**Database Design for a** **Children’s Hospital**

**Milestone Report 2**

**INFO 605: Fall 2024**

**December 12th , 2024**

**Drexel University**

Astrid Ellis, David Gorski, Josh Hoffman, Steven Sullivan

**TABLE OF CONTENTS**

Executive Summary 3

Project Statement 4-5

Overall Goals of the System 4

Content and Importance of the System 4

Scope of the Project 4

Related Systems and Any Open-Source Tools 5

Requirements Specifications 5-6

Data Requirements 5

Business Rules and Data Logic 5-6

Sample Output 6

Other Assumptions 6

The Software the Database was Designed in 6

How the Fake Data was Created 6

Conceptual Design 7

Conceptual Model 7

Explanations of the ERD 7-8

Relational Schema 9-12

Logical ERD 9

Relational Schema 10-11

Referential Integrity Diagram 12

Data Dictionary 12-16

Database Implementation 17

Create Table Commands 17

Data 17

Insert Into Commands 17

Data from Select All Queries 17

Data Queries, Update, and Delete Commands 17-35

Summary 35-36

**EXECUTIVE SUMMARY**

The following project report documents the design and development of a comprehensive database for a children’s hospital, modeling a potential subset of its operations that could be implemented in production. This database system is designed from the ground up to manage critical healthcare data effectively, enabling the hospital to leverage robust data collection practices. By centralizing information, the system provides hospital administrators and stakeholders with actionable insights to support business decision-making and operational efficiency. The database models various aspects of the hospital's operations, including medical insurance, charitable contributions, guardians, financial payments, patient data, medical history, equipment usage, visits, and residencies. These elements are vital to ensuring that hospital workflows are accurate and well-documented.

The project scope intentionally excludes several areas to maintain focus on essential operations. Out-of-scope elements include medications, physical spaces within the hospital, hospital staff, non-healthcare services such as parking and cafeteria management, and outpatient services. By narrowing the scope, the project aims to provide a precise, scalable framework for the most critical aspects of healthcare delivery within the hospital.

The development of the database followed a structured methodology, beginning with the creation of business rules and data logic to inform the conceptual design. A detailed Entity-Relationship Diagram (ERD) was produced to define the cardinality and participation of tracked entities. This conceptual design served as the foundation for building a logical ERD, a relational schema, and a referential integrity diagram, which established the tables and their interrelationships. These foundational designs ensured a clear, consistent, and comprehensive representation of the hospital’s data structure.

Subsequently, over five hundred lines of SQL commands were executed to create and populate the database's twenty-one tables with realistic sample data. This data was carefully constructed to mimic real-world hospital operations, maintaining accurate relationships between entities such as patients, guardians, payments, and equipment usage. Once populated, the database was tested using complex, business-driven queries designed to simulate practical use cases, including financial analysis, resource tracking, and patient management. Furthermore, update and delete commands were implemented to demonstrate the database's dynamic functionality, with examples showing the impact of data manipulations before and after execution.

The database was designed not only for operational accuracy but also for adaptability. Its structure ensures scalability to accommodate future needs, such as adding new hospital services or integrating with advanced analytics tools. The system also supports compliance with healthcare regulations like HIPAA, ensuring that patient confidentiality and data security remain priorities. These features position the database as a reliable and efficient solution for modern hospital operations.

This report demonstrates the potential of the proposed database to enhance hospital operations through robust data management and insightful reporting capabilities. By integrating well-defined business rules, scalable data architecture, and realistic testing scenarios, the project lays a strong foundation for future implementation and expansion.

**(1) PROJECT STATEMENT**

**1.1 Overall Goals of the System**

The goal of this project is to develop a comprehensive database management system for a children’s hospital. The system aims to streamline critical hospital operations by managing patient appointments, medical records, financial transactions, equipment usage, and healthcare-related information. By centralizing data, the system enhances efficiency, ensures accuracy, and provides medical staff with quick access to essential data. This enables the hospital to improve patient care, reduce administrative burden, and support informed decision-making for hospital management and stakeholders.

**1.2 Context and Importance of the System**

A children’s hospital requires a reliable database system to manage complex healthcare-related data efficiently. Paper-based or isolated systems are prone to errors, delays, and inefficiencies, which can compromise patient care. A centralized system ensures streamlined workflows, accurate recordkeeping, and enhanced data accessibility for staff. Furthermore, the database adheres to healthcare regulations such as HIPAA to ensure patient confidentiality and data security, supporting the hospital's mission of delivering high-quality, secure, and patient-centric care.

**1.3 Scope of the Project**

**1.3.1 IN-Scope**

The following features are included within the scope of this project:

* **Patient Management**: Recording patient details, including personal information, medical history, and allergies.
* **Appointment Scheduling**: Tracking patient appointments, doctor schedules, and future visits.
* **Billing**: Managing payments, insurance claims, and invoices for services provided.
* **Equipment Management**: Tracking hospital equipment usage, maintenance schedules, and vendor information.
* **Medical History**: Recording details of patient allergies, chronic conditions, and past surgeries.

**1.3.2 Out-Scope**

The following features are excluded from the scope of this project:

* **Non-Healthcare Services**: Management of services such as cafeteria, parking, or laundry.
* **Outpatient Services**: Only in-hospital services are addressed in this system.
* **Pharmaceutical Management**: The system does not include medication inventory or dispensing records.
* **Provider information:** This project does not include records pertaining to medical providers at the hospital location

**1.4 Related systems and any open-source tools**

The project leverages Oracle SQL Developer for database design and implementation. Additionally, tools such as Draw.io were used for conceptual modeling, and SQL scripts were created to generate realistic sample data. The system is designed to be interoperable with existing hospital management systems and electronic health record (EHR) platforms, ensuring seamless integration and adherence to healthcare standards.

**(2) REQUIREMENTS SPECIFICATION**

**2.1 Data Requirements**

The database system must support a wide array of data requirements for managing hospital operations. Key data requirements include:

* **Patient Information**: Tracking details such as name, date of birth, social security number, medical history, allergies, and past surgeries.
* **Residence Information**: Recording patient addresses, including street, city, state, zip code, and residence type.
* **Guardian Details**: Maintaining guardian information, such as contact details, relationship to the patient, and hours of availability.
* **Visit Information**: Capturing details of patient visits, including the date, chief complaint, total cost, and services used.
* **Service Details**: Storing a catalog of medical services, including service description, cost, and associated medical codes.
* **Equipment Management**: Logging information about hospital equipment, such as purchase date, warranty expiration, last serviced date, and vendor details.
* **Insurance Policies**: Recording insurance details, including policyholder information, policy expiration date, and insurer contact details.
* **Payments and Financial Transactions**: Tracking payment amounts, payment methods, and timestamps for each transaction.
* **Charitable Contributions**: Logging contributions from charitable organizations to patient payments.

**2.2 Business Rules and Data Logic**

Key business rules govern the relationships and logic within the database:

* **Patient and Residence**: Each patient must have a linked residence, and multiple patients can share the same residence.
* **Patient and Guardian**: A patient must have at least one guardian, and a guardian can manage multiple patients.
* **Patient and Visit**: Each patient can have zero or many visits, and each visit must be associated with one patient.
* **Visit and Services**: Each patient visit may include zero or many services, with service details recorded in the system.
* **Equipment Usage**: Equipment is tracked for each patient visit to monitor its usage and maintenance status.
* **Insurance and Payments**: Payments can be made by the patient, a guardian, or a charitable organization. Insurance policies are linked to a single patient’s guardian.
* **Charitable Contributions**: Charitable organizations may provide funds to assist with patient bills, and these payments must be logged and traceable.
* **Referential Integrity**: All foreign keys are enforced to maintain database integrity.

**2.3 Sample Output**

Key screens and reports leveraging our database structure:

* A list of patients with upcoming appointments and their scheduled services.
* Reports on equipment usage and maintenance schedules.
* Summaries of financial contributions by charitable organizations.
* Screens displaying outstanding payments and linked patients.
* Detailed patient summaries, including medical history and visit details.
* A report of the top services provided, based on usage and revenue.

**2.4 Other Assumptions**

* **Scope Limitation:** The database only models healthcare-related operations, excluding non-medical services such as cafeteria or parking management.
* **Data Granularity:** The focus is on high-level operations, including only the essential details for patients, guardians, and medical resources.

**2.5 The Software the Database was Designed in**

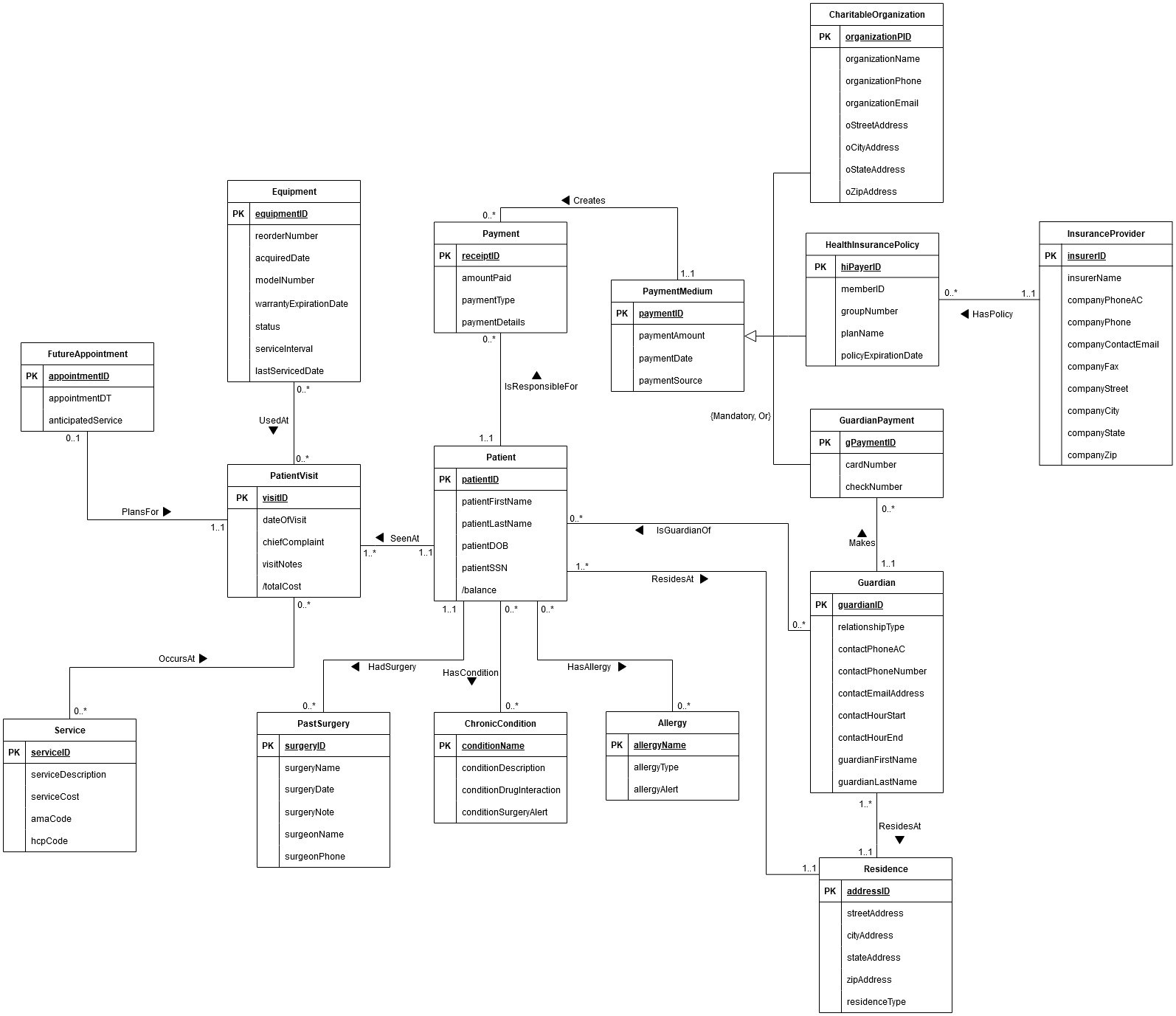
The database was designed in Oracle’s LiveSQL environment, an open-source relational database management system (RDBMS). LiveSQL was chosen for its robustness, compatibility with various operating systems, and extensive support for SQL commands. The conceptual design was initially created using draw.io, a visual tool for diagramming, which helped in developing the ERD. This ERD was then converted into the logical and relational schemas to define the structure of the database. The SQL scripts for creating tables, populating data, and running queries were developed and tested using the LiveSQL Workbench. LiveSQL’s support for foreign keys and referential integrity made it particularly suitable for modeling the hospital's complex relationships and constraints.

**2.6 How the Fake Data was Created**

Fake data was created to populate all 21 tables of our database. The goal was to create a substantial amount of data to enable meaningful queries that could simulate real world situations. The data was created using a random string generator, which then populated an Excel document. From there, Excel formulas were used to convert the data into insert commands to prepare it for database entry. A critical element of this process was ensuring the data could tell a logical story of the database usage. For example, even though random numbers were used, a patient should not have an appointment prior to their date of birth. While not actual constraints on the data from the database, these acted as logical constraints in the story of our data.

**(3) CONCEPTUAL DESIGN**

**3.1 Conceptual model**



**3.2 Explanations on the ERD**

The Entity-Relationship Diagram (ERD) captures the logical structure of the database for a children’s hospital, organizing it into entities, attributes, relationships, and cardinalities. The core entity is Patient, which stores essential patient details, including a unique identifier, name, date of birth, and social security number. Each patient resides at an address represented by the Residence entity, which holds attributes like street address, city, state, zip code, and type of residence. This forms a one-to-many relationship where multiple patients can share a single address.

The PatientVisit entity logs all visits a patient makes to the healthcare facility, storing attributes such as the visit ID, date of visit, chief complaint, and notes. Each visit is uniquely identified and linked to one patient (one-to-many relationship). A visit can involve zero to multiple services, represented by the ServiceAtVisit entity. The ServiceListing entity defines available services with attributes such as service ID, description, cost, and associated medical codes (AMA and HCP). Although each patient may have 0 to many services, there will be 0 to 1 corresponding service listings.

Each patient can have zero or multiple PatientCondition and PatientAllergy entries, reflecting their diagnosed conditions and known allergies. These entities map patients to the ChronicCondition and Allergy entities, which store standardized details about medical conditions and allergy types, including attributes such as severity, drug interactions, and reaction notes. The relationships ensure detailed tracking of a patient’s health data.

The PastSurgery entity records a patient’s surgical history, including details such as the surgery date, surgeon name, and follow-up notes. Similarly, the FutureAppointment entity schedules anticipated services for patients, linking appointments to specific patient visits. Each patient may have 0 or many past surgeries. Each patient can schedule 0 or 1 future appointments at a time.

Guardianship is managed through the Guardian entity, which holds information about individuals responsible for patients, such as contact details, relationship type, and availability hours. The Guardianship associative entity connects guardians to patients in a one-to-many relationship.

Financial data is managed through entities like HealthInsurancePolicy, GuardianPayment, and CharitableOrganization. The Payment entity is tied to all 3 of these entities and tracks payments made by or on behalf of patients, including attributes such as amount, type, and payment details. It connects to the PaymentMedium entity, which categorizes the source of payment as mentioned.

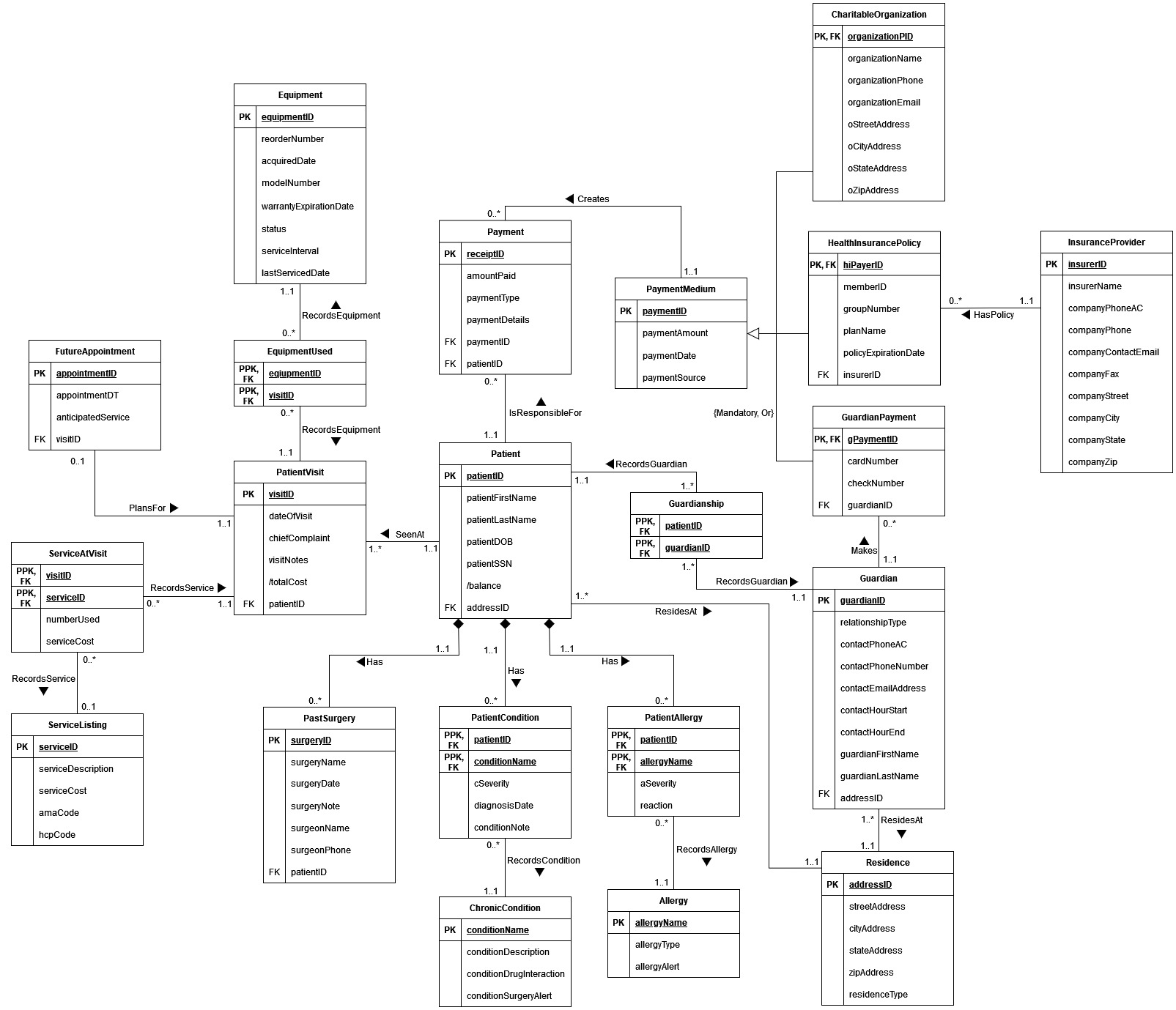
Insurance data is captured through HealthInsurancePolicy, which links patients to their policies and insurers. The InsuranceProvider entity stores insurer details, including contact information and addresses. Each insurance policy is associated with one insurer, but a patient can have zero or multiple policies, creating a zero-to-many relationship between HealthInsurancePolicy and InsuranceProvider.

Entities such as Equipment and EquipmentUsed are used to manage medical devices. The Equipment entity records attributes like model number, and last serviced date. Through the EquipmentUsed associative entity, equipment is linked to specific patient visits, establishing a zero-to-many relationship between EquipmentUsed and PatientVisit.

This ERD provides a comprehensive framework for managing patient care, financial transactions, and operational resources within a healthcare system. Its design supports scalability and detailed recordkeeping, ensuring the system can handle complex workflows while maintaining data integrity.

**(4) RELATIONAL SCHEMA**

**4.1 Logical ERD**

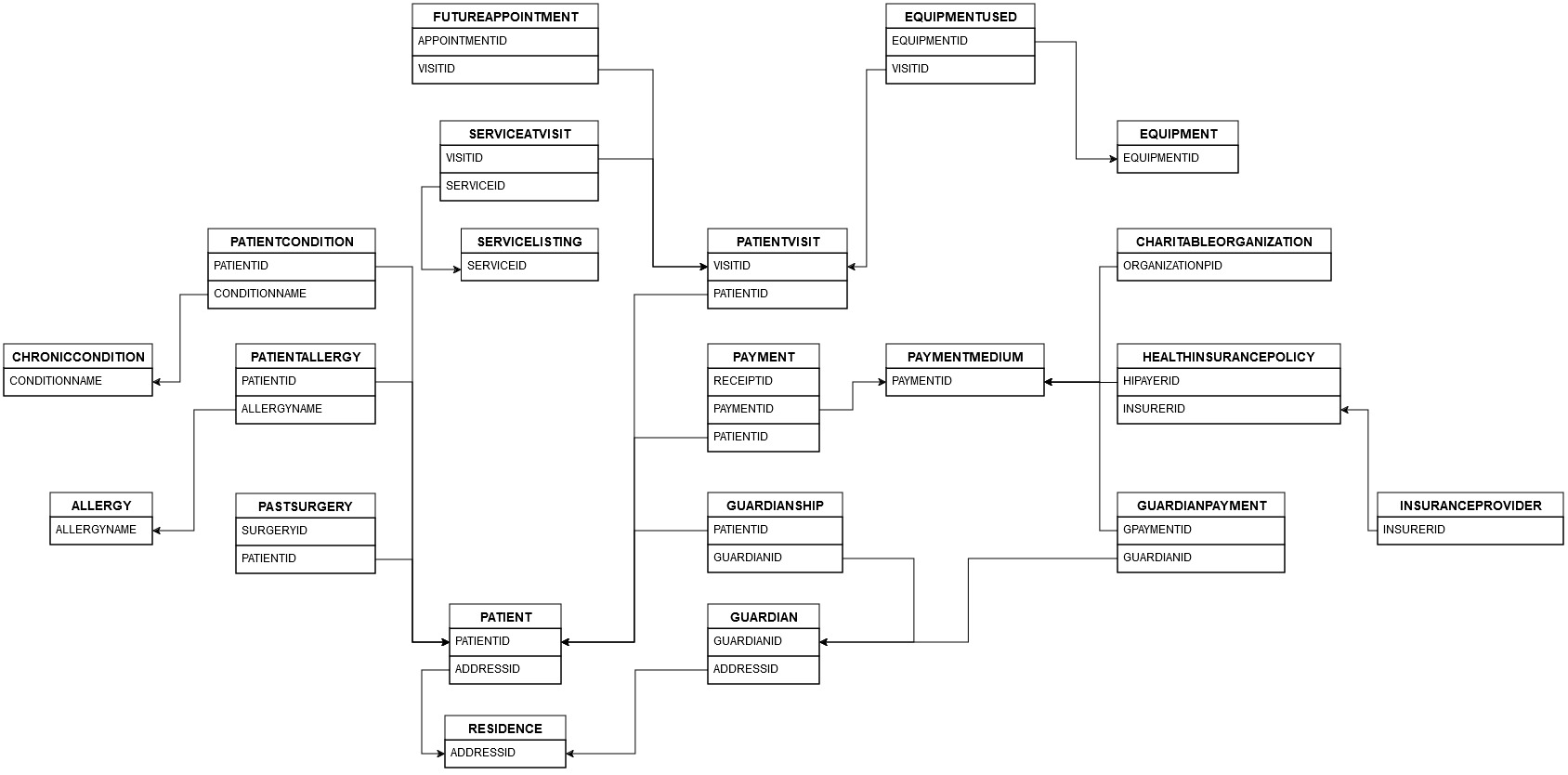


**4.2 Relational Schema**

Alphabetical List of Relational Schemas

|  |  |
| --- | --- |
| Relation | Schema |
| Allergy | * **Allergy** (allergyName, allergyType, allergyAlert) |
| CharitableOrganization | * **CharitableOrganization** (organizationPID, organizationName, organizationPhone, organizationEmail, oStreetAddress, oCityAddress, oStateAddress, oZipAddress)   + Foreign key **organizationPID** references **PaymentMedium** (**paymentID**) |
| ChronicCondition | * **ChronicCondition** (conditionName, conditionDescription, conditionDrugInteraction, conditionSurgeryAlert) |
| Equipment | * **Equipment** (equipmentID, reorderNumber, acquiredDate, modelNumber, warrantyExpirationDate, status, serviceInterval, lastServicedDate) |
| EquipmentUsed | * **EquipmentUsed** (equipmentID, visitID)   + Foreign key **equipmentID** references **Equipment** (**equipmentID**)   + Foreign key **visitID** references **PatientVisit** (**visitID**) |
| FutureAppointment | * **FutureAppointment** (appointmentID, appointmentDT, anticipatedService, visitID)   + Foreign key **visitID** references **PatientVisit** (**visitID**) |
| Guardian | * **Guardian** (guardianID, relationshipType, contactPhoneAC, contactPhoneNumber, contactEmailAddress, contactHourStart, contactHourEnd, guardianFirstName, guardianLastName, addressID)   + Foreign key **addressID** references **Residence** (**addressID**) |
| GuardianPayment | * **GuardianPayment** (gPaymentID, cardNumber, checkNumber, guardianID)   + Foreign key **gPaymentID** references **PaymentMedium** (**paymentID**)   + Foreign key **guardianID** references **Guardian** (**guardianID**) |
| Guardianship | * **Guardianship** (patientID, guardianID)   + Foreign key **patientID** references **Patient** (**patientID**)   + Foreign key **guardianID** references **Guardian** (**guardianID**) |
| HealthInsurancePolicy | * **HealthInsurancePolicy** (hiPayerID, memberID, groupNumber, planName, policyExpirationDate, insurerID)   + Foreign key **hiPayerID** references **PaymentMedium** (**paymentID**)   + Foreign key **insurerID** references **InsuranceProvider (insurerID)** |
| InsuranceProvider | * **InsuranceProvider** (insurerID, insurerName, companyPhoneAC, companyPhone, companyContactEmail, companyFax, companyStreet, companyCity, companyState, companyZip) |
| PastSurgery | * **PastSurgery** (surgeryID, surgeryName, surgeryDate, surgeryNote, surgeonName, surgeonPhone, patientID)   + Foreign key **patientID** references **Patient** (**patientID**) |
| Patient | * **Patient** (patientID, patientFirstName, patientLastName, patientDOB, patientSSN, /balance, addressID)   + Foreign key **addressID** references **Residence** (**addressID**) |
| PatientAllergy | * **PatientAllergy** (patientID, allergyName, aSeverity, reaction)   + Foreign key **patientID** references **Patient** (**patientID**)   + Foreign key **allergyName** references **Allergy** (**allergyName**) |
| PatientCondition | * **PatientCondition** (patientID, conditionName, cSeverity, diagnosisDate, conditionNote)   + Foreign key **patientID** references **Patient** (**patientID**)   + Foreign key **conditionName** references **ChronicCondition** (**conditionName**) |
| PatientVisit | * **PatientVisit** (visitID, dateofVisit, chiefComplaint, visitNotes, /totalcost, patientID)   + Foreign key **patientID** references **Patient** (**patientID**) |
| Payment | * **Payment** (receiptID, amountPaid, paymentType, paymentDetails, paymentID, patientID)   + Foreign key **paymentID** references **PaymentMedium** (**paymentID**)   + Foreign key **patientID** references **Patient** (**patientID**) |
| PaymentMedium | * **PaymentMedium** (paymentID, paymentAmount, paymentDate, paymentSource) |
| Residence | * **Residence** (addressID, streetAddress, cityAddress, stateAddress, zipAddress, residenceType) |
| ServiceAtVisit | * **ServiceAtVisit** (visitID, serviceID, numberUsed, serviceCost)   + Foreign key **visitID** references **PatientVisit** (**visitID**)   + Foreign key **serviceID** references **ServiceListing** (**serviceID**) |
| ServiceListing | * **ServiceListing** (serviceID, serviceDescription, serviceCost, amaCode, hcpCode) |

**4.3 Referential Integrity Diagram**



**(5) DATA DICTIONARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **INSURANCEPROVIDER** |  |  |  |
| INSURERID | NOT NULL | VARCHAR2(4) | Primary Key |
| INSURERNAME | NOT NULL, UNIQUE | VARCHAR2(20) | Name of insurance company |
| COMPANYPHONEAC | - | CHAR(3) | Area code |
| COMPANYPHONE | - | CHAR(7) | Phone number (without area code) |
| COMPANYCONTACTEMAIL | - | VARCHAR2(20) | Email address for contact person |
| COMPANYFAX | - | CHAR(10) | Fax number |
| COMPANYSTREET | - | VARCHAR2(30) | Street address |
| COMPANYCITY | - | VARCHAR2(15) | City |
| COMPANYSTATE | - | CHAR(2) | State abbreviation |
| COMPANYZIP | - | CHAR(5) | 5-digit ZIP Code |
|  |  |  |  |
| **RESIDENCE** |  |  |  |
| ADDRESSID | NOT NULL | VARCHAR2(6) | Primary Key |
| STREETADDRESS | - | VARCHAR2(30) | Street address |
| CITYADDRESS | - | VARCHAR2(15) | City |
| STATEADDRESS | - | CHAR(2) | State abbreviation |
| ZIPADDRESS | - | CHAR(5) | 5-digit ZIP Code |
| RESIDENCETYPE | CHECK | VARCHAR2(10) | Valid residence types include: ‘permanent’, ‘temporary’, ‘transient’ |
|  |  |  |  |
| **ALLERGY** |  |  |  |
| ALLERGYNAME | NOT NULL | VARCHAR2(20) | Name of the allergy and Primary Key |
| ALLERGYTYPE | - | VARCHAR2(20) | Major class of allergy |
| ALLERGYALERT | - | VARCHAR2(40) | Allergy alerts for the patient's treatment |
|  |  |  |  |
| **CHRONICCONDITION** |  |  |  |
| CONDITIONNAME | NOT NULL | VARCHAR2(20) | Name of the condition and Primary Key |
| CONDITIONDESCRIPTION | - | VARCHAR2(60) | A longer description of the condition |
| CONDITIONDRUGINTERACTION | - | VARCHAR2(30) | Any serious drug interactions with the condition |
| CONDITIONSURGERYALERT | - | VARCHAR2(60) | Condition alerts for the patient's treatment |
|  |  |  |  |
| **SERVICELISTING** |  |  |  |
| SERVICEID | NOT NULL | VARCHAR2(6) | Primary Key |
| SERVICEDESCRIPTION | - | VARCHAR2(40) | Description of medical service |
| SERVICECOST | NOT NULL | NUMBER(7,2) | Cost of medical service |
| AMACODE | UNIQUE | VARCHAR2(3) | American Medical Association code |
| HCPCODE | UNIQUE | VARCHAR2(3) | Healthcare Common Procedure code |
|  |  |  |  |
| **EQUIPMENT** |  |  |  |
| EQUIPMENTID | NOT NULL | VARCHAR2(10) | Primary Key |
| REORDERNUMBER | - | VARCHAR2(5) | Reorder number for equipment |
| ACQUIREDDATE | - | DATE | Date the equipment was acquired at the hospital |
| MODELNUMBER | - | VARCHAR2(5) | Equipment model number |
| WARRANTYEXPIRATIONDATE | - | DATE | Date of warranty expiration |
| STATUS | CHECK | VARCHAR2(11) | Valid status types include: ‘service’, ‘maintenance’, ‘repair’, ‘storage’, ‘disposed’ |
| SERVICEINTERVAL | - | NUMBER(4,0) | Recommended interval for equipment maintenance |
| LASTSERVICEDDATE | - | DATE | Date of equipment's last maintenance |
|  |  |  |  |
| **PAYMENTMEDIUM** |  |  |  |
| PAYMENTID | NOT NULL | VARCHAR2(12) | Primary Key |
| PAYMENTAMOUNT | - | NUMBER(8,2) | Amount that was paid towards services rendered |
| PAYMENTDATE | - | DATE | Date of payment |
| PAYMENTSOURCE | CHECK | CHAR(1) | Valid payment source includes: ‘g’, ‘i’, ‘c’, ‘o’ |
|  |  |  |  |
| **PATIENT** |  |  |  |
| PATIENTID | NOT NULL | VARCHAR2(10) | Primary Key |
| PATIENTFIRSTNAME | - | VARCHAR2(20) | First name |
| PATIENTLASTNAME | - | VARCHAR2(20) | Last name |
| PATIENTDOB | - | DATE | Date of birth |
| PATIENTSSN | NOT NULL, UNIQUE | CHAR(9) | Social security number |
| ADDRESSID | - | VARCHAR2(6) | Patient's residence PK |
|  |  |  |  |
| **GUARDIAN** |  |  |  |
| GUARDIANID | NOT NULL | VARCHAR2(10) | Primary Key |
| RELATIONSHIPTYPE | - | VARCHAR2(15) | Relationship to patient |
| CONTACTPHONEAC | - | CHAR(3) | Phone area code |
| CONTACTPHONENUMBER | - | VARCHAR2(10) | Phone number (excluding area code) |
| CONTACTEMAILADDRESS | - | VARCHAR2(20) | Guardian's email address |
| CONTACTHOURSTART | - | CHAR(5) | Start of preferred contact hours (e.g. 12:30, 19:45) |
| CONTACTHOUREND | - | CHAR(5) | End of preferred contact hours (e.g. 16:00, 22:30) |
| GUARDIANFIRSTNAME | - | VARCHAR2(20) | Guardian's first name |
| GUARDIANLASTNAME | - | VARCHAR2(20) | Guardian's last name |
| ADDRESSID | - | VARCHAR2(6) | Guardian's residence PK |
|  |  |  |  |
| **HEALTHINSURANCEPOLICY** |  |  |  |
| HIPAYERID | NOT NULL | VARCHAR2(12) | Primary Key |
| MEMBERID | NOT NULL, UNIQUE | VARCHAR2(10) | Policy member number |
| GROUPNUMBER | - | VARCHAR2(9) | Policy group number |
| PLANNAME | - | VARCHAR2(15) | Name of health insurance plan |
| POLICYEXPIRATIONDATE | - | DATE | Expiration date of health insurance plan |
| INSURERID | - | VARCHAR2(4) | Insurance provider's PK |
|  |  |  |  |
| **CHARITABLEORGANIZATION** |  |  |  |
| ORGANIZATIONPID | NOT NULL | VARCHAR2(12) | Primary Key |
| ORGANIZATIONNAME | UNIQUE | VARCHAR2(20) | Name of charity |
| ORGANIZATIONPHONE | - | CHAR(10) | Contact phone number (including area code) |
| ORGANIZATIONEMAIL | - | VARCHAR2(20) | Email address of charity |
| OSTREETADDRESS | - | VARCHAR2(30) | Street address |
| OCITYADDRESS | - | VARCHAR2(15) | City |
| OSTATEADDRESS | - | CHAR(2) | State abbreviation |
| OZIPADDRESS | - | CHAR(5) | 5-digit ZIP Code |
|  |  |  |  |
| **GUARDIANPAYMENT** |  |  |  |
| GPAYMENTID | NOT NULL | VARCHAR2(12) | Primary Key |
| CARDNUMBER | - | VARCHAR2(20) | Card number (if used) |
| CHECKNUMBER | - | VARCHAR2(20) | Check number (if used) |
| GUARDIANID | - | VARCHAR2(10) | Guardian's PK |
|  |  |  |  |
| **PAYMENT** |  |  |  |
| RECEIPTID | NOT NULL | VARCHAR2(16) | Primary Key |
| AMOUNTPAID | - | NUMBER(8,2) | Amount paid |
| PAYMENTTYPE | CHECK | VARCHAR2(10) | Valid type of payment includes: ‘check’, ‘cash’, ‘credit’, ‘transfer’, ‘debit’ |
| PAYMENTDETAILS | - | VARCHAR2(30) | Additional information about the payment |
| PAYMENTID | - | VARCHAR2(12) | Payment Medium's PK |
| PATIENTID | - | VARCHAR2(10) | Patient's PK |
|  |  |  |  |
| **GUARDIANSHIP** |  |  |  |
| PATIENTID | NOT NULL | VARCHAR2(10) | Partial Primary Key |
| GUARDIANID | NOT NULL | VARCHAR2(10) | Partial Primary Key |
|  |  |  |  |
| **PATIENTALLERGY** |  |  |  |
| PATIENTID | NOT NULL | VARCHAR2(10) | Partial Primary Key |
| ALLERGYNAME | NOT NULL | VARCHAR2(20) | Partial Primary Key |
| ASEVERITY | CHECK | VARCHAR2(20) | Valid severity includes: ‘undetectable’, ‘mild’, ‘moderate’, ‘severe’, ‘life-threatening’ |
| REACTION | - | VARCHAR2(25) | Allergic reaction (e.g. hives, anaphylaxis) |
|  |  |  |  |
| **PATIENTCONDITION** |  |  |  |
| PATIENTID | NOT NULL | VARCHAR2(10) | Partial Primary Key |
| CONDITIONNAME | NOT NULL | VARCHAR2(20) | Partial Primary Key |
| CSEVERITY | CHECK | VARCHAR2(20) | Valid severity includes: ‘undetectable’, ‘mild’, ‘moderate’, ‘severe’, ‘life-threatening’ |
| DIAGNOSISDATE | - | DATE | Date of condition diagnosis |
| CONDITIONNOTE | - | VARCHAR2(50) | Notes about condition |
|  |  |  |  |
| **PATIENTVISIT** |  |  |  |
| VISITID | NOT NULL | VARCHAR2(16) | Primary Key |
| DATEOFVISIT | - | DATE | Date of the visit |
| CHIEFCOMPLAINT | - | VARCHAR2(40) | Patient's chief complaint |
| VISITNOTES | - | VARCHAR2(80) | Notes on the visit |
| PATIENTID | - | VARCHAR2(10) | Patient's PK |
|  |  |  |  |
| **PASTSURGERY** |  |  |  |
| SURGERYID | NOT NULL | VARCHAR2(12) | Primary Key |
| SURGERYNAME | - | VARCHAR2(30) | Name of surgery |
| SURGERYDATE | - | DATE | Date of surgery |
| SURGERYNOTE | - | VARCHAR2(80) | Surgeon's notes |
| SURGEONNAME | - | VARCHAR2(40) | Name of surgeon |
| SURGEONPHONE | - | VARCHAR2(15) | Surgeon's phone (including area code) |
| PATIENTID | - | VARCHAR2(10 | Patient's PK |
|  |  |  |  |
| **EQUIPMENTUSED** |  |  |  |
| EQUIPMENTID | NOT NULL | VARCHAR2(10) | Partial Primary Key |
| VISITID | NOT NULL | VARCHAR2(16) | Partial Primary Key |
|  |  |  |  |
| **FUTUREAPPOINTMENT** |  |  |  |
| APPOINTMENTID | NOT NULL | VARCHAR2(16) | Primary Key |
| APPOINTMENTDT | NOT NULL | DATE | Datetime of appointment |
| ANTICIPATEDSERVICE | - | VARCHAR2(40) | Anticipated service at appointment |
| VISITID | - | VARCHAR2(16) | Patient Visit's PK |
|  |  |  |  |
| **SERVICEATVISIT** |  |  |  |
| VISITID | NOT NULL | VARCHAR2(16) | Partial Primary Key |
| SERVICEID | NOT NULL | VARCHAR2(6) | Partial Primary Key |
| NUMBERUSED | - | NUMBER(3,0) | Number of a particular service used |
| SERVICECOST | - | NUMBER(8,2) | Cost of the service |

**(6) DATABASE IMPLEMENTATION**

**6.1 CREATE TABLE Commands (in separate .SQL file)**

**(7) DATA**

**7.1 INSERT INTO Commands (in separate .SQL file)**

**7.2 Data from SELECT \* QUERIES (in separate MS Word file)**

**(8) DATA QUERIES**

**8.1 Queries by Astrid Ellis**

**8.2 Queries by David Gorski**

1. ***Display the first name and last name of the patient, the first and last name of their guardian and the full contact phone number of their guardian for any*** ***patients who*** ***have an outstanding balance.***

SELECT (Patient.patientFirstName || ' ' || Patient.patientLastName) AS patient\_full\_name,

(Guardian.guardianFirstName || ' ' || Guardian.guardianLastName) AS guardian\_full\_name,

(Guardian.contactPhoneAC || Guardian.contactPhoneNumber) AS phone\_number,

(SUM(ServiceAtVisit.numberUsed \* ServiceAtVisit.serviceCost) - (SUM(amountPaid))) AS outstanding\_balance

FROM Guardian

JOIN Guardianship ON Guardian.guardianID = Guardianship.guardianID

JOIN Patient ON Guardianship.patientID = Patient.patientID

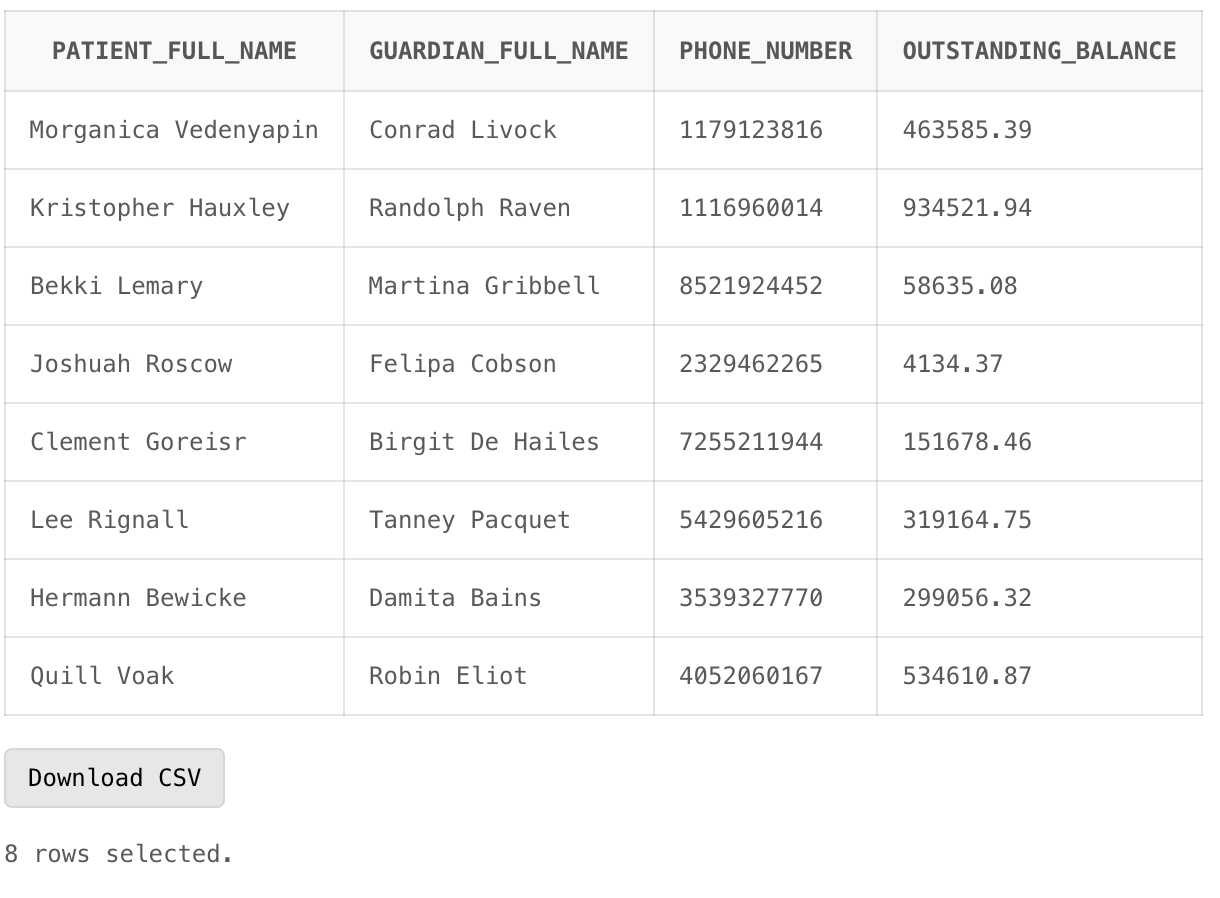
JOIN Payment ON Patient.patientID = Payment.patientID

JOIN PatientVisit ON Patient.patientID = PatientVisit.patientID

JOIN ServiceAtVisit ON PatientVisit.visitID = ServiceAtVisit.visitID

GROUP BY patient.patientFirstName, Patient.patientLastName, Guardian.contactPhoneAC, Guardian.contactPhoneNumber, Guardian.guardianFirstName, guardian.guardianLastName

HAVING (SUM(ServiceAtVisit.numberUsed \* ServiceAtVisit.serviceCost) - (SUM(amountPaid))) > 0;



1. ***We need to mail out letters to patients to ensure that they were not injured by equipment that was in disrepair. Show all the patients (and their full addresses) who have had visits*** ***at the hospital in which the equipment used was listed as in service when it should have been receiving maintenance. Also show the ID numbers of the equipment so it can be repaired.***

SELECT Patient.patientFirstName AS first\_name,

Patient.patientLastName AS lastName,

Residence.streetAddress,

Residence.stateAddress,

Residence.zipAddress,

Equipment.equipmentID

FROM Patient

JOIN Residence ON Patient.addressID = Residence.addressID

JOIN PatientVisit ON Patient.patientID = PatientVisit.patientID

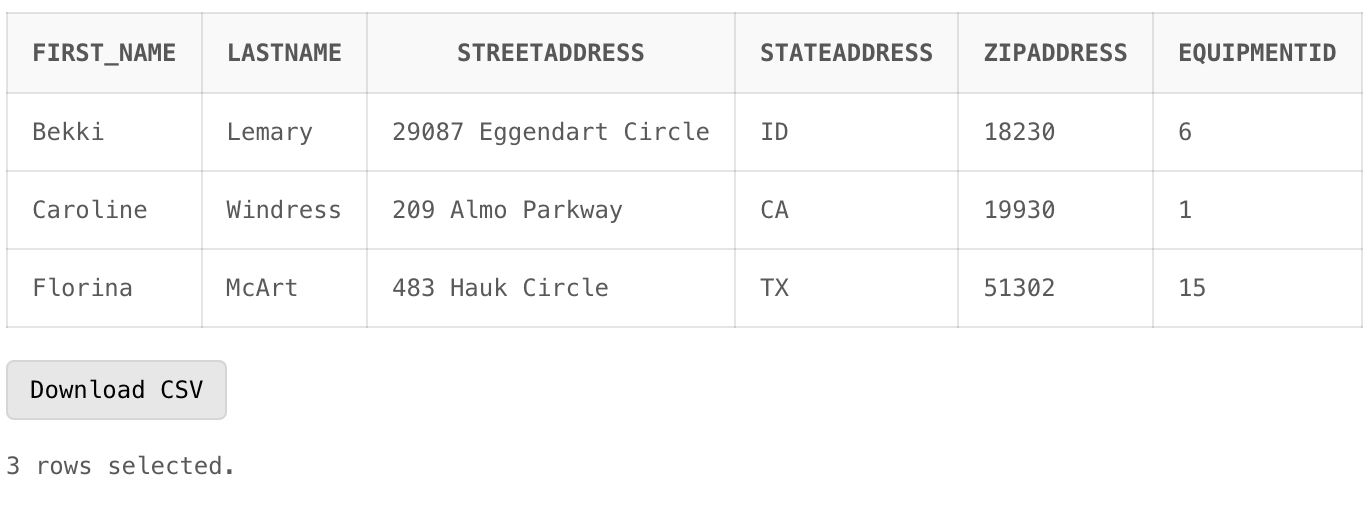
JOIN EquipmentUsed ON PatientVisit.visitID = EquipmentUsed.visitID

JOIN Equipment ON EquipmentUsed.equipmentID = Equipment.equipmentID

WHERE (patientVisit.dateOfVisit > (Equipment.lastServicedDate + Equipment.ServiceInterval))

AND Equipment.status = 'service'

ORDER BY Patient.patientFirstName, Patient.patientLastName;



1. ***St. Jude, one of the charitable organizations with which we work, needs some financial information for an upcoming marketing campaign. For each patient in 2023 on which they spent at least $200, they would like to know the total amount spent on the patient and the patient's first name.***

SELECT SUM(PaymentMedium.paymentAmount) as total\_paid\_for\_patient,

Patient.patientFirstName

FROM CharitableOrganization

JOIN PaymentMedium ON CharitableOrganization.organizationPID = PaymentMedium.paymentID

JOIN Payment ON PaymentMedium.paymentID = Payment.paymentID

JOIN Patient ON Payment.patientID = Patient.patientID

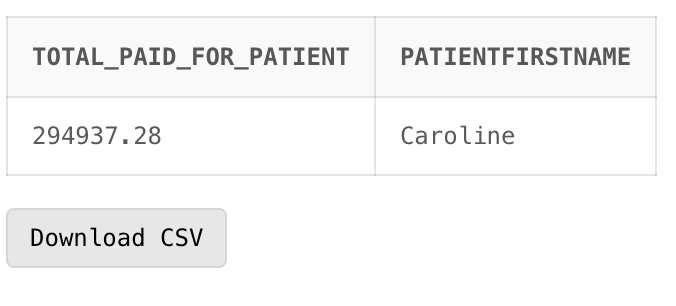
WHERE CharitableOrganization.organizationName = 'St. Jude'

AND PaymentMedium.paymentDate BETWEEN '01-JAN-2023' AND '31-DEC-2023'

GROUP BY Patient.patientFirstName

HAVING SUM(PaymentMedium.paymentAmount) >= 200

ORDER BY Patient.patientFirstName;



**8.3 Queries by Josh Hoffman**

1. The hospital has been accused of malpractice for not taking into account patient allergies in surgery prep. Show all of the surgeries performed on patients with allergies from 2010 to 2015, what kind of allergy the patient had, and the number of surgeries per kind of allergy.

SELECT PastSurgery.surgeryName AS surgery, PatientAllergy.allergyName AS allergy, PastSurgery.surgeryDate,

COUNT(PastSurgery.surgeryID) AS num\_surgeries

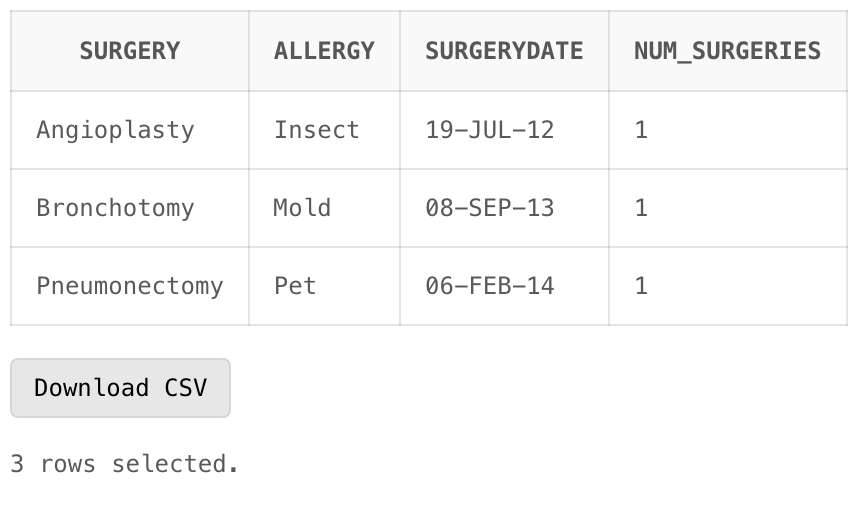
FROM PatientAllergy

JOIN Patient ON PatientAllergy.patientID = Patient.patientID

JOIN PastSurgery ON Patient.patientID = PastSurgery.patientID

WHERE pastSurgery.surgeryDate BETWEEN '01-JAN-2010' AND '31-DEC-2015'

GROUP BY PastSurgery.surgeryName, PatientAllergy.allergyName, PastSurgery.surgeryDate



1. Keeping on top of scheduling appointments is a vital part of ensuring your patients receive the medical care they need.  To aid administrative staff in finding the patients who might have slipped through the cracks, show the patients who have visited the hospital but do not yet have a future appointment booked.  These patients would then be called to book an appointment.

SELECT P.PATIENTFIRSTNAME as "Patient First Name", P.PATIENTLASTNAME as "Patient Last Nane", G.CONTACTPHONEAC as "Area Code", G.CONTACTPHONENUMBER as "Phone Number", FA.APPOINTMENTDT as "Appointment Date"

FROM PATIENT P

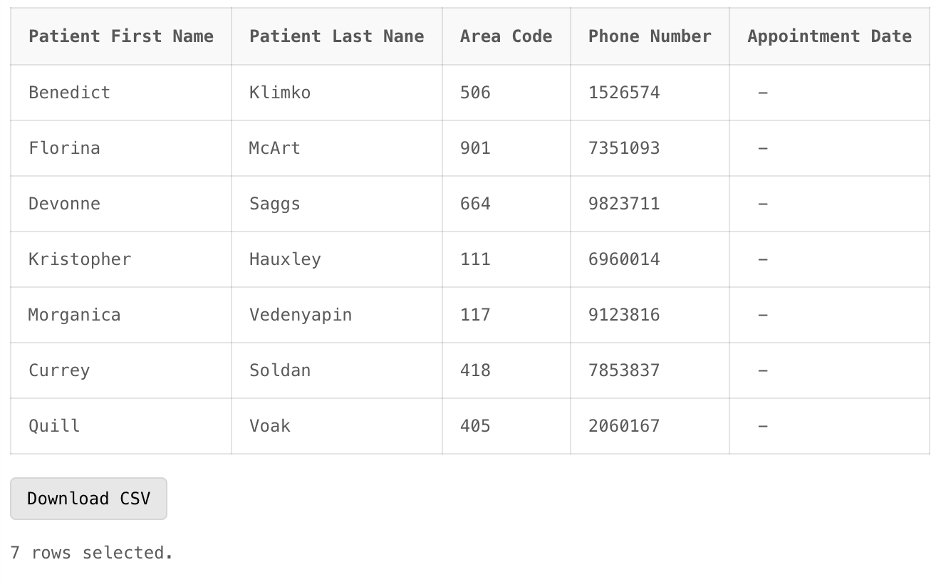
JOIN PATIENTVISIT PV ON P.PATIENTID = PV.PATIENTID

JOIN GUARDIANSHIP GS ON P.PATIENTID = GS.PATIENTID

JOIN GUARDIAN G ON GS.GUARDIANID = G.GUARDIANID

LEFT JOIN FUTUREAPPOINTMENT FA ON PV.VISITID = FA.VISITID

WHERE (FA.APPOINTMENTDT <= SYSDATE OR FA.APPOINTMENTDT IS NULL)



1. The hospital is considering expanding staff who provide medical services but wants to ensure they add positions where it will have the most impact. They want to evaluate which services are being used the most and add more positions for those departments. Rank the services used and how many times they have been used.

SELECT SL.SERVICEDESCRIPTION as "Service Used", SUM(SV.numberUsed) as "Number Of Uses"

FROM SERVICELISTING SL

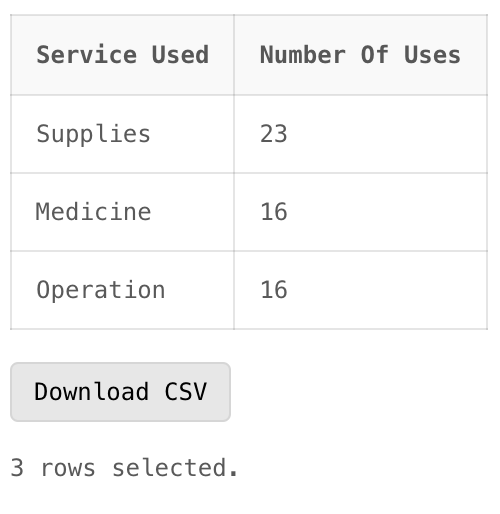
JOIN SERVICEATVISIT SV ON SL.SERVICEID = SV.SERVICEID

JOIN PATIENTVISIT PV ON SV.VISITID = PV.VISITID

JOIN PATIENT P ON PV.PATIENTID = P.PATIENTID

GROUP BY SL.SERVICEDESCRIPTION

ORDER BY (SUM(SV.numberUsed)) DESC



**8.4 Queries by Steven Sullivan**

1. **Total Service Costs for Patients with Allergies:**

Display the first name and last name of patients, their allergy details, and the total service cost incurred by each patient.

SELECT

Patient.patientFirstName || ' ' || Patient.patientLastName AS patient\_full\_name,

Allergy.allergyName AS allergy\_name,

SUM(ServiceAtVisit.numberUsed \* ServiceAtVisit.serviceCost) AS total\_service\_cost

FROM

Patient

JOIN

PatientAllergy ON Patient.patientID = PatientAllergy.patientID

JOIN

Allergy ON PatientAllergy.allergyName = Allergy.allergyName

JOIN

PatientVisit ON Patient.patientID = PatientVisit.patientID

JOIN

ServiceAtVisit ON PatientVisit.visitID = ServiceAtVisit.visitID

GROUP BY

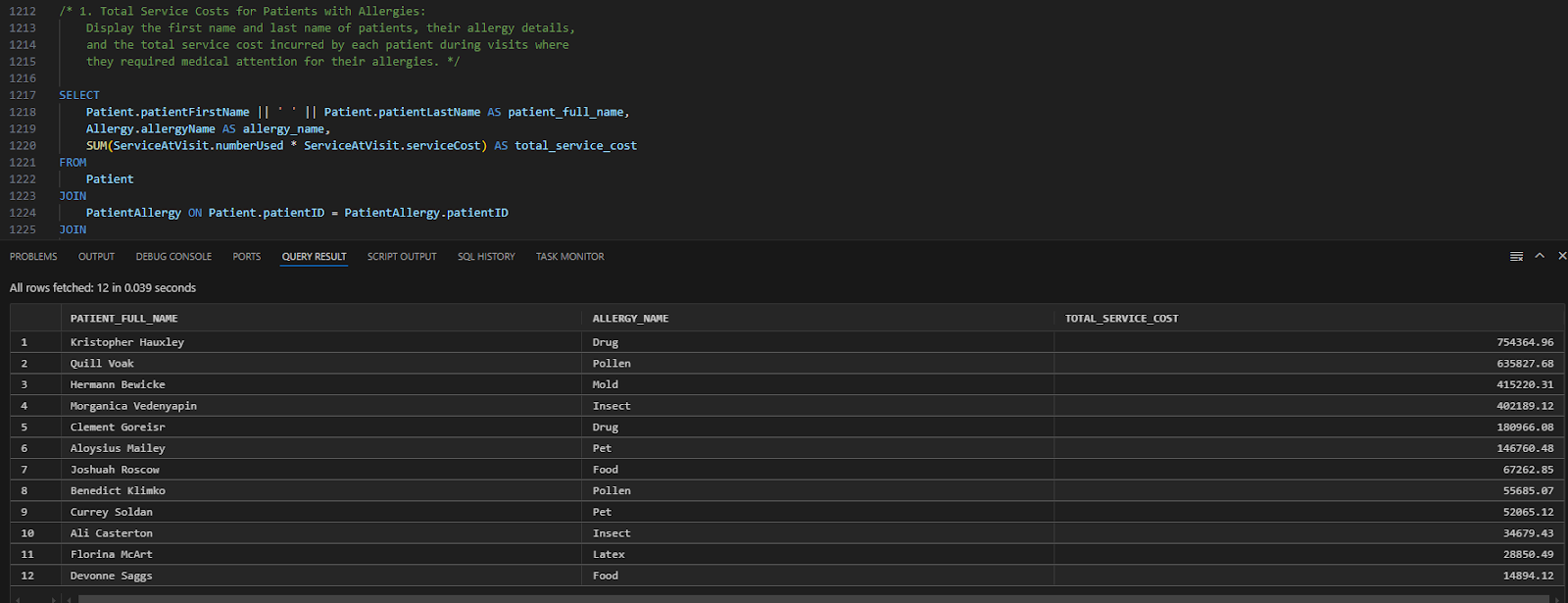
Patient.patientFirstName, Patient.patientLastName, Allergy.allergyName

HAVING

SUM(ServiceAtVisit.numberUsed \* ServiceAtVisit.serviceCost) > 0

ORDER BY

total\_service\_cost DESC;



**2. Total Healthcare Expenditure by City:**

Identify and display the total healthcare expenditure incurred by patients grouped by their city of residence. This query helps to analyze healthcare costs at the community level, showcasing which cities contribute the most to healthcare spending.

SELECT Residence.cityAddress AS city,

SUM(ServiceAtVisit.numberUsed \* ServiceAtVisit.serviceCost) AS total\_expenditure

FROM Patient

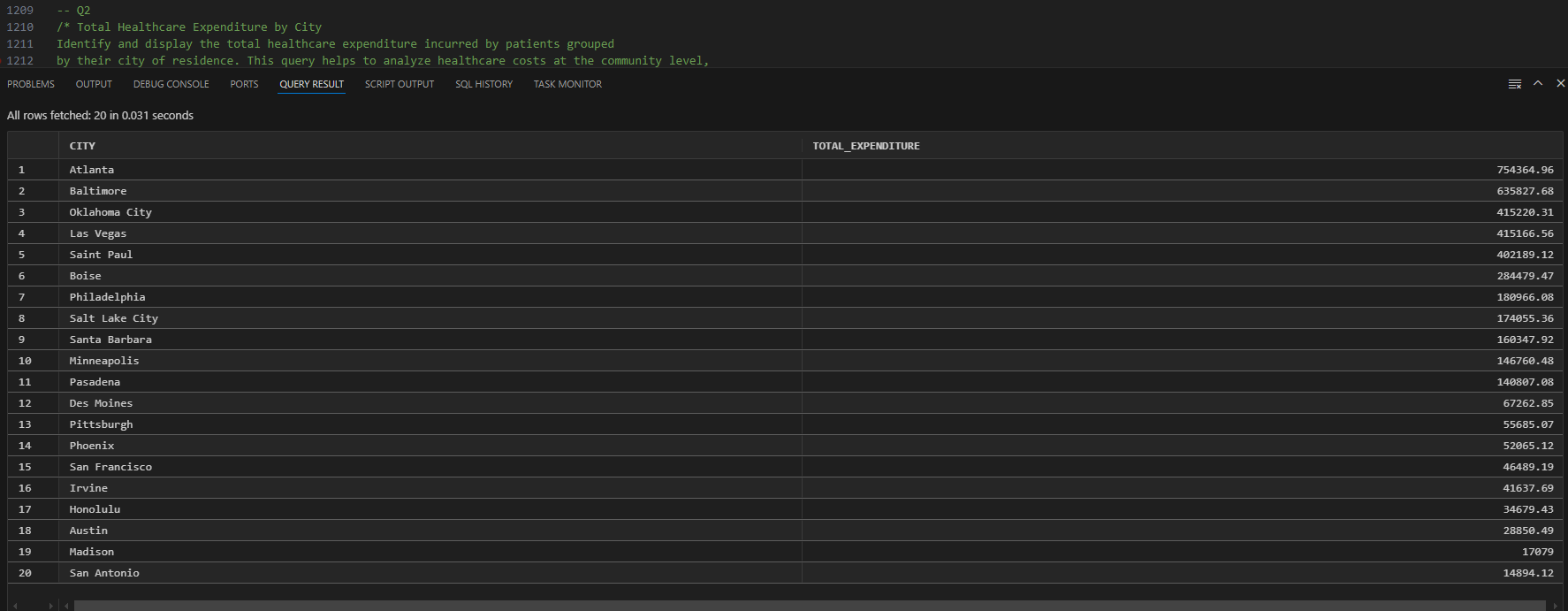
JOIN Residence ON Patient.addressID = Residence.addressID

JOIN PatientVisit ON Patient.patientID = PatientVisit.patientID

JOIN ServiceAtVisit ON PatientVisit.visitID = ServiceAtVisit.visitID

GROUP BY Residence.cityAddress

ORDER BY total\_expenditure DESC;



**3. Total Healthcare Expenditure by Guardians**

This query calculates the total healthcare expenditure incurred by each guardian’s associated patients across all their visits. It displays the guardian's full name and the total costs of services consumed by their linked patients, ordered by expenditure from highest to lowest. This is useful for analyzing the financial responsibility of guardians in the system.

SELECT

(Guardian.guardianFirstName || ' ' || Guardian.guardianLastName) AS guardian\_full\_name,

SUM(ServiceAtVisit.numberUsed \* ServiceAtVisit.serviceCost) AS total\_expenditure

FROM Guardian

JOIN Guardianship ON Guardian.guardianID = Guardianship.guardianID

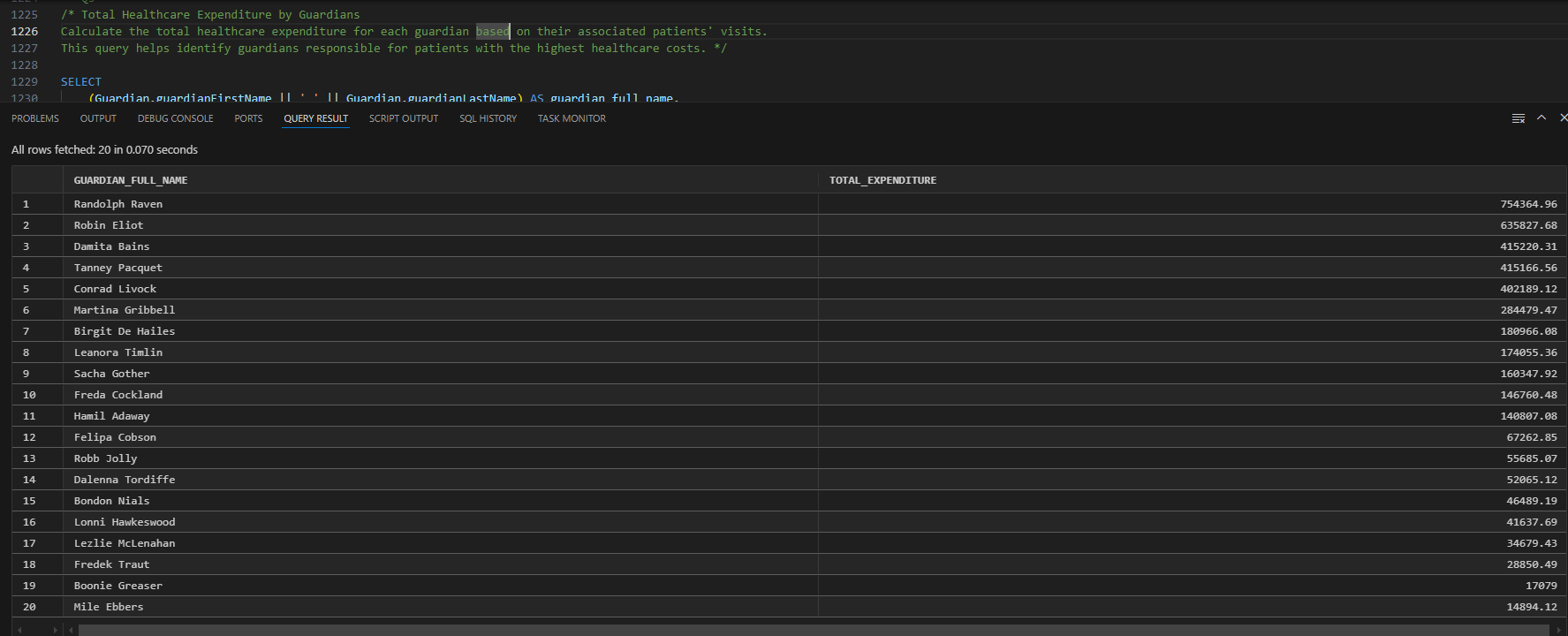
JOIN Patient ON Guardianship.patientID = Patient.patientID

JOIN PatientVisit ON Patient.patientID = PatientVisit.patientID

JOIN ServiceAtVisit ON PatientVisit.visitID = ServiceAtVisit.visitID

GROUP BY Guardian.guardianFirstName, Guardian.guardianLastName

ORDER BY total\_expenditure DESC;



**(9) DATA MANIPULATION**

**9.1 DML by Astrid Ellis**

**9.2 DML by David Gorski**

**Data before the UPDATE command:**  
 SELECT \* FROM Equipment

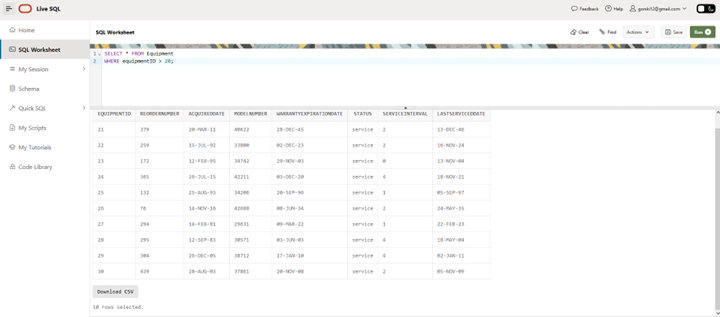
WHERE equipmentID > 20;

*Oracle output:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EQUIPMENTID** | **REORDERNUMBER** | **ACQUIREDDATE** | **MODELNUMBER** | **WARRANTYEXPIRATIONDATE** | **STATUS** | **SERVICEINTERVAL** | **LASTSERVICEDDATE** |
| 21 | 379 | 20-MAR-11 | 40622 | 28-DEC-45 | service | 2 | 13-DEC-46 |
| 22 | 259 | 15-JUL-92 | 33800 | 02-DEC-23 | service | 2 | 16-NOV-24 |
| 23 | 172 | 12-FEB-95 | 34742 | 29-NOV-03 | service | 0 | 13-NOV-04 |
| 24 | 365 | 26-JUL-15 | 42211 | 03-DEC-20 | service | 4 | 18-NOV-21 |
| 25 | 132 | 25-AUG-93 | 34206 | 20-SEP-96 | service | 1 | 05-SEP-97 |
| 26 | 76 | 14-NOV-16 | 42688 | 08-JUN-34 | service | 2 | 24-MAY-35 |
| 27 | 294 | 14-FEB-81 | 29631 | 09-MAR-22 | service | 1 | 22-FEB-23 |
| 28 | 295 | 12-SEP-83 | 30571 | 03-JUN-03 | service | 4 | 18-MAY-04 |
| 29 | 304 | 26-DEC-05 | 38712 | 17-JAN-10 | service | 4 | 02-JAN-11 |
| 30 | 439 | 28-AUG-03 | 37861 | 20-NOV-08 | service | 2 | 05-NOV-09 |

10 rows selected.

**Screenshot – Data before the UPDATE command**



**UPDATE command:**

UPDATE Equipment

SET status = 'repair'

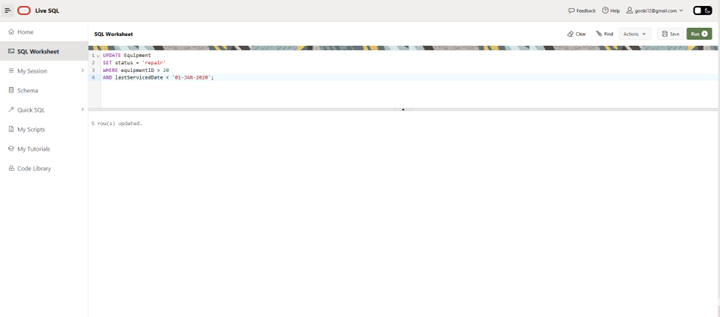
WHERE equipmentID > 20

AND lastServicedDate < '01-JAN-2020';

*Oracle output:*

5 row(s) updated.

**Screenshot – UPDATE command**



**Data after the UPDATE command:**

SELECT \* FROM Equipment

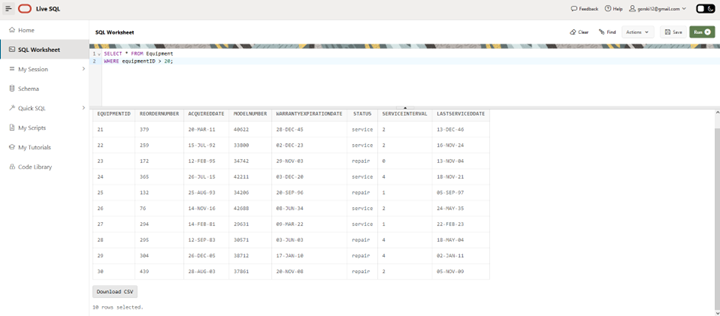
WHERE equipmentID > 20;

*Oracle output:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EQUIPMENTID** | **REORDERNUMBER** | **ACQUIREDDATE** | **MODELNUMBER** | **WARRANTYEXPIRATIONDATE** | **STATUS** | **SERVICEINTERVAL** | **LASTSERVICEDDATE** |
| 21 | 379 | 20-MAR-11 | 40622 | 28-DEC-45 | service | 2 | 13-DEC-46 |
| 22 | 259 | 15-JUL-92 | 33800 | 02-DEC-23 | service | 2 | 16-NOV-24 |
| 23 | 172 | 12-FEB-95 | 34742 | 29-NOV-03 | repair | 0 | 13-NOV-04 |
| 24 | 365 | 26-JUL-15 | 42211 | 03-DEC-20 | service | 4 | 18-NOV-21 |
| 25 | 132 | 25-AUG-93 | 34206 | 20-SEP-96 | repair | 1 | 05-SEP-97 |
| 26 | 76 | 14-NOV-16 | 42688 | 08-JUN-34 | service | 2 | 24-MAY-35 |
| 27 | 294 | 14-FEB-81 | 29631 | 09-MAR-22 | service | 1 | 22-FEB-23 |
| 28 | 295 | 12-SEP-83 | 30571 | 03-JUN-03 | repair | 4 | 18-MAY-04 |
| 29 | 304 | 26-DEC-05 | 38712 | 17-JAN-10 | repair | 4 | 02-JAN-11 |
| 30 | 439 | 28-AUG-03 | 37861 | 20-NOV-08 | repair | 2 | 05-NOV-09 |

10 rows selected.

**Screenshot – Data after the UPDATE command**



**Data before the DELETE command:**

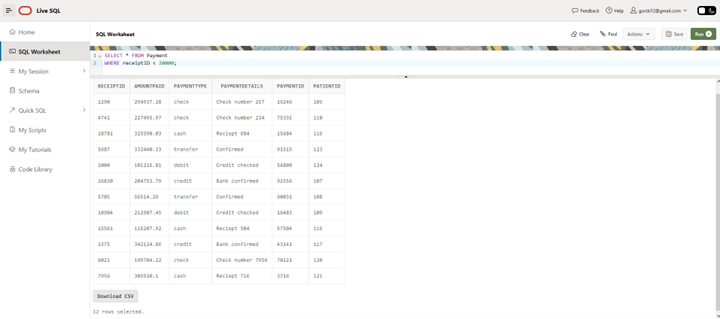
SELECT \* FROM Payment

WHERE receiptID < 20000;

*Oracle output:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RECEIPTID** | **AMOUNTPAID** | **PAYMENTTYPE** | **PAYMENTDETAILS** | **PAYMENTID** | **PATIENTID** |
| 1290 | 294937.28 | check | Check number 257 | 16246 | 105 |
| 4741 | 227495.97 | check | Check number 234 | 75392 | 110 |
| 18781 | 325590.03 | cash | Reciept 684 | 15684 | 116 |
| 5687 | 332440.33 | transfer | Confirmed | 91515 | 123 |
| 1000 | 101216.81 | debit | Credit checked | 54808 | 124 |
| 16830 | 204753.79 | credit | Bank confirmed | 92556 | 107 |
| 5705 | 56514.29 | transfer | Confirmed | 60031 | 108 |
| 18904 | 212507.45 | debit | Credit checked | 16483 | 109 |
| 15561 | 116207.92 | cash | Reciept 504 | 57504 | 116 |
| 1375 | 342124.86 | credit | Bank confirmed | 43141 | 117 |
| 6021 | 199704.22 | check | Check number 7956 | 70121 | 120 |
| 7956 | 305920.1 | cash | Reciept 716 | 3716 | 121 |

12 rows selected.

**Screenshot – Data before the DELETE command**

**DELETE command:**

DELETE FROM Payment

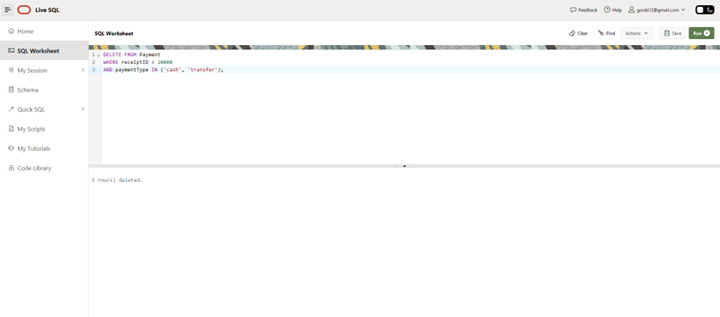
WHERE receiptID < 20000

AND paymentType IN (‘cash’, ‘transfer’);

*Oracle output:*

5 row(s) deleted.

**Screenshot – DELETE command**



**Data after the DELETE command:**

SELECT \* FROM Payment

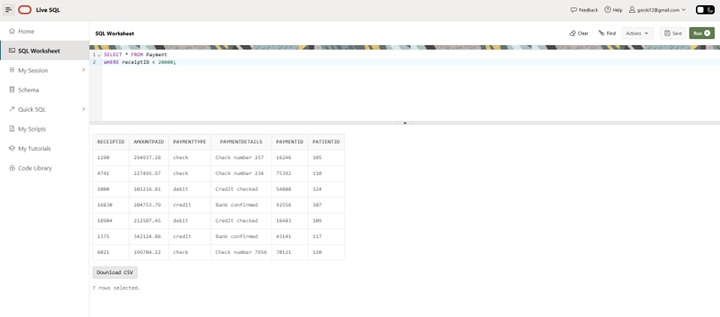
WHERE receiptID < 20000;

*Oracle output:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RECEIPTID** | **AMOUNTPAID** | **PAYMENTTYPE** | **PAYMENTDETAILS** | **PAYMENTID** | **PATIENTID** |
| 1290 | 294937.28 | check | Check number 257 | 16246 | 105 |
| 4741 | 227495.97 | check | Check number 234 | 75392 | 110 |
| 1000 | 101216.81 | debit | Credit checked | 54808 | 124 |
| 16830 | 204753.79 | credit | Bank confirmed | 92556 | 107 |
| 18904 | 212507.45 | debit | Credit checked | 16483 | 109 |
| 1375 | 342124.86 | credit | Bank confirmed | 43141 | 117 |
| 6021 | 199704.22 | check | Check number 7956 | 70121 | 120 |

7 rows selected.

**Screenshot – Data after the DELETE command**



**9.3 DML by Josh Hoffman**

**Update Command**

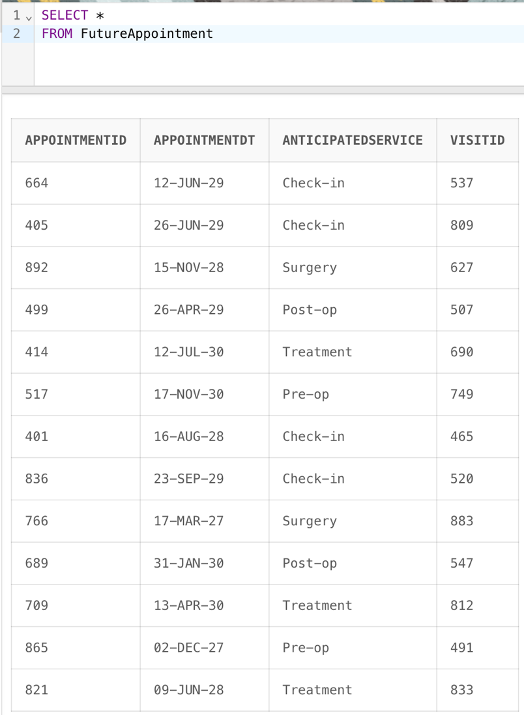
Change a future appointment for patent Caroline Windress from June 12th, 2029 to December 15th, 2024. The patient called, and they need to move up their appointment due to changing orders from their doctor.

UPDATE FutureAppointment

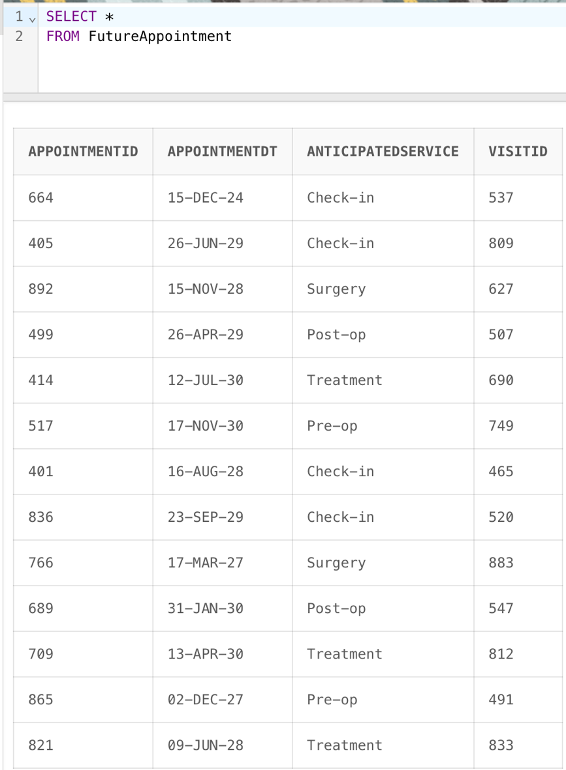
SET APPOINTMENTDT = '15-DEC-2024'

WHERE VISITID = '537'

BEFORE



AFTER



Delete Command

The hospital has discovered that during the intake process, contact information was given for an individual who does not have guardianship over patient Bat Mason. Their information needs to be deleted to prevent the hospital from contacting them with privileged information. The name of the false-guardian is Lonni Hawkeswood.

DELETE FROM GUARDIANSHIP

WHERE GUARDIANID = '99316';

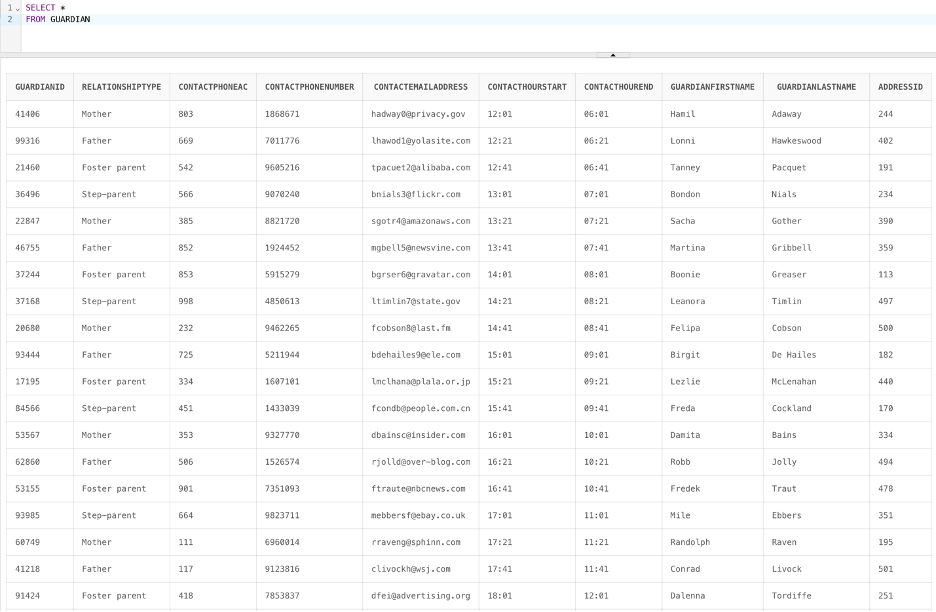
DELETE FROM GUARDIANPAYMENT

WHERE GUARDIANID = '99316';

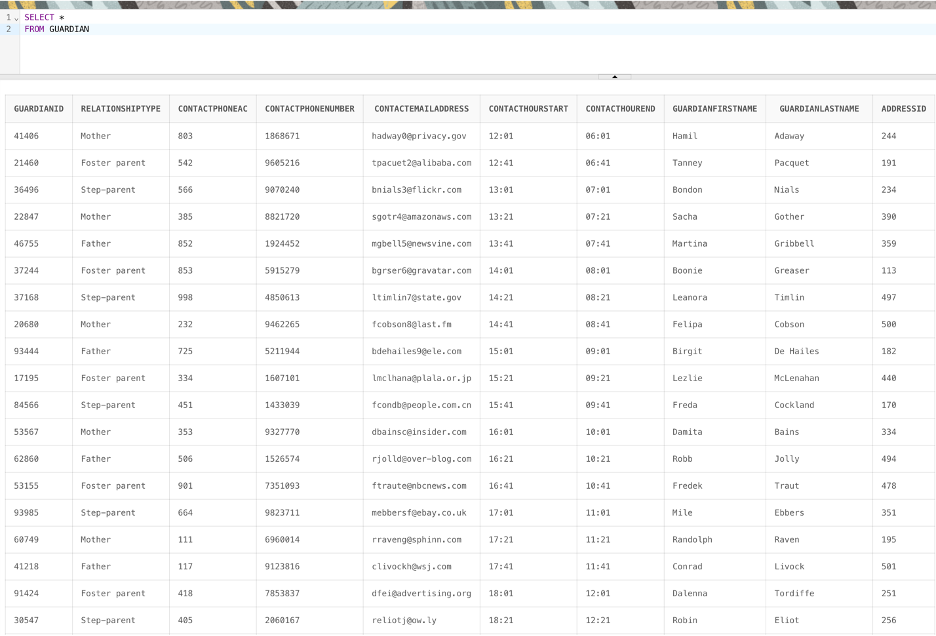
DELETE FROM GUARDIAN

WHERE GUARDIANID = '99316';

BEFORE



AFTER



**9.4 DML by Steven Sullivan**

**Update Command:**

Update the groupNumber in the HealthInsurancePolicy table to "999999999" for all patients whose policy expiration date is in the past.

-- Before Update

SELECT receiptID, amountPaid, paymentType

FROM Payment

WHERE amountPaid > 300000;

-- Update Command

UPDATE Payment

SET paymentType = 'transfer'

WHERE amountPaid > 300000;

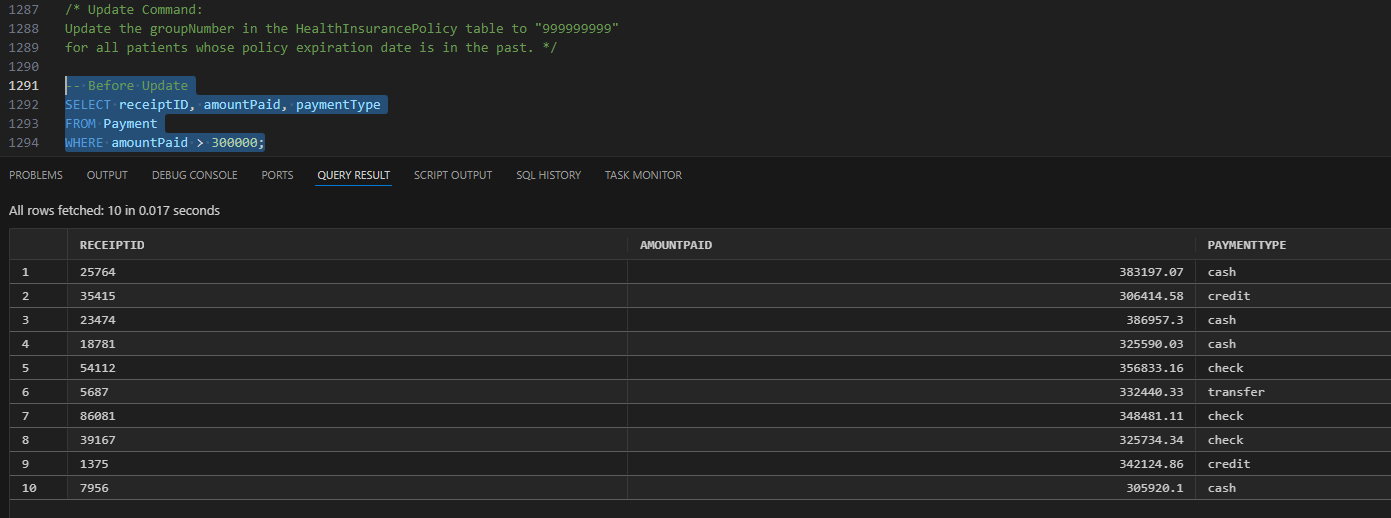
-- After Update

SELECT receiptID, amountPaid, paymentType

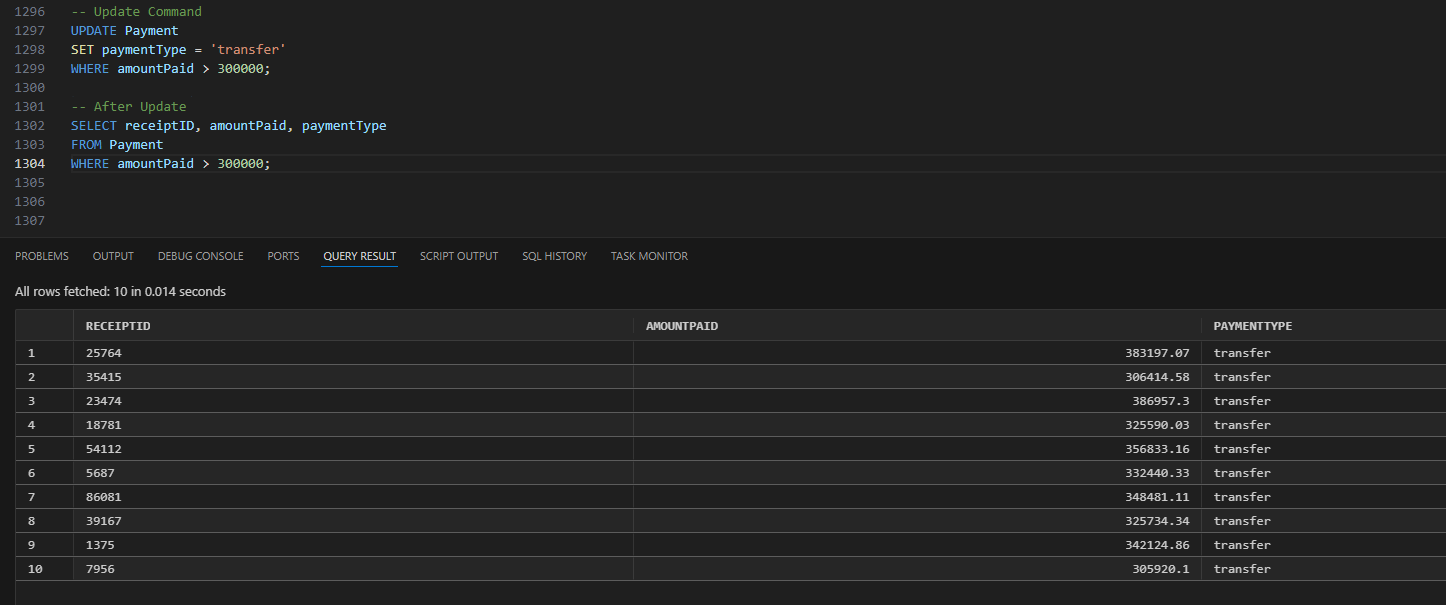
FROM Payment

WHERE amountPaid > 300000;

**Before Screenshot:**



**After Screenshot:**



**Delete Command:**

Delete all rows from the PatientCondition table where the severity (cSeverity) is marked as undetectable and the diagnosisDate is before January 1, 2010.

-- Before Delete

SELECT patientID, conditionName, cSeverity, diagnosisDate

FROM PatientCondition

WHERE cSeverity = 'undetectable' AND diagnosisDate < TO\_DATE('01-JAN-2010', 'DD-MON-YYYY');

-- Delete Command

DELETE FROM PatientCondition

WHERE cSeverity = 'undetectable' AND diagnosisDate < TO\_DATE('01-JAN-2010', 'DD-MON-YYYY');

-- After Delete

SELECT patientID, conditionName, cSeverity, diagnosisDate

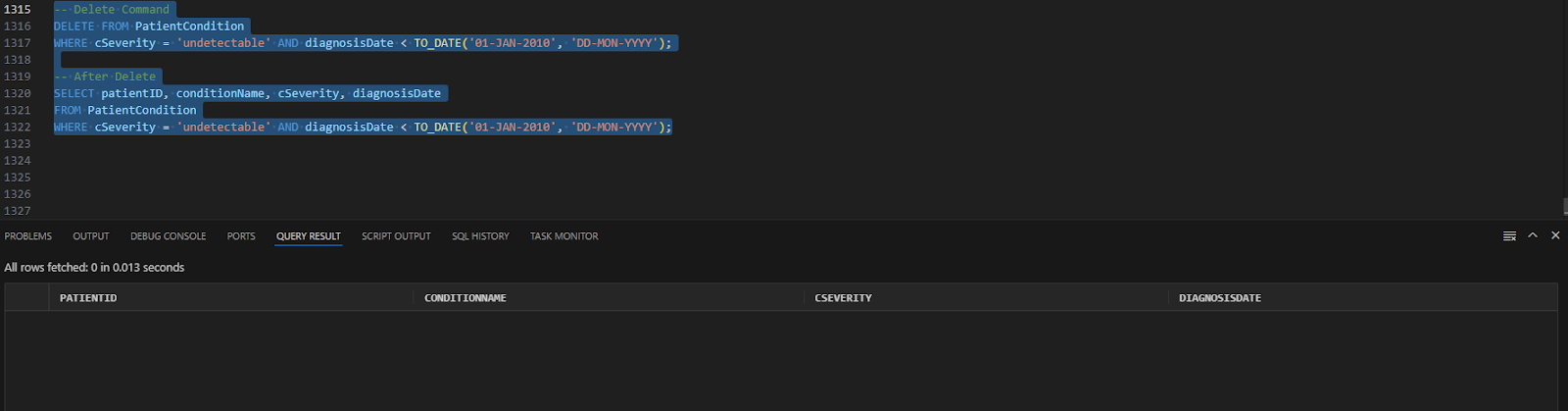
FROM PatientCondition

WHERE cSeverity = 'undetectable' AND diagnosisDate < TO\_DATE('01-JAN-2010', 'DD-MON-YYYY');

**Before Screenshot:**



**After Screenshot:**



**(10) SUMMARY**

**10.1 Summary by Astrid Ellis**

**10.2 Summary by David Gorski**

Our team project for INFO 605, developing a database for a Children's Hospital, provided a comprehensive foundation of database architecture. By translating the hospital's real-world system into conceptual and logical ERDs, I significantly improved my database design skills and found a real passion for the process. This project also deepened my SQL knowledge. I learned to write more accurate and efficient queries by understanding their logical processing order, effectively seeing what goes on 'under the hood.' The practical application of these concepts to a real-world scenario solidified my understanding and gave me the confidence to tackle complex database challenges. I was so engaged with this project that, given more time, I would have loved to fully implement a functional hospital database, exploring the complex relationships and data requirements of a working medical facility. This hands-on experience was incredibly beneficial to my learning, and I feel much more confident in my database construction and management skills. I am eager to apply these newfound skills to future projects and explore more advanced database concepts.

**10.3 Summary by Josh Hoffman**

In conclusion, this project shepherds the creation of a database for a hospital from start to finish. It begins in the initial stages of understanding the requirements of the system, the conceptual level understanding the involved entities, the logical level of creating the relationships in the data, as well as the physical level in designing the database itself, and ends with the operation of the database to satisfy operational goals. The final product of this process is a database representing a subsect of a viable database used to aid in the operation of a hospital. To make the project feasible in the class timeframe, the scope was limited to include payments, equipment, patients, guardians, medical history, appointments, and insurance. To fully implement such a database in production, several additional entities would need to be added to the scope, like medication, staff, locations, and room information. The lesson learned that stands out to me the most is how supportive having a fully fleshed logical ERD is in the process of creating an Oracle database. As demonstrated through the business goal-driven SQL queries, our database allows the hospital to answer critical business questions like who needs to make an appointment, who still needs to pay for services, evaluate liability exposure, and where to add more staff strategically. If more time permitted, I would add staff information to the database as they have a critical role in the operation of a hospital.

**10.4 Summary by Steven Sullivan**

This project focuses on developing a comprehensive database system for a children’s hospital, addressing the domain of healthcare management and patient care. The database effectively organizes critical information, including patient records, visit logs, guardian details, surgical histories, allergies, chronic conditions, financial transactions, and medical equipment tracking. The database’s scope ensures that essential entities such as Patient, Guardian, Residence, InsuranceProvider, and Equipment are connected through logical relationships. These relationships facilitate seamless workflows for patient care, scheduling appointments, tracking services provided, and managing insurance and payment records. By employing a normalized design, the database minimizes redundancy, enhances data integrity, and supports efficient data retrieval.

The project offered valuable lessons. Understanding the importance of normalization was vital for creating a strong and scalable database. Writing SQL queries involving joins, aggregation functions, and conditional criteria provided hands-on experience with relational database management. Moreover, mapping real-world healthcare requirements into the database model highlighted the necessity of domain knowledge in database design. Collaboration within the team ensured unique contributions and required efficient communication to avoid redundancy in tasks, reinforcing teamwork and version control practices.

The database aligns with the hospital’s goals by supporting efficient and accurate data management. It enhances patient care through detailed health records, improves financial transparency, and optimizes resource allocation, such as tracking medical equipment usage and insurance claims. In the future, the database could be expanded to incorporate predictive analytics for patient health trends, additional non-healthcare services like cafeteria management, and advanced reporting tools for hospital administrators. This project demonstrates how structured data management can address complex organizational needs in a healthcare setting.