL UPDATE QUERIES:

1) Increase the Hourly wage of all managers to 35

update Employee set wage=35 where Type='Manager';

```
SQL> update Employee set wage=35 where Type='Manager';

30 rows updated.

SQL>
```

2) Update the capacity of warehouse W2 to increase it by 20%

update Warehouse set capacity= capacity*1.2 where warehouseid='W2';

```
SQL> update Warehouse set capacity= capacity*1.2 where warehouseid='W2';

1 row updated.

SQL>
```

3) Change the phone number of patient Pa3 to NPPHNO3

update Patients set PhoneNo='NPPHNO3' where PID='Pa3';

```
SQL> update Patients set PhoneNo='NPPHNO3' where PID='Pa3';

1 row updated.

SQL>
```

4) Update the cost of Drug Dg4 to decrease its cost by 10%

update Drug set Price= Price*0.9 where Drugid='Dg4';

```
SQL> update Drug set Price= Price*0.9 where Drugid='Dg4';

1 row updated.

SQL>
```

5) Update the inventory to status of Drug Dg6 in Warehouse W6

update Inventory set currentstock=200 where buildingid='W6' and drugid='Dg6';

```
SQL> update Inventory set currentstock=200 where buildingid='W6' and drugid='Dg6';

1 row updated.

SQL>
```

6) Update all the logistic orders between Warehouse W5 and Store S5 to Delivered

update Logistics

set status='Delivered'

where warehouseid='W5' and storeid='S5';

```
SQL> update Logistics
  2  set status='Delivered'
  3  where warehouseid='W5' and storeid='S5';

1 row updated.

SQL>
```

ADDITIONAL DELETE QUERIES:

1) Delete Information regarding Patient Pa7

delete from Patients where PID='Pa7';

```
SQL> delete from Patients where PID='Pa7';
1 row deleted.
SQL>
```

2) Delete all Payroll records that were entered before 2013

delete from Payroll where Work_Date<TO_DATE('2013-01-01','yyyy/mm/dd');

```
SQL> delete from Payroll where Work_Date<TO_DATE('2013-01-01','yyyy/mm/dd');
0 rows deleted.
SQL>
```

3) Delete all logistics orders that were placed before 2010

delete from logistics where dateoforder<TO_DATE('2010-01-01','yyyy/mm/dd');

```
SQL> delete from logistics where dateoforder<TO_DATE('2010-01-01','yyyy/mm/dd');
0 rows deleted.
SQL>
```

4) Delete the records of all Prescriptions that were issued before 2018

delete from prescription where date_prescribed<TO_DATE('2018-01-01','yyyy/mm/dd');

```
SQL> delete from prescription where date_prescribed<TO_DATE('2018-01-01','yyyy/mm/dd');
0 rows deleted.
SQL>
```

5) Delete the information about drugs that expire on or before Dec 31st 2019

delete from drug where expirydate<=TO_DATE('2019-12-31','yyyy/mm/dd');

```
SQL> delete from drug where expirydate<=TO_DATE('2019-12-31','yyyy/mm/dd');
0 rows deleted.
SQL>
```

6) Delete information regarding employees working in S10

delete from employee where location='S10';

```
SQL> delete from employee where location='S10';
3 rows deleted.
SQL>
```

LIST OF ADDITIONAL QUERIES SOLVED:

1) List the StoreId of all the Stores in a Region R6

2) List the warehouseID of all warehouses in Region R1

3) List the name and phone number of all employees working in stores with StoreID S1, S2

4) Get the name(s) of employee who worked for the maximum hours in a day in the month of March 2023

5) List all the PrescriptionIDs and the name of Patients who bought them issued on June 2nd 2021

6) List all the drugs names and type of drugs manufactured by ManfCom2

7) List all the item names currently in stock in warehouse W7

8) Get the name(s) of Drug with the highest No of Sales per Month in Jan 2023

9) Get the list of logistics order that are in 'In Transit' stage

10) Get the name of the Region in which the Warehouse with highest storage capacity is in

LIST OF ADDITIONAL UPDATE QUERIES SOLVED:

1) Increase the Hourly wage of all managers to 35

2) Update the capacity of warehouse W2 to increase it by 20%

3) Change the phone number of patient Pa3 to NPPHNO3

4) Update the cost of Drug Dg4 to decrease its cost by 10%

5) Update the inventory to status of Drug Dg6 in Warehouse W6

6) Update all the logistic orders between Warehouse W5 and Store S5 to Delivered

LIST OF ADDITIONAL DELETE QUERIES SOLVED:

1) Delete Information regarding Patient Pa7

2) Delete all Payroll records that were entered before 2013

3) Delete all logistics orders that were placed before 2010

4) Delete the records of all Prescriptions that were issued before 2018

5) Delete the information about drugs that expire on or before Dec 31st 2019

6) Delete information regarding employees working in S10

CSCE 5350 004

FUNDAMENTALS OF DATABASE SYSTEMS

PROJECT - PART 4

GROUP-8

Name	ID
Nitin Chakravarthy Chilukuri	11601099
Surya Vamsi Chintapalli	11645442
Prathyusharani Dumpala	11656342
Yamini Gollamudi	11642723
Lohitha Sai Bonthu	11611601

ATTRIBUTES:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Payroll: EID, Date, Hours Worked

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

Prescription: PrescriptionID, DoctorID, PatientID, Date_Prescribed

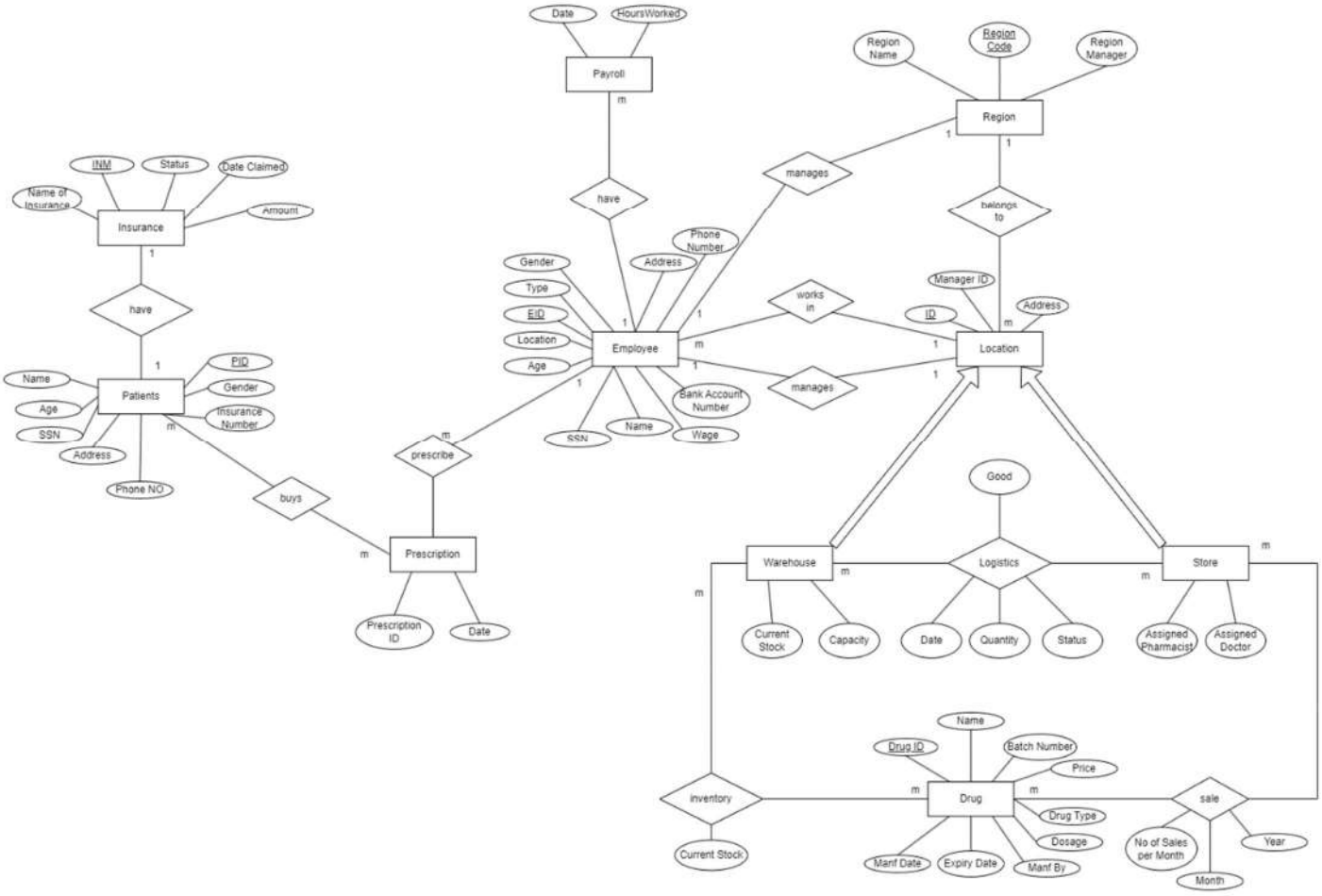
Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Good, Date, Warehouse ID, Store ID, Quantity, Status

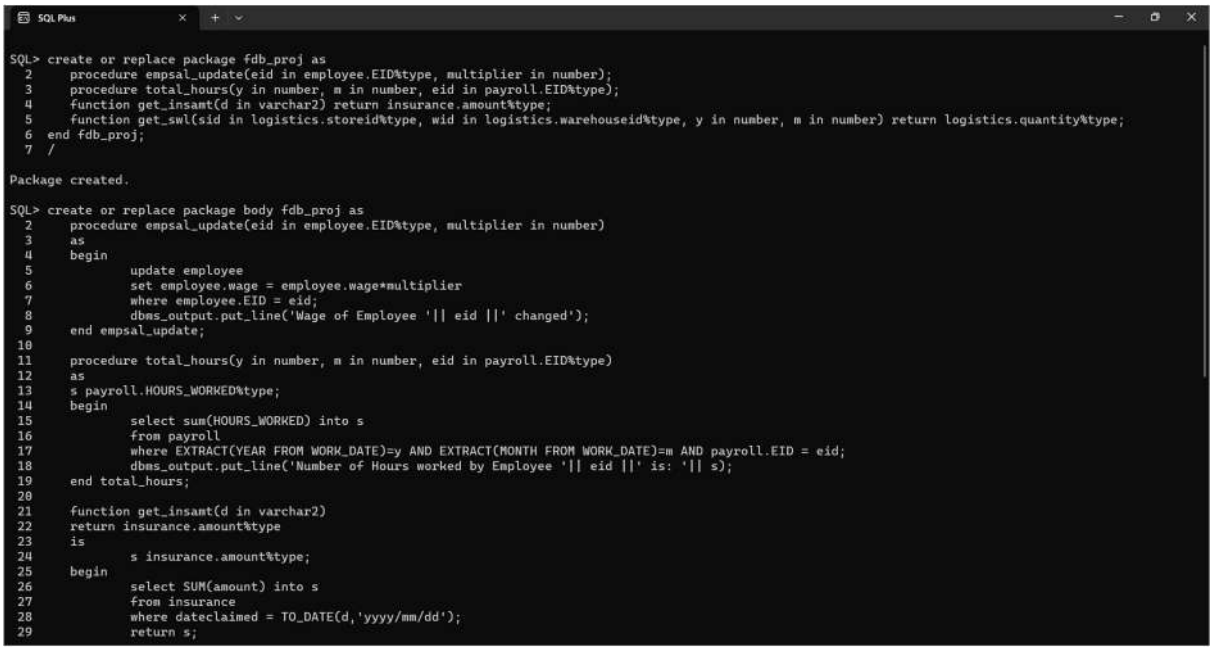
Sales: No of sales per month, Month, Year, Drug ID, Store ID

ER DIAGRAM OF ENTITIES:

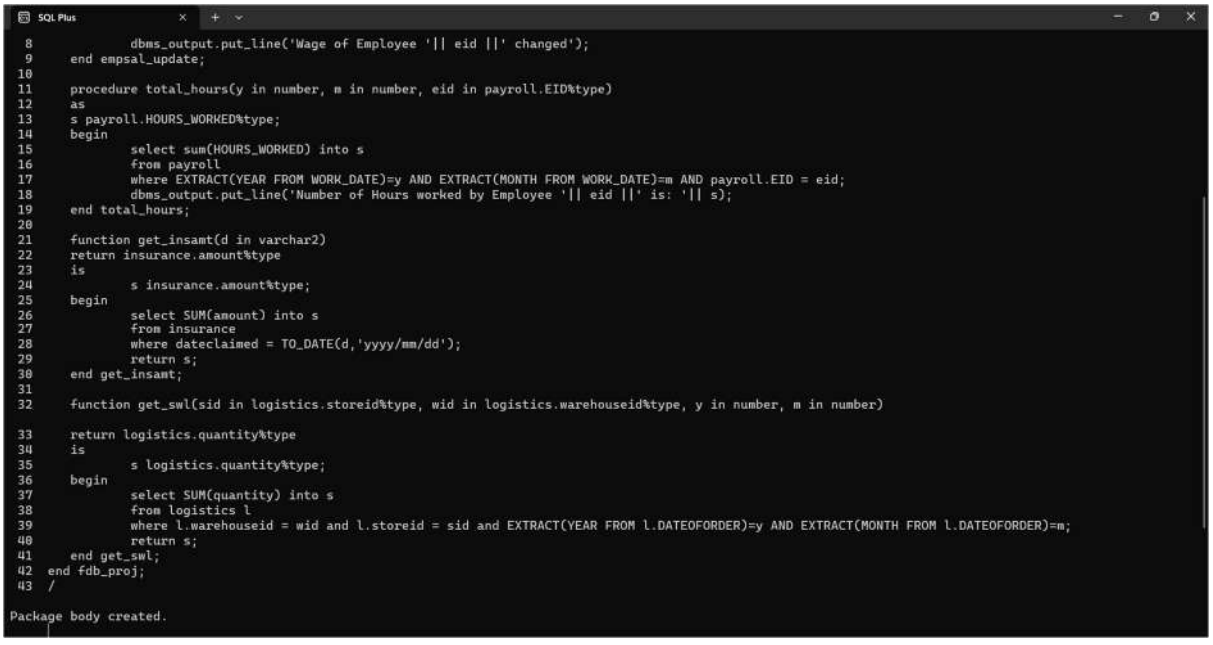


CREATION & BODY OF PACKAGE:

```
create or replace package fdb_proj as
        procedure empsal_update(eid in employee.EID%type, multiplier in number);
        procedure total_hours(y in number, m in number, eid in payroll.EID%type);
        function get_insamt(d in varchar2) return insurance.amount%type;
        function get_swl(sid in logistics.storeid%type, wid in logistics.warehouseid%type, y in number, m in number) return logistics.quantity%type;
end fdb_proj;
```



```
SQL> create or replace package fdb_proj as
  2     procedure empsal_update(eid in employee.EID%type, multiplier in number);
  3     procedure total_hours(y in number, m in number, eid in payroll.EID%type);
  4     function get_insamt(d in varchar2) return insurance.amount%type;
  5     function get_swl(sid in logistics.storeid%type, wid in logistics.warehouseid%type, y in number, m in number) return logistics.quantity%type;
  6  end fdb_proj;
  7  /

Package created.

SQL> create or replace package body fdb_proj as
  2     procedure empsal_update(eid in employee.EID%type, multiplier in number)
  3     as
  4     begin
  5             update employee
  6             set employee.wage = employee.wage*multiplier
  7             where employee.EID = eid;
  8             dbms_output.put_line('Wage of Employee '|| eid ||' changed');
  9     end empsal_update;
 10
 11     procedure total_hours(y in number, m in number, eid in payroll.EID%type)
 12     as
 13     s payroll.HOURS_WORKED%type;
 14     begin
 15             select sum(HOURS_WORKED) into s
 16             from payroll
 17             where EXTRACT(YEAR FROM WORK_DATE)=y AND EXTRACT(MONTH FROM WORK_DATE)=m AND payroll.EID = eid;
 18             dbms_output.put_line('Number of Hours worked by Employee '|| eid ||' is: '|| s);
 19     end total_hours;
 20
 21     function get_insamt(d in varchar2)
 22     return insurance.amount%type
 23     is
 24             s insurance.amount%type;
 25     begin
 26             select SUM(amount) into s
 27             from insurance
 28             where dateclaimed = TO_DATE(d,'yyyy/mm/dd');
 29             return s;
```



```
  8             dbms_output.put_line('Wage of Employee '|| eid ||' changed');
  9     end empsal_update;
 10
 11     procedure total_hours(y in number, m in number, eid in payroll.EID%type)
 12     as
 13     s payroll.HOURS_WORKED%type;
 14     begin
 15             select sum(HOURS_WORKED) into s
 16             from payroll
 17             where EXTRACT(YEAR FROM WORK_DATE)=y AND EXTRACT(MONTH FROM WORK_DATE)=m AND payroll.EID = eid;
 18             dbms_output.put_line('Number of Hours worked by Employee '|| eid ||' is: '|| s);
 19     end total_hours;
 20
 21     function get_insamt(d in varchar2)
 22     return insurance.amount%type
 23     is
 24             s insurance.amount%type;
 25     begin
 26             select SUM(amount) into s
 27             from insurance
 28             where dateclaimed = TO_DATE(d,'yyyy/mm/dd');
 29             return s;
 30     end get_insamt;
 31
 32     function get_swl(sid in logistics.storeid%type, wid in logistics.warehouseid%type, y in number, m in number)

 33     return logistics.quantity%type
 34     is
 35             s logistics.quantity%type;
 36     begin
 37             select SUM(quantity) into s
 38             from logistics l
 39             where l.warehouseid = wid and l.storeid = sid and EXTRACT(YEAR FROM l.DATEOFORDER)=y AND EXTRACT(MONTH FROM l.DATEOFORDER)=m;
 40             return s;
 41     end get_swl;
 42  end fdb_proj;
 43  /

Package body created.
```

This Package contains 2 Procedures and 2 Functions.

These are:

- empsal_update: This is a procedure used to update the wage of an employee. It takes EID and multiplier as inputs. New_Wage = Old_Wage*Multiplier.

```
create or replace procedure empsal_update(eid in employee.EID%type, multiplier in number)
as
begin
        update employee
        set employee.wage = employee.wage*multiplier
        where employee.EID = eid;
        dbms_output.put_line('Wage of Employee '|| eid ||' changed');
end;
```

- total_hours: This is a procedure to get the total number of hours an employee worked in a particular month in a particular year. This takes year, month and EID as input.

```
create or replace procedure total_hours(y in number, m in number, eid in payroll.EID%type)
as
s payroll.HOURS_WORKED%type;
begin
        select sum(HOURS_WORKED) into s
        from payroll
        where EXTRACT(YEAR FROM WORK_DATE)=y AND EXTRACT(MONTH FROM WORK_DATE)=m AND payroll.EID = eid;
        dbms_output.put_line('Number of Hours worked by Employee '|| eid ||' is: '|| s);
end;
```

- get_insamt: This is a function to get the total amount of insurance claimed on a particular date. It takes a date as input.

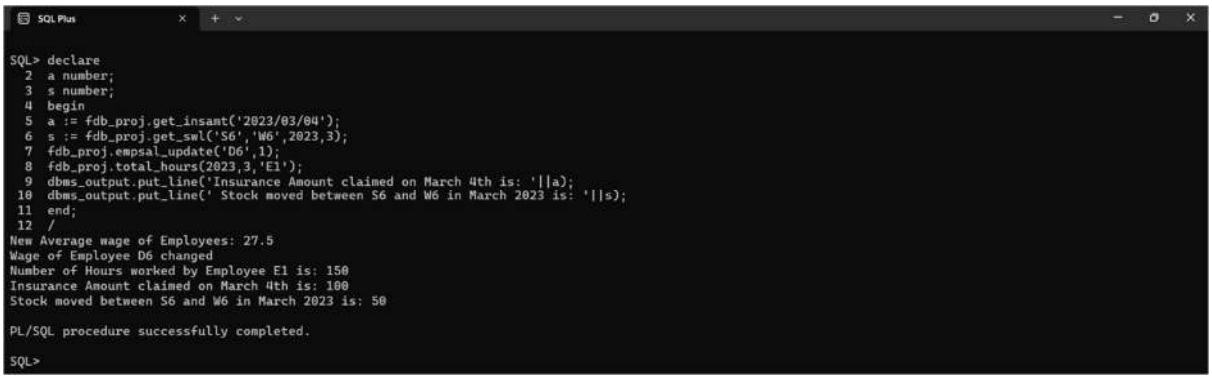
```
create or replace function get_insamt(d in varchar2)
return insurance.amount%type
is
s insurance.amount%type;
```

```
begin
        select SUM(amount) into s
        from insurance
        where dateclaimed = TO_DATE(d,'yyyy/mm/dd');
        return s;
end;
```

- get_swl: This is a function that gives us how much stock is moved from a warehouse to a store in a month. It takes StoreId, WarehouseID, month and year as inputs.

```
create or replace function get_swl(sid in logistics.storeid%type, wid in
logistics.warehouseid%type, y in number, m in number)
return logistics.quantity%type
is
s logistics.quantity%type;
begin
        select SUM(quantity) into s
        from logistics l
        where l.warehouseid = wid and l.storeid = sid and EXTRACT(YEAR FROM
        l.DATEOFORDER)=y AND EXTRACT(MONTH FROM l.DATEOFORDER)=m;
        return s;
end;
```

When executed, the above four results in the following:



We can observe a line ‘New Average wage of Employees: 27.5’ in the output. This is the result of a trigger which we will see later.

PROCEDURES & FUNCTIONS:

1) avg_hw: This is a procedure that displays the average working hours and average wage per hour for an employee in a month.

```
create or replace procedure avg_hw
as
c1 number;
s1 number;
avg_h number;
c2 number;
s2 number;
avg_w number;
begin
        select SUM(count(DISTINCT EID)) into c1
        from payroll
        group by payroll.EID;
        select SUM(HOURS_WORKED) into s1 from payroll;
        avg_h := s1/c1;

        select SUM(count(DISTINCT EID)) into c2
        from employee
        group by employee.EID;
        select SUM(wage) into s2 from employee;
        avg_w := s2/c2;

        dbms_output.put_line('Avereage Number of Hours worked by an Employee is: '|| avg_h);
        dbms_output.put_line('Avereage Wage of an Employee is: '|| avg_w);
end;
```

```
    create or replace procedure avg_hw
  2 as
  3 c1 number;
  4 s1 number;
  5 avg_h number;
  6 c2 number;
  7 s2 number;
  8 avg_w number;
  9 begin
 10    select SUM(count(DISTINCT EID)) into c1
 11    from payroll
 12    group by payroll.EID;
 13    select SUM(HOURS_WORKED) into s1 from payroll;
 14    avg_h := s1/c1;
 15
 16    select SUM(count(DISTINCT EID)) into c2
 17    from employee
 18    group by employee.EID;
 19    select SUM(wage) into s2 from employee;
 20    avg_w := s2/c2;
 21
 22    dbms_output.put_line('Avereage Number of Hours worked by an Employee is: '|| avg_h);
 23    dbms_output.put_line('Avereage Wage of an Employee is: '|| avg_w);
 24 end;
 25 /

Procedure created.
```

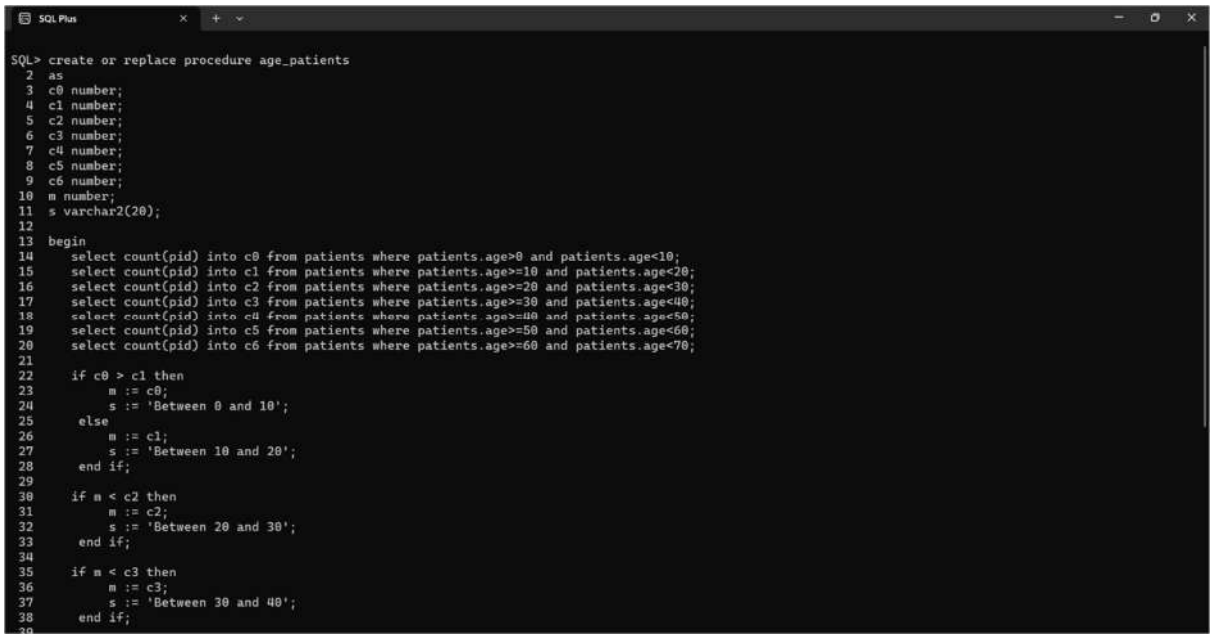
```
SQL> execute avg_hw;
Avereage Number of Hours worked by an Employee is: 37.4
Avereage Wage of an Employee is: 27.5

PL/SQL procedure successfully completed.

SQL>
```

2) age_patients: This procedure tells us which age group contains the greatest number of patients.

```
create or replace procedure age_patients
as
c0 number;
c1 number;
c2 number;
c3 number;
c4 number;
c5 number;
c6 number;
m number;
s varchar2(20);

begin
        select count(pid) into c0 from patients where patients.age>0 and patients.age<10;
        select count(pid) into c1 from patients where patients.age>=10 and patients.age<20;
        select count(pid) into c2 from patients where patients.age>=20 and patients.age<30;
        select count(pid) into c3 from patients where patients.age>=30 and patients.age<40;
```

```
select count(pid) into c4 from patients where patients.age>=40 and patients.age<50;
select count(pid) into c5 from patients where patients.age>=50 and patients.age<60;
select count(pid) into c6 from patients where patients.age>=60 and patients.age<70;

if c0 > c1 then
        m := c0;
        s := 'Between 0 & 10';
else
        m := c1;
        s := 'Between 10 & 20';
end if;
if m < c2 then
        m := c2;
        s := 'Between 20 & 30';
end if;
if m < c3 then
        m := c3;
        s := 'Between 30 & 40';
end if;
if m < c4 then
        m := c4;
        s := 'Between 40 & 50';
end if;
if m < c5 then
        m := c5;
        s := 'Between 50 & 60';
end if;
if m < c6 then
        m := c6;
        s := 'Between 60 & 70';
```

```
    end if;
    dbms_output.put_line('There are more patients in the ages ' || s);
end;
```



3) logistics_status: This is a procedure that tells us how many logistics orders are in pending state and how many orders are In Transit.

```
create or replace procedure logistics_status
as
p number;
t number;
begin
    select count(status) into p
    from logistics
    where status = 'Pending';
    select count(status) into t
    from logistics
    where status = 'In Transit';
```

```
    dbms_output.put_line('Total Number of Orders in Pending Status is: '|| p);
    dbms_output.put_line('Total Number of Orders in Transit are: '|| t);
end;
```

```
SQL> create or replace procedure logistics_status
  2  as
  3  p number;
  4  t number;
  5  begin
  6     select count(status) into p
  7     from logistics
  8     where status = 'Pending';
  9
 10     select count(status) into t
 11     from logistics
 12     where status = 'In Transit';
 13
 14     dbms_output.put_line('Total Number of Orders in Pending Status is: '|| p);
 15     dbms_output.put_line('Total Number of Orders in Transit are: '|| t);
 16  end;
 17  /

Procedure created.
```

```
SQL> execute logistics_status;
Total Number of Orders in Pending Status is: 5
Total Number of Orders in Transit are: 3

PL/SQL procedure successfully completed.
```

4) get_stock: This is a function that gives us the total amount of stock present for a certain drug. It takes DrugId as the input.

```
create or replace function get_stock(did in drug.DRUGID%type)
return inventory.CURRENTSTOCK%type
is
s inventory.CURRENTSTOCK%type;
begin
    select SUM(currentstock) into s
    from inventory
    where inventory.DRUGID = did;
    return s;
end;
```

```
SQL> create or replace function get_stock(did in drug.DRUGID%type)
  2  return inventory.CURRENTSTOCK%type
  3  is
  4  s inventory.CURRENTSTOCK%type;
  5  begin
  6     select SUM(currentstock) into s
  7      from inventory
  8     where inventory.DRUGID = did;
  9
 10     return s;
 11  end;
 12  /

Function created.

SQL>
```

5) get_stockr: This is a function that gives us the total stock available of a drug in all the warehouses in a particular region. It takes DrugId and RegionCode as inputs.

```
create or replace function get_stockr(did in drug.drugid%type, region in
region.regioncode%type)
return inventory.currentstock%type
is
        s inventory.currentstock%type;
        begin
        select SUM(currentstock) into s
        from inventory
        where buildingid in
        (
                select warehouseid
                from warehouse
                where warehouse.region = region
        )
        and inventory.drugid = did;
        return s;
end;
```

```
SQL> create or replace function get_stockr(did in drug.drugid%type, region in region.regioncode%type)
  2  return inventory.currentstock%type
  3  is
  4     s inventory.currentstock%type;
  5     begin
  6     select SUM(currentstock) into s
  7     from inventory
  8     where buildingid in
  9     (
 10             select warehouseid
 11             from warehouse
 12             where warehouse.region = region
 13     )
 14     and inventory.drugid = did;
 15     return s;
 16  end;
 17  /

Function created.

SQL>
```

6) get_totalsalesd: This is a function that that gives us how many units a particular drug has been sold across all the stores in a particular year.

```
create or replace function get_totalsalesd(did in drug.drugid%type, y in sales.year%type)
return sales.number_of_sales%type
is
```

s sales.number_of_sales%type;

begin

select sum(number_of_sales) into s

from sales

where drugid = did and sales.year=y;

return s;

end;

```
SQL> create or replace function get_totalsalesd(did in drug.drugid%type, y in sales.year%type)
  2  return sales.number_of_sales%type
  3  is
  4  s sales.number_of_sales%type;
  5  begin
  6     select sum(number_of_sales) into s
  7     from sales
  8     where drugid = did and sales.year=y;
  9
 10     return s;
 11  end;
 12  /

Function created.
```

When the above three functions get_stock, get_stockr and get_totalsalesd are executed we get the following result.

```
SQL> declare
  2  a number;
  3  b number;
  4  c number;
  5  begin
  6  a := get_stock('Dg6');
  7  b := get_stockr('Dg6', 'R1');
  8  c := get_totalsalesd('Dg6', 2022);
  9  dbms_output.put_line('Total Stock of Dg6 is: ' || a);
 10  dbms_output.put_line('Total Stock of Dg6 in Region R1 is: ' || b);
 11  dbms_output.put_line('Total Sales of Dg6 is: ' || c);
 12  end;
 13  /
Total Stock of Dg6 is: 250
Total Stock of Dg6 in Region R1 is: 250
Total Sales of Dg6 is: 25

PL/SQL procedure successfully completed.

SQL>
```

TRIGGERS:

1) num_of_emp: This is a trigger on employee table that gets triggered after delete or insert operations on the employee table. This displays the number of employees present after delete or insert operations are performed.

CREATE OR REPLACE TRIGGER num_of_emp

AFTER DELETE OR INSERT ON employee

DECLARE

c number;

BEGIN

select count(*) into c

```
    from employee;
    dbms_output.put_line('Total Number of Employees: '|| c);
END;
```

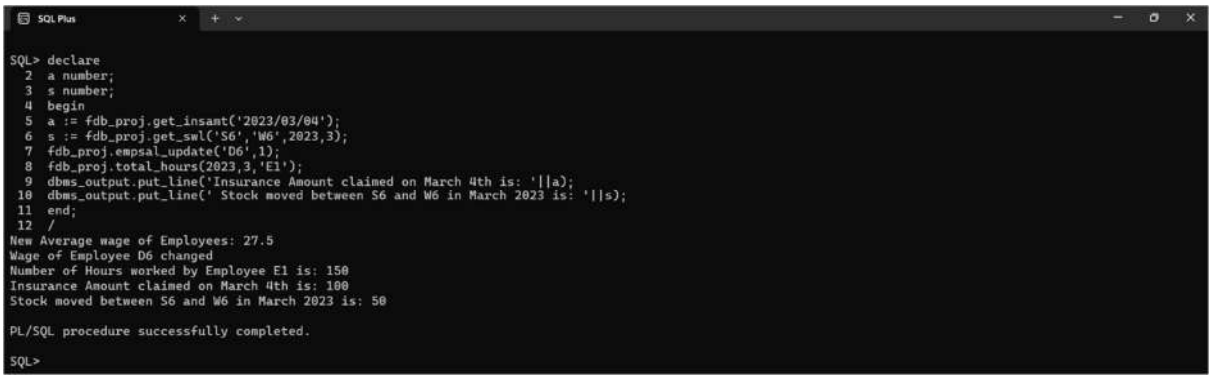
```
SQL> CREATE OR REPLACE TRIGGER num_of_emp
  2  AFTER DELETE OR INSERT ON employee
  3  DECLARE
  4     c number;
  5  BEGIN
  6     select count(*) into c
  7     from employee;
  8    dbms_output.put_line('Total Number of Employees: '|| c);
  9  END;
 10  /

Trigger created.
```

```
SQL> insert into employee values ('M31','NA61','SSN61',37,'Female','AD61','PH61','30','Manager','R10','BAC61');
Total Number of Employees: 61

1 row created.

SQL> delete from employee where eid = 'M31';
Total Number of Employees: 60

1 row deleted.
```

2) avg_wage: This trigger gives us the average hourly wage of all employees after there has been an update operation on the wage column of employee table.

```
CREATE OR REPLACE TRIGGER avg_wage
AFTER UPDATE of wage ON employee
DECLARE
    c number;
BEGIN
    select AVG(wage) into c
    from employee;
    dbms_output.put_line('New Average wage of Employees: '|| c);
END;
```

```
SQL> CREATE OR REPLACE TRIGGER avg_wage
  2  AFTER UPDATE of wage ON employee
  3  DECLARE
  4     c number;
  5  BEGIN
  6     select AVG(wage) into c
  7     from employee;
  8    dbms_output.put_line('New Average wage of Employees: '|| c);
  9  END;
 10  /

Trigger created.
```



In this we can observe the line ‘New Average wage of Employees: 27.5’ in the output.

This is the result of the above trigger which was triggered due to the empsal_update procedure in the fdb_proj package.

3) accidental_log_order: This is a trigger on logistics table. This gets triggered before an insert operation on logistics table. When we want to record a new logistics order, this trigger checks whether the warehouse have enough stock of the drug in question to satisfy the order and warns us to get more stock if not present.

```
CREATE OR REPLACE TRIGGER accidental_log_order
BEFORE INSERT ON LOGISTICS
FOR EACH ROW
DECLARE
   q number;
   oq number;
        w warehouse.warehouseid%type;
        d drug.drugid%type;
BEGIN
   oq := :new.quantity;
   w := :new.warehouseid;
   d := :new.good;

   select SUM(currentstock) into q
   from inventory
   where buildingid = w and drugid = d;
```

```
    if q >= oq then
        dbms_output.put_line('Order Accepted.');
    else
        dbms_output.put_line('Order Cannot be Fulfilled ! Get More Stock.');
    end if;
END;
```

```
SQL> CREATE OR REPLACE TRIGGER accidental_log_order
  2  BEFORE INSERT ON LOGISTICS
  3  FOR EACH ROW
  4  DECLARE
  5      q number;
  6      oq number;
  7     w warehouse.warehouseid%type;
  8             d drug.drugid%type;
  9  BEGIN
 10      oq := :new.quantity;
 11      w := :new.warehouseid;
 12      d := :new.good;
 13
 14      select SUM(currentstock) into q
 15      from inventory
 16      where buildingid = w and drugid = d;
 17
 18      if q >= oq then
 19          dbms_output.put_line('Order Accepted.');
 20      else
 21          dbms_output.put_line('Order Cannot be Fulfilled ! Get More Stock.');
 22      end if;
 23  END;
 24  /

Trigger created.

SQL>
```

```
SQL> insert into logistics values('Dg5','W5','S5',TO_DATE('2023-03-06','yyyy/mm/dd'),'5000','Pending');
Order Cannot be Fulfilled ! Get More Stock.

1 row created.

SQL>
```

SUMMARY:

In this part of the project, we have created

5 Procedures:

- empsal_update
- total_hours
- avg_hw
- age_patients
- logistics_status

5 Functions:

- get_insamt()

- get_swl()

- get_stock()

- get_stockr()

-get_totalsalesd()

3 Triggers:

- num_of_emp

- avg_wage

- accidental_log_order

1 Package: Which contains two procedures emsal_update, total_hours and the two functions get_insamt(), get_swl() in it.

CSCE 5350 004

FUNDAMENTALS OF DATABASE SYSTEMS

PROJECT - PART 5

GROUP-8

Name	ID
Nitin Chakravarthy Chilukuri	11601099
Surya Vamsi Chintapalli	11645442
Prathyusharani Dumpala	11656342
Yamini Gollamudi	11642723
Lohitha Sai Bonthu	11611601

ATTRIBUTES:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Payroll: EID, Date, Hours Worked

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

Prescription: PrescriptionID, DoctorID, PatientID, Date_Prescribed

Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

IDENTIFIED FUNCTIONAL DEPENDENCIES:

Stores:

Store Id->Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses:

Warehouse Id-> Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region:

Region Code-> Region Name, Region Manager

Employees:

EID-> Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

SSN-> Name, Age, Gender

Type-> Wage

Payroll:

EID, Date->Hours Worked

Patients:

PID-> Name, Age, Gender, Ph NO, Address, SSN

SSN-> Name, Age, Gender

Insurance:

INM->Name of Insurance, PID, Amount, Date Claimed, Status

PID->INM

Prescription:

Prescription ID->Doctor ID, Patient ID, Date_Prescribed

Drug:

Drug ID->Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory:

Drug ID, Building ID->Current Stock

Logistics:

Good, Date, Warehouse ID, Store ID->Quantity, Status

Sales:

Month, Year, Drug ID, Store ID->No of sales per month

NORMALIZATION:

1. Stores:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Store Id->Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

NORMAL FORM:

Store Id+ = {Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code}

Therefore, Store Id is the candidate key.

Since there is only one FD in which the LHS contains the candidate key which is also a super key this table is in BCNF.

2. Warehouses:

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region

Code

Warehouse Id-> Address, Warehouse Manager, Current Stock, Capacity, Region Code

NORMAL FORM:

Warehouse Id+ = {Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code}

Therefore, Warehouse Id is the candidate key.

Since there is only one FD in which the LHS contains the candidate key which is also a super key this table is in BCNF.

3. Region:

Region: Region Name, Region Code, Region Manager

Region Code-> Region Name, Region Manager

NORMAL FORM:

Region Code+ = {Region Code, Region Name, Region Manager}

Region Name+ = {Region Name}

Region Manager+ = {Region Manager}

We have Region Code as the candidate key.

Since there is only one FD in which the LHS contains the candidate key which is also a super key this table is in BCNF.

4. Employees:

Employees: EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

EID-> Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

SSN-> Name, Age, Gender

Type-> Wage

EID+ = {EID, Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number}

SSN+ = {SSN, Name, Age, Gender}

Type+ = {Type, Wage}

NORMAL FORM:

We have EID as the candidate key.

Prime Attributes = {EID}

Non-Prime Attributes = {Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number}

-We know that the table is in 1NF as multiple values are not allowed.

-The table is in 2NF as there are no partial dependencies.

-The table is not in 3NF as we have (SSN-> Name, Age, Gender) and (Type-> Wage) where a non-prime attribute is dependent on another non-prime attribute.

Therefore, the highest normal form of Employees table is 2NF.

NORMALIZING TO BCNF:

Divide Employees table into R1, R2 such that:

R1(EID, Name, SSN, Age, Gender, Address, Ph NO, Type, Location, Bank Account Number)

R2(Type, Wage).

FD corresponding to R2: Type-> Wage

In R2 ‘Type’ is the candidate key from (Type-> Wage) and all FD’s corresponding to R2 have a super key in the LHS. So R2 is in BCNF.

Now Decompose R1 into R3, R4 such that:

R3(SSN, Name, Age, Gender)

R4(EID, SSN, Address, Ph NO, Type, Location, Bank Account Number)

FD corresponding to R3: SSN-> Name, Age, Gender

In R3 ‘SSN’ is the candidate key from (SSN-> Name, Age, Gender) and all FD’s corresponding to R3 have a super key in the LHS. So R3 is in BCNF.

FD corresponding to R4: EID-> SSN, Address, Ph NO, Type, Location, Bank Account Number.

This FD is decomposed from (EID-> Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number) using Armstrong’s Axioms Decomposition Rule.

In R4 ‘EID’ is the candidate key and all FD’s corresponding to R4 have a super key in the LHS. So R4 is in BCNF.

Dependency Preserving Check:

We have (Type-> Wage) preserved in R2 and (SSN-> Name, Age, Gender) is preserved in R3.

We need to check for EID-> Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

This can done by computing w = w U ((w ∩ Ri)+ ∩ Ri) and check whether w gives the FD we are checking for or not.

Initially w = EID

Iteration 1:

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R2:

w = EID U ((EID ∩ (Type, Wage))+ ∩ (Type, Wage)) = EID

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R3:

w = EID U ((EID ∩ (SSN, Name, Age, Gender))+ ∩ (SSN, Name, Age, Gender))

w = EID

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R4:

w = EID U ((EID ∩ (EID, SSN, Address, Ph NO, Type, Location, Bank Account Number))+ ∩ (EID, SSN, Address, Ph NO, Type, Location, Bank Account Number))

w = (EID, SSN, Address, Ph NO, Type, Location, Bank Account Number)

Iteration 2:

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R2:

w = (EID, SSN, Address, Ph NO, Type, Location, Bank Account Number) U (((EID, SSN, Address, Ph NO, Type, Location, Bank Account Number) ∩ (Type, Wage))+ ∩ (Type, Wage))

w = (EID, SSN, Address, Ph NO, Type, Location, Bank Account Number, Wage)

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R3:

w = (EID, SSN, Address, Ph NO, Type, Location, Bank Account Number, Wage) U (((EID, SSN, Address, Ph NO, Type, Location, Bank Account Number, Wage) ∩ (SSN, Name, Age, Gender))+ ∩ (SSN, Name, Age, Gender))

w = (EID, SSN, Address, Ph NO, Type, Location, Bank Account Number, Wage, Name, Age, Gender)

We can obtain the FD EID-> Name, SSN, Age, Gender, Address, Ph NO, Wage, Type, Location, Bank Account Number

Hence Dependency is Preserved and converted into BCNF.

5. Payroll

Payroll: EID, Date, Hours Worked

EID, Date->Hours Worked

NORMAL FORM:

EID+ = {EID}

Date+ = {Date}

Hours Worked+ = {Hours Worked}

(EID, Date)+ = { EID, Date, Hours Worked}

We have (EID, Date) as the candidate key.

Since there is only one FD and it contains candidate key in its LHS which is also a super key Payroll is in BCNF.

6. Patients

Patients: PID, Name, Age, Gender, Ph NO, Address, SSN

PID-> Name, Age, Gender, Ph NO, Address, SSN

SSN-> Name, Age, Gender

NORMAL FORM:

PID+ = {Name, Age, Gender, Ph NO, Address, SSN}

SSN+ = {SSN, Name, Age, Gender}

Therefore, PID is the candidate keys.

Prime Attributes = {PID}

Non-Prime Attributes = {Name, Age, Gender, Ph NO, Address, SSN}

-We know that the table is in 1NF as multiple values are not allowed.

- It is in 2NF as there are no partial dependencies.

- It is not in 3NF as there are transitive dependencies.

NORMALIZING TO BCNF:

Divide the Patients Table into R1, R2 such that:

R1(SSN, Name, Age, Gender)

R2(PID, SSN, Ph NO, Address)

FD corresponding to R1: SSN-> Name, Age, Gender

In R1 ‘SSN’ is the candidate key from (SSN-> Name, Age, Gender) and all FD’s corresponding to R1 have a super key in the LHS. So R2 is in BCNF.

FD corresponding to R2: PID-> SSN, Ph NO, Address

This FD is decomposed from (PID-> Name, Age, Gender, Ph NO, Address, SSN) using Armstrong's Axioms Decomposition Rule.

In R2 'PID' is the candidate key and all FD's corresponding to R2 have a super key in the LHS. So R2 is in BCNF.

Dependency Preserving Check: We have SSN-> Name, Age, Gender preserved in R1. We need to check for PID-> Name, Age, Gender, Ph NO, Address, SSN

This can done by computing w = w U ((w ∩ Ri)+ ∩ Ri) and check whether w gives the FD we are checking for or not.

Initially w = PID

Iteration 1:

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R1:

w = PID U ((PID ∩ (SSN,Name,Age,Gender))+ ∩ (SSN,Name,Age,Gender)) = PID

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R2:

w = PID U ((PID ∩ (PID, SSN, Ph NO, Address))+ ∩ (PID, SSN, Ph NO, Address))

w = (PID, SSN, Ph NO, Address)

Iteration 2:

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R1:

w = (PID, SSN, Ph NO, Address) U (((PID, SSN, Ph NO, Address) ∩ (SSN,Name,Age,Gender))+ ∩ (SSN,Name,Age,Gender))

w = (PID, Name, Age, Gender, Ph NO, Address, SSN)

Since we can obtain PID-> Name, Age, Gender, Ph NO, Address, SSN, the decomposition of R into R1, R2 is in BCNF and Dependency Preserving.

7. Insurance

Insurance: INM, Name of Insurance, PID, Amount, Date Claimed, Status

INM->Name of Insurance, PID, Amount, Date Claimed, Status

PID->INM

NORMAL FORM:

INM+ = {INM, Name of Insurance, PID, Amount, Date Claimed, Status}

PID+ = {PID, INM, Name of Insurance, Amount, Date Claimed, Status}

Therefore, INM and PID are the candidate keys.

Prime Attributes = {INM, PID}

Non-Prime Attributes = {Name of Insurance, Amount, Date Claimed, Status}

-We know that the table is in 1NF as multiple values are not allowed.

-The table is not in 2NF as we have PID -> INM which is a partial dependency.

Therefore, the highest normal form of Employees table is 1NF.

NORMALIZING TO BCNF:

Divide the Insurance Table into R1, R2 such that:

R1(INM, Name of Insurance, Amount, Date Claimed, Status)

R2(PID, INM)

FD corresponding to R2: PID-> INM

In R2 'PID' is the candidate key from (PID-> INM) and all FD's corresponding to R2 have a super key in the LHS. So R2 is in BCNF.

FD corresponding to R1: INM-> Name of Insurance, Amount, Date Claimed, Status

This FD is decomposed from (INM->Name of Insurance, PID, Amount, Date Claimed, Status) using Armstrong's Axioms Decomposition Rule.

In R1 'INM' is the candidate key and all FD's corresponding to R1 have a super key in the LHS. So R1 is in BCNF.

Dependency Preserving Check: We have (PID-> INM) preserved in R2.

We need to check for INM->Name of Insurance, PID, Amount, Date Claimed, Status

This can done by computing w = w U ((w ∩ Ri)+ ∩ Ri) and check whether w gives the FD we are checking for or not.

Initially w = INM

Iteration 1:

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R2:

w = INM U ((INM ∩ (INM,PID))+ ∩ (INM,PID)) = (INM, PID)

w = w U ((w ∩ Ri)+ ∩ Ri) for Ri = R1:

w = (INM, PID) U (((INM, PID) ∩ (INM, Name of Insurance, Amount, Date Claimed, Status))+ ∩ (INM, Name of Insurance, Amount, Date Claimed, Status))

w = (INM, PID, Name of Insurance, Amount, Date Claimed, Status)

Hence Dependency is Preserved and converted into BCNF.

8. Prescription

Precsription: PrescriptionID, DoctorID, PatientID, Date_Prescribed

Prescription ID->Doctor ID, Patient ID, Date_Prescribed

NORMAL FORM:

Prescription ID+ = {PrescriptionID, DoctorID, PatientID, Date_Prescribed}

We have (Prescription ID) as the candidate key.

Since there is only one FD and it contains candidate key in its LHS which is also a super key Payroll is in BCNF.

9. Drug

Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Drug ID->Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

NORMAL FORM:

Drug ID+ = {Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date}

We have (Drug ID) as the candidate key.

Since there is only one FD and it contains candidate key in its LHS which is also a super key Payroll is in BCNF.

10. Inventory

Inventory: Drug ID, Building ID, Current Stock

Drug ID, Building ID->Current Stock

NORMAL FORM:

Drug ID+ = {Drug ID}

Building ID+ = {Building ID}

Current Stock+ = {Current Stock}

(Drug ID, Building ID)+ = {Drug ID, Building ID, Current Stock}

We have (Drug ID, Building ID) as the candidate key.

Since there is only one FD and it contains candidate key in its LHS which is also a super key Payroll is in BCNF.

11. Logistics

Logistics: Good, Date, Warehouse ID, Store ID, Quantity, Status

Good, Date, Warehouse ID, Store ID->Quantity, Status

NORMAL FORM:

We have (Good, Date, Warehouse ID, Store ID) as the candidate key.

Since there is only one FD and it contains candidate key in its LHS which is also a super key Payroll is in BCNF.

12. Sales

Sales: No of sales per month, Month, Year, Drug ID, Store ID

Month, Year, Drug ID, Store ID->No of sales per month

NORMAL FORM:

We have (Month, Year, Drug ID, Store ID) as the candidate key.

Since there is only one FD and it contains candidate key in its LHS which is also a super key Payroll is in BCNF.

UPDATED TABLES & ENTITIES:

Stores: Store Id, Address, Manager, Assigned Pharmacist, Assigned Doctor, Region Code

Warehouses: Warehouse Id, Address, Warehouse Manager, Current Stock, Capacity, Region Code

Region: Region Name, Region Code, Region Manager

Employees: EID, SSN, Address, Ph NO, Type, Location, Bank Account Number

Employee SSN: SSN, Name, Age, Gender

Employee Type: Type, Wage

Patients: PID, Ph NO, Address, SSN

Patients SSN: SSN**,** Name, Age, Gender

Patients Insurance: PID, INM

Insurance Details: INM, Name of Insurance, Amount, Date Claimed, Status

Prescription: PrescriptionID, DoctorID, PatientID, Date_Prescribed

Drug: Drug ID, Name, Price, Drug Type, Dosage, Manf By, Manf Date, Batch NO, Expiry Date

Inventory: Drug ID, Building ID, Current Stock

Logistics: Good, Date, Warehouse ID, Store ID, Quantity, Status

Sales: No of sales per month, Month, Year, Drug ID, Store ID

The New tables are those related to Employees, Patients, and Insurance.

- The Employee Personal Information like Name, Age, Gender is now stored in a new table Employee SSN as we can get the info using SSN.

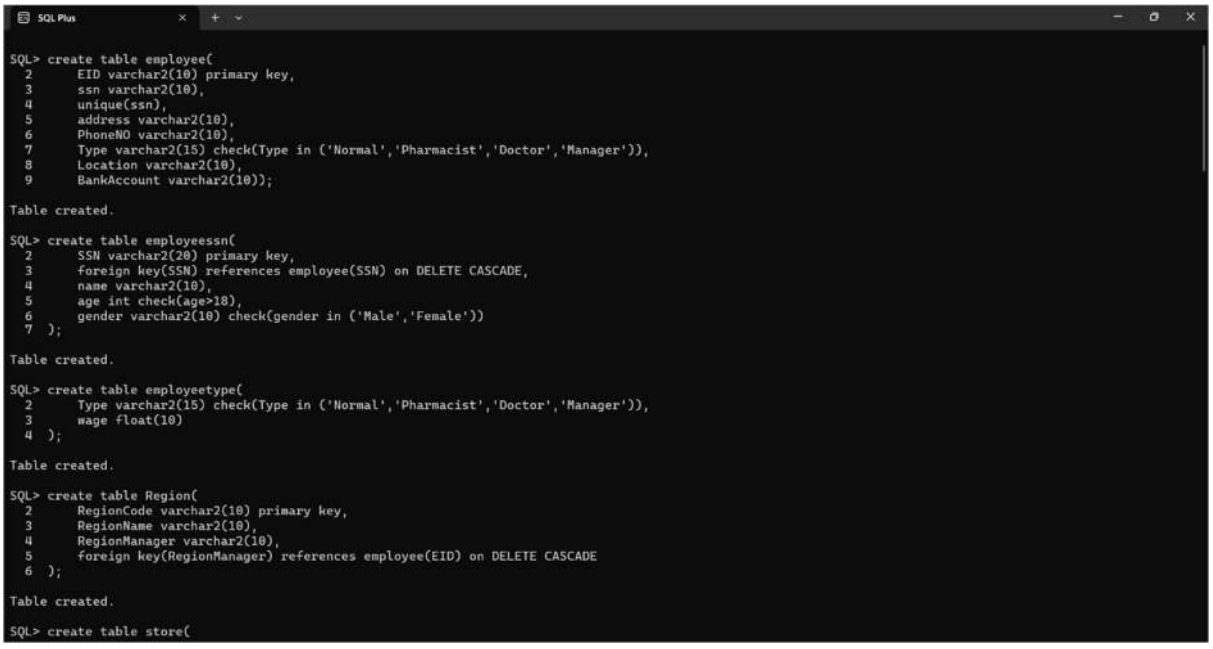
- The Wage of employee which depends on his type is also now stored in a separate table named Employee Type.

- Similar to employee table the personal information of patients is now stored in a separate table named Patients SSN.

- The Insurance table is split into two tables. One contains the Insurance Number and the corresponding Patient ID, whereas the other contains the details of the Insurance.

CREATION OF NEW TABLES:

Creation of Employee, Employees SSN, Employee Type and Region Tables:



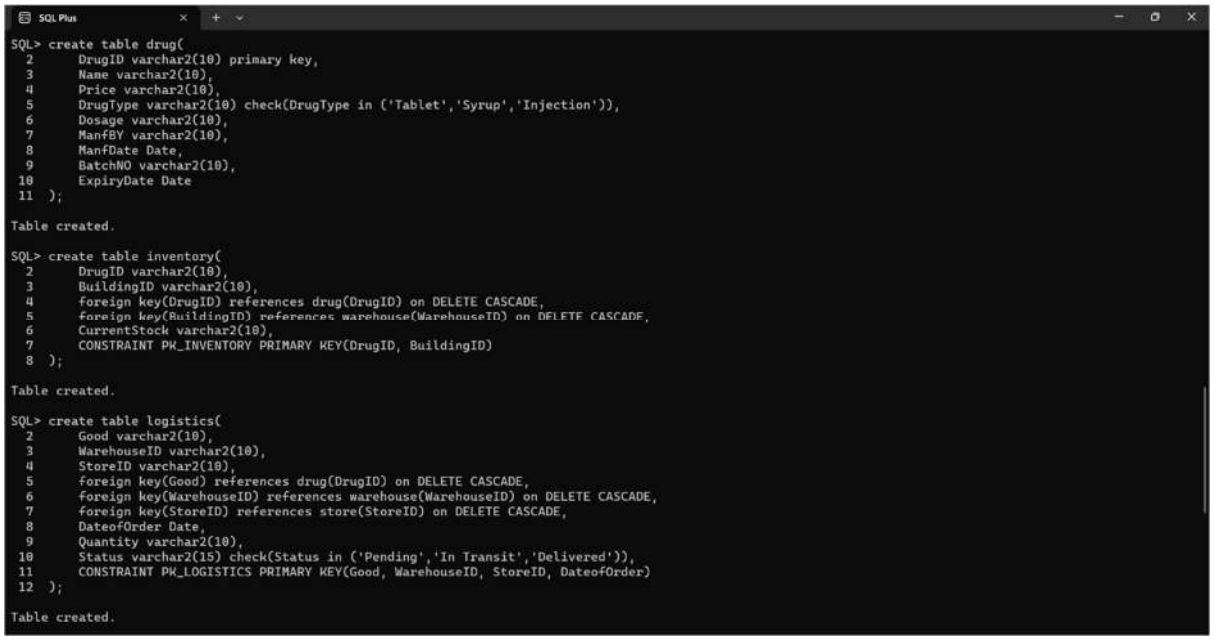
Creation of Stores, Warehouse and Patients Table:

```
SQL> create table store(
  2      StoreID varchar2(10) primary key,
  3      address varchar2(10),
  4      ManagerID varchar2(10),
  5      PharmacistID varchar2(10),
  6      foreign key(PharmacistID) references employee(EID) on DELETE CASCADE,
  7      DoctorID varchar2(10),
  8      foreign key(DoctorID) references employee(EID) on DELETE CASCADE,
  9      Region varchar2(10),
 10      foreign key(Region) references region(RegionCode) on DELETE CASCADE
 11  );

Table created.

SQL> create table warehouse(
  2      WarehouseID varchar2(10) primary key,
  3      address varchar2(10),
  4      ManagerID varchar2(10),
  5      foreign key(ManagerID) references employee(EID) on DELETE CASCADE,
  6      CurrentStock varchar2(10),
  7      Capacity varchar2(10),
  8      Region varchar2(10),
  9      foreign key(Region) references region(RegionCode) on DELETE CASCADE
 10  );

Table created.

SQL> create table Patients(
  2      PID varchar2(10) primary key,
  3      SSN varchar2(10),
  4      unique(SSN),
  5      PhoneNO varchar2(10),
  6      Address varchar2(10)
  7  );

Table created.
```

Creation of Patients SSN, Patients INS and Insurance Details Tables:

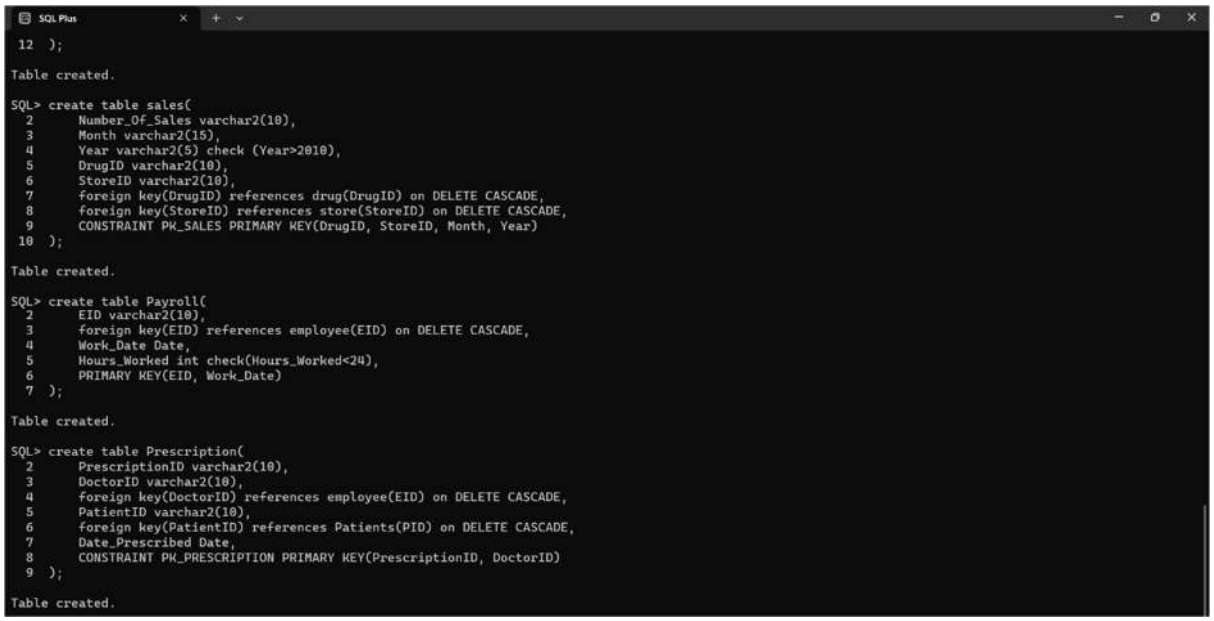
```
SQL> create table PatientSSN(
  2      SSN varchar2(10) primary key,
  3      foreign key(SSN) references Patients(SSN) on DELETE CASCADE,
  4      Age int,
  5      Gender varchar2(10) check(Gender in ('Male','Female'))
  6  );

Table created.

SQL> create table PatientsINS(
  2      InsuranceNUM varchar2(10),
  3      PID varchar2(10),
  4      CONSTRAINT PK_INSURANCE PRIMARY KEY(InsuranceNUM, PID)
  5  );

Table created.

SQL>
SQL> create table INSDetails(
  2      InsuranceNUM varchar2(10) primary key,
  3      Name varchar2(10),
  4      Amount varchar2(10),
  5      DateClaimed Date,
  6      Status varchar2(10) check(Status in ('Pending','Approved','Failed'))
  7  );

Table created.
```

Creation of Drug, Inventory and Logistics Tables:



```
SQL Plus
SQL> create table drug(
  2      DrugID varchar2(10) primary key,
  3      Name varchar2(10),
  4      Price varchar2(10),
  5      DrugType varchar2(10) check(DrugType in ('Tablet','Syrup','Injection')),
  6      Dosage varchar2(10),
  7      ManfBY varchar2(10),
  8      ManfDate Date,
  9      BatchNO varchar2(10),
 10      ExpiryDate Date
 11  );

Table created.

SQL> create table inventory(
  2      DrugID varchar2(10),
  3      BuildingID varchar2(10),
  4      foreign key(DrugID) references drug(DrugID) on DELETE CASCADE,
  5      foreign key(BuildingID) references warehouse(WarehouseID) on DELETE CASCADE,
  6      CurrentStock varchar2(10),
  7      CONSTRAINT PK_INVENTORY PRIMARY KEY(DrugID, BuildingID)
  8  );

Table created.

SQL> create table logistics(
  2      Good varchar2(10),
  3      WarehouseID varchar2(10),
  4      StoreID varchar2(10),
  5      foreign key(Good) references drug(DrugID) on DELETE CASCADE,
  6      foreign key(WarehouseID) references warehouse(WarehouseID) on DELETE CASCADE,
  7      foreign key(StoreID) references store(StoreID) on DELETE CASCADE,
  8      DateofOrder Date,
  9      Quantity varchar2(10),
 10      Status varchar2(15) check(Status in ('Pending','In Transit','Delivered')),
 11      CONSTRAINT PK_LOGISTICS PRIMARY KEY(Good, WarehouseID, StoreID, DateofOrder)
 12  );

Table created.
```

Creation of Sales, Payroll and Prescription Tables:



```
SQL Plus
 12  );

Table created.

SQL> create table sales(
  2      Number_Of_Sales varchar2(10),
  3      Month varchar2(15),
  4      Year varchar2(5) check (Year>2010),
  5      DrugID varchar2(10),
  6      StoreID varchar2(10),
  7      foreign key(DrugID) references drug(DrugID) on DELETE CASCADE,
  8      foreign key(StoreID) references store(StoreID) on DELETE CASCADE,
  9      CONSTRAINT PK_SALES PRIMARY KEY(DrugID, StoreID, Month, Year)
 10  );

Table created.

SQL> create table Payroll(
  2      EID varchar2(10),
  3      foreign key(EID) references employee(EID) on DELETE CASCADE,
  4      Work_Date Date,
  5      Hours_Worked int check(Hours_Worked<24),
  6      PRIMARY KEY(EID, Work_Date)
  7  );

Table created.

SQL> create table Prescription(
  2      PrescriptionID varchar2(10),
  3      DoctorID varchar2(10),
  4      foreign key(DoctorID) references employee(EID) on DELETE CASCADE,
  5      PatientID varchar2(10),
  6      foreign key(PatientID) references Patients(PID) on DELETE CASCADE,
  7      Date_Prescribed Date,
  8      CONSTRAINT PK_PRESCRIPTION PRIMARY KEY(PrescriptionID, DoctorID)
  9  );

Table created.
```