**Question 1: Basic database design and queries**

Summary: 4 tables, author, book, reviewer, report

1. Minimum database for ER Diagram with key identification and relationship multiplicities.

A screenshot of a computer

Description automatically generated

1. Global Relation Diagram (GRD), matching with ERD above, with all attributes

A screenshot of a computer

Description automatically generated

1. SQL script creating tables

|  |
| --- |
| -- Optional  -- Drop everything if wish to reset the data  DROP TABLE Report;  DROP TABLE Book;  DROP TABLE Author, Reviewer; |
| -- iii. Create 4 tables (Main part)  CREATE TABLE Author(  authorID NUMERIC(10,0),  name varchar(50) NOT NULL,  CONSTRAINT author\_pk PRIMARY KEY (authorID)  );  CREATE TABLE Reviewer(  reviewerID NUMERIC(10,0),  name varchar(50) NOT NULL,  CONSTRAINT reviewer\_pk PRIMARY KEY (reviewerID)  );  CREATE TABLE Book(  bookID NUMERIC(10,0),  title varchar(100) NOT NULL,  authorID NUMERIC(10,0),  CONSTRAINT book\_pk PRIMARY KEY (bookID),  CONSTRAINT book\_fk FOREIGN KEY (authorID) REFERENCES Author  );  CREATE TABLE Report(  bookID NUMERIC(10,0),  reviewerID NUMERIC(10,0),  rating INTEGER CHECK (rating BETWEEN 1 AND 5) NOT NULL,  postedDate DATE NOT NULL,  CONSTRAINT report\_pk PRIMARY KEY (bookID, reviewerID),  CONSTRAINT report\_fk1 FOREIGN KEY (bookID) REFERENCES Book,  CONSTRAINT report\_fk2 FOREIGN KEY (reviewerID) REFERENCES Reviewer  ); |
| -- See (blank) table creation  SELECT \* FROM Author;  SELECT \* FROM Reviewer;  SELECT \* FROM Book;  SELECT \* FROM Report; |

1. SQL script inserting records with at least 3 per table, at least 2 books having at least 2 reviewer’s report (Screenshot after code)

|  |
| --- |
| -- iv. Insert values  INSERT INTO Author VALUES -- authorID, name  (1, 'JK Rowling'),  (2, 'George RR Martin'),  (3, 'JRR Tolkien'),  (4, 'CS Lewis'),  (5, 'Elio Nguyen'),  (6, 'Du Nguyen'),  (7, 'Lan Kim'); |
| INSERT INTO Reviewer VALUES -- reviewerID, name  (11, 'Charlie Nguyen'),  (12, 'Rosy Ta'),  (13, 'Ken Lin'),  (14, 'Celine Tran'),  (15, 'BaHoa Truong'),  (16, 'BaThuy Nguyen'),  (17, 'QuinQuen Nguyen'); |
| INSERT INTO Book VALUES –- bookID, title, authorID  (1, 'Harry Potter and the Deathly Hallows', 1),  (2, 'Harry Potter and the Chamber of Secrets', 1),  (3, 'Harry Potter and the Prisoner of Azkaban', 1),  (4, 'A Game of Thrones', 2),  (5, 'The Fellowship of the Ring', 3),  (6, 'The Two Towers', 3),  (7, 'My life from nqphu\_412', 5),  (8, 'Kieu Story', 6),  (9, 'Picked Wife - The Future Hunter', 7),  (10, 'Village', 7); |
| INSERT INTO Report VALUES -- bookID, reviewerID, rating, postedDate  (7, 11, 5, '2002-12-04'),  (8, 15, 4, '2002-12-04'),  (9, 15, 5, '2002-12-04'),  (1, 17, 3, '2002-12-04'),  (10, 15, 5, '2002-12-04'),  (5, 15, 2, '2002-12-04'),  (8, 16, 4, '2002-12-04'),  (9, 16, 3, '2002-12-04'),  (9, 11, 2, '2002-12-04'); |
| -- See table after inserting value  SELECT \* FROM Author;  SELECT \* FROM Reviewer;  SELECT \* FROM Book;  SELECT \* FROM Report; |

Screenshots:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. SQL script listing authors with same last name as mine (‘Nguyen’) and making sure there are at least 2 records shown (Screenshot after code)

|  |
| --- |
| -- v. List authors whose name contain my family name  SELECT \*  FROM Author  WHERE name LIKE '%Nguyen%'; |

Screenshots:

A screenshot of a computer

Description automatically generated

1. SQL script listing all reviewers’ names and corresponding rating for a given book and making sure there are at least 2 records shown (Screenshot after code)

|  |
| --- |
| -- vi. List reviewer's name and rating for a book  SELECT Reviewer.name AS reviewerName, rating  FROM Report, Reviewer  WHERE Report.reviewerID = Reviewer.reviewerID  and Report.bookID = 9; -- Replace this if wish to see rating for different books |

Screenshots:

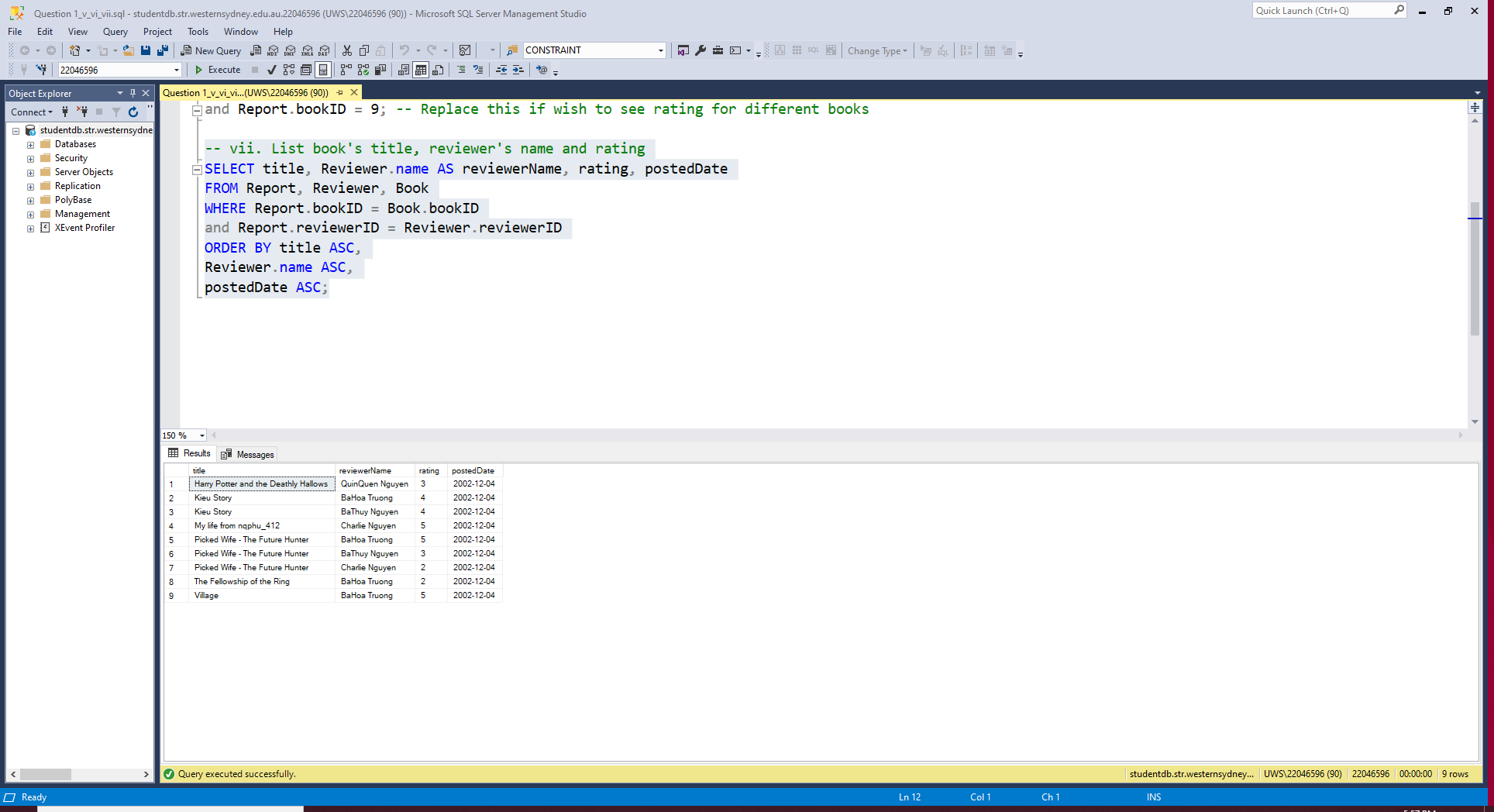
A screenshot of a computer

Description automatically generated

1. SQL script listing all books’ titles, reviewers’ name, rating, and posted date, and sorting output in terms of title, reviewers’ name, completion date alphabetically (Screenshot after code)

|  |
| --- |
| -- vii. List book's title, reviewer's name and rating  SELECT title, Reviewer.name AS reviewerName, rating, postedDate  FROM Report, Reviewer, Book  WHERE Report.bookID = Book.bookID  and Report.reviewerID = Reviewer.reviewerID  ORDER BY title ASC,  Reviewer.name ASC,  postedDate ASC; |

Screenshots:



**Question 2: More on SQL queries**

1. ER Diagram matching with given GRD with fewer attributes and omitting tables that serve as a relationship

A diagram with text and words

Description automatically generated

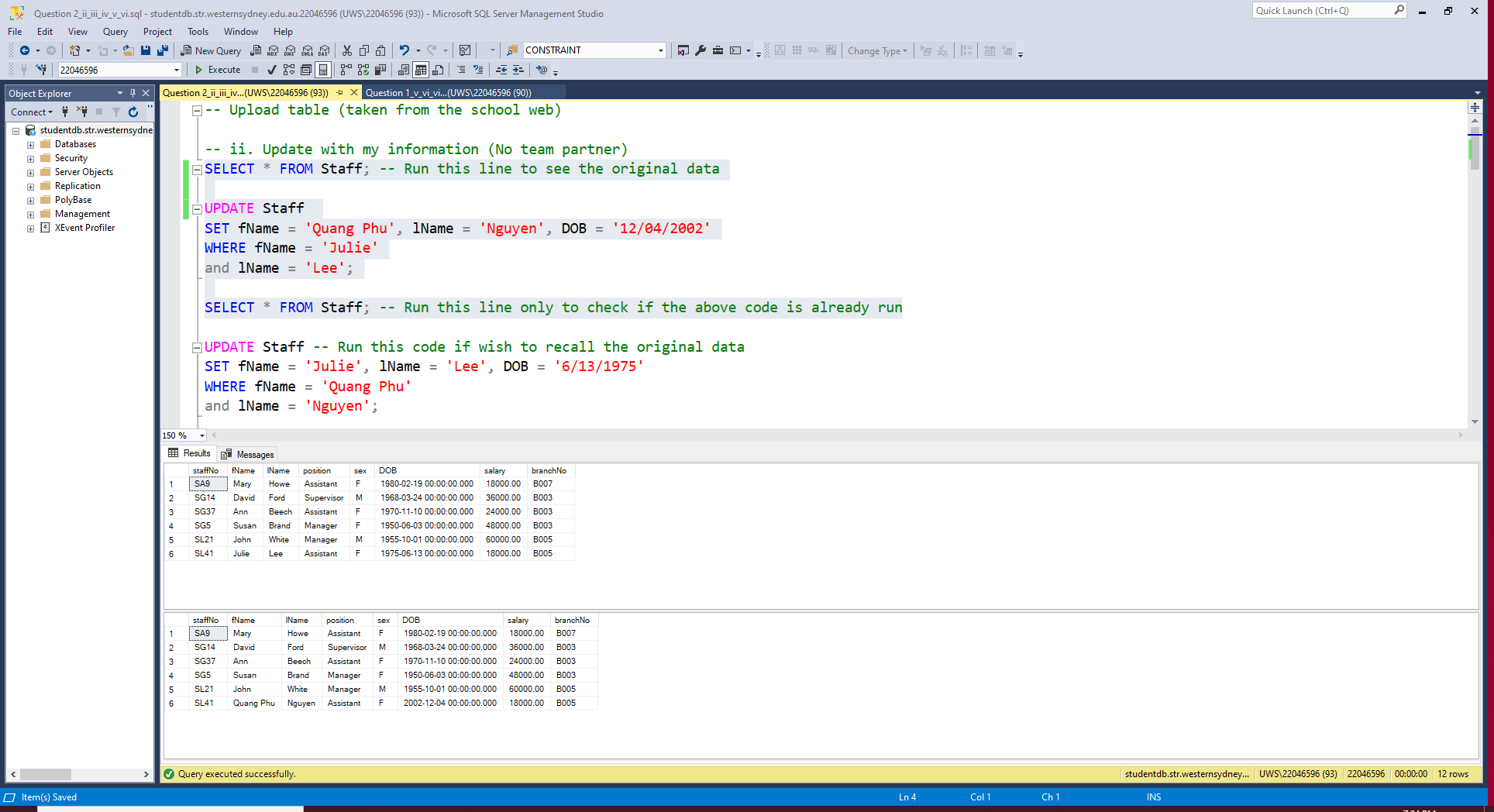
1. SQL script running a given code that has all information, updating staff’s record for staffNo = ‘SL21’ into my information. Note: no teammate (Screenshot after code)

Given code: [Link](https://staff.cdms.westernsydney.edu.au/~zhuhan/current/pracs/dreamhome/dreamhome.txt) to the school web

Update step:

|  |
| --- |
| -- ii. Update with my information (No team partner)  SELECT \* FROM Staff; -- Run this line to see the original data  UPDATE Staff  SET fName = 'Quang Phu', lName = 'Nguyen', DOB = '12/04/2002'  WHERE fName = 'Julie'  and lName = 'Lee';  SELECT \* FROM Staff; -- Run this line to check if the above code is already run |
| UPDATE Staff -- Run this code if wish to recall the original data  SET fName = 'Julie', lName = 'Lee', DOB = '6/13/1975'  WHERE fName = 'Quang Phu'  and lName = 'Nguyen'; |

Screenshot:



1. SQL script listing name, position, gender, and salary of staff earning at least 25000 per annum (Screenshot after code)

|  |
| --- |
| -- iii. Name, position, gender, annual salary for those who earn 25000 per annum  SELECT fName, lName, position, sex, salary  FROM Staff  WHERE salary > 25000; |

Screenshot:

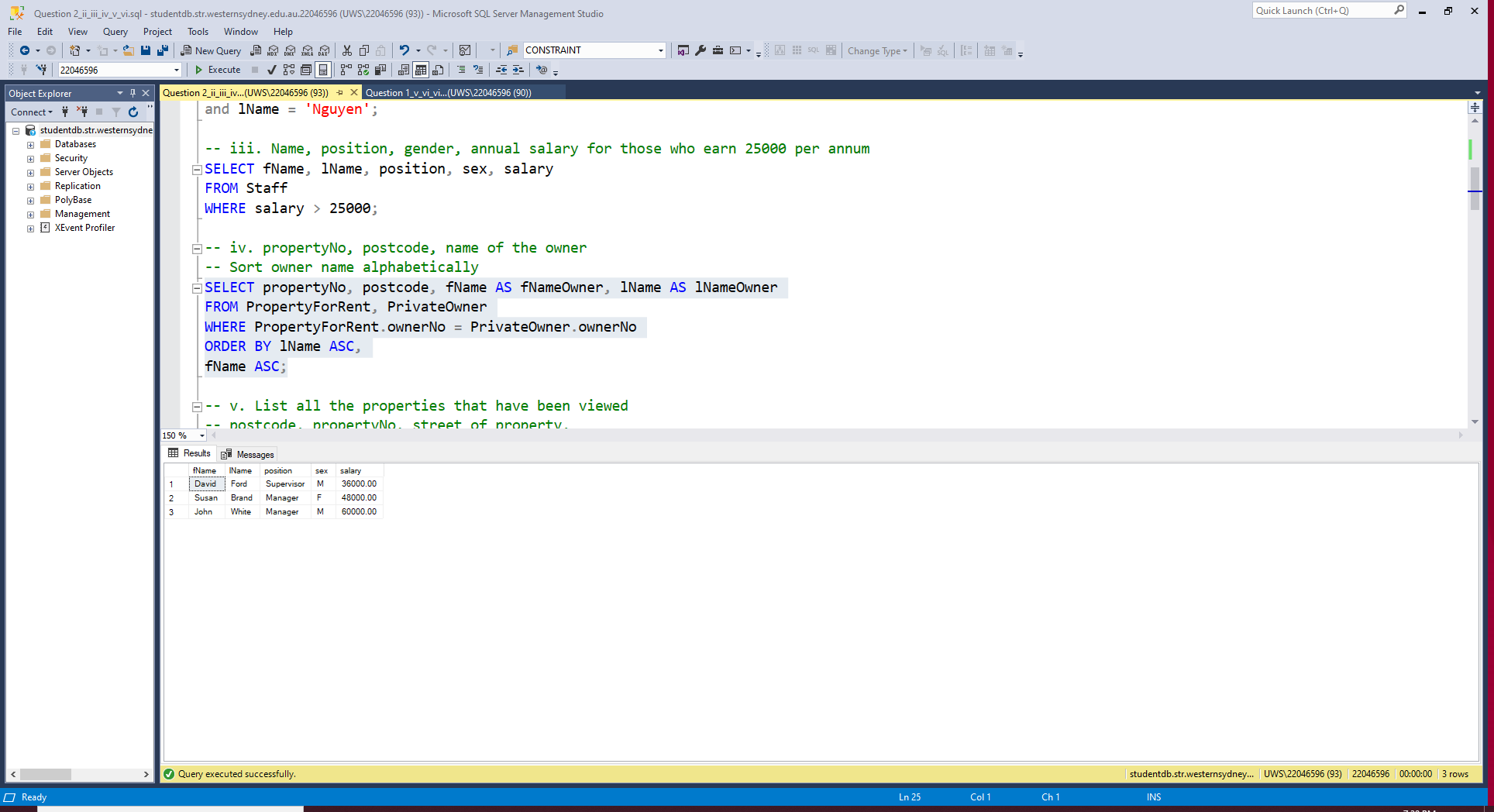
A screenshot of a computer

Description automatically generated

1. SQL script listing propertyNo, postcode of the property and the corresponding owner with owner name alphabetical sorting (Screenshot after code)

|  |
| --- |
| -- iv. propertyNo, postcode, name of the owner  -- Sort owner name alphabetically  SELECT propertyNo, postcode, fName AS fNameOwner, lName AS lNameOwner  FROM PropertyForRent, PrivateOwner  WHERE PropertyForRent.ownerNo = PrivateOwner.ownerNo  ORDER BY lName ASC,  fName ASC; |

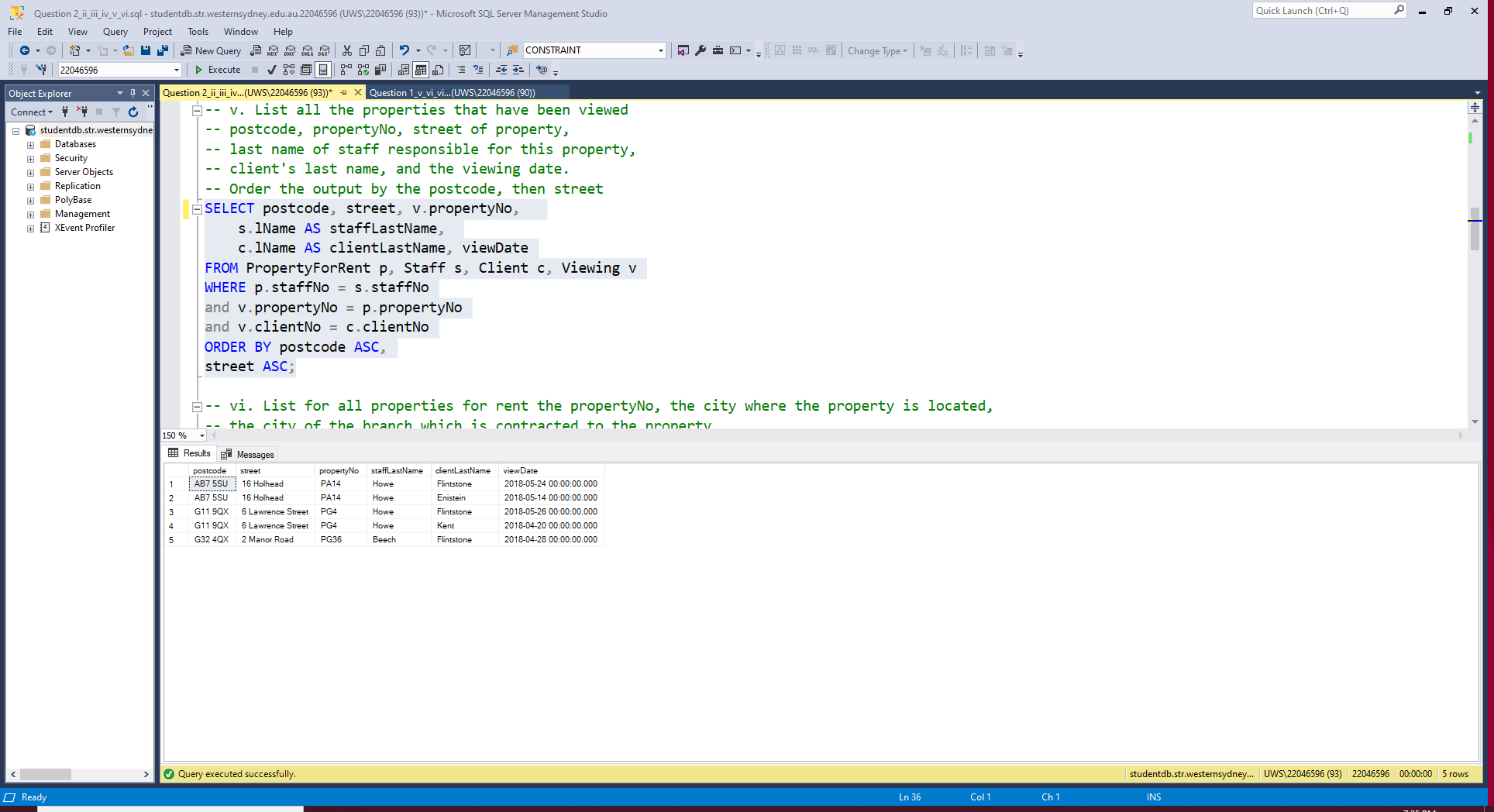
Screenshot:



1. SQL script listing all property being viewed with their postcode, propertyNo, property’s street, staff’s last name, clients’ last names, and viewing date. Ordering the output by postcode then street (Screenshot after code)

|  |
| --- |
| -- v. List all the properties that have been viewed  -- postcode, propertyNo, street of property,  -- last name of staff responsible for this property,  -- client's last name, and the viewing date.  -- Order the output by the postcode, then street  SELECT postcode, street, v.propertyNo,  s.lName AS staffLastName,  c.lName AS clientLastName, viewDate  FROM PropertyForRent p, Staff s, Client c, Viewing v  WHERE p.staffNo = s.staffNo  and v.propertyNo = p.propertyNo  and v.clientNo = c.clientNo  ORDER BY postcode ASC,  street ASC; |

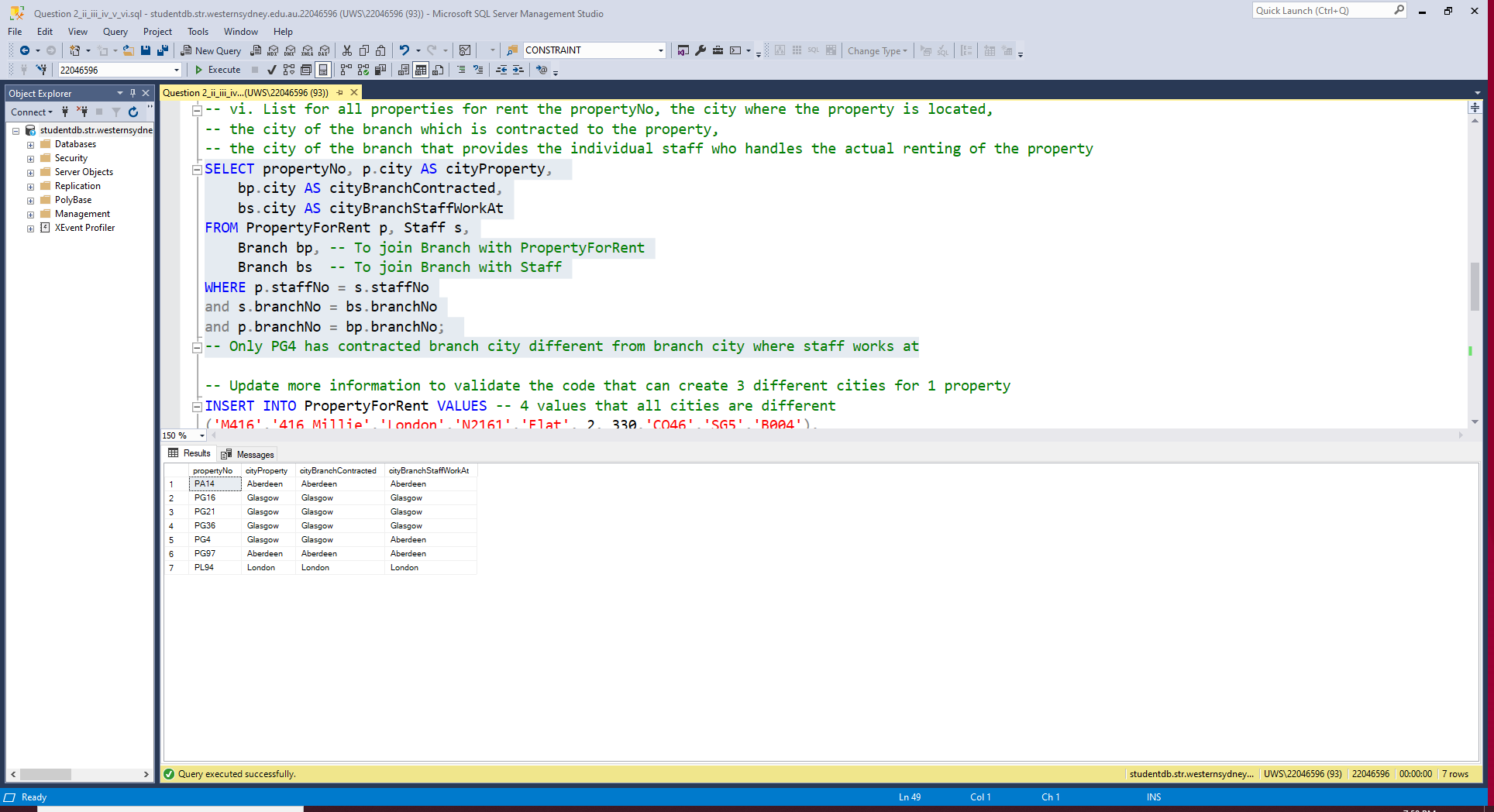
Screenshot:



1. SQL script listing propertyNo, (first) city where the property is located, (second) city of the branch contracted, and (third) city of the branch where the in-charge staff comes from. Moreover, insert more records that could give totally different 3 cities in the query (Screenshot after code)

|  |
| --- |
| -- vi. List for all properties for rent the propertyNo, the city where the property is located,  -- the city of the branch which is contracted to the property,  -- the city of the branch that provides the individual staff who handles the actual renting of the property  SELECT propertyNo, p.city AS cityProperty,  bp.city AS cityBranchContracted,  bs.city AS cityBranchStaffWorkAt  FROM PropertyForRent p, Staff s,  Branch bp, -- To join Branch with PropertyForRent  Branch bs -- To join Branch with Staff  WHERE p.staffNo = s.staffNo  and s.branchNo = bs.branchNo  and p.branchNo = bp.branchNo;  -- Only PG4 has contracted branch city different from branch city where staff works at (First screenshot) |
| -- Update more information to validate the code that can create 3 different cities for 1 property  INSERT INTO PropertyForRent VALUES -- 4 values that all cities are different  ('M416','416 Millie','London','N2161','Flat', 2, 330,'CO46','SG5','B004'),  ('M16','16 Military','Aberdeen','QP2002','Ensuite', 1, 570,'CO40','SL21','B003'),  ('P37A','37A Parramatta','Aberdeen','TP 1123','House', 5, 880,'CO87','SL21','B004'),  ('HE79','79 Hello','Bristol','TNH 65','Apartment', 3, 580,'CO93','SL41','B007'); |
| -- Run the Query (vi) above again to see the difference (Second screenshot)  SELECT propertyNo, p.city AS cityProperty,  bp.city AS cityBranchContracted,  bs.city AS cityBranchStaffWorkAt  FROM PropertyForRent p, Staff s,  Branch bp, -- To join Branch with PropertyForRent  Branch bs -- To join Branch with Staff  WHERE p.staffNo = s.staffNo  and s.branchNo = bs.branchNo  and p.branchNo = bp.branchNo; |
| -- Delete additional values after using  DELETE FROM PropertyForRent  WHERE propertyNo IN ('M416','M16','P37A','HE79'); |

Screenshot:



A screenshot of a computer

Description automatically generated

(4 new records/4 first rows have 3 different cities shown)

**Question 3: Database modelling – case study**

1. ER Diagram with minimal database design. Identifying keys. Briefly explain the roles played by each entity type and relationship type in your design in terms of the design goals.

ER Diagram:

A screenshot of a computer

Description automatically generated

Entity: 5 main entities but due to many-to-many relation between herb and prescription, table prescriptionHerb is created.

* Table medic (Strong entity): doctors / healthcare professionals who can diagnose patient and issue prescription
  + Primary key: medicID
  + Other attributes: fullName
* Table patients (Strong entity): individuals who receive medical attention
  + Primary key: patientID
  + Other attributes: fullName
* Table herb (Strong entity): herbal medicines that are given to patients based on prescription
  + Primary key: herbID
  + Other attributes: herbName, unit, price
* Table diagnosis (Strong entity): medical diagnoses performed by medic and is given to patients
  + Primary key: diagnosisID
  + Foreign keys: medicID references from medic, patientID references from patient
  + Other attributes: diagnosis\_description, diagnosisDate
* Table prescription (Strong entity): medication orders issued by medics, that show which herbal medicines should the patients use
  + Primary key: prescriptionID
  + Foreign keys: medicID references from medic, patientID references from patient
  + Other attributes: prescriptioncheckedDate
* Table prescriptionHerb (Weak entity): data shows which and how many herbal medicines are used in each prescription
  + Primary keys: prescriptionID references from prescription, herbID references from herb
  + Other attributes: quantity

Relationship:

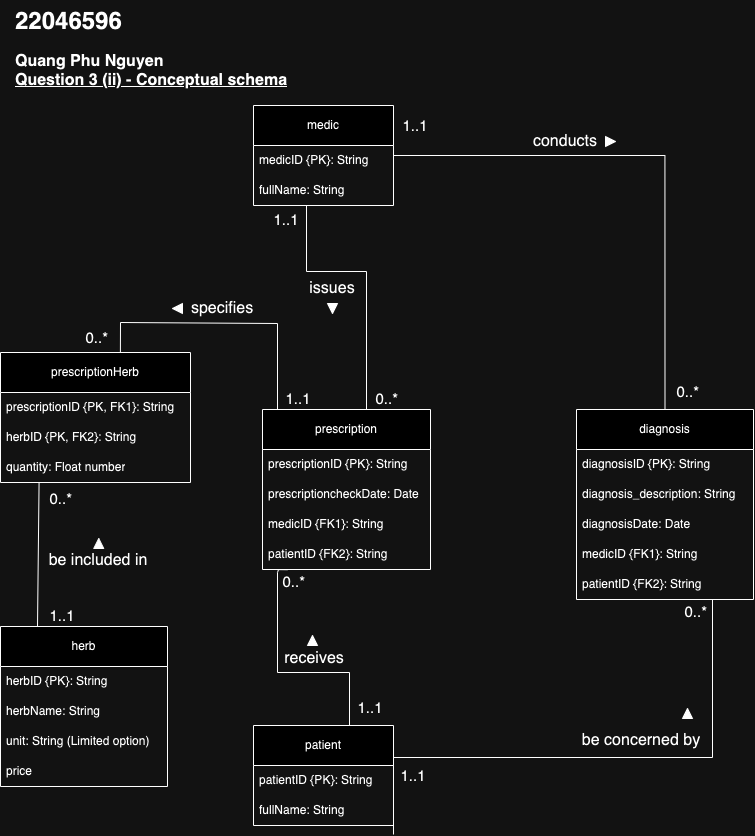
* Medic conducts diagnosis: One-to-Many type
  + A medic conducts 0 or many diagnoses.
  + A diagnosis is performed by 1 medic.
* Diagnosis is concerned by patient: One-to-Many type
  + A patient goes through 0 or many diagnoses.
  + A diagnosis record is matched with 1 patient.
* Prescription includes herbal medicines: Many-to-Many type (prescriptionHerb created)
  + A prescription includes at least 1 herbal medicine.
  + Herbal medicine can be included in 0 or many prescriptions.
* Medic issues prescription: One-to-Many type
  + A prescription is issued by 1 medic.
  + A medic issues 0 or many prescriptions.
* Patient receives prescription: One-to-Many type
  + A prescription is issued for 1 patient.
  + A patient receives 0 or many prescriptions.

1. List the schemas and indicate the respective keys where applicable

Internal schema (Creating table):

|  |
| --- |
| -- Drop table if wish to reset  -- DROP TABLE prescriptionHerb;  -- DROP TABLE diagnosis, prescription;  -- DROP TABLE medic, patient, herb; |
| -- Create structure for 6 tables with key definition  CREATE TABLE medic (  medicID varchar(10) PRIMARY KEY,  fullName nvarchar(50) NOT NULL  ); |
| CREATE TABLE patient (  patientID varchar(10) PRIMARY KEY,  fullName nvarchar(50) NOT NULL  ); |
| CREATE TABLE herb (  herbID varchar(10) PRIMARY KEY,  herbName nvarchar(100) NOT NULL,  unit varchar(10) NOT NULL, -- Constraint below that only allow a few inputs  price numeric(5,2) NOT NULL,  CONSTRAINT herb\_unit CHECK (unit IN ('capsule', 'gram', 'kilogram', 'piece', 'bunch', 'box', 'bottle'))  ); |
| CREATE TABLE diagnosis (  diagnosisID varchar(10) PRIMARY KEY,  diagnosis\_description varchar(200) NOT NULL,  diagnosisDate DATE NOT NULL,  medicID varchar(10) FOREIGN KEY REFERENCES medic,  patientID varchar(10) FOREIGN KEY REFERENCES patient  ); |
| CREATE TABLE prescription (  prescriptionID varchar(10) PRIMARY KEY,  prescriptioncheckDate DATE NOT NULL,  medicID varchar(10) FOREIGN KEY REFERENCES medic,  patientID varchar(10) FOREIGN KEY REFERENCES patient  ); |
| CREATE TABLE prescriptionHerb (  prescriptionID varchar(10) FOREIGN KEY REFERENCES prescription,  herbID varchar(10) FOREIGN KEY REFERENCES herb,  quantity numeric(5,2) NOT NULL, -- not integer because using kilogram and gram  CONSTRAINT pre\_herb\_pk PRIMARY KEY (prescriptionID, herbID)  ); |
| -- To see the (blank) table structure  SELECT \* FROM medic;  SELECT \* FROM patient;  SELECT \* FROM herb;  SELECT \* FROM diagnosis;  SELECT \* FROM prescription;  SELECT \* FROM prescriptionHerb; |

Conceptual schema:



External schema: Queries in following question (iii and v) could be schemas of some users.

Additional: Inserting data

|  |
| --- |
| -- Insert table  INSERT INTO medic VALUES -- ID, name  ('M01', 'Ai Nu Pham'),  ('M02', 'David Juju'),  ('M03', 'Ari Nguyen'),  ('M04', 'Namnam Nguyen'),  ('M05', 'Tuyet Sam Tran'),  ('M06', 'Swuzz Ba'); |
| INSERT INTO patient VALUES -- ID, name  ('P01', 'BaThuy Nguyen'),  ('P02', 'QuinQuen Nguyen'),  ('P03', 'BaHoa Truong'),  ('P04', 'Minki Kim'),  ('P05', 'GeonHee Oh'),  ('P06', 'Elio Nguyen'),  ('P07', 'Sakura Miyawaki'),  ('P08', 'Francis Nguyen'),  ('P09', 'Lucy Hy'),  ('P10', 'Robbie Hilder'); |
| INSERT INTO herb VALUES -- ID, name, unit, price  ('H001', 'Echinacea', 'capsule', 15.00),  ('H012', 'Ginseng', 'gram', 30.00),  ('H014', 'Lavender', 'bunch', 10.00),  ('H016', 'Peppermint', 'bottle', 25.00),  ('H020', 'Ginger', 'piece', 5.00),  ('H028', 'Chamomile', 'box', 20.00),  ('H032', 'Turmeric', 'gram', 12.00),  ('H049', 'Valerian Root', 'capsule', 18.00); |
| INSERT INTO diagnosis VALUES -- ID, description, date, medicID, patientID  ('D001', 'Flu and fever', '2024-07-01', 'M03', 'P02'),  ('D003', 'Tired', '2024-07-02', 'M04', 'P03'),  ('D005', 'Narcissistic Personality Disorder', '2024-08-03', 'M06', 'P06'),  ('D006', 'Depression and anxiety', '2024-08-04', 'M05', 'P07'),  ('D009', 'Diarrhea', '2024-09-02', 'M02', 'P01'),  ('D011', 'Allergic reaction', '2024-09-04', 'M06', 'P08'),  ('D014', 'Flu, fever and headaches', '2024-09-05', 'M03', 'P02'),  ('D027', 'Lack of sleep', '2024-09-06', 'M01', 'P08'); |
| INSERT INTO prescription VALUES -- prescriptionID, date, medicID, patientID  ('PCT001', '2024-07-03', 'M03', 'P02'),  ('PCT002', '2024-07-03', 'M04', 'P03'),  ('PCT003', '2024-08-04', 'M06', 'P06'),  ('PCT004', '2024-08-04', 'M05', 'P07'),  ('PCT005', '2024-09-05', 'M02', 'P01'),  ('PCT006', '2024-09-05', 'M06', 'P08'),  ('PCT007', '2024-09-06', 'M03', 'P02'),  ('PCT008', '2024-09-06', 'M01', 'P08'); |
| INSERT INTO prescriptionHerb VALUES -- prescriptionID, herbID, quantity  ('PCT001', 'H001', 2),  ('PCT001', 'H016', 1),  ('PCT002', 'H012', 3),  ('PCT002', 'H049', 1),  ('PCT003', 'H028', 2),  ('PCT003', 'H014', 1),  ('PCT004', 'H014', 2),  ('PCT004', 'H032', 1),  ('PCT004', 'H049', 1),  ('PCT005', 'H016', 3),  ('PCT005', 'H020', 2),  ('PCT006', 'H012', 1),  ('PCT007', 'H012', 2),  ('PCT007', 'H014', 1),  ('PCT008', 'H028', 3),  ('PCT008', 'H049', 2); |
| -- See the table  SELECT \* FROM medic;  SELECT \* FROM patient;  SELECT \* FROM herb;  SELECT \* FROM diagnosis;  SELECT \* FROM prescription;  SELECT \* FROM prescriptionHerb; |

Screenshot (Tables with data inserted):

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

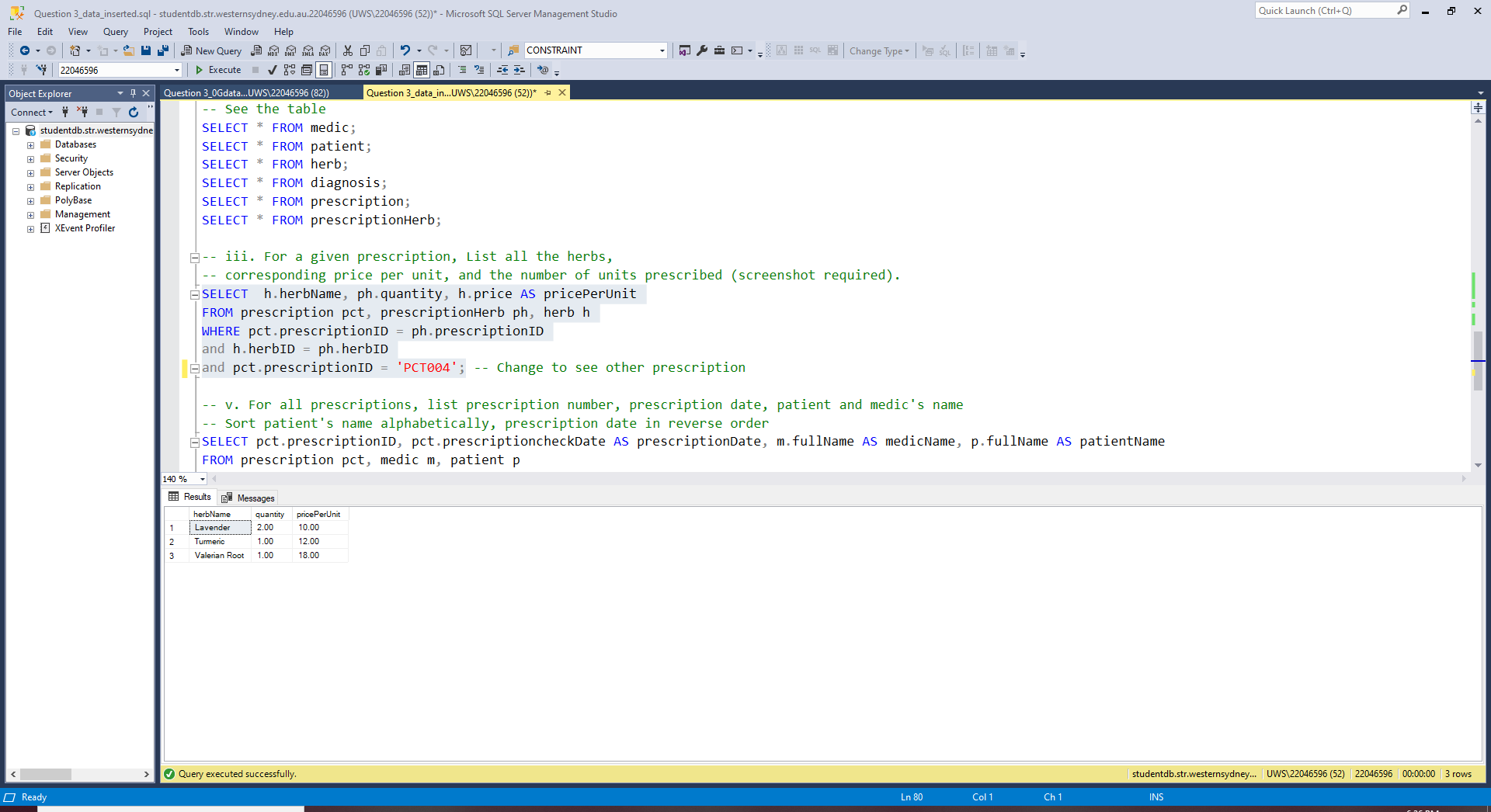
A screenshot of a computer

Description automatically generated

1. SQL script listing all herbs, price per unit, quantity with a given prescription (Screenshot after code)

|  |
| --- |
| -- iii. For a given prescription, List all the herbs,  -- corresponding price per unit, and the number of units prescribed  SELECT h.herbName, ph.quantity, h.price AS pricePerUnit  FROM prescription pct, prescriptionHerb ph, herb h  WHERE pct.prescriptionID = ph.prescriptionID  and h.herbID = ph.herbID  and pct.prescriptionID = 'PCT004'; -- Change to see other prescription |

Screenshot:



1. Rewrite the query (iii) in the form of relational algebra (Using prescriptionID 'PCT004')

**Step 1**: PCT004 is a table showing Cartesian product between 2 tables whose values are filtered.

* table will only have 1 row and all columns from table prescription because prescriptionID here is primary key
* table have multiple (let say 3) rows and all columns from table prescriptionHerb
* Therefore, after Cartesian product, PCT004 table will have 3 rows and all columns

**Step 2**: Again, using Cartesian product for all PCT004 table and herb table into Summary table

**Step 3**: Using selection algebra to filter where herbID is matching in Summary table

**Step 4**: Finally, projection (choosing) only some features (columns) that we want to see

**In summary**, alias of 3 tables: prescription (pct), prescriptionHerb (ph), and herb (h), we have

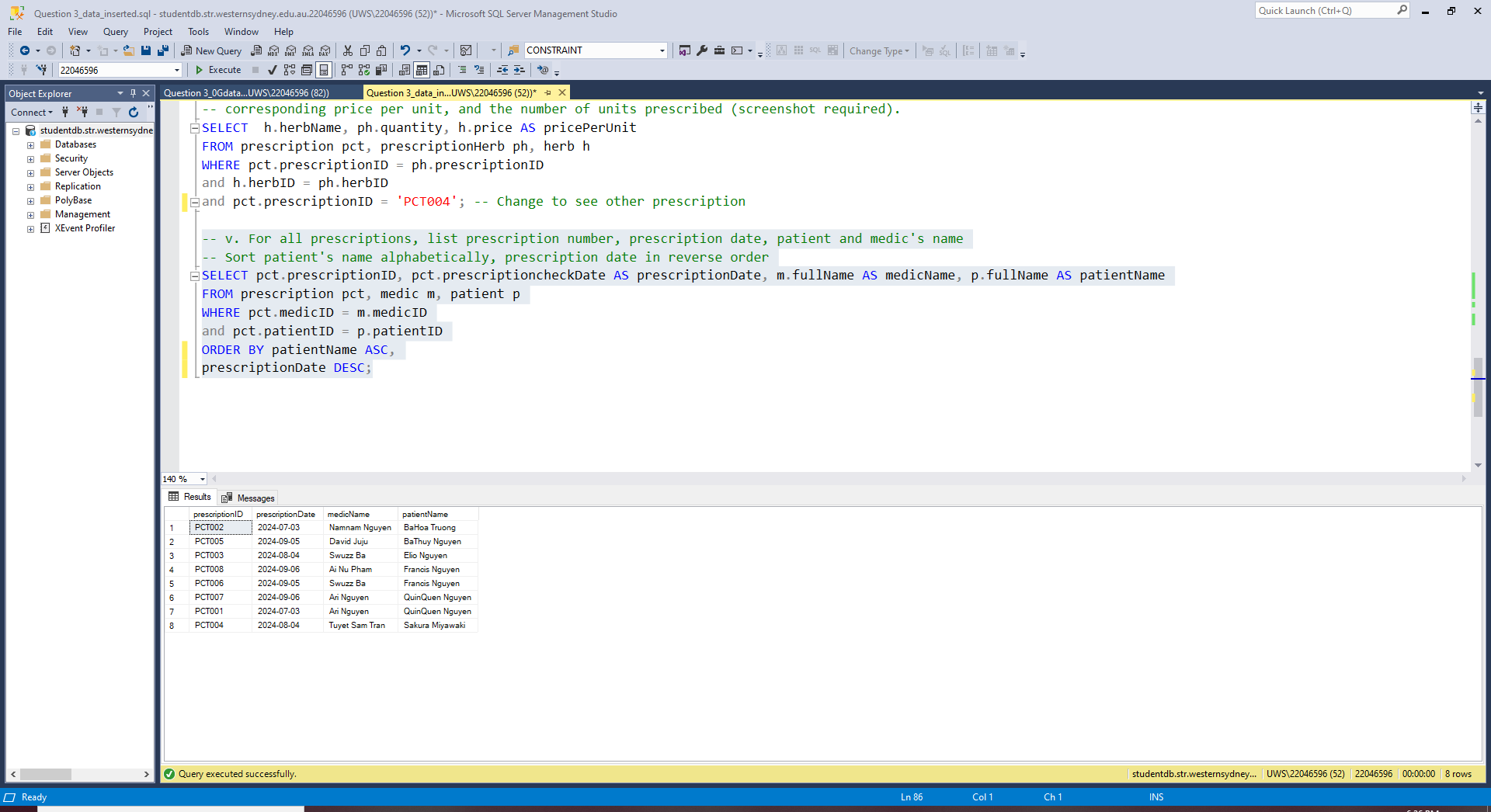
OR

WITH

1. SQL script for all prescriptions, listing patient name, prescription date, prescription number, and corresponding doctors’ name. Sorting the output alphabetically in terms of patient name, but prescription date in the reverse order (Screenshot after code)

|  |
| --- |
| -- v. For all prescriptions, list prescription number, prescription date, patient and medic's name  -- Sort patient's name alphabetically, prescription date in reverse order  SELECT pct.prescriptionID, pct.prescriptioncheckDate AS prescriptionDate, m.fullName AS medicName, p.fullName AS patientName  FROM prescription pct, medic m, patient p  WHERE pct.medicID = m.medicID  and pct.patientID = p.patientID  ORDER BY patientName ASC,  prescriptionDate DESC; |

Screenshot:

****

**Question 4: Selected Additional Exercises**

1. Question from Practical 5: Describe briefly a database application case of your choice and represent your data in terms of 3 linked (via foreign key/s) tables. List their respective primary key and foreign key/s if any.

Scenario: Database of students that joining school clubs.

* + 1 club has at least 10 members
  + 1 student can join many clubs

Tables:

|  |  |  |
| --- | --- | --- |
| **Club** | **Student** | **Role** |
| * Purpose: storing information of all student clubs in the university * Primary key: clubID * Columns:   - clubID (Primary Key)  - name  - description  - establishedDate | * Purpose: storing information of all students studying at Western Sydney University * Primary key: studentID * Columns:   - studentID (Primary Key)  - name  - major  - DOB | * Purpose: storing information of students' role within a club for all students who engage in * Primary key: combination of clubID and studentID * Foreign keys: clubID, studentID * Columns:   - clubID (Primary Key, Foreign Key)  - studentID (Primary Key, Foreign Key)  - role  - joinDate  - activeStatus |

Relationship:

* 1 club can have many students, at least 10
* 1 student can join many clubs, at least 1 (if student doesn’t join any club, he/she is not recorded in the database)
* Many-to-many relationship so that's why the table role is created

ER Diagram:



Global Relation Diagram:

A screenshot of a computer

Description automatically generated

1. Question from Practical 6: Describe the relation that is produced by the following relational algebra operation. We note that "∨" is the mathematical notation for the logical OR operator, while "∧" is for the AND operator.

Answer:

Working from inside to outside:

It means we choose all lines from table Room that have 'Family' and 'Suite' in the column type.

It means creating a new table by letting table Hotel theta join with filter table Room, which has Cartesian product between 2 tables and then filtering the rows where values in two hotelNo columns are the same.

It means only showing all rows but only 4 columns, including city, hotelName, type, and price, from the table created in the previous test.

1. Question from Practical 7: What is the typical output or artefact generated at each of the 3 phases of database design? For the database case in the Additional Exercises of the practical 4, Book/User/Borrows trio, show the output/artefact (e.g. ERD, GRD, SQL) for all the 3 design phases.

Conceptual database design phase creates (Enhanced) Entity-Relation Diagram (E-ERD) and data dictionary.



Logical database design phase creates Global Relation Diagram (GRD).



Physical database design phase creates tables (by using SSMS, MySQL for example via command CREATE TABLE) with detailed information and constraints.

|  |
| --- |
| CREATE TABLE User (  userID varchar(10) PRIMARY KEY,  userName varchar(50) NOT NULL  ); |
| CREATE TABLE Book (  bookID varchar(10) PRIMARY KEY,  title varchar(100) NOT NULL,  dateBorrow DATE, -- NULL when no one borrow it  dateReturn DATE, -- NULL when no one borrow it  userID varchar(10),  CONSTRAINT book\_borrower FOREIGN KEY (userID) REFERENCES User  ); |
| -- See the (blank) table  SELECT \* FROM User;  SELECT \* FROM Book; |