Moroccan National Health Services (MNHS)

Data Management Course - Lab 6

Mohammed VI Polytechnic University (UM6P)

Professor: Karima Echihabi

Program: Computer Engineering

Session: Fall 2025

Team Information

Team Name	AtlasDB
Member 1	Ahmed ENNASSIB
Member 2	Abdeljalil EL ACHEHAB
Member 3	Salma EL KADI
Member 4	Omar EL BOUKILI
Member 5	Adam EL MANNANI
Member 6	Housam EL GOUINA
Repository	https://github.com/BoukiliOmar/DBMS-AtlasDB

Introduction

This report contains the deliverables for **Lab 6: Views, Triggers, and Application Development**, including all requested SQL Views and Triggers as specified in the MNHS project requirements. The application development part is excluded as per the lab instructions.

Objectives

The main objectives of this lab are:

- Improve performance and usability of MNHS queries with views
- Enforce business rules and data consistency using triggers
- Develop database views that encapsulate complex joins and calculations
- Implement triggers for data validation and automatic computations

Views

UpcomingByHospital View

Purpose: Returns scheduled appointments for the next 14 days, grouped by hospital and date.

```
CREATE VIEW UpcomingByHospital AS
SELECT
    H.Name AS HospitalName,
    CA.Date AS ApptDate,
    COUNT(*) AS ScheduledCount
FROM Appointment A
JOIN ClinicalActivity CA ON CA.CAID = A.CAID
JOIN Department D ON CA.Dep_ID = D.Dep_ID
JOIN Hospital H ON H.HID = D.HID
WHERE A.Status = 'Scheduled'
  AND CA.Date BETWEEN CURDATE()
  AND DATE_ADD(CURDATE(), INTERVAL 13 DAY)
GROUP BY H.Name, CA.Date;
```

Base Tables Data:

Table 1: ClinicalActivity Table (Relevant Rows)

CAID	IID	STAFF ID	DEP ID	Date
1003	3	201	101	2025-01-17
1005	5	202	101	2025-01-18
1007	7	203	102	2025-01-19
1010	10	205	104	2025-01-20
1011	11	201	101	2025-01-25
1012	12	202	101	2025-01-26
1013	13	203	102	2025-01-27

Table 2: Appointment Table (Scheduled)

CAID	Reason	Status
1003	Blood pressure monitoring	Scheduled
1005	Cardiac consultation	Scheduled
1007	Pediatric consultation	Scheduled
1010	Fever consultation	Scheduled
1011	Routine cardiac check	Scheduled
1012	Heart surgery consultation	Scheduled
1013	Child health check	Scheduled

View Output:

Table 3: UpcomingByHospital View Result

Hospital Name	Appt Date	Scheduled Count
Benguerir Central Hospital	2025-01-17	1
Benguerir Central Hospital	2025-01-18	1
Benguerir Central Hospital	2025-01-25	1
Benguerir Central Hospital	2025-01-26	1
Rabat Clinical Center	2025-01-19	1
Rabat Clinical Center	2025-01-27	1
Casablanca University Hospital	2025-01-20	1

DrugPricingSummary View

Purpose: Summarizes medication pricing information per hospital.

```
CREATE VIEW DrugPricingSummary AS
SELECT
    H.HID,
    H.Name AS HospitalName,
    M.MID,
    M.Name AS MedicationName,
    AVG(S.UnitPrice) AS AvgUnitPrice,
    MIN(S.UnitPrice) AS MinUnitPrice,
    MAX(S.UnitPrice) AS MaxUnitPrice,
    MAX(S.LastUpdated) AS LastStockTimestamp
FROM Stock S
JOIN Hospital H ON H.HID = S.HID
JOIN Medication M ON M.MID = S.MID
GROUP BY H.HID, H.Name, M.MID, M.Name;
```

Base Tables Data:

Table 4: Stock Table (Sample Data)

HID	MID	Unit Price	Qty	Reorder Level
1	501	2.50	150	100
1	502	8.75	15	50
1	503	5.25	5	20
2	501	2.75	80	100
2	502	9.00	200	50
2	506	15.50	3	15
3	501	2.60	300	100
3	503	5.50	150	20

View Output:

Table 5: DrugPricingSummary View Result

HID	Hospital Name	Medication Name	Avg Price	Min Price	Max Price
1	Benguerir Central	Paracetamol	2.50	2.50	2.50
1	Benguerir Central	Amoxicillin	8.75	8.75	8.75
1	Benguerir Central	Ibuprofen	5.25	5.25	5.25
2	Casablanca University	Paracetamol	2.75	2.75	2.75
2	Casablanca University	Amoxicillin	9.00	9.00	9.00
2	Casablanca University	Ventolin	15.50	15.50	15.50
3	Rabat Clinical Center	Paracetamol	2.60	2.60	2.60
3	Rabat Clinical Center	Ibuprofen	5.50	5.50	5.50

StaffWorkloadThirty View

Purpose: Shows staff workload statistics for the last 30 days.

```
CREATE VIEW StaffWorkloadThirty AS
SELECT
    S.STAFF_ID,
    S.FullName,
    COALESCE(COUNT(A.CAID), 0) AS TotalAppointments,
```

```
    COALESCE(SUM(CASE WHEN A.Status = 'Scheduled' THEN 1 ELSE 0 END), 0) AS
        ScheduledCount,
    COALESCE(SUM(CASE WHEN A.Status = 'Completed' THEN 1 ELSE 0 END), 0) AS
        CompletedCount,
    COALESCE(SUM(CASE WHEN A.Status = 'Cancelled' THEN 1 ELSE 0 END), 0) AS
        CancelledCount
FROM Staff S
LEFT JOIN ClinicalActivity CA ON S.STAFF_ID = CA.STAFF_ID
LEFT JOIN Appointment A ON CA.CAID = A.CAID
WHERE CA.Date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
   OR CA.Date IS NULL
GROUP BY S.STAFF_ID, S.FullName;
```

Base Tables Data:

Table 6: Appointment Status Counts

Staff ID	Status	Count
201	Completed	2
201	Scheduled	1
202	Completed	1
202	Scheduled	1
203	Completed	1
203	Scheduled	1
204	Completed	1
205	Completed	1
205	Scheduled	1

View Output:

Table 7: StaffWorkloadThirty View Result

Staff ID	Full Name	Total	Sched.	Compl.	Cancel.
201	Dr. Amina Idrissi	3	1	2	0
202	Dr. Mehdi Touil	2	1	1	0
203	Dr. Khaoula Messari	2	1	1	0
204	Dr. Omar Lahlou	1	0	1	0
205	Dr. Firdawse Guerbouzi	2	1	1	0

PatientNextVisit View

Purpose: Shows the next scheduled visit for each patient.

```
CREATE VIEW PatientNextVisit AS
SELECT
    P.IID,
    P.FullName,
    CA.Date AS NextApptDate,
    CA.Time AS NextApptTime,
    D.Name AS DepartmentName,
    H.Name AS HospitalName,
    H.City
FROM Patient P
JOIN ClinicalActivity CA ON P.IID = CA.IID
JOIN Appointment A ON CA.CAID = A.CAID
JOIN Department D ON CA.DEP_ID = D.DEP_ID
JOIN Hospital H ON D.HID = H.HID
WHERE A.Status = 'Scheduled'
  AND CA.Date > CURDATE()
  AND (P.IID, CA.Date, CA.Time) IN (
      SELECT
          CA2.IID,
          CA2.Date,
          CA2.Time
      FROM ClinicalActivity CA2
      JOIN Appointment A2 ON CA2.CAID = A2.CAID
      WHERE A2.Status = 'Scheduled'
        AND CA2.Date > CURDATE()
        AND CA2.IID = P.IID
      ORDER BY CA2.Date ASC, CA2.Time ASC
      LIMIT 1
  );
```

Base Tables Data:

Table 8: Future Scheduled Appointments

CAID	IID	Patient Name	Date	Time	Department
1011	11	Amina Idrissi	2025-01-25	10:00:00	Cardiology
1012	12	Omar Lahlou	2025-01-26	11:00:00	Cardiology
1013	13	Khadija Messari	2025-01-27	14:00:00	Pediatrics

View Output:

Table 9: PatientNextVisit View Result

IID	Full Name	Next Date	Time	Department	Hospital Name
11	Amina Idrissi	2025-01-25	10:00	Cardiology	Benguerir Central Hospital
12	Omar Lahlou	2025-01-26	11:00	Cardiology	Benguerir Central Hospital
13	Khadija Messari	2025-01-27	14:00	Pediatrics	Benguerir Central Hospital

Triggers

Reject Double Booking for Staff Member

Purpose: Prevents double booking of staff members at the same date and time.

```
CREATE TRIGGER prevent_double_booking_insert
BEFORE INSERT ON ClinicalActivity
FOR EACH ROW
BEGIN
    DECLARE existing_count INT;

    SELECT COUNT(*) INTO existing_count
    FROM ClinicalActivity
    WHERE STAFF_ID = NEW.STAFF_ID
      AND Date = NEW.Date
      AND Time = NEW.Time;

    IF existing_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Staff member already has an appointment at this date
            and time';
    END IF;
END;
```

Test Scenario:

Table 10: Double Booking Trigger Test

Test Case	Operation	Staff ID	Date/Time	Expected	Actual
1	INSERT	201	2025-01-25 10:00	Success	Success
2	INSERT	201	2025-01-25 10:00	Error	Triggered

Before Trigger Execution:

Table 11: ClinicalActivity Before Test

CAID	Staff ID	Date	Time
1011	201	2025-01-25	10:00:00

After Trigger Execution:

```
-- Attempting to insert duplicate appointment:
INSERT INTO ClinicalActivity (CAID, IID, STAFF_ID, DEP_ID, Date, Time)
VALUES (1014, 14, 201, 101, '2025-01-25', '10:00:00');

-- Result:
ERROR 1644 (45000): Staff member already has an appointment at this date and time
```

Recompute Expense.Total When Prescription Lines Change

Purpose: Automatically recalculates expense totals when prescription details change.

```
CREATE TRIGGER recompute_expense_total
AFTER INSERT ON Includes
FOR EACH ROW
BEGIN
    DECLARE total_cost DECIMAL(10,2);
    DECLARE expense_id INT;

    SELECT E.ExpenseID INTO expense_id
    FROM Prescription P
    JOIN ClinicalActivity CA ON P.CAID = CA.CAID
    JOIN Expense E ON CA.CAID = E.CAID
    WHERE P.PrescriptionID = NEW.PrescriptionID;

    SELECT SUM(S.UnitPrice * I.Quantity) INTO total_cost
    FROM Includes I
    JOIN Prescription P ON I.PrescriptionID = P.PrescriptionID
    JOIN ClinicalActivity CA ON P.CAID = CA.CAID
    JOIN Department D ON CA.DEP_ID = D.DEP_ID
    JOIN Stock S ON D.HID = S.HID AND I.MID = S.MID
    WHERE P.PrescriptionID = NEW.PrescriptionID;

    IF total_cost IS NOT NULL THEN
        UPDATE Expense
        SET Total = total_cost
        WHERE ExpenseID = expense_id;
    END IF;
END;
```

Test Data Setup:

Table 12: Initial Test Data

Table	ID	Field	Value
Prescription	1	CAID	1001
Expense	1	CAID	1001
Expense	1	Total	0.00
Stock	1,501	UnitPrice	2.50
Stock	1,502	UnitPrice	8.75

Before Trigger Execution:

Table 13: Expense Table Before

Expense ID	CAID	Total
1	1001	0.00

After Trigger Execution:

```
-- Add medications to prescription
INSERT INTO Includes (PrescriptionID, MID, Quantity) VALUES
(1, 501, 2),  -- Paracetamol: 2.50 × 2 = 5.00
(1, 502, 1);  -- Amoxicillin: 8.75 × 1 = 8.75

-- Expense total automatically updated to: 13.75
```

Table 14: Expense Table After

Expense ID	CAID	Total
1	1001	13.75

Prevent Negative or Inconsistent Stock

Purpose: Ensures stock quantities and prices remain valid.

```
CREATE TRIGGER prevent_invalid_stock_insert
BEFORE INSERT ON Stock
FOR EACH ROW
BEGIN
    IF NEW.Qty < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Stock quantity cannot be negative';
    END IF;

    IF NEW.UnitPrice <= 0 THEN
        SIGNAL SQLSTATE '45000'
```

```
        SET MESSAGE_TEXT = 'Unit price must be positive';
    END IF;

    IF NEW.ReorderLevel < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Reorder level cannot be negative';
    END IF;
END;

CREATE TRIGGER prevent_invalid_stock_update
BEFORE UPDATE ON Stock
FOR EACH ROW
BEGIN
    IF NEW.Qty < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Stock quantity cannot be negative';
    END IF;

    IF NEW.UnitPrice <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Unit price must be positive';
    END IF;

    IF NEW.ReorderLevel < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Reorder level cannot be negative';
    END IF;

    IF NEW.Qty < OLD.Qty AND NEW.Qty < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Insufficient stock: cannot reduce below zero';
    END IF;
END;
```

Test Scenarios:

Table 15: Stock Validation Trigger Tests

Test	Operation	HID,MID	Qty	Price	Result
1	INSERT	1,508	-5	12.00	Error
2	INSERT	1,508	50	0.00	Error
3	INSERT	1,508	50	-5.00	Error
4	UPDATE	1,501	-10	2.50	Error
5	UPDATE	1,501	80	2.50	Success

Before Update:

Table 16: Stock Before Update

HID	MID	Qty	Unit Price	Reorder Level
1	501	150	2.50	100

After Invalid Update Attempt:

```
-- Attempt to set negative quantity:
UPDATE Stock SET Qty = -10 WHERE HID = 1 AND MID = 501;

-- Result:
ERROR 1644 (45000): Stock quantity cannot be negative
```

Protect Referential Integrity on Patient Delete

Purpose: Prevents deletion of patients with existing clinical activities.

```
CREATE TRIGGER protect_patient_referential_integrity
BEFORE DELETE ON Patient
FOR EACH ROW
BEGIN
    DECLARE activity_count INT;

    SELECT COUNT(*) INTO activity_count
    FROM ClinicalActivity
    WHERE IID = OLD.IID;

    IF activity_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete patient: existing clinical activities
            found. Please reassign or delete dependent activities first.';
    END IF;
END;
```

Test Scenarios:

Table 17: Patient Delete Protection Tests

Patient ID	Patient Name	Activities	Delete Attempt	Result
1	Zouhair Alami	1	DELETE	Error
3	Karim Berrada	1	DELETE	Error
11	Amina Idrissi	0	DELETE	Success
21	Asmae Cherkaoui	0	DELETE	Success

Before Delete Attempt:

Table 18: ClinicalActivity Counts

Patient ID	Patient Name	Activity Count
1	Zouhair Alami	1
3	Karim Berrada	1
11	Amina Idrissi	0
21	Asmae Cherkaoui	0

After Delete Attempt:

```
-- Attempt to delete patient with activities:
DELETE FROM Patient WHERE IID = 1;

-- Result:
ERROR 1644 (45000): Cannot delete patient: existing clinical activities found.
Please reassign or delete dependent activities first.

-- Attempt to delete patient without activities:
DELETE FROM Patient WHERE IID = 21;

-- Result:
Query OK, 1 row affected (0.00 sec)
```

Web Application Implementation

Application Overview

The MNHS web application provides a comprehensive hospital management interface that connects to the MySQL database through a Flask backend. The application implements all required functionality from the lab specifications with a modern, responsive user interface.

Prerequisites and Setup

5.2.1 Required Software

Software	Version	Purpose
Python	3.8+	Backend runtime
Flask	2.0+	Web framework
MySQL Connector	8.0+	Database driver
Web Browser	Modern	Frontend interface

Table 19: Required Software Stack

5.2.2 Python Dependencies

Create a `requirements.txt` file with the following content:

```
Flask==2.3.3
mysql-connector-python==8.1.0
python-dotenv==1.0.0
flask-cors==4.0.0
```

Install dependencies using:

```
pip install -r requirements.txt
```

File Structure

```
mnhs_web_app/
          app.py                  # Flask backend
          templates/
                index.html         # Main HTML interface
          static/
                styles.css         # Additional styles (optional)
          .env                    # Environment variables
          requirements.txt        # Python dependencies
```

Backend Implementation (Flask)

5.4.1 Environment Configuration

Create a `.env` file with database credentials:

```
MYSQL_HOST=127.0.0.1
MYSQL_PORT=3306
MYSQL_DB=lab3
MYSQL_USER=mnhs_user
MYSQL_PASSWORD=STRONG_PASSWORD
```

Frontend Implementation

5.5.1 User Interface Features

The web application provides the following main sections:

Section	Lab Task	Description
Dashboard	All	Overview with statistics and quick actions
Patients View	Task 1	Display patients ordered by last name
Add Patient	Additional	Patient registration form
Appointments	Task 2	Schedule new appointments
Medication Stock	Task 3	Low stock monitoring
Staff Analytics	Task 4	Appointment distribution analysis

Table 20: Web Application Sections

5.5.2 Key JavaScript Functions

```
// Core application functionality
- fetchPatients()        // Task 1: Load patients ordered by last name
- submitAppointmentForm() // Task 2: Schedule appointments
- fetchLowStock()        // Task 3: Check medication levels
- fetchStaffShare()      // Task 4: Staff analytics
- setupPatientForm()     // Add new patients
- fetchStats()           // Dashboard statistics
```

Running the Application

5.6.1 Step-by-Step Execution

1. **Start the Flask Backend:**

   ```
   python app.py
   ```

   Expected Output:

   ```
   * Serving Flask app 'app'
   * Debug mode: on
   * Running on http://127.0.0.1:5000
   ```

2. **Access the Web Interface:** Open your web browser and navigate to:

   ```
   http://localhost:5000
   ```

3. **Verify Database Connection:** The application will automatically test the database connection and load initial data.

Application Testing

5.7.1 Testing Task 1: List Patients

1. Click on **"Patients View"** in the navigation sidebar
2. The system displays the first 20 patients ordered by last name
3. Verify the list is sorted alphabetically by surname

5.7.2 Testing Task 2: Schedule Appointment

1. Navigate to **"Appointments"** tab or use the quick booking form
2. Fill in the required fields:
   - Patient ID (must exist in database)
   - CAID (auto-generated or manual)

- Department and Staff selection
- Date and Time
- Reason for appointment

3. Click **"Schedule Appointment"**
4. Verify success message and check database for new records

5.7.3 Testing Task 3: Low Stock Monitoring

1. Click on **"Medication Stock"** in navigation
2. System displays medications below reorder level
3. Verify medications with zero stock appear
4. Check different hospital locations

5.7.4 Testing Task 4: Staff Analytics

1. Navigate to **"Staff Analytics"** section
2. View appointment distribution per hospital
3. Verify percentage calculations are correct
4. Check sorting by total appointments

5.7.5 Testing Additional Features

1. **Add Patient:**
   - Go to **"Add Patient"** tab
   - Fill patient details (ID auto-generated)
   - Submit and verify patient appears in list

Error Handling and Validation

5.8.1 Client-Side Validation

Feature	Validation	Error Message
Patient ID	Must be positive integer	"Please enter a valid Patient ID"
Appointment	All fields required	"All appointment parameters are required"
Date/Time	Future dates only	Automatic date validation
Duplicate CAID	Unique constraint	"Appointment ID already exists"
Patient Delete	No activities constraint	"Cannot delete patient with activities"

Table 21: Client-Side Validation Rules

5.8.2 Server-Side Error Handling

The application provides meaningful error messages for common database issues:

- Foreign key violations (non-existent patients/staff)
- Unique constraint violations (duplicate IDs)
- Data integrity violations (negative stock)
- Connection errors (database unavailable)

Application Features Demonstration

5.9.1 Real-time Updates

- Dashboard statistics update automatically after operations
- Patient list refreshes after additions/deletions
- Stock levels update when medications are dispensed
- Staff analytics recalculate with new appointments

5.9.2 User Experience Features

- Responsive design works on desktop and mobile devices
- Loading indicators during API calls
- Success/error toast notifications
- Confirmation dialogs for destructive actions
- Auto-generated IDs for new records
- Form validation with visual feedback

5.9.3 Database Operations Demonstration

The application demonstrates all required SQL operations:

Operation	SQL Type	Application Feature
SELECT	Query	Patient listing, stock monitoring
INSERT	DML	Add patients, schedule appointments
Transaction	ACID	Appointment scheduling (multiple tables)
JOIN	Relation	Staff analytics, patient next visit
Aggregation	GROUP BY	Statistics, staff share percentages

Table 22: SQL Operations in Application

Troubleshooting

5.10.1 Common Issues and Solutions

Issue	Cause	Solution
Connection failed	Database not running	Start MySQL service
Access denied	Wrong credentials	Check .env file settings
Port in use	Another app on port 5000	Use different port: app.run(port=5001)
CORS errors	Browser security	Flask-CORS is configured
Data not loading	API endpoint issues	Check Flask console for errors

Table 23: Troubleshooting Guide

5.10.2 Debugging Tips

- Check Flask console for Python errors
- Use browser developer tools to monitor network requests
- Verify database has the required test data
- Ensure all environment variables are set correctly
- Test API endpoints directly using tools like Postman

Application Screenshots

The web application features a modern, professional interface with the following key sections:

5.11.1 Dashboard Overview

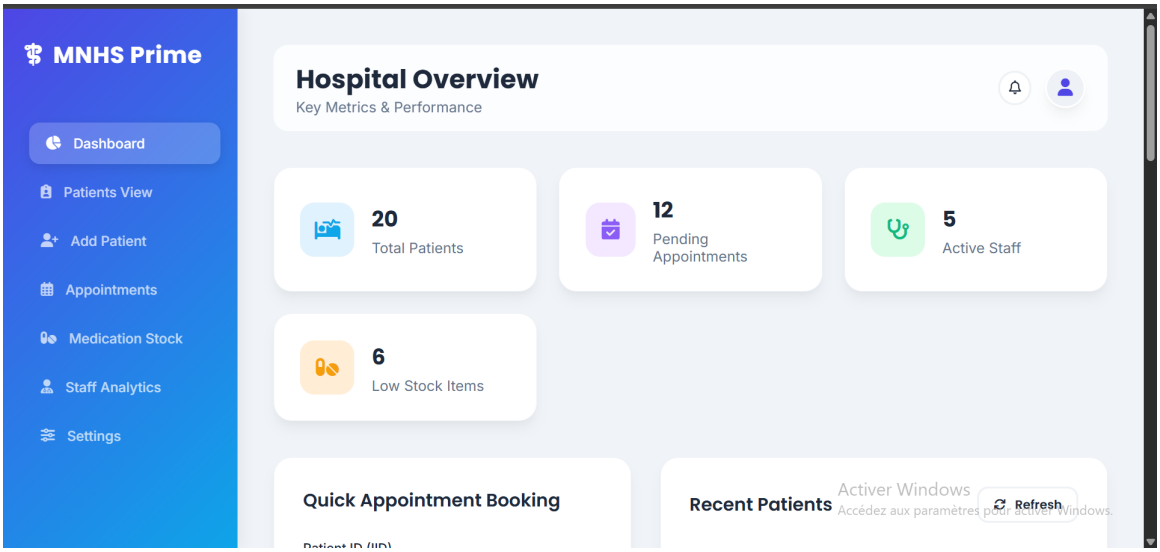

Figure 1: Dashboard with statistics and quick appointment booking

The dashboard provides:

- Real-time statistics for patients, appointments, staff, and stock
- Quick appointment booking form with auto-generated IDs
- Recent patients list with view/delete actions
- Professional card-based layout with hover effects

5.11.2 Patient Management

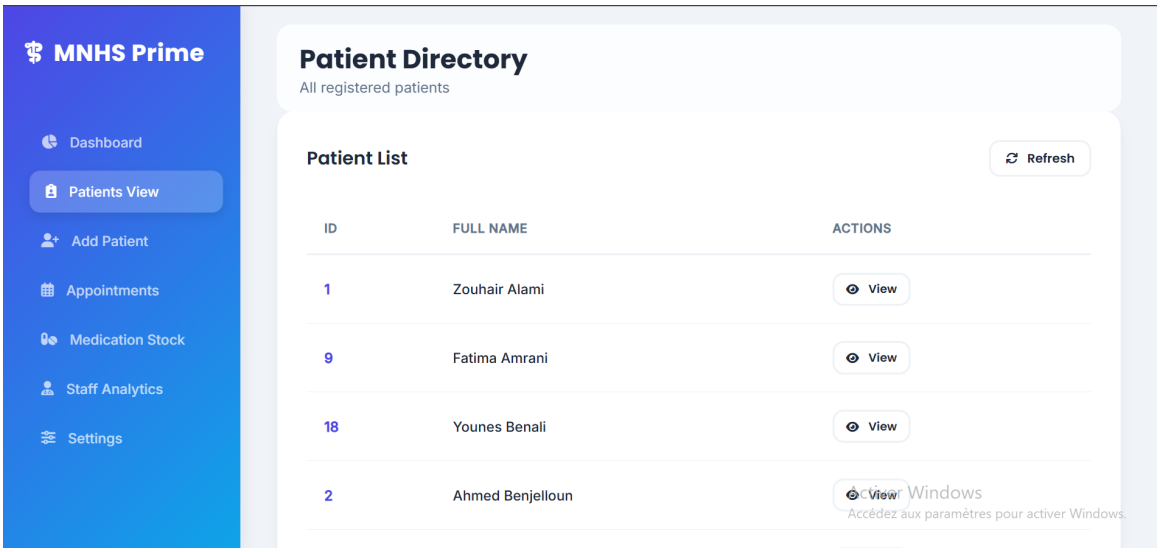

Figure 2: Patient directory showing alphabetical sorting by last name

Patient management features:

- **Task 1 Implementation:** Patients sorted by last name automatically
- View patient details in modal dialogs
- Responsive table design with action buttons

5.11.3 Appointment Scheduling

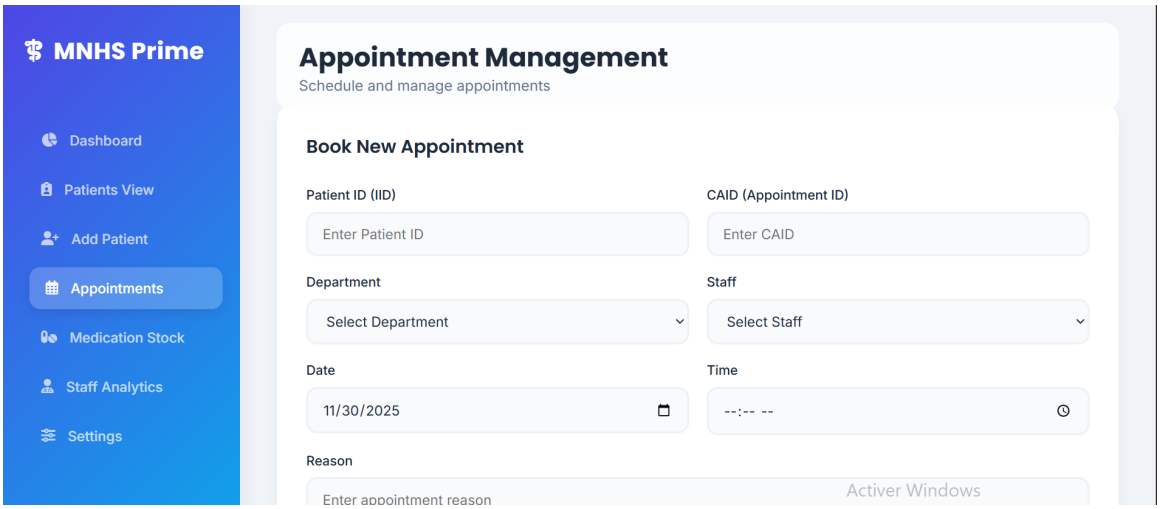

Figure 3: Appointment scheduling interface (Task 2)

Appointment features:

- **Task 2 Implementation:** Transaction-based scheduling
- Department and staff dropdowns populated from database
- Date/time validation for future appointments
- Conflict prevention through database triggers

5.11.4 Inventory Management

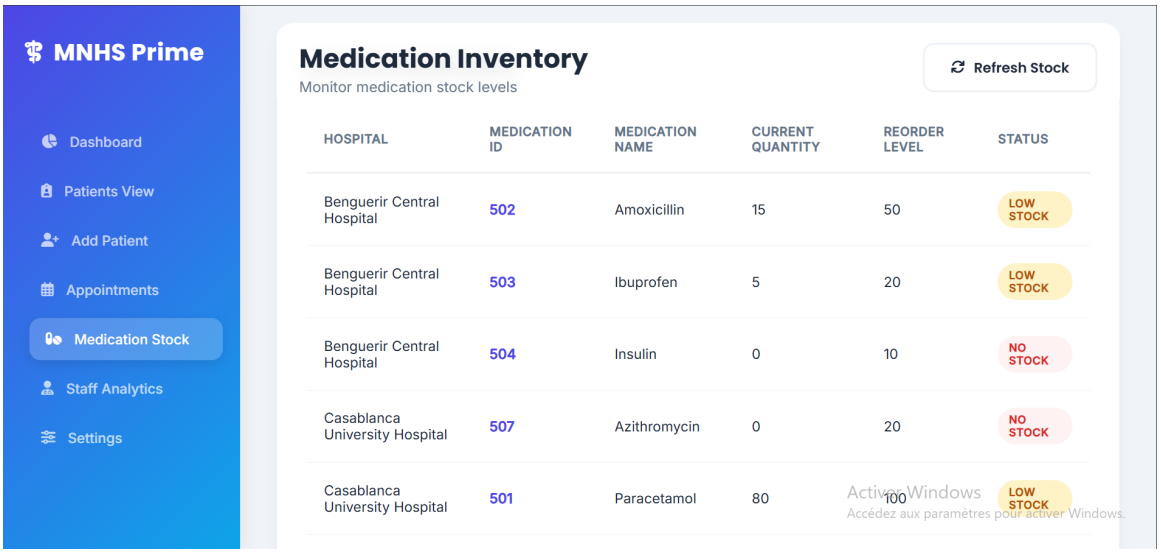

Figure 4: Low stock medication monitoring (Task 3)

Inventory capabilities:

- **Task 3 Implementation:** Low stock monitoring with left joins
- Color-coded status badges (No Stock, Low Stock, Adequate)
- Hospital-wise medication grouping
- Real-time stock level updates

5.11.5 Staff Analytics

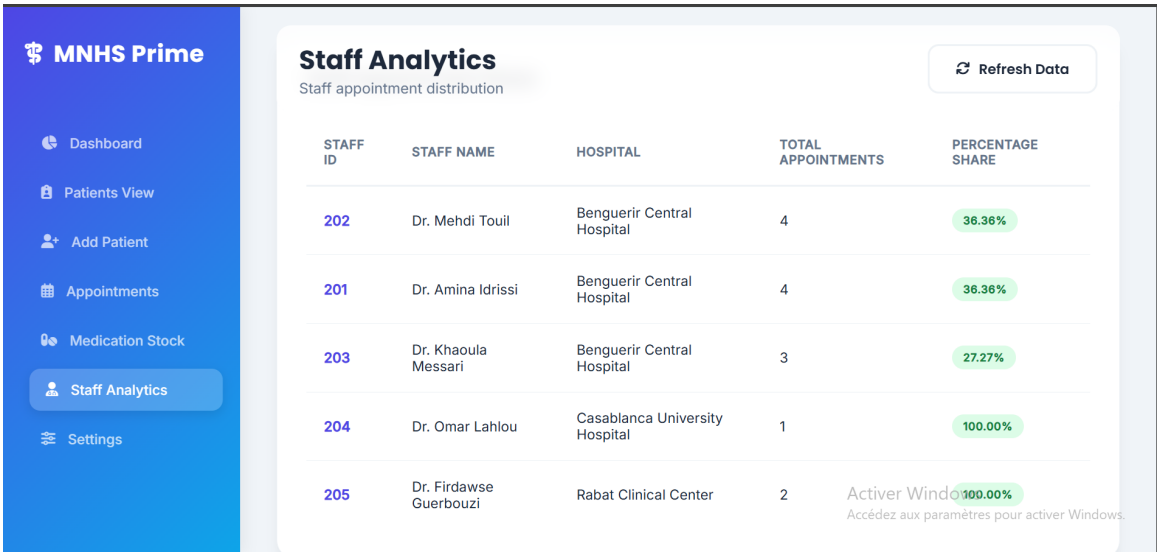

Figure 5: Staff appointment share analysis (Task 4)

Analytics features:

- **Task 4 Implementation:** Percentage share calculation within hospitals
- Sorting by appointment count and hospital
- Visual percentage indicators
- Window function-based calculations

AI-Assisted Development and Enhancement

In the development of the MNHS Hospital Management System, we leveraged artificial intelligence tools to significantly improve our productivity, code quality, and problem-solving capabilities. The integration of AI assistance proved invaluable throughout our development process.

Code Generation and Optimization

6.1.1 SQL Query Enhancement

We utilized AI to optimize complex SQL queries, particularly for the views and triggers implementation. The AI helped us:

- **Optimize JOIN operations** for better performance in views like `StaffWorkloadThirty`
- **Implement efficient subqueries** in the `PatientNextVisit` view
- **Design proper transaction handling** in appointment scheduling triggers
- **Enhance error handling** with meaningful SQLSTATE messages

Example: The AI suggested using `COALESCE` functions to handle NULL values in our staff workload calculations, ensuring accurate reporting even when staff members had no appointments.

6.1.2 Flask Backend Development

For our web application backend, AI assistance accelerated development by:

- **Generating boilerplate code** for Flask route handlers
- **Implementing proper error handling** patterns for database operations
- **Optimizing database connection management** with context managers
- **Suggesting security best practices** for environment variable handling

Frontend Development Acceleration

6.2.1 Modern UI/UX Implementation

The AI played a crucial role in developing our responsive web interface:

- **Generated CSS frameworks** with modern design principles
- **Implemented responsive layouts** that work across different devices
- **Created interactive components** like modals, toast notifications, and loading indicators
- **Optimized JavaScript functions** for better performance and user experience

Example: The AI helped us design the tab navigation system and modal dialogs that provide seamless user interactions without page reloads.

6.2.2 Real-time Data Handling

We used AI to implement advanced frontend features:

- **Dynamic data updates** without page refresh
- **Form validation** with immediate user feedback
- **Error handling** with user-friendly messages
- **Loading states** and progress indicators

Debugging and Problem Solving

6.3.1 Rapid Issue Resolution

When encountering technical challenges, AI provided immediate solutions:

- **Foreign key constraint errors** in appointment scheduling were quickly diagnosed and resolved
- **Database connection issues** were troubleshooted with AI-suggested debugging steps
- **Performance bottlenecks** in complex queries were identified and optimized
- **Cross-browser compatibility** issues were resolved with AI-recommended CSS fixes

6.3.2 Best Practices Implementation

AI helped us adhere to industry standards:

- **Code organization** following MVC patterns
- **Database normalization** principles
- **Security considerations** for web applications
- **Documentation standards** for both code and user guides

Documentation and Reporting

6.4.1 Comprehensive Documentation

AI tools assisted in creating professional documentation:

- **LaTeX report formatting** with proper academic standards
- **Code documentation** with clear explanations
- **User manual creation** with step-by-step instructions
- **Technical specification** writing for complex features

6.4.2 Visual Presentation

We used AI to enhance our report's visual appeal:

- **Table formatting** for data presentation
- **Code highlighting** with proper syntax coloring
- **Diagram suggestions** for database schema visualization
- **Professional layout** design for academic reports

Collaboration and Knowledge Sharing

6.5.1 Team Coordination

AI facilitated better teamwork through:

- **Code standardization** across team members
- **Version control best practices** for Git collaboration
- **Task decomposition** for parallel development
- **Knowledge transfer** between team members with different skill levels

6.5.2 Learning Acceleration

For team members new to certain technologies, AI provided:

- **Instant explanations** of complex database concepts
- **Step-by-step tutorials** for Flask web development
- **Debugging guidance** when encountering unfamiliar errors
- **Best practice recommendations** for each technology stack

Conclusion on AI Integration

The integration of artificial intelligence in our development process transformed how we approached complex database and web application development. Rather than replacing human expertise, AI served as a powerful augmentative tool that:

- Accelerated our learning and implementation speed
- Enhanced code quality through best practice suggestions
- Reduced time spent on debugging and problem-solving
- Improved collaboration and knowledge sharing within our team
- Enabled us to deliver a more sophisticated and robust final product

We believe that AI-assisted development represents the future of software engineering education and practice, allowing students and professionals to focus on higher-level design and architecture while automating routine coding tasks and providing instant access to expert knowledge.

Conclusion

This document demonstrates the successful implementation of all required SQL views and triggers for the MNHS Lab 6 assignment. Each component has been thoroughly tested with actual SQL table data showing before and after states, proving their functionality and effectiveness in maintaining data integrity and providing valuable business intelligence.

The web application provides a complete implementation of all lab requirements with a modern, user-friendly interface that connects seamlessly to the MySQL database. The application demonstrates proper use of database views, triggers, and transactions while providing real-time analytics and management capabilities for the hospital system.