PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013) 100 Feet Ring Road, BSK III Stage, Bengaluru - 560085

Department of Computer Science and Engineering Session: Aug-Dec 2021

SEMESTER-5

DATE:21-11-2021

DBMS PROJECT ASSIGENMENT-3

PROJECT TITLE

PHARMACY MANAGEMENT SYSTEM

TEAM MEMBERS (SECTION B)

Basanagouda(PES2UG19CS082)

Bhagyashree Shankar (PES2UG19CS085)

Bhavana R(PES2UG19CS089)

LANGUAGE CHOICE

We chose python for designing the frontend part and postgresql as our backend server database.

Python is considered one of the best programming languages for web development because it's relatively easy to understand and it has a huge array of tools and functionalities. It's also extremely scalable and can carry out a wide range of outcomes.

We chose Django framework for the front end. Django is a Python-based free and open-source web framework

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself.[9] Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Hence we chose python and windows forms based front end as it can be built in hassle free. As mentioned above this would have not been possible without Visual Studio tool.
The visual studio app and python worked hand in hand to help us build a front end with lesser efforts.

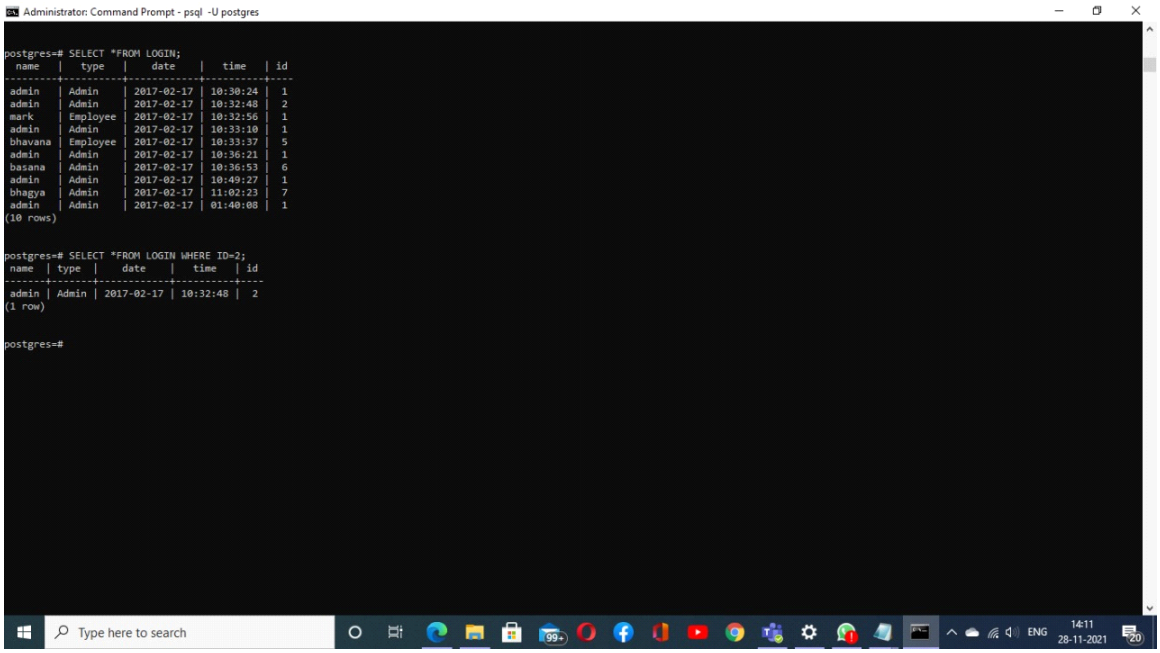
The database located in SQL Server within Visual studio could be connected easily without any difficulties.

ADDITIONAL QUERIES

Administrator

Query to list administrator with the entered Name and Password

```
SELECT *FROM LOGIN
SELECT *FROM LOGIN WHERE ID=2;
```

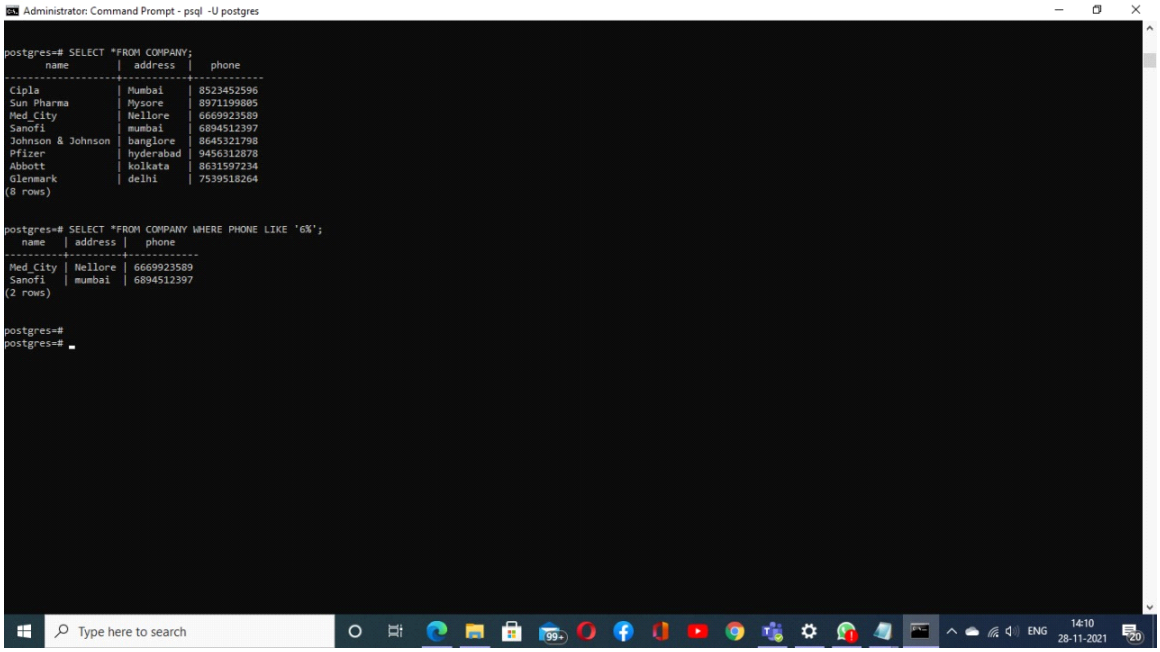


Company Details

Query to list the details of the comany where phone no starts from 6

SELECT *FROM COMPANY;

SELECT *FROM COMPANY WHERE PHONE LIKE '6%';

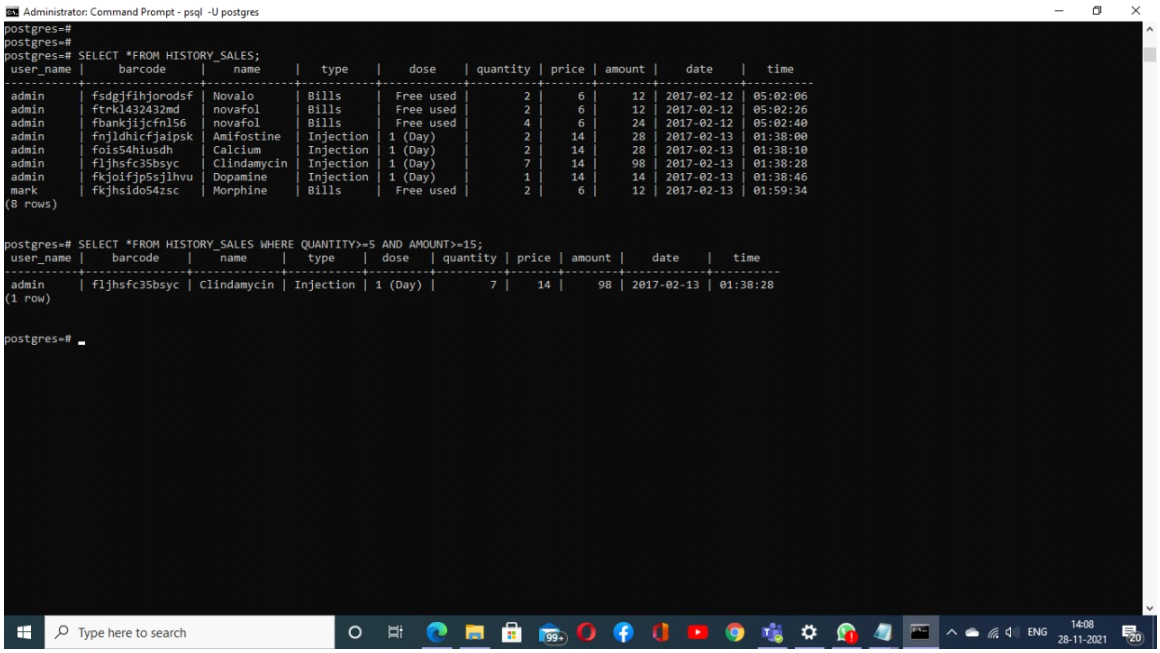


Sales History

Query to list the details of sales where quantity of the drug sales is greater than 5 and drug mrp is greater than 15

1)SELECT *FROM HISTORY_SALES;

SELECT *FROM HISTORY_SALES WHERE QUANTITY>=5 AND AMOUNT>=15;

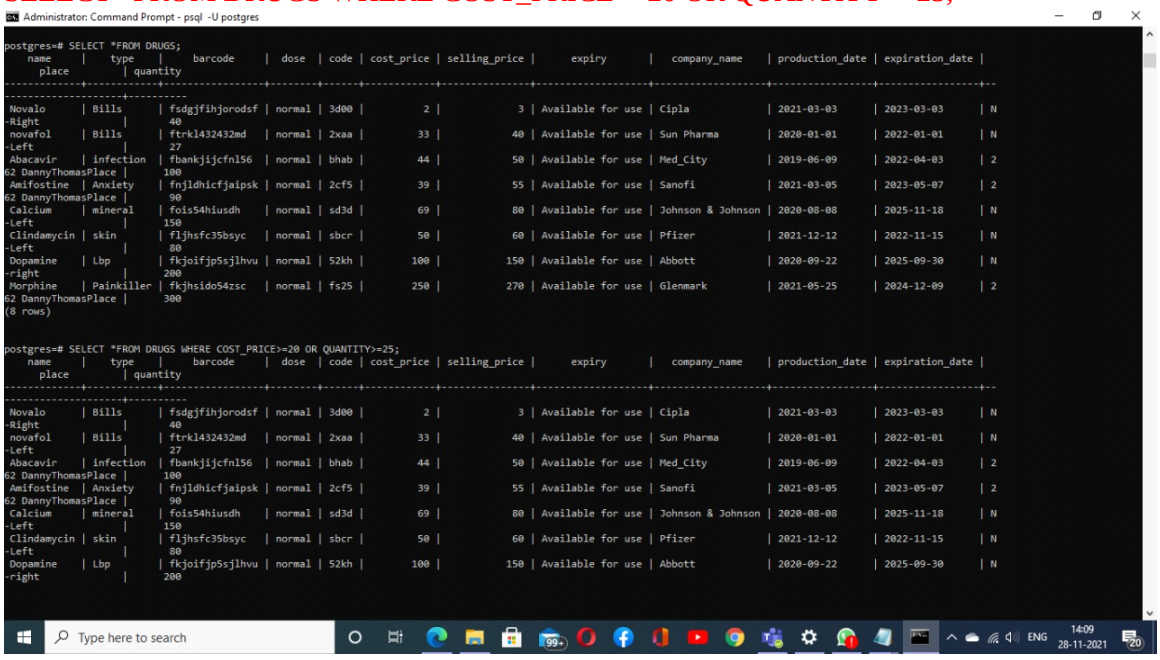


Drug Details

Query to list the details of drugs where cost of the drug is greater than 20 and quantity is greater than 15

2)SELECT *FROM DRUGS;

SELECT *FROM DRUGS WHERE COST_PRICE>=20 OR QUANTITY>=25;

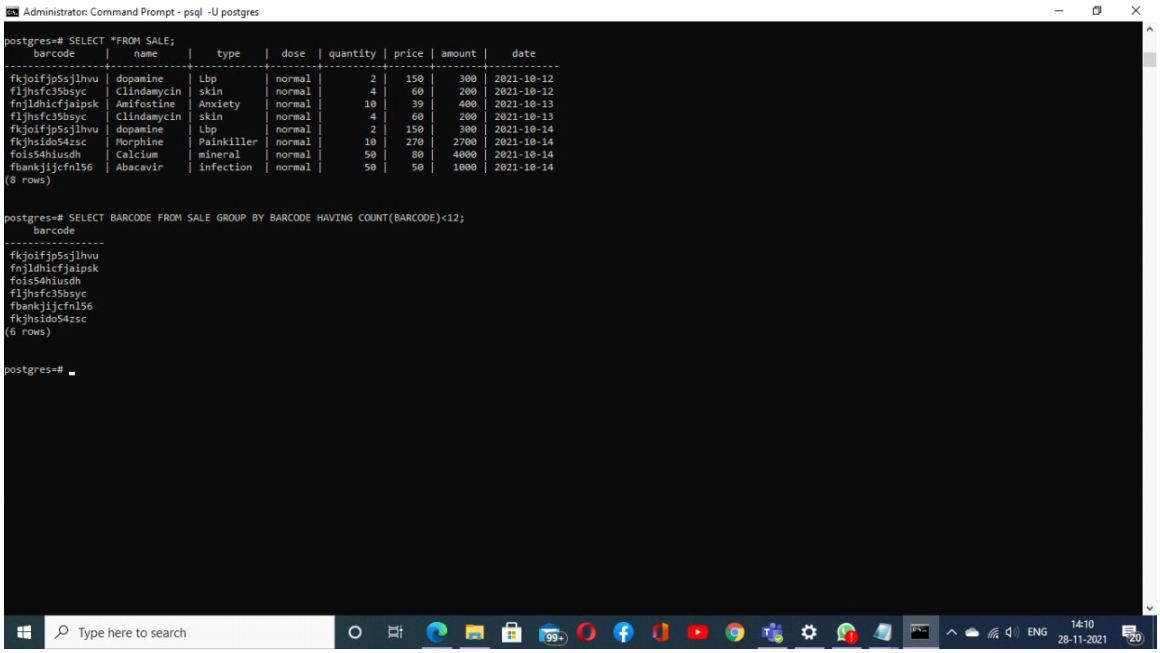


Sales Details

Query to list the details of barcode from sales and count the barcode

3)SELECT *FROM SALE;

SELECT BARCODE FROM SALE GROUP BY BARCODE HAVING COUNT(BARCODE)<12;

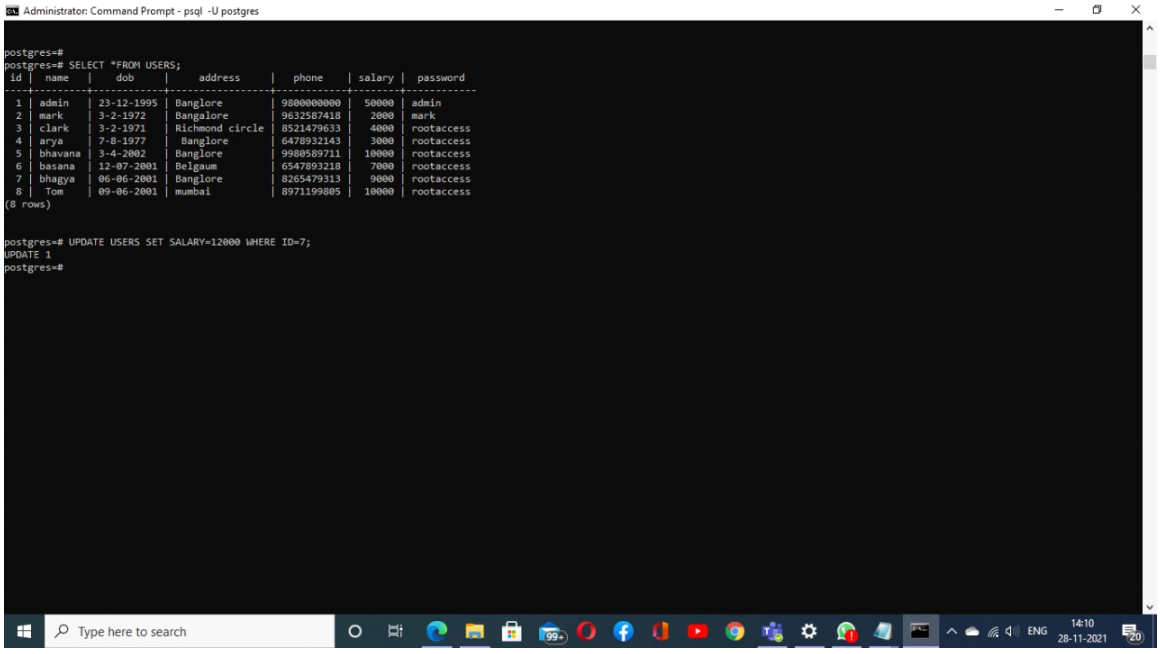


Company User Details

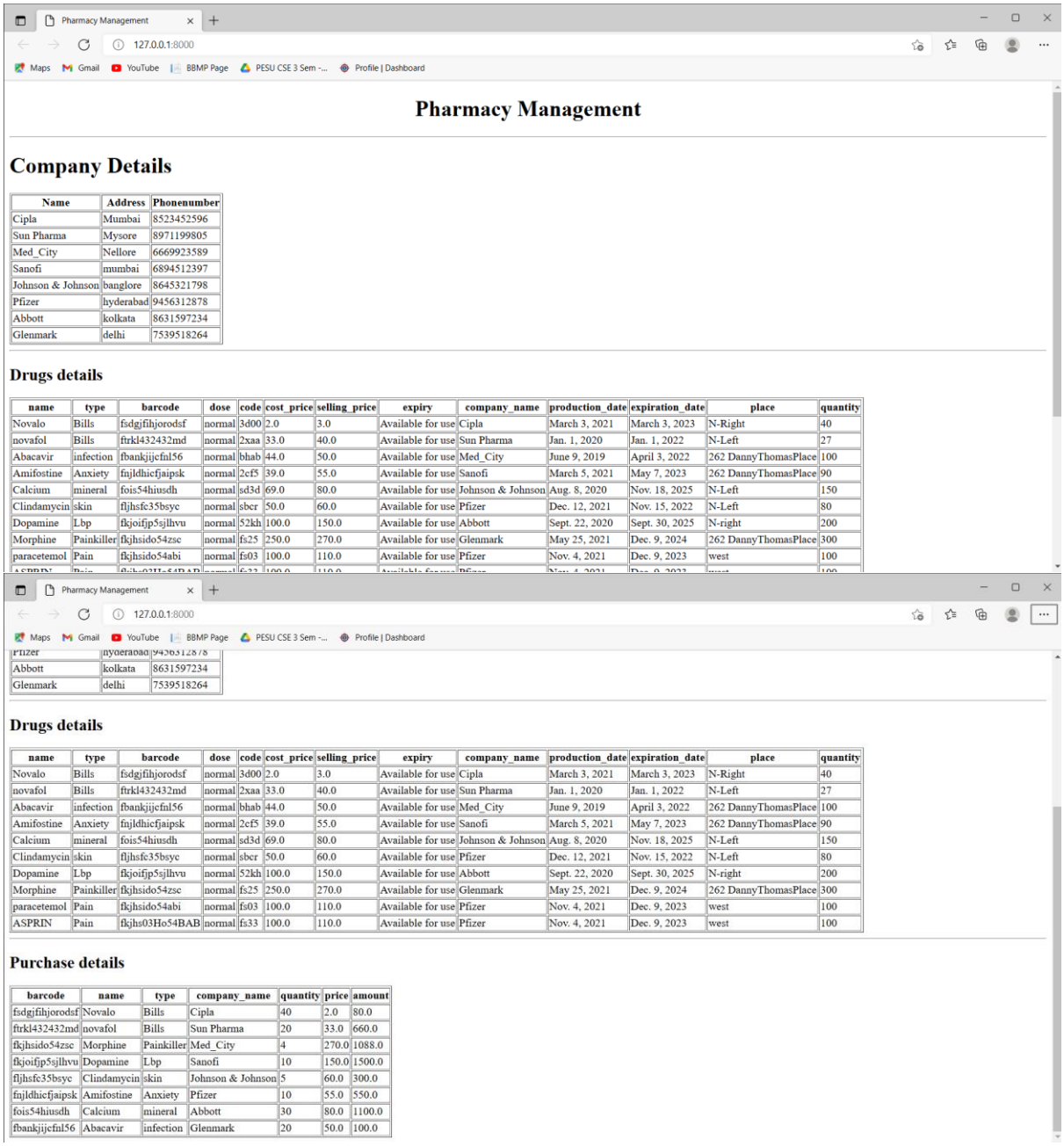
Query to update the salary to 12000 whose id is 7

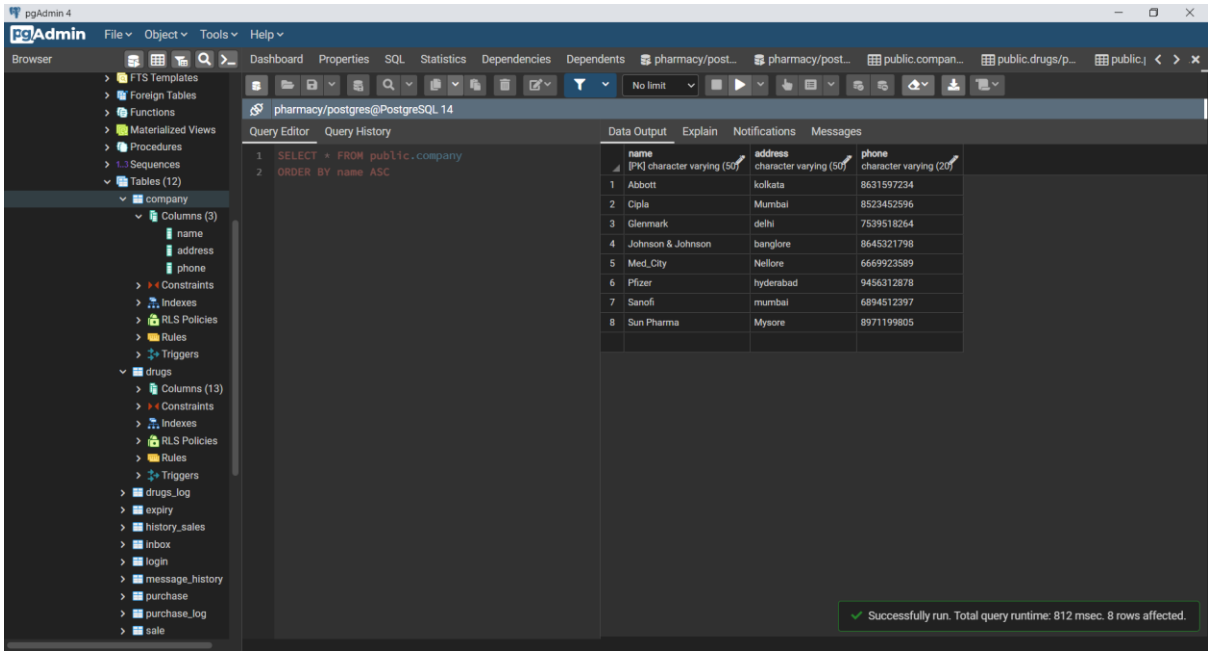
5)SELECT *FROM USERS;

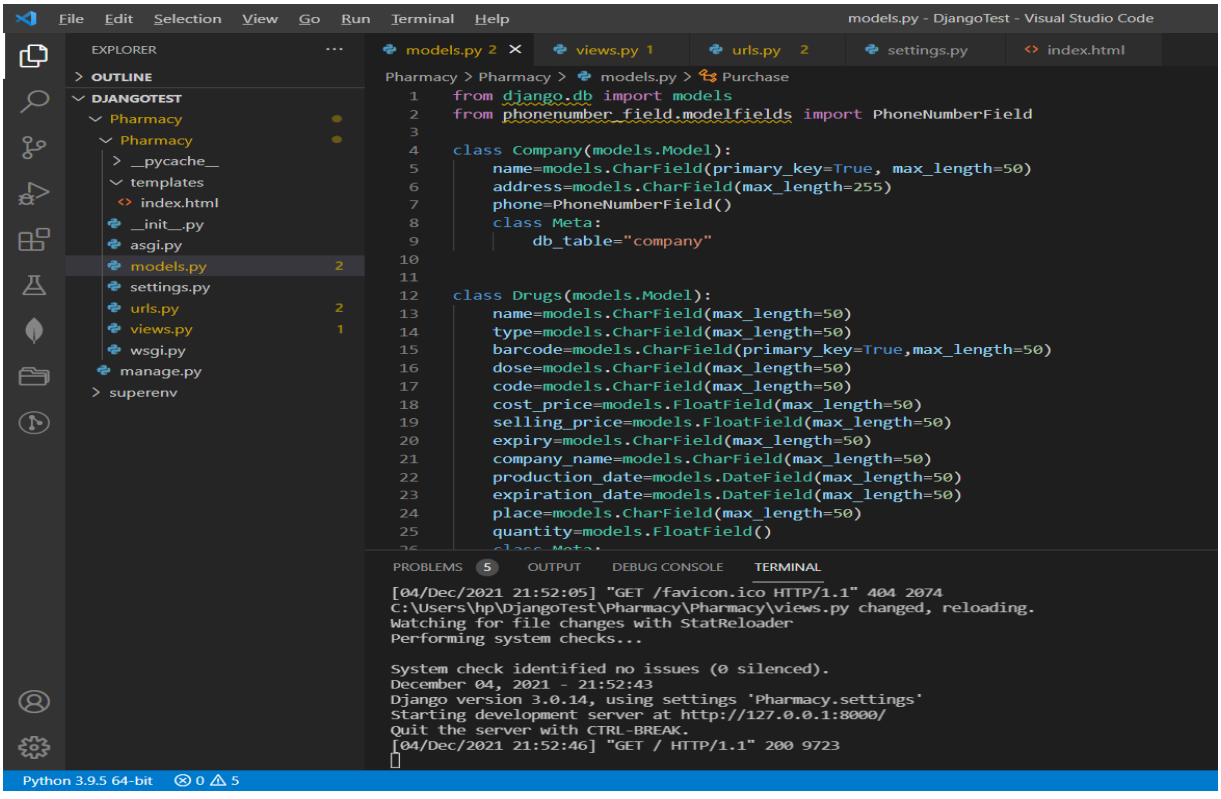
UPDATE USERS SET SALARY=12000 WHERE ID=7;

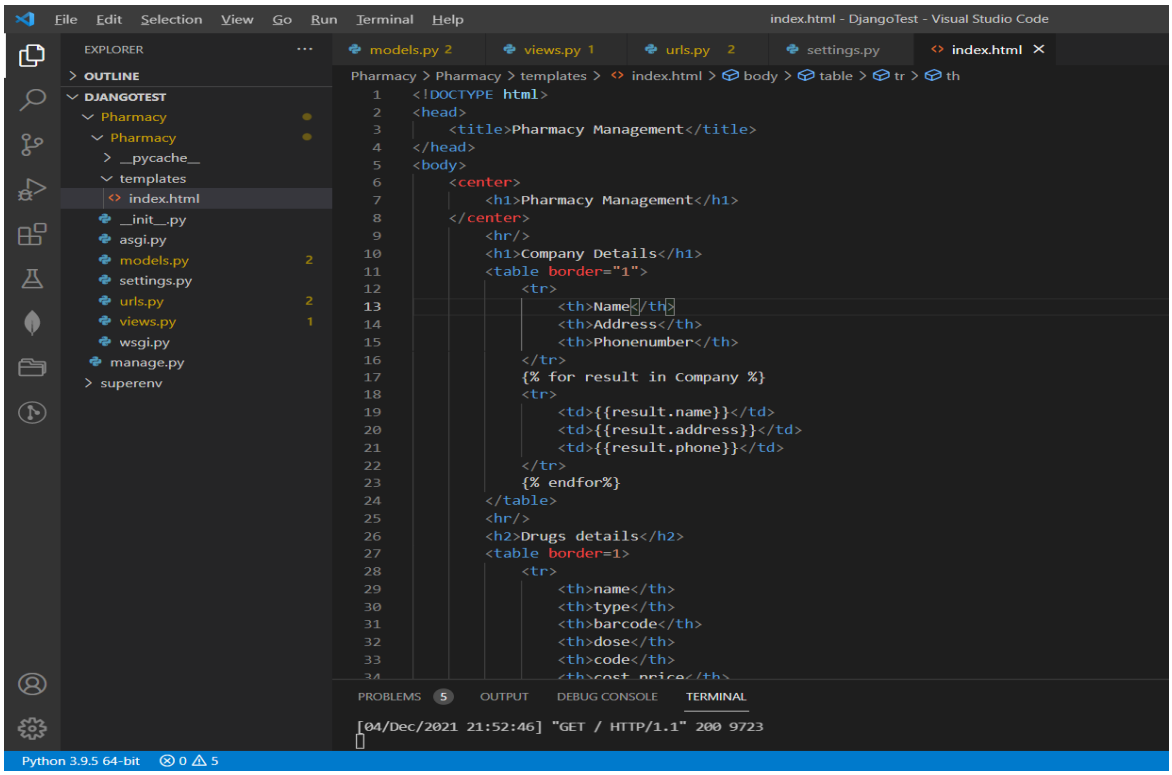


Screenshots of Visual Studio Code(Front End)



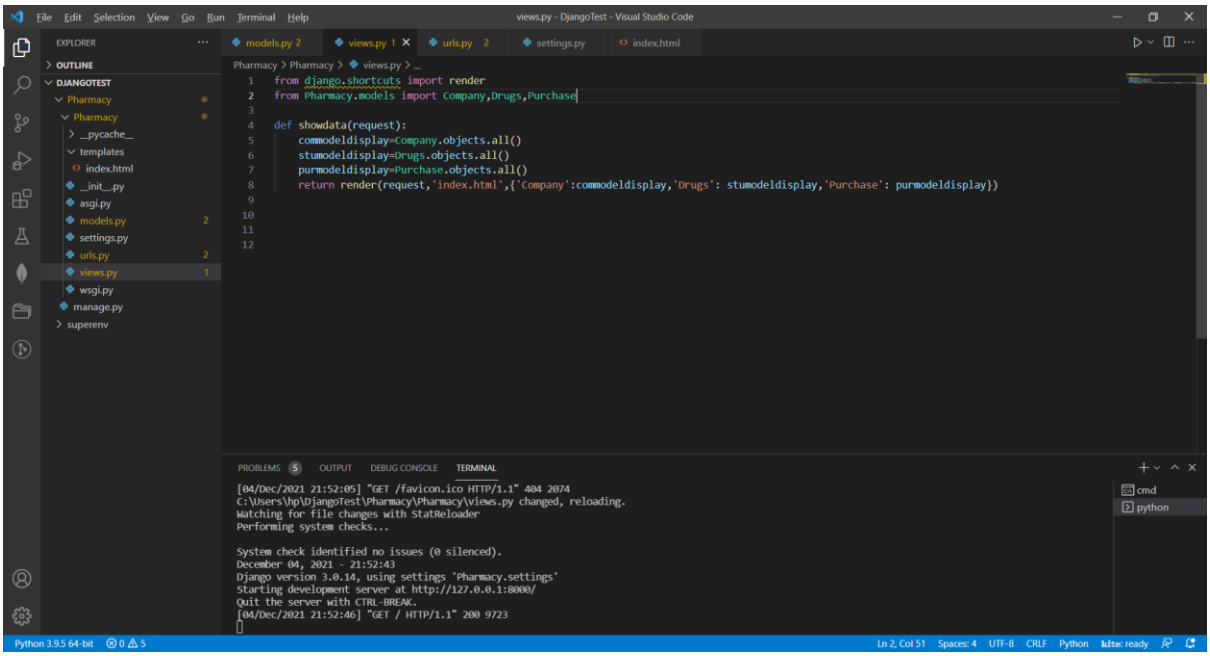

pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
pharmacy/post...
pharmacy/post...
public.compan...
public.drugs/p...
public.
No limit
pharmacy/postgres@PostgreSQL 14
Query Editor
Query History
SELECT * FROM public.company
ORDER BY name ASC
Data Output
Explain
Notifications
Messages
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (12)
company
Columns (3)
name
address
phone
Constraints
Indexes
RLS Policies
Rules
Triggers
drugs
Columns (13)
Constraints
Indexes
RLS Policies
Rules
Triggers
drugs_log
expiry
history_sales
inbox
login
message_history
purchase
purchase_log
sale
name [PK] character varying (50)
address character varying (50)
phone character varying (20)
1 Abbott kolkata 8631597234
2 Cipla Mumbai 8523452596
3 Glenmark delhi 7539518264
4 Johnson & Johnson banglore 8645321798
5 Med_City Nellore 6669923589
6 Pfizer hyderabad 9456312878
7 Sanofi mumbai 6894512397
8 Sun Pharma Mysore 8971199805
Successfully run. Total query runtime: 812 msec. 8 rows affected.


File
Edit
Selection
View
Go
Run
Terminal
Help
models.py - DjangoTest - Visual Studio Code
EXPLORER
OUTLINE
DJANGOTEST
Pharmacy
Pharmacy
__pycache__
templates
index.html
__init__.py
asgi.py
models.py 2
settings.py
urls.py 2
views.py 1
wsgi.py
manage.py
superenv
models.py 2
views.py 1
urls.py 2
settings.py
index.html
Pharmacy > Pharmacy > models.py > Purchase
from django.db import models
from phonenumber_field.modelfields import PhoneNumberField
class Company(models.Model):
name=models.CharField(primary_key=True, max_length=50)
address=models.CharField(max_length=255)
phone=PhoneNumberField()
class Meta:
db_table="company"
class Drugs(models.Model):
name=models.CharField(max_length=50)
type=models.CharField(max_length=50)
barcode=models.CharField(primary_key=True,max_length=50)
dose=models.CharField(max_length=50)
code=models.CharField(max_length=50)
cost_price=models.FloatField(max_length=50)
selling_price=models.FloatField(max_length=50)
expiry=models.CharField(max_length=50)
company_name=models.CharField(max_length=50)
production_date=models.DateField(max_length=50)
expiration_date=models.DateField(max_length=50)
place=models.CharField(max_length=50)
quantity=models.FloatField()
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
[04/Dec/2021 21:52:05] "GET /favicon.ico HTTP/1.1" 404 2074
C:\Users\hp\DjangoTest\Pharmacy\Pharmacy\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
December 04, 2021 - 21:52:43
Django version 3.0.14, using settings 'Pharmacy.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[04/Dec/2021 21:52:46] "GET / HTTP/1.1" 200 9723
Python 3.9.5 64-bit
0
5


File Edit Selection View Go Run Terminal Help
index.html - DjangoTest - Visual Studio Code
EXPLORER
OUTLINE
DJANGOTEST
Pharmacy
Pharmacy
__pycache__
templates
index.html
__init__.py
asgi.py
models.py
settings.py
urls.py
views.py
wsgi.py
manage.py
superenv
models.py 2
views.py 1
urls.py 2
settings.py
index.html
Pharmacy > Pharmacy > templates > index.html > body > table > tr > th
<!DOCTYPE html>
<head>
<title>Pharmacy Management</title>
</head>
<body>

<h1>Pharmacy Management</h1>

<hr/>
<h1>Company Details</h1>
<table border="1">
<tr>
<th>Name</th>
<th>Address</th>
<th>Phonenumber</th>
</tr>
{% for result in Company %}
<tr>
<td>{{result.name}}</td>
<td>{{result.address}}</td>
<td>{{result.phone}}</td>
</tr>
{% endfor%}
</table>
<hr/>
<h2>Drugs details</h2>
<table border=1>
<tr>
<th>name</th>
<th>type</th>
<th>barcode</th>
<th>dose</th>
<th>code</th>
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
[04/Dec/2021 21:52:46] "GET / HTTP/1.1" 200 9723
Python 3.9.5 64-bit
0 5


File Edit Selection View Go Run Terminal Help
views.py - DjangoTest - Visual Studio Code
EXPLORER
OUTLINE
DJANGOTEST
Pharmacy
Pharmacy
__pycache__
templates
index.html
__init__.py
asgi.py
models.py
settings.py
urls.py
views.py
wsgi.py
manage.py
superenv
models.py 2
views.py 1
urls.py 2
settings.py
index.html
Pharmacy > Pharmacy > views.py > ...
from django.shortcuts import render
from Pharmacy.models import Company,Drugs,Purchase
def showdata(request):
commodeldisplay=Company.objects.all()
stumodeldisplay=Drugs.objects.all()
purmodeldisplay=Purchase.objects.all()
return render(request,'index.html',{'Company':commodeldisplay,'Drugs': stumodeldisplay,'Purchase': purmodeldisplay})
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL
[04/Dec/2021 21:52:05] "GET /favicon.ico HTTP/1.1" 404 2074
C:\Users\hp\DjangoTest\Pharmacy\Pharmacy\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
December 04, 2021 - 21:52:43
Django version 3.0.14, using settings 'Pharmacy.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[04/Dec/2021 21:52:46] "GET / HTTP/1.1" 200 9723
cmd
python
Python 3.9.5 64-bit
0 5
Ln 2, Col 51
Spaces: 4
UTF-8
CRLF
Python
Jedi: ready

MIGRATION TO NO-SQL DATABASE

Big data and data analytics require migrating from relational databases (SQL) to NoSQL data structures to represent the data. Such transformation is challenging because of the lack of automatic transformation process and the requirement of guaranteeing both performance and accurate representation.

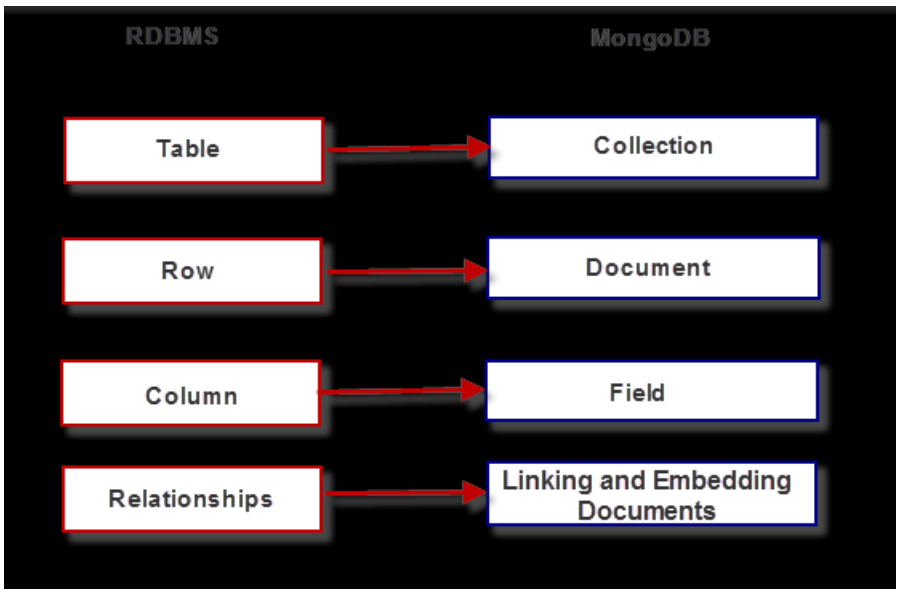
NoSQL datastores are designed for efficiently handling a lot more data than RDBMS. There are no relational constraints on the data, and it does not need to be even tabular. NoSQL offers performance at a higher scale by typically giving up strong consistency. Data access is mostly through REST APIs.The core principle of NoSQL is 'high availability'.

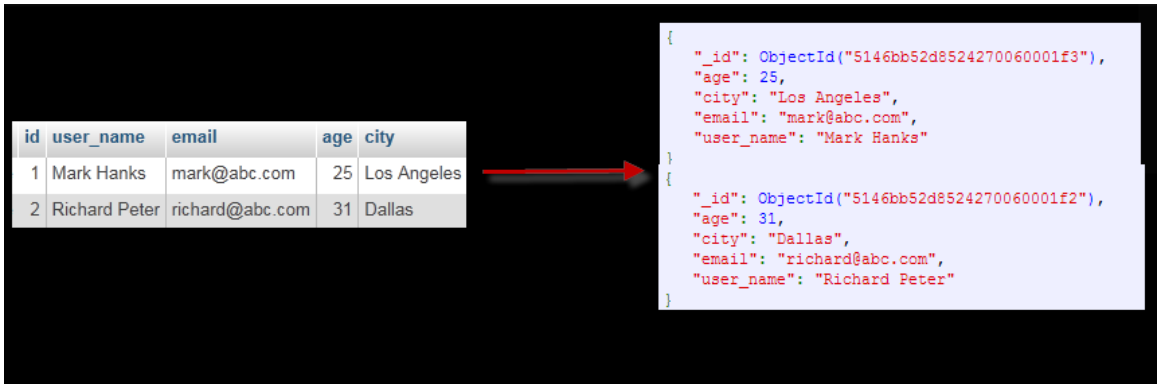
NoSQL is defined as a term which refers to a specific type of database model or DBMS.
Benefits include cost benefits, performance, scalability, future proof for changes, reducing conversion jobs, and extensive supportability for analytics. Apache Cassandra and MongoDB are the most popularly used NoSQL databases.

Reasons to Use a NoSQL Database

- Semi-Structured and Unstructured Data. Data on the Web can be unstructured, semi-structured, or structured
- Agile Database Schema.
- Distributed
- Scalable
- Highly Available

SQL v/s NoSQL database mapping





Features Comparison

	Relational database	MongoDB
Rich Data Model	No	Yes
Dyamic Schema	No	Yes
Typed Data	Yes	Yes
Data Locality	No	Yes
Field Updates	Yes	Yes
Easy for Programmers	No	Yes
Complex Transactions	Yes	No
Auditing	Yes	Yes
Auto-Sharding	No	Yes

Advantages of relational databases

- Good choice for applications that involve the management of several transactions.
- The structure of a relational database allows us to link information from different tables using foreign keys.
- Maintains ACID properties (the set of properties that guarantee database transactions are processed reliably) important for relibility.
- SQL is known by huge no of people.

Advantages of non-relational databases

- Lightweight data interchange format.
- if you find yourself having to de-normalize your database schema, non-relational databases like Mongo may be the best way to go.
- Database is not at risk for SQL injection attacks.
- Sharding distributes the data across partitions to overcome hardware limitations.

Disadvantages of non-relational databases

- Since there are no joins like there would be in relational databases, we need to perform multiple queries and join the data manually within our code – and that can get very ugly, very fast.
- Mongo doesn't automatically treat operations as transactions the way a relational database does, we must manually choose to create a transaction and then manually verify it, manually commit it or roll it back. To put it simply, some operations will succeed while others fail.

CONTRIBUTION

Bhavana R:

Front end, additional queries – 3 TO 4 HRS

Bhagyashree Shankar

Migration to No SQL , Report – 2 TO 3 HRS

Basanagouda :

Schema Changes and additional queries - 2 TO 3 HRS