CS 631 – MORRIS HEALTH SERVICES PROJECT

FINAL DELIVERABLE

HRITHIKKA GUDA (hg345)

VISHWA THEERTHALA (vt327)

RAGHU RAIKANTI (rr777)

Description of implementation, problems faced:

(1) Database Setup:
- We have created a MySQL database named MHS.
- We have designed and implemented the necessary tables based on an updated EER (Enhanced Entity-Relationship) diagram.
- Data has been inserted into each table to populate them with initial values.

(2) Backend Logic (main.py):
- The main.py file serves as the backend logic for your application.
- You establish a connection to the MySQL database.
- Functions related to the database schema are defined to handle operations like querying, inserting, updating, and deleting data according to the application's requirements.
- These functions encapsulate the logic needed to interact with the database.

(3) Frontend Development:
- Utilizing Streamlit, you build the frontend of your application within the same main.py file.
- You design the user interface (UI) using Streamlit's intuitive syntax and widgets.
- The frontend interacts with the backend functions you've defined earlier to display and manipulate data.

(4) Requirements:
- Before running the application, ensure that the necessary Python packages are installed, namely Streamlit and the MySQL connector for Python.
- This can be done using pip install streamlit mysql-connector-python.

(5) Running the Application:
- To launch the application, execute the command streamlit run main.py.
- This command starts the Streamlit server and runs your main.py file.
- Once the server is up and running, you'll receive a message indicating that you can view your Streamlit app in your browser.
- You can access the application locally via the provided URL.

PROBLEMS FACED:

We have encountered some issues related to case sensitivity in database table creations and queries. To address these issues, we've wisely implemented error handling using try-catch-finally blocks to ensure that the application continues running smoothly even if such errors occur.

USER GUIDE:

MHS Application User Guide

Introduction:

Welcome to the MHS (Medical Healthcare System) application! This guide will help you navigate and utilize the various features of the application, including Employee and Facility Management, Patient Management, and Management and Reporting functionalities.

Getting Started:

1. *System Requirements:*

   - Ensure you have Python installed on your system.

   - Install required Python packages using:

```
pip install streamlit mysql-connector-python
```

2. *Database Setup:*

   - Make sure you have a MySQL database named MHS set up with necessary tables populated.

3. *Running the Application:*

   - Open your terminal/command prompt.

   - Navigate to the directory containing the main.py file.

   - Execute the following command:

```
streamlit run main.py
```

Navigating the Application:

- Upon launching the application, you'll be presented with a main menu.
- From the main menu, select the desired application program:
  - Employee and Facility Management
  - Patient Management
  - Management and Reporting

Employee and Facility Management:

- *Insert / Update / View:*
  - Navigate through options to manage employees, medical offices, out-patient surgery facilities, etc.

Patient Management:

- *Create New Patient Records:*
  - Enter details to create new patient records.
- *Appointments and Charges:*
  - Schedule appointments and update charges upon completion.
- *Generate Invoices:*
  - Create daily insurance company invoices with patient subtotals.

Management and Reporting:

- *Revenue Report:*
  - Generate revenue report by facility for a given day.
- *Appointment List:*
  - View appointments for a selected date and physician.
- *Appointment Details:*
  - List appointments with detail for a selected time period and facility.
- *Financial Performance:*
  - Compute statistics for income, facilities, employees, and patients.

Troubleshooting:

- If you encounter any issues or errors:

- Check your internet connection and database connectivity.
- Ensure correct input format and data integrity.

Best Practices:

- Regularly backup your database to prevent data loss.
- Follow proper data entry procedures to maintain accuracy.
- Optimize database queries for better performance.

The source code.

Main.py

```
# Function to connect to MySQL database

import streamlit as st
import mysql.connector
from datetime import datetime

# Function to connect to MySQL database
def connect_to_database():
    return mysql.connector.connect(
        host="127.0.0.1",
        port=3308,
        user="root",
        password="root",
        database="MHS"
    )

# Function to get primary FACILITIES based on job class
def get_primary_FACILITIES(job_class):
```

```
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Execute query based on job class
        if job_class == "physician":
            cursor.execute("select facid, concat(street,', ',city,', ',state,', ',zip) as primary_office
from FACILITIES where ftype = 'office'")
        else:
            cursor.execute("select facid, concat(street,', ',city,', ',state,', ',zip) as primary_office
from FACILITIES where ftype <> 'office'")
        results = cursor.fetchall()
        return [str(result[0])+" "+result[1] for result in results]  # Extract facility names from the
results
    except mysql.connector.Error as error:
        st.error(f"Error getting primary FACILITIES: {error}")
    finally:
        if connection.is_connected():
            cursor.close() # best practice
            connection.close()


def get_secondary_FACILITIES():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        cursor.execute("select facid,concat(street,', ',city,', ',state,', ',zip) as secondary_office
from FACILITIES")
        results = cursor.fetchall()
        return [str(result[0])+" "+result[1] for result in results]  # Extract facility names from the
results
```

```
    except mysql.connector.Error as error:
        st.error(f"Error getting secondary FACILITIES: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

# Function to get all employee names from the database
def get_all_employee_names():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        cursor.execute("SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM
EMPLOYEES")
        results = cursor.fetchall()
        return [result[0] for result in results]
    except mysql.connector.Error as error:
        st.error(f"Error getting employee names: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

# Function to get employee details by name
def get_employee_details_by_name(full_name):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
```

```
        cursor.execute("SELECT * FROM EMPLOYEES WHERE CONCAT(first_name, ' ',
last_name) = %s", (full_name,))
        return cursor.fetchone()
    except mysql.connector.Error as error:
        st.error(f"Error getting employee details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


# Function to update employee information
def update_employee_info(empid, new_info):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Update employee information
        sql = "UPDATE EMPLOYEES SET first_name = %s, middle_name = %s, last_name =
%s, ssn = %s, salary = %s, hire_date = %s, street = %s, city = %s, state = %s, zip = %s
WHERE empid = %s"
        values = (new_info['first_name'], new_info['middle_name'], new_info['last_name'],
new_info['ssn'], new_info['salary'], new_info['hire_date'], new_info['street'], new_info['city'],
new_info['state'], new_info['zipcode'], empid)
        cursor.execute(sql, values)
        connection.commit()
        st.success("Employee information updated successfully.")
    except mysql.connector.Error as error:
        st.error(f"Error updating employee information: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
```

```
def get_all_employee_details():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM EMPLOYEES")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting employee details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def get_all_employee_assignments():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("select EMPLOYEES.first_name, EMPLOYEES.middle_name,
EMPLOYEES.last_name, EMPLOYEES.job_class, FACILITIES.* from EMPLOYEES,
FACILITIES where EMPLOYEES.facid = FACILITIES.facid order by
EMPLOYEES.empid")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting employee assignment details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
```

```
        connection.close()

# Function to get all medical office details
def get_medical_office_details():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM FACILITIES WHERE ftype = 'office'")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting medical office details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

# Function to update medical office information
def update_medical_office_info(facid, new_info):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Update medical office information
        sql = "UPDATE FACILITIES SET size = %s, street = %s, city = %s, state = %s, zip =
%s WHERE facid = %s"
        values = (new_info['size'], new_info['street'], new_info['city'], new_info['state'],
new_info['zip'], facid)
        cursor.execute(sql, values)
        connection.commit()
        st.success("Medical office information updated successfully.")
```

```
    except mysql.connector.Error as error:
        st.error(f"Error updating medical office information: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()



# Function to get all medical ops details
def get_medical_ops_details():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM FACILITIES WHERE ftype = 'outpatient surgery'")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting medical ops details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

# Function to update medical ops information
def update_medical_ops_info(facid, new_info):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Update medical office information
```

```
        sql = "UPDATE FACILITIES SET size = %s, street = %s, city = %s, state = %s, zip =
%s WHERE facid = %s"
        values = (new_info['size'], new_info['street'], new_info['city'], new_info['state'],
new_info['zip'], facid)
        cursor.execute(sql, values)
        connection.commit()
        st.success("OPS information updated successfully.")
    except mysql.connector.Error as error:
        st.error(f"Error updating medical OPS information: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

# Function to get all insurance company details
def get_insurance_company_details():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM INSURANCE_COMPANIES")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting insurance company details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()

# Function to add a new insurance company
def add_insurance_company(company, street, city, state, zip_code):
```

```
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Add new insurance company
        sql = "INSERT INTO INSURANCE_COMPANIES (company, street, city, state, zip)
VALUES (%s, %s, %s, %s, %s)"
        values = (company, street, city, state, zip_code)
        cursor.execute(sql, values)
        connection.commit()
        st.success("Insurance company added successfully.")
    except mysql.connector.Error as error:
        st.error(f"Error adding insurance company: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()



def update_insurance_company_info(ins_id, new_info):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Update insurance company information
        sql = "UPDATE INSURANCE_COMPANIES SET company = %s, street = %s, city =
%s, state = %s, zip = %s WHERE ins_id = %s"
        values = (new_info['company'], new_info['street'], new_info['city'], new_info['state'],
new_info['zip'], ins_id)
        cursor.execute(sql, values)
        connection.commit()
```

```
        st.success("Insurance company information updated successfully.")
    except mysql.connector.Error as error:
        st.error(f"Error updating insurance company information: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


# Function to get all primary PHYSICIANS
def get_primary_physician_details():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM
EMPLOYEES WHERE job_class='physician'")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting physician details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def get_INSURANCE_COMPANIES():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT company FROM INSURANCE_COMPANIES")
        return cursor.fetchall()
```

```
    except mysql.connector.Error as error:
        st.error(f"Error getting company details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def add_patient(first_name, middle_name, last_name, street, city, state,zipcode,
primary_physician_id, ins_id):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Add new insurance company
        sql = "INSERT INTO PATIENTS (first_name, middle_name, last_name, street, city,
state,zip, primary_physician_id, ins_id) VALUES (%s, %s, %s, %s, %s,%s, %s, %s, %s)"
        values = (first_name, middle_name, last_name, street, city, state,zipcode,
primary_physician_id, ins_id)
        cursor.execute(sql, values)
        connection.commit()
        st.success("Patient Details added successfully.")
    except mysql.connector.Error as error:
        st.error(f"Error adding patient detials: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


# Function to get all PATIENTS
def get_patient_details():
    try:
```

```
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM
PATIENTS")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting patient details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def add_appointment(formatted_datetime, description, patient_id, physician_id, facid):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Add new insurance company
        sql = "INSERT INTO APPOINTMENTS (appt_date_time, appt_description, patient_id,
physician_id, facid) VALUES (%s, %s, %s, %s, %s)"
        values = (formatted_datetime, description, patient_id, physician_id, facid)
        cursor.execute(sql, values)
        connection.commit()
        st.success("Appointment Confirmed")
    except mysql.connector.Error as error:
        st.error(f"Error adding appointment detials: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
```

```
def add_invoice(selected_appt_date_time, appointment_cost, selected_appt_id):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Add new insurance company
        sql = "INSERT INTO INVOICES (inv_date, amount,appt_id) VALUES (%s, %s, %s)"
        values = (selected_appt_date_time, appointment_cost, selected_appt_id)
        cursor.execute(sql, values)
        connection.commit()
        st.success("Invoice added to the appointment")
    except mysql.connector.Error as error:
        st.error(f"Error adding invoice detials: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def display_totals(date_input,ins_id):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        sql1 = "select p.first_name, p.last_name, sum(inv.amount) as patient_total from
PATIENTS p, APPOINTMENTS a, INVOICES inv where p.pid = a.patient_id and a.appt_id
= inv.appt_id and inv.inv_date = %s and p.ins_id = %s group by p.pid"
        values = (date_input,ins_id)
        cursor.execute(sql1,values)
        return cursor.fetchall()
```

```
    except mysql.connector.Error as error:
        st.error(f"Error getting patient details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def revenue_report(f_dt):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        sql1 = "select f.facid, f.street, f.city, f.state, f.zip, sum(inv.amount) as total from
FACILITIES f, APPOINTMENTS a , INVOICES inv where a.facid = f.facid and a.appt_id =
inv.appt_id and inv.inv_date = %s group by f.facid"
        values = (f_dt,)
        cursor.execute(sql1,values)
        return cursor.fetchall()


    except mysql.connector.Error as error:
        st.error(f"Error getting patient details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def apt_list_physician(f_dt,physician_id):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
```

```
        sql1 = "select a.appt_id, a.appt_date_time, a.appt_description, CONCAT(p.first_name, '
', p.last_name) as patient_name,  CONCAT(e.first_name, ' ', e.last_name) as physician_name
, CONCAT(f.street, ' ', f.city, ' ',f.state,' ',f.zip) as facility_location from APPOINTMENTS a,
PATIENTS p, EMPLOYEES e, FACILITIES f where a.patient_id = p.pid and
a.physician_id= e.empid and a.facid = f.facid and date(a.appt_date_time) = %s and
a.physician_id = %s"
        values = (f_dt,physician_id)
        cursor.execute(sql1,values)
        return cursor.fetchall()


    except mysql.connector.Error as error:
        st.error(f"Error getting patient details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def apt_list_facility(begin_dt,end_dt,facid):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        sql1 = "select a.appt_id, a.appt_date_time, a.appt_description, CONCAT(p.first_name, '
', p.last_name) as patient_name,  CONCAT(e.first_name, ' ', e.last_name) as physician_name
, CONCAT(f.street, ' ', f.city, ' ',f.state,' ',f.zip) as facility_location from APPOINTMENTS a,
PATIENTS p, EMPLOYEES e, FACILITIES f where a.patient_id = p.pid and
a.physician_id= e.empid and a.facid = f.facid and date(a.appt_date_time) >= %s and
date(a.appt_date_time) <= %s and a.facid = %s"
        values = (begin_dt,end_dt,facid)
        cursor.execute(sql1,values)
        return cursor.fetchall()


    except mysql.connector.Error as error:
        st.error(f"Error getting patient details: {error}")
```

```
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def get_facility_details():
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM FACILITIES")
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting medical office details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def generate_best_days(month_number):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        csql1 = '''select day(a.appt_date_time) as DAY , sum(inv.amount) as
TOTAL_REVENUE
                    from INVOICES inv, APPOINTMENTS a
                    where inv.appt_id = a.appt_id
                    and month(a.appt_date_time) = %s
                    group by day(a.appt_date_time)
                    order by sum(inv.amount) desc
```

```
                    limit 5'''
        values = (month_number,)
        cursor.execute(csql1,values)
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting medical office details: {error}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()


def generate_average_revenue(begin_dt, end_dt):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor(dictionary=True)
        csql1 = '''select ins.company , avg(inv.amount)
                    from APPOINTMENTS a, PATIENTS p, INSURANCE_COMPANIES ins,
INVOICES inv
                    where a.patient_id = p.pid
                    and p.ins_id = ins.ins_id
                    and a.appt_id = inv.appt_id
                    and date(a.appt_date_time) >= %s
                    and date(a.appt_date_time) <= %s
                    group by ins.company'''
        values = (begin_dt, end_dt)
        cursor.execute(csql1,values)
        return cursor.fetchall()
    except mysql.connector.Error as error:
        st.error(f"Error getting medical office details: {error}")
    finally:
```

```
    if connection.is_connected():
        cursor.close()
        connection.close()


#main front-end part
# Sidebar navigation
st.sidebar.title("Navigation")
option = st.sidebar.radio("Go to:", ("Home", "Employee and Facility Management", "Patient Management", "Management Reporting"))


# Main content
if option == "Home":
    st.title("Morris Health Services")
    st.write("Welcome to Morris Health Services. Please select an option from the sidebar to navigate.")
    # Add About Us section
    st.subheader("About Us")
    st.write("At MHS we provide comprehensive health services to the residents of northern New Jersey.")
elif option == "Employee and Facility Management":
    st.title("Welcome to Employee and Facility Management Portal at MHS")
    # Add Ask a Question section
    question_option = st.selectbox("What do you want to manage today?", ("EMPLOYEES", "Medical OFFICES", "Out Patient Surgery FACILITIES", "Insurance Companies"))
    if question_option == "EMPLOYEES":
        st.subheader("Employee Management Options")
        employee_option = st.radio("Select an option:", ("Add new employee", "Edit Employee Information", "View EMPLOYEES", "Employee Assignments"))
        if employee_option == "Add new employee":
            st.subheader("Add a New Employee")#shows everything related to add new employee
            # Collect employee information
            first_name = st.text_input("First Name")
```

```
middle_name = st.text_input("Middle Name")
last_name = st.text_input("Last Name")
ssn = st.text_input("Social Security Number (SSN)")
salary = st.number_input("Salary")
salary = int(salary) #changing into int
hire_date = st.date_input("Hire Date")
job_class = st.selectbox("Job Class", ("physician", "nurse", "HCP", "admin staff"))
street = st.text_input("Street")
city = st.text_input("City")
state = st.text_input("State")
zipcode = st.text_input("Zip Code")
primary_facility_options = get_primary_FACILITIES(job_class)
pf = st.selectbox("Primary Facility", primary_facility_options)
facid = pf.split(" ")[0]
facid = int(facid)
secondary_facility_options = get_secondary_FACILITIES()
secondary_FACILITIES = st.multiselect("Secondary FACILITIES",secondary_facility_options) #to select multiple
# Save button to upload data to MySQL database
if st.button("Save"):
    try:
        # Connect to MySQL database
        connection = connect_to_database()
        cursor = connection.cursor()
        # Insert employee data into the database
        sql = "INSERT INTO EMPLOYEES (first_name, middle_name, last_name, ssn, salary, hire_date, job_class, street, city, state, zip, facid) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
        values = (first_name, middle_name, last_name, ssn, salary, hire_date, job_class, street, city, state, zipcode, facid)
        cursor.execute(sql, values)
        connection.commit()
```

```
            st.success("Employee information saved successfully.")
        except mysql.connector.Error as error:
            st.error(f"Error saving employee information: {error}")
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
                st.write("MySQL connection is closed.")
    elif employee_option == "Edit Employee Information":
        st.subheader("Edit Existing Employee Info")
        employee_names = get_all_employee_names()
        selected_employee = st.selectbox("Select an employee to edit:", employee_names)
        if selected_employee:
            employee_details = get_employee_details_by_name(selected_employee)
            if employee_details:
                st.write(f"Editing Employee: {selected_employee}")
                # Display existing employee information
                st.write("Existing Information:")
                st.write(f"SSN: {employee_details['ssn']}")
                st.write(f"Salary: {employee_details['salary']}")
                st.write(f"Hire Date: {employee_details['hire_date']}")
                st.write(f"Job Class: {employee_details['job_class']}")
                st.write(f"Street: {employee_details['street']}")
                st.write(f"City: {employee_details['city']}")
                st.write(f"State: {employee_details['state']}")
                st.write(f"Zip Code: {employee_details['zip']}")
                # Collect updated information
                new_ssn = st.text_input("New SSN", value=employee_details['ssn'])
                new_salary = st.number_input("New Salary", value=employee_details['salary'])
                new_hire_date = st.date_input("New Hire Date",
value=employee_details['hire_date'])
```

```
            #new_job_class = st.selectbox("New Job Class", ("physician", "nurse", "HCP",
"admin staff"), index=employee_details['job_class'])
            new_street = st.text_input("New Street", value=employee_details['street'])
            new_city = st.text_input("New City", value=employee_details['city'])
            new_state = st.text_input("New State", value=employee_details['state'])
            new_zipcode = st.text_input("New Zip Code", value=employee_details['zip'])
            # Update button to save changes
            if st.button("Update Information"):
                new_info = {
                    'first_name': employee_details['first_name'],
                    'middle_name': employee_details['middle_name'],
                    'last_name': employee_details['last_name'],
                    'ssn': new_ssn,
                    'salary': new_salary,
                    'hire_date': new_hire_date,
                    #'job_class': new_job_class,
                    'street': new_street,
                    'city': new_city,
                    'state': new_state,
                    'zipcode': new_zipcode
                }
                update_employee_info(employee_details['empid'], new_info)

        elif employee_option == "View EMPLOYEES":
            st.subheader("View EMPLOYEES")
            employee_details = get_all_employee_details()
            if employee_details:
                st.write("List of EMPLOYEES:")
                st.table(employee_details)
            else:
                st.write("No EMPLOYEES found.")
```

```
        elif employee_option == "Employee Assignments":
            st.subheader("View Employee Assignments")
            employee_assignments = get_all_employee_assignments()
            if employee_assignments:
                st.write("List of Employee Assignments:")
                st.table(employee_assignments)
            else:
                st.write("No EMPLOYEES found.")



    elif question_option == "Medical OFFICES":
        st.subheader("Medical OFFICES Management Options")
        office_option = st.radio("Select an option:", ("Add new office", "Edit existing office",
"View OFFICES"))
        if office_option == "Add new office":
            st.subheader("Add a New Ofiice")
            # Collect office information
            size = st.number_input("Office_Size")
            size = int(size)
            ftype = "office"
            street = st.text_input("Street")
            city = st.text_input("City")
            state = st.text_input("State")
            zipcode = st.text_input("Zip Code")
            office_count = st.number_input("Office_Count")
            office_count  = int(office_count)

            if st.button("Save"):
                try:
                    # Connect to MySQL database
```

```
            connection = connect_to_database()
            cursor = connection.cursor()
            # Insert employee data into the database
            sql1 = "INSERT INTO FACILITIES (size,ftype,street, city, state, zip) VALUES
(%s, %s, %s, %s, %s, %s)"
            values = (size,ftype,street, city, state, zipcode)
            cursor.execute(sql1, values)
            sql3 = "SELECT MAX(facid) from FACILITIES"
            cursor.execute(sql3)
            facid = cursor.fetchone()[0]
            sql2 = "INSERT INTO OFFICES (facid, office_count) VALUES (%s, %s)"
            values = (facid, office_count)
            cursor.execute(sql2, values)
            connection.commit()
            st.success("Office information saved successfully.")
        except mysql.connector.Error as error:
            st.error(f"Error saving office information: {error}")
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
                st.write("MySQL connection is closed.")


    elif office_option == "Edit existing office":
        st.subheader("Edit Existing Medical OFFICES")
        medical_office_details = get_medical_office_details()
        selected_office = st.selectbox("Select medical office to edit:", [office['facid'] for office
in medical_office_details])
        selected_office_details = next((office for office in medical_office_details if
office['facid'] == selected_office), None)
        if selected_office_details:
```

```
        st.write(f"Facility ID: {selected_office_details['facid']}")
        st.write(f"Size: {selected_office_details['size']}")
        st.write(f"Street: {selected_office_details['street']}")
        st.write(f"City: {selected_office_details['city']}")
        st.write(f"State: {selected_office_details['state']}")
        st.write(f"Zip: {selected_office_details['zip']}")
        # Collect updated information
        new_size = st.text_input("New Size", value=selected_office_details['size'])
        new_street = st.text_input("New Street", value=selected_office_details['street'])
        new_city = st.text_input("New City", value=selected_office_details['city'])
        new_state = st.text_input("New State", value=selected_office_details['state'])
        new_zip = st.text_input("New Zip", value=selected_office_details['zip'])
        # Update button to save changes
        if st.button("Update Information"):
            new_info = {
                'size': new_size,
                'street': new_street,
                'city': new_city,
                'state': new_state,
                'zip': new_zip
            }
            update_medical_office_info(selected_office_details['facid'], new_info)
    else:
        st.write("No medical office found.")

elif office_option == "View OFFICES":
    st.subheader("View Medical OFFICES")
    medical_office_details = get_medical_office_details()
    if medical_office_details:
        st.write("List of Medical OFFICES:")
```

```
            st.table(medical_office_details)
        else:
            st.write("No medical OFFICES found.")


    elif question_option == "Out Patient Surgery FACILITIES":
        st.subheader("Out Patient Surgery FACILITIES Management Options")
        office_option = st.radio("Select an option:", ("Add new OPS", "Edit existing OPS",
"View OPS"))
        if office_option == "Add new OPS":
            st.subheader("Add a New OPS")
            # Collect office information
            size = st.number_input("Office_Size")
            size = int(size)
            ftype = "office"
            street = st.text_input("Street")
            city = st.text_input("City")
            state = st.text_input("State")
            zipcode = st.text_input("Zip Code")
            room_count = st.number_input("Room_Count")
            room_count  = int(room_count)
            procedures = st.text_input("Procedures")


            if st.button("Save"):
                try:
                    # Connect to MySQL database
                    connection = connect_to_database()
                    cursor = connection.cursor()
                    # Insert employee data into the database
                    sql1 = "INSERT INTO FACILITIES (size,ftype,street, city, state, zip) VALUES
(%s, %s, %s, %s, %s, %s)"
                    values = (size,ftype,street, city, state, zipcode)
```

```
            cursor.execute(sql1, values)
            sql3 = "SELECT MAX(facid) from FACILITIES"
            cursor.execute(sql3)
            facid = cursor.fetchone()[0]
            sql2 = "INSERT INTO OUTPATIENT_SURGERY_ROOMS (facid,
room_count, procedures) VALUES (%s, %s, %s)"
            values = (facid, room_count, procedures)
            cursor.execute(sql2, values)
            connection.commit()
            st.success("OPS information saved successfully.")
        except mysql.connector.Error as error:
            st.error(f"Error saving OPS information: {error}")
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
                st.write("MySQL connection is closed.")


    elif office_option == "Edit existing OPS":
        st.subheader("Edit Existing OPS")
        medical_ops_details = get_medical_ops_details()
        selected_office = st.selectbox("Select medical ops to edit:", [office['facid'] for office
in medical_ops_details])
        selected_office_details = next((office for office in medical_ops_details if
office['facid'] == selected_office), None)
        if selected_office_details:
            st.write(f"Facility ID: {selected_office_details['facid']}")
            st.write(f"Size: {selected_office_details['size']}")
            st.write(f"Street: {selected_office_details['street']}")
            st.write(f"City: {selected_office_details['city']}")
            st.write(f"State: {selected_office_details['state']}")
```

```
            st.write(f"Zip: {selected_office_details['zip']}")
            # Collect updated information
            new_size = st.text_input("New Size", value=selected_office_details['size'])
            new_street = st.text_input("New Street", value=selected_office_details['street'])
            new_city = st.text_input("New City", value=selected_office_details['city'])
            new_state = st.text_input("New State", value=selected_office_details['state'])
            new_zip = st.text_input("New Zip", value=selected_office_details['zip'])
            # Update button to save changes
            if st.button("Update Information"):
                new_info = {
                    'size': new_size,
                    'street': new_street,
                    'city': new_city,
                    'state': new_state,
                    'zip': new_zip
                }
                update_medical_ops_info(selected_office_details['facid'], new_info)
        else:
            st.write("No medical office found.")

    elif office_option == "View OPS":
        st.subheader("View Medical OPS")
        medical_office_details = get_medical_ops_details()
        if medical_office_details:
            st.write("List of Medical OPS:")
            st.table(medical_office_details)
        else:
            st.write("No medical OFFICES found.")

elif question_option == "Insurance Companies":
```

```
    st.subheader("Insurance Company Management Options")
    insurance_option = st.radio("Select an option:", ("Add New Insurance Company", "Edit Insurance Company", "View Insurance Companies"))
    if insurance_option == "Add New Insurance Company":
        st.subheader("Add New Insurance Company")
        company = st.text_input("Company")
        street = st.text_input("Street")
        city = st.text_input("City")
        state = st.text_input("State")
        zip_code = st.text_input("Zip Code")
        if st.button("Add Insurance Company"):
            add_insurance_company(company, street, city, state, zip_code)
    elif insurance_option == "Edit Insurance Company":
        st.subheader("Edit Insurance Company")
        INSURANCE_COMPANIES = get_insurance_company_details()
        selected_company = st.selectbox("Select insurance company to edit:", [company['company'] for company in INSURANCE_COMPANIES])
        selected_company_details = next((company for company in INSURANCE_COMPANIES if company['company'] == selected_company), None)
        if selected_company_details:
            st.write(f"Insurance Company ID: {selected_company_details['ins_id']}")
            st.write(f"Company: {selected_company_details['company']}")
            st.write(f"Street: {selected_company_details['street']}")
            st.write(f"City: {selected_company_details['city']}")
            st.write(f"State: {selected_company_details['state']}")
            st.write(f"Zip: {selected_company_details['zip']}")
            # Collect updated information
            new_name = st.text_input("New Company Name", value=selected_company_details['company'])
            new_street = st.text_input("New Street", value=selected_company_details['street'])
            new_city = st.text_input("New City", value=selected_company_details['city'])
            new_state = st.text_input("New State", value=selected_company_details['state'])
```

```
            new_zip = st.text_input("New Zip", value=selected_company_details['zip'])
            # Update button to save changes
            if st.button("Update Information"):
                new_info = {
                    'company': new_name,
                    'street': new_street,
                    'city': new_city,
                    'state': new_state,
                    'zip': new_zip
                }
                update_insurance_company_info(selected_company_details['ins_id'], new_info)
        else:
            st.write("No insurance company found.")
    elif insurance_option == "View Insurance Companies":
        st.subheader("View Insurance Companies")
        INSURANCE_COMPANIES = get_insurance_company_details()
        if INSURANCE_COMPANIES:
            st.write("List of Insurance Companies:")
            st.table(INSURANCE_COMPANIES)
        else:
            st.write("No insurance companies found.")


elif option == "Patient Management":
    st.title("Welcome to Patient Management Portal at MHS")
    question_option = st.radio("Choose to manage?", ("PATIENTS", "APPOINTMENTS",
"Insurance_INVOICES"))
    if question_option == "PATIENTS":
        st.subheader("Add a new patient")
        first_name = st.text_input("First Name")
        middle_name = st.text_input("Middle Name")
        last_name = st.text_input("Last Name")
```

```
    street = st.text_input("Street")
    city = st.text_input("City")
    state = st.text_input("State")
    zipcode = st.text_input("Zip Code")

    PHYSICIANS = get_primary_physician_details()
    physician_names = [physician['full_name'] for physician in PHYSICIANS]
    primary_physician = st.selectbox("Primary Physician", physician_names)
    connection = connect_to_database()
    cursor = connection.cursor()
    sql3 = "SELECT empid from EMPLOYEES where CONCAT(first_name, ' ', last_name)
= %s"
    values = (primary_physician,)
    cursor.execute(sql3,values)
    primary_physician_id = cursor.fetchall()[0][0]



    companies = get_INSURANCE_COMPANIES()
    company_names = [company['company'] for company in companies]
    insurance_company = st.selectbox("Insurance Company", company_names)
    sql4 = "SELECT ins_id from INSURANCE_COMPANIES where company = %s"
    values = (insurance_company,)
    cursor.execute(sql4,values)
    ins_id = cursor.fetchall()[0][0]
    cursor.close()
    connection.close()

    if st.button("Add Patient"):
        add_patient(first_name, middle_name, last_name, street, city, state,zipcode,
primary_physician_id, ins_id)
```

```
    elif question_option == "APPOINTMENTS":
        appt_option = st.radio("Do you want to schedule an appointment or update an
appointment with amount?", ("Create", "Update"))
        if(appt_option=="Create"):
            st.subheader("Create an appointment")
            date_input = st.date_input("Select Date")
            time_input = st.time_input("Select Time")
            datetime_obj = datetime.combine(date_input, time_input)
            formatted_datetime = datetime_obj.strftime('%Y-%m-%d %H:%M:%S')
            description = st.text_input("Description")

            PATIENTS = get_patient_details()
            patient_names = [patient['full_name'] for patient in PATIENTS]
            pname = st.selectbox("Patient", patient_names)
            connection = connect_to_database()
            cursor = connection.cursor()
            sql3 = "SELECT pid from PATIENTS where CONCAT(first_name, ' ', last_name) =
%s"
            values = (pname,)
            cursor.execute(sql3,values)
            patient_id = cursor.fetchall()[0][0]
            cursor.close()
            connection.close()

            make_appointment = st.radio("Do you want to make an appointment with your
Primary Physician?", ("Yes", "No"))
            connection = connect_to_database()
            cursor = connection.cursor()
            sql3 = "SELECT CONCAT(emp.first_name, ' ', emp.last_name) as doc_name from
PATIENTS p, PHYSICIANS phy, EMPLOYEES emp where p.primary_physician_id =
phy.empid and phy.empid = emp.empid and p.pid = '%s'"
            values = (patient_id,)
```

```
        cursor.execute(sql3,values)
        doctor = cursor.fetchall()[0][0]
        cursor.close()
        connection.close()
        if make_appointment == "Yes":
            st.write("Great! Let's proceed to schedule your appointment with ",doctor)
            connection = connect_to_database()
            cursor = connection.cursor()
            sql3 = "SELECT empid from EMPLOYEES where CONCAT(first_name, ' ',
last_name) = %s"
            values = (doctor,)
            cursor.execute(sql3,values)
            physician_id = cursor.fetchall()[0][0]
            cursor.close()
            connection.close()

            connection = connect_to_database()
            cursor = connection.cursor()
            sql3 = "SELECT facid from EMPLOYEES where empid = %s"
            values = (physician_id,)
            cursor.execute(sql3,values)
            facid = cursor.fetchall()[0][0]
            cursor.close()
            connection.close()

            connection = connect_to_database()
            cursor = connection.cursor()
            sql3 = "SELECT street, city, state, zip from FACILITIES where facid = %s"
            values = (facid,)
            cursor.execute(sql3,values)
            address = cursor.fetchall()
```

```
        address = address[0]
        address = " ".join(address)
        cursor.close()
        connection.close()

        st.write("Location :",address)



    else:
        st.write("No worries, Let's go ahead and book with other physician")
        PHYSICIANS = get_primary_physician_details()
        physician_names = [physician['full_name'] for physician in PHYSICIANS]
        physician = st.selectbox("Choose a Physician", physician_names)

        connection = connect_to_database()
        cursor = connection.cursor()
        sql3 = "SELECT empid from EMPLOYEES where CONCAT(first_name, ' ',
last_name) = %s"
        values = (physician,)
        cursor.execute(sql3,values)
        physician_id = cursor.fetchall()[0][0]
        cursor.close()
        connection.close()



        connection = connect_to_database()
        cursor = connection.cursor()
        sql3 = "SELECT facid from EMPLOYEES where empid = %s"
        values = (physician_id,)
        cursor.execute(sql3,values)
        facid = cursor.fetchall()[0]
```

```
        facid = facid[0]
        cursor.close()
        connection.close()

        connection = connect_to_database()
        cursor = connection.cursor()
        sql3 = "SELECT street, city, state, zip from FACILITIES where facid = %s"
        values = (facid,)
        cursor.execute(sql3,values)
        address = cursor.fetchall()
        address = address[0]
        address = " ".join(address)
        cursor.close()
        connection.close()

        st.write("Location :",address)

    if st.button("Confirm Appointment"):
        add_appointment(formatted_datetime, description, patient_id, physician_id, facid)

else:
    st.subheader("Update an appointment with amount")
    connection = connect_to_database()
    cursor = connection.cursor()
    cursor.execute("SELECT appt_id, appt_date_time FROM APPOINTMENTS")
    APPOINTMENTS = cursor.fetchall()
    cursor.close()
    connection.close()

    appointment_options = {appt[0]: appt[1] for appt in APPOINTMENTS}
```

```
        selected_appt_id = st.selectbox("Select Appointment",
list(appointment_options.keys()))

        if selected_appt_id:
            # Display selected appointment details
            selected_appt_date_time = appointment_options[selected_appt_id]
            st.write("Selected Appointment ID:", selected_appt_id)
            st.write("Selected Appointment Date Time:", selected_appt_date_time)
            selected_appt_date_time= selected_appt_date_time.strftime('%Y-%m-%d')

        appointment_cost = int(st.number_input("Appointment Cost"))

        if st.button("Update Appointment Cost"):
            add_invoice(selected_appt_date_time, appointment_cost, selected_appt_id)

    else:
        st.subheader("Generate INVOICES to Insurance Company")
        INSURANCE_COMPANIES = get_insurance_company_details()
        selected_company = st.selectbox("Select the insurance company:",
[company['company'] for company in INSURANCE_COMPANIES])
        selected_company_details = next((company for company in
INSURANCE_COMPANIES if company['company'] == selected_company), None)

        date_input = st.date_input("Select Date")

        ins_id = selected_company_details['ins_id']

        if st.button("Generate Invoice"):
            result = display_totals(date_input,ins_id)
            st.table(result)
```

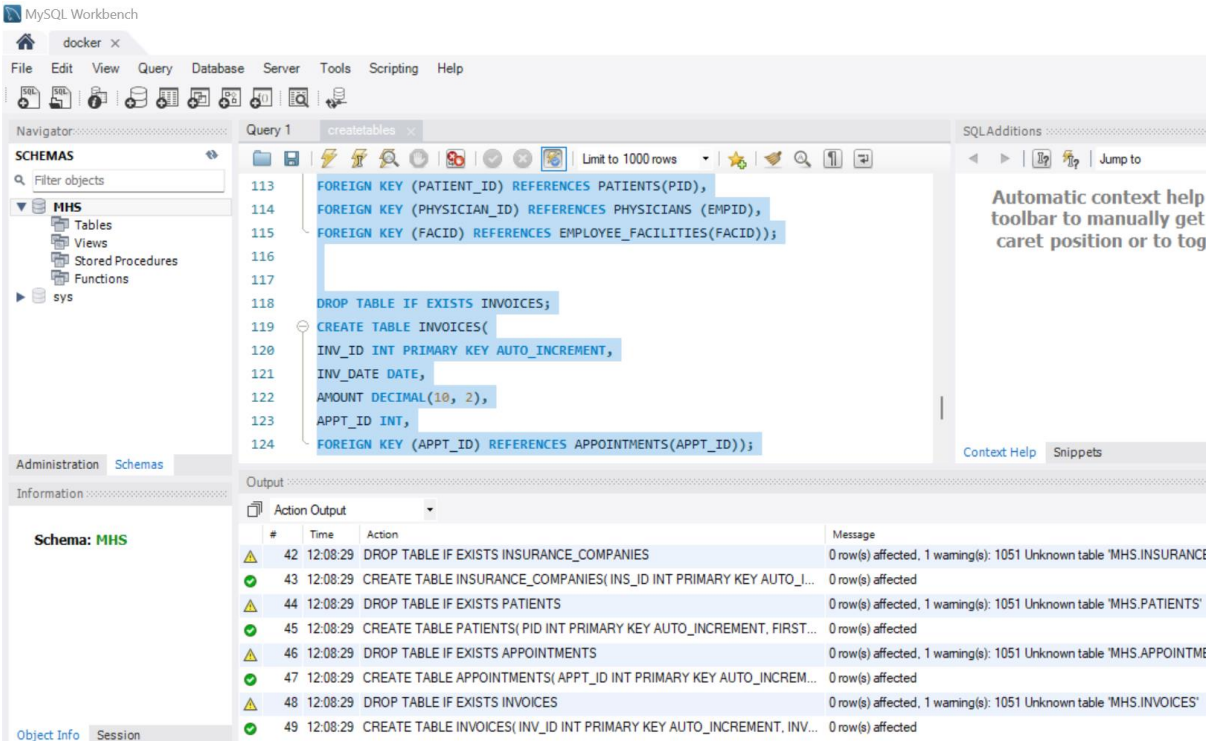
```
#Management Reporting
elif option == "Management Reporting":
    st.title("Welcome to Management Reporting Portal at MHS")
    question_option = st.radio("Choose to generate?", ("Revenue Report", "Appointment List by Physician", "Appointment List by Facility","5 best days in the month","Average daily revenue for insurance companies"))
    if question_option == "Revenue Report":
        st.subheader("Revenue Report")
        date_input = st.date_input("Select Date")
        f_dt = date_input.strftime('%Y-%m-%d')

        if st.button("Generate Revenue Report"):
            result = revenue_report(f_dt)
            st.table(result)

    elif question_option == "Appointment List by Physician":
        st.subheader("Appointment List by Physician")
        date_input = st.date_input("Select Date")
        f_dt = date_input.strftime('%Y-%m-%d')

        PHYSICIANS = get_primary_physician_details()
        physician_names = [physician['full_name'] for physician in PHYSICIANS]
        physician = st.selectbox("Choose a Physician", physician_names)

        connection = connect_to_database()
        cursor = connection.cursor()
        sql3 = "SELECT empid from EMPLOYEES where CONCAT(first_name, ' ', last_name) = %s"
        values = (physician,)
        cursor.execute(sql3,values)
```

```
        physician_id = cursor.fetchall()[0][0]
        cursor.close()
        connection.close()

        if st.button("Generate Appointment List"):
            result = apt_list_physician(f_dt,physician_id)
            st.table(result)

    elif question_option == "Appointment List by Facility":
        st.subheader("Appointment List by Facility")
        date_input = st.date_input("Select Begin Date")
        begin_dt = date_input.strftime('%Y-%m-%d')
        date_input = st.date_input("Select End Date")
        end_dt = date_input.strftime('%Y-%m-%d')

        medical_office_details = get_facility_details()
        selected_office = st.selectbox("Select the facility:", [office['facid'] for office in
medical_office_details])
        selected_office_details = next((office for office in medical_office_details if
office['facid'] == selected_office), None)
        facid = selected_office_details['facid']

        if st.button("Generate Appointment List"):
            result = apt_list_facility(begin_dt,end_dt,facid)
            st.table(result)

    elif question_option == "5 best days in the month":
        st.subheader("5 best days in the month wrt Revenue")
        months = [datetime(2000, i, 1).strftime('%B') for i in range(1, 13)]
        selected_month = st.selectbox("Select a month", months)
        month_number = datetime.strptime(selected_month, '%B').month
```

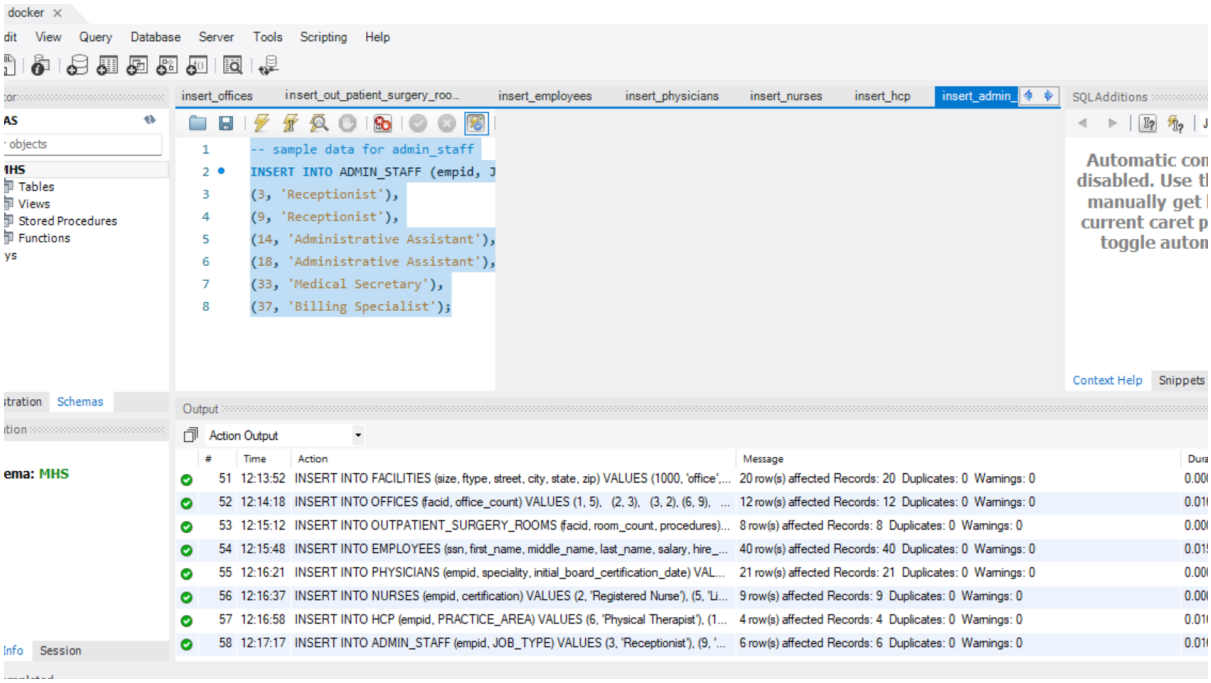
```
    if st.button("Generate the best 5 days"):
        result = generate_best_days(month_number)
        st.table(result)

elif question_option == "Average daily revenue for insurance companies":
    st.subheader("Average daily revenue for insurance companies")
    date_input = st.date_input("Select Begin Date")
    begin_dt = date_input.strftime('%Y-%m-%d')
    date_input = st.date_input("Select End Date")
    end_dt = date_input.strftime('%Y-%m-%d')

    if st.button("Generate Average Revenue"):
        result = generate_average_revenue(begin_dt, end_dt)
        st.table(result)
```

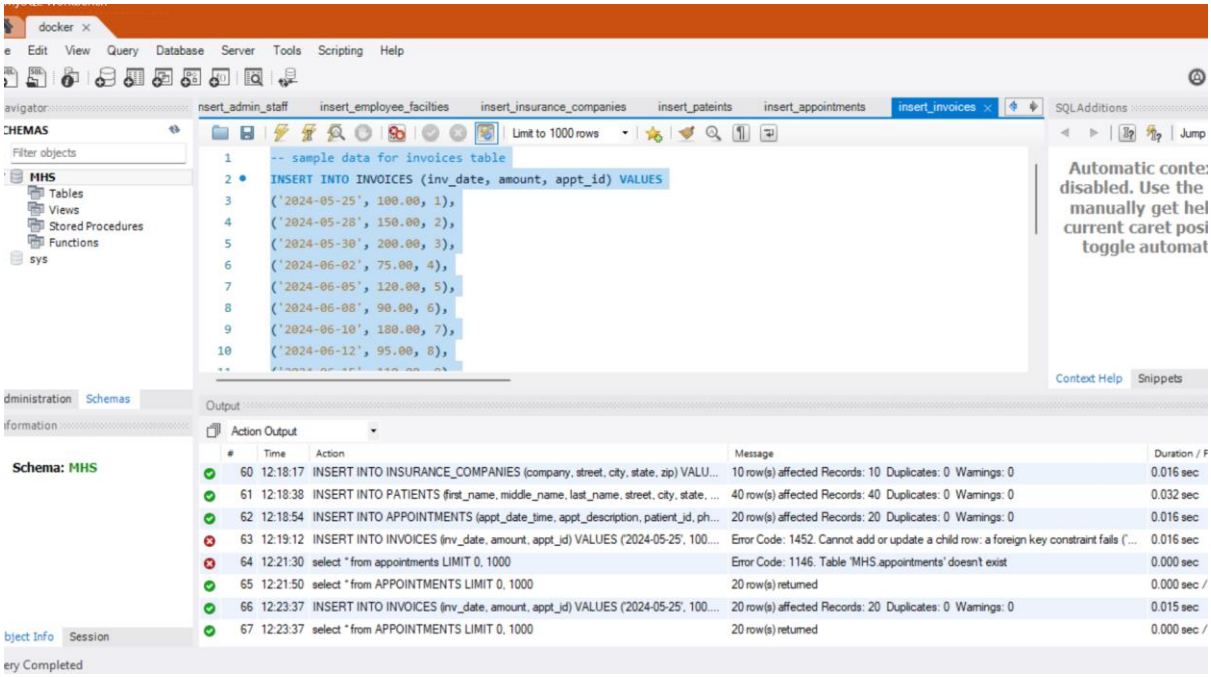
DATABASE

CREATETABLES

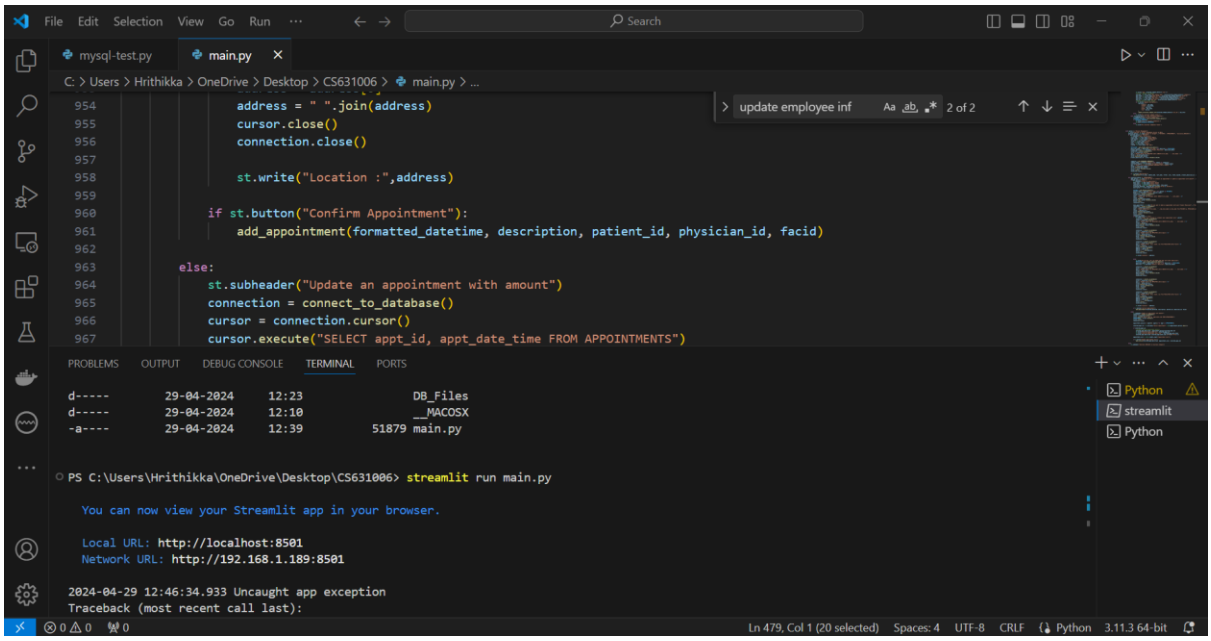


INSERT DATA

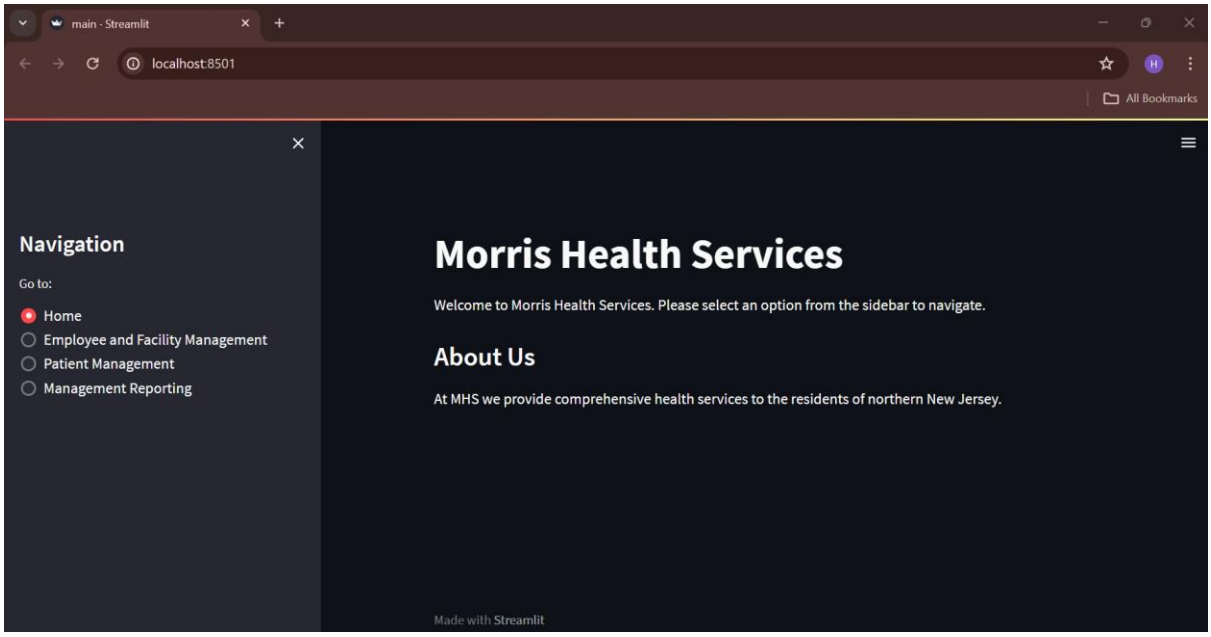

docker
Query Database Server Tools Scripting Help
insert_offices
insert_employees
insert_physicians
insert_nurses
insert_hcp
SQLAdditions
-- sample data for admin_staff
INSERT INTO ADMIN_STAFF (empid, J
(3, 'Receptionist'),
(9, 'Receptionist'),
(14, 'Administrative Assistant'),
(18, 'Administrative Assistant'),
(33, 'Medical Secretary'),
(37, 'Billing Specialist');
Tables
Views
Stored Procedures
Functions
Schemas
Output
Action Output
51 12:13:52 INSERT INTO FACILITIES (size, ftype, street, city, state, zip) VALUES (1000, 'office',... 20 row(s) affected Records: 20 Duplicates: 0 Warnings: 0
52 12:14:18 INSERT INTO OFFICES (facid, office_count) VALUES (1, 5), (2, 3), (3, 2), (6, 9), ... 12 row(s) affected Records: 12 Duplicates: 0 Warnings: 0
53 12:15:12 INSERT INTO OUTPATIENT_SURGERY_ROOMS (facid, room_count, procedures)... 8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0
54 12:15:48 INSERT INTO EMPLOYEES (ssn, first_name, middle_name, last_name, salary, hire_... 40 row(s) affected Records: 40 Duplicates: 0 Warnings: 0
55 12:16:21 INSERT INTO PHYSICIANS (empid, speciality, initial_board_certification_date) VAL... 21 row(s) affected Records: 21 Duplicates: 0 Warnings: 0
56 12:16:37 INSERT INTO NURSES (empid, certification) VALUES (2, 'Registered Nurse'), (5, 'Li... 9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0
57 12:16:58 INSERT INTO HCP (empid, PRACTICE_AREA) VALUES (6, 'Physical Therapist'), (1... 4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0
58 12:17:17 INSERT INTO ADMIN_STAFF (empid, JOB_TYPE) VALUES (3, 'Receptionist'), (9, '... 6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0
Context Help
Snippets
Session

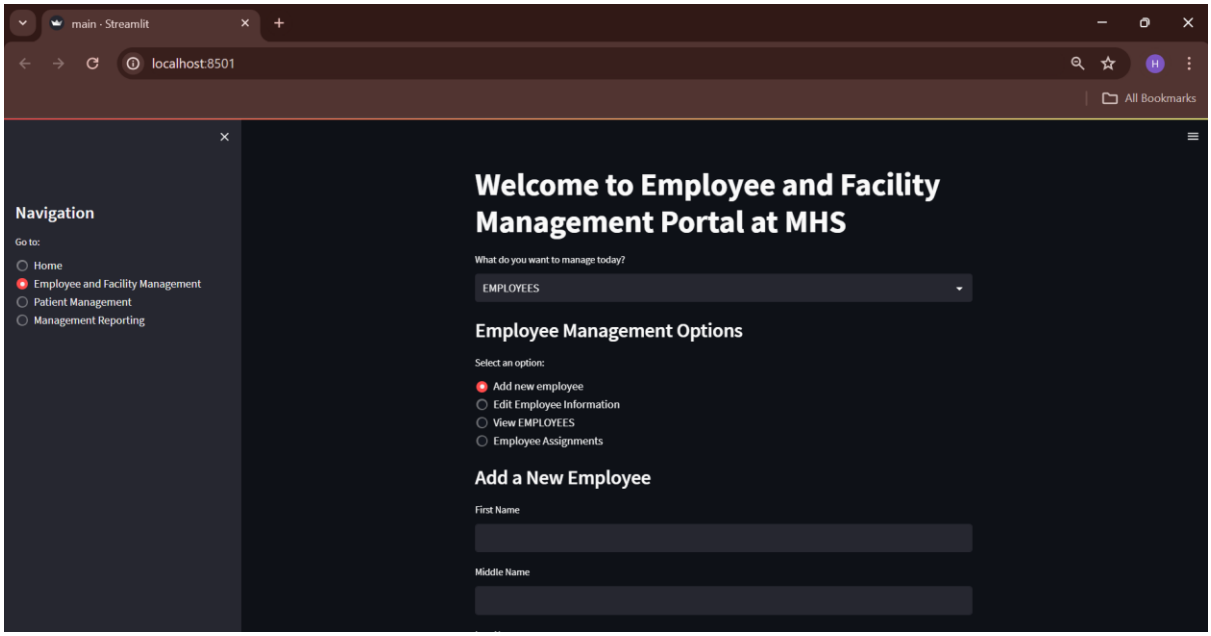

docker
Edit View Query Database Server Tools Scripting Help
insert_admin_staff
insert_employee_facilties
insert_insurance_companies
insert_pateints
insert_appointments
insert_invoices
SQLAdditions
Limit to 1000 rows
-- sample data for invoices table
INSERT INTO INVOICES (inv_date, amount, appt_id) VALUES
('2024-05-25', 100.00, 1),
('2024-05-28', 150.00, 2),
('2024-05-30', 200.00, 3),
('2024-06-02', 75.00, 4),
('2024-06-05', 120.00, 5),
('2024-06-08', 90.00, 6),
('2024-06-10', 180.00, 7),
('2024-06-12', 95.00, 8),
Filter objects
MHS
Tables
Views
Stored Procedures
Functions
sys
Schema: MHS
Schemas
Output
Action Output
60 12:18:17 INSERT INTO INSURANCE_COMPANIES (company, street, city, state, zip) VALU... 10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0 0.016 sec
61 12:18:38 INSERT INTO PATIENTS (first_name, middle_name, last_name, street, city, state, ... 40 row(s) affected Records: 40 Duplicates: 0 Warnings: 0 0.032 sec
62 12:18:54 INSERT INTO APPOINTMENTS (appt_date_time, appt_description, patient_id, ph... 20 row(s) affected Records: 20 Duplicates: 0 Warnings: 0 0.016 sec
63 12:19:12 INSERT INTO INVOICES (inv_date, amount, appt_id) VALUES ('2024-05-25', 100.... Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('... 0.016 sec
64 12:21:30 select * from appointments LIMIT 0, 1000 Error Code: 1146. Table 'MHS.appointments' doesn't exist 0.000 sec
65 12:21:50 select * from APPOINTMENTS LIMIT 0, 1000 20 row(s) returned 0.000 sec
66 12:23:37 INSERT INTO INVOICES (inv_date, amount, appt_id) VALUES ('2024-05-25', 100.... 20 row(s) affected Records: 20 Duplicates: 0 Warnings: 0 0.015 sec
67 12:23:37 select * from APPOINTMENTS LIMIT 0, 1000 20 row(s) returned 0.000 sec
Context Help
Snippets
Session

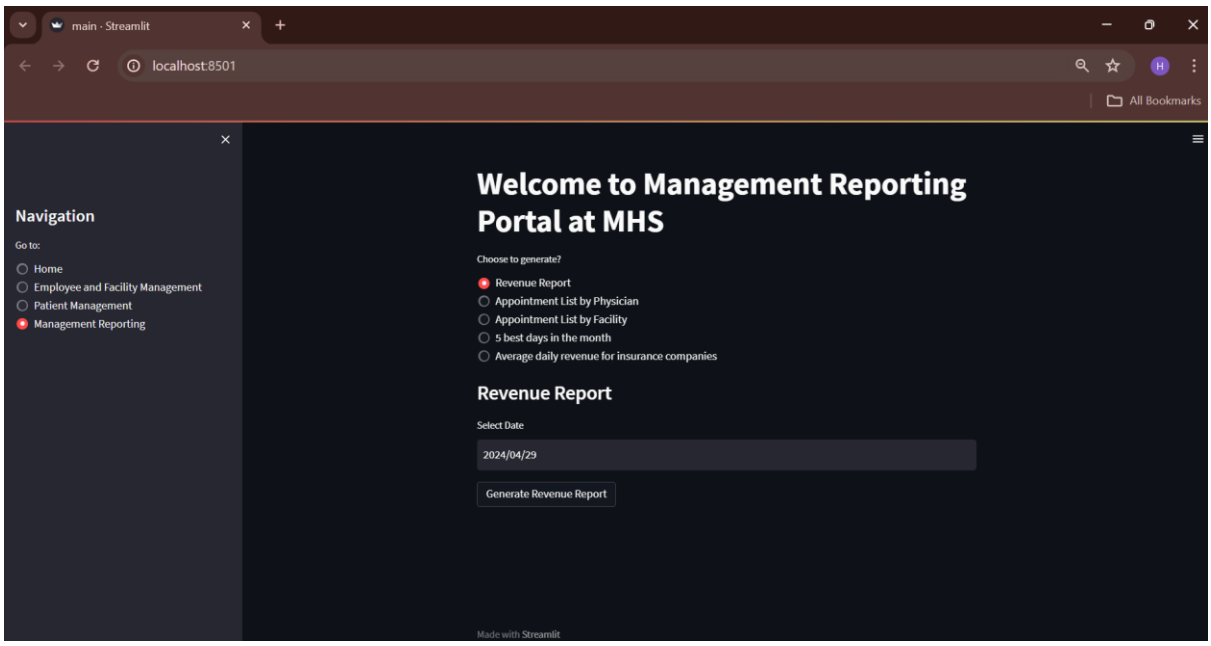
STREAMLIT EXECUTION

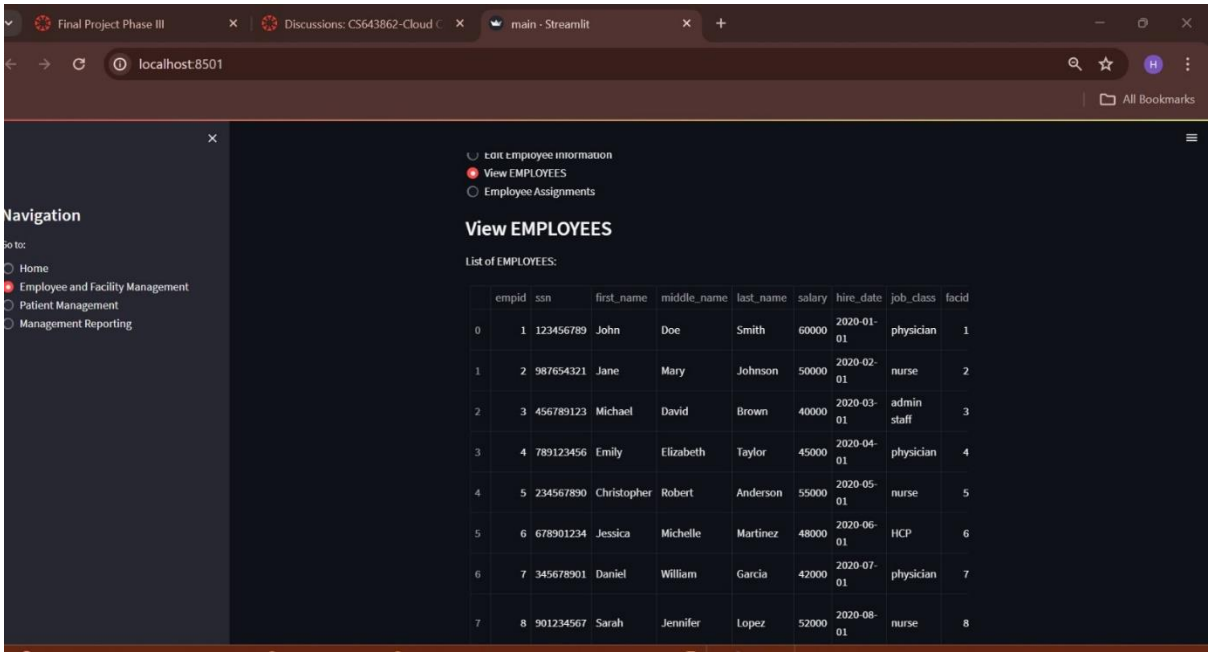


APPLICATION OUTPUT

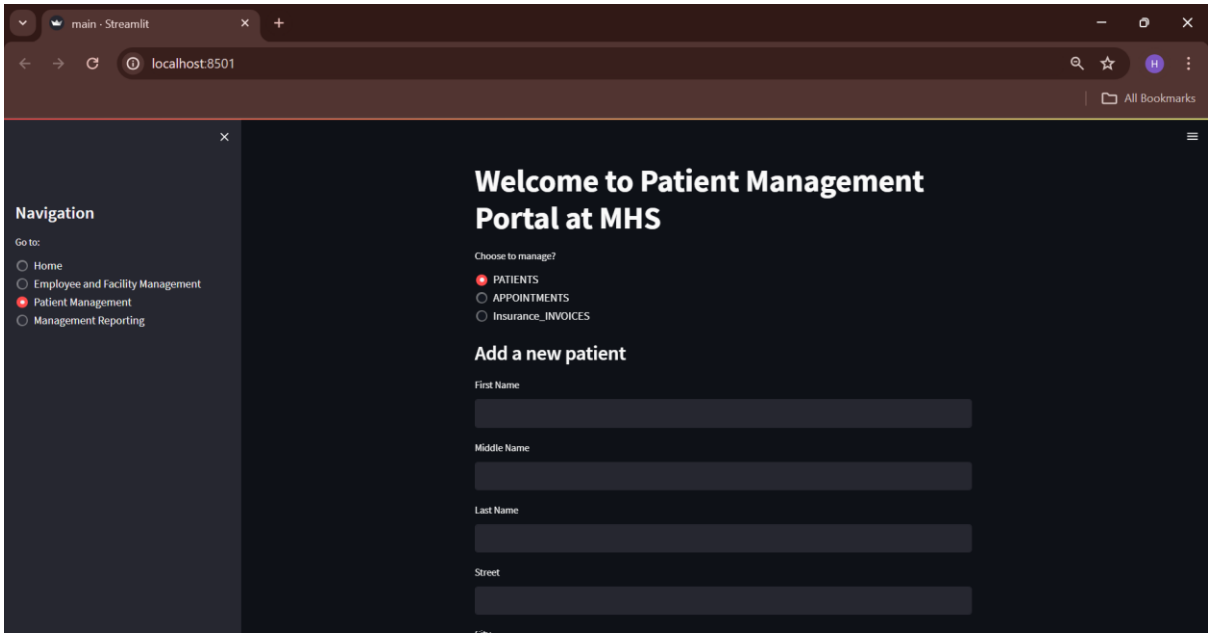


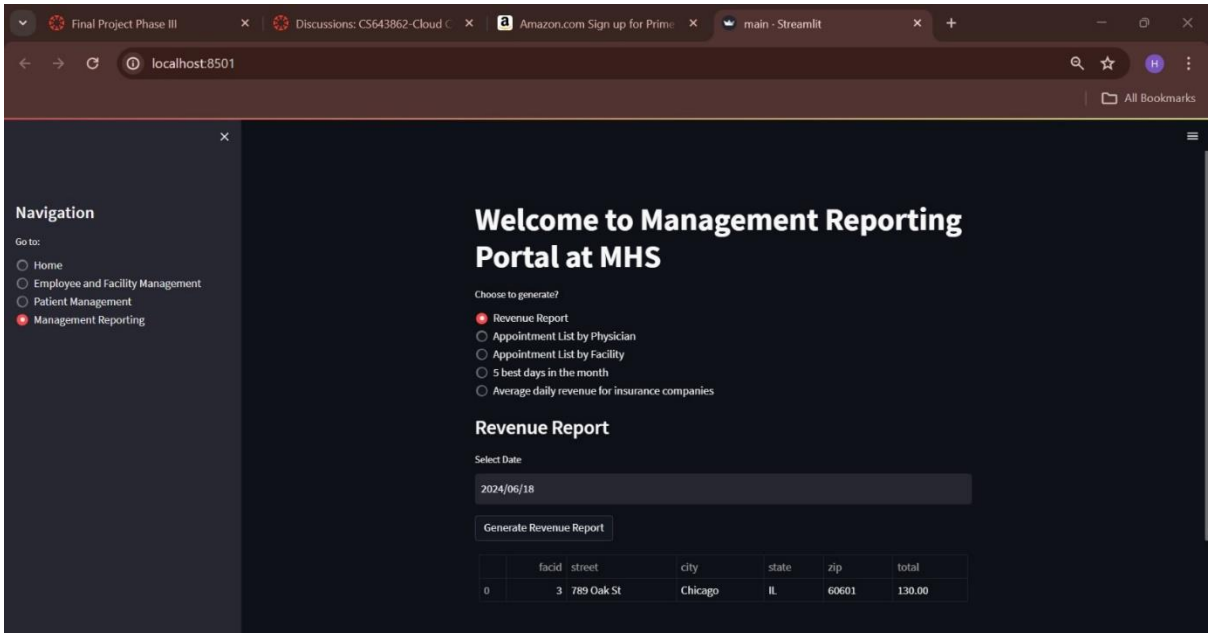

main · Streamlit
localhost:8501
All Bookmarks
Navigation
Go to:
Home
Employee and Facility Management
Patient Management
Management Reporting
Welcome to Employee and Facility Management Portal at MHS
What do you want to manage today?
EMPLOYEES
Employee Management Options
Select an option:
Add new employee
Edit Employee Information
View EMPLOYEES
Employee Assignments
Add a New Employee
First Name
Middle Name

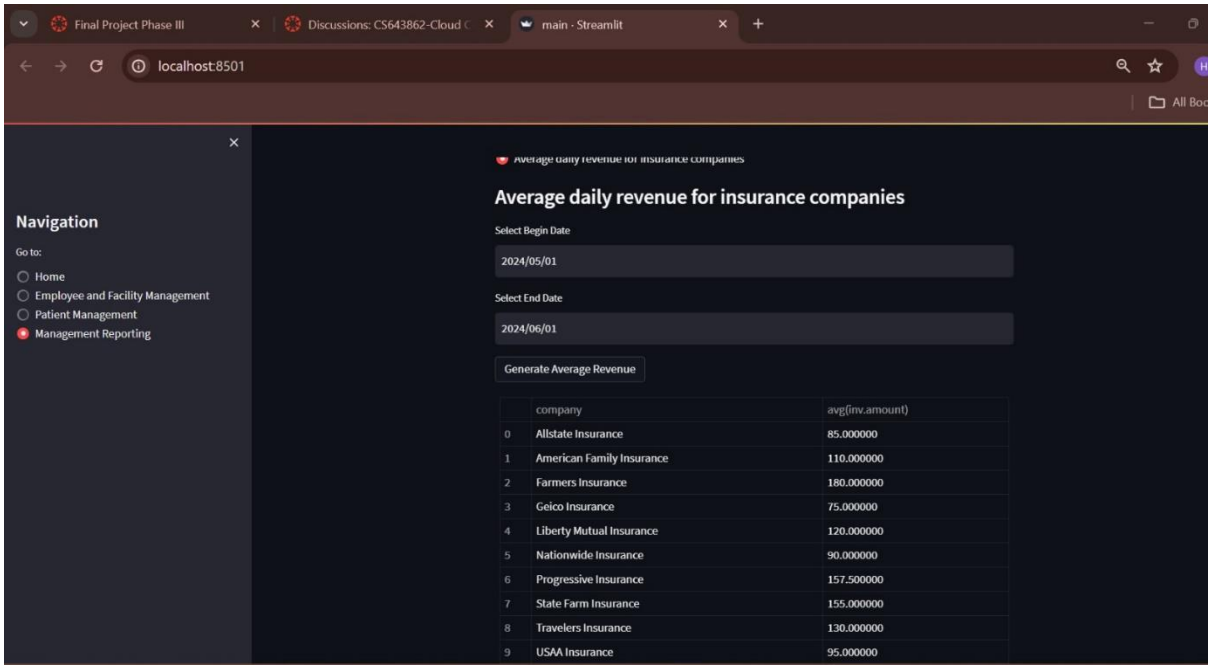

main · Streamlit
localhost:8501
All Bookmarks
Navigation
Go to:
Home
Employee and Facility Management
Patient Management
Management Reporting
Welcome to Management Reporting Portal at MHS
Choose to generate?
Revenue Report
Appointment List by Physician
Appointment List by Facility
5 best days in the month
Average daily revenue for insurance companies
Revenue Report
Select Date
2024/04/29
Generate Revenue Report
Made with Streamlit

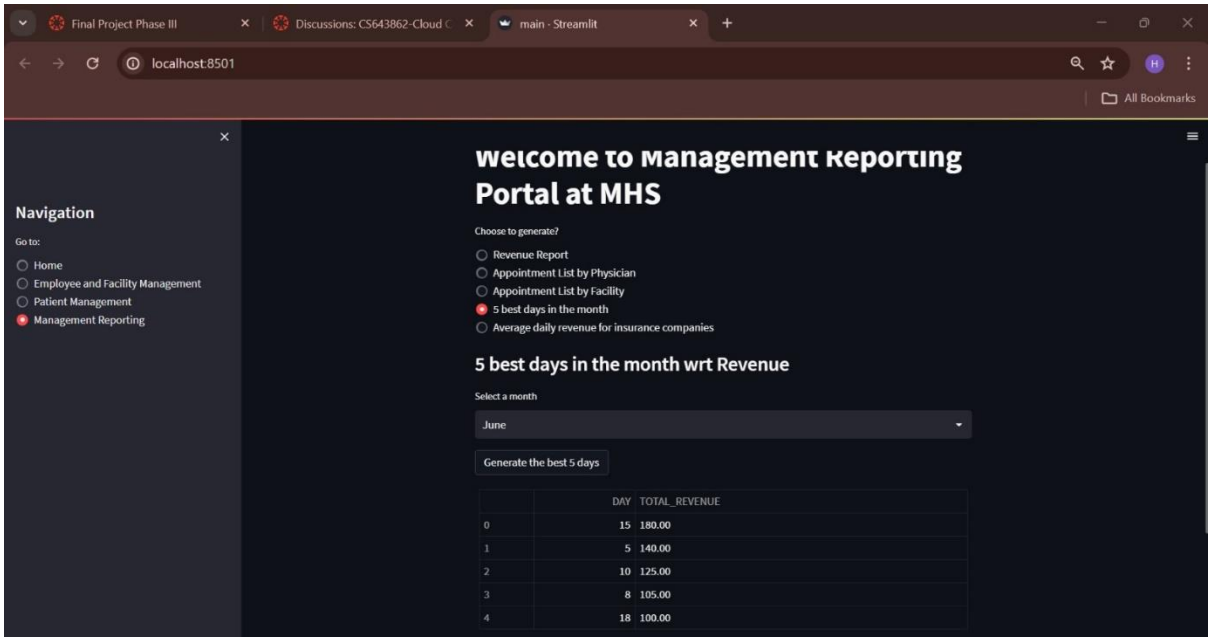


	empid	ssn	first_name	middle_name	last_name	salary	hire_date	job_class	facid
0	1	123456789	John	Doe	Smith	60000	2020-01-01	physician	1
1	2	987654321	Jane	Mary	Johnson	50000	2020-02-01	nurse	2
2	3	456789123	Michael	David	Brown	40000	2020-03-01	admin staff	3
3	4	789123456	Emily	Elizabeth	Taylor	45000	2020-04-01	physician	4
4	5	234567890	Christopher	Robert	Anderson	55000	2020-05-01	nurse	5
5	6	678901234	Jessica	Michelle	Martinez	48000	2020-06-01	HCP	6
6	7	345678901	Daniel	William	Garcia	42000	2020-07-01	physician	7
7	8	901234567	Sarah	Jennifer	Lopez	52000	2020-08-01	nurse	8




Final Project Phase III
Discussions: CS643862-Cloud C
Amazon.com Sign up for Prime
main · Streamlit
localhost:8501
All Bookmarks
Navigation
Go to:
Home
Employee and Facility Management
Patient Management
Management Reporting
Welcome to Management Reporting Portal at MHS
Choose to generate?
Revenue Report
Appointment List by Physician
Appointment List by Facility
5 best days in the month
Average daily revenue for insurance companies
Revenue Report
Select Date
2024/06/18
Generate Revenue Report
facid street city state zip total
0 3 789 Oak St Chicago IL 60601 130.00


Final Project Phase III
Discussions: CS643862-Cloud C
main · Streamlit
localhost:8501
Navigation
Go to:
Home
Employee and Facility Management
Patient Management
Management Reporting
Average daily revenue for insurance companies
Select Begin Date
2024/05/01
Select End Date
2024/06/01
Generate Average Revenue
company avg(inv.amount)
0 Allstate Insurance 85.000000
1 American Family Insurance 110.000000
2 Farmers Insurance 180.000000
3 Geico Insurance 75.000000
4 Liberty Mutual Insurance 120.000000
5 Nationwide Insurance 90.000000
6 Progressive Insurance 157.500000
7 State Farm Insurance 155.000000
8 Travelers Insurance 130.000000
9 USAA Insurance 95.000000


Final Project Phase III
Discussions: CS643862-Cloud C
main - Streamlit
localhost:8501
All Bookmarks
Navigation
Go to:
Home
Employee and Facility Management
Patient Management
Management Reporting
Welcome to Management Reporting Portal at MHS
Choose to generate?
Revenue Report
Appointment List by Physician
Appointment List by Facility
5 best days in the month
Average daily revenue for insurance companies
5 best days in the month wrt Revenue
Select a month
June
Generate the best 5 days
DAY TOTAL_REVENUE
0 15 180.00
1 5 140.00
2 10 125.00
3 8 105.00
4 18 100.00