*Healthcare Management System*

**COMP 122 – Section 006 – Group 6**

Abdu Aimen

Riyadh Ahmed

Caglayan Arcan

Dawd Khalid

Lee Jin Hoi

Saifullah Md Khalid

Figure 1: Osborne, C. (2019, November 21). *The latest healthcare data breaches in 2019/2020*. The Daily Swig | Cybersecurity News and Views; The Daily Swig. <https://portswigger.net/daily-swig/the-latest-healthcare-data-breaches>

# Table of Contents

‌

[Table of Contents 1](#_gjdgxs)

[1. Introduction 2](#_30j0zll)

[2. Entity Relationships 3](#_1fob9te)

[3. Table Insertion 7](#_3znysh7)

[4. Table Creation 17](#_2et92p0)

[5. Queries 28](#_tyjcwt)

[6 – Conclusion, Challenges, References 34](#_3dy6vkm)

# 1. Introduction

1.1. Introduction

The Healthcare Management System database aims to streamline the management of healthcare-related information, including patient data, medical records, insurance details, and billing. This document serves as a comprehensive guide to understanding the database structure, table design, and data relationships.

1..2 Purpose of the Document

The purpose of this document is to provide developers, stakeholders, and system users with a clear understanding of the Healthcare Management System database. It explains the tables' purpose, the data they store, the relationships between tables, and sample queries to retrieve valuable information.

1..3 Scope and Objectives

The scope of this document encompasses the database design and structure for the Healthcare Management System. It includes the creation queries for each table and sample data to populate the tables for demonstration purposes. The document aims to:

* Provide an overview of the database schema and its components.
* Explain the purpose and usage of each table in the system.
* Describe the relationships between tables to highlight data dependencies.
* Offer sample data to illustrate how the tables store real-world information.
* Showcase sample queries to retrieve relevant information from the database.

# 2. Entity Relationships

2.1 Entity – Relationship Diagram (ERD)

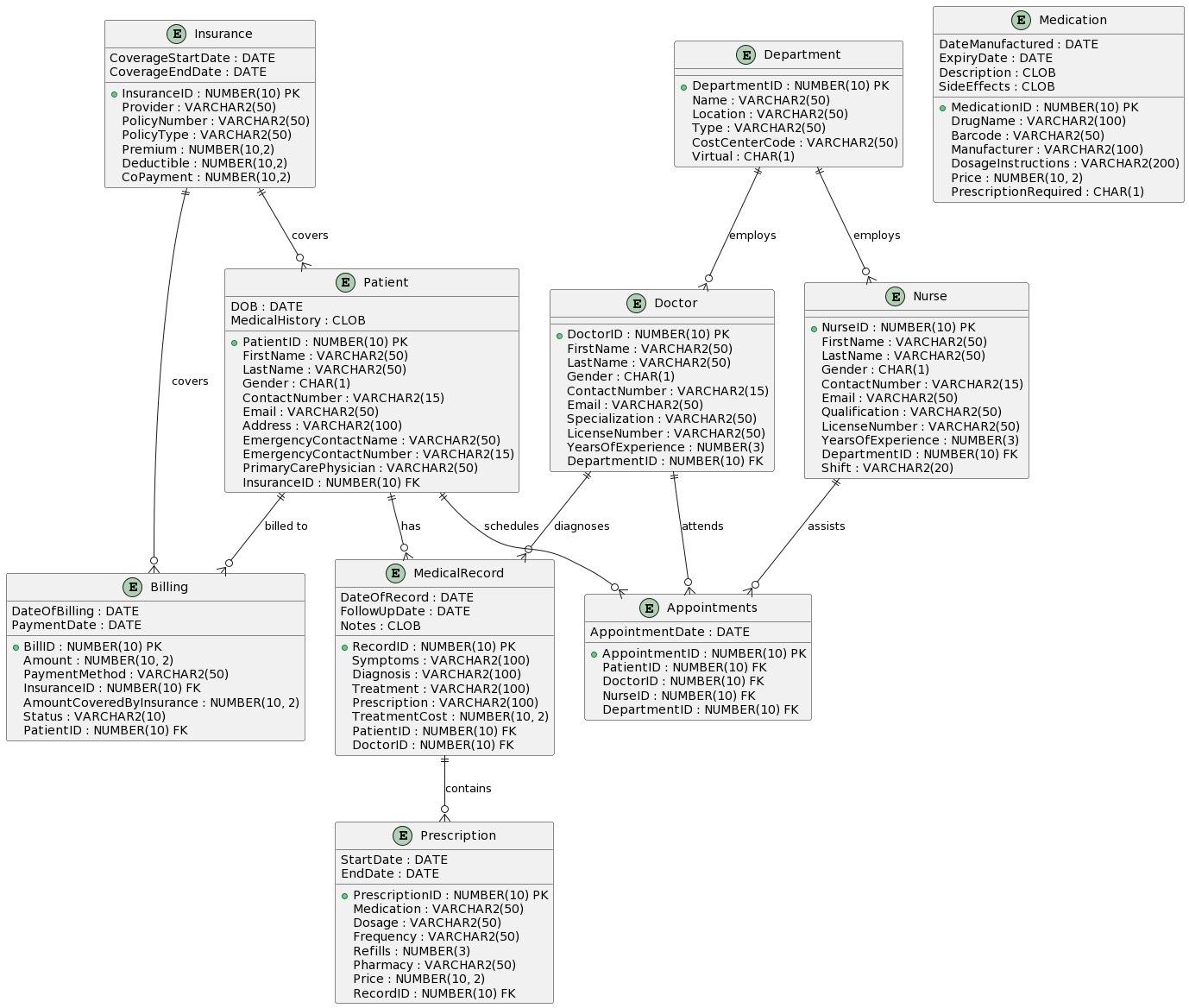


Figure 2.1.1 ERD Diagram of the Healthcare Management System

2.2 Entities

Insurance: Represents insurance policies that patients can have.

Department: different departments within the healthcare facility.

Patient: Represents individuals receiving medical care.

Doctor: Represents medical professionals providing diagnoses and treatments.

Nurse: Represents nursing staff assisting doctors and caring for patients.

MedicalRecord: Represents records of medical consultations, including symptoms, diagnosis, and treatment.

Prescription: Represents medication prescriptions given to patients.

Billing: Represents billing information related to medical services.

Appointments: Represents scheduled meetings between patients and medical staff.

Medication: Represents different medications that can be prescribed.

2.3 Relationships among Entities

Insurance - Patient:

Type: One-to-Many

Description: One insurance policy can cover multiple patients, but each patient has only one insurance policy. This is represented by the InsuranceID foreign key in the Patient table.

Department - Doctor:

Type: One-to-Many

Description: One department can employ multiple doctors, but each doctor belongs to only one department. This relationship is represented by the DepartmentID foreign key in the Doctor table.

Department - Nurse:

Type: One-to-Many

Description: One department can employ multiple nurses, but each nurse belongs to only one department. This is represented by the DepartmentID foreign key in the Nurse table.

Patient - MedicalRecord:

Type: One-to-Many

Description: One patient can have multiple medical records (from multiple visits or consultations), but each medical record pertains to only one patient. This relationship is represented by the PatientID foreign key in the MedicalRecord table.

Doctor - MedicalRecord:

Type: One-to-Many

Description: One doctor can diagnose and create multiple medical records for different patients, but each medical record belongs with only one doctor. This is represented by the DoctorID foreign key in the MedicalRecord table.

MedicalRecord - Prescription:

Type: One-to-Many

Description: One medical record can have multiple prescriptions (if multiple medications are prescribed during a consultation), but each prescription is linked to one medical record. Therefore this relationship is represented by the RecordID foreign key in the Prescription table.

Patient - Billing:

Type: One-to-Many

Description: One patient can have multiple bills (from multiple services or visits), although each bill is associated with only one patient. This is represented by the PatientID foreign key in the Billing table.

Insurance - Billing:

Type: One-to-Many

Description: One insurance policy can cover multiple bills, but each bill is associated with only one insurance policy. This relationship is represented by the InsuranceID foreign key in the Billing table.

Patient - Appointments:

Type: One-to-Many

Description: A patient can have multiple appointments, but each appointment is for one patient. This relationship is represented by the PatientID foreign key in the Appointments table.

Doctor - Appointments:

Type: One-to-Many

Description: A doctor can have multiple appointments with different patients, but each appointment is with one specific doctor. This is represented by the DoctorID foreign key in the Appointments table.

Nurse - Appointments:

Type: One-to-Many (optional)

Description: One nurse can assist in multiple appointments, but each appointment might be assisted by one specific nurse. This is represented by the NurseID foreign key in the Appointments table.

# 3. Table Insertion

3.1 Database Schema Description

The Healthcare Management System database consists of seven interconnected tables:

3.1.1 Department Table

The Department table stores information about different medical departments within the healthcare facility. It includes the following columns:

DepartmentID (Primary Key): Unique identifier

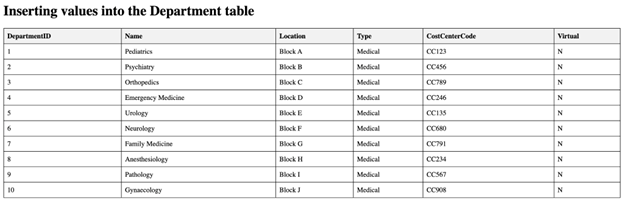
Name: name of the medical department.

Location: location of the department.

Type: The type of department (e.g., Medical, Surgical, etc.).

CostCenterCode: The cost center code associated with the department.

Virtual: A flag indicating whether the department is virtual (Y/N).



3.1.2 Insurance Table

The Insurance table stores information about various insurance policies provided by different insurance providers. It includes the following columns:

InsuranceID (Primary Key): Unique identifier(unique ID)

Provider: name of the insurance provider.

PolicyNumber: The policy number for the insurance .

PolicyType: The type of insurance policy (e.g., Health, Life, etc.).

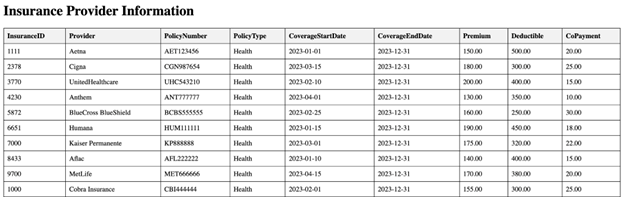
CoverageStartDate: insurance coverage start date

CoverageEndDate: insurance coverage end date

Premium: premium amount paid for the insurance policy.

Deductible: The deductible amount for the policy.

CoPayment: The copayment amount for medical services.



3.1.3 Medication Table

The Medication table stores data related to the medicines available in the healthcare facility. It includes the following columns:

MedicationID (Primary Key)\*: Unique identifier

DrugName\*: The name of the drug

Barcode\*: A unique barcode associated with the medication.

Manufacturer\*: The company or manufacturer that manufactures the medication.

DateManufactured\*: When the drugs were manufactured

ExpiryDate\*: date when the medication will expire.

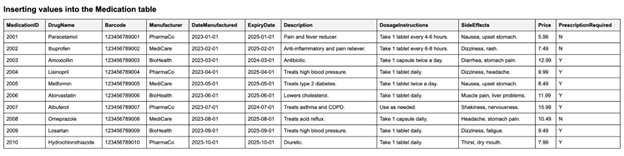
Description\*: detailed description of the medication.

DosageInstructions\*: Instructions on how the medication should be taken.

SideEffects\*: Information about any potential side effects of the medication.

Price\*: The price of the medication.

PrescriptionRequired\*: Indicates whether a prescription is required to obtain the medication (Y/N).



3.1.4 Nurse Table

The Nurse table stores data related to the nursing staff in the healthcare facility. It includes the following columns:

NurseID (Primary Key)\*: Unique identifier for each nurse in the healthcare system

FirstName: first name of the nurse.

LastName: last name of the nurse.

Gender\*: gender of the nurse (M/F).

ContactNumber: contact number of the nurse.

Email\*: The nurse’s email address.

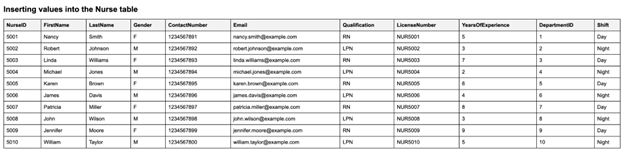
Qualification\*: The educational or professional qualification of the nurse.

LicenseNumber\*: The license number of the nurse.

YearsOfExperience\*: Years of exprience

DepartmentID\*: Foreign key referencing the Department table, representing the department to which the nurse belongs.

Shift\*: The shift timings or period during which the nurse works (e.g., Morning, Evening, Night).



3.1.5 Doctor Table

The Doctor table stores data related to healthcare providers (doctors) in the healthcare facility. It includes the following columns:

DoctorID (Primary Key): Unique identifier

FirstName: first name of the doctor.

LastName: The last name of the doctor.

Gender: gender of the doctor (M/F).

ContactNumber: The contact number of the doctor.

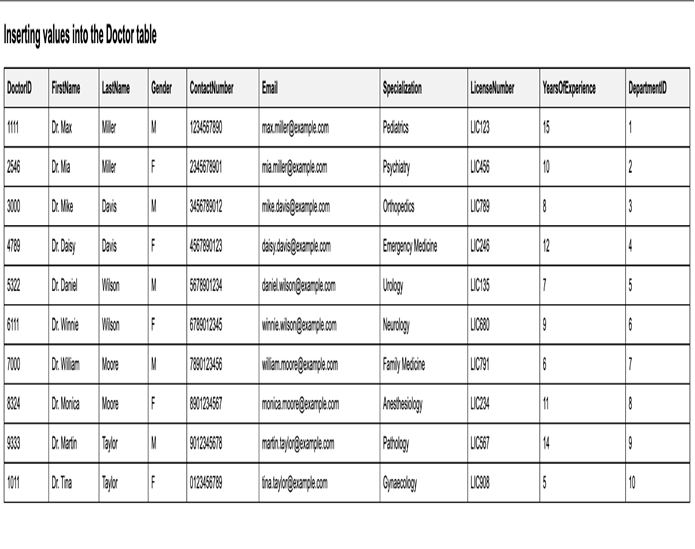
Email: The email address of the doctor.

Specialization: The medical specialization of the doctor (e.g., Cardiology, Pediatrics, etc.).

LicenseNumber: The license number of the doctor.

YearsOfExperience: The number of years of experience of the doctor.

DepartmentID: Foreign key referencing the Department table, representing the department to which the doctor belongs.



3.1.6 Patient Table

The Patient table stores data related to individual patients availing healthcare services. It includes the following columns:

PatientID (Primary Key): Unique identifier

FirstName: first name of the patient.

LastName: last name of the patient.

DOB: date of birth of the patient.

Gender: gender of the patient (M/F).

ContactNumber: The contact number of the patient.

Email: The email address of the patient.

Address: The address of the patient.

EmergencyContactName: The name of the patient's emergency contact person.

EmergencyContactNumber: The contact number of the patient's emergency contact person.

PrimaryCarePhysician: The name of the patient's primary care physician

MedicalHistory: A CLOB field to store the patient's medical history.

InsuranceID: Foreign key referencing the Insurance table, representing the patient's insurance coverage.



3.1.7 Medical Record Table

The MedicalRecord table stores medical records related to patient visits and consultations. It includes the following columns:

RecordID (Primary Key): Unique identifier, record id

PatientID: Foreign key referencing the Patient table, representing the patient associated with the medical record.

DoctorID: Foreign key referencing the Doctor table, representing the doctor associated with the medical record.

Symptoms: A brief description of the patient's symptoms.

Diagnosis: The doctor's diagnosis is based on the examination.

Treatment: The treatment plan prescribed by the doctor.

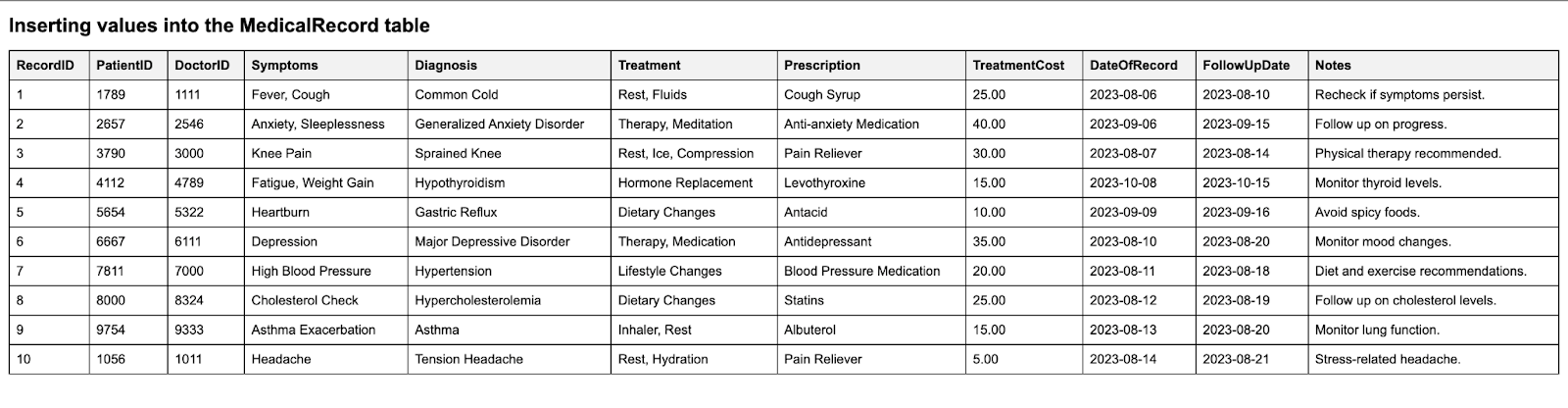
Prescription: Details of the medications prescribed to the patient.

TreatmentCost: The cost of the treatment provided.

DateOfRecord: The date of the medical record entry.

FollowUpDate: The date of the follow-up visit (if applicable).

Notes: Additional notes or comments regarding the medical record.



3.1.8 Prescription Table

The Prescription table stores data related to prescriptions issued to patients during medical visits. It includes the following columns:

PrescriptionID (Primary Key): Unique identifier for each prescription in format of prescriptionid.

RecordID: Foreign key referencing the MedicalRecord table, representing the medical record associated with the prescription.

Medication: Name of the medication that is prescribed to the patient.

Dosage: dosage of the medication

Frequency: frequency, how often the patient has to take the medication

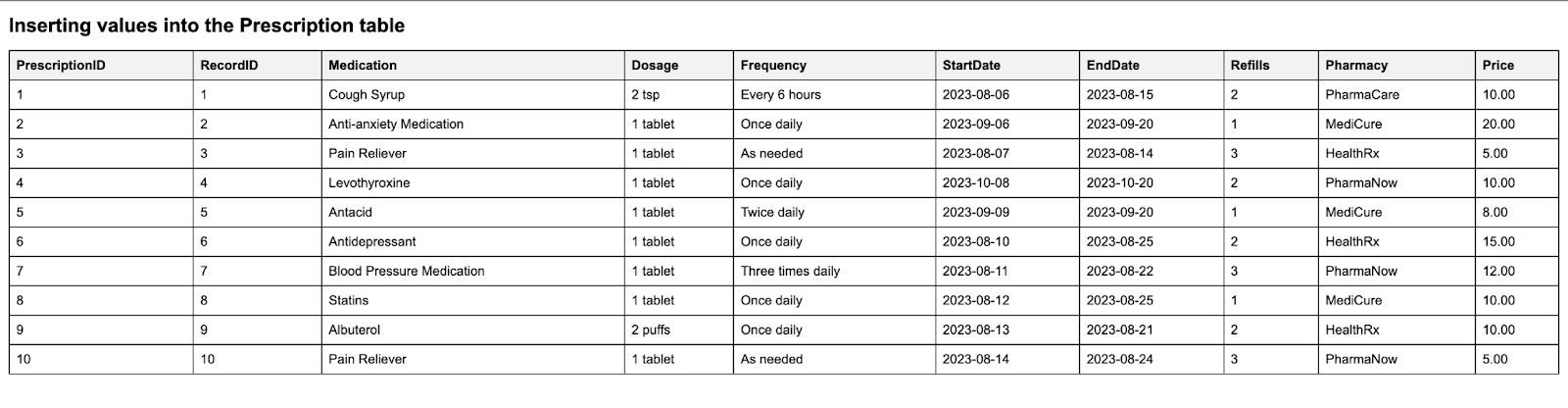
StartDate: Start date of the prescription.

EndDate: End date of the prescription.

Refills: number of refills permitted for the prescription.

Pharmacy: Which pharmacy the prescription can be filled at

Price: The price of the medication.



3.1.9 Billing Table

The Billing table stores data related to patient billing and payments. It includes the following columns:

BillID (Primary Key): Unique identifier for each billing record.

PatientID: Foreign key referencing the Patient table, representing the patient associated with the billing record.

Amount: total amount required to be paid for services.

DateOfBilling: The date of the billing record.

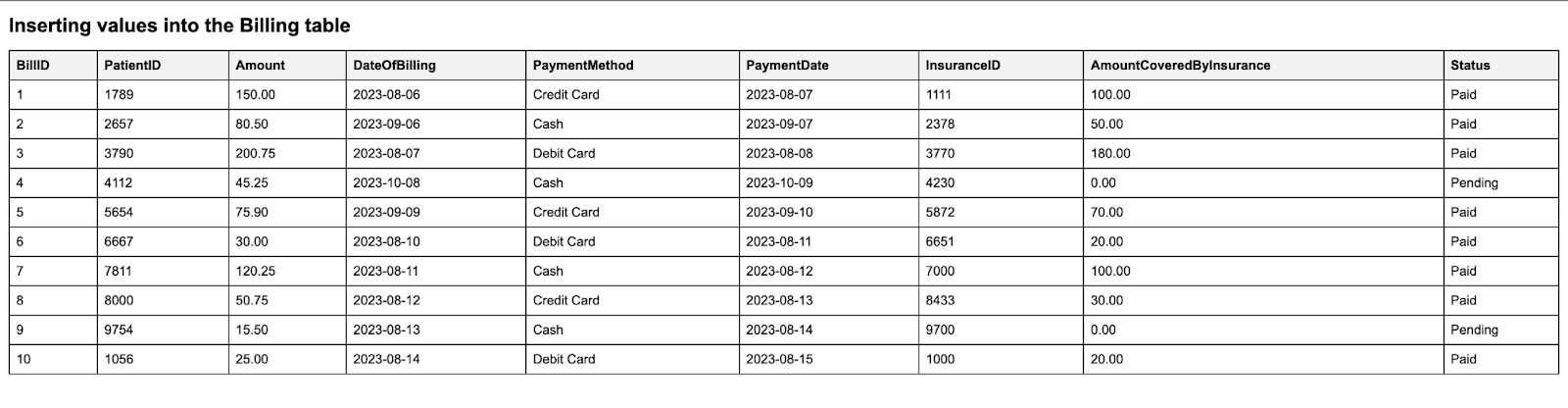
PaymentMethod: The method used for payment

PaymentDate: The date of payment.

InsuranceID: Foreign key referencing the Insurance table, representing the insurance associated with the billing record.

AmountCoveredByInsurance: The amount covered by insurance for the services.

Status: The status of the billing record (e.g., Paid, Pending, etc.).



3.1.10 Appointment Table

The Appointments table stores data related to the appointments scheduled in the healthcare facility. It includes the following columns:

AppointmentID (Primary Key) \*: Unique identifier for each appointment.

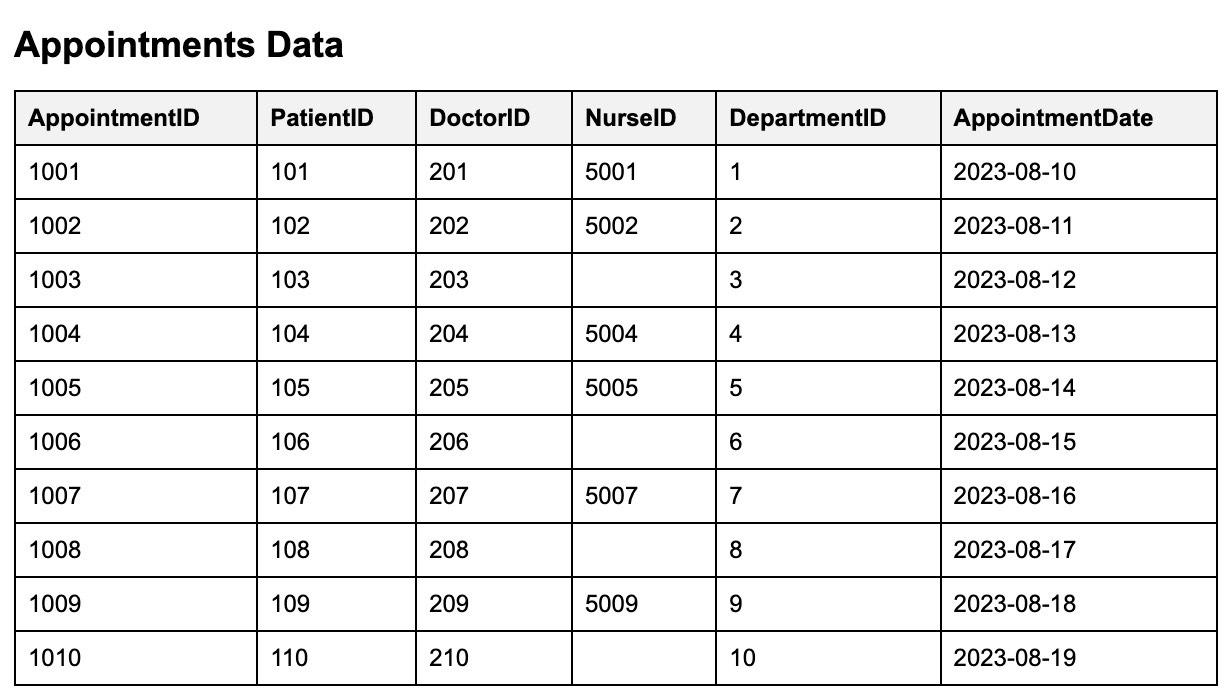
PatientID\*: Foreign key referencing the Patient table, representing the patient for whom the appointment is scheduled.

DoctorID\*: Foreign key referencing the Doctor table, representing the doctor with whom the appointment is scheduled.

NurseID\*: Optional foreign key referencing the Nurse table, representing the nurse assigned for the appointment, if any.

DepartmentID\*: Foreign key referencing the Department table, representing the department where the appointment will take place.

AppointmentDate\*: The date of the appointment.



# 4. Table Creation

4.1 - Department

CREATE TABLE Department (

DepartmentID NUMBER(10) PRIMARY KEY,

Name VARCHAR2(50),

Location VARCHAR2(50),

Type VARCHAR2(50),

CostCenterCode VARCHAR2(50),

Virtual CHAR(1) CHECK (Virtual IN ('Y', 'N'))

);

Department Table Explanation:

DepartmentID (Primary Key, NUMBER(10)): This unique identifier ensures each department has a distinct reference. Using a NUMBER data type allows for a large range of unique department IDs.

Name, Location, Type, CostCenterCode (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing department-related textual data.

Virtual (CHAR(1), CHECK (Virtual IN ('Y', 'N'))): CHAR is used for fixed-length character strings. The CHECK constraint ensures that only 'Y' or 'N' values are entered, indicating whether the department offers virtual consultations.

4.2 - Insurance

CREATE TABLE Insurance (

InsuranceID NUMBER(10) PRIMARY KEY,

Provider VARCHAR2(50),

PolicyNumber VARCHAR2(50),

PolicyType VARCHAR2(50),

CoverageStartDate DATE,

CoverageEndDate DATE,

Premium NUMBER(10,2),

Deductible NUMBER(10,2),

CoPayment NUMBER(10,2)

);

Insurance Table Explanation:

InsuranceID (Primary Key, NUMBER(10)): This unique identifier ensures each insurance entry is distinct. The NUMBER data type allows for a vast range of unique insurance entries.

Provider, PolicyNumber, PolicyType (VARCHAR2): VARCHAR2 is chosen for its flexibility in storing variable-length character strings, suitable for various insurance-related textual data.

CoverageStartDate, CoverageEndDate (DATE): The DATE data type ensures precision in storing the specific start and end dates of insurance coverage.

Premium, Deductible, CoPayment (NUMBER(10,2)): The NUMBER data type with two decimal places is used to accurately represent monetary values, ensuring precise billing and financial calculations.

4.3 - Medication

CREATE TABLE Medication (

MedicationID NUMBER(10) PRIMARY KEY,

DrugName VARCHAR2(100) NOT NULL,

Barcode VARCHAR2(50) UNIQUE NOT NULL,

Manufacturer VARCHAR2(100),

DateManufactured DATE,

ExpiryDate DATE,

Description CLOB,

DosageInstructions VARCHAR2(200),

SideEffects CLOB,

Price NUMBER(10, 2),

PrescriptionRequired CHAR(1) CHECK (PrescriptionRequired IN ('Y', 'N'))

);

Medication Table Explanation:

MedicationID (Primary Key, NUMBER(10)): This unique identifier ensures each medication entry is distinct. Using a NUMBER data type allows for a large range of unique medication entries.

DrugName, Barcode, Manufacturer, DosageInstructions (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing medication-related textual data.

DateManufactured, ExpiryDate (DATE): The DATE data type ensures precision in storing specific manufacturing and expiry dates of medications.

Description, SideEffects (CLOB): CLOB is used to store large text values, allowing for detailed descriptions and side effects of the medication.

Price (NUMBER(10, 2)): The NUMBER data type with two decimal places is used to accurately represent the price of the medication.

PrescriptionRequired (CHAR(1), CHECK (PrescriptionRequired IN ('Y', 'N'))): CHAR is used for fixed-length character strings. The CHECK constraint ensures that only 'Y' or 'N' values are entered, indicating whether a prescription is required.

4.4 - Nurse

CREATE TABLE Nurse (

NurseID NUMBER(10) PRIMARY KEY,

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

Gender CHAR(1),

ContactNumber VARCHAR2(15),

Email VARCHAR2(50),

Qualification VARCHAR2(50),

LicenseNumber VARCHAR2(50),

YearsOfExperience NUMBER(3),

DepartmentID NUMBER(10),

Shift VARCHAR2(20),

FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)

);

Nurse Table Explanation:

NurseID (Primary Key, NUMBER(10)): This unique identifier ensures each nurse has a distinct reference. Using a NUMBER data type accommodates a vast number of unique nurse entries.

FirstName, LastName, Email, Qualification, Shift (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing names, emails, and other textual data.

Gender (CHAR(1)): CHAR is used for fixed-length character strings. Since gender is represented by a single character (M/F), CHAR(1) is appropriate.

ContactNumber, LicenseNumber (VARCHAR2(15), VARCHAR2(50)): VARCHAR2 provides flexibility in storing variable-length phone numbers and license numbers.

YearsOfExperience (NUMBER(3)): The NUMBER data type allows for a range from 0 to 999, accommodating a wide range of experience years.

DepartmentID (NUMBER(10)): This foreign key establishes a relationship with the Department table, ensuring the referenced department exists.

4.5 - Doctor

CREATE TABLE Doctor (

DoctorID NUMBER(10) PRIMARY KEY,

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

Gender CHAR(1),

ContactNumber VARCHAR2(15),

Email VARCHAR2(50) NOT NULL,

Specialization VARCHAR2(50),

LicenseNumber VARCHAR2(50),

YearsOfExperience NUMBER(3),

DepartmentID NUMBER(10),

FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)

);

Doctor Table Explanation:

DoctorID (Primary Key, NUMBER(10)): This unique identifier ensures each doctor has a distinct reference. Using a NUMBER data type allows for a large range of unique doctor IDs.

FirstName, LastName, Email, Specialization (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing doctor-related textual data.

Gender (CHAR(1)): CHAR is used for fixed-length character strings. Since gender is represented by a single character (M/F), CHAR(1) is appropriate.

ContactNumber, LicenseNumber (VARCHAR2): VARCHAR2 provides flexibility in storing variable-length phone numbers and license numbers.

YearsOfExperience (NUMBER(3)): The NUMBER data type allows for a range from 0 to 999, accommodating a wide range of experience years.

DepartmentID (NUMBER(10)): This foreign key establishes a relationship with the Department table, ensuring the referenced department exists.

4.6 - Patient

CREATE TABLE Patient (

PatientID NUMBER(10) PRIMARY KEY,

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

DOB DATE,

Gender CHAR(1),

ContactNumber VARCHAR2(15),

Email VARCHAR2(50),

Address VARCHAR2(100),

EmergencyContactName VARCHAR2(50),

EmergencyContactNumber VARCHAR2(15),

PrimaryCarePhysician VARCHAR2(50),

MedicalHistory CLOB,

InsuranceID NUMBER(10),

FOREIGN KEY (InsuranceID) REFERENCES Insurance(InsuranceID)

);

Patient Table Explanation:

PatientID (Primary Key, NUMBER(10)): This unique identifier ensures each patient has a distinct reference. Using a NUMBER data type allows for a large range of unique patient IDs.

FirstName, LastName, Email, Address, EmergencyContactName, PrimaryCarePhysician (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing patient-related textual data.

DOB (DATE): The DATE data type ensures precision in storing the specific birth date of patients.

Gender (CHAR(1)): CHAR is used for fixed-length character strings. Since gender is represented by a single character (M/F), CHAR(1) is appropriate.

ContactNumber, EmergencyContactNumber (VARCHAR2(15)): VARCHAR2 provides flexibility in storing variable-length phone numbers.

MedicalHistory (CLOB): CLOB (Character Large Object) is used to store large text values, allowing for a detailed medical history.

InsuranceID (NUMBER(10)): This foreign key establishes a relationship with the Insurance table, ensuring the referenced insurance policy exists.

4.7 - Medical Record

CREATE TABLE MedicalRecord (

RecordID NUMBER(10) PRIMARY KEY,

PatientID NUMBER(10),

DoctorID NUMBER(10),

Symptoms VARCHAR2(100),

Diagnosis VARCHAR2(100),

Treatment VARCHAR2(100),

PrescriptionID NUMBER(10),

TreatmentCost NUMBER(10, 2),

DateOfRecord DATE,

FollowUpDate DATE,

Notes CLOB,

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),

FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)

);

MedicalRecord Table Explanation:

RecordID (Primary Key, NUMBER(10)): This unique identifier ensures each medical record has a distinct reference. Using a NUMBER data type allows for a large range of unique record IDs.

PatientID, DoctorID (NUMBER(10)): These foreign keys establish relationships with the Patient and Doctor tables, ensuring the referenced entities exist.

Symptoms, Diagnosis, Treatment (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing medical record-related textual data.

PrescriptionID (NUMBER(10)): This foreign key establishes a relationship with the Prescription table, ensuring the referenced prescription exists.

TreatmentCost (NUMBER(10, 2)): The NUMBER data type with two decimal places is used to accurately represent monetary values related to treatment.

DateOfRecord, FollowUpDate (DATE): The DATE data type ensures precision in storing specific dates related to the medical record.

Notes (CLOB): CLOB (Character Large Object) is used to store large text values, allowing for extensive notes about the medical record.

4.8 - Prescription

CREATE TABLE Prescription (

PrescriptionID NUMBER(10) PRIMARY KEY,

RecordID NUMBER(10),

MedicationID NUMBER(10),

Dosage VARCHAR2(50),

Frequency VARCHAR2(50),

StartDate DATE,

EndDate DATE,

Refills NUMBER(3),

Pharmacy VARCHAR2(50),

Price NUMBER(10, 2),

FOREIGN KEY (RecordID) REFERENCES MedicalRecord(RecordID),

FOREIGN KEY (MedicationID) REFERENCES Medication(MedicationID)

);

Prescription Table Explanation:

PrescriptionID (Primary Key, NUMBER(10)): This unique identifier ensures each prescription has a distinct reference. Using a NUMBER data type allows for a large range of unique prescription IDs.

RecordID, MedicationID (NUMBER(10)): These foreign keys establish relationships with the MedicalRecord and Medication tables, ensuring the referenced entities exist.

Dosage, Frequency, Pharmacy (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing prescription-related textual data.

StartDate, EndDate (DATE): The DATE data type ensures precision in storing specific start and end dates of the prescription.

Refills (NUMBER(3)): The NUMBER data type allows for a range from 0 to 999, accommodating a wide range of refill counts.

Price (NUMBER(10, 2)): The NUMBER data type with two decimal places is used to accurately represent the price of the prescription.

4.9 - Billing

CREATE TABLE Billing (

BillID NUMBER(10) PRIMARY KEY,

PatientID NUMBER(10),

Amount NUMBER(10, 2),

DateOfBilling DATE,

PaymentMethod VARCHAR2(50),

PaymentDate DATE,

InsuranceID NUMBER(10),

AmountCoveredByInsurance NUMBER(10, 2),

Status VARCHAR2(10),

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),

FOREIGN KEY (InsuranceID) REFERENCES Insurance(InsuranceID)

);

Billing Table Explanation:

BillID (Primary Key, NUMBER(10)): This unique identifier ensures each bill has a distinct reference. Using a NUMBER data type allows for a large range of unique bill IDs.

PatientID, InsuranceID (NUMBER(10)): These foreign keys establish relationships with the Patient and Insurance tables, ensuring the referenced entities exist.

Amount, AmountCoveredByInsurance (NUMBER(10, 2)): The NUMBER data type with two decimal places is used to accurately represent monetary values related to billing.

DateOfBilling, PaymentDate (DATE): The DATE data type ensures precision in storing specific billing and payment dates.

PaymentMethod, Status (VARCHAR2): VARCHAR2 is chosen for these fields to store variable-length character strings, providing flexibility in storing billing-related textual data.

4.10 - Appointment

CREATE TABLE Appointments (

AppointmentID NUMBER(10) PRIMARY KEY,

PatientID NUMBER(10),

DoctorID NUMBER(10),

NurseID NUMBER(10) NULL, -- Optional: In case a nurse is also assigned for the appointment

DepartmentID NUMBER(10),

AppointmentDate DATE,

);

Appointments Table Explanation:

AppointmentID (Primary Key, NUMBER(10)): This unique identifier ensures that each appointment has a distinct reference. Using a NUMBER data type allows for a large range of unique IDs, accommodating a vast number of appointments.

PatientID, DoctorID, NurseID, DepartmentID (NUMBER(10)): The NUMBER data type is used for these IDs to maintain consistency across tables and to allow for a large range of unique IDs. These foreign keys establish relationships with other tables, ensuring that the referenced entities (like patients or doctors) exist.

AppointmentDate (DATE): The DATE data type is chosen to store the specific day of the appointment, ensuring clarity and precision.

StartTime, EndTime (TIMESTAMP): TIMESTAMP is used to store both date and time, providing precision up to fractions of a second. This ensures accurate scheduling.

Purpose (VARCHAR2(200)): VARCHAR2 is used to store variable-length character strings, allowing for a detailed description of the appointment's purpose.

Status (VARCHAR2(50), DEFAULT 'Scheduled'): VARCHAR2 allows for variable-length status descriptions. The default constraint ensures that every new appointment is initially set to 'Scheduled'.

Notes (CLOB): CLOB (Character Large Object) is used to store large text values, allowing for extensive notes or comments about the appointment.

# 5. Queries

5.1 Complex Queries

5.1.1 - Retrieve Patients and Doctors with Their Latest Medical Record:

SELECT

p.PatientID,

p.FirstName || ' ' || p.LastName AS PatientName,

d.FirstName || ' ' || d.LastName AS DoctorName,

mr.Symptoms,

mr.Diagnosis,

mr.Treatment

FROM Patient p

JOIN (

SELECT

PatientID,

MAX(DateOfRecord) AS LatestRecordDate

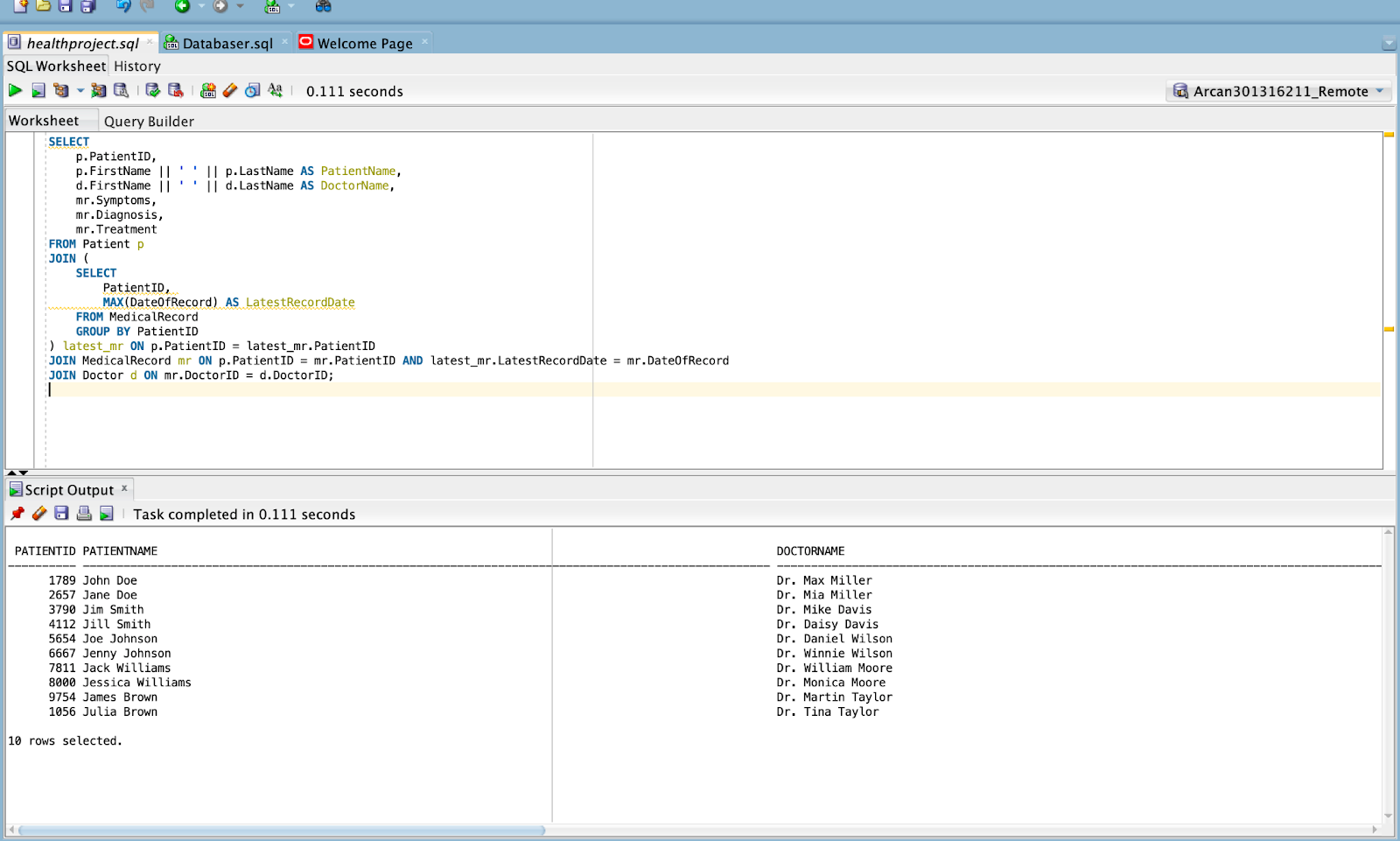
FROM MedicalRecord

GROUP BY PatientID

) latest\_mr ON p.PatientID = latest\_mr.PatientID

JOIN MedicalRecord mr ON p.PatientID = mr.PatientID AND latest\_mr.LatestRecordDate = mr.DateOfRecord

JOIN Doctor d ON mr.DoctorID = d.DoctorID;



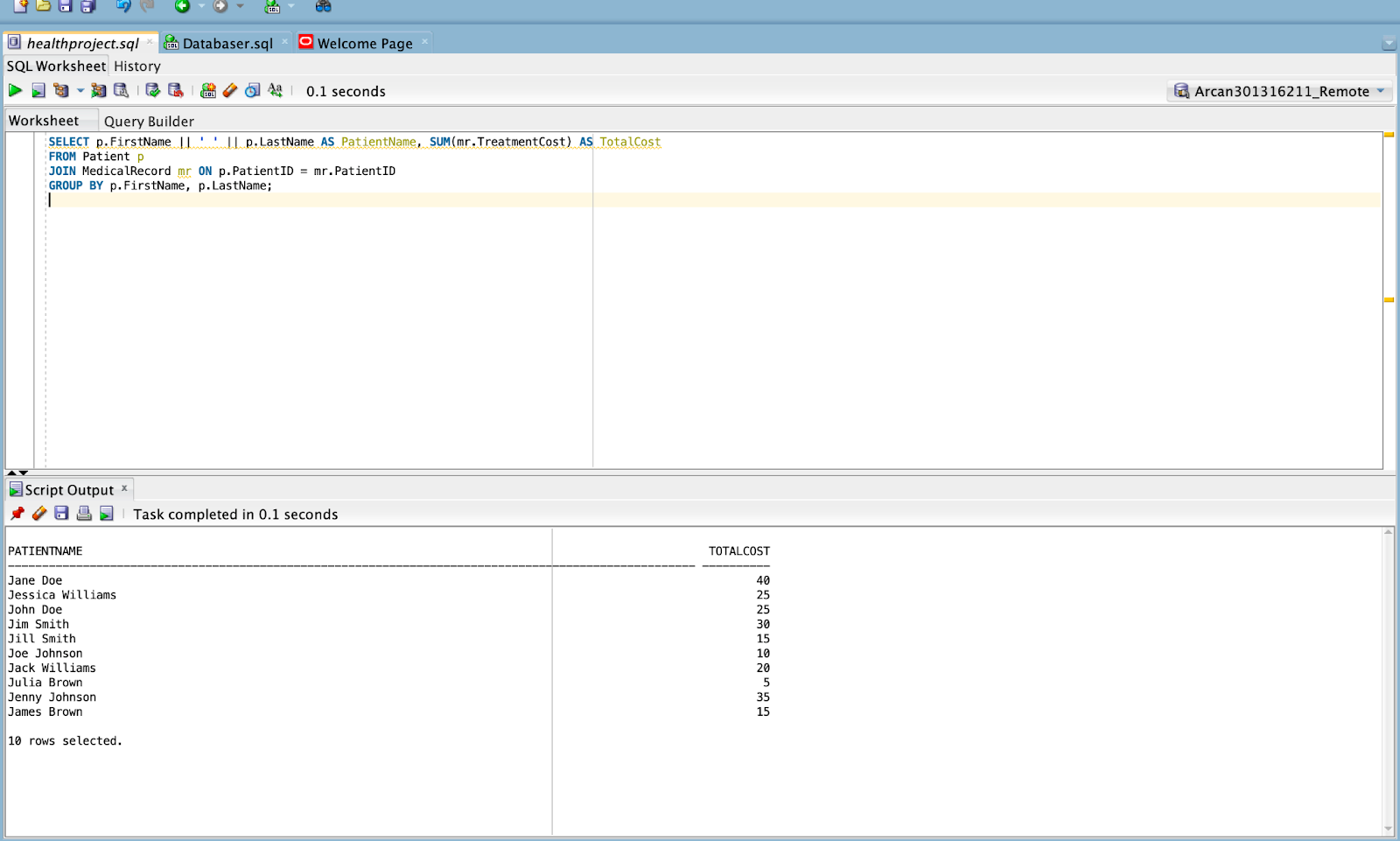
5.1.2 Calculate Total Treatment Cost for Each Patient:

SELECT p.FirstName || ' ' || p.LastName AS PatientName, SUM(mr.TreatmentCost) AS TotalCost

FROM Patient p

JOIN MedicalRecord mr ON p.PatientID = mr.PatientID

GROUP BY p.FirstName, p.LastName;

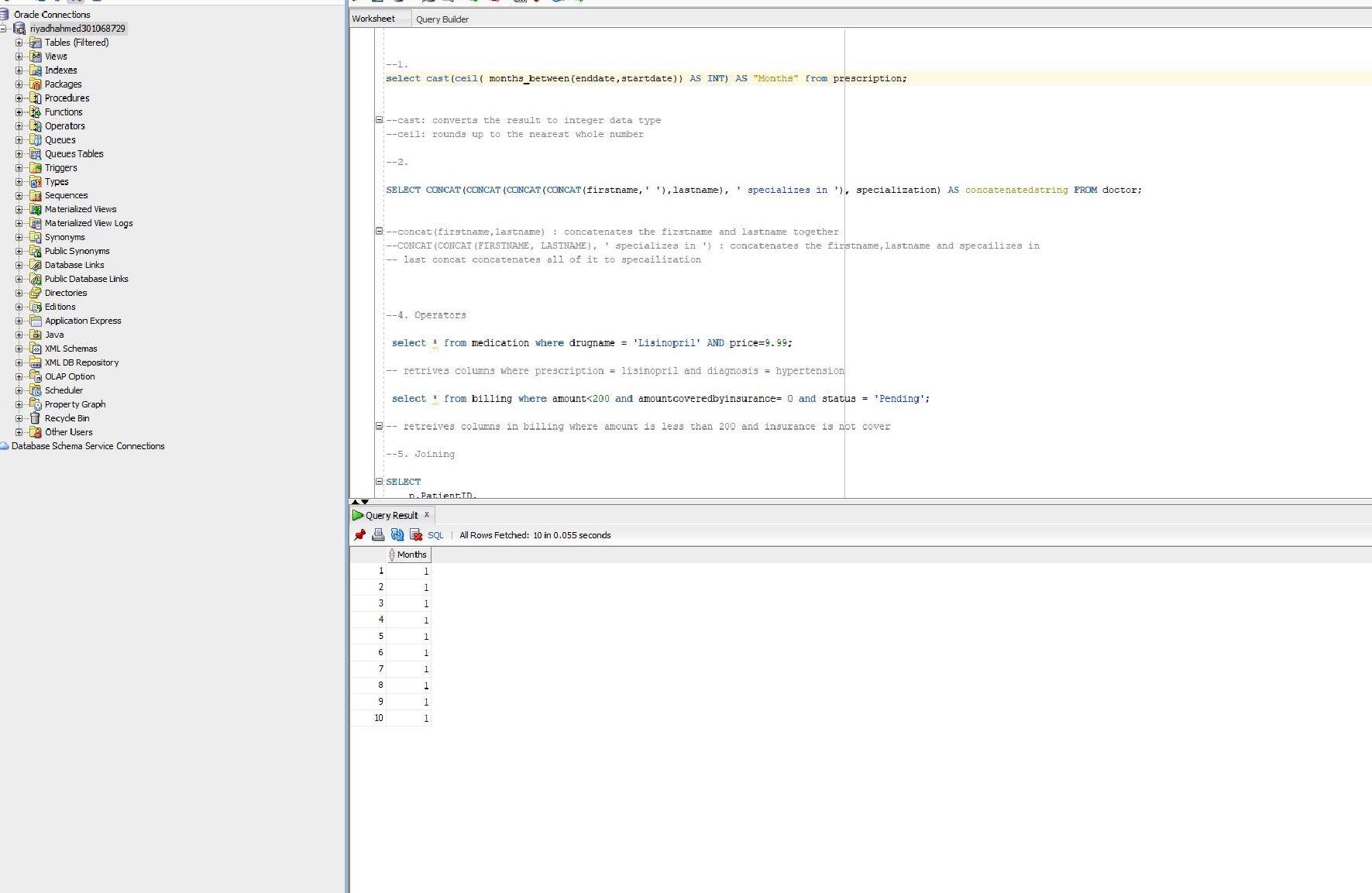


5.1.3 Calculate months between start date and end date of the prescription:

select cast(ceil( months\_between(enddate,startdate)) AS INT) AS "Months" from prescription;

--cast: converts the result to integer data type

--ceil: rounds up to the nearest whole number



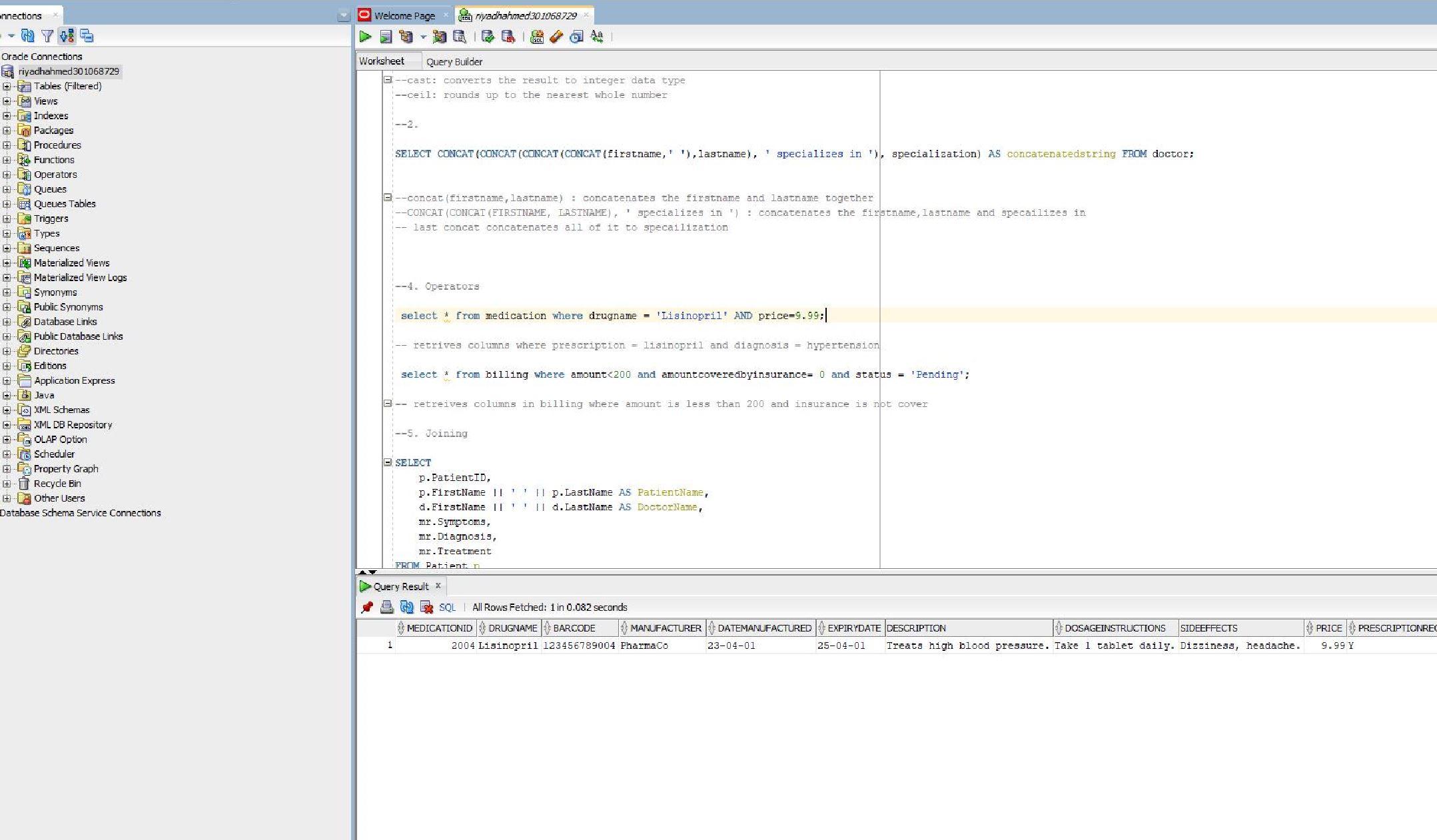
5.1.4 Concatenates first name and specializes in “specialization”

SELECT CONCAT(CONCAT(CONCAT(CONCAT(firstname,' '),lastname), ' specializes in '), specialization) AS concatenatedstring FROM doctor;

--concat(firstname,lastname) : concatenates the firstname and lastname together

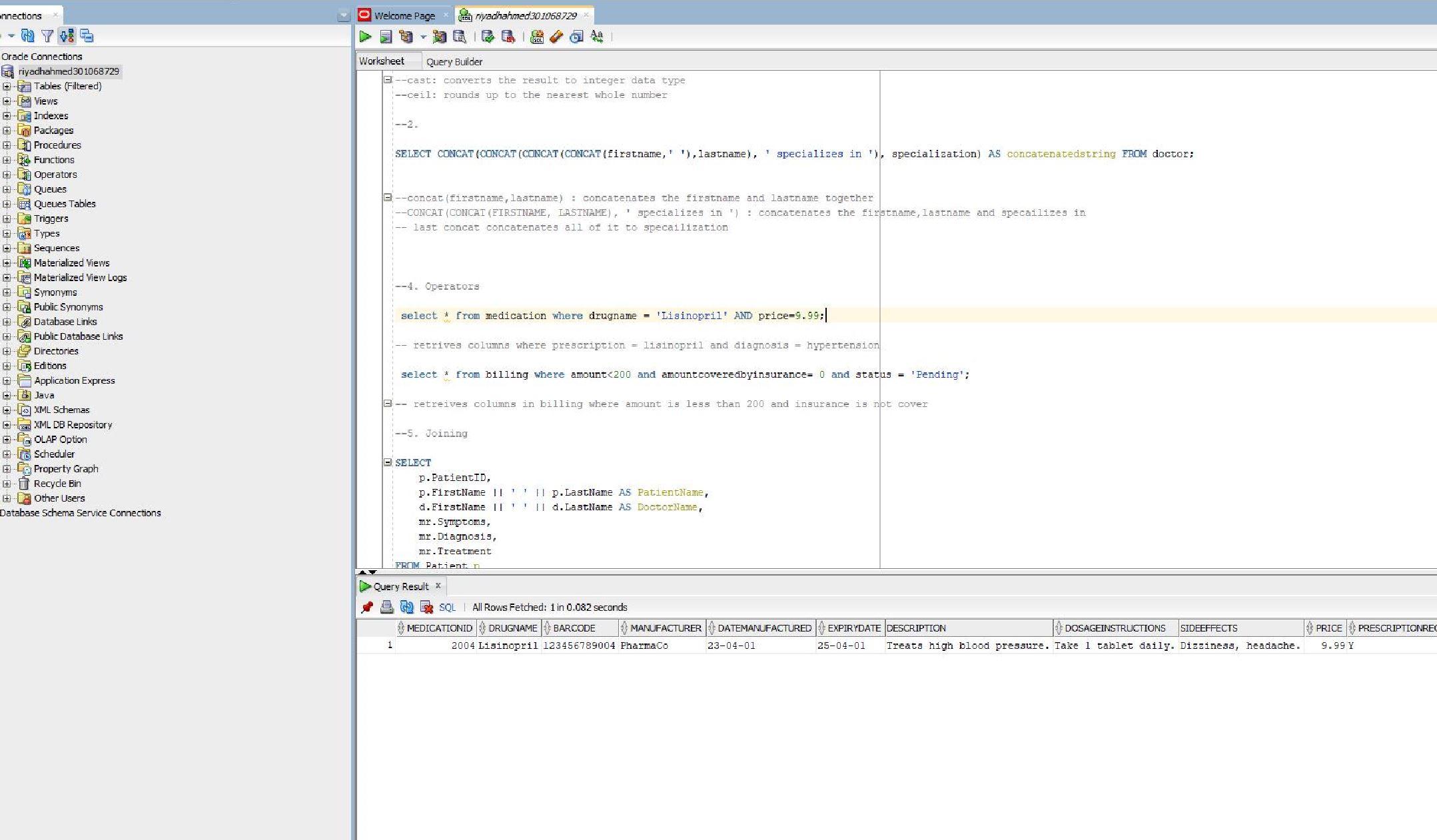
--CONCAT(CONCAT(FIRSTNAME, LASTNAME), ' specializes in ') : concatenates the firstname,lastname and specailizes in

-- last concat concatenates all of it to specailization



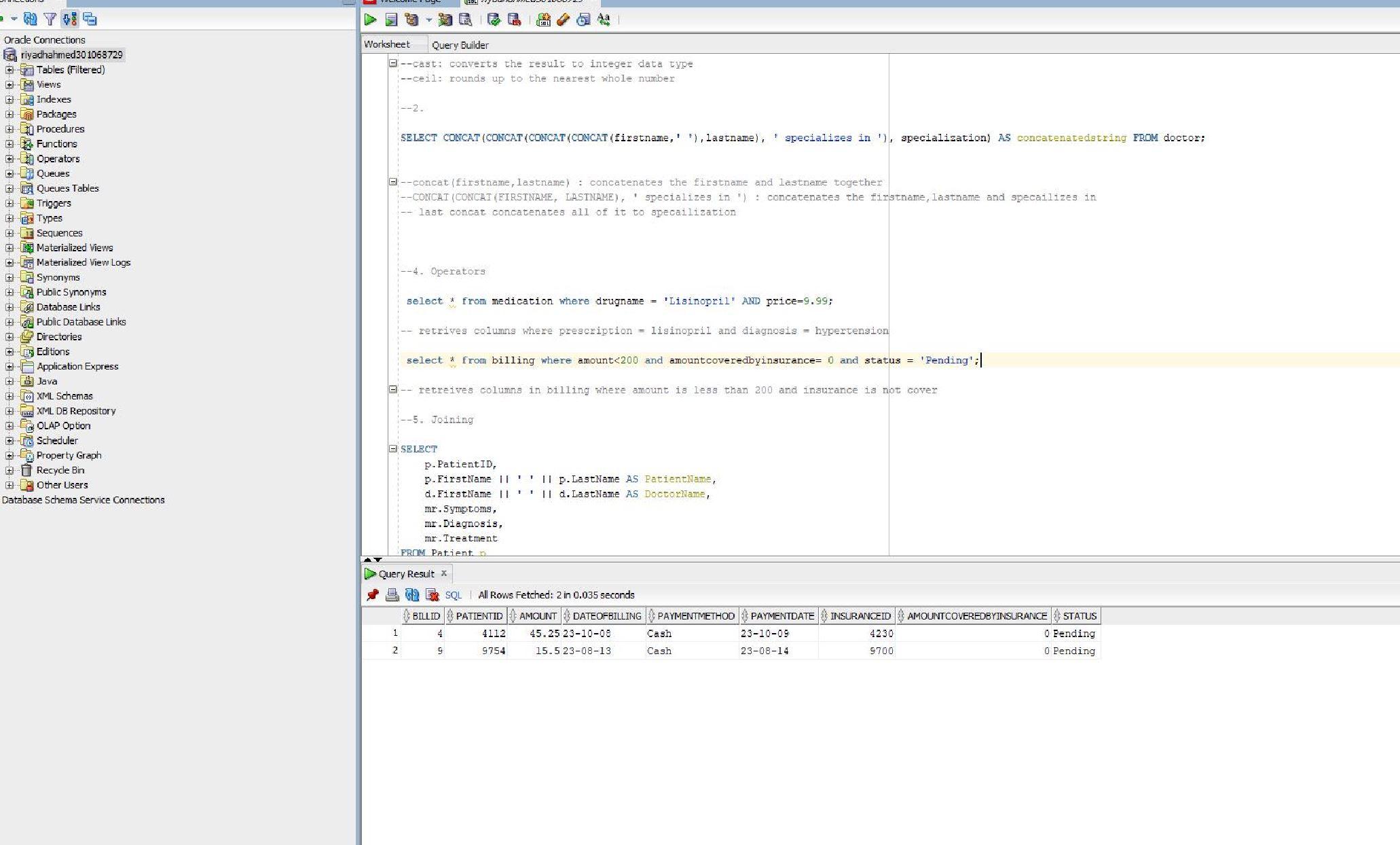
5.1.5 Finds record in medication where drugname = Lisinopril and price is 9.99

select \* from medication where drugname = 'Lisinopril' AND price=9.99;



5.1.6 Finds record from billing where amount is less than 200, amountcoveredbyinsurance = 0 and status = pending

select \* from billing where amount<200 and amountcoveredbyinsurance= 0 and status = 'Pending';



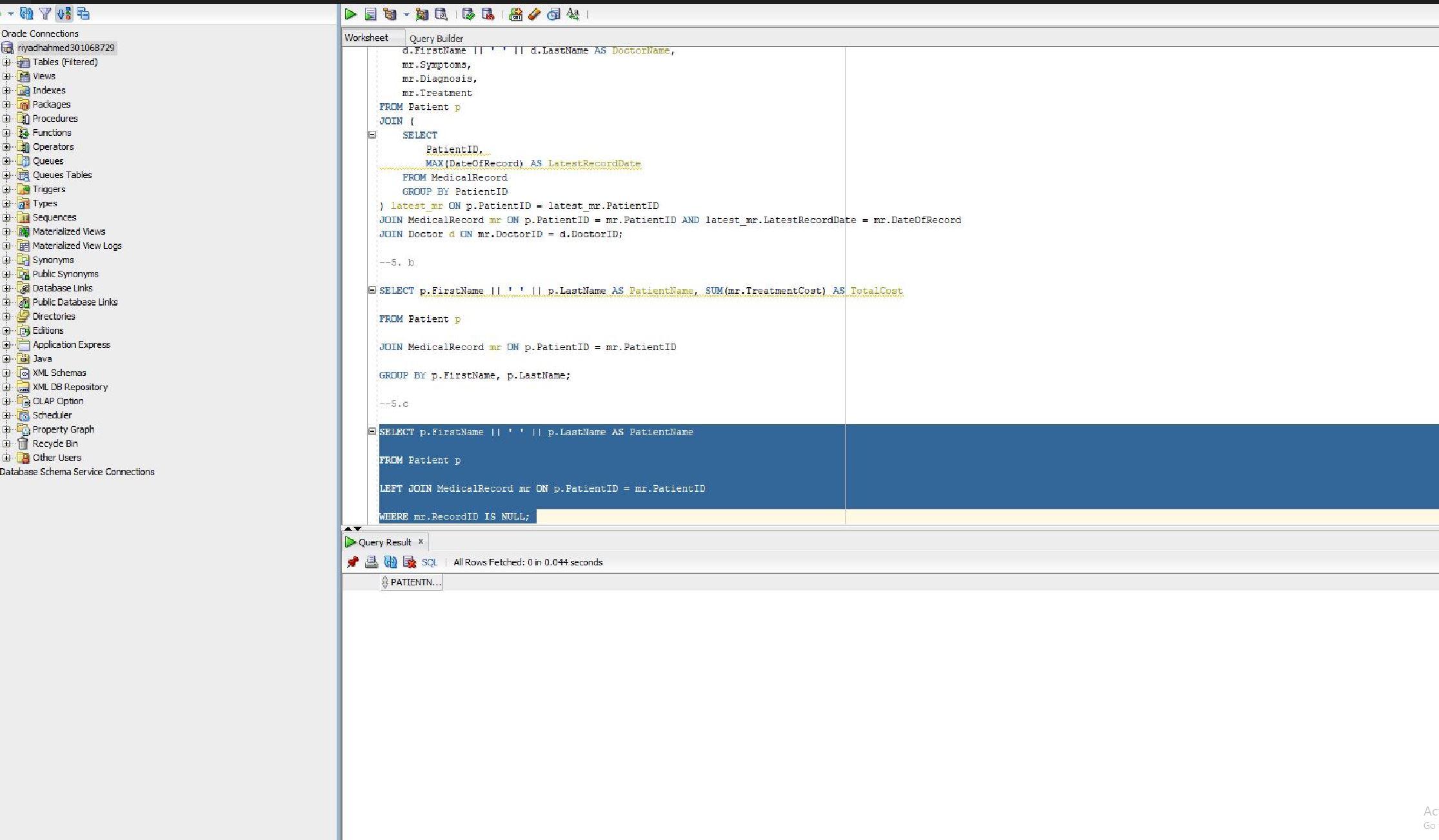
5.1.7 List Patients with No Medical Records:

SELECT p.FirstName || ' ' || p.LastName AS PatientName

FROM Patient p

LEFT JOIN MedicalRecord mr ON p.PatientID = mr.PatientID

WHERE mr.RecordID IS NULL;



# 6 – Conclusion, Challenges, References

6.1 - Conclusion

In conclusion, we were able to create a healthcare management system comprising of crucial records such as patient, doctor, prescription etc. We were able to demonstrate a clear understanding of the relationships between the tables and display various complex queries (single row functions, joins, operators) as illustrated in the document.

6.2 - Challenges

First building a database from scratch. -It was a challenging process for all group members to conjure up a more comprehensive database for the healthcare management system. We searched websites for inspiration such as Reddit, YouTube etc. Furthermore, it was challenging creating the tables, because each table has constraints and same attributes stored in different tables because of this, it was essential to create the classes in order, inserting values as well.

6.3 References

*Week 10 - 23M --Introduction To Database Conce (SEC. 006)*. (2023). Centennialcollege.ca. <https://e.centennialcollege.ca/d2l/le/content/964318/Home>

Osborne, C. (2019, November 21). *The latest healthcare data breaches in 2019/2020*. The Daily Swig | Cybersecurity News and Views; The Daily Swig. <https://portswigger.net/daily-swig/the-latest-healthcare-data-breaches>

National Center for Biotechnology Information. (n.d.). Health Databases and Health Database Organizations: Uses, Benefits, and Concerns. NCBI Bookshelf. Retrieved August 10, 2023, fromHealth Databases and Health Database Organizations: Uses, Benefits, and Concerns - Health Data in the Information Age - NCBI Bookshelf ([nih.gov](http://nih.gov/)) Stanford University. (n.d.). Databases in Healthcare. Stanford University. Retrieved August 10, 2023, from 5

Unknown ([stanford.edu](http://stanford.edu/))

Casteel, J. (2015). Oracle 12c: SQL (3rd ed.). Cengage Learning. ISBN-13: 978-1305251038. ISBN-10: 1305251032.

‌