Faculty of Engineering and Computer Science
Expectations of Originality

This form has been created to ensure that all students in the Faculty of Engineering and Computer Science comply with principles of academic integrity prior to submitting coursework to their instructors for evaluation: namely reports, assignments, lab reports and/or software. All students should become familiar with the University's Code of Conduct (Academic) located at http://web2.concordia.ca/Legal_Counsel/policies/english/AC/Code.html

Please read the back of this document carefully before completing the section below. This form must be attached to the front of all coursework submitted to instructors in the Faculty of Engineering and Computer Science.

Group Account: gxc353_1

Course Number: COMP 353 **Instructor:** Khaled Jababo

Type of Submission (Please check off reponses to both a & b)

a. X Report __ Assignment __ Lab Report __ Software

b. __ Individual submission X Group Submission (All members of the team must sign below)

Having read both sides of this form, I certify that I/we have conformed to the Faculty's expectations of originality and standards of academic integrity.

Name: Arunraj Adlee (please print clearly) ID No: 40059206 Signature: Date: 2020-07-19

Name: Gordon Pham-Nguyen (please print clearly) ID No: 40018402 Signature: Date: 2020-07-1

Name: Leo Jr Silao (please print clearly) ID No: 40056822 Signature: Date: 2020-07-1

Name: Tiffany Zeng (please print clearly) ID No: 40063115 Signature: Date: 2020-07-1

Name: _____ (please print clearly) ID No: _____ Signature: _____ Date: _____

Name: _____ (please print clearly) ID No: _____ Signature: _____ Date: _____

Do Not Write in this Space – Reserved for Instructor

EXPECTATIONS OF ORIGINALITY & STANDARDS OF ACADEMIC INTEGRITY

ALL SUBMISSIONS must meet the following requirements:

1. The decision on whether a submission is a group or individual submission is determined by the instructor. Individual submissions are done alone and should not be identical to the submission made by any other student. In the case of group submissions, all individuals in the group must be listed on and must sign this form prior to its submission to the instructor.
2. All individual and group submissions constitute original work by the individual(s) signing this form.
3. Direct quotations make up a very small proportion of the text, i.e., not exceeding 5% of the word count.
4. Material paraphrased from a source (e.g., print sources, multimedia sources, web-based sources, course notes or personal interviews) has been identified by a numerical reference citation.
5. All of the sources consulted and/or included in the report have been listed in the Reference section of the document.
6. All drawings, diagrams, photos, maps or other visual items derived from other sources have been identified by numerical reference citations in the caption.
7. No part of the document has been submitted for any other course.
8. Any exception to these requirements are indicated on an attached page for the instructor's review.

REPORTS and ASSIGNMENTS must also meet the following additional requirements:

1. A report or assignment consists entirely of ideas, observations, information and conclusions composed by the student(s), except for statements contained within quotation marks and attributed to the best of the student's/students' knowledge to their proper source in footnotes or references.
2. An assignment may not use solutions to assignments of other past or present students/instructors of this course or of any other course.
3. The document has not been revised or edited by another student who is not an author.
4. For reports, the guidelines found in Form and Style, by Patrick MacDonagh and Jack Borden (Fourth Edition: May 2000, available at http://www.encs.concordia.ca/scs/Forms/Form&Style.pdf) have been used for this submission.

LAB REPORTS must also meet the following requirements:

1. The data in a lab report represents the results of the experimental work by the student(s), derived only from the experiment itself. There are no additions or modifications derived from any outside source.
2. In preparing and completing the attached lab report, the labs of other past or present students of this course or any other course have not been consulted, used, copied, paraphrased or relied upon in any manner whatsoever.

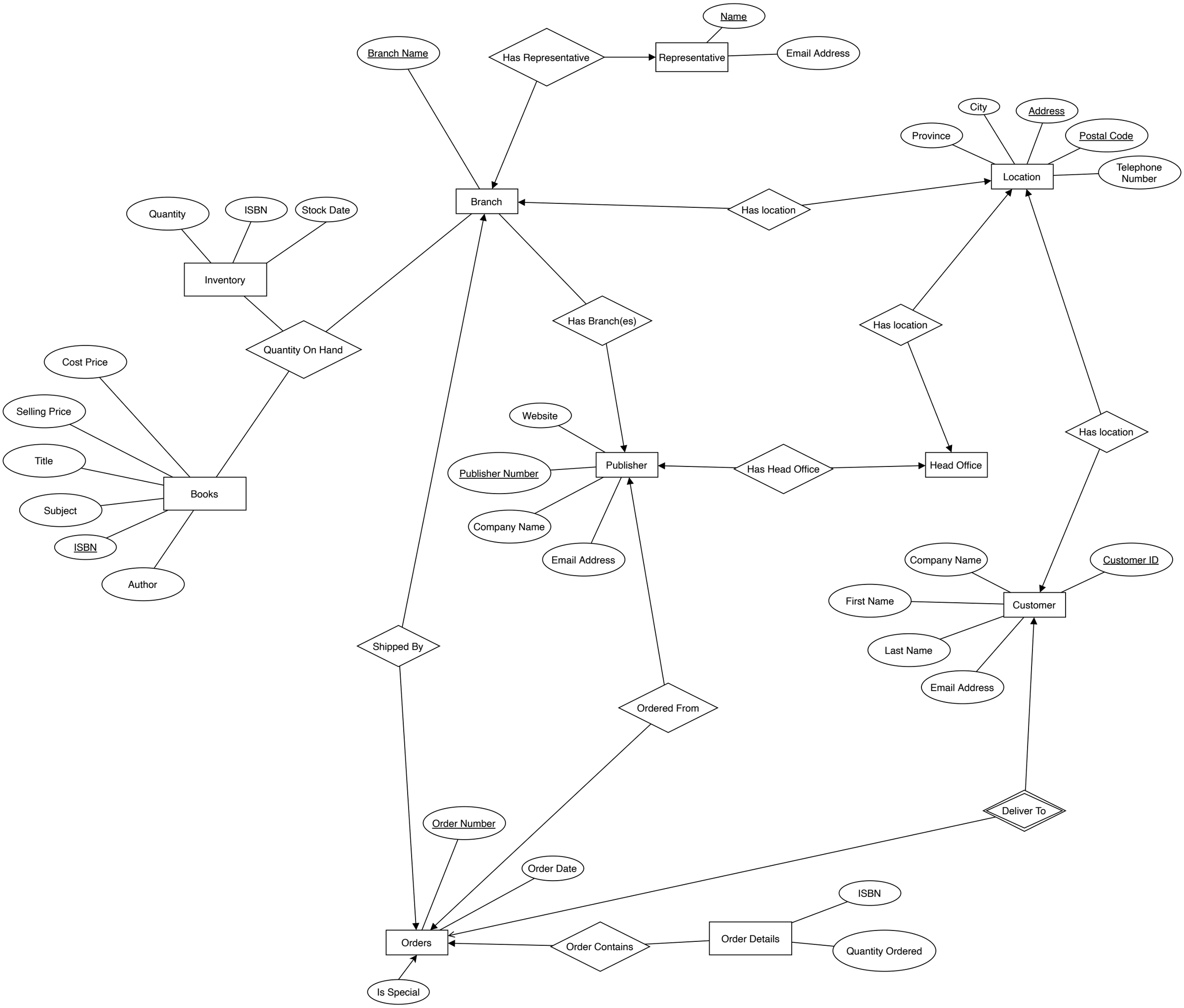
SOFTWARE must also meet the following requirements:

1. The software represents independent work of the student(s).
2. No other past or present student work (in this course or any other course) has been used in writing this software, except as explicitly documented.
3. The software consists entirely of code written by the undersigned, except for the use of functions and libraries in the public domain, all of which have been documented on an attached page.
4. No part of the software has been used in previous submissions except as identified in the documentation.
5. The documentation of the software includes a reference to any component that the student(s) did not write.
6. All of the sources consulted while writing this code are listed in the documentation.

Important: Should you require clarification on any of the above items please contact your instructor.

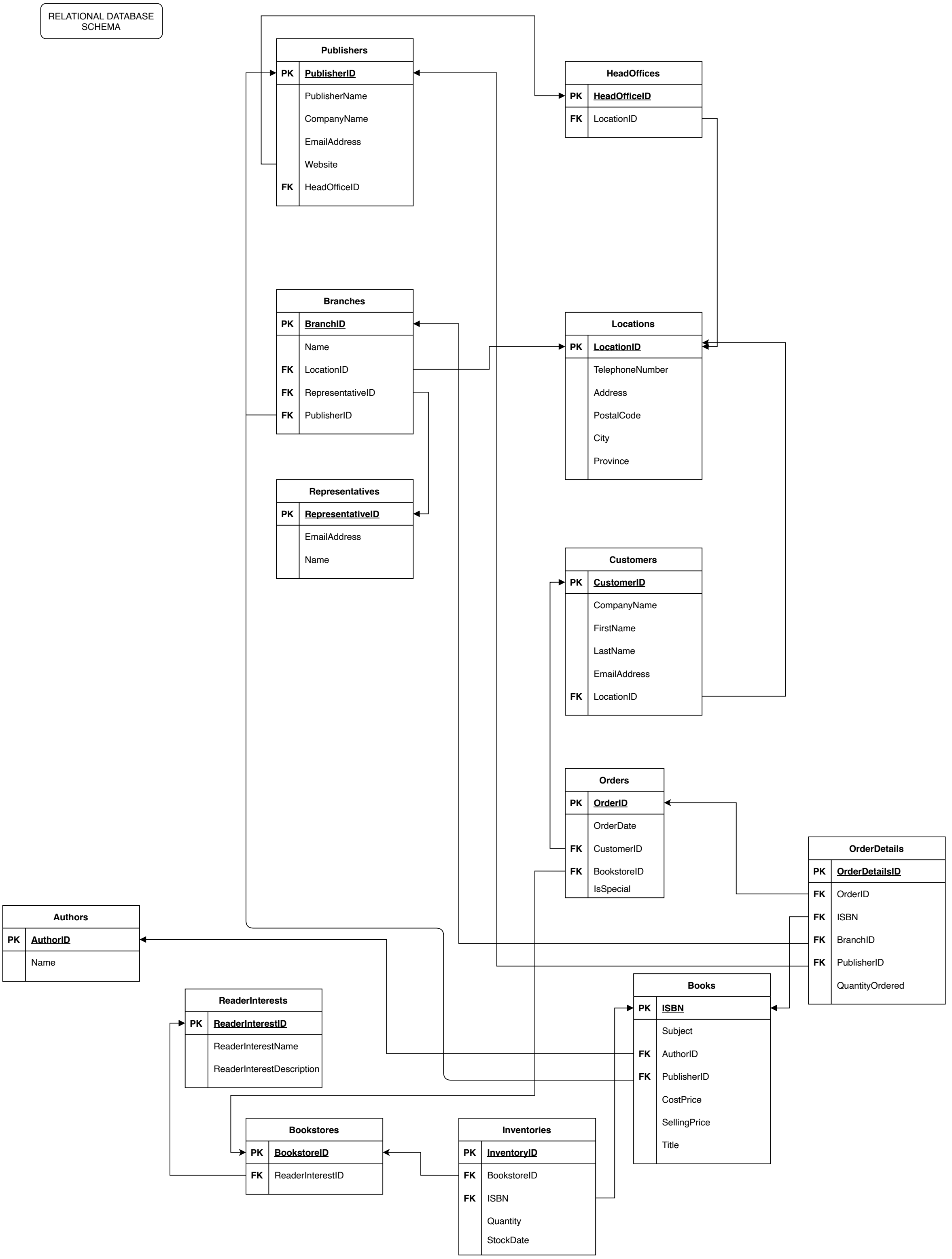
PART 1

E/R DIAGRAM



PART 1

RELATIONAL DATABASE SCHEMA



PART 2

> The SQL statements formulated and used to create the database. Pick appropriate data types for the attributes and include them in your report.

```sql
CREATE TABLE ReaderInterests
(
    ReaderInterestID          int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ReaderInterestName        varchar(255),
    ReaderInterestDescription varchar(255)
);
```

```sql
CREATE TABLE Authors
(
    AuthorID int          NOT NULL AUTO_INCREMENT PRIMARY KEY,
    Name     varchar(255) NOT NULL
);
```

```sql
CREATE TABLE Locations
(
    LocationID      int          NOT NULL AUTO_INCREMENT PRIMARY KEY,
    TelephoneNumber int,
    Address         varchar(255) NOT NULL,
    PostalCode      varchar(255) NOT NULL,
    City            varchar(255) NOT NULL,
    Province        varchar(255) NOT NULL
);
```

```sql
CREATE TABLE Representatives
(
    RepresentativeID int          NOT NULL AUTO_INCREMENT PRIMARY KEY,
    EmailAddress     varchar(255),
    Name             varchar(255) NOT NULL
);
```

```sql
CREATE TABLE HeadOffices
(
    HeadOfficeID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    LocationID   int NOT NULL,
    FOREIGN KEY (LocationID) REFERENCES Locations (LocationID)
);
```

```sql
CREATE TABLE Publishers
(
    PublisherID   int          NOT NULL AUTO_INCREMENT PRIMARY KEY,
    PublisherName varchar(255) NOT NULL,
    CompanyName   varchar(255) NOT NULL,
    EmailAddress  varchar(255),
    Website       varchar(255),
    HeadOfficeID  int          NOT NULL,
    FOREIGN KEY (HeadOfficeID) REFERENCES HeadOffices (HeadOfficeID)
);
```

```sql
CREATE TABLE Branches
(
    BranchID         int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    Name             varchar(255),
    LocationID       int NOT NULL,
    RepresentativeID int NOT NULL,
    PublisherID      int NOT NULL,
    FOREIGN KEY (LocationID) REFERENCES Locations (LocationID),
    FOREIGN KEY (RepresentativeID) REFERENCES Representatives (RepresentativeID),
    FOREIGN KEY (PublisherID) REFERENCES Publishers (PublisherID)
);
```

```sql
CREATE TABLE Customers
(
    CustomerID   int          NOT NULL AUTO_INCREMENT PRIMARY KEY,
    CompanyName  varchar(255) NOT NULL,
    FirstName    varchar(255),
    LastName     varchar(255),
    EmailAddress varchar(255),
    LocationID   int          NOT NULL,
    FOREIGN KEY (LocationID) REFERENCES Locations (LocationID)
);
```

```sql
CREATE TABLE Bookstores
(
    BookstoreID      int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ReaderInterestID int NOT NULL,
    FOREIGN KEY (ReaderInterestID) REFERENCES ReaderInterests (ReaderInterestID)
);
```

```sql
SET time_zone = '-04:00';

CREATE TABLE Orders
(
    OrderID     int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    OrderDate   timestamp DEFAULT CURRENT_TIMESTAMP,
    CustomerID  int,
    BookstoreID int,
    IsSpecial   boolean DEFAULT false,
    FOREIGN KEY (CustomerID) REFERENCES Customers (CustomerID),
    FOREIGN KEY (BookstoreID) REFERENCES Bookstores (BookstoreID)
);
```

```sql
CREATE TABLE Books
(
    ISBN         int           NOT NULL AUTO_INCREMENT PRIMARY KEY,
    Subject      varchar(255),
    AuthorID     int           NOT NULL,
    PublisherID  int           NOT NULL,
    Title        varchar(255)  NOT NULL,
    SellingPrice decimal(5, 2) NOT NULL,
    CostPrice    decimal(5, 2) NOT NULL,
    FOREIGN KEY (AuthorID) REFERENCES Authors (AuthorID),
    FOREIGN KEY (PublisherID) REFERENCES Publishers (PublisherID)
);
```

```sql
CREATE TABLE OrderDetails
(
    OrderDetailsID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    OrderID        int NOT NULL,
    ISBN           int NOT NULL,
    Quantity       int NOT NULL,
    BranchID       int NOT NULL,
    PublisherID    int NOT NULL,
    FOREIGN KEY (OrderID) REFERENCES Orders (OrderID),
    FOREIGN KEY (ISBN) REFERENCES Books (ISBN),
    FOREIGN KEY (BranchID) REFERENCES Branches (BranchID),
    FOREIGN KEY (PublisherID) REFERENCES Publishers (PublisherID)
);
```

```sql
CREATE TABLE Inventories
(
    InventoryID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
    BookstoreID int NOT NULL,
    ISBN        int NOT NULL,
    Quantity    int NOT NULL,
    StockDate   timestamp DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (BookstoreID) REFERENCES Bookstores (BookstoreID),
    FOREIGN KEY (ISBN) REFERENCES Books (ISBN)
);
```

PART 3

> The SQL statements formulated to express the required queries and transactions mentioned.

```sql
/* question 3.1 Get details of all books in stock ordered by
year-to-date-qty-sold in descending order. */
SELECT b.ISBN,
       b.Subject,
       a.Name,
       b.SellingPrice,
       b.CostPrice,
       b.Title,
       SUM(od.Quantity) AS Quantity,
       o.OrderDate
FROM Books AS b,
     Inventories AS i,
     Orders AS o,
     OrderDetails AS od,
     Authors AS a
WHERE o.OrderID = od.OrderID
  AND od.ISBN = b.ISBN
  AND b.ISBN = i.ISBN
  AND b.AuthorID = a.AuthorID
  AND i.Quantity > 0
GROUP BY b.ISBN
ORDER BY o.OrderDate DESC, SUM(od.Quantity) DESC;
```

```sql
-- question 3.2 Get details of all back orders for a given publisher.
SELECT od.OrderDetailsID, od.OrderID, od.ISBN, od.Quantity
FROM OrderDetails AS od,
     Publishers AS p,
     Orders AS o,
     Books AS b,
     Inventories AS i
WHERE p.PublisherID = o.PublisherID
  AND b.PublisherID = [GIVEN PublisherID]
  AND o.OrderID = od.OrderID
  AND b.ISBN = od.ISBN
  AND i.ISBN = b.ISBN
  AND i.Quantity < 1;
```

```sql
-- question 3.3 For a given customer, get details of all his/her special orders
SELECT od.OrderDetailsID, od.OrderID, od.ISBN, od.Quantity
FROM OrderDetails AS od,
     Orders AS o,
     Customers c
WHERE od.OrderID = o.OrderID
  AND c.CustomerID = o.CustomerID
  AND c.CustomerID = [GIVEN CustomerID]
  AND o.IsSpecial = true;
```

```sql
/* question 3.4 For a given customer, get details of all his/her purchases
 made during a specific period of time from a given branch. */
SELECT od.OrderDetailsID, od.OrderID, od.ISBN, od.Quantity, b.title AS Title
FROM OrderDetails AS od,
     Orders AS o,
     Customers AS c,
     Books AS b
WHERE od.OrderID = o.OrderID
  AND c.CustomerID = o.CustomerID
  AND od.ISBN = b.ISBN
  AND od.BranchID = [GIVEN BranchID]
  AND o.CustomerID = [GIVEN CustomerID]
  AND DATE (o.OrderDate) BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD';
```

```sql
/* question 3.5 Give a report of sales during a specific period of time
for a given branch. */
SELECT o.OrderID,
       bo.Title,
       b.Name,
       p.PublisherName,
       o.BookstoreID,
       (SELECT DISTINCT boo.SellingPrice
        FROM Books AS boo
        WHERE od.ISBN = boo.ISBN) * od.Quantity AS Total,
       o.OrderDate,
       b.BranchID
FROM Orders AS o,
     Customers AS c,
     Branches AS b,
     Books AS bo,
     Publishers AS p,
     OrderDetails AS od
WHERE o.OrderID = od.OrderID
      AND b.BranchID = od.BranchID
      AND b.BranchID = 1
      AND bo.ISBN = od.ISBN
      AND p.PublisherID = od.PublisherID
      AND Date(o.OrderDate) BETWEEN 'YYYY-MM-DD' AND 'YYYY-MM-DD'
GROUP BY od.OrderDetailsID;
```

```sql
/* question 3.6 Find the title and name of publisher of book(s)
that have the highest backorder */
SELECT b.Title,
       p.PublisherName,
       SUM(od.Quantity) AS Total_Sold
FROM   Books as b,
       Publishers as p,
       OrderDetails as od
WHERE b.ISBN = od.ISBN
        AND p.PublisherID = od.PublisherID
        AND b.PublisherID = p.PublisherID
        AND od.PublisherID = b.PublisherID
GROUP BY b.ISBN
HAVING Total_Sold= (SELECT MAX(q2.Quantity)
    FROM (
            SELECT  od.ISBN,
                    SUM(od.Quantity) as Quantity
            FROM OrderDetails AS od
            GROUP BY od.ISBN
          ) as q2 );
```

```sql
/* question 3.7 Give details of books that are supplied by a given publisher
ordered by their sale price in increasing order. */
SELECT b.ISBN, a.Name, b.Title, b.SellingPrice, b.CostPrice, p.PublisherName
FROM Books AS b,
     Authors AS a,
     Publishers AS p
WHERE b.PublisherID = p.PublisherID
  AND b.AuthorID = a.AuthorID
  AND b.PublisherID = [GIVEN PUBLISHER ID]
ORDER BY b.SellingPrice ASC;
```

```sql
/* question 3.8 For all publishers who have at least three branches,
get details of the head office and all the branches for those publishers.*/
SELECT p.PublisherID,
       b.Name            AS 'Branch Name',
       r.Name            AS 'Representative Name',
       l.TelephoneNumber AS 'Head Office Telephone Number',
       l.City            AS 'Head Office City',
       l.Province        AS 'Head Office Province',
       l.Address         AS 'Head Office Address'
FROM Branches AS b,
     Representatives AS r,
     Publishers AS p,
     Locations AS l,
     HeadOffices AS h
WHERE b.RepresentativeID = r.RepresentativeID
  AND b.PublisherID = p.PublisherID
  AND p.HeadOfficeID = h.HeadOfficeID
  AND (h.LocationID = l.LocationID)
  AND b.PublisherID IN (
    SELECT PublisherID
    FROM Branches
    GROUP BY PublisherID
    HAVING Count(*) >= 3)
ORDER BY p.PublisherID ASC;
```

```sql
/* question 3.9 Get details of books that are in the inventory for at
least one year but there have never been a purchase for that specific book. */
SELECT b.ISBN, a.Name, b.Title, b.SellingPrice, b.CostPrice, p.PublisherName
FROM Books AS b,
     Authors AS a,
     Publishers AS p,
     Inventories AS i
WHERE b.PublisherID = p.PublisherID
  AND b.AuthorID = a.AuthorID
  AND i.ISBN = b.ISBN
  AND b.ISBN NOT IN (SELECT od.ISBN FROM OrderDetails AS od)
  AND DATEDIFF(NOW(), i.StockDate) >= 1;
```

```sql
/* question 3.10 Get details of all books that are in the
inventory for a given author. */
SELECT b.ISBN, a.Name, b.Title, b.SellingPrice, b.CostPrice, p.PublisherName
FROM Books AS b,
     Authors AS a,
     Publishers AS p,
     Inventories AS i
WHERE b.PublisherID = p.PublisherID
  AND b.AuthorID = a.AuthorID
  AND i.ISBN = b.ISBN
  AND i.Quantity > 0
  AND a.Name = [GIVEN AUTHOR NAME]
```

PART 4

> Populate each table in the database with at least 10 representative and appropriate tuples.

```sql
INSERT INTO ReaderInterests(ReaderInterestName, ReaderInterestDescription)
VALUES ('Science fiction', 'science that is fiction'),
       ('Fiction', 'fiction that is not science'),
       ('Romance', 'people in love'),
       ('Comedy', 'makes you laugh'),
       ('Mystery', 'some sort of crime'),
       ('Fantasy', 'imagination'),
       ('Classics', 'assigned in English class'),
       ('Adventure', 'Amos Daragon'),
       ('Horror', 'paranormal stuff'),
       ('Cookbooks', 'makes you hungry');
```

```sql
INSERT INTO Authors(Name)
VALUES ('Leo Jr Silao'),
       ('Gordon Pham-Nguyen'),
       ('Tiffany Zeng'),
       ('Arunraj Adlee'),
       ('Malcolm Gladwell'),
       ('Yuval Noah Harari'),
       ('Daniel Kahneman'),
       ('Khaled Jababo'),
       ('Robert Bourassa'),
       ('Donald Trump');
```

```sql
INSERT INTO Locations(Address, PostalCode, City, Province)
VALUES ('1500 McGill College Ave', 'H3A 3J5', 'Montreal', 'Quebec'),
       ('116 Bond St', 'M5B 1X8', 'Toronto', 'Ontario'),
       ('1198 Commercial Dr', 'V5L 3X2', 'Vancouver', 'British Columbia'),
       ('12 E 49th St', '10017', 'Manhattan', 'New York'),
       ('2131 Timber Ridge Road', '95814', 'Sacramento', 'California'),
       ('3010 Jarvisville Road', '10016', 'Manhattan', 'New York'),
       ('2114 Goldleaf Lane', '07662', 'Rochelle Park', 'New Jersey'),
       ('4489 Pine Street', '15219', 'Pittsburgh', 'Pennsylvania'),
       ('3788 Lowland Drive', '61081', 'Sterling', 'Illinois'),
       ('1300 Goodwin Avenue', '99206', 'Spokane Valley', 'Washington');
```

```sql
INSERT INTO Representatives(Name)
VALUES ('Laura J Bell'),
       ('Violet J Hazelip'),
       ('Gladys C Clay'),
       ('Tyrell M Barret'),
       ('James R Hadsell'),
       ('Karen D Damon'),
       ('Frances A Kitson'),
       ('Thomas R Stevenson'),
       ('Janet E Fields'),
       ('Marissa J Poindexter');
```

```sql
INSERT INTO HeadOffices(LocationID)
VALUES (1),
       (2),
       (3),
       (4),
       (5),
       (6),
       (7),
       (8),
       (9),
       (10);
```

```sql
INSERT INTO Publishers(PublisherName, CompanyName, HeadOfficeID)
VALUES ('Sound Advice', 'Sound Advice', 1),
       ('Parade of Shoes', 'Parade of Shoes', 2),
       ('Jacob Reed and Sons', 'Jacob Reed and Sons', 3),
       ('Body Toning', 'Body Toning', 4),
       ('Copeland Sports', 'Copeland Sports', 5),
       ('Xray Eye & Vision Clinics', 'Xray Eye & Vision Clinics', 6),
       ('Pomeroy', 'Pomeroy', 7),
       ('Silverwoods', 'Silverwoods', 8),
       ('Boston Sea Party', 'Boston Sea Party', 9),
       ('Rolling Thunder', 'Rolling Thunder', 10);
```

```sql
INSERT INTO Branches(LocationID, RepresentativeID, PublisherID)
VALUES (1, 1, 1),
       (2, 2, 2),
       (3, 3, 3),
       (4, 4, 4),
       (5, 5, 5),
       (6, 6, 6),
       (7, 7, 7),
       (8, 8, 8),
       (9, 9, 9),
       (10, 10, 10);
```

```sql
INSERT INTO Customers(CompanyName, LocationID)
VALUES ('Mistimba', 1),
       ('Cervor', 2),
       ('Peric', 3),
       ('Geotri', 4),
       ('Fronter', 5),
       ('Cynize', 6),
       ('Metatude', 7),
       ('Aurous', 8),
       ('Capive', 9),
       ('Camimbee', 10);
```

```sql
INSERT INTO Bookstores(ReaderInterestID)
VALUES (1),
       (2),
       (3),
       (4),
       (5),
       (6),
       (7),
       (8),
       (9),
       (10);
```

```sql
INSERT INTO Orders(CustomerID, BookstoreID, IsSpecial)
VALUES (NULL, 4, false),
       (NULL, 2, true),
       (5, NULL, true),
       (1, NULL, true),
       (NULL, 5, false),
       (5, NULL, false),
       (4, NULL, false),
       (NULL, 6, false),
       (NULL, 8, false),
       (NULL, 7, false),
       (8, NULL, false),
       (5, NULL, false);
```

```sql
INSERT INTO Books(AuthorID, PublisherID, Title, SellingPrice, CostPrice)
VALUES (1, 1, 'Clue of the Split Creek', 54.97, 19.22),
       (2, 2, 'Sign of the Ghostly Amulet', 58.38, 99.88),
       (3, 3, 'The Ebony Window', 10.84, 19.54),
       (4, 4, 'Death of the Shrieking Shih Tzu', 18.74, 40.29),
       (5, 5, 'The Cobalt Curtain', 55.11, 85.04),
       (6, 6, 'The Crown in the Abyss', 60.81, 59.54),
       (7, 7, 'Zenith of Polaris', 80.64, 1.51),
       (8, 8, 'The Stranger in the Painting', 20.91, 43.47),
       (9, 9, 'Crime of the Pock-Marked Poet', 82.07, 32.20),
       (10, 10, 'Fatal Gun', 72.93, 22.25);
```

```sql
INSERT INTO OrderDetails(OrderID, ISBN, Quantity, BranchID, PublisherID)
VALUES (1, 1, 50, 1, 1),
       (2, 2, 40, 2, 2),
       (3, 3, 2, 3, 3),
       (4, 4, 10, 4, 4),
       (5, 5, 30, 5, 5),
       (6, 6, 30, 6, 6),
       (7, 7, 20, 7, 7),
       (8, 8, 10, 8, 8),
       (9, 9, 90, 9, 9),
       (10, 10, 5, 10, 10);
```

```sql
INSERT INTO Inventories(BookstoreID, ISBN, Quantity)
VALUES (1, 10, 10),
       (1, 2, 45),
       (1, 3, 20),
       (1, 4, 30),
       (1, 2, 10),
       (2, 8, 10),
       (4, 9, 20),
       (3, 6, 20),
       (8, 10, 10),
       (3, 4, 20),
       (8, 3, 40),
       (7, 3, 5),
       (6, 7, 5),
       (6, 2, 4),
       (5, 10, 1),
       (4, 1, 2),
       (9, 2, 4),
       (10, 1, 100),
       (3, 4, 50);
```

PART 5

> For each relation R created in your database, report the result of the following SQL statement:
> `SELECT COUNT(*) FROM R;`

Relation R	COUNT(*)
Authors	10
Books	10
Bookstores	10
Branches	14
Customers	10
HeadOffices	10
Inventories	19
Locations	10
OrderDetails	13
Orders	12
Publishers	10
ReaderInterests	10
Representatives	10