PROJECT REPORT

Project Description:

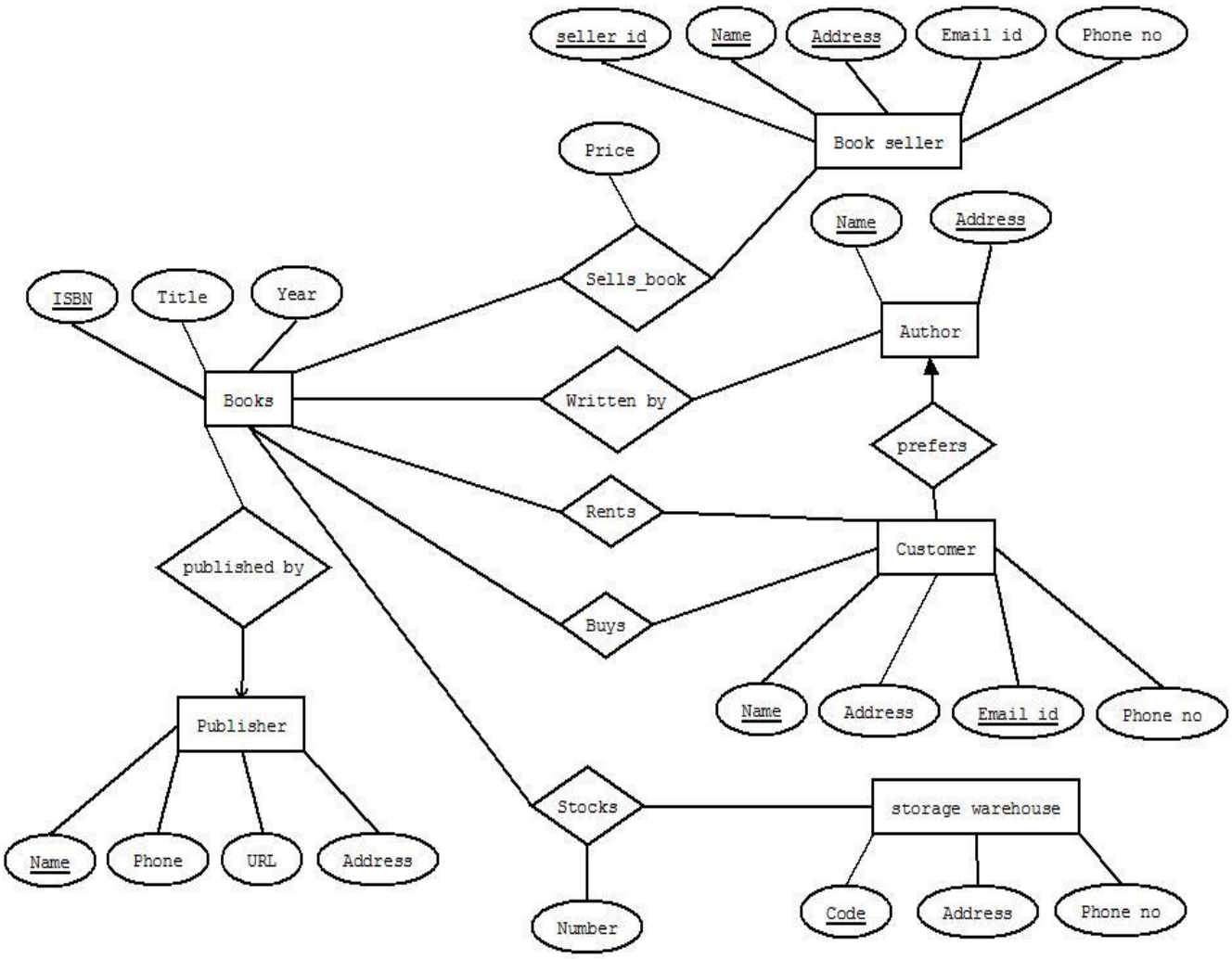
This project is about creating an inventory for the Book store. The project deals about having a Database System for the Books where the users can rent them or buy them. This database maintains information about all the books and their publishers along with the information about their customers and the book dealers/sellers. This also maintains the stock information about the books in various book warehouses. Here the various dealers/sellers have various prices for the books and the customers can buy them with which ever seller they want to.

Project Functionalities:

A Few of the functionalities that the user can use in this project are:

- The user will be able to create new records.
- The user will be able to retrieve the records and view/verify them.
- In case of any changes the user will be able to update the records and save them.
- Each record will have a unique key value so that each record will be unique so that no two records will be the same.
- If the record is obsolete and is no longer required the user will be able to delete that record.
- Here the user corresponds to the owner of the book store.
- The project also has foreign keys and constraints such that data cannot be incorrect at any given point of time.

The E/R diagram is as given below:



Conversion of the E/R diagram into the equivalent tables:

Books (ISBN, title, Pub_year)

Publisher (Name, phone_no, URL, Address)

Book_seller (Seller_id, Name, Address, email_id, phone_no)

Author (Name, Address)

Customer (Name, Address, email_id, phone_no)

Storage_warehouse (Code, Address, phone_no)

Published_by (ISBN, Name)

Sells_book (Seller_id, Name, Address, ISBN, price)

Written_by (Name, Address, ISBN)

Rents (Name, email_id, ISBN)

Buys (Name, email_id, ISBN)

Stocks (ISBN, code, Numbers)

Prefers (customer_Name, email_id, Author_name, Address)

The database used in here is PSQL:

The data manipulation commands are as shown below:

SQL SCHEMA (Dropping and Creating Tables)

```
--To drop the tables that had been created earlier
DROP TABLE Books;
DROP TABLE Publisher;
DROP TABLE Book_seller;
DROP TABLE Author;
DROP TABLE Customer;
DROP TABLE Storage_warehouse;
DROP TABLE Published_by;
DROP TABLE Sells_book;
DROP TABLE Written_by;
DROP TABLE Rents;
DROP TABLE Buys;
DROP TABLE Stocks;
DROP TABLE Prefers;

--Create table, Books
CREATE TABLE Books (
  ISBN  VARCHAR(20),
```

```
  title  VARCHAR(50),
  Pub_year VARCHAR(20),
  PRIMARY KEY(ISBN)
 );

--Create table, Publisher
CREATE TABLE Publisher (
  Name  VARCHAR(20),
  phone_no  VARCHAR(20),
  URL VARCHAR(20),
  Address VARCHAR(20),
  PRIMARY KEY(Name)
 );

--Create table, Book_seller
CREATE TABLE Book_seller (
  Seller_id  VARCHAR(20),
  Name  VARCHAR(20),
  Address VARCHAR(20),
  email_id VARCHAR(20),
  phone_no  VARCHAR(20),
  PRIMARY KEY(Seller_id,Name,Address)
 );

--Create table, Author
CREATE TABLE Author (
  Name  VARCHAR(20),
```

```
  Address VARCHAR(20),
  PRIMARY KEY(Name, Address)
 );

--Create table, Customer
CREATE TABLE Customer (
  Name  VARCHAR(20),
  Address VARCHAR(20),
  email_id VARCHAR(20),
  phone_no  VARCHAR(20),
  PRIMARY KEY(Name, email_id)
 );

--Create table, Storage_warehouse
CREATE TABLE Storage_warehouse (
  Code  VARCHAR(20),
  Address VARCHAR(20),
  phone_no  VARCHAR(20),
  PRIMARY KEY(Code)
 );

--Create table, Published_by
CREATE TABLE Published_by (
  ISBN  VARCHAR(20),
  Name VARCHAR(20),
  PRIMARY KEY(Name, ISBN)
 );
```

```
--Create table, Sells_book
CREATE TABLE Sells_book (
  Seller_id VARCHAR(20),
  Name  VARCHAR(20),
  Address VARCHAR(20),
  ISBN  VARCHAR(20),
  price REAL,
  PRIMARY KEY(Seller_id, Name, Address, ISBN)
 );

--Create table, Written_by
CREATE TABLE Written_by (
  Name  VARCHAR(20),
  Address VARCHAR(20),
  ISBN  VARCHAR(20),
  PRIMARY KEY(Name, Address, ISBN)
 );

--Create table, Rents
CREATE TABLE Rents (
  Name  VARCHAR(20),
  email_id VARCHAR(20),
  ISBN  VARCHAR(20),
  PRIMARY KEY(Name, email_id, ISBN)
 );
```

```
--Create table, Buys
CREATE TABLE Buys (
  Name  VARCHAR(20),
  email_id VARCHAR(20),
  ISBN  VARCHAR(20),
  PRIMARY KEY(Name, email_id, ISBN)
 );

--Create table, Stocks
CREATE TABLE Stocks (
  ISBN VARCHAR(20),
  Code  VARCHAR(20),
  Numbers VARCHAR(20),
  PRIMARY KEY(ISBN, Code)
 );

--Create table, Prefers
CREATE TABLE Prefers (
  customer_Name  VARCHAR(20),
  email_id VARCHAR(20),
  Author_name  VARCHAR(20),
  Address VARCHAR(20),
  PRIMARY KEY(customer_Name, email_id, Author_name, Address)
 );

--Inserting data into tables
--Inserting data into Books table
```

```
INSERT INTO Books (ISBN, title, Pub_year) VALUES (9781133187790, 'Theory of Computation', 2010);

INSERT INTO Books (ISBN, title, Pub_year) VALUES (9845646465732, 'Database Systems', 2009);

INSERT INTO Books (ISBN, title, Pub_year) VALUES (9781165498486, 'Analysis of Algorithms', 2004);

INSERT INTO Books (ISBN, title, Pub_year) VALUES (9765465654654, 'Computer Networks', 2013);

INSERT INTO Books (ISBN, title, Pub_year) VALUES (9831643216348, 'Bioinformatics', 2010);


--Inserting data into Publisher table

INSERT INTO Publisher (Name, phone_no, URL, Address) VALUES ('McGrahill', '365-789-5268', 'McGrahill.com', '2135, jonesboro');

INSERT INTO Publisher (Name, phone_no, URL, Address) VALUES ('Tata', '654-654-3489', 'tata.com', '8395, little-rock');

INSERT INTO Publisher (Name, phone_no, URL, Address) VALUES ('Adiban', '597-126-5412', 'Adiban.com', '8496, little-rock');

INSERT INTO Publisher (Name, phone_no, URL, Address) VALUES ('Hilltop', '845-828-6534', 'hilltop.com', '9879, jonesboro');


--Inserting data into Book_seller table

INSERT INTO Book_seller (Seller_id, Name, Address, email_id, phone_no)

VALUES ('6985412', 'Higgin-Bothms', '8745, washington', 'bothms@higgin.com', '125-965-8532');

INSERT INTO Book_seller (Seller_id, Name, Address, email_id, phone_no)

VALUES ('9876249', 'morebooks', '9559, jonesboro', 'more@book.com', '698-249-0965');

INSERT INTO Book_seller (Seller_id, Name, Address, email_id, phone_no)

VALUES ('6873154', 'text-book brokers', '2106, jonesboro', 'textbook@brokers.com', '870-935-2325');


--Inserting data into Author table
```

```
INSERT INTO Author (Name, Address) VALUES ('tony-gaddis','8536, little-rock');
INSERT INTO Author (Name, Address) VALUES ('sipser','9845, fayetteville');
INSERT INTO Author (Name, Address) VALUES ('mark roman','6545,bentenville');
INSERT INTO Author (Name, Address) VALUES ('michael','8956, jonesboro');
INSERT INTO Author (Name, Address) VALUES ('jackman','4452, fayetteville');

--Inserting data into Customer table
INSERT INTO Customer (Name, Address, email_id, phone_no)
VALUES ('Tony', '5268, bridge-port', 'tony@gmail.com', '125-985-1548');
INSERT INTO Customer (Name, Address, email_id, phone_no)
VALUES ('Mark', '6465, Nashville', 'mark@gmail.com', '654-321-9854');
INSERT INTO Customer (Name, Address, email_id, phone_no)
VALUES ('arnold', '9524, california', 'arnold@gmail.com', '445-326-8954');
INSERT INTO Customer (Name, Address, email_id, phone_no)
VALUES ('Natasha', '9875, new-port', 'nat@yahoo.com', '181-845-2652');
INSERT INTO Customer (Name, Address, email_id, phone_no)
VALUES ('Brenda', '8624, Jonesboro', 'brenda@hotmail.com', '126-984-4715');
INSERT INTO Customer (Name, Address, email_id, phone_no)
VALUES ('Billy', '1368, little-rock', 'billy@yahoo.com', '258-384-7426');
INSERT INTO Customer (Name, Address, email_id, phone_no)
VALUES ('Hommer', '2682, jonesboro', 'hommer@hotmail.com', '569-256-7458');

--Inserting data into Storage_warehouse table
INSERT INTO Storage_warehouse (Code, Address, phone_no) VALUES ('store56248', '4578,
Jonesboro', '201-652-8954');
INSERT INTO Storage_warehouse (Code, Address, phone_no) VALUES ('store59526', '8756,
Memphis', '698-856-2415');
```

```
INSERT INTO Storage_warehouse (Code, Address, phone_no) VALUES ('store2685', '1226, Jonesboro', '268-235-7126');

INSERT INTO Storage_warehouse (Code, Address, phone_no) VALUES ('store2384', '9523, little-rock', '238-412-5841');


--Inserting data into Published_by table

INSERT INTO Published_by (ISBN, Name) VALUES (9781133187790, 'McGrahill');

INSERT INTO Published_by (ISBN, Name) VALUES (9845646465732, 'Tata');

INSERT INTO Published_by (ISBN, Name) VALUES (9781165498486, 'Adiban');

INSERT INTO Published_by (ISBN, Name) VALUES (9765465654654, 'Hilltop');

INSERT INTO Published_by (ISBN, Name) VALUES (9831643216348, 'McGrahill');


--Inserting data into Sells_book table

INSERT INTO Sells_book (Seller_id, Name, Address, ISBN, price)

VALUES ('6985412', 'Higgin-Bothms', '8745, washington', '9781133187790', '150.50');

INSERT INTO Sells_book (Seller_id, Name, Address, ISBN, price)

VALUES ('9876249', 'morebooks', '9559, jonesboro', '9845646465732', '100.75');

INSERT INTO Sells_book (Seller_id, Name, Address, ISBN, price)

VALUES ('6873154', 'text-book brokers', '2106, jonesboro', '9831643216348', '200.00');

INSERT INTO Sells_book (Seller_id, Name, Address, ISBN, price)

VALUES ('6873154', 'text-book brokers', '2106, jonesboro', '9765465654654', '175.20');

INSERT INTO Sells_book (Seller_id, Name, Address, ISBN, price)

VALUES ('6985412', 'Higgin-Bothms', '8745, washington', '9781165498486', '75.50');

--Inserting data into Written_by table

INSERT INTO Written_by (Name, Address, ISBN) VALUES ('tony-gaddis','8536, little-rock', 9781133187790);

INSERT INTO Written_by (Name, Address, ISBN) VALUES ('sipser','9845, fayetteville', 9845646465732);
```

```
INSERT INTO Written_by (Name, Address, ISBN) VALUES ('mark roman','6545,bentenville', 9831643216348);

INSERT INTO Written_by (Name, Address, ISBN) VALUES ('michael','8956, jonesboro', 9765465654654);

INSERT INTO Written_by (Name, Address, ISBN) VALUES ('jackman','4452, fayetteville', 9781165498486);


--Inserting data into Rents table

INSERT INTO Rents (Name, email_id, ISBN) VALUES ('Tony', 'tony@gmail.com', 9781133187790);

INSERT INTO Rents (Name, email_id, ISBN) VALUES ('Mark', 'mark@gmail.com', 9845646465732);

INSERT INTO Rents (Name, email_id, ISBN) VALUES ('Mark', 'mark@gmail.com', 9831643216348);

INSERT INTO Rents (Name, email_id, ISBN) VALUES ('Natasha', 'nat@yahoo.com', 9831643216348);

INSERT INTO Rents (Name, email_id, ISBN) VALUES ('Brenda', 'brenda@hotmail.com', 9781165498486);

INSERT INTO Rents (Name, email_id, ISBN) VALUES ('Tony', 'tony@gmail.com', 9781165498486);

INSERT INTO Rents (Name, email_id, ISBN) VALUES ('Hommer', 'hommer@hotmail.com', 9765465654654);


--Inserting data into Buys table

INSERT INTO Buys (Name, email_id, ISBN) VALUES ('Tony', 'tony@gmail.com', 9781133187790);

INSERT INTO Buys (Name, email_id, ISBN) VALUES ('Mark', 'mark@gmail.com', 9831643216348);

INSERT INTO Buys (Name, email_id, ISBN) VALUES ('Mark', 'mark@gmail.com', 9781165498486);

INSERT INTO Buys (Name, email_id, ISBN) VALUES ('Billy', 'billy@yahoo.com', 9781165498486);
```

```
INSERT INTO Buys (Name, email_id, ISBN) VALUES ('Brenda', 'brenda@hotmail.com',
9765465654654);


--Inserting data into Stocks table
INSERT INTO Stocks (ISBN, Code, Numbers) VALUES (9781133187790, 'store56248', '10 nos');
INSERT INTO Stocks (ISBN, Code, Numbers) VALUES (9845646465732, 'store59526', '15 nos');
INSERT INTO Stocks (ISBN, Code, Numbers) VALUES (9781165498486, 'store56248', '5 nos');
INSERT INTO Stocks (ISBN, Code, Numbers) VALUES (9765465654654, 'store2685', '20 nos');
INSERT INTO Stocks (ISBN, Code, Numbers) VALUES (9831643216348, 'store2384', '10 nos');


--Inserting data into Prefers table
INSERT INTO Prefers (customer_Name, email_id, Author_name, Address)
VALUES ('Tony', 'tony@gmail.com', 'tony-gaddis','8536, little-rock' );
INSERT INTO Prefers (customer_Name, email_id, Author_name, Address)
VALUES ('arnold', 'arnold@gmail.com', 'tony-gaddis','8536, little-rock' );
INSERT INTO Prefers (customer_Name, email_id, Author_name, Address)
VALUES ('Brenda', 'brenda@hotmail.com', 'sipser','9845, fayetteville' );
INSERT INTO Prefers (customer_Name, email_id, Author_name, Address)
VALUES ('Natasha', 'nat@yahoo.com', 'jackman','4452, fayetteville' );
INSERT INTO Prefers (customer_Name, email_id, Author_name, Address)
VALUES ('Hommer', '2682, jonesboro', 'michael','8956, jonesboro' );


--Simple Queries SELECT ... FROM ... WHERE ...
-- 1) Find all the books that were published in the year 2010
SELECT title FROM books WHERE pub_year='2010';
```

```
--2) Find all the name of the authors whose names starts with m

SELECT name FROM author WHERE name LIKE 'm%';


--3) Find all the isbns for the books published by mcGrahill

SELECT isbn FROM published_by WHERE name = 'McGrahill';


--4) Find all the books that are sold by the seller whose seller_id is 6873154

SELECT * FROM sells_book WHERE seller_id='6873154';


--5) Find all the isbns that are rented by mark

SELECT * FROM rents WHERE name = 'Mark';

--6) Find all the authors from fayetteville

SELECT name FROM author WHERE address LIKE '%fayetteville';


--7) Find all the customers and their emails who prefers tony-gaddis

SELECT customer_name, email_id FROM prefers WHERE author_name = 'tony-gaddis';


--8) Find the isbns and price of the books that are sold in jonesboro

SELECT isbn, price FROM sells_book WHERE address LIKE '%jonesboro';


--Multi-relational queries, AND, OR commands

--9) Find all the isbn and the codes for the books which has exactly 10 books in stock

SELECT isbn,code FROM stocks WHERE numbers = '10 nos';

--10) Find all the names for the books published by mcGrahill

SELECT books.title FROM published_by, books WHERE published_by.name = 'McGrahill' AND
books.isbn = published_by.isbn;
```

--11) Find the book stocks for the books in store 56248

```
SELECT books.title, stocks.numbers FROM stocks,books WHERE stocks.code='store56248' AND
books.isbn = stocks.isbn;
```

--12) Find the author name of the books published by McGrahill

```
SELECT written_by.name FROM written_by, published_by WHERE
published_by.name='McGrahill'
                    AND published_by.isbn = written_by.isbn;
```

--13) Find the cost of the books that mark buys

```
SELECT sells_book.price FROM sells_book,buys WHERE buys.name='Mark' AND buys.isbn =
sells_book.isbn;
```

--14) Find the books that are sold at Higgin-Bothms

```
SELECT books.title FROM books,sells_book WHERE sells_book.name='Higgin-Bothms' AND
sells_book.isbn = books.isbn;
```

--Subqueries

--15) Find the isbn of the books that are not in buys

```
SELECT isbn FROM books EXCEPT (SELECT isbn FROM buys);
```

--16) Find the isbns of the books whose author is from fayetteville

```
SELECT isbn FROM books WHERE isbn IN (SELECT isbn FROM written_by WHERE address LIKE
'%fayetteville');
```

--17) Find the isbn of books bought by mark using the subquery in from

```
SELECT isbn FROM(SELECT * FROM buys WHERE name = 'Mark')Markbuys;
```

--18) Find the isbns of books which are not in both buys and rents using NOT IN

```
SELECT isbn FROM rents WHERE isbn NOT IN (SELECT isbn FROM buys);


--19) Find the isbns of the books with the highest price in sells_book

SELECT isbn FROM sells_book WHERE price >= ALL (SELECT price FROM sells_book);


--20) Find the name of the seller who sells the book at the lowest price

SELECT name FROM sells_book WHERE price <= ALL (SELECT price FROM sells_book);


--SQL statements using UNION, INTERSECT, DIFFERENCE

--21)Find the isbns of the books that are in both buys and rents

(SELECT isbn FROM buys) INTERSECT (SELECT isbn FROM rents);


--22)Find the isbns of the books in rents that are not in buys

(SELECT isbn FROM rents) EXCEPT (SELECT isbn FROM buys);


--23)Unite the names of both authors and customers

(SELECT name FROM author) UNION (SELECT name FROM customer);


--24)Unite customers and authors from little-rock

(SELECT name FROM author WHERE address LIKE '%little-rock') UNION (SELECT name FROM
customer WHERE address LIKE '%little-rock');



--SQL statements using join

--25) Select the isbns and names from the rents and buys by using natural join

SELECT isbn,name FROM rents NATURAL JOIN buys;
```

--26) Select the isbns and names from the rents and buys by using natural full outer join

```
SELECT isbn,name FROM rents NATURAL FULL OUTER JOIN buys;
```

--27) Join the rents and books by using Natural left outer join

```
SELECT * FROM rents NATURAL LEFT OUTER JOIN books;
```

--28) Join the rents and books by using Natural Right outer join

```
SELECT * FROM rents NATURAL RIGHT OUTER JOIN books;
```

--29) Select the isbns, name from written_by and buys by using theta join

```
SELECT buys.isbn,buys.name FROM written_by JOIN buys ON written_by.isbn = buys.isbn;
```

--30) Select the isbns, name from buys and rents by using cross join

```
SELECT buys.isbn,buys.name FROM buys CROSS JOIN rents;
```

--Aggregate functions

--31) Count the number of records in the column name of customers

```
SELECT COUNT(name) FROM customer;
```

--32) Find the min price of the books sold by the vendors in sells_book

```
SELECT name, MIN(price) FROM sells_book GROUP BY name;
```

--33) Find the max price of the books sold by the vendors in sells_book

```
SELECT name, MAX(price) FROM sells_book GROUP BY name;
```

--34) Find the average price of the books sold by the vendors in sells_book

```
SELECT name, AVG(price) FROM sells_book GROUP BY name;


--35) Find the sum of the price of the books sold by the vendors in sells_book

SELECT name, SUM(price) FROM sells_book GROUP BY name;


--36) Find the price of the books sold by each seller and the price is should by more than 120

SELECT seller_id, price FROM sells_book GROUP BY seller_id, price HAVING price > 120;


--Database Modification

--37) Insert a new book in the table books

INSERT INTO Books (ISBN, title, Pub_year) VALUES (9781654981256, 'Operating systems',
2013);

INSERT INTO Sells_book (Seller_id, Name, Address, ISBN, price)

VALUES ('9876249', 'morebooks', '9559, jonesboro', '9781654981256', '120.05');

INSERT INTO Published_by (ISBN, Name) VALUES (9781654981256, 'Hilltop');


--38) create a table which holds common books from the table rents and buys and insert a row
into it using a subquery

--Drop table common_books if already existing

DROP TABLE common_books;

--Create table,common_books

CREATE TABLE common_books (

  ISBN  VARCHAR(20)

 );

--Inserting via a subquery

INSERT INTO common_books (isbn) (SELECT rents.isbn FROM rents,buys WHERE
rents.isbn=buys.isbn);
```

--39) Delete a tuple from customer

```
DELETE FROM customer WHERE phone_no = '258-384-7426';
```

--40) Delete a tuple from customer using subquery

```
DELETE FROM rents WHERE isbn = (SELECT rents.isbn FROM rents,buys WHERE
rents.name=buys.name AND rents.isbn=buys.isbn AND rents.name='Mark');
```

--41) Update the price in the sells_book table by 10$ whose price is less than 100$

```
UPDATE sells_book SET price = price+10 WHERE price < 100;
```

--42) Update the store code which is in little-rock using a subquery

```
UPDATE storage_warehouse SET code = 'store23845' WHERE code = (SELECT code FROM
storage_warehouse WHERE address LIKE '%little-rock');
```

--creating VIEW

--43)Create a view to display the name of the books, seller whose prices are above 150$

```
CREATE VIEW books_sellers AS

SELECT books.title,sells_book.name FROM books, sells_book WHERE sells_book.price>150 AND
sells_book.isbn=books.isbn;
```

--44)Re-creating tables with constraints

--To drop the tables that had been created earlier and to recreate them with constraints

```
DROP TABLE Books;

DROP TABLE Publisher;

DROP TABLE Book_seller;

DROP TABLE Author;

DROP TABLE Customer;

DROP TABLE Storage_warehouse;
```

```
DROP TABLE Published_by;
DROP TABLE Sells_book;
DROP TABLE Written_by;
DROP TABLE Rents;
DROP TABLE Buys;
DROP TABLE Stocks;
DROP TABLE Prefers;

--Create table, Books
CREATE TABLE Books (
  ISBN  VARCHAR(20),
  title  VARCHAR(50),
  Pub_year VARCHAR(20),
  PRIMARY KEY(ISBN)
 );

--Create table, Publisher
CREATE TABLE Publisher (
  Name  VARCHAR(20),
  phone_no  VARCHAR(20),
  URL VARCHAR(20),
  Address VARCHAR(20),
  PRIMARY KEY(Name)
 );

--Create table, Book_seller
CREATE TABLE Book_seller (
```

```
  Seller_id  VARCHAR(20),
  Name  VARCHAR(20),
  Address VARCHAR(20),
  email_id VARCHAR(20),
  phone_no  VARCHAR(20),
  PRIMARY KEY(Seller_id,Name,Address)
 );

--Create table, Author
CREATE TABLE Author (
  Name  VARCHAR(20),
  Address VARCHAR(20),
  PRIMARY KEY(Name, Address)
 );

--Create table, Customer
CREATE TABLE Customer (
  Name  VARCHAR(20),
  Address VARCHAR(20),
  email_id VARCHAR(20),
  phone_no  VARCHAR(20),
  PRIMARY KEY(Name, email_id)
 );

--Create table, Storage_warehouse
CREATE TABLE Storage_warehouse (
  Code  VARCHAR(20),
```

```
  Address VARCHAR(20),
  phone_no  VARCHAR(20),
  PRIMARY KEY(Code)
 );

--Create table, Published_by
CREATE TABLE Published_by (
  ISBN  VARCHAR(20),
  Name VARCHAR(20),
  PRIMARY KEY(Name, ISBN)
  FOREIGN KEY (ISBN) REFERENCES books (ISBN),
  FOREIGN KEY (Name) REFERENCES publisher (name)
 );

--Create table, Sells_book
CREATE TABLE Sells_book (
  Seller_id VARCHAR(20),
  Name  VARCHAR(20),
  Address VARCHAR(20),
  ISBN  VARCHAR(20),
  price REAL,
  PRIMARY KEY(Seller_id, Name, Address, ISBN),
  FOREIGN KEY (ISBN) REFERENCES books (ISBN),
  FOREIGN KEY (seller_id) REFERENCES Book_seller (seller_id)
 );

--Create table, Written_by
```

```
CREATE TABLE Written_by (
  Name  VARCHAR(20) REFERENCES Author (Name),
  Address VARCHAR(20),
  ISBN  VARCHAR(20) REFERENCES books (ISBN),
  PRIMARY KEY(Name, Address, ISBN)
 );

--Create table, Rents
CREATE TABLE Rents (
  Name  VARCHAR(20)REFERENCES Customer (Name),
  email_id VARCHAR(20),
  ISBN  VARCHAR(20) REFERENCES books (ISBN),
  PRIMARY KEY(Name, email_id, ISBN)
 );

--Create table, Buys
CREATE TABLE Buys (
  Name  VARCHAR(20) REFERENCES Customer (Name),
  email_id VARCHAR(20),
  ISBN  VARCHAR(20) REFERENCES books (ISBN),
  PRIMARY KEY(Name, email_id, ISBN),
 );

--Create table, Stocks
CREATE TABLE Stocks (
  ISBN VARCHAR(20) REFERENCES books (ISBN),
  Code  VARCHAR(20) REFERENCES storage_warehouse (Code),
```

```
  Numbers VARCHAR(20),
  PRIMARY KEY(ISBN, Code)
 );

--Create table, Prefers
CREATE TABLE Prefers (
  customer_Name  VARCHAR(20) REFERENCES Customer (Name),
  email_id VARCHAR(20),
  Author_name  VARCHAR(20),
  Address VARCHAR(20),
  PRIMARY KEY(customer_Name, email_id, Author_name, Address));

--TRIGGERS
--45) Create a trigger for the table sells_book
--i)
CREATE TRIGGER sells_trig
AFTER INSERT ON sells_book
REFERENCING NEW ROW AS NewTuple
FOR EACH ROW
WHEN (NewTuple.ISBN NOT IN (SELECT ISBN FROM books))
INSERT INTO books(ISBN) VALUES (NewTuple.ISBN);
--ii)
CREATE TRIGGER seller_id_trig
AFTER INSERT ON sells_book
REFERENCING NEW ROW AS NewTuple1
FOR EACH ROW
WHEN (NewTuple1.seller_id NOT IN (SELECT seller_id FROM Book_seller))
```

```
INSERT INTO Book_seller(seller_id) VALUES (NewTuple1.seller_id);


--Create a trigger for the table published_by
--i)
CREATE TRIGGER publish_trig
AFTER INSERT ON Published_by
REFERENCING NEW ROW AS NewTuple
FOR EACH ROW
WHEN (NewTuple.ISBN NOT IN (SELECT ISBN FROM books))
INSERT INTO books(ISBN) VALUES (NewTuple.ISBN);
--ii)
CREATE TRIGGER publish_trig_name
AFTER INSERT ON Published_by
REFERENCING NEW ROW AS NewTuple1
FOR EACH ROW
WHEN (NewTuple.name NOT IN (SELECT name FROM Publisher))
INSERT INTO Publisher(name) VALUES (NewTuple.name);


--Create a trigger for the table written_by
--i)
CREATE TRIGGER written_trig
AFTER INSERT ON written_by
REFERENCING NEW ROW AS NewTuple
FOR EACH ROW
WHEN (NewTuple.name NOT IN (SELECT name FROM Author))
INSERT INTO Author(name) VALUES (NewTuple.name);
--ii)
```

```
CREATE TRIGGER written_trig
AFTER INSERT ON written_by
REFERENCING NEW ROW AS NewTuple1
FOR EACH ROW
WHEN (NewTuple.ISBN NOT IN (SELECT ISBN FROM books))
INSERT INTO books(ISBN) VALUES (NewTuple.ISBN);

--create a trigger for the table Rents
--i)
CREATE TRIGGER rents_trig
AFTER INSERT ON rents
REFERENCING NEW ROW AS NewTuple
FOR EACH ROW
WHEN (NewTuple.ISBN NOT IN (SELECT ISBN FROM books))
INSERT INTO books(ISBN) VALUES (NewTuple.ISBN);
--ii)
CREATE TRIGGER rents_trig_cust
AFTER INSERT ON rents
REFERENCING NEW ROW AS NewTuple1
FOR EACH ROW
WHEN (NewTuple.name NOT IN (SELECT name FROM customer))
INSERT INTO customer(name) VALUES (NewTuple.name);

--create a trigger for the table Buys
--i)
CREATE TRIGGER buys_trig
AFTER INSERT ON buys
```

```
REFERENCING NEW ROW AS NewTuple
FOR EACH ROW
WHEN (NewTuple.ISBN NOT IN (SELECT ISBN FROM books))
INSERT INTO books(ISBN) VALUES (NewTuple.ISBN);
--ii)
CREATE TRIGGER buys_trig_cust
AFTER INSERT ON buys
REFERENCING NEW ROW AS NewTuple1
FOR EACH ROW
WHEN (NewTuple.name NOT IN (SELECT name FROM customer))
INSERT INTO customer(name) VALUES (NewTuple.name);

--create a trigger for the table stocks
--i)
CREATE TRIGGER stocks_trig
AFTER INSERT ON stocks
REFERENCING NEW ROW AS NewTuple
FOR EACH ROW
WHEN (NewTuple.ISBN NOT IN (SELECT ISBN FROM books))
INSERT INTO books(ISBN) VALUES (NewTuple.ISBN);
--ii)
CREATE TRIGGER stocks_trig_code
AFTER INSERT ON stocks
REFERENCING NEW ROW AS NewTuple1
FOR EACH ROW
WHEN (NewTuple.code NOT IN (SELECT code FROM storage_warehouse))
INSERT INTO storage_warehouse(code) VALUES (NewTuple.code);
```

```
--create a trigger for the table prefers

--i)

CREATE TRIGGER prefers_trig

AFTER INSERT ON prefers

REFERENCING NEW ROW AS NewTuple

FOR EACH ROW

WHEN (NewTuple.customer_Name NOT IN (SELECT name FROM customer))

INSERT INTO customer(name) VALUES (NewTuple.customer_Name);


--PSM

--46) Create a PSM for the table Books

CREATE PROCEDURE book_procedure (ISBN IN VARCHAR, title IN VARCHAR, pub_year IN
VARCHAR) IS

BEGIN

INSERT INTO Books VALUES ('9856321458756', 'High Performance Computing', '2015');

END;

run;


--Create a PSM for the table Publisher

CREATE PROCEDURE publisher_procedure ( Name IN  VARCHAR, phone_no IN VARCHAR, URL IN
VARCHAR, Address INOUT VARCHAR) IS

BEGIN

INSERT INTO Publisher VALUES ('Norcom', '589-325-6598', 'norcom.com', '8324, jonesboro');

END;

run;


--Create a PSM for the table Published_by
```

```
CREATE PROCEDURE Published_by_procedure ( ISBN INOUT  VARCHAR, Name INOUT
VARCHAR) IS

BEGIN

INSERT INTO Published_by VALUES ('9856321458756','Norcom');

END;

run;
```

--Relational Algebra

--47) Write a relational algebra for Finding the isbns of the books that are in both buys and rents

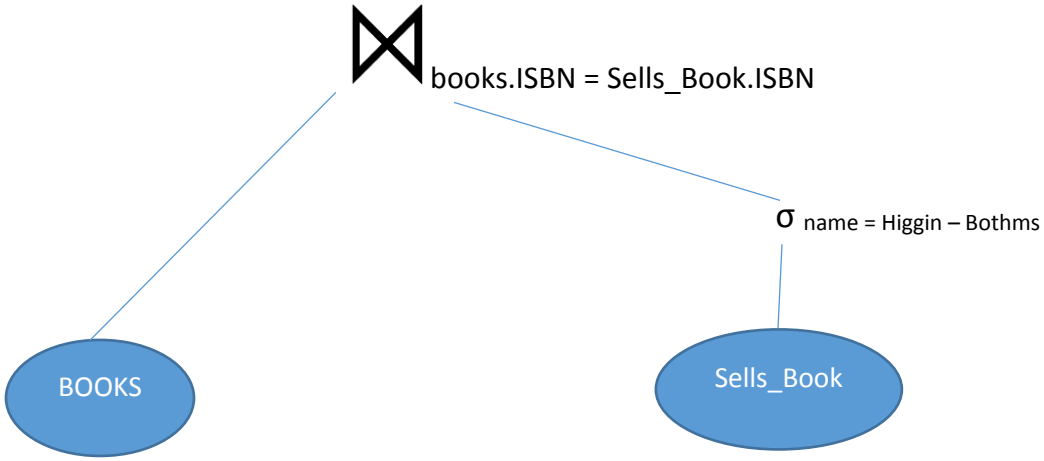
$\pi_{ISBN} (\pi_{ISBN} (buys) \cap \pi_{ISBN} (rents))$

-- Write a relational algebra for Finding the name of the books that are sold at Higgin-Bothms

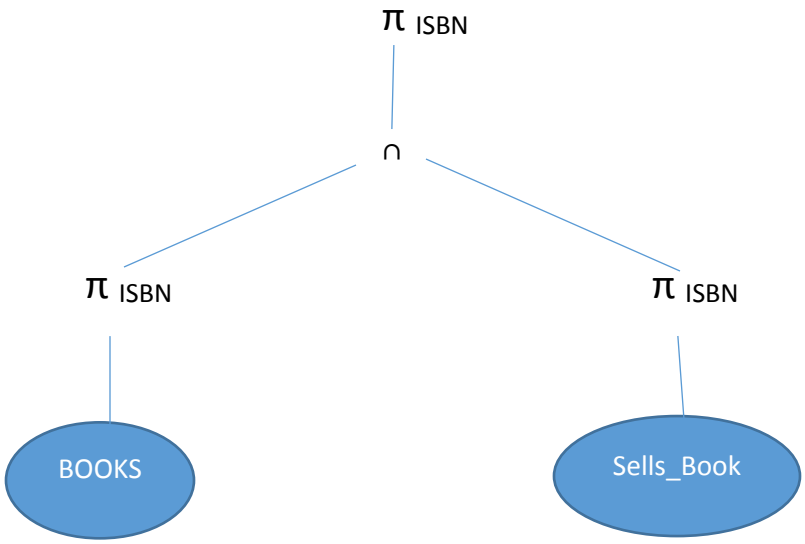
$\pi_{books, title} (Books \bowtie_{books.ISBN = sells\_Book.ISBN} (\sigma_{name = Higgin - Bothms} (Sells\_Book))$

--48) Draw a relational algebra tree for Finding the name of the books that are sold at Higgin-Bothms

π books, title



--Draw a relational algebra Tree for Finding the isbns of the books that are in both buys and rents



--49) Functional dependencies for each table.

- Books (<u>ISBN,</u> title, Pub_year)
  ISBN -> title
  ISBN -> Pub_year

- Publisher (<u>Name,</u> phone_no, URL, Address)
  Name,phone_no -> URL, Address

URL -> phone_no

- Book_seller (Seller_id, Name, Address, email_id, phone_no)
  Seller_id -> Name
  Seller_id -> Address
  Name, Address -> email_id, phone_no
  Address, email_id -> Seller_id

- Author (Name, Address)
  Name -> Address

- Customer (Name, Address, email_id, phone_no)
  Name, Address -> email_id
  email_id -> phone_no
  phone_no -> Name

- Storage_warehouse (Code, Address, phone_no)
  Code -> Address
  Code -> phone_no

- Published_by (ISBN, Name)
  ISBN -> Name

- Sells_book (Seller_id, Name, Address, ISBN, price)
  Seller_id -> Name
  Seller_id -> Address
  Name, Address -> ISBN, price
  Address, ISBN -> Seller_id

- Written_by (Name, Address, ISBN)
  Name, Address -> ISBN
  Name -> Address

- Rents (Name, email_id, ISBN)
  email_id, ISBN -> Name
  email_id -> ISBN

- Buys (Name, email_id, ISBN)
  email_id, ISBN -> Name

email_id -> ISBN

- Stocks (ISBN, code, Numbers)
  ISBN -> Numbers
  ISBN -> code

- Prefers (customer_Name, email_id, Author_name, Address)
  Customer_Name, email_id -> Author_name
  Author_name -> Address
  Address -> customer_Name

--50) Indicate 3NF or BCNF or 4NF for each table

- Books (ISBN, title, Pub_year) => Table is in BCNF
  $ISBN^+$ = {ISBN, title, Pub_year} => BCNF

- Publisher (Name, phone_no, URL, Address) => Table is in 3NF
  Name,$phone\_no^+$ = {Name, Phone_no, URL, Address} => BCNF
  URL^+ = {URL, phone_no} => 3NF since URL and phone_no is prime

- Book_seller (Seller_id, Name, Address, email_id, phone_no) => Table is in BCNF
  $Seller\_id^+$ = { Seller_id, Name, Address, email_id, phone_no} => BCNF
  Name, Address $^+$ = {Name, Address, email_id, phone_no, Seller_id} => BCNF
  Address, email_id $^+$ = { Name, Address, email_id, phone_no, Seller_id} => BCNF

- Author (Name, Address) => Table is in BCNF
  $Name^+$ = {Name, Address} => BCNF

- Customer (Name, Address, email_id, phone_no) => Table is in 3NF
  Name, $Address^+$ = { Name, Address, email_id, phone_no} => BCNF
  $email\_id^+$ = { email_id, phone_no, Name} => 3NF
  $phone\_no^+$ = {phone_no, Name} => 3NF

- Storage_warehouse (Code, Address, phone_no) => Table is in BCNF
  $Code^+$ = {Code, Address, phone_no} => BCNF

- Published_by (ISBN, Name) => Table is in BCNF
  $ISBN^+$ = {ISBN, Name} => BCNF

- Sells_book (Seller_id, Name, Address, ISBN, price) => Table is in BCNF
  Seller_id$^+$ = {seller_id, Name, Address, ISBN, price} => BCNF
  Name, Address$^+$ = {Name, Address, ISBN, price, Seller_id} => BCNF
  Address, ISBN$^+$ = {Name, Address, ISBN, price, Seller_id} => BCNF

- Written_by (Name, Address, ISBN) => Table is in BCNF
  Name, Address$^+$ = {Name, Address, ISBN} => BCNF
  Name$^+$ = {ISBN, Name, Address} => BCNF

- Rents (Name, email_id, ISBN) => Table is in BCNF
  Email_id, ISBN$^+$ = {ISBN, Name, email_id} => BCNF
  Email_id$^+$ = {Name, email_id} => BCNF

- Buys (Name, email_id, ISBN) => Table is in BCNF
  Email_id, ISBN$^+$ = {ISBN, Name, email_id} => BCNF
  Email_id$^+$ = {Name, email_id} => BCNF

- Stocks (ISBN, code, Numbers) => Table is in BCNF
  ISBN$^+$ = {ISBN, Numbers, code} => BCNF

- Prefers (customer_Name, email_id, Author_name, Address) => Table is in 3NF
  Customer_Name, email_id$^+$ = {customer_Name, email_id, Author_name, Address } => BCNF
  Author_name$^+$ = {Author_name, Address, Customer_Name} => 3NF since all are prime on right side
  Address$^+$ = { Address, Customer_Name} => 3NF since all are prime on right side

--51) One simple interface to access your data from class machine.

http://147.97.156.242/~rakesh.tirupath/Book_Store.php

The PHP file for the above link is given as:

```
<html>

<head>

<title> Show </title>

</head>

<body>
```

```
<?PHP
                $dsn="pgsql:host=localhost;dbname=rakesh.tirupath";   // data source
name
                $dbuser='rakesh.tirupath';
                $password = 'Tirupathi';

                $conn = new PDO($dsn, $dbuser, $password);

                if (!$conn)
                {
                echo "Could not connect!!!!\n";
                exit;
                }
                //else
                //{
                //  echo "connected<br>\n";  //for debug
                //}


                // Get the records for those which gender is 'M'.
                //
                $query = "SELECT * FROM sells_book WHERE seller_id=:seller_id";
                echo "<h4> $query </h4> \n";

                //prepare the SQL statement
                $sqlquery=$conn->prepare($query, array(PDO::ATTR_CURSOR =>
PDO::CURSOR_FWDONLY));
                // execute the SQL statement
```

```
$sqlquery->execute(array(':seller_id' => '6873154'));

// get the results of the sql statement
if ($row = $sqlquery->fetch(PDO::FETCH_ASSOC))
{
echo "<table border=\"1\">\n";  //table
echo "<tr>";
foreach ($row as $key=>$value)
        echo "<th>".strtoupper($key)."</th> ";
echo "</tr>\n";
do {
        echo "<tr>";
        foreach ($row as $key => $value)
        {
                //echo "$key: $value ";
                echo "<td>" . $row["$key"] . "</td> ";
        }
        echo "</tr>\n";
} while($row = $sqlquery->fetch(PDO::FETCH_ASSOC));
echo "</table>";
}

$query = "SELECT isbn,code FROM stocks WHERE numbers=:numbers";
echo "<h4> $query </h4> \n";

//prepare the SQL statement
```

```
                $sqlquery=$conn->prepare($query, array(PDO::ATTR_CURSOR =>
PDO::CURSOR_FWDONLY));
                // execute the SQL statement
                $sqlquery->execute(array(':numbers' => '10 nos'));

                // get the results of the sql statement
                if ($row = $sqlquery->fetch(PDO::FETCH_ASSOC))
                {
                echo "<table border=\"1\">\n";  //table
                echo "<tr>";
                foreach ($row as $key=>$value)
                        echo "<th>".strtoupper($key)."</th> ";
                echo "</tr>\n";
                do {
                        echo "<tr>";
                        foreach ($row as $key => $value)
                        {
                                //echo "$key: $value ";
                                echo "<td>" . $row["$key"] . "</td> ";
                        }
                        echo "</tr>\n";
                } while($row = $sqlquery->fetch(PDO::FETCH_ASSOC));
                echo "</table>";
                }

?>
<h3>Done!</h3>
```

```
</body>
</html>
```