College of Engineering

Northeastern University

Project Report

Healthcare Clinic Patient Management System

DAMG 7275 Advance Database Management System

TEAM 4

Name	NUID
Siddharth Bahekar	002417718
Rachita Shah	002482615
Chitra Periya	002893640
Vedant Ukarde	002206626

Contents

Problem Statement

Challenges and identified gaps in traditional clinic systems

Traditional clinic management systems struggle with manual record-keeping, fragmented data, and scheduling conflicts, leading to delays in patient care and increased administrative burden. Inefficient resource allocation, scattered medical records, and lack of integration between departments create communication gaps and hinder follow-up care. Additionally, the absence of automated scheduling and notification systems results in higher no-show rates and underutilized resources. Weak data security and non-standardized billing and insurance processes further complicate compliance and accuracy.

To address these issues, an integrated healthcare management system is crucial. The proposed solution will centralize patient registration, billing, and medical records while ensuring secure, real-time access to information. By automating routine tasks and standardizing workflows, it will improve care delivery, optimize resource use, enhance data security, and reduce administrative overhead, creating a more efficient and patient-focused system.

Vision

The Healthcare Patient Clinic Management System envisions transforming traditional healthcare administration into a streamlined, integrated digital ecosystem that enhances patient care delivery and operational efficiency. This comprehensive database solution aims to create a seamless connection between patients, healthcare providers, and administrative staff by centralizing patient information, automating routine tasks, and ensuring secure access to medical records. The system's vision encompasses improving appointment management, reducing administrative overhead, enhancing billing accuracy, and ultimately delivering a better healthcare experience through efficient data management and automated workflows. By implementing robust database architecture with Oracle SQL, the system strives to establish a reliable, scalable, and secure platform that not only meets current healthcare management needs but also adapts to evolving medical practices and regulatory requirements, ultimately contributing to improved patient outcomes and healthcare service delivery.

Requirement gathering

The requirement gathering process for the Healthcare Patient Clinic Management System followed a comprehensive and systematic approach to ensure all stakeholder needs were effectively captured and addressed. The process began with identifying key stakeholders across the healthcare ecosystem, including medical professionals (doctors and nurses), administrative staff, patients, laboratory technicians, billing department personnel, and pharmacy staff. This diverse stakeholder group provided valuable insights into the various aspects of healthcare service delivery and management.

Our data collection methodology involved a thorough analysis of existing healthcare systems to understand current practices and identify areas for improvement. We conducted detailed reviews of medical workflow processes to ensure the system would align with established healthcare protocols and procedures. Special attention was paid to healthcare regulations and compliance requirements, ensuring the system would meet all necessary standards for medical data management and patient privacy.

The core business requirements emerged from this analysis, focusing on six critical areas: patient information management, appointment scheduling and tracking, medical records management, billing and payment processing, prescription management, and laboratory test management. These requirements were carefully mapped to the database structure, resulting in 19 interconnected tables that capture all necessary data relationships and workflows.

The technical requirements were formulated to support these business needs, with Oracle SQL Database chosen as the implementation platform due to its robustness and reliability. Key technical considerations included ensuring data security and privacy compliance, implementing real-time data processing capabilities, establishing multi-user access control, and building a scalable system architecture that could grow with the organization's needs.

This structured approach to requirements gathering enabled us to develop a comprehensive database system that effectively addresses the complex needs of modern healthcare management while maintaining the flexibility to adapt to future requirements and regulatory changes.

System Requirement Analysis

The system requirements analysis for the Healthcare Patient Clinic Management System was conducted through a structured approach, encompassing functional, non-functional, data, and security requirements to ensure comprehensive coverage of all healthcare management aspects.

The functional requirements were identified to support core operational needs of the healthcare facility. These include a robust user authentication and authorization system to manage access control, comprehensive patient registration and profile management capabilities, and an efficient appointment scheduling system. The system also incorporates detailed medical records documentation, integrated billing and payment processing, prescription management, and laboratory results tracking. These functionalities form the backbone of daily healthcare operations and ensure smooth workflow management.

Non-functional requirements focus on system quality attributes essential for healthcare operations. The system prioritizes optimal performance and response time to handle critical medical situations, maintains high system availability for continuous healthcare service delivery, and ensures user interface accessibility for diverse staff members. Additionally, robust data backup and recovery mechanisms are implemented to protect critical medical information, while system scalability allows for future growth and expansion.

The data requirements specification outlines the critical information needed to support healthcare operations. This includes comprehensive patient demographic information, detailed medical history records, appointment scheduling data, billing and insurance information, laboratory results, and prescription records. The database schema is designed to maintain relationships between these different data elements while ensuring data integrity and easy accessibility.

Security requirements received particular attention due to the sensitive nature of healthcare data. The implementation includes role-based access control to manage user permissions, robust data encryption to protect sensitive information, and comprehensive audit trailing to track all system activities. Secure data transmission protocols are implemented to protect information during transfer, and all features are designed to comply with healthcare regulations and privacy standards.

These comprehensive requirements analysis formed the foundation for developing a robust and efficient healthcare management system. The requirements were carefully mapped to database structures and functionalities, ensuring that the final implementation would meet the complex needs of modern healthcare facilities while maintaining industry standards and regulatory compliance.

Business Problems Addressed

Patient Information Management

- Centralized storage and access to patient demographics, medical history, and contact information
- Integration with emergency contacts and insurance details for comprehensive patient profiles

Appointment Scheduling Challenges

- Automated system to prevent scheduling conflicts and manage doctor availability
- Real-time status tracking and notification system for appointment changes

Medical Records Accessibility

- Unified storage of diagnoses, treatments, and lab results
- Secure access to patient history for authorized healthcare providers

Billing and Payment Processing

- Integrated billing system linking appointments, services, and insurance information
- Automated payment status tracking and financial record maintenance

Healthcare Service Coordination

- Streamlined communication between departments (doctors, labs, pharmacy)
- Efficient resource allocation through room and staff management

Regulatory Compliance and Security

- Role-based access control for sensitive medical information
- Comprehensive audit trail of all system activities and changes

These solutions address critical operational inefficiencies while maintaining healthcare service quality and data security.

Business Rules

1. **User Authentication and Access Control**
- Users must have unique account credentials with specific roles (Patient/Doctor)

2. **Patient Registration**
- Each patient must have a unique identifier and complete demographic information
- Emergency contact information is mandatory for all registered patients

3. **Appointment Management**
- Appointments must be linked to both a valid patient and doctor
- Status can only be 'Scheduled', 'Completed', 'Cancelled', or 'No Show'

4. **Doctor Scheduling**
- Each doctor must be associated with a specific department and specialization
- Doctors must maintain defined availability schedules for appointments

5. **Room Management**
- Room status must be tracked as 'Available', 'Occupied', or 'Maintenance'

6. **Billing Process**
- Each bill must be associated with a specific appointment and patient
- Payment status must follow defined states: 'Paid', 'Unpaid', 'Refunded', or 'No Show'

7. **Medical Records**
- Medical records can only be created for registered patients
- Each record must include diagnosis, treatment, and creation timestamp

8. **Laboratory Management**
- Lab results must be linked to specific tests and patients and have defined costs and descriptions

9. **Prescription Handling**
- Prescriptions require both valid patient and doctor references

10. **Insurance Management**
- Each patient's insurance information must include provider and policy details
- Coverage details must be documented for claim processing

These business rules ensure data integrity, operational efficiency, and compliance with healthcare management standards.

Relationships/Associations

1. **Patients to Appointments:** One patient can have multiple appointments (1:M)
2. **Doctors to Appointments:** One doctor can have multiple appointments (1:M)
3. **Appointments to Billing:** One appointment can have one billing record (1:1)
4. **Patients to Medical Records:** One patient can have multiple medical records (1:M)
5. **Patients to Insurance:** One patient can have multiple insurance records (1:M)
6. **Patients to Emergency Contacts:** One patient can have multiple emergency contacts (1:M)
7. **Patients to Lab Results**: One patient can have multiple lab test results (1:M)
8. **Lab Tests to Lab Results:** One lab test can have multiple results for different patients (1:M)
9. **Doctors to Prescriptions:** One doctor can issue multiple prescriptions (1:M)
10. **Prescriptions to Medications:** One prescription can have multiple medications (1:M)
11. **Rooms to Services**: One room can offer multiple services (1:M)
12. **Staff to Services:** One staff member can provide multiple services (1:M)
13. **User Accounts to Notifications:** One user account can receive multiple notifications (1:M)
14. **Doctors to Schedules:** One doctor can have multiple schedule entries (1:M)
15. **Patients to Feedback:** One patient can provide multiple feedback entries (1:M)

Table Description

Database Table Descriptions for Healthcare Patient Clinic Management System:

1. **User_Accounts**
- Serves as the central authentication system managing user credentials and access control
- Maintains distinct user roles (Patient/Doctor) with unique account identifiers

2. **Patients**
- Stores comprehensive patient demographic information including contact details and medical identifiers
- Maintains relationships with multiple healthcare services through unique patient identification

3. **Doctors**
- Manages healthcare provider information including specializations and department affiliations
- Controls doctor availability and scheduling through departmental organization

4. **Appointments**
- Coordinates patient-doctor interactions with specific time slots and scheduling
- Tracks appointment status through various states (Scheduled, Completed, Cancelled, No Show)

5. **Billing**
- Handles financial transactions related to medical services and appointments
- Maintains payment status tracking and appointment-specific billing records

6. **Visits**
- Records patient visit information including duration and purpose
- Tracks historical patient attendance and visit patterns

7. **Insurance**
- Manages patient insurance policies and coverage information
- Maintains provider relationships and policy details for billing coordination

8. **Lab_Tests**
- Catalogs available medical tests with associated costs and descriptions
- Standardizes test information for consistent laboratory operations

9. **Lab_Results**
- Records patient-specific test outcomes and medical findings
- Links test results to specific patients and test types for medical tracking

10. Feedback

- Captures patient experiences and satisfaction metrics
- Maintains historical feedback data for service improvement

11. Rooms

- Manages hospital room allocation and availability
- Tracks room types, capacity, and occupancy status

12. Emergency_Contacts

- Stores critical contact information for patient emergencies
- Maintains relationship details between patients and their emergency contacts

13. Staff

- Manages non-physician healthcare staff information and roles
- Tracks staff assignments and contact information

14. Services

- Catalogs available medical services with associated costs
- Links services to specific rooms and staff members

15. Medical_Records

- Maintains comprehensive patient medical history and treatments
- Records diagnoses, treatments, and clinical notes

5. Prescriptions

- Manages medication orders and dosage information
- Links prescriptions to both patients and prescribing doctors

17. Medications

- Tracks medication inventory and prescription fulfillment
- Maintains pharmaceutical details including dosage forms and stock levels

18. Notifications

- Manages system-generated alerts and communications
- Tracks message delivery and notification types

19. Schedules

- Manages doctor availability and working hours
- Coordinates medical staff scheduling and availability status

This comprehensive table structure ensures efficient healthcare service delivery while maintaining data integrity and relationship management across all aspects of clinic operations.

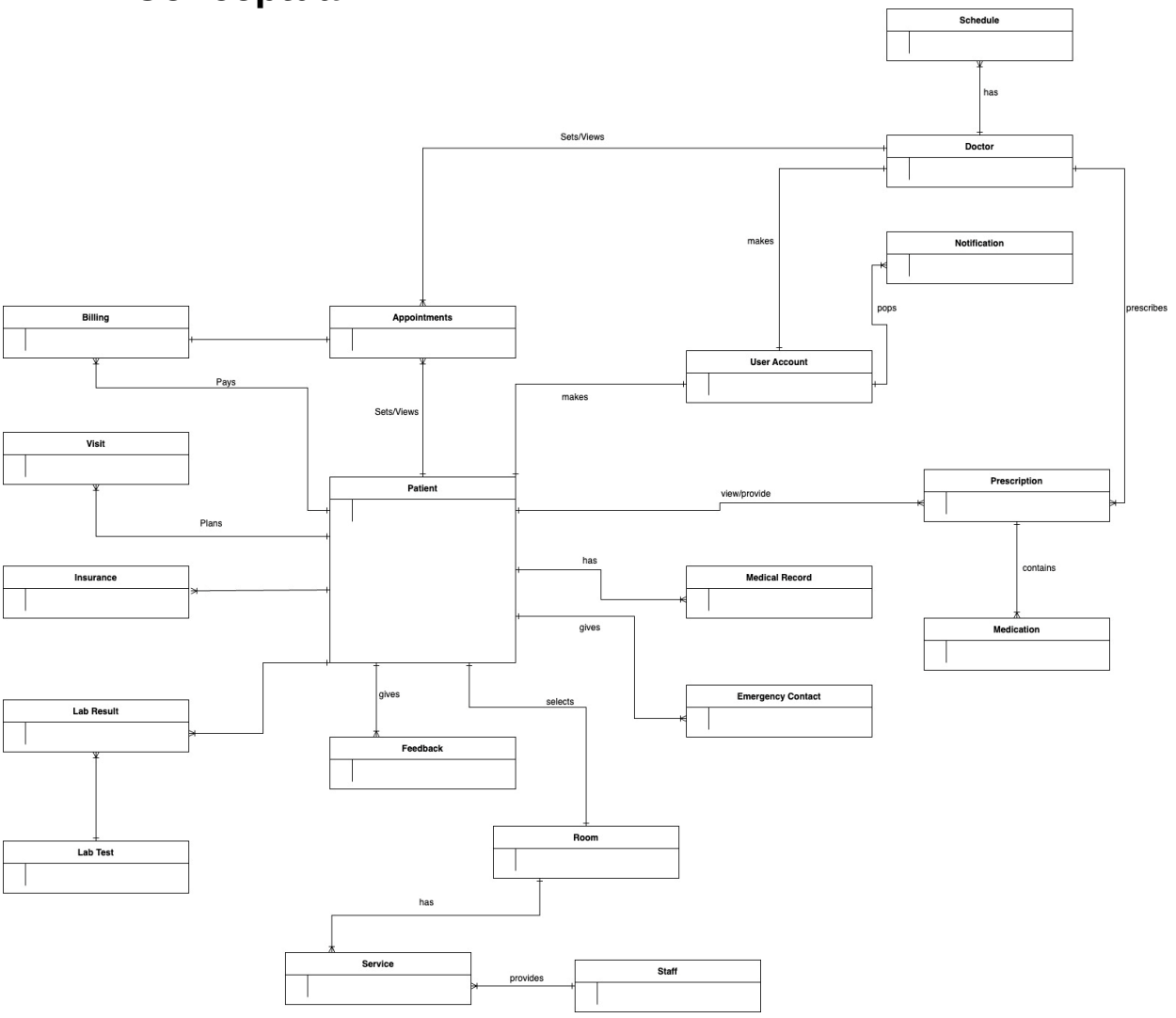
Tools and Techniques

Database Lifecycle (DBLC)

We have incorporated the different steps in the DBLC lifecycle like database initial study, requirement collection and analysis, normalization, implementation, testing, and furthermore into our Clinic Management System. Below are the steps followed:

- Database Initial Study: Extensive research, defined problems, proposed solutions, and the scope of the project
- Database Requirements Analysis: Gathering and analyzing requirements for the database, including data types, volumes, performance requirements, and user needs
- Entity-Relationship Diagrams (ERD): Creating ERDs to visualize the relationships between different data entities and understand the database structure
- Normalization: Applying normalization techniques to ensure data integrity, referential integrity, and reduce redundancy in the database schema
- Database Modeling Tools: Utilizing tools like DrawIO, SQL Developer, Oracle SQL and React for designing and visualizing the database schema
- Database Design Review: Conducting peer reviews and walkthroughs to validate the database design against requirements and best practices
- SQL Scripting: Writing SQL scripts to create database tables, define relationships, and implement constraints based on the finalized database design
- Data Quality Assessment: Evaluating data quality and consistency within the database through data profiling and validation checks
- Backup and Recovery Procedures: Establishing backup and recovery procedures to ensure data integrity and minimize downtime in case of failures
- Database Administration (DBA) Tasks: Performing routine maintenance tasks such as index optimization, database tuning, and space management

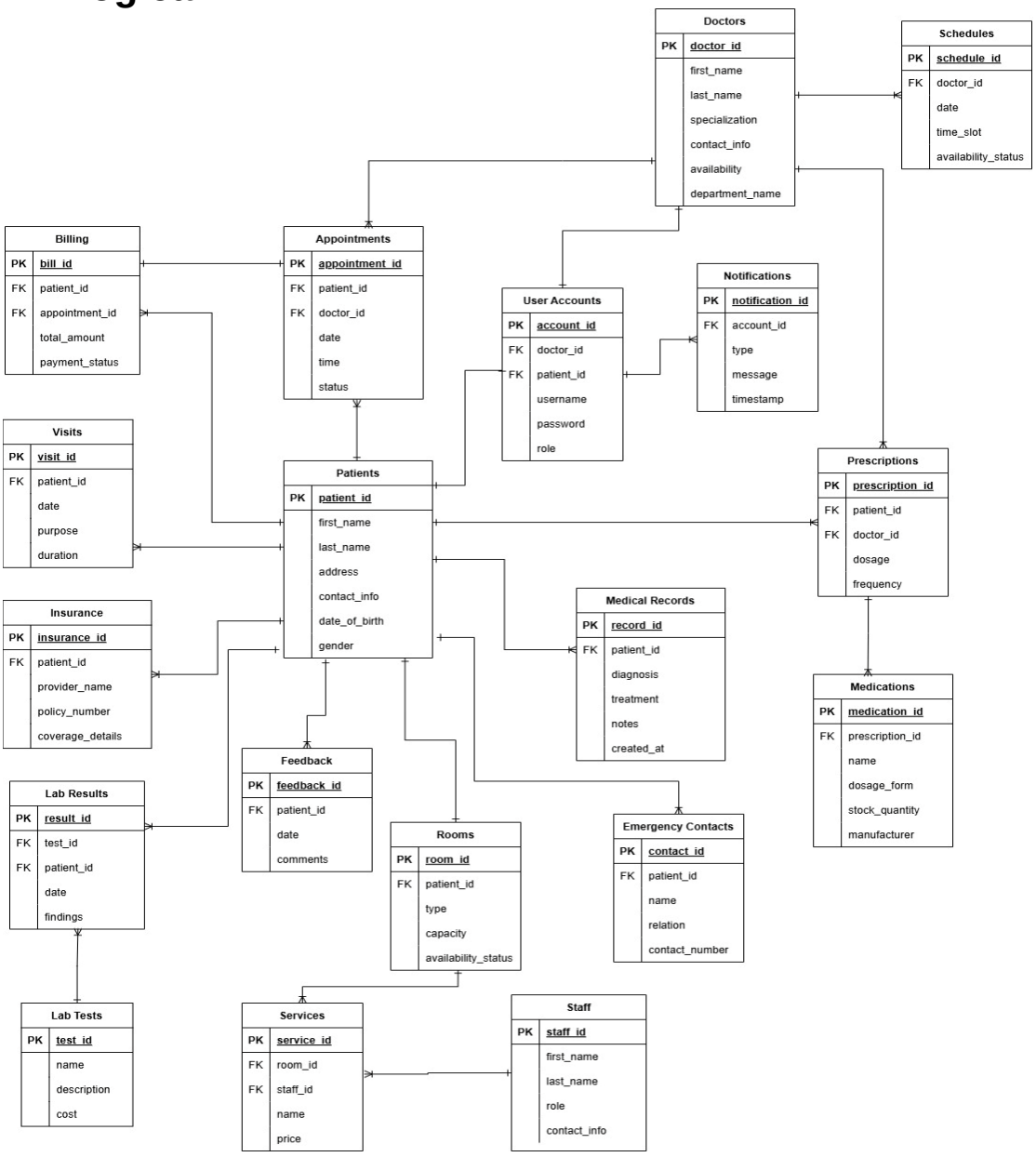
ERD – Conceptual



The Conceptual ERD for a healthcare management system centers on the Patient entity, connecting key components for efficient care delivery. Patients schedule consultations with Doctors via Appointments, managed by their Schedules. User Accounts enable booking and Notifications. Prescriptions link to Medications, while Billing integrates with Appointments and Insurance for seamless payment handling.

Visits document treatments, linking Feedback, Rooms, and Staff. Medical Records store health data, with Lab Tests and Results providing diagnostics. Emergency Contacts support urgent needs, and Services ensure staff-driven care within the system's infrastructure.

ERD – Logical

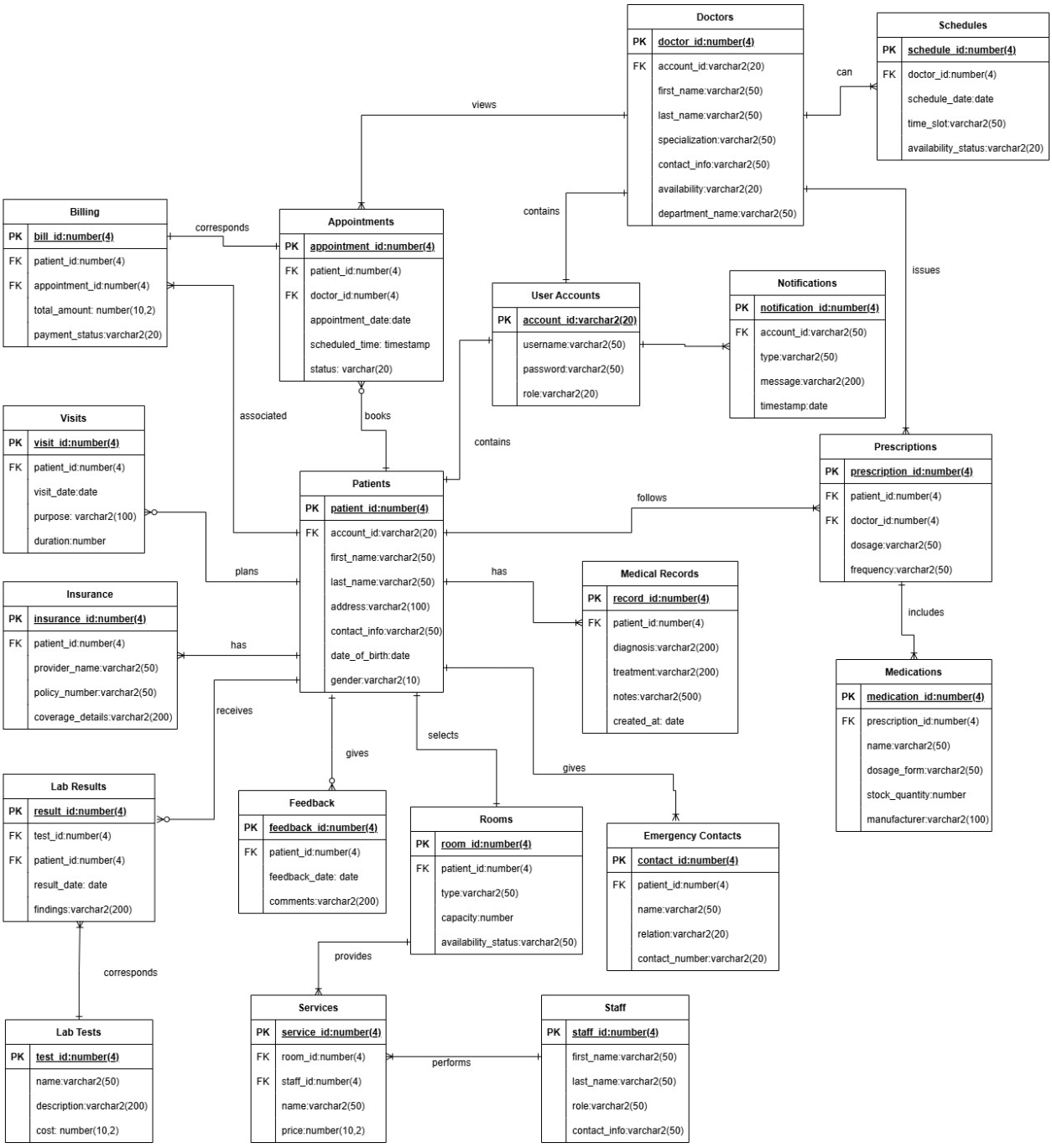


The Logical ERD for a healthcare management system outlines entities, attributes, and relationships for efficient patient care. Patients store personal details and connect to Appointments, Visits, Billing, Insurance, and Medical Records for comprehensive management.

Appointments link patients to Doctors, with Schedules managing availability. Billing handles payments, integrating with Insurance for claims. Visits document interactions, while Medical Records, Prescriptions, and Medications track care details. Lab Tests and Results support diagnostics.

Rooms and Staff ensure operational efficiency. User Accounts provide secure access, and Notifications streamline communication.

ERD - Final


Doctors
PK doctor_id:number(4)
FK account_id:varchar2(20)
first_name:varchar2(50)
last_name:varchar2(50)
specialization:varchar2(50)
contact_info:varchar2(50)
availability:varchar2(20)
department_name:varchar2(50)
Schedules
PK schedule_id:number(4)
FK doctor_id:number(4)
schedule_date:date
time_slot:varchar2(50)
availability_status:varchar2(20)
can
views
contains
issues
Billing
PK bill_id:number(4)
FK patient_id:number(4)
FK appointment_id:number(4)
total_amount: number(10,2)
payment_status:varchar2(20)
corresponds
Appointments
PK appointment_id:number(4)
FK patient_id:number(4)
FK doctor_id:number(4)
appointment_date:date
scheduled_time: timestamp
status: varchar(20)
User Accounts
PK account_id:varchar2(20)
username:varchar2(50)
password:varchar2(50)
role:varchar2(20)
Notifications
PK notification_id:number(4)
FK account_id:varchar2(50)
type:varchar2(50)
message:varchar2(200)
timestamp:date
Visits
PK visit_id:number(4)
FK patient_id:number(4)
visit_date:date
purpose: varchar2(100)
duration:number
associated
books
contains
Patients
PK patient_id:number(4)
FK account_id:varchar2(20)
first_name:varchar2(50)
last_name:varchar2(50)
address:varchar2(100)
contact_info:varchar2(50)
date_of_birth:date
gender:varchar2(10)
follows
Prescriptions
PK prescription_id:number(4)
FK patient_id:number(4)
FK doctor_id:number(4)
dosage:varchar2(50)
frequency:varchar2(50)
includes
plans
has
Insurance
PK insurance_id:number(4)
FK patient_id:number(4)
provider_name:varchar2(50)
policy_number:varchar2(50)
coverage_details:varchar2(200)
has
Medical Records
PK record_id:number(4)
FK patient_id:number(4)
diagnosis:varchar2(200)
treatment:varchar2(200)
notes:varchar2(500)
created_at: date
Medications
PK medication_id:number(4)
FK prescription_id:number(4)
name:varchar2(50)
dosage_form:varchar2(50)
stock_quantity:number
manufacturer:varchar2(100)
receives
gives
selects
gives
Lab Results
PK result_id:number(4)
FK test_id:number(4)
FK patient_id:number(4)
result_date: date
findings:varchar2(200)
Feedback
PK feedback_id:number(4)
FK patient_id:number(4)
feedback_date: date
comments:varchar2(200)
Rooms
PK room_id:number(4)
FK patient_id:number(4)
type:varchar2(50)
capacity:number
availability_status:varchar2(50)
Emergency Contacts
PK contact_id:number(4)
FK patient_id:number(4)
name:varchar2(50)
relation:varchar2(20)
contact_number:varchar2(20)
corresponds
provides
Lab Tests
PK test_id:number(4)
name:varchar2(50)
description:varchar2(200)
cost: number(10,2)
Services
PK service_id:number(4)
FK room_id:number(4)
FK staff_id:number(4)
name:varchar2(50)
price:number(10,2)
performs
Staff
PK staff_id:number(4)
first_name:varchar2(50)
last_name:varchar2(50)
role:varchar2(50)
contact_info:varchar2(50)

Physical Data Store Organization

Table Name	Column Name	Data Type	Constraints
User_Accounts	account_id	VARCHAR2(20)	PRIMARY KEY
	username	VARCHAR2(50)	NOT NULL, UNIQUE
	password	VARCHAR2(50)	NOT NULL
	role	VARCHAR2(20)	NOT NULL
Patients	patient_id	NUMBER(4)	PRIMARY KEY
	account_id	VARCHAR2(20)	UNIQUE, FOREIGN KEY
	first_name	VARCHAR2(50)	NOT NULL
	last_name	VARCHAR2(50)	NOT NULL
	address	VARCHAR2(100)	NOT NULL
	contact_info	VARCHAR2(50)	NOT NULL
	date_of_birth	DATE	NOT NULL
	gender	VARCHAR2(10)	NOT NULL
Doctors	doctor_id	NUMBER(4)	PRIMARY KEY
	account_id	VARCHAR2(20)	UNIQUE, FOREIGN KEY
	first_name	VARCHAR2(50)	NOT NULL
	last_name	VARCHAR2(50)	NOT NULL
	specialization	VARCHAR2(50)	NOT NULL
	contact_info	VARCHAR2(50)	NOT NULL
	availability	VARCHAR2(20)	NOT NULL
	department_name	VARCHAR2(50)	NOT NULL
Appointments	appointment_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	doctor_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	appointment_date	DATE	NOT NULL
	scheduled_time	TIMESTAMP	NOT NULL
	status	VARCHAR2(20)	NOT NULL
Billing	bill_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	appointment_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	total_amount	NUMBER(10,2)	NOT NULL
	payment_status	VARCHAR2(20)	NOT NULL
Visits	visit_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	visit_date	DATE	NOT NULL

	purpose	VARCHAR2(100)	NOT NULL
	duration	NUMBER	NOT NULL
Insurance	insurance_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	provider_name	VARCHAR2(50)	NOT NULL
	policy_number	VARCHAR2(50)	NOT NULL
	coverage_details	VARCHAR2(200)	NOT NULL
Lab_Tests	test_id	NUMBER(4)	PRIMARY KEY
	name	VARCHAR2(50)	NOT NULL
	description	VARCHAR2(200)	NOT NULL
	cost	NUMBER(10,2)	NOT NULL
Lab_Results	result_id	NUMBER(4)	PRIMARY KEY
	test_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	result_date	DATE	NOT NULL
	findings	VARCHAR2(200)	NOT NULL
Feedback	feedback_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	feedback_date	DATE	NOT NULL
	comments	VARCHAR2(200)	NOT NULL
Rooms	room_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	FOREIGN KEY
	type	VARCHAR2(50)	NOT NULL
	capacity	NUMBER	NOT NULL
	availability_status	VARCHAR2(20)	NOT NULL
Emergency_Contacts	contact_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	name	VARCHAR2(50)	NOT NULL
	relation	VARCHAR2(50)	NOT NULL
	contact_number	VARCHAR2(20)	NOT NULL
Staff	staff_id	NUMBER(4)	PRIMARY KEY
	first_name	VARCHAR2(50)	NOT NULL
	last_name	VARCHAR2(50)	NOT NULL
	role	VARCHAR2(50)	NOT NULL

	contact_info	VARCHAR2(50)	NOT NULL
Services	service_id	NUMBER(4)	PRIMARY KEY
	room_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	staff_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	name	VARCHAR2(50)	NOT NULL
	price	NUMBER(10,2)	NOT NULL
Medical_Records	record_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	diagnosis	VARCHAR2(200)	NOT NULL
	treatment	VARCHAR2(200)	NOT NULL
	notes	VARCHAR2(500)	
	created_at	DATE	NOT NULL
Prescriptions	prescription_id	NUMBER(4)	PRIMARY KEY
	patient_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	doctor_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	dosage	VARCHAR2(50)	NOT NULL
	frequency	VARCHAR2(50)	NOT NULL
Medications	medication_id	NUMBER(4)	PRIMARY KEY
	prescription_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	name	VARCHAR2(50)	NOT NULL
	dosage_form	VARCHAR2(50)	NOT NULL
	stock_quantity	NUMBER	NOT NULL
	manufacturer	VARCHAR2(100)	NOT NULL
Notifications	notification_id	NUMBER(4)	PRIMARY KEY
	account_id	VARCHAR2(20)	NOT NULL, FOREIGN KEY
	type	VARCHAR2(50)	NOT NULL
	message	VARCHAR2(200)	NOT NULL
	timestamp	DATE	NOT NULL
Schedules	schedule_id	NUMBER(4)	PRIMARY KEY
	doctor_id	NUMBER(4)	NOT NULL, FOREIGN KEY
	schedule_date	DATE	NOT NULL
	time_slot	VARCHAR2(50)	NOT NULL

	availability_status	VARCHAR2(20)	NOT NULL

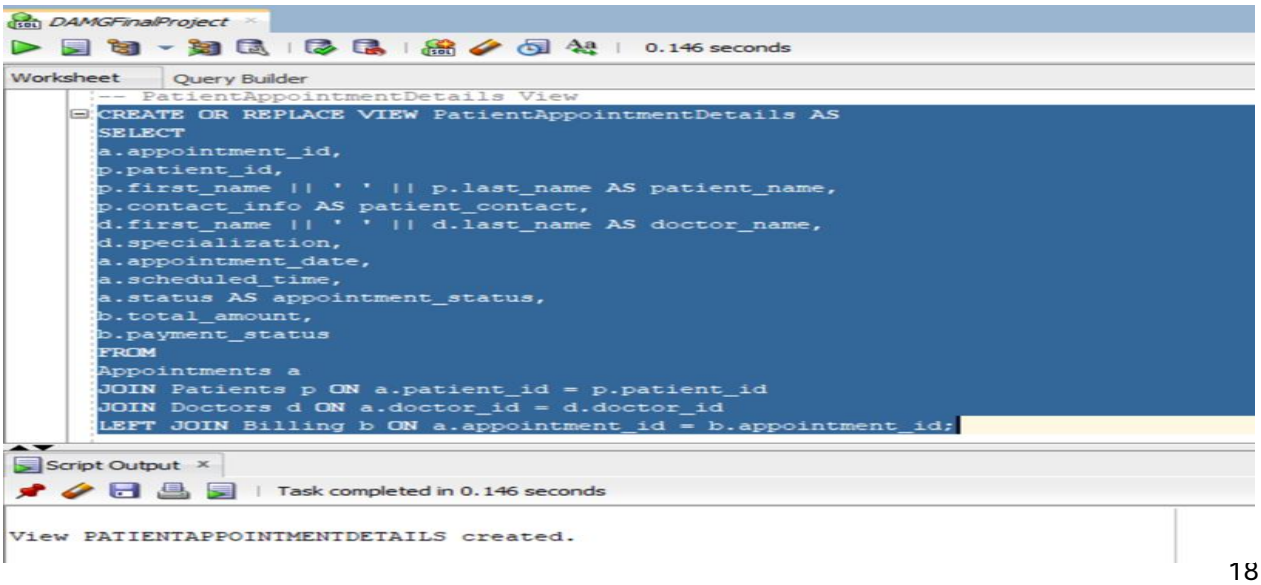
DDL Commands

Create Table and Insert Queries Refer to appendix 1.1

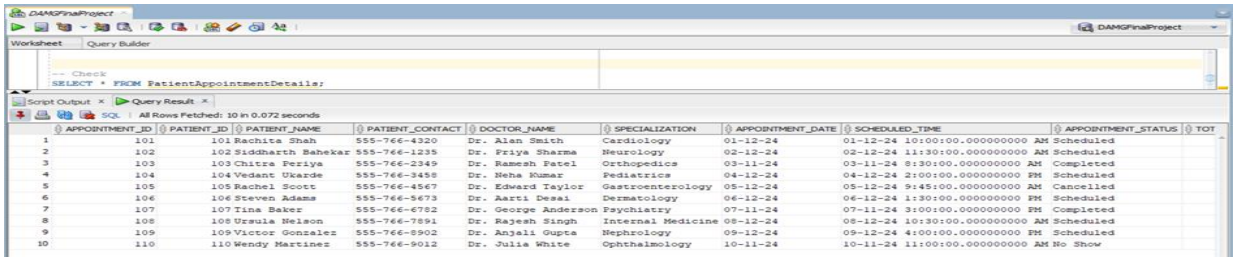
Concept implementation code pieces all with screenshot

PatientAppointmentDetails View

```
CREATE OR REPLACE VIEW PatientAppointmentDetails AS
SELECT
a.appointment_id,
p.patient_id,
p.first_name || ' ' || p.last_name AS patient_name,
p.contact_info AS patient_contact,
d.first_name || ' ' || d.last_name AS doctor_name,
d.specialization,
a.appointment_date,
a.scheduled_time,
a.status AS appointment_status,
b.total_amount,
b.payment_status
FROM
Appointments a
JOIN Patients p ON a.patient_id = p.patient_id
JOIN Doctors d ON a.doctor_id = d.doctor_id
LEFT JOIN Billing b ON a.appointment_id = b.appointment_id;
```



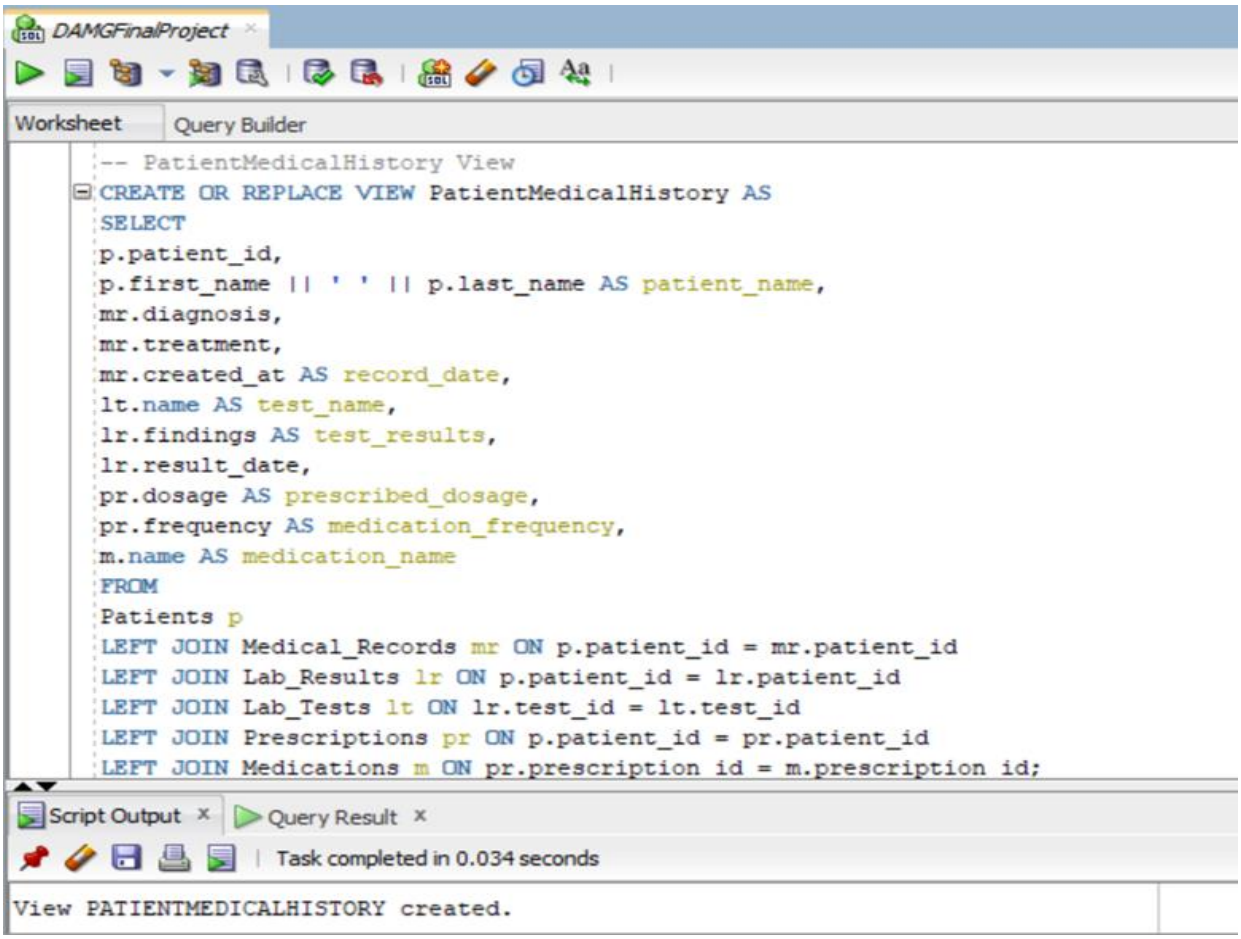
-- Check
SELECT * FROM PatientAppointmentDetails;



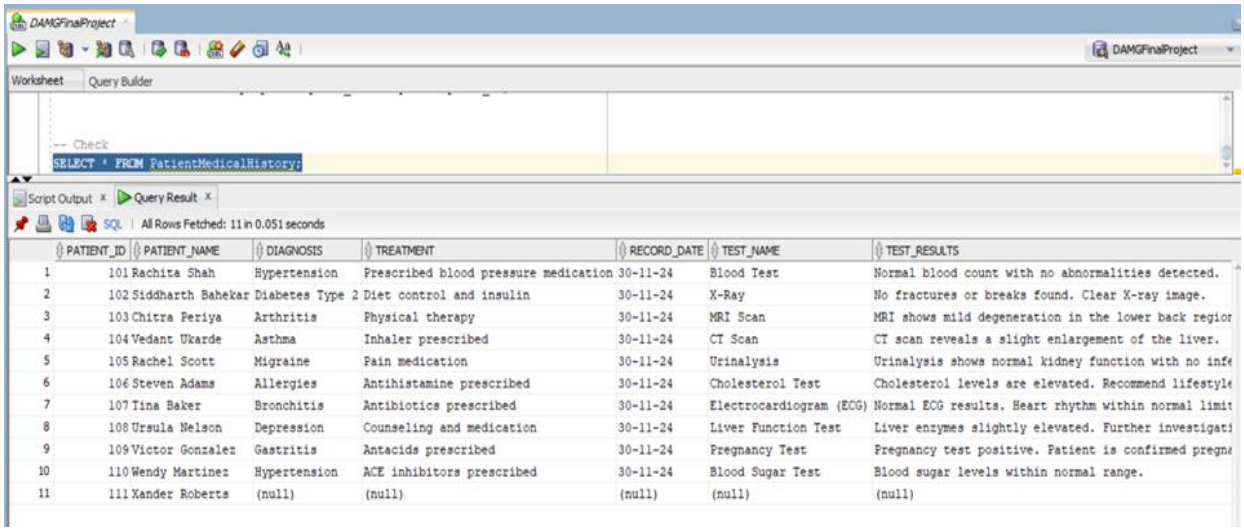
APPOINTMENT_ID	PATIENT_ID	PATIENT_NAME	PATIENT_CONTACT	DOCTOR_NAME	SPECIALIZATION	APPOINTMENT_DATE	SCHEDULED_TIME	APPOINTMENT_STATUS	TOT
1	101	101 Rachita Shah	555-766-4320	Dr. Alan Smith	Cardiology	01-12-24	01-12-24 10:00:00.000000000 AM	Scheduled	
2	102	102 Siddharth Bahekar	555-766-1235	Dr. Priya Sharma	Neurology	02-12-24	02-12-24 11:30:00.000000000 AM	Scheduled	
3	103	103 Chitra Periya	555-766-2349	Dr. Ramesh Patel	Orthopedics	03-11-24	03-11-24 8:30:00.000000000 AM	Completed	
4	104	104 Vedant Ukarde	555-766-3458	Dr. Neha Kumar	Pediatrics	04-12-24	04-12-24 2:00:00.000000000 PM	Scheduled	
5	105	105 Rachel Scott	555-766-4567	Dr. Edward Taylor	Gastroenterology	05-12-24	05-12-24 9:45:00.000000000 AM	Cancelled	
6	106	106 Steven Adams	555-766-5673	Dr. Aarti Desai	Dermatology	06-12-24	06-12-24 1:30:00.000000000 PM	Scheduled	
7	107	107 Tina Baker	555-766-6782	Dr. George Anderson	Psychiatry	07-11-24	07-11-24 3:00:00.000000000 PM	Completed	
8	108	108 Ursula Nelson	555-766-7891	Dr. Rajesh Singh	Internal Medicine	08-12-24	08-12-24 10:30:00.000000000 AM	Scheduled	
9	109	109 Victor Gonzalez	555-766-8902	Dr. Anjali Gupta	Nephrology	09-12-24	09-12-24 4:00:00.000000000 PM	Scheduled	
10	110	110 Wendy Martinez	555-766-9012	Dr. Julia White	Ophthalmology	10-11-24	10-11-24 11:00:00.000000000 AM	No Show	

PatientMedicalHistory View

CREATE OR REPLACE VIEW PatientMedicalHistory AS
SELECT
p.patient_id,
p.first_name || ' ' || p.last_name AS patient_name,
mr.diagnosis,
mr.treatment,
mr.created_at AS record_date,
lt.name AS test_name,
lr.findings AS test_results,
lr.result_date,
pr.dosage AS prescribed_dosage,
pr.frequency AS medication_frequency,
m.name AS medication_name
FROM
Patients p
LEFT JOIN Medical_Records mr ON p.patient_id = mr.patient_id
LEFT JOIN Lab_Results lr ON p.patient_id = lr.patient_id
LEFT JOIN Lab_Tests lt ON lr.test_id = lt.test_id
LEFT JOIN Prescriptions pr ON p.patient_id = pr.patient_id
LEFT JOIN Medications m ON pr.prescription_id = m.prescription_id;



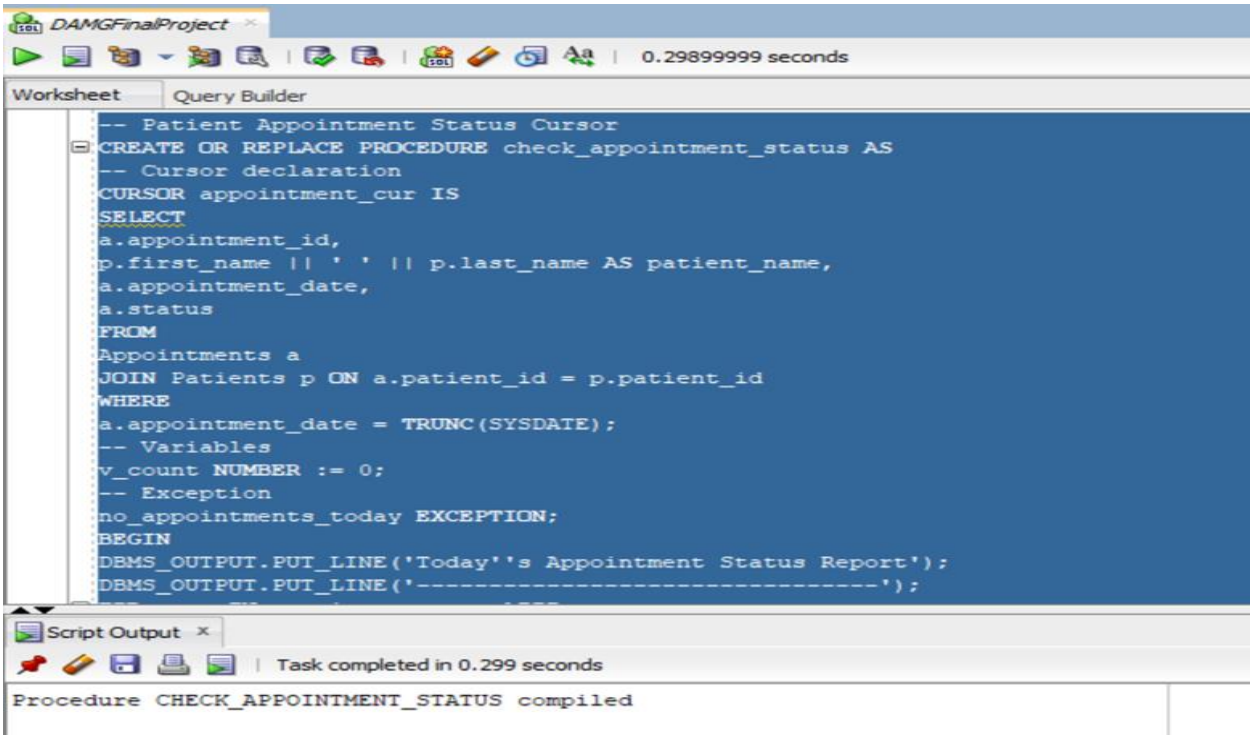
-- Check
SELECT * FROM PatientMedicalHistory;



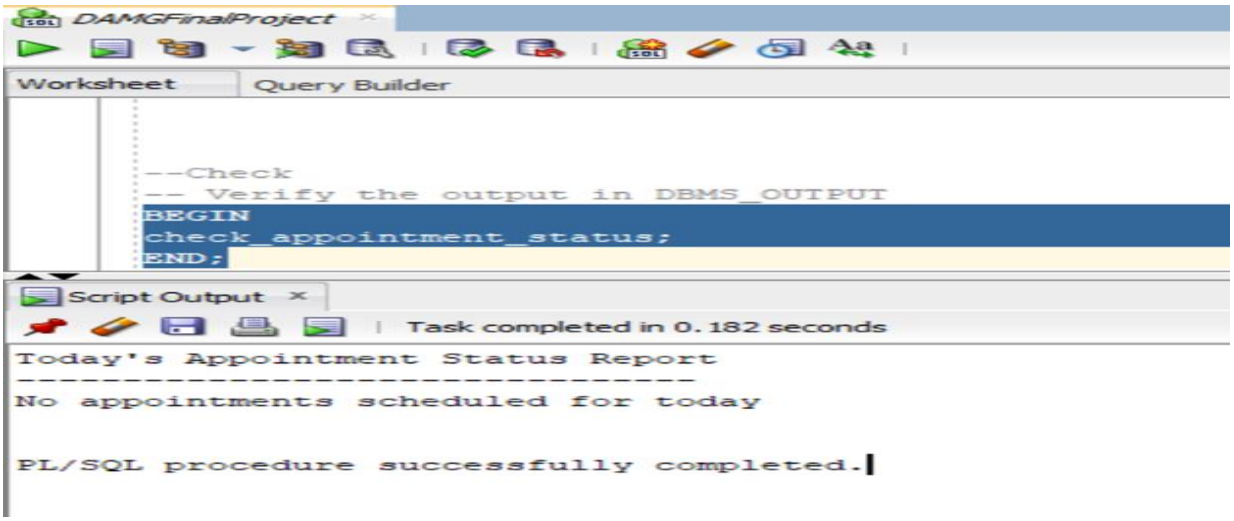
	PATIENT_ID	PATIENT_NAME	DIAGNOSIS	TREATMENT	RECORD_DATE	TEST_NAME	TEST_RESULTS
1	101	Rachita Shah	Hypertension	Prescribed blood pressure medication	30-11-24	Blood Test	Normal blood count with no abnormalities detected.
2	102	Siddharth Bahekar	Diabetes Type 2	Diet control and insulin	30-11-24	X-Ray	No fractures or breaks found. Clear X-ray image.
3	103	Chitra Periya	Arthritis	Physical therapy	30-11-24	MRI Scan	MRI shows mild degeneration in the lower back regior
4	104	Vedant Ukarde	Asthma	Inhaler prescribed	30-11-24	CT Scan	CT scan reveals a slight enlargement of the liver.
5	105	Rachel Scott	Migraine	Pain medication	30-11-24	Urinalysis	Urinalysis shows normal kidney function with no infe
6	106	Steven Adams	Allergies	Antihistamine prescribed	30-11-24	Cholesterol Test	Cholesterol levels are elevated. Recommend lifestyle
7	107	Tina Baker	Bronchitis	Antibiotics prescribed	30-11-24	Electrocardiogram (ECG)	Normal ECG results. Heart rhythm within normal limit
8	108	Ursula Nelson	Depression	Counseling and medication	30-11-24	Liver Function Test	Liver enzymes slightly elevated. Further investigati
9	109	Victor Gonzalez	Gastritis	Antacids prescribed	30-11-24	Pregnancy Test	Pregnancy test positive. Patient is confirmed pregna
10	110	Wendy Martinez	Hypertension	ACE inhibitors prescribed	30-11-24	Blood Sugar Test	Blood sugar levels within normal range.
11	111	Xander Roberts	(null)	(null)	(null)	(null)	(null)

Patient Appointment Status Cursor

```
CREATE OR REPLACE PROCEDURE check_appointment_status AS
-- Cursor declaration
CURSOR appointment_cur IS
SELECT
a.appointment_id,
p.first_name || ' ' || p.last_name AS patient_name,
a.appointment_date,
a.status
FROM
Appointments a
JOIN Patients p ON a.patient_id = p.patient_id
WHERE
a.appointment_date = TRUNC(SYSDATE);
-- Variables
v_count NUMBER := 0;
-- Exception
no_appointments_today EXCEPTION;
BEGIN
DBMS_OUTPUT.PUT_LINE('Today''s Appointment Status Report');
DBMS_OUTPUT.PUT_LINE('--------------------------------');
FOR appt IN appointment_cur LOOP
DBMS_OUTPUT.PUT_LINE(
'Appointment ID: ' || appt.appointment_id ||
' | Patient: ' || appt.patient_name ||
' | Status: ' || appt.status
);
v_count := v_count + 1;
END LOOP;
IF v_count = 0 THEN
RAISE no_appointments_today;
END IF;
EXCEPTION
WHEN no_appointments_today THEN
DBMS_OUTPUT.PUT_LINE('No appointments scheduled for today');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END check_appointment_status;
```



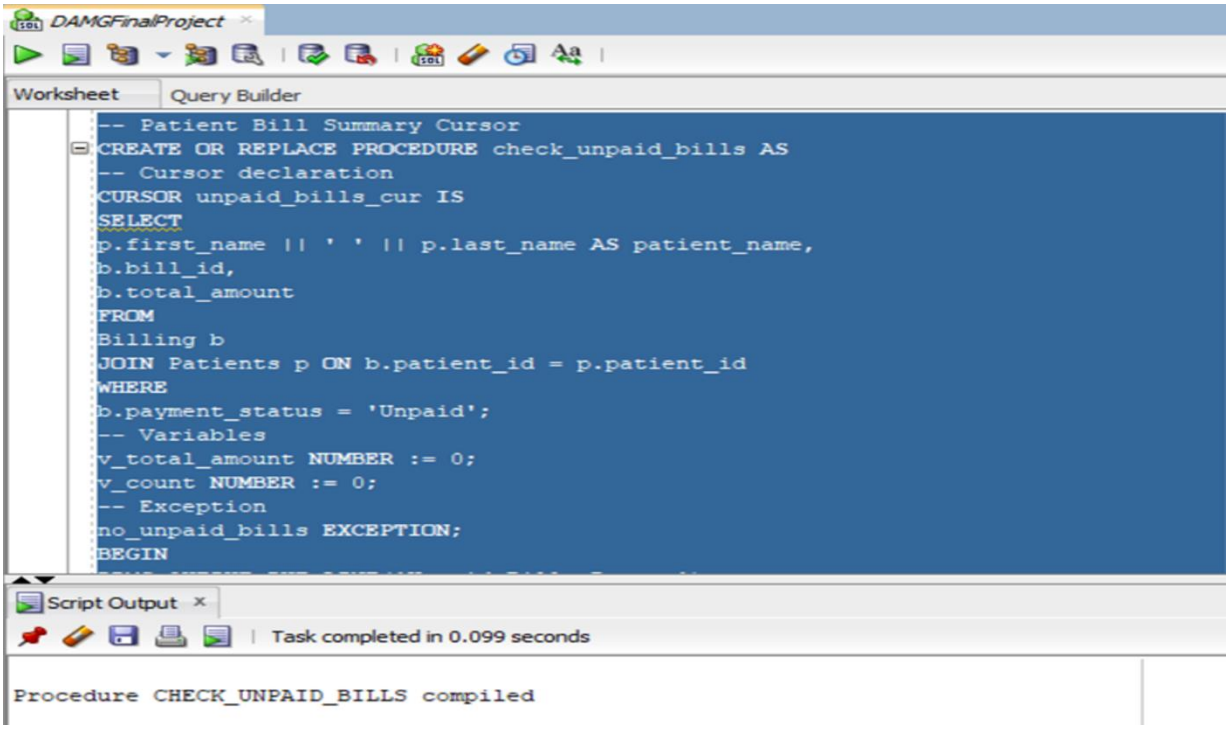
--Check
-- Verify the output in DBMS_OUTPUT
BEGIN
check_appointment_status;
END;



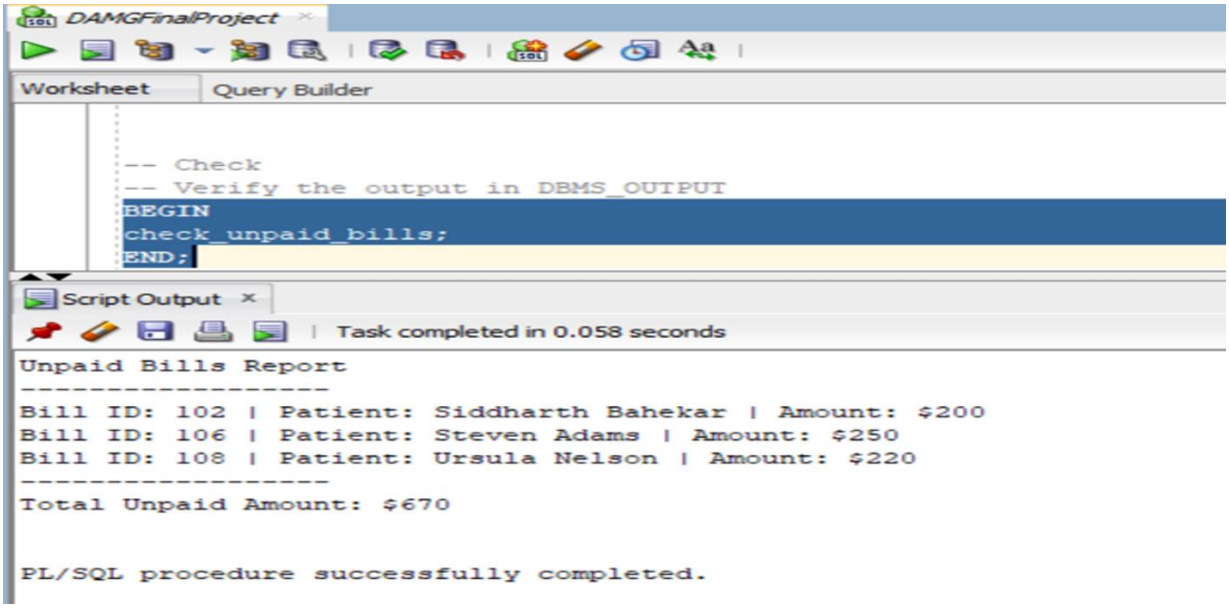
Patient Bill Summary Cursor

```
CREATE OR REPLACE PROCEDURE check_unpaid_bills AS
-- Cursor declaration
CURSOR unpaid_bills_cur IS
SELECT
p.first_name || ' ' || p.last_name AS patient_name,
b.bill_id,
b.total_amount
FROM
Billing b
JOIN Patients p ON b.patient_id = p.patient_id
WHERE
b.payment_status = 'Unpaid';
-- Variables
v_total_amount NUMBER := 0;
v_count NUMBER := 0;
-- Exception
no_unpaid_bills EXCEPTION;
BEGIN
DBMS_OUTPUT.PUT_LINE('Unpaid Bills Report');
DBMS_OUTPUT.PUT_LINE('-------------------');
FOR bill IN unpaid_bills_cur LOOP
DBMS_OUTPUT.PUT_LINE(
'Bill ID: ' || bill.bill_id ||
' | Patient: ' || bill.patient_name ||
' | Amount: $' || bill.total_amount
);
v_total_amount := v_total_amount + bill.total_amount;
v_count := v_count + 1;
END LOOP;
IF v_count = 0 THEN
RAISE no_unpaid_bills;
ELSE
DBMS_OUTPUT.PUT_LINE('-------------------');
DBMS_OUTPUT.PUT_LINE('Total Unpaid Amount: $' || v_total_amount);
END IF;
EXCEPTION
WHEN no_unpaid_bills THEN
DBMS_OUTPUT.PUT_LINE('No unpaid bills found');
```

WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END check_unpaid_bills;

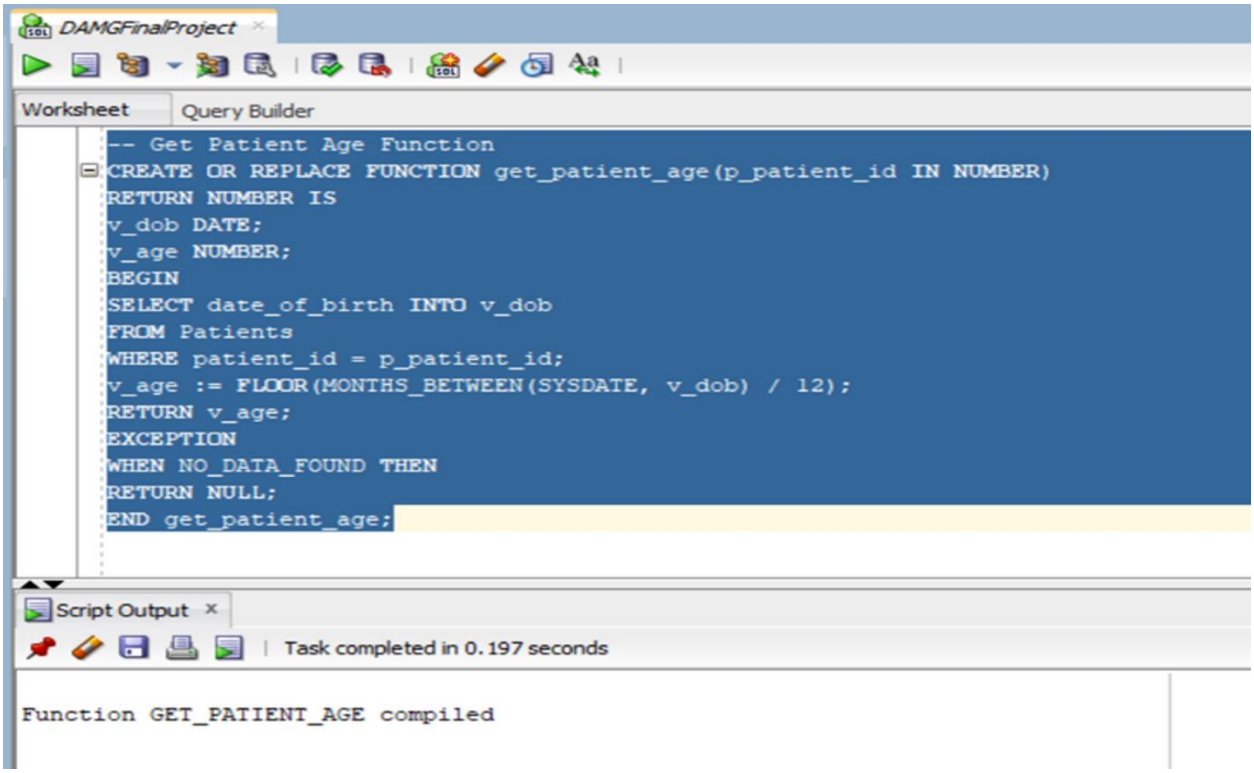


-- Check
-- Verify the output in DBMS_OUTPUT
BEGIN
check_unpaid_bills;
END;

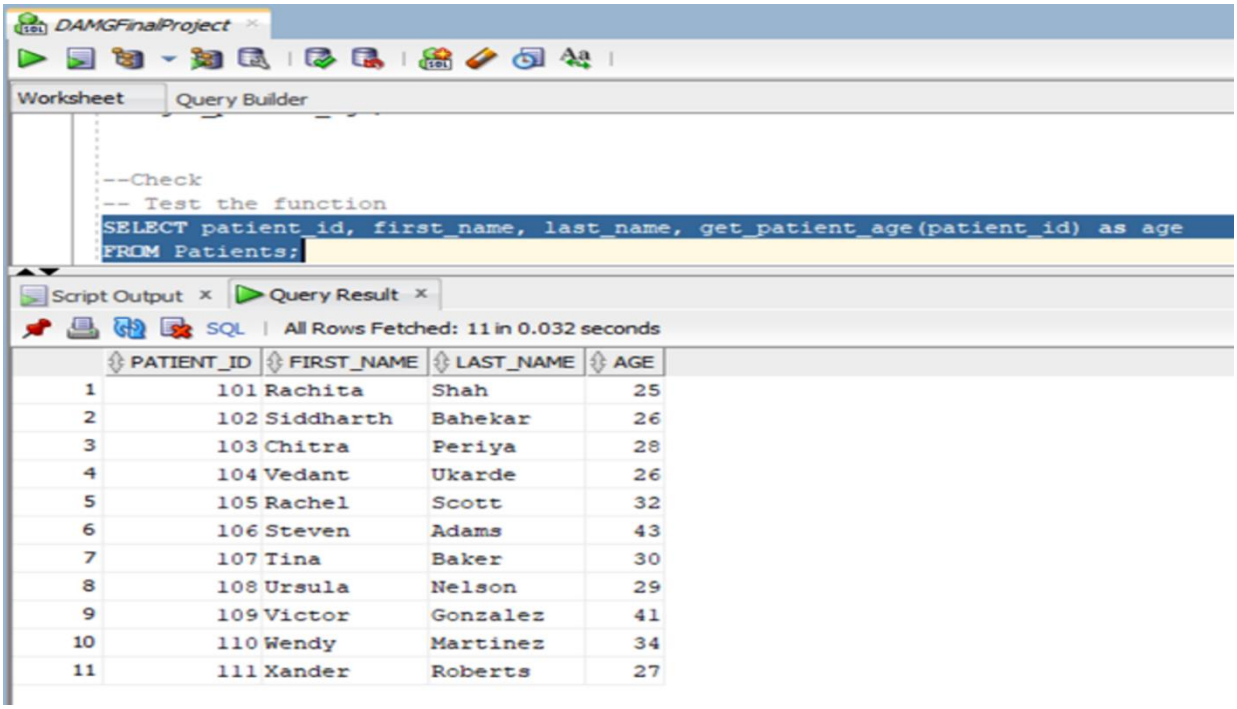


Get Patient Age Function

```
CREATE OR REPLACE FUNCTION get_patient_age(p_patient_id IN NUMBER)
RETURN NUMBER IS
v_dob DATE;
v_age NUMBER;
BEGIN
SELECT date_of_birth INTO v_dob
FROM Patients
WHERE patient_id = p_patient_id;
v_age := FLOOR(MONTHS_BETWEEN(SYSDATE, v_dob) / 12);
RETURN v_age;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN NULL;
END get_patient_age;
```

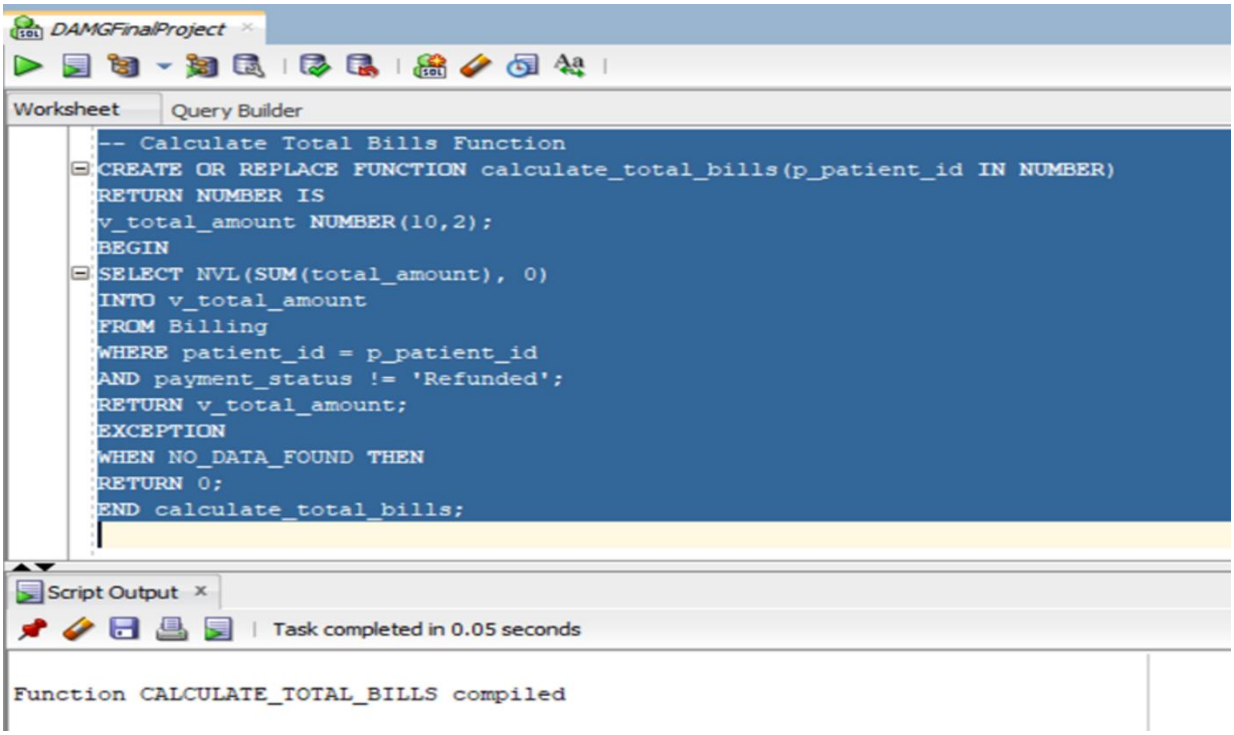



```
--Check
-- Test the function
SELECT patient_id, first_name, last_name, get_patient_age(patient_id) as age
FROM Patients;
```

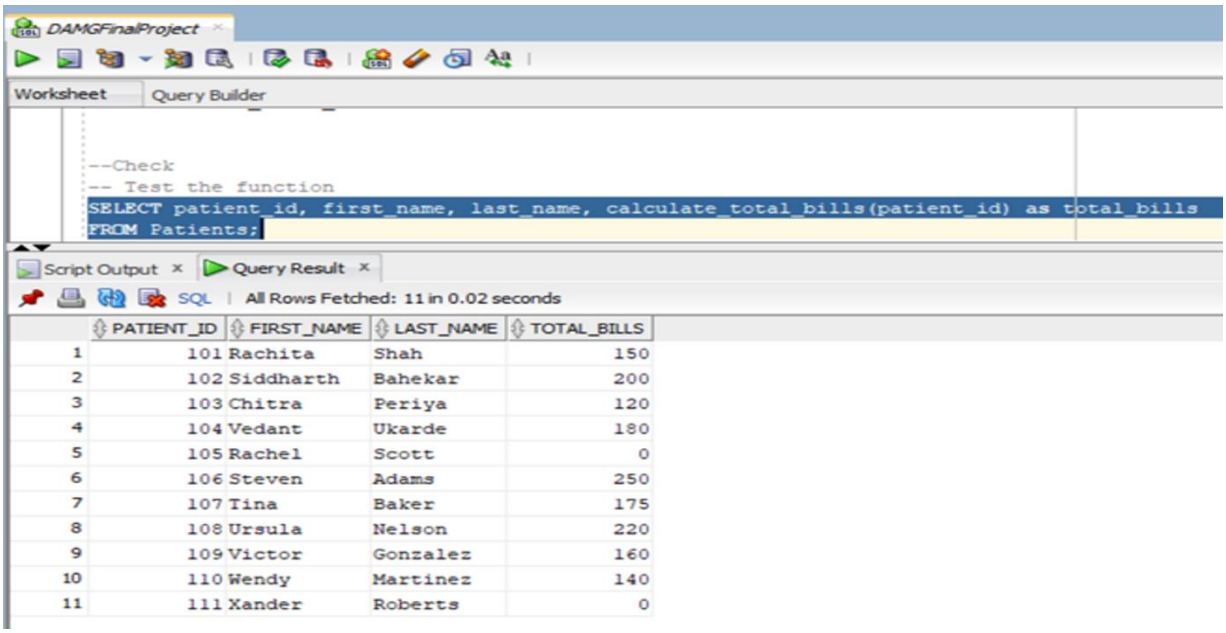


Calculate Total Bills Function

```
CREATE OR REPLACE FUNCTION calculate_total_bills(p_patient_id IN NUMBER)
RETURN NUMBER IS
v_total_amount NUMBER(10,2);
BEGIN
SELECT NVL(SUM(total_amount), 0)
INTO v_total_amount
FROM Billing
WHERE patient_id = p_patient_id
AND payment_status != 'Refunded';
RETURN v_total_amount;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN 0;
END calculate_total_bills;
```



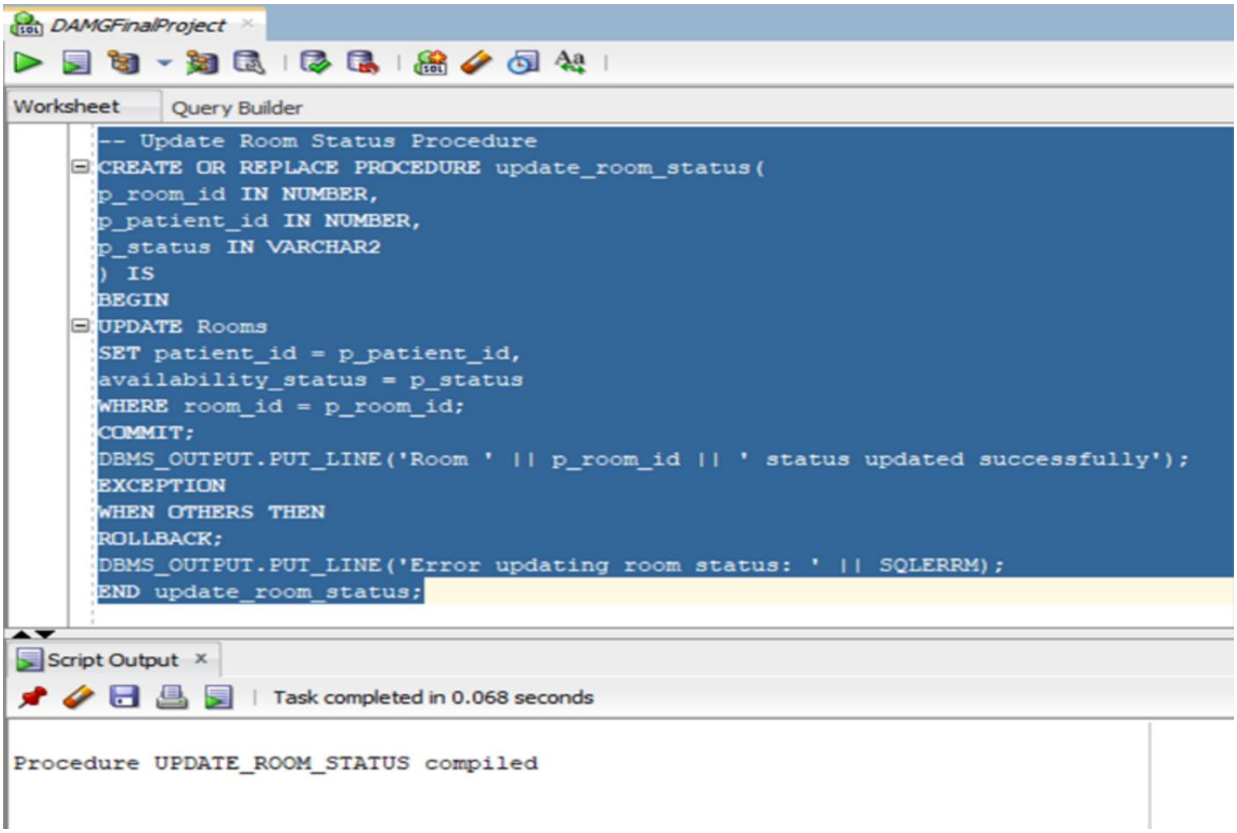
--Check
-- Test the function
SELECT patient_id, first_name, last_name, calculate_total_bills(patient_id) as total_bills
FROM Patients;



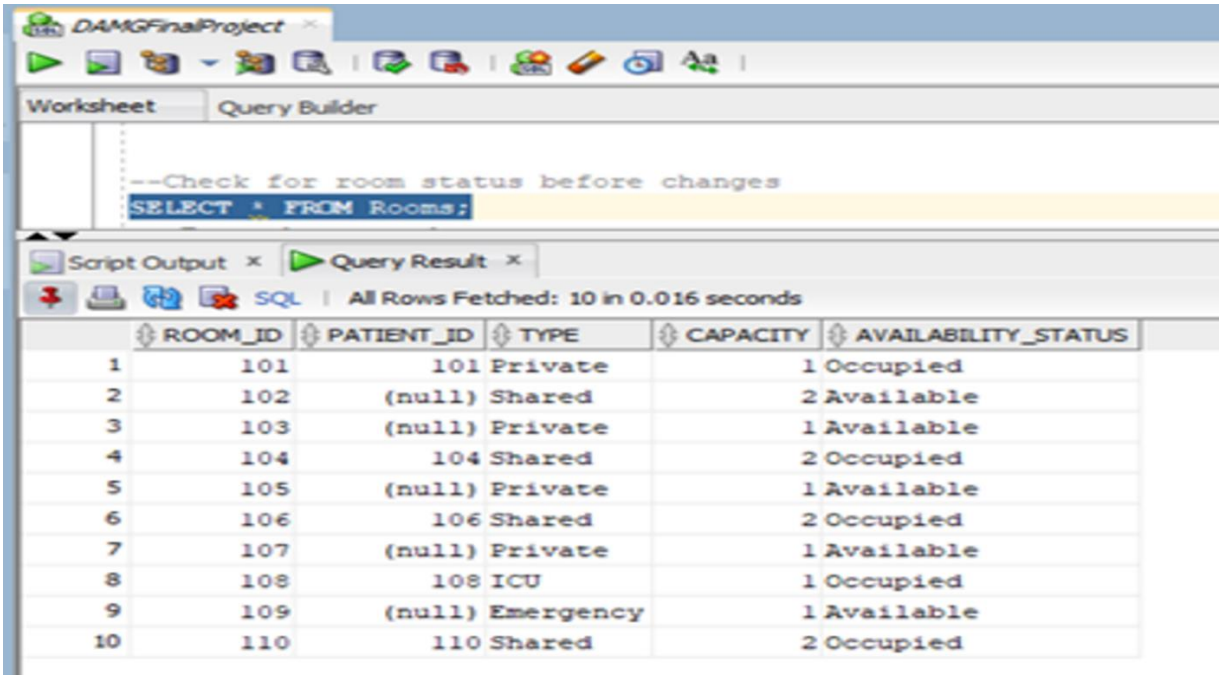
	PATIENT_ID	FIRST_NAME	LAST_NAME	TOTAL_BILLS
1	101	Rachita	Shah	150
2	102	Siddharth	Bahekar	200
3	103	Chitra	Periya	120
4	104	Vedant	Ukarde	180
5	105	Rachel	Scott	0
6	106	Steven	Adams	250
7	107	Tina	Baker	175
8	108	Ursula	Nelson	220
9	109	Victor	Gonzalez	160
10	110	Wendy	Martinez	140
11	111	Xander	Roberts	0

Update Room Status Procedure

```
CREATE OR REPLACE PROCEDURE update_room_status(
p_room_id IN NUMBER,
p_patient_id IN NUMBER,
p_status IN VARCHAR2
) IS
BEGIN
UPDATE Rooms
SET patient_id = p_patient_id,
availability_status = p_status
WHERE room_id = p_room_id;
COMMIT;
DBMS_OUTPUT.PUT_LINE('Room ' || p_room_id || ' status updated successfully');
EXCEPTION
WHEN OTHERS THEN
ROLLBACK;
DBMS_OUTPUT.PUT_LINE('Error updating room status: ' || SQLERRM);
END update_room_status;
```

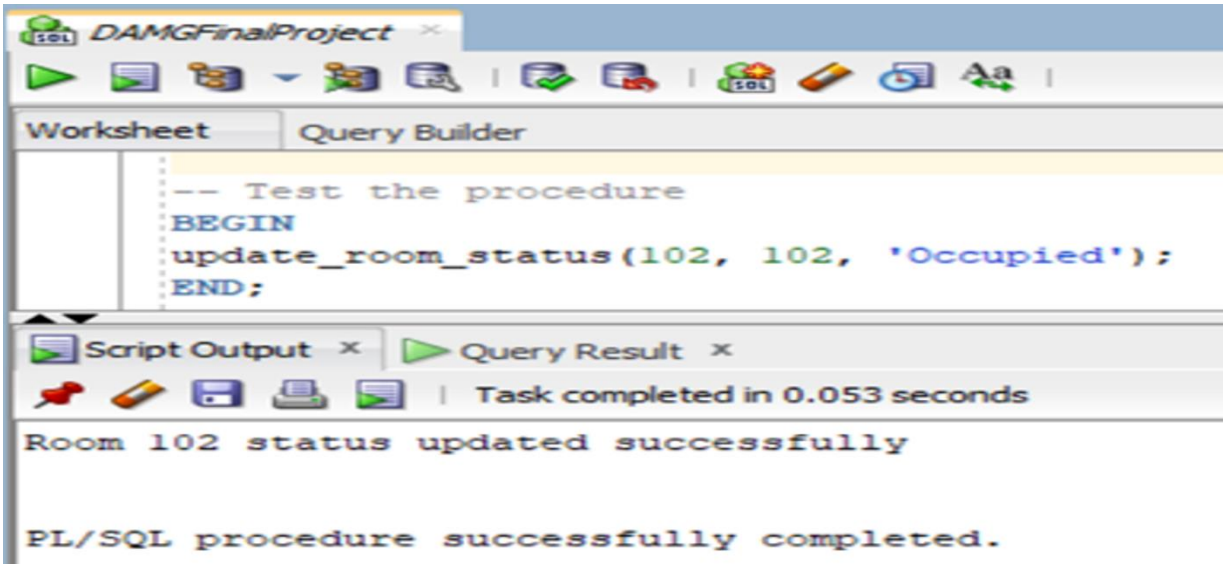


--Check for room status before changes
SELECT * FROM Rooms;

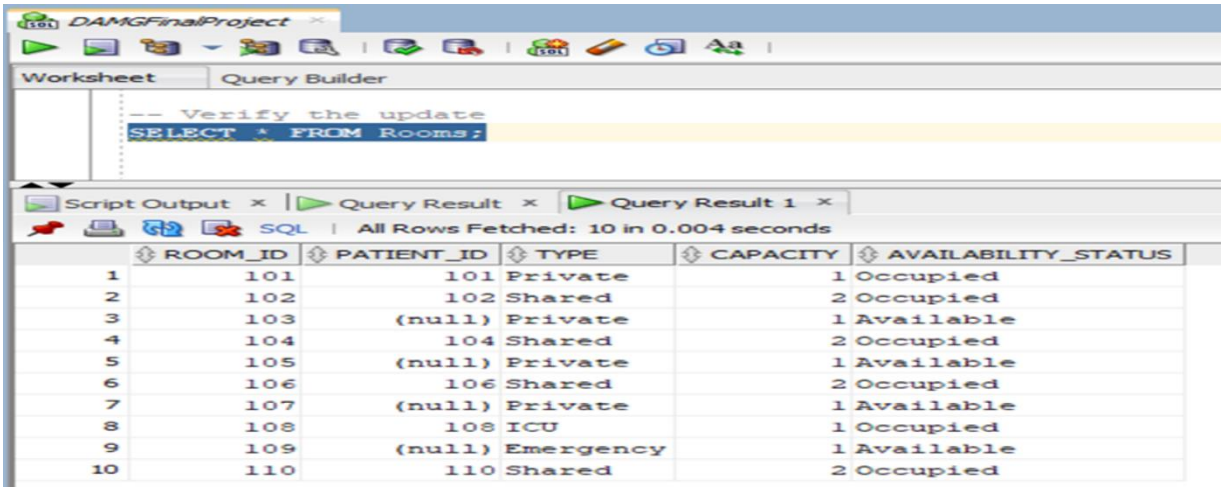


	ROOM_ID	PATIENT_ID	TYPE	CAPACITY	AVAILABILITY_STATUS
1	101	101	Private	1	Occupied
2	102	(null)	Shared	2	Available
3	103	(null)	Private	1	Available
4	104	104	Shared	2	Occupied
5	105	(null)	Private	1	Available
6	106	106	Shared	2	Occupied
7	107	(null)	Private	1	Available
8	108	108	ICU	1	Occupied
9	109	(null)	Emergency	1	Available
10	110	110	Shared	2	Occupied

-- Test the procedure
BEGIN
update_room_status(102, 102, 'Occupied');
END;



-- Verify the update
SELECT * FROM Rooms;

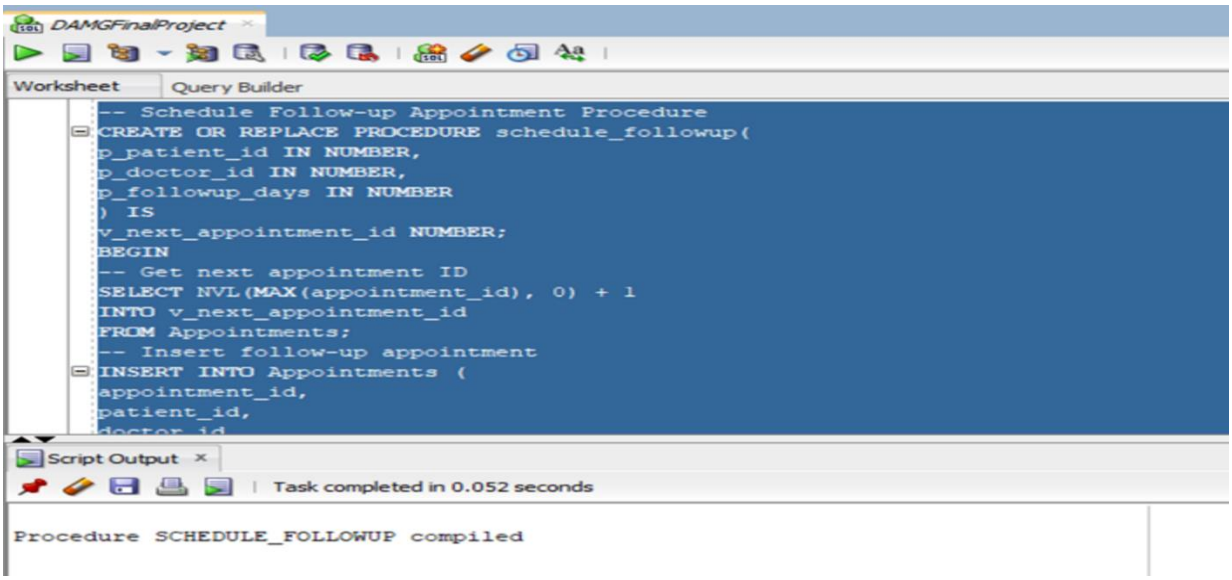


	ROOM_ID	PATIENT_ID	TYPE	CAPACITY	AVAILABILITY_STATUS
1	101	101	Private	1	Occupied
2	102	102	Shared	2	Occupied
3	103	(null)	Private	1	Available
4	104	104	Shared	2	Occupied
5	105	(null)	Private	1	Available
6	106	106	Shared	2	Occupied
7	107	(null)	Private	1	Available
8	108	108	ICU	1	Occupied
9	109	(null)	Emergency	1	Available
10	110	110	Shared	2	Occupied

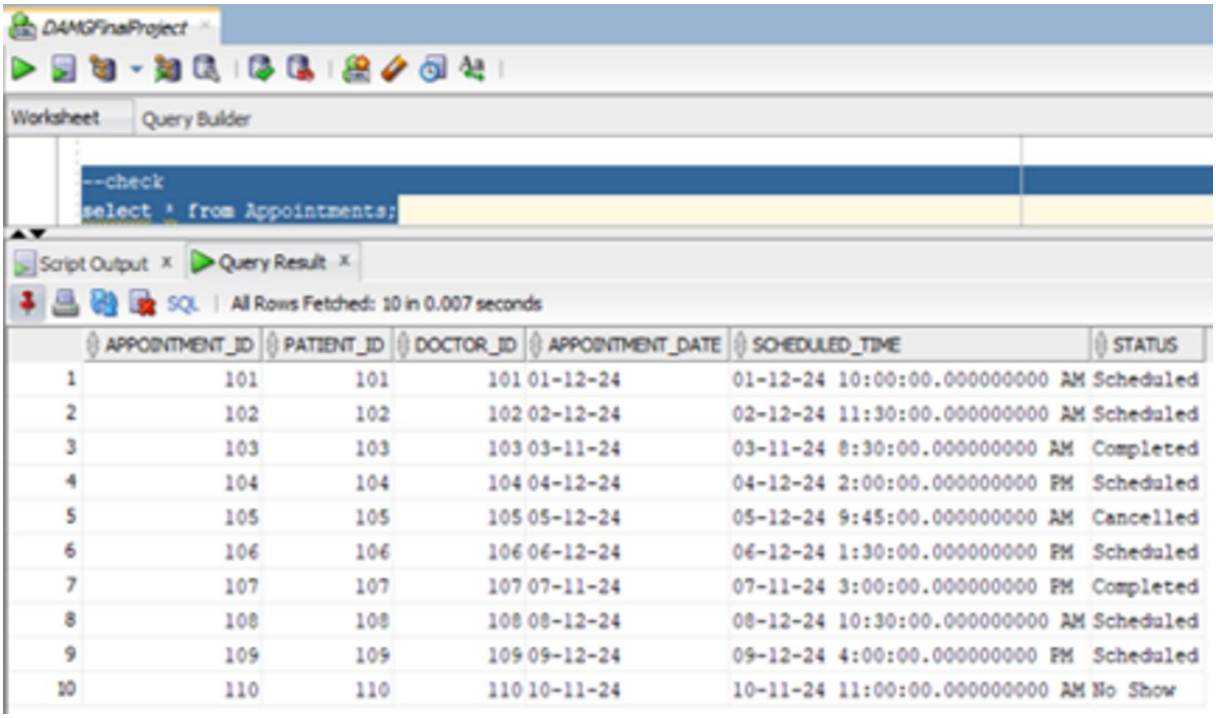
Schedule Follow-up Appointment Procedure

```
CREATE OR REPLACE PROCEDURE schedule_followup(
p_patient_id IN NUMBER,
p_doctor_id IN NUMBER,
p_followup_days IN NUMBER
) IS
v_next_appointment_id NUMBER;
BEGIN
-- Get next appointment ID
SELECT NVL(MAX(appointment_id), 0) + 1
INTO v_next_appointment_id
FROM Appointments;
-- Insert follow-up appointment
INSERT INTO Appointments (
appointment_id,
patient_id,
doctor_id,
appointment_date,
scheduled_time,
status
) VALUES (
v_next_appointment_id,
p_patient_id,
p_doctor_id,
SYSDATE + p_followup_days,
TO_TIMESTAMP(TO_CHAR(SYSDATE + p_followup_days, 'YYYY-MM-DD') || ' 10:00:00',
'YYYY-MM-DD HH24:MI:SS'),
'Scheduled'
```

);
DBMS_OUTPUT.PUT_LINE('Follow-up appointment scheduled for patient ' || p_patient_id);
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error scheduling follow-up: ' || SQLERRM);
END schedule_followup;



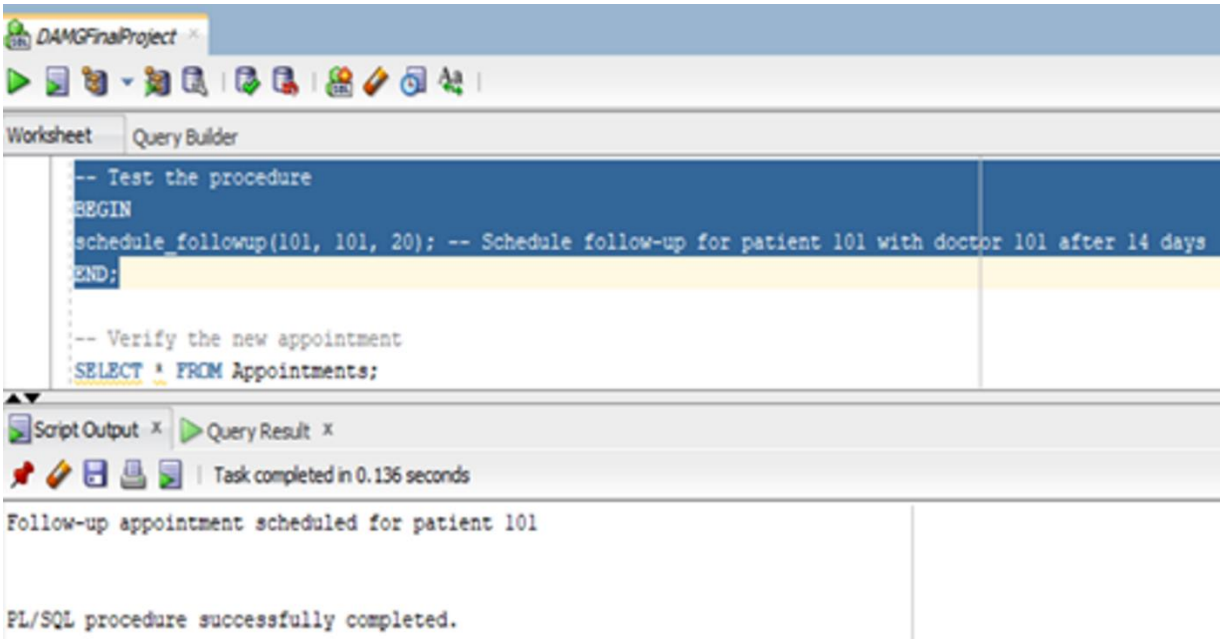
--check
select * from Appointments;



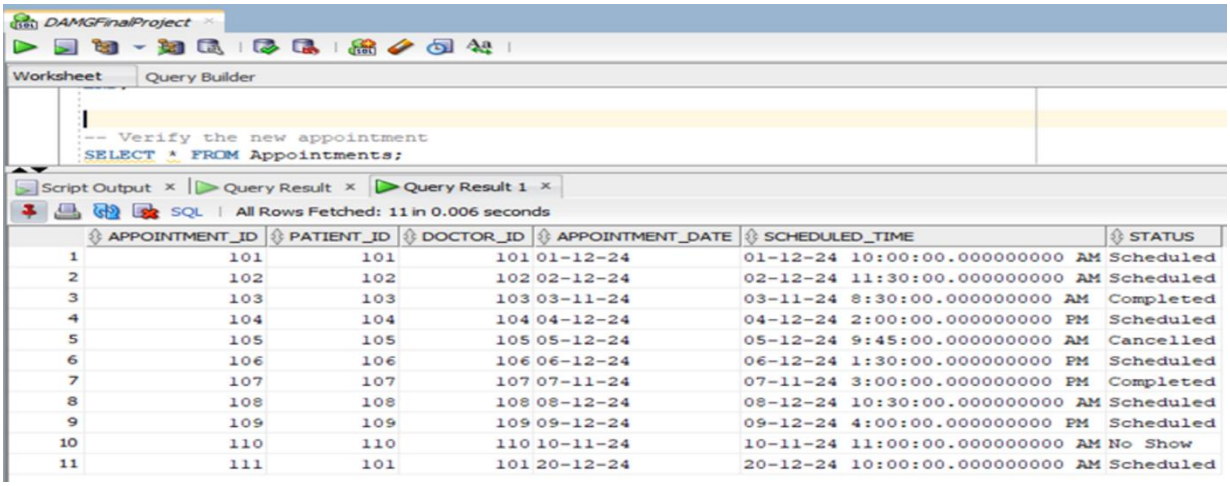
	APPOINTMENT_ID	PATIENT_ID	DOCTOR_ID	APPOINTMENT_DATE	SCHEDULED_TIME	STATUS
1	101	101	101	01-12-24	01-12-24 10:00:00.000000000 AM	Scheduled
2	102	102	102	02-12-24	02-12-24 11:30:00.000000000 AM	Scheduled
3	103	103	103	03-11-24	03-11-24 8:30:00.000000000 AM	Completed
4	104	104	104	04-12-24	04-12-24 2:00:00.000000000 PM	Scheduled
5	105	105	105	05-12-24	05-12-24 9:45:00.000000000 AM	Cancelled
6	106	106	106	06-12-24	06-12-24 1:30:00.000000000 PM	Scheduled
7	107	107	107	07-11-24	07-11-24 3:00:00.000000000 PM	Completed
8	108	108	108	08-12-24	08-12-24 10:30:00.000000000 AM	Scheduled
9	109	109	109	09-12-24	09-12-24 4:00:00.000000000 PM	Scheduled
10	110	110	110	10-11-24	10-11-24 11:00:00.000000000 AM	No Show

-- Test the procedure

BEGIN
schedule_followup(101, 101, 20); -- Schedule follow-up for patient 101 with doctor 101 after 14 days
END;



-- Verify the new appointment
SELECT * FROM Appointments;

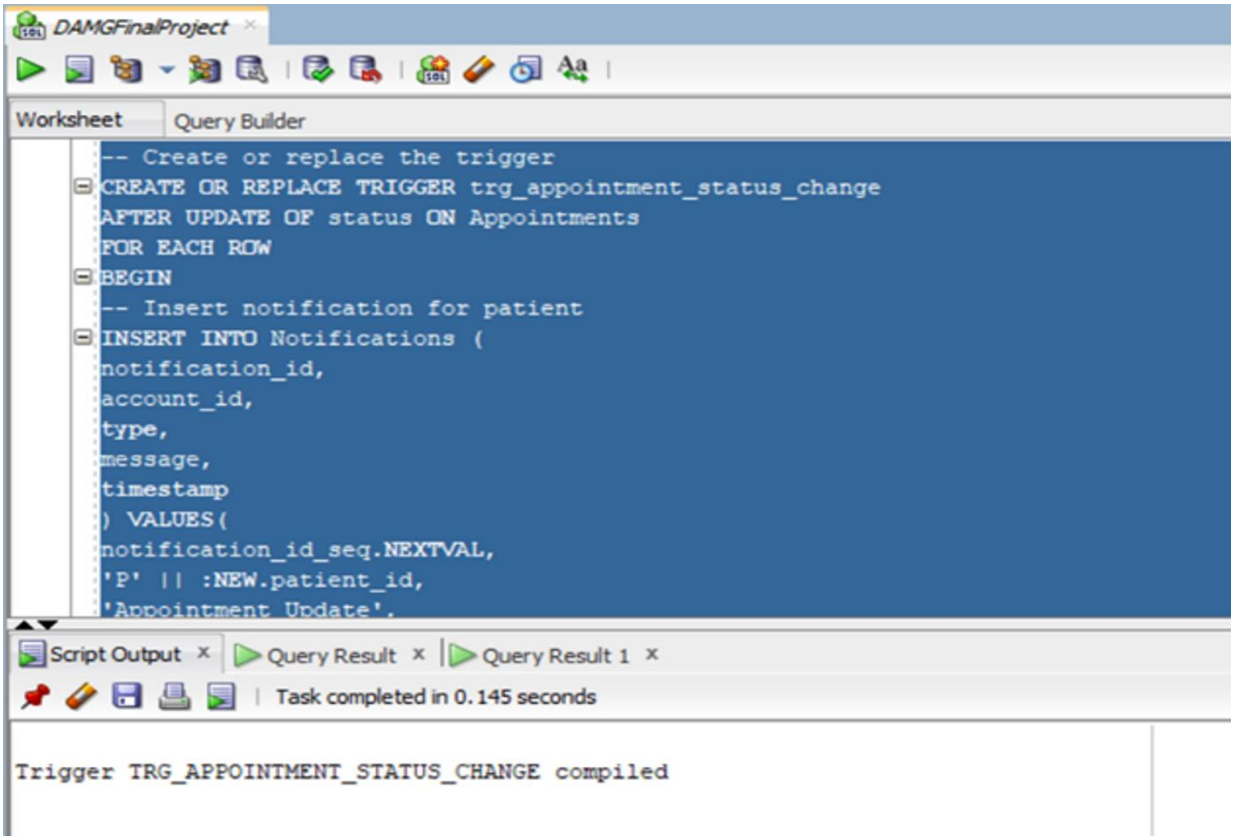


	APPOINTMENT_ID	PATIENT_ID	DOCTOR_ID	APPOINTMENT_DATE	SCHEDULED_TIME	STATUS
1	101	101	101	01-12-24	01-12-24 10:00:00.000000000 AM	Scheduled
2	102	102	102	02-12-24	02-12-24 11:30:00.000000000 AM	Scheduled
3	103	103	103	03-11-24	03-11-24 8:30:00.000000000 AM	Completed
4	104	104	104	04-12-24	04-12-24 2:00:00.000000000 PM	Scheduled
5	105	105	105	05-12-24	05-12-24 9:45:00.000000000 AM	Cancelled
6	106	106	106	06-12-24	06-12-24 1:30:00.000000000 PM	Scheduled
7	107	107	107	07-11-24	07-11-24 3:00:00.000000000 PM	Completed
8	108	108	108	08-12-24	08-12-24 10:30:00.000000000 AM	Scheduled
9	109	109	109	09-12-24	09-12-24 4:00:00.000000000 PM	Scheduled
10	110	110	110	10-11-24	10-11-24 11:00:00.000000000 AM	No Show
11	111	101	101	20-12-24	20-12-24 10:00:00.000000000 AM	Scheduled

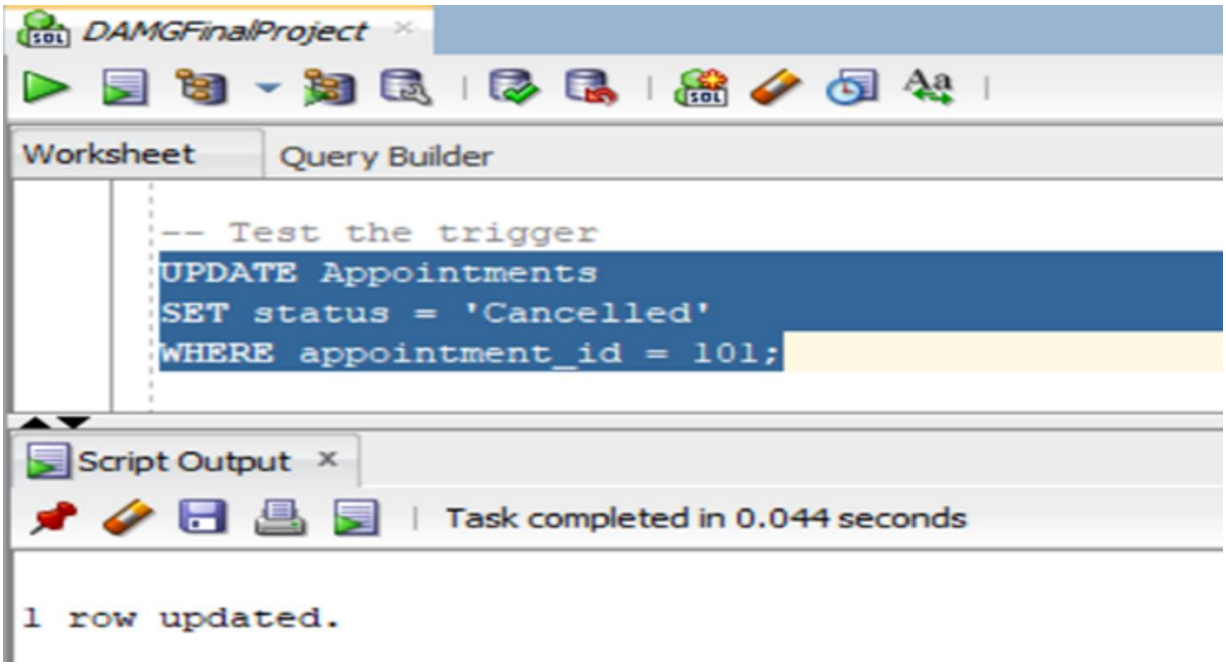
Appointment Status Update Notification Trigger

-- First, create sequence for notification_id if not exists
DROP SEQUENCE notification_id_seq;
CREATE SEQUENCE notification_id_seq
START WITH 1001
INCREMENT BY 1
NOCACHE

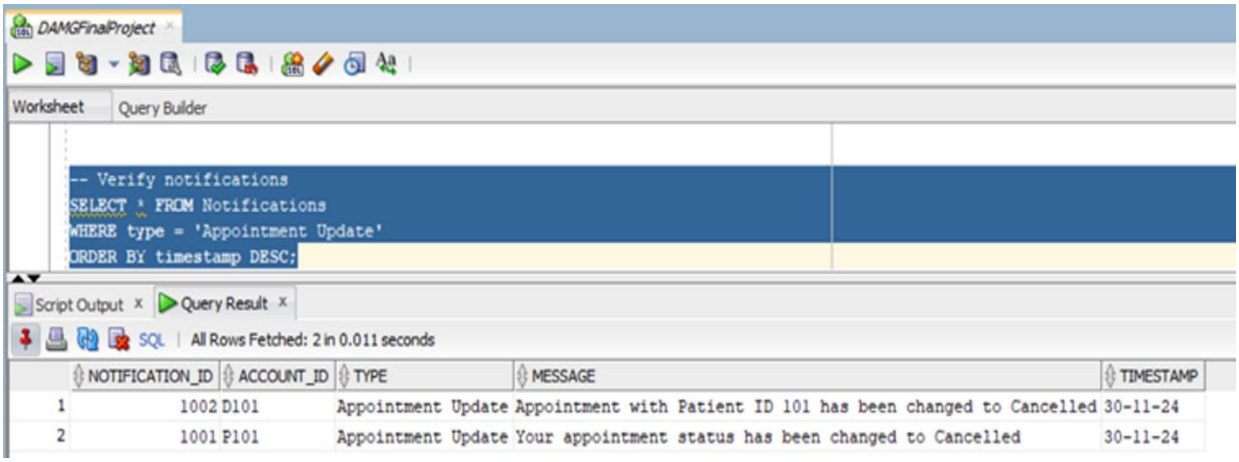
```
NOCYCLE;
-- Create or replace the trigger
CREATE OR REPLACE TRIGGER trg_appointment_status_change
AFTER UPDATE OF status ON Appointments
FOR EACH ROW
BEGIN
-- Insert notification for patient
INSERT INTO Notifications (
notification_id,
account_id,
type,
message,
timestamp
) VALUES (
notification_id_seq.NEXTVAL,
'P' || :NEW.patient_id,
'Appointment Update',
'Your appointment status has been changed to ' || :NEW.status,
SYSDATE
);
-- Insert notification for doctor
INSERT INTO Notifications (
notification_id,
account_id,
type,
message,
timestamp
) VALUES (
notification_id_seq.NEXTVAL,
'D' || :NEW.doctor_id,
'Appointment Update',
'Appointment with Patient ID ' || :NEW.patient_id || ' has been changed to ' || :NEW.status,
SYSDATE
);
END;
```



-- Test the trigger
UPDATE Appointments
SET status = 'Cancelled'
WHERE appointment_id = 101;

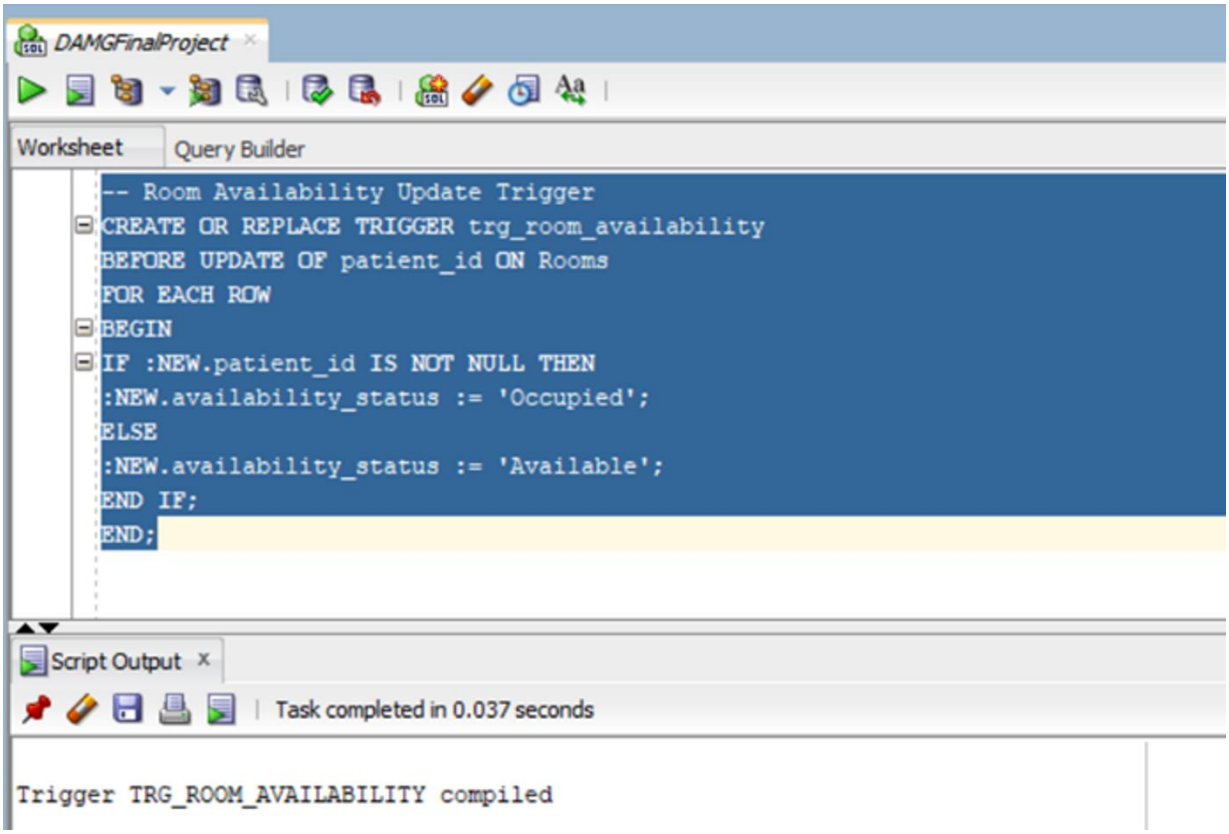


```
-- Verify notifications
SELECT * FROM Notifications
WHERE type = 'Appointment Update'
ORDER BY timestamp DESC;
```

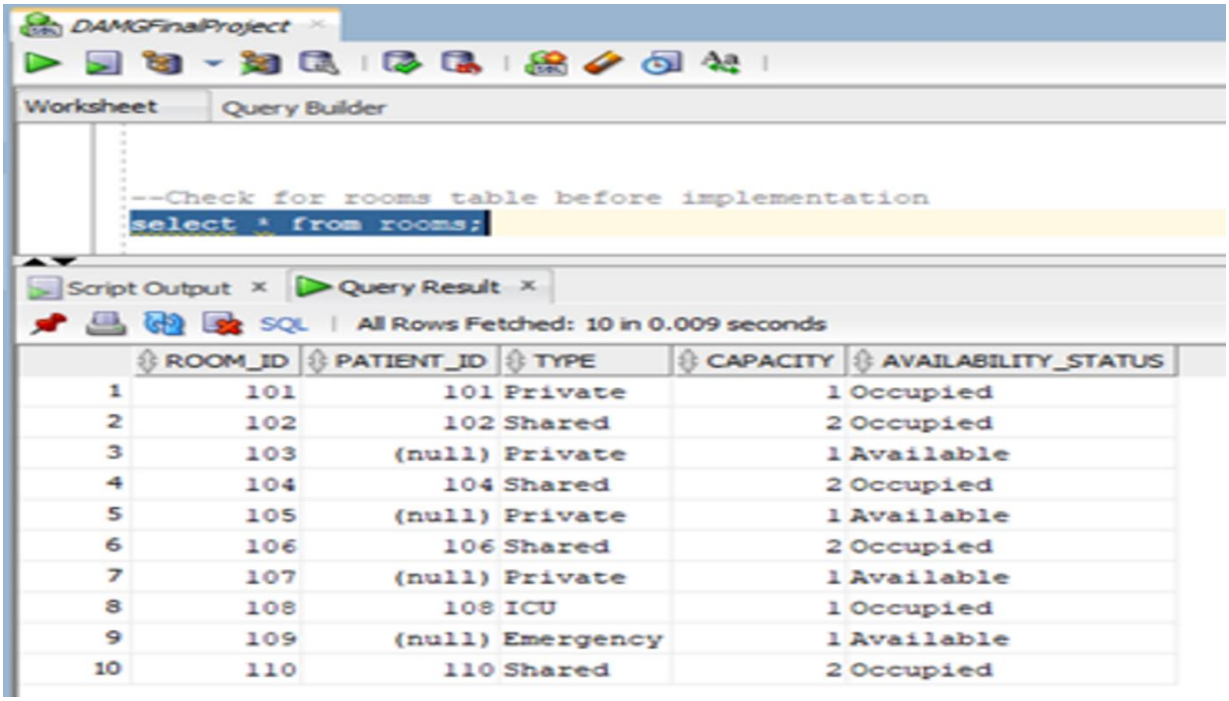


Room Availability Update Trigger

```
CREATE OR REPLACE TRIGGER trg_room_availability
BEFORE UPDATE OF patient_id ON Rooms
FOR EACH ROW
BEGIN
IF :NEW.patient_id IS NOT NULL THEN
:NEW.availability_status := 'Occupied';
ELSE
:NEW.availability_status := 'Available';
END IF;
END;
```

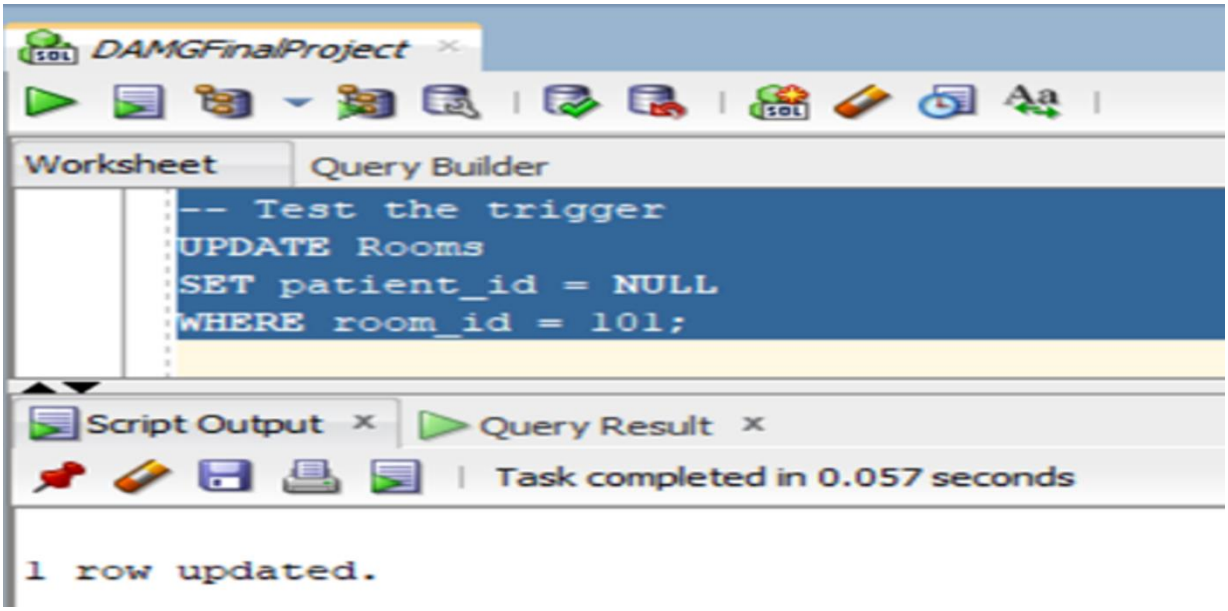


--Check for rooms table before implementation
select * from rooms;

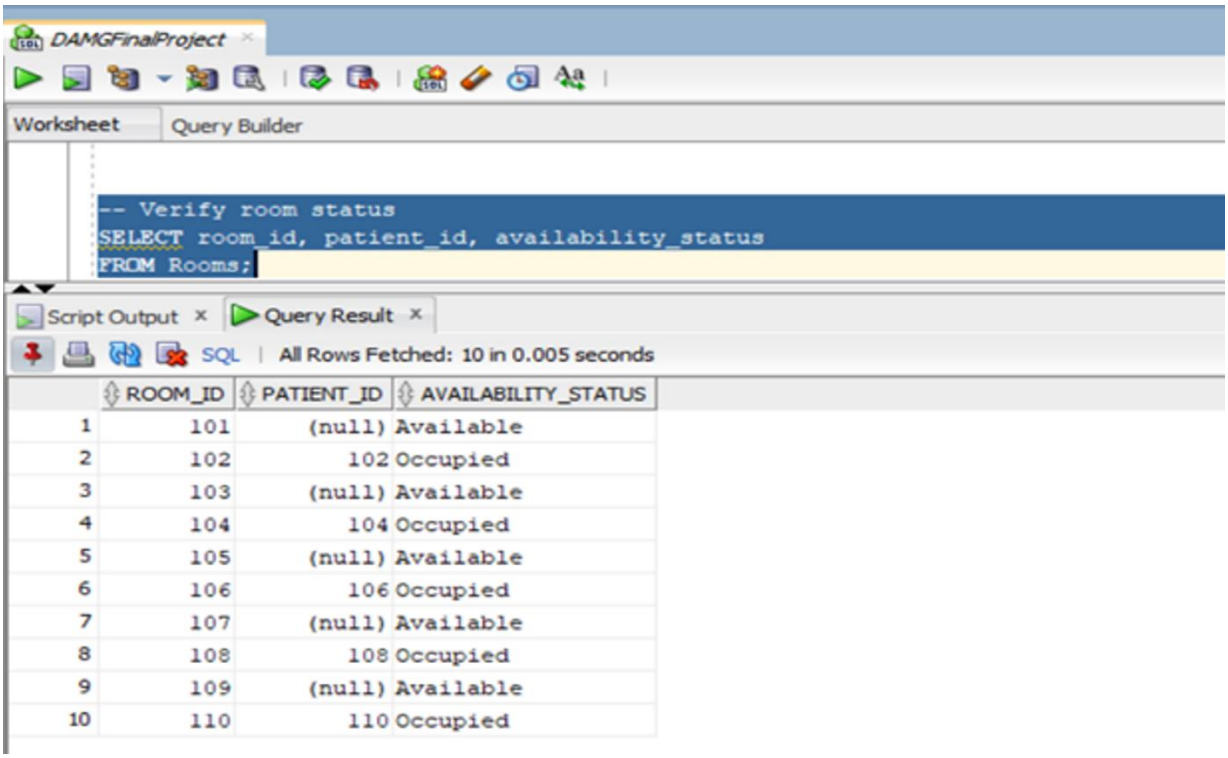


	ROOM_ID	PATIENT_ID	TYPE	CAPACITY	AVAILABILITY_STATUS
1	101	101	Private	1	Occupied
2	102	102	Shared	2	Occupied
3	103	(null)	Private	1	Available
4	104	104	Shared	2	Occupied
5	105	(null)	Private	1	Available
6	106	106	Shared	2	Occupied
7	107	(null)	Private	1	Available
8	108	108	ICU	1	Occupied
9	109	(null)	Emergency	1	Available
10	110	110	Shared	2	Occupied

-- Test the trigger
UPDATE Rooms
SET patient_id = NULL
WHERE room_id = 101;



-- Verify room status
SELECT room_id, patient_id, availability_status
FROM Rooms;

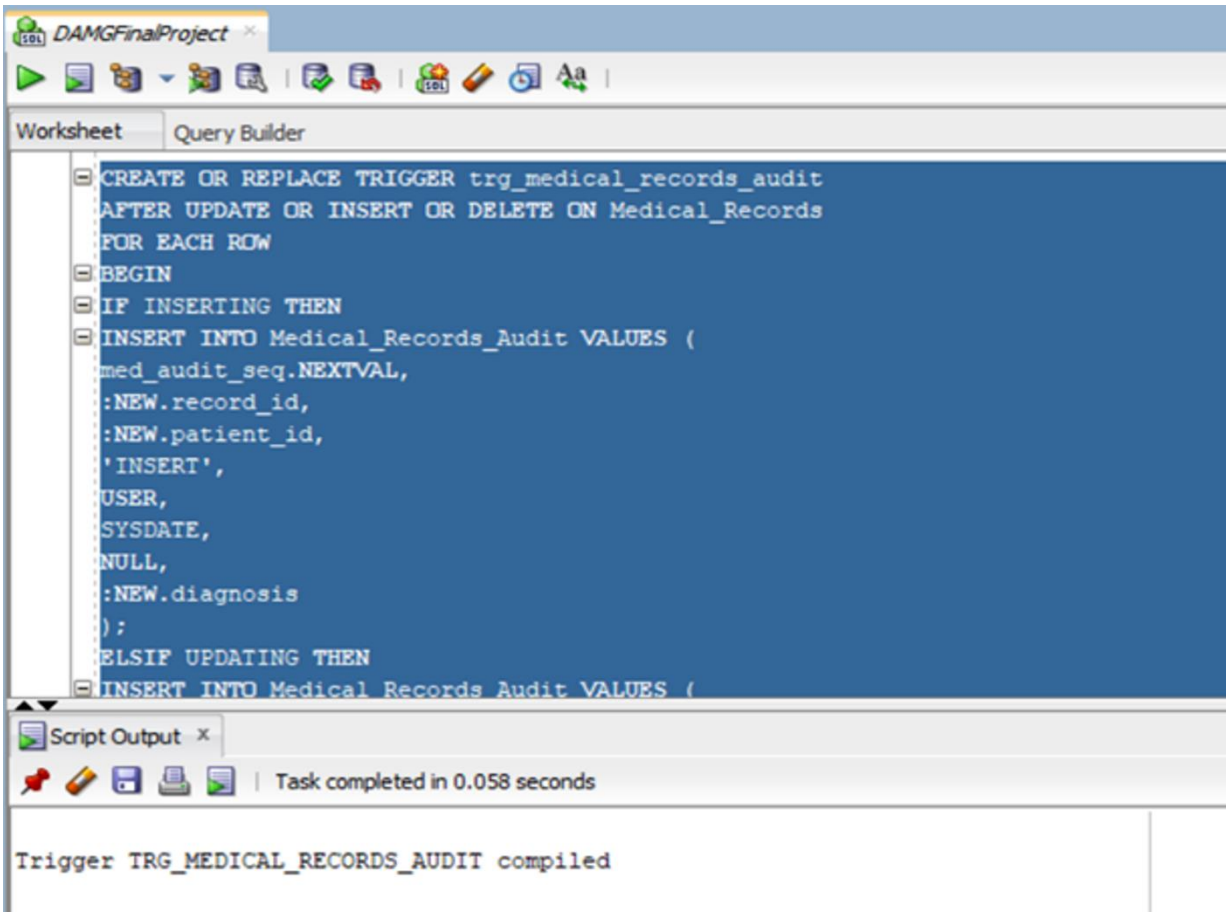


	ROOM_ID	PATIENT_ID	AVAILABILITY_STATUS
1	101	(null)	Available
2	102	102	Occupied
3	103	(null)	Available
4	104	104	Occupied
5	105	(null)	Available
6	106	106	Occupied
7	107	(null)	Available
8	108	108	Occupied
9	109	(null)	Available
10	110	110	Occupied

Medical Record Audit Trigger

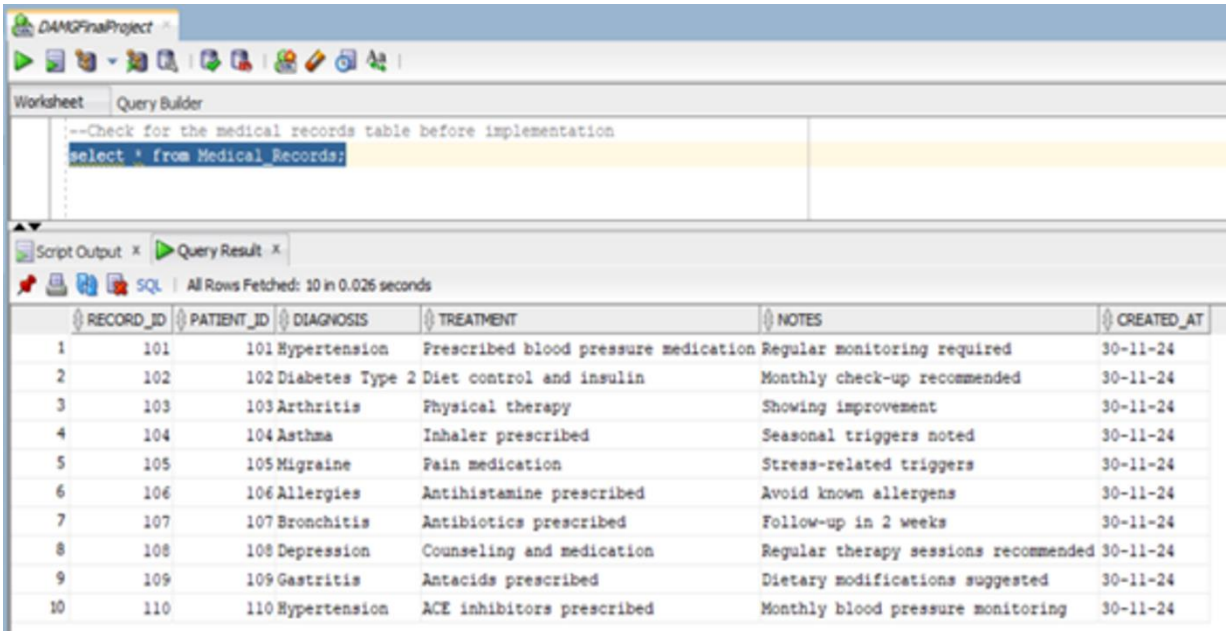
```
-- First create audit table
CREATE TABLE Medical_Records_Audit (
audit_id NUMBER PRIMARY KEY,
record_id NUMBER,
patient_id NUMBER,
action_type VARCHAR2(20),
changed_by VARCHAR2(30),
change_date DATE,
old_diagnosis VARCHAR2(200),
new_diagnosis VARCHAR2(200)
);

DROP SEQUENCE med_audit_seq;
CREATE SEQUENCE med_audit_seq START WITH 1 INCREMENT BY 1;


CREATE OR REPLACE TRIGGER trg_medical_records_audit
AFTER UPDATE OR INSERT OR DELETE ON Medical_Records
FOR EACH ROW
BEGIN
IF INSERTING THEN
INSERT INTO Medical_Records_Audit VALUES (
med_audit_seq.NEXTVAL,
:NEW.record_id,
:NEW.patient_id,
'INSERT',
USER,
SYSDATE,
NULL,
:NEW.diagnosis
);
ELSIF UPDATING THEN
INSERT INTO Medical_Records_Audit VALUES (
med_audit_seq.NEXTVAL,
:OLD.record_id,
:OLD.patient_id,
'UPDATE',
USER,
```

```
SYSDATE,
:OLD.diagnosis,
:NEW.diagnosis
);
ELSIF DELETING THEN
INSERT INTO Medical_Records_Audit VALUES (
med_audit_seq.NEXTVAL,
:OLD.record_id,
:OLD.patient_id,
'DELETE',
USER,
SYSDATE,
:OLD.diagnosis,
NULL
);
END IF;
END;
```



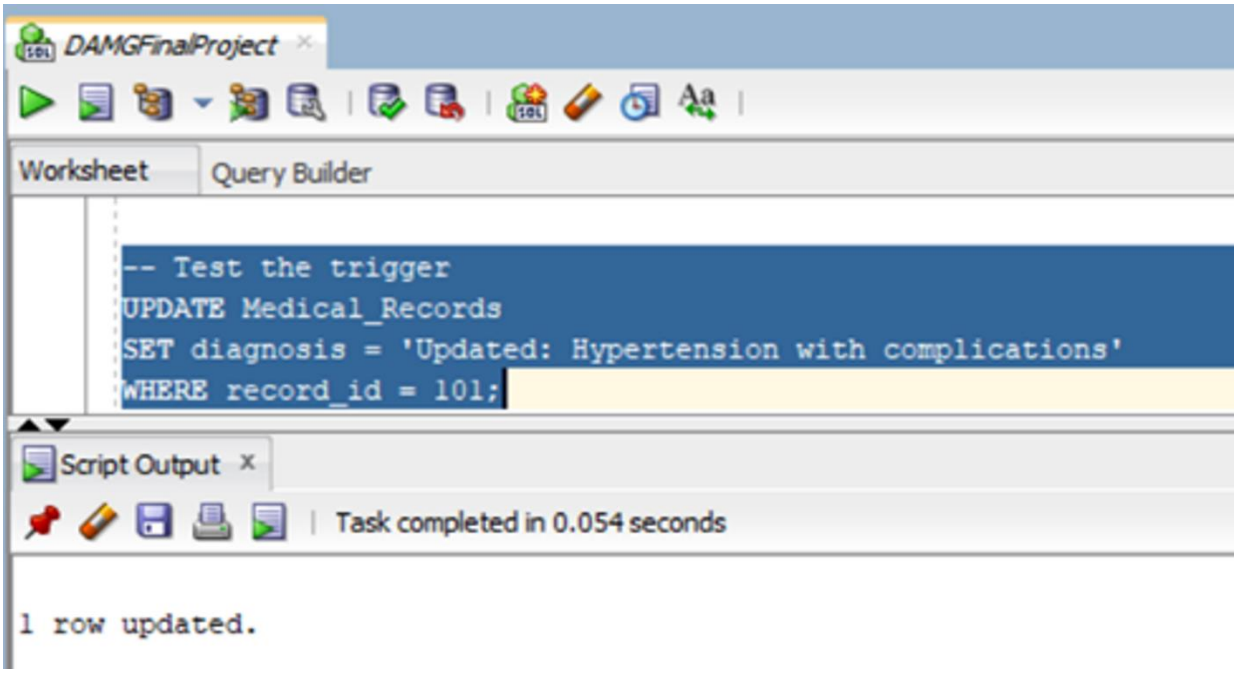
--Check for the medical records table before implementation

select * from Medical_Records;

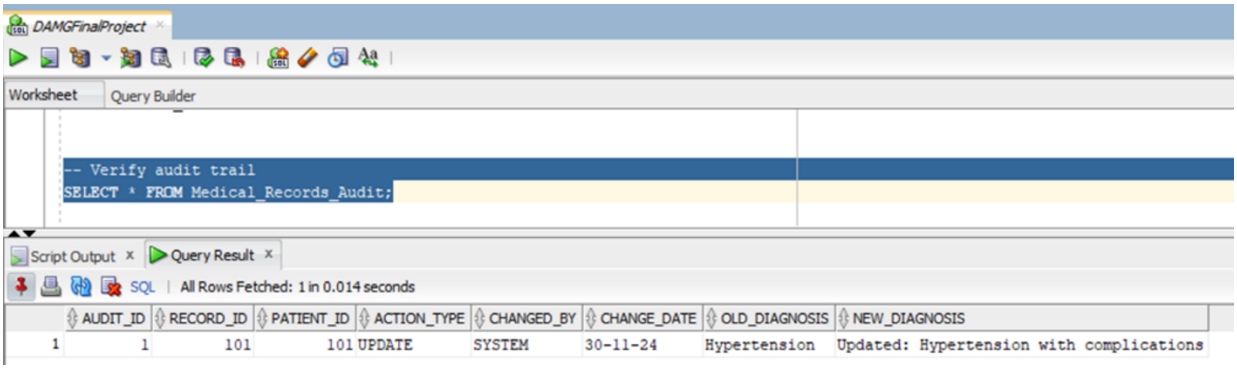


	RECORD_ID	PATIENT_ID	DIAGNOSIS	TREATMENT	NOTES	CREATED_AT
1	101	101	Hypertension	Prescribed blood pressure medication	Regular monitoring required	30-11-24
2	102	102	Diabetes Type 2	Diet control and insulin	Monthly check-up recommended	30-11-24
3	103	103	Arthritis	Physical therapy	Showing improvement	30-11-24
4	104	104	Asthma	Inhaler prescribed	Seasonal triggers noted	30-11-24
5	105	105	Migraine	Pain medication	Stress-related triggers	30-11-24
6	106	106	Allergies	Antihistamine prescribed	Avoid known allergens	30-11-24
7	107	107	Bronchitis	Antibiotics prescribed	Follow-up in 2 weeks	30-11-24
8	108	108	Depression	Counseling and medication	Regular therapy sessions recommended	30-11-24
9	109	109	Gastritis	Antacids prescribed	Dietary modifications suggested	30-11-24
10	110	110	Hypertension	ACE inhibitors prescribed	Monthly blood pressure monitoring	30-11-24

-- Test the trigger
UPDATE Medical_Records
SET diagnosis = 'Updated: Hypertension with complications'
WHERE record_id = 101;



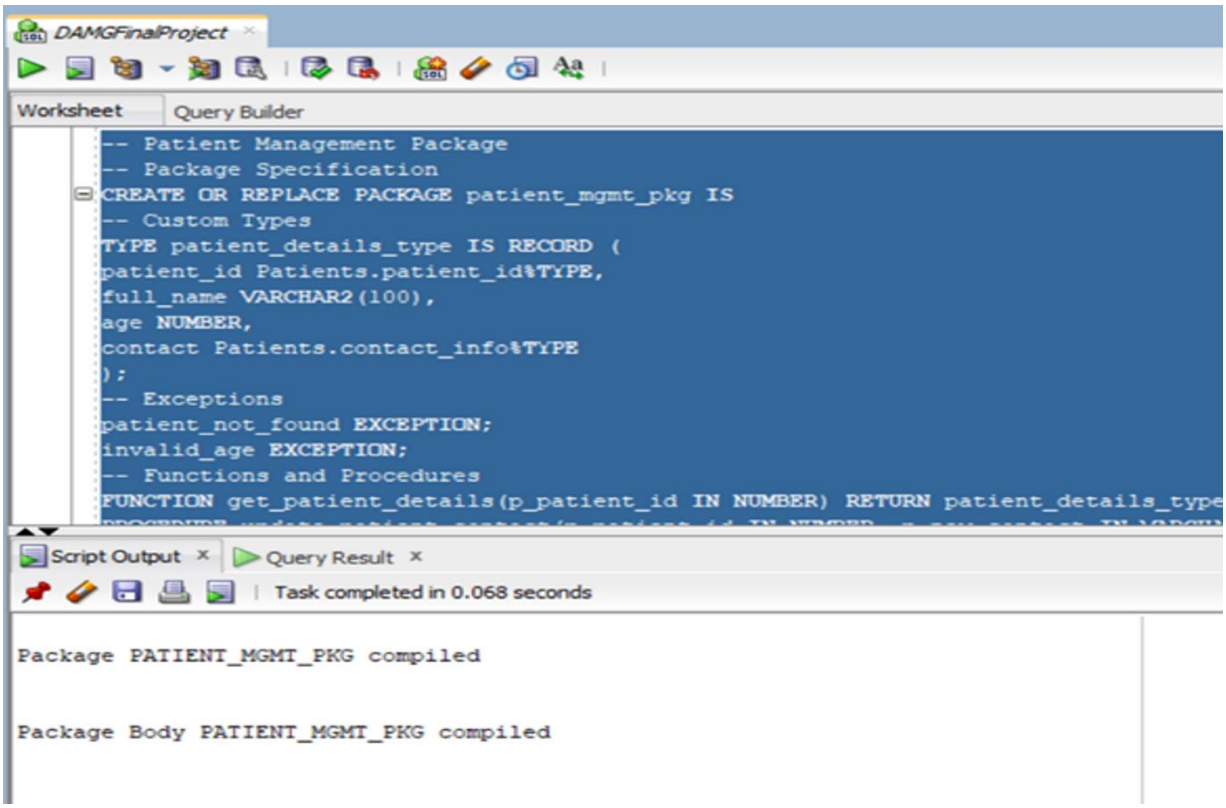
-- Verify audit trail
SELECT * FROM Medical_Records_Audit;



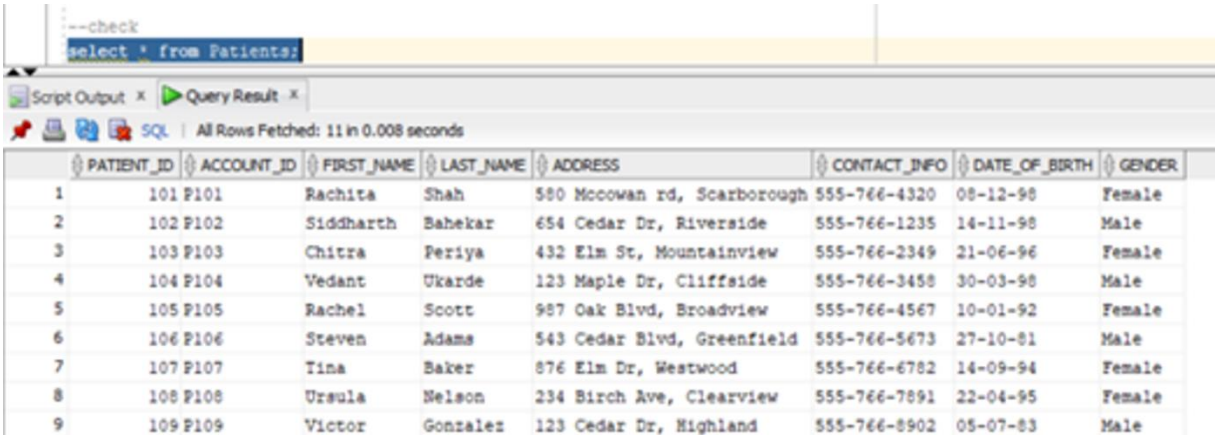
Patient Management Package

-- Package Specification
CREATE OR REPLACE PACKAGE patient_mgmt_pkg IS
-- Custom Types
TYPE patient_details_type IS RECORD (
patient_id Patients.patient_id%TYPE,
full_name VARCHAR2(100),
age NUMBER,
contact Patients.contact_info%TYPE
);
-- Exceptions
patient_not_found EXCEPTION;
invalid_age EXCEPTION;
-- Functions and Procedures
FUNCTION get_patient_details(p_patient_id IN NUMBER) RETURN patient_details_type;
PROCEDURE update_patient_contact(p_patient_id IN NUMBER, p_new_contact IN VARCHAR2);
FUNCTION get_medical_history(p_patient_id IN NUMBER) RETURN SYS_REFCURSOR;
END patient_mgmt_pkg;
-- Package Body
CREATE OR REPLACE PACKAGE BODY patient_mgmt_pkg IS
FUNCTION get_patient_details(p_patient_id IN NUMBER)
RETURN patient_details_type IS
v_details patient_details_type;
v_dob DATE;
BEGIN
SELECT
p.patient_id,
p.first_name || ' ' || p.last_name,
p.contact_info,

```
p.date_of_birth
INTO
v_details.patient_id,
v_details.full_name,
v_details.contact,
v_dob
FROM Patients p
WHERE p.patient_id = p_patient_id;
v_details.age := FLOOR(MONTHS_BETWEEN(SYSDATE, v_dob)/12);
RETURN v_details;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE patient_not_found;
END get_patient_details;
PROCEDURE update_patient_contact(p_patient_id IN NUMBER, p_new_contact IN
VARCHAR2) IS
BEGIN
UPDATE Patients
SET contact_info = p_new_contact
WHERE patient_id = p_patient_id;
IF SQL%NOTFOUND THEN
RAISE patient_not_found;
END IF;
COMMIT;
EXCEPTION
WHEN patient_not_found THEN
RAISE_APPLICATION_ERROR(-20001, 'Patient not found');
END update_patient_contact;
FUNCTION get_medical_history(p_patient_id IN NUMBER)
RETURN SYS_REFCURSOR IS
v_result SYS_REFCURSOR;
BEGIN
OPEN v_result FOR
SELECT mr.diagnosis, mr.treatment, mr.created_at
FROM Medical_Records mr
WHERE mr.patient_id = p_patient_id
ORDER BY mr.created_at DESC;
RETURN v_result;
END get_medical_history;
END patient_mgmt_pkg;
```



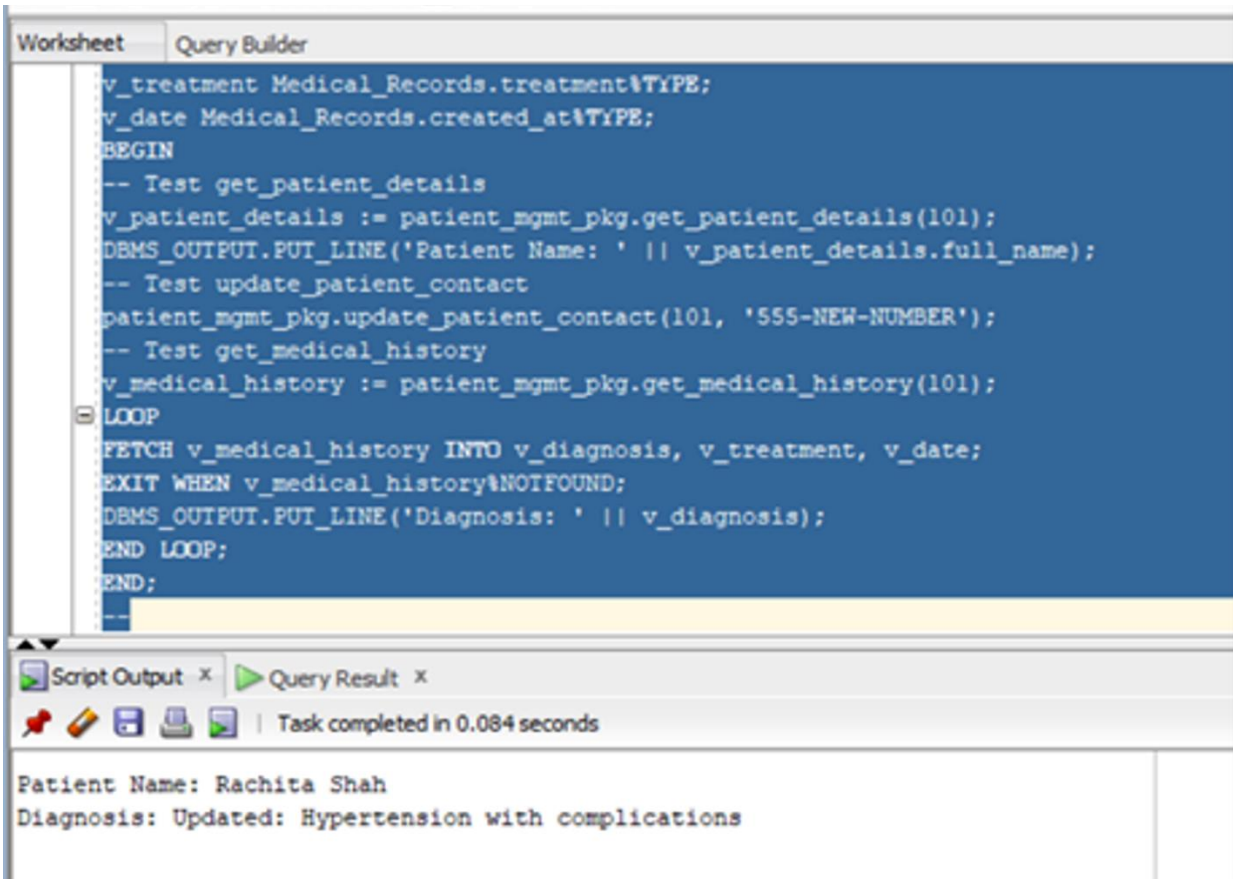
--check
select * from Patients;



	PATIENT_ID	ACCOUNT_ID	FIRST_NAME	LAST_NAME	ADDRESS	CONTACT_INFO	DATE_OF_BIRTH	GENDER
1	101	P101	Rachita	Shah	580 Mccowan rd, Scarborough	555-766-4320	08-12-98	Female
2	102	P102	Siddharth	Bahekar	654 Cedar Dr, Riverside	555-766-1235	14-11-98	Male
3	103	P103	Chitra	Periya	432 Elm St, Mountainview	555-766-2349	21-06-96	Female
4	104	P104	Vedant	Ukarde	123 Maple Dr, Cliffside	555-766-3458	30-03-98	Male
5	105	P105	Rachel	Scott	987 Oak Blvd, Broadview	555-766-4567	10-01-92	Female
6	106	P106	Steven	Adams	543 Cedar Blvd, Greenfield	555-766-5673	27-10-81	Male
7	107	P107	Tina	Baker	876 Elm Dr, Westwood	555-766-6782	14-09-94	Female
8	108	P108	Ursula	Nelson	234 Birch Ave, Clearview	555-766-7891	22-04-95	Female
9	109	P109	Victor	Gonzalez	123 Cedar Dr, Highland	555-766-8902	05-07-83	Male

-- Test Patient Management Package
DECLARE
v_patient_details patient_mgmt_pkg.patient_details_type;
v_medical_history SYS_REFCURSOR;
v_diagnosis Medical_Records.diagnosis%TYPE;

```
v_treatment Medical_Records.treatment%TYPE;
v_date Medical_Records.created_at%TYPE;
BEGIN
-- Test get_patient_details
v_patient_details := patient_mgmt_pkg.get_patient_details(101);
DBMS_OUTPUT.PUT_LINE('Patient Name: ' || v_patient_details.full_name);
-- Test update_patient_contact
patient_mgmt_pkg.update_patient_contact(101, '555-NEW-NUMBER');
-- Test get_medical_history
v_medical_history := patient_mgmt_pkg.get_medical_history(101);
LOOP
FETCH v_medical_history INTO v_diagnosis, v_treatment, v_date;
EXIT WHEN v_medical_history%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Diagnosis: ' || v_diagnosis);
END LOOP;
END;
```

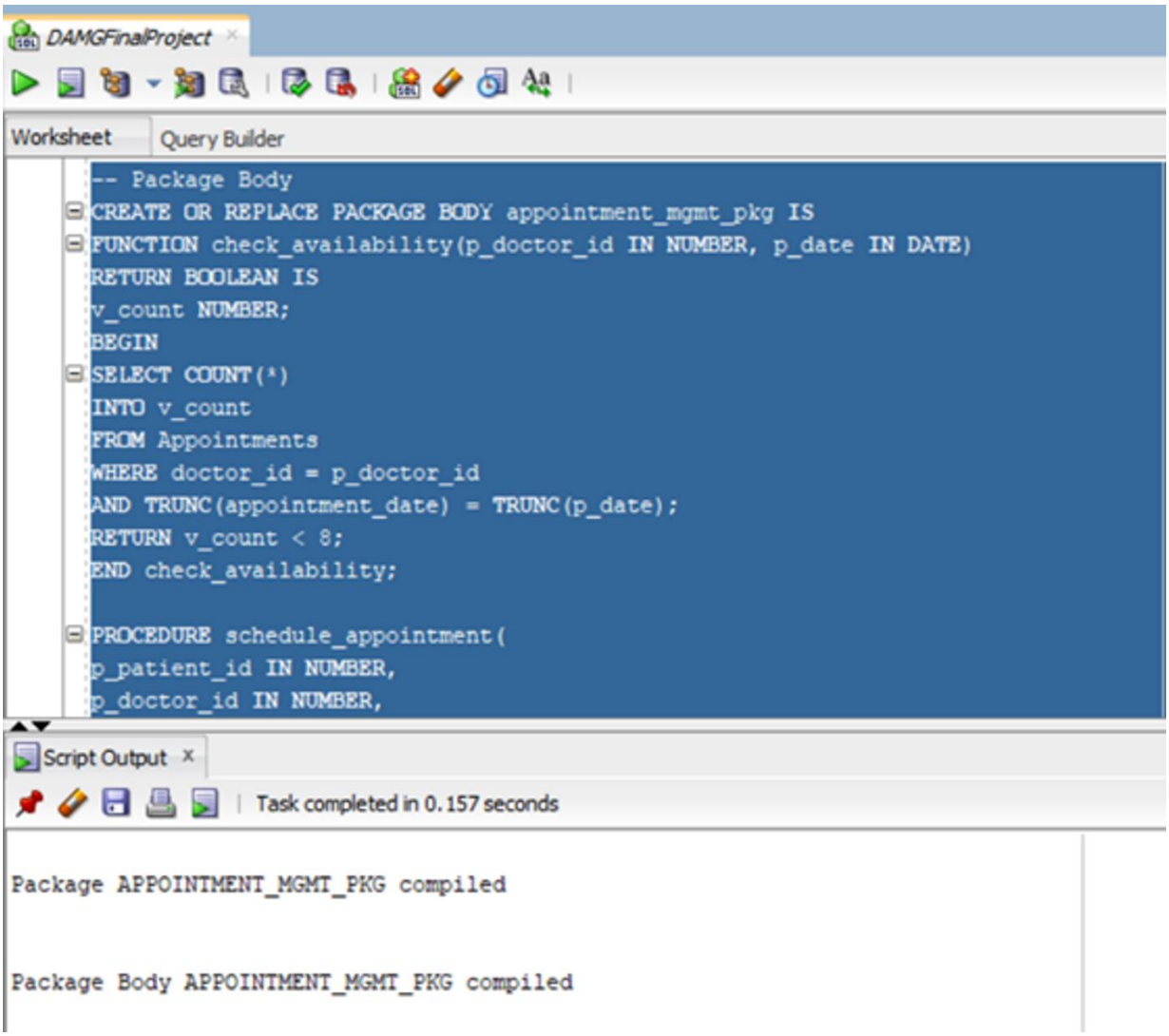


Appointment Management Package

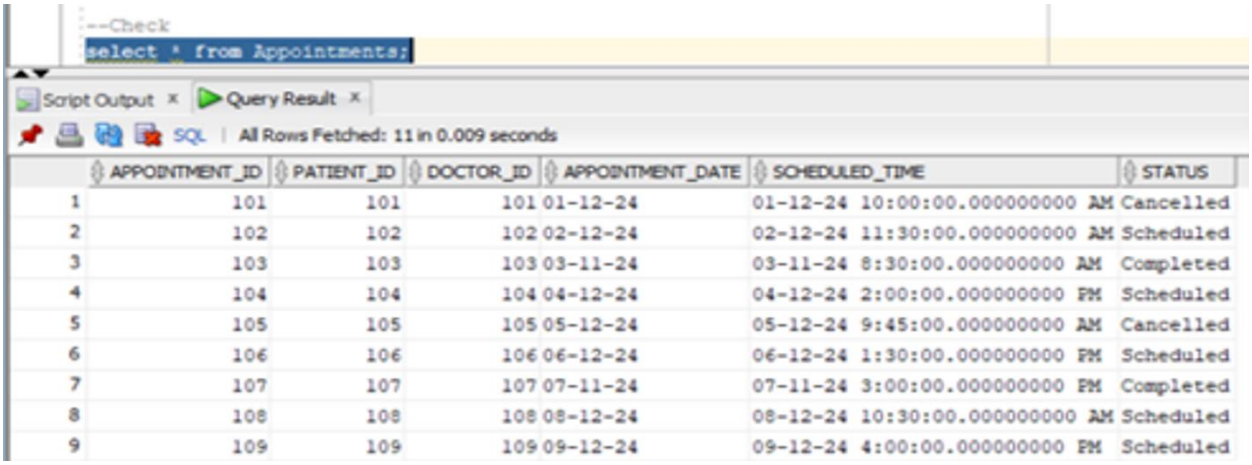
```
-- Package Specification
CREATE OR REPLACE PACKAGE appointment_mgmt_pkg IS
TYPE appointment_type IS RECORD (

appointment_id Appointments.appointment_id%TYPE,
patient_name VARCHAR2(100),
doctor_name VARCHAR2(100),
appointment_date DATE,
status Appointments.status%TYPE
);

slot_not_available EXCEPTION;
invalid_appointment EXCEPTION;
FUNCTION check_availability(p_doctor_id IN NUMBER, p_date IN DATE) RETURN
BOOLEAN;
PROCEDURE schedule_appointment(
p_patient_id IN NUMBER,
p_doctor_id IN NUMBER,
p_date IN DATE,
p_time IN VARCHAR2
);
FUNCTION get_appointments_by_doctor(p_doctor_id IN NUMBER) RETURN
SYS_REFCURSOR;
END appointment_mgmt_pkg;
-- Package Body
CREATE OR REPLACE PACKAGE BODY appointment_mgmt_pkg IS
FUNCTION check_availability(p_doctor_id IN NUMBER, p_date IN DATE)
RETURN BOOLEAN IS
v_count NUMBER;
BEGIN
SELECT COUNT(*)
INTO v_count
FROM Appointments
WHERE doctor_id = p_doctor_id
AND TRUNC(appointment_date) = TRUNC(p_date);

RETURN v_count < 8;
END check_availability;
PROCEDURE schedule_appointment(
```

```
p_patient_id IN NUMBER,
p_doctor_id IN NUMBER,
p_date IN DATE,
p_time IN VARCHAR2
) IS
v_timestamp TIMESTAMP;
BEGIN
IF NOT check_availability(p_doctor_id, p_date) THEN
RAISE slot_not_available;
END IF;
-- Create timestamp from date and time
v_timestamp := TO_TIMESTAMP(TO_CHAR(p_date, 'YYYY-MM-DD') || ' ' || p_time,
'YYYY-MM-DD HH24:MI');
INSERT INTO Appointments (
appointment_id,
patient_id,
doctor_id,
appointment_date,
scheduled_time,
status
) VALUES (
appointment_id_seq.NEXTVAL,
p_patient_id,
p_doctor_id,
p_date,
v_timestamp,
'Scheduled'
);
COMMIT;
EXCEPTION
WHEN slot_not_available THEN
RAISE_APPLICATION_ERROR(-20002, 'No available slots for this date');
WHEN OTHERS THEN
ROLLBACK;
RAISE_APPLICATION_ERROR(-20003, 'Error scheduling appointment: ' || SQLERRM);
END schedule_appointment;
FUNCTION get_appointments_by_doctor(p_doctor_id IN NUMBER)
RETURN SYS_REFCURSOR IS
v_result SYS_REFCURSOR;
BEGIN
```

```
OPEN v_result FOR
SELECT
a.appointment_id,
p.first_name || ' ' || p.last_name as patient_name,
a.appointment_date,
a.status
FROM Appointments a
JOIN Patients p ON a.patient_id = p.patient_id
WHERE a.doctor_id = p_doctor_id
ORDER BY a.appointment_date;
RETURN v_result;
END get_appointments_by_doctor;
END appointment_mgmt_pkg;
```



--Check
select * from Appointments;



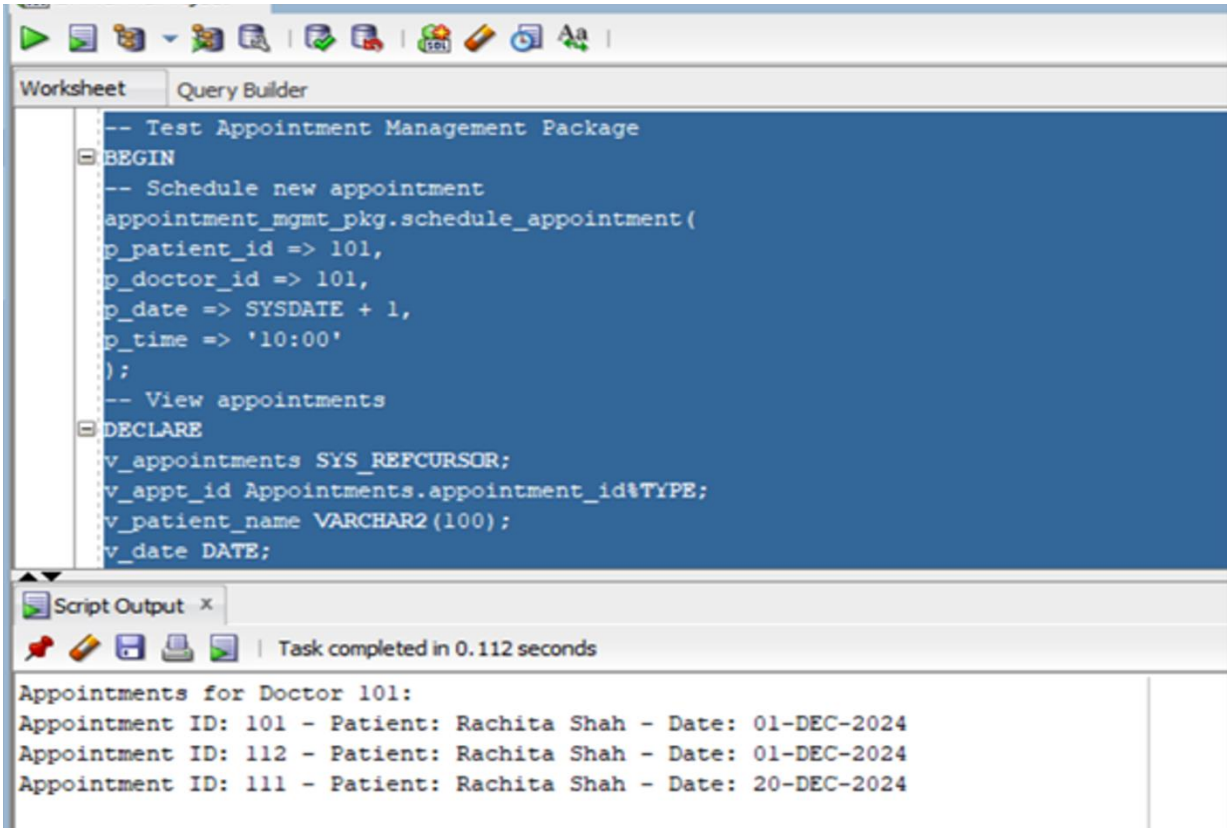
```
--Check
select * from Appointments;
```

Script Output | Query Result | All Rows Fetched: 11 in 0.009 seconds

	APPOINTMENT_ID	PATIENT_ID	DOCTOR_ID	APPOINTMENT_DATE	SCHEDULED_TIME	STATUS
1	101	101	101	01-12-24	01-12-24 10:00:00.000000000 AM	Cancelled
2	102	102	102	02-12-24	02-12-24 11:30:00.000000000 AM	Scheduled
3	103	103	103	03-11-24	03-11-24 8:30:00.000000000 AM	Completed
4	104	104	104	04-12-24	04-12-24 2:00:00.000000000 PM	Scheduled
5	105	105	105	05-12-24	05-12-24 9:45:00.000000000 AM	Cancelled
6	106	106	106	06-12-24	06-12-24 1:30:00.000000000 PM	Scheduled
7	107	107	107	07-11-24	07-11-24 3:00:00.000000000 PM	Completed
8	108	108	108	08-12-24	08-12-24 10:30:00.000000000 AM	Scheduled
9	109	109	109	09-12-24	09-12-24 4:00:00.000000000 PM	Scheduled

```
-- Test Appointment Management Package
BEGIN
-- Schedule new appointment
appointment_mgmt_pkg.schedule_appointment(
p_patient_id => 101,
p_doctor_id => 101,
p_date => SYSDATE + 1,
p_time => '10:00'
);
-- View appointments
DECLARE
v_appointments SYS_REFCURSOR;
v_appt_id Appointments.appointment_id%TYPE;
v_patient_name VARCHAR2(100);
v_date DATE;
v_status VARCHAR2(20);
BEGIN
v_appointments := appointment_mgmt_pkg.get_appointments_by_doctor(101);
DBMS_OUTPUT.PUT_LINE('Appointments for Doctor 101:');
LOOP
FETCH v_appointments INTO v_appt_id, v_patient_name, v_date, v_status;
EXIT WHEN v_appointments%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Appointment ID: ' || v_appt_id ||
' - Patient: ' || v_patient_name ||
' - Date: ' || TO_CHAR(v_date, 'DD-MON-YYYY'));
END LOOP;
```

CLOSE v_appointments;
END;
END;
--

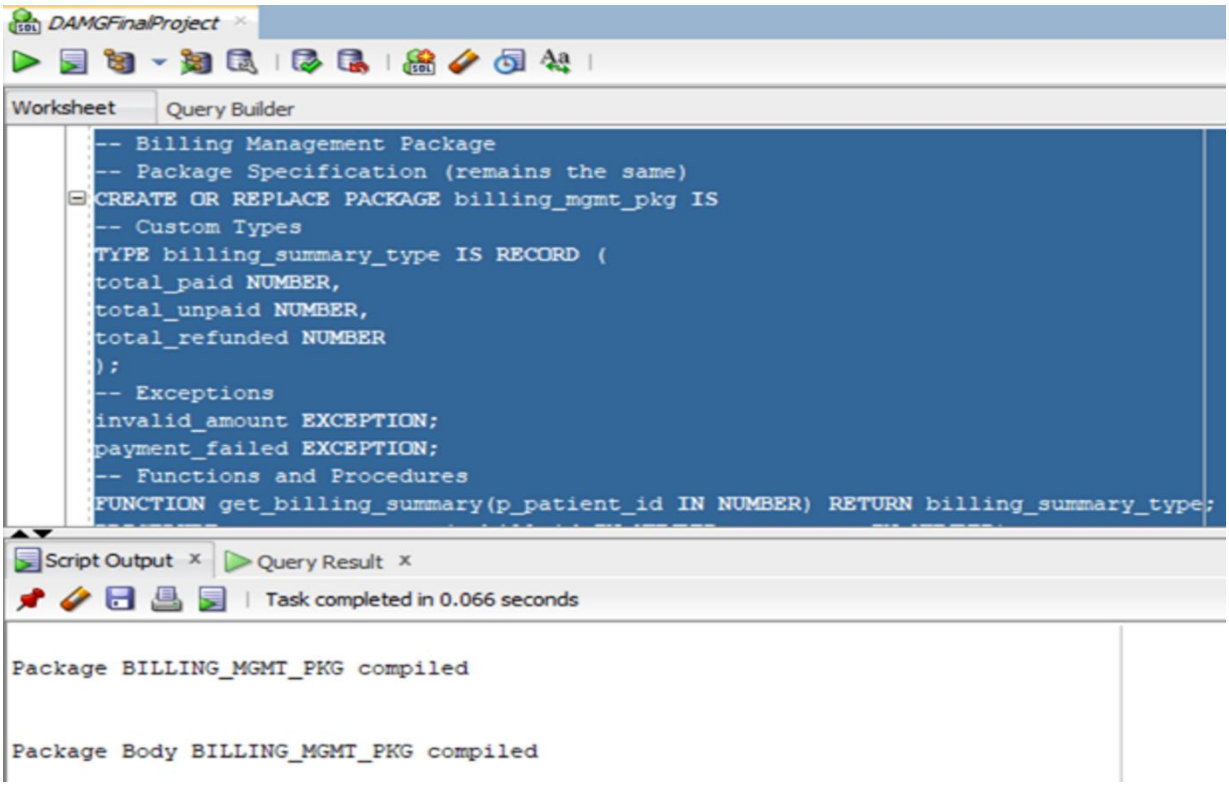


Billing Management Package

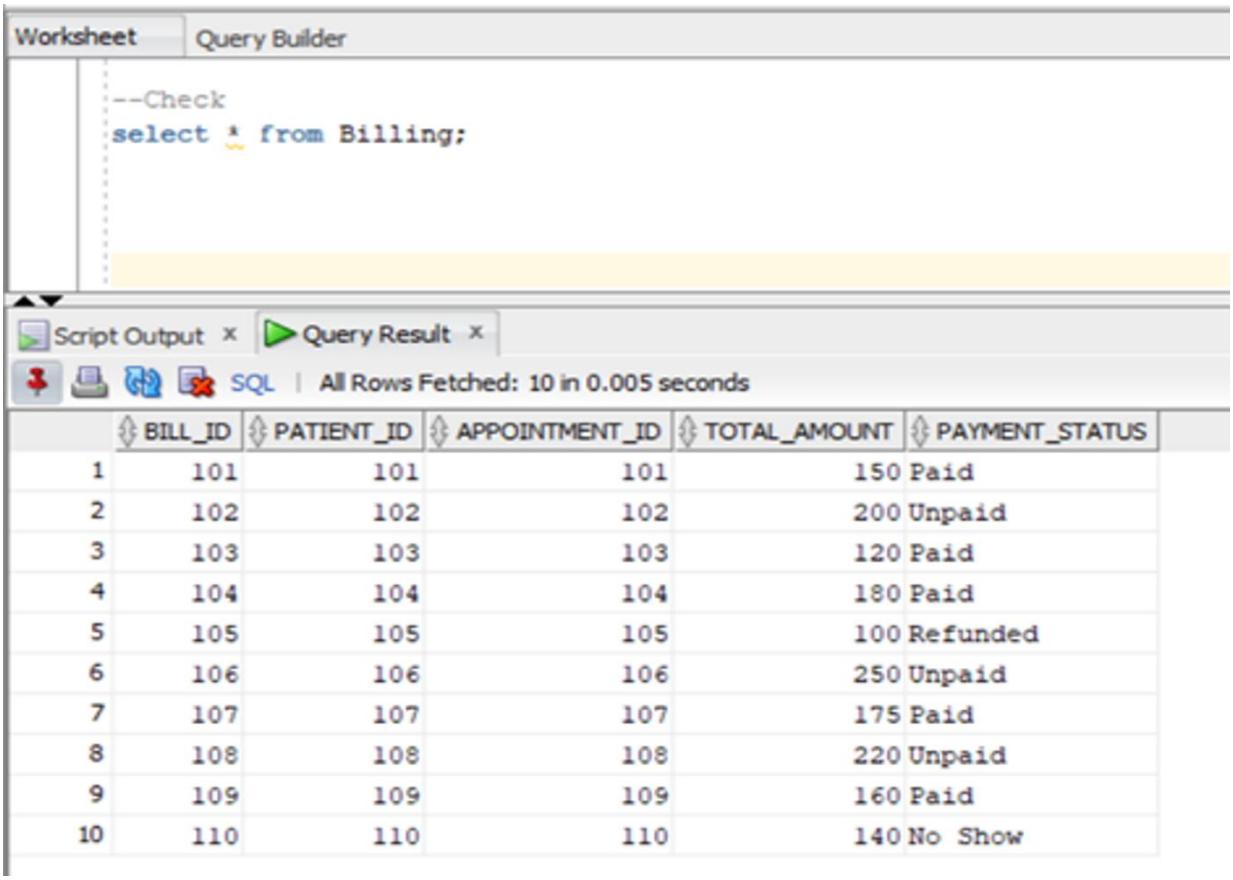
-- Package Specification (remains the same)
CREATE OR REPLACE PACKAGE billing_mgmt_pkg IS
-- Custom Types
TYPE billing_summary_type IS RECORD (
total_paid NUMBER,
total_unpaid NUMBER,
total_refunded NUMBER
);
-- Exceptions
invalid_amount EXCEPTION;
payment_failed EXCEPTION;
-- Functions and Procedures

```
FUNCTION get_billing_summary(p_patient_id IN NUMBER) RETURN
billing_summary_type;
PROCEDURE process_payment(p_bill_id IN NUMBER, p_amount IN NUMBER);
FUNCTION get_unpaid_bills(p_patient_id IN NUMBER) RETURN SYS_REFCURSOR;
END billing_mgmt_pkg;
-- Package Body
CREATE OR REPLACE PACKAGE BODY billing_mgmt_pkg IS
FUNCTION get_billing_summary(p_patient_id IN NUMBER)
RETURN billing_summary_type IS
v_summary billing_summary_type;
BEGIN
SELECT
SUM(CASE WHEN payment_status = 'Paid' THEN total_amount ELSE 0 END),
SUM(CASE WHEN payment_status = 'Unpaid' THEN total_amount ELSE 0 END),
SUM(CASE WHEN payment_status = 'Refunded' THEN total_amount ELSE 0 END)
INTO
v_summary.total_paid,
v_summary.total_unpaid,
v_summary.total_refunded
FROM Billing
WHERE patient_id = p_patient_id;
RETURN v_summary;
END get_billing_summary;
PROCEDURE process_payment(p_bill_id IN NUMBER, p_amount IN NUMBER) IS
v_bill_amount NUMBER;
BEGIN
SELECT total_amount
INTO v_bill_amount
FROM Billing
WHERE bill_id = p_bill_id;
IF p_amount < v_bill_amount THEN
RAISE invalid_amount;
END IF;
UPDATE Billing
SET payment_status = 'Paid'
WHERE bill_id = p_bill_id;
COMMIT;
EXCEPTION
WHEN invalid_amount THEN
RAISE_APPLICATION_ERROR(-20003, 'Payment amount is less than bill amount');
```

```
END process_payment;
FUNCTION get_unpaid_bills(p_patient_id IN NUMBER)
RETURN SYS_REFCURSOR IS
v_result SYS_REFCURSOR;
BEGIN
OPEN v_result FOR
SELECT bill_id, total_amount, appointment_id -- Added appointment_id
FROM Billing
WHERE patient_id = p_patient_id
AND payment_status = 'Unpaid';
RETURN v_result;
END get_unpaid_bills;
END billing_mgmt_pkg;
```



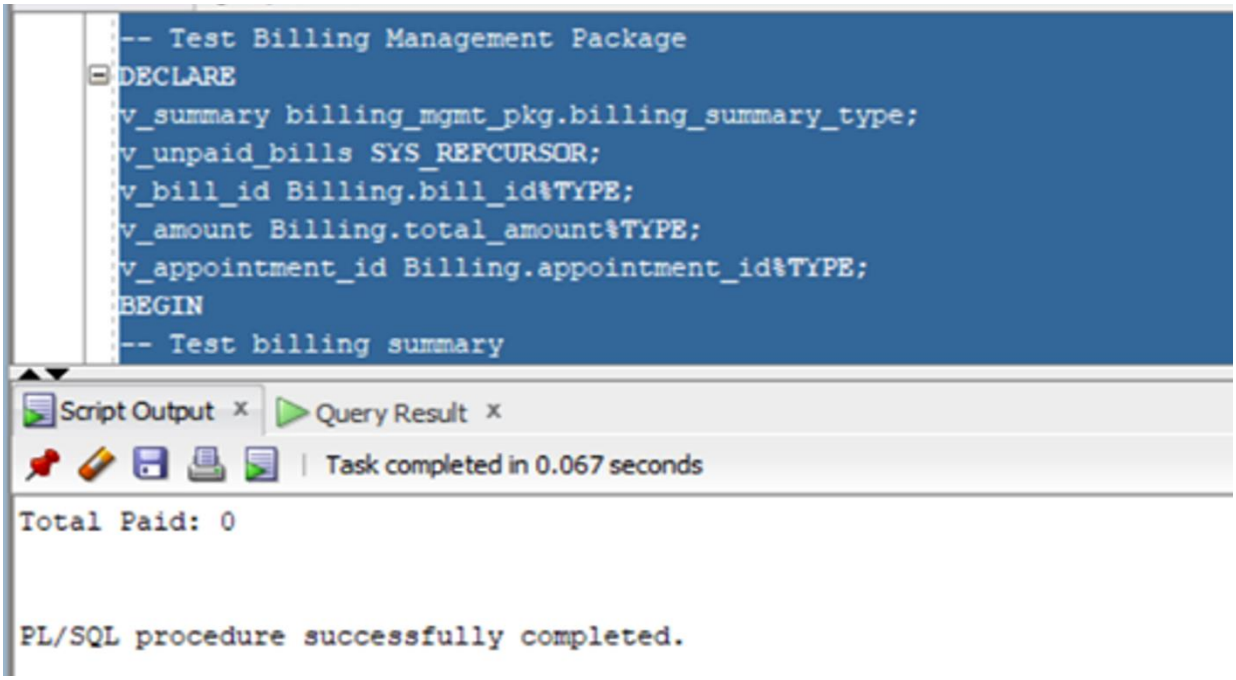
--Check
select * from Billing;



	BILL_ID	PATIENT_ID	APPOINTMENT_ID	TOTAL_AMOUNT	PAYMENT_STATUS
1	101	101	101	150	Paid
2	102	102	102	200	Unpaid
3	103	103	103	120	Paid
4	104	104	104	180	Paid
5	105	105	105	100	Refunded
6	106	106	106	250	Unpaid
7	107	107	107	175	Paid
8	108	108	108	220	Unpaid
9	109	109	109	160	Paid
10	110	110	110	140	No Show

-- Test Billing Management Package
DECLARE
v_summary billing_mgmt_pkg.billing_summary_type;
v_unpaid_bills SYS_REFCURSOR;
v_bill_id Billing.bill_id%TYPE;
v_amount Billing.total_amount%TYPE;
v_appointment_id Billing.appointment_id%TYPE;
BEGIN
-- Test billing summary
v_summary := billing_mgmt_pkg.get_billing_summary(102);
DBMS_OUTPUT.PUT_LINE('Total Paid: ' || v_summary.total_paid);
-- Test process payment
billing_mgmt_pkg.process_payment(102, 200.00);
-- Test unpaid bills with corrected fetch
v_unpaid_bills := billing_mgmt_pkg.get_unpaid_bills(101);
LOOP
FETCH v_unpaid_bills INTO v_bill_id, v_amount, v_appointment_id; -- Added appointment_id

```
EXIT WHEN v_unpaid_bills%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Unpaid Bill: ' || v_bill_id || ' Amount: ' || v_amount ||
' Appointment: ' || v_appointment_id);
END LOOP;
END;
```



Applications of Clinic Management System

1. **Patient Management Application**
   This core application handles comprehensive patient information management, from registration to medical history tracking. It maintains detailed patient profiles including demographics, contact information, and emergency contacts. The system enables easy access to patient records while ensuring data security through role-based access control, making it essential for healthcare providers to deliver personalized care.

2. **Appointment Scheduling System**
   The scheduling application manages the complete lifecycle of patient appointments, integrating doctor availability with patient preferences. It handles scheduling, rescheduling, and cancellations while automatically generating notifications for all parties involved. The system prevents scheduling conflicts and optimizes healthcare resource utilization through automated time slot management.

3. **Medical Records Management**
   This application maintains comprehensive electronic health records, including diagnoses, treatments, and medical history. It provides secure storage and retrieval of patient medical information, lab results, and treatment plans. Healthcare providers can access complete patient medical histories, enabling better-informed medical decisions and continuity of care.

4. **Billing and Insurance Management**
   The financial management application handles all aspects of healthcare billing, from service charges to insurance claims processing. It maintains payment records, generates bills automatically after appointments, and tracks payment statuses. The system also manages insurance policy information and coverage details, streamlining the billing process for both patients and healthcare providers.

5. **Laboratory Information System**
   This application manages all laboratory-related operations, from test ordering to result management. It tracks various types of medical tests, maintains result records, and links them to patient profiles. The system ensures efficient communication between laboratory staff and healthcare providers while maintaining accurate test records and results.

6. **Healthcare Staff Management**
   The staff management application handles healthcare provider information, including doctors' specializations, schedules, and department assignments. It manages staff availability, tracks service assignments, and maintains contact information. The system optimizes resource allocation and ensures efficient healthcare service delivery.
7. **Feedback and Quality Management**
   This application captures and manages patient feedback about their healthcare experiences. It enables healthcare facilities to track service quality, identify areas for improvement, and maintain service standards. The system helps in continuous service improvement through structured feedback collection and analysis.

8. **Room and Resource Management**
   The facility management application handles room allocation, tracks occupancy, and manages healthcare facility resources. It maintains real-time status of room availability, manages different room types, and ensures optimal resource utilization. The system helps in efficient facility management and patient accommodation planning.

Learning Outcomes

1. Database Design and Implementation

- Successfully implemented a complex healthcare database system with 19 interconnected tables
- Gained practical experience in designing and maintaining referential integrity through proper primary and foreign key relationships
- Learned the importance of proper table sequencing for database creation and data insertion

2. Business Process Understanding

- Developed deep insights into healthcare workflow management
- Understood the criticality of maintaining patient data privacy and security
- Learned to translate business requirements into technical specifications
- Gained knowledge about healthcare operations including appointment scheduling, billing, and medical record management

3. Technical Skills Enhancement

- Mastered Oracle SQL for database implementation
- Developed expertise in PL/SQL for creating packages, procedures, and triggers
- Learned to implement automated workflows through database triggers
- Gained experience in handling complex data relationships and constraints

4. System Integration Knowledge

- Understanding of how different modules interact in a healthcare system
- Knowledge of integrating various healthcare services (appointments, billing, lab results)
- Experience in maintaining data consistency across multiple related tables
- Insights into managing real-time data updates and notifications

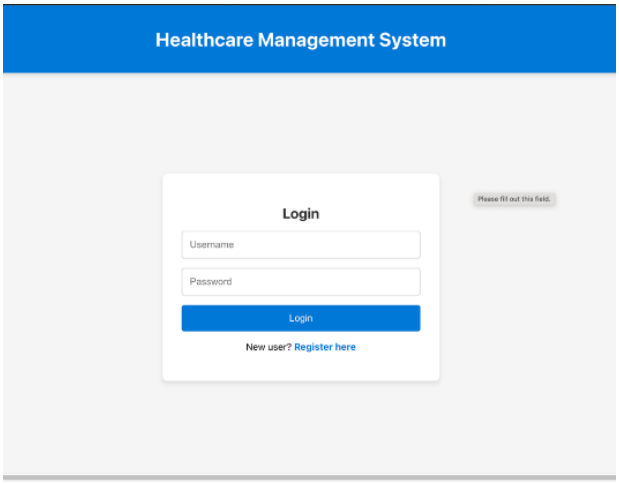
6. Project Management Insights

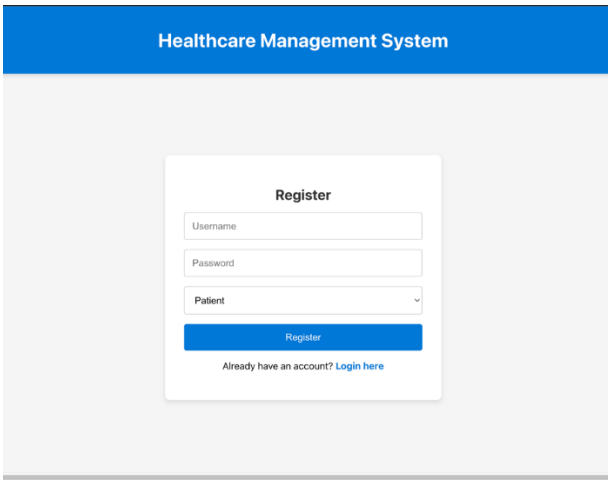
- Importance of proper requirement gathering and analysis
- Need for systematic approach in database design and implementation
- Value of proper documentation and code organization
- Understanding of scalability and maintenance requirements in healthcare systems

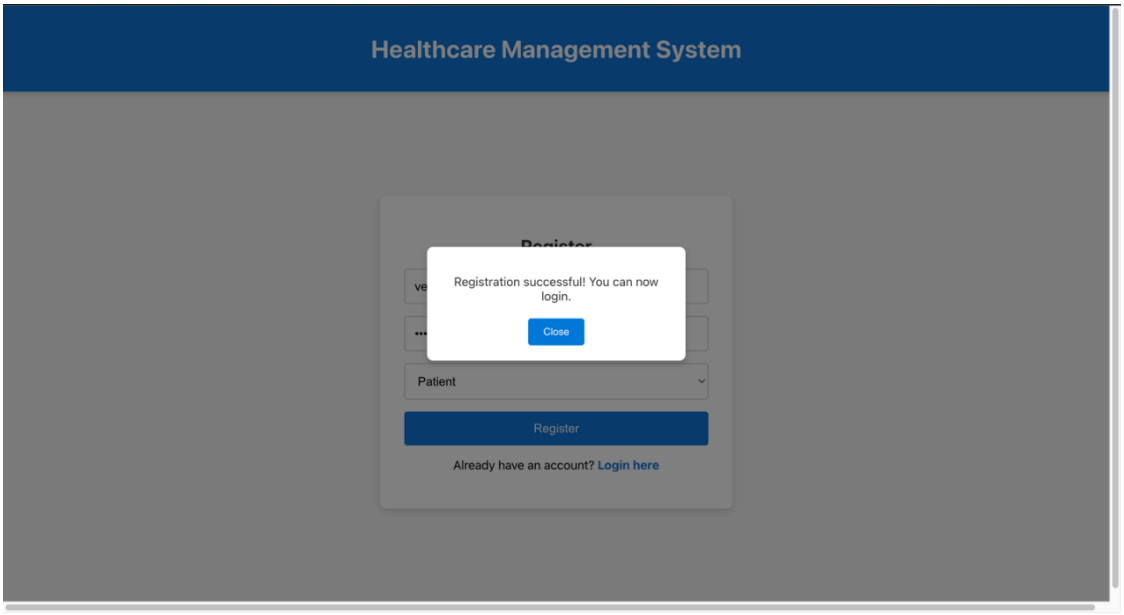
These learnings provide a strong foundation for future database development projects while highlighting the importance of combining technical expertise with domain knowledge in healthcare information systems.

Front end Design

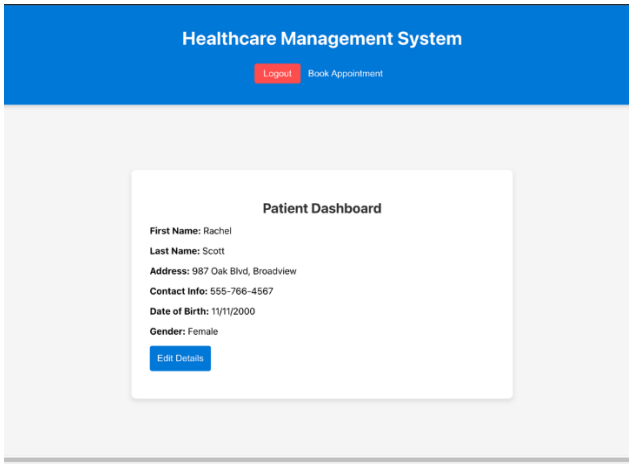
Login and registration page:

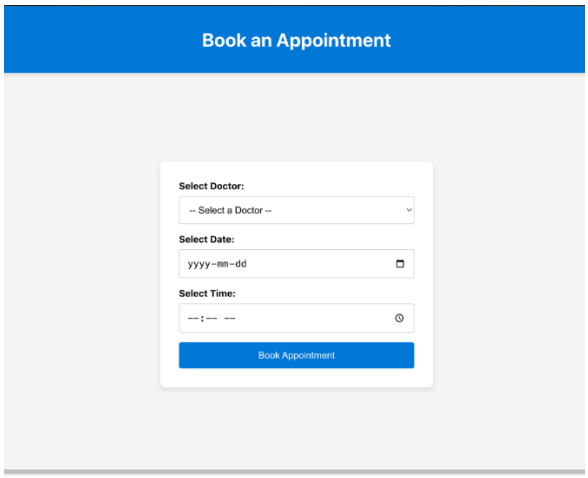


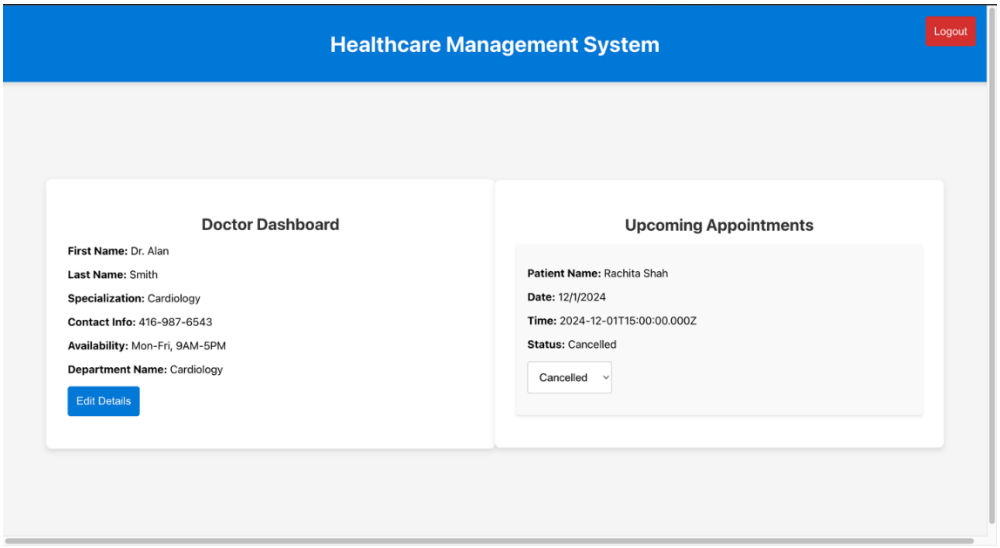


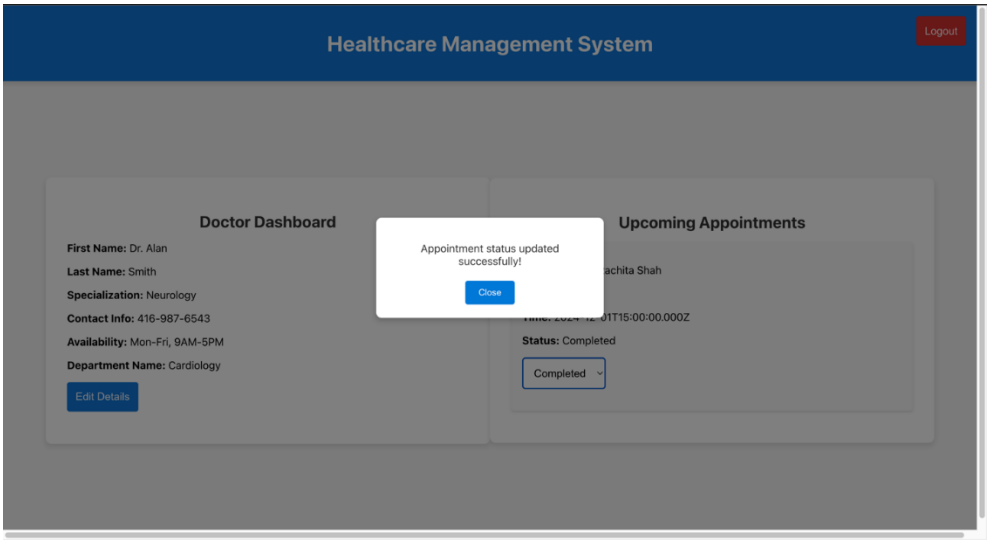


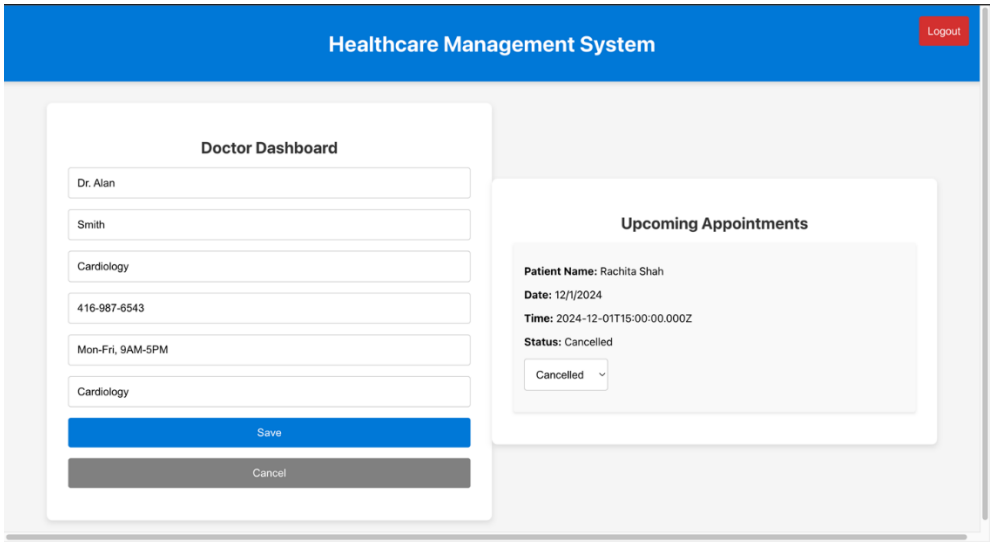
Doctor and Patient Dashboard:

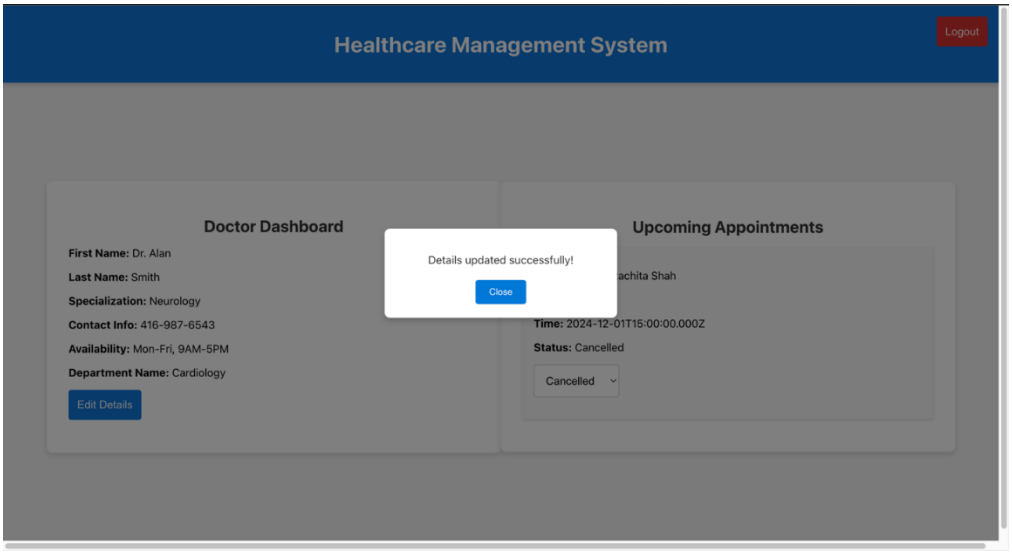









Healthcare Management System
Logout
Doctor Dashboard
Dr. Alan
Smith
Cardiology
416-987-6543
Mon-Fri, 9AM-5PM
Cardiology
Save
Cancel
Upcoming Appointments
Patient Name: Rachita Shah
Date: 12/1/2024
Time: 2024-12-01T15:00:00.000Z
Status: Cancelled
Cancelled


Healthcare Management System
Logout
Doctor Dashboard
First Name: Dr. Alan
Last Name: Smith
Specialization: Neurology
Contact Info: 416-987-6543
Availability: Mon-Fri, 9AM-5PM
Department Name: Cardiology
Edit Details
Details updated successfully!
Close
Upcoming Appointments
Time: 2024-12-01T15:00:00.000Z
Status: Cancelled
Cancelled

Conclusion

The Healthcare Patient Clinic Management System successfully demonstrates the implementation of a comprehensive database solution that effectively addresses the complex requirements of modern healthcare management. Through the development of 19 interconnected tables with robust relationships and constraints, the system provides a scalable and efficient platform for managing patient care, administrative tasks, and clinical operations.

The implementation showcases several key achievements:

- A secure and organized patient information management system
- Automated appointment scheduling and notification mechanisms
- Integrated billing and payment processing
- Comprehensive medical records management
- Laboratory results tracking and prescription management

The system's architecture, built on Oracle SQL with implemented PL/SQL packages, procedures, and triggers, ensures data integrity while maintaining high performance and security standards. The implementation of role-based access control and audit trailing mechanisms demonstrates the system's commitment to data privacy and regulatory compliance.

Through careful database design and normalization, the system successfully eliminates data redundancy while maintaining referential integrity across all healthcare operations. The implemented business rules and constraints ensure accurate data management and maintain operational consistency across various healthcare processes.

This project not only meets its initial objectives of streamlining healthcare operations but also provides a foundation for future enhancements and integrations. The modular design allows for easy scalability and adaptation to evolving healthcare needs, making it a valuable solution for modern healthcare facilities.

The successful implementation of this system demonstrates the effective application of database management principles in solving real-world healthcare challenges, ultimately contributing to improved patient care delivery and operational efficiency in healthcare settings.

Appendix:

Healthcare Clinic-DDL&DML

1.1