International University - VNUHCM

**School of Computer Science and Engineering**

-----------------

**Midterm Examination (online)**

Date: 20 April 2020. Duration: from April 20, 2020 to April 29, 2020

**Open-book,** electric devices **are allowed**

**SUBJECT: SOFTWARE ENGINEERING (IT076IU)**

|  |  |
| --- | --- |
| Approval | Lecturer: |
| Signature | Signature |



|  |  |
| --- | --- |
| Full name: | Full name: Nguyễn Thị Thúy Loan |
|  |  |
| Proctor 1 | Proctor 2 |
| Signature | Signature |

|  |  |
| --- | --- |
| Full name: | Full name: |

**STUDENT INFO**

**Student names: Trần Tô Quế Phương – ITITIU17001**

**Nguyễn Hải Nguyên – ITITIU17109**

**Lê Khuê Triền – ITITIU17066**

INSTRUCTIONS: the total of point is 100 (equivalent to 25% of the course)

|  |  |  |
| --- | --- | --- |
| Question | Points | Score |
|  |  |  |
| 1 | 40 |  |
|  |  |  |
| 2 | 20 |  |
|  |  |  |
| 3 | 35 |  |
|  |  |  |
| 4 | 15 |  |
|  |  |  |
| Total: (including 10 points bonus) | 110 |  |
|  |  |  |

* You have 10 days for solving problems from April 20, 2020 (8:30 AM) to April 29, 2020 (6:00 PM).

Good luck!

HCMC National University *Student Name:………...……………………….*

International University *Student ID:…………………………….*

------------

**Guidelines for students**

The following list are topics for a midterm test in Principles of Database Management courses, at the School of Computer Science and Engineering, International University, HCM National University.

Three students will be randomly selected to form a group. After formed, students in their group will work together on a given topic. Once again, a topic of this list will be randomly assigned to each group. During midterm test, students learn how to implement and analyze the database step by step. After two weeks, each group upload their result on the Blackboard IU.

After choosing a topic, you have to study it and follow tasks in the topic specification document carefully. Try to discuss with your teammates to gain a deeper understanding about your topic. During two weeks, you can clarify unclear issues with your teammates. Remember that all the hard works are yours; teammates only give you some brief advices based on what you have done.

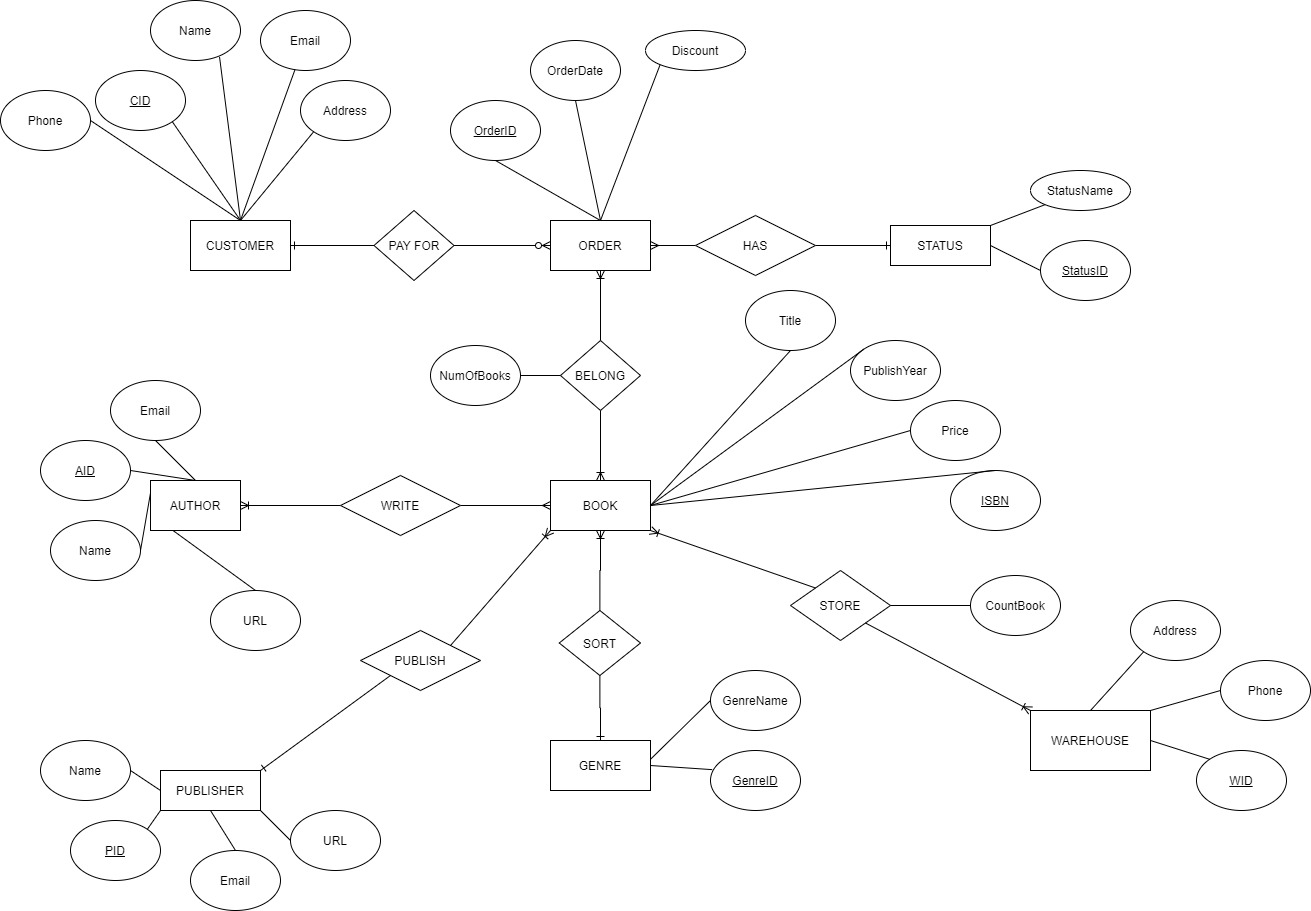
**Requirements**

1. (40 pts) Analyzing requirements and designing ER model.
2. (20 pts) Converting from ERD to relational model.
3. (35 pts) Using SQL to create a database with integrity constraints. Next, input data into the database. Then, using SQL query to update, manipulate, query, retrieve data from the database.
4. (15) Using relational algebra to perform data queries.

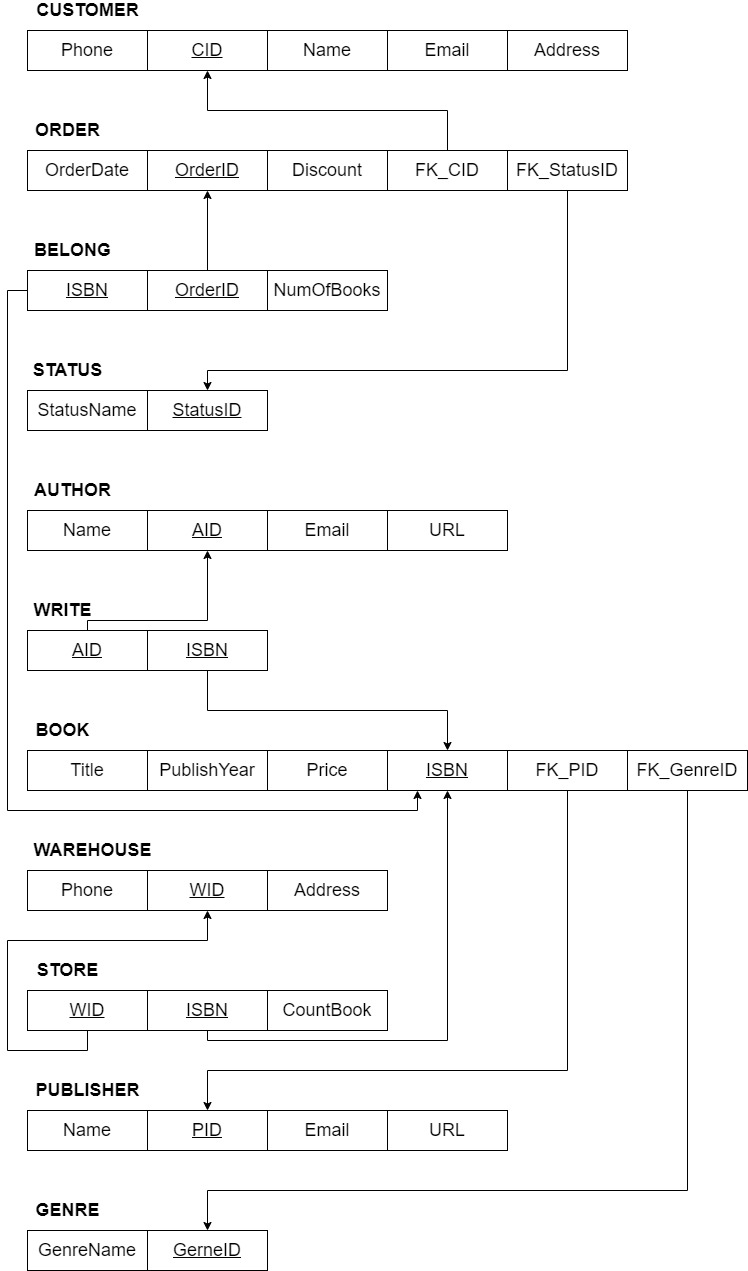
**TOPIC: Book Store**

**SOLUTION:**

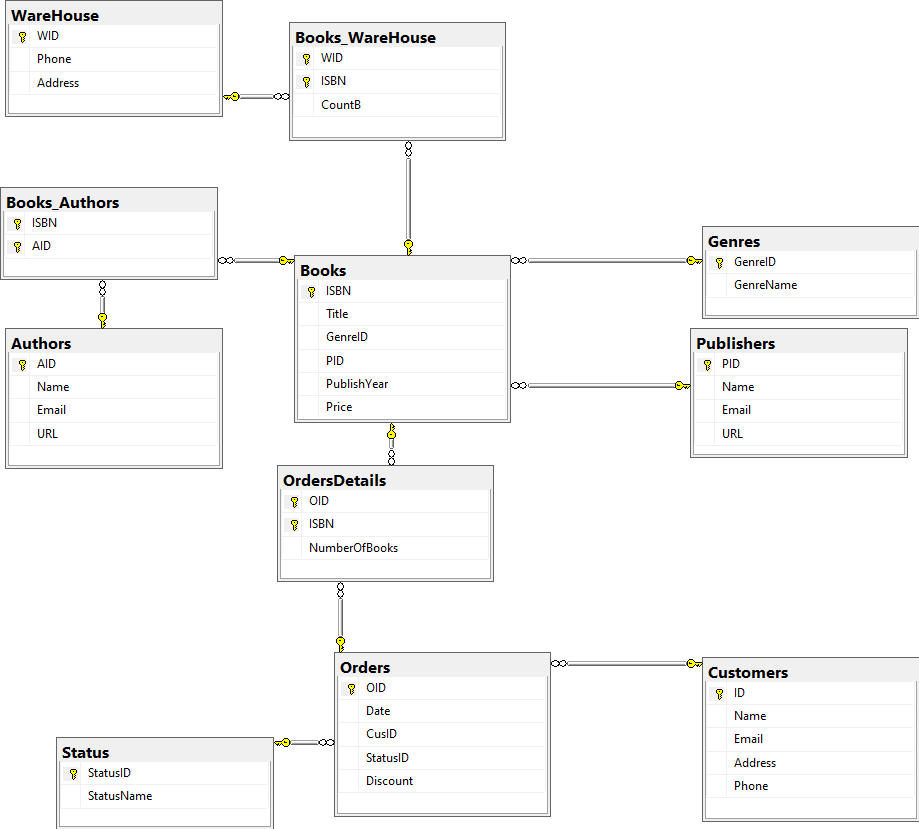
**1. (40 pts) Analyzing requirements and designing ER model.**



**2. (20 pts) Converting from ERD to relational model.**



**DATABASE DIAGRAM:**



**3. (35 pts) Using SQL to create a database with integrity constraints. Next, input data into the database. Then, using SQL query to update, manipulate, query, retrieve data from the database.**

**a. Create a database with integrity constraints**

CREATE DATABASE bookstore;

CREATE TABLE Status(

StatusID nvarchar(50) NOT NULL,

StatusName nvarchar(50) NOT NULL,

PRIMARY KEY (StatusID)

);

CREATE TABLE Genres(

GenreID nvarchar(50) NOT NULL,

GenreName nvarchar(50) NOT NULL,

PRIMARY KEY (GenreID)

);

CREATE TABLE Publishers(

PID nvarchar(50) NOT NULL,

Name nvarchar(50) NOT NULL,

Email nvarchar(50) NOT NULL,

URL nvarchar(50) NOT NULL,

PRIMARY KEY (PID)

);

CREATE TABLE WareHouse (

WID nvarchar(50) NOT NULL,

Phone nvarchar(50),

Address nvarchar(MAX) NOT NULL,

PRIMARY KEY (WID)

);

CREATE TABLE Customers(

ID nvarchar(50) NOT NULL,

Name nvarchar(50) NOT NULL,

Email nvarchar(50) NOT NULL,

Address nvarchar(max) NOT NULL,

Phone nvarchar(50) NOT NULL,

PRIMARY KEY (ID)

);

CREATE TABLE Authors(

AID nvarchar(50) NOT NULL,

Name nvarchar(50) NOT NULL,

Email nvarchar(50),

URL nvarchar(50) NOT NULL,

PRIMARY KEY (AID)

);

CREATE TABLE Books(

ISBN nvarchar(50) NOT NULL,

Title nvarchar(50) NOT NULL,

GenreID nvarchar(50) NOT NULL,

PID nvarchar(50) NOT NULL,

PublishYear varchar(55),

Price FLOAT NOT NULL,

PRIMARY KEY (ISBN),

/\*Relationship B(N-1)Genres\*/

CONSTRAINT FK\_Genres FOREIGN KEY (GenreID)

REFERENCES Genres(GenreID)

ON UPDATE CASCADE ON DELETE CASCADE,

/\*Relationship B(N-1)Publishers\*/

CONSTRAINT FK\_PID FOREIGN KEY (PID)

REFERENCES Publishers(PID)

ON UPDATE CASCADE ON DELETE CASCADE,

) ;

CREATE TABLE Books\_Authors(

ISBN nvarchar(50) NOT NULL,

AID nvarchar(50) NOT NULL,

/\*Relationship Orders(M-N)Books\*/

CONSTRAINT FK\_AID FOREIGN KEY (AID)

REFERENCES Authors(AID)

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT FK\_AISBN FOREIGN KEY (ISBN)

REFERENCES Books(ISBN)

ON UPDATE CASCADE ON DELETE CASCADE,

PRIMARY KEY (AID,ISBN)

);

CREATE TABLE Books\_WareHouse (

WID nvarchar(50) NOT NULL,

ISBN nvarchar(50) NOT NULL,

CountB INT CHECK (CountB>=0),

PRIMARY KEY(WID, ISBN),

CONSTRAINT FK\_WID FOREIGN KEY (WID)

REFERENCES WareHouse(WID)

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT FK\_WISBN FOREIGN KEY (ISBN)

REFERENCES Books(ISBN)

ON UPDATE CASCADE ON DELETE CASCADE,

);

CREATE TABLE Orders(

OID nvarchar(50) NOT NULL,

Date datetime NOT NULL,

CusID nvarchar(50) NOT NULL,

StatusID nvarchar(50) NOT NULL,

Discount INT DEFAULT 10,

PRIMARY KEY (OID),

/\*Relationship Orders(N-1)Status\*/

CONSTRAINT FK\_Status FOREIGN KEY (StatusID)

REFERENCES Status(StatusID)

ON UPDATE CASCADE ON DELETE CASCADE,

/\*Relationship Orders(N-1)Users\*/

CONSTRAINT FK\_Cus FOREIGN KEY (CusID)

REFERENCES Customers(ID)

ON UPDATE CASCADE ON DELETE CASCADE,

);

ALTER TABLE Orders ADD CONSTRAINT [DF\_Orders\_Date] DEFAULT (getdate()) FOR Date;

CREATE TABLE OrdersDetails(

OID nvarchar(50) NOT NULL,

ISBN nvarchar(50) NOT NULL,

NumberOfBooks bigint NOT NULL,

/\*Relationship Orders(M-N)Books\*/

CONSTRAINT FK\_OrdersID FOREIGN KEY (OID)

REFERENCES Orders(OID)

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT FK\_ISBN FOREIGN KEY (ISBN)

REFERENCES Books(ISBN)

ON UPDATE CASCADE ON DELETE CASCADE,

PRIMARY KEY (OID,ISBN)

);

**b. Input data into the database**

INSERT INTO Status (StatusID, StatusName) VALUES

(1, 'Pending'),

(2, 'Cancelled'),

(3, 'Completed');

INSERT INTO Genres(GenreID, GenreName) VALUES

(1,'Fantasy Fiction'),

(2,'Speculative Fiction'),

(3,'Science Fiction'),

(4,'Adventure Fiction'),

(5,'Crime Fiction'),

(6,'Literary Fiction'),

(7,'Horror Fiction'),

(8,'Historical Fiction'),

(9,'Mystery'),

(10,'Biography');

INSERT INTO Genres(GenreID, GenreName) VALUES

(11,'Self-help');

INSERT INTO Authors (AID, Name, Email, URL) VALUES

(1, 'J.R.R Tolken', 'jrrtolken@gmail.com', 'www.jrrtolken.com'),

(2, 'Tanith Lee', 'tanithlee@gmail.com', 'www.tanithlee.com'),

(3, 'Edgar Rice Burroughs', 'edgarRB@gmail.com', 'www.edgarRB.com'),

(4, 'Robert Ludlum', 'robertLudlum@gmail.com', 'www.robertLudlum.com'),

(5, 'James Clavel', 'jamesClavel@gmail.com','www.jamesClavels.com'),

(6, 'Venessa Walters', 'venessaWalters@gmail.com','www.venessaWalters.com'),

(7, 'Anne Rice', 'anneRice@gmail.com','www.anneRice.com'),

(8, 'Mark Stevenson', 'markStev@gmail.com','www.markStev.com'),

(9, 'Ribbly Scott', 'ribblyScott@gmail.com', 'www.ribblyScott.com'),

(10, 'Megan Miranda','meganMir@gmail.com','www.meganMir.com'),

(11, 'Sarah Mass','sarahMass@gmail.com','www.sarahMass.com'),

(12, 'J.D Vance','jdVance@gmail.com','www.jdVance.com'),

(13, 'Joe Hill','joeHill@gmail.com','www.joeHill.com'),

(14, 'Lee Child','leeChild@gmail.com','www.leeChild.com'),

(15, 'Jack Canfield','jackCanfield@gmail.com','www.jackCF.com'),

(16, 'Mark Victor Hansen','markVH@gmail.com','www.markVH.com'),

(17, 'Sharon J.Wohlmuth','sharonJW@gmail.com','www.sharonJW.com');

INSERT INTO Publishers (PID, Name, Email, URL) VALUES

(1,'Allen & Unwin','au@gmail.com','www.au.com'),

(2,'Macmilla','macmilla@gmail.com','www.macmilla.com'),

(3,'New English Library','nel@gmail.com','www.nel.com'),

(4,'Dial Press','dp@gmail.com','www.dp.com'),

(5,'Atheneum','atheneum@gmail.com','www.atheneum.com'),

(6,'Penguin','penguin@gmail.com','www.penguin.com'),

(7,'American Publishing Co','apc@gmail.com','www.apc.com'),

(8,'Morpheus','morpheus@gmail.com','www.morpheus.com'),

(9,'Paramount','paramount@gmail.com','www.paramount.com'),

(10,'H & R','hrPLC@gmail.com','www.hrPLC.com'),

(11,'Pearson Plc','pearson@gmail.com','www.pearson.com'),

(12,'Wiley Publisher','wp@gmail.com','www.wp.com'),

(13,'Scholastic','scholastic@gmail.com','www.scholastic.com'),

(14,'Simon & Schuster','ss@gmail.com','www.ss.com');

INSERT INTO Customers (ID, Name,Email, Address, Phone) VALUES

(1, 'Jakob', 'jakob@gmail.com', '12311 ABC', '0911812311'),

(2, 'Baines', 'baines@gmail.com', '12312 DEF', '0912312312'),

(3, 'Hanssen', 'hanssen@gmail.com', '12313 GHI', '0707112313'),

(4, 'Zakas', 'zakas@gmail.com', '12314 JKL', '0987612314'),

(5, 'Surewal', 'surewal@gmail.com', '12315 MNO', '0922212315'),

(6, 'Iversen', 'iversen@gmail.com', '12316 TUV', '0888112316'),

(7, 'Peter', 'peter@gmail.com', '12317 XYZ', '0998712317'),

(8, 'Dag', 'dag@gmail.com', '123 Nguyen Du', '0765412318'),

(9, 'Nicholas', 'nicholas@gmail.com', '1245 XVNT', '0723412319'),

(10, 'Michael', 'michael@gmail.com', '12 Hai Ba Trung', '0889912320');

INSERT INTO Books(ISBN, Title, GenreID, PID, PublishYear, Price ) VALUES

('0-332-3233-4','The Hobbit', 1 , 1 ,'1937' , '16.99'),

('0-3113-443-4','The Lord of the Rings : Fellowship of the ring',1,1,'2000','26.99'),

('0-113-473-8', 'The Lord of the Rings : The two towers',1,1,'2006', '20.99'),

('0-993-433-3', 'The Lord of the Rings : Return of the King',1,1,'2014', '29.99'),

('0-293-333-6', 'The Castle of Dark',2,2,'1987', '29.99'),

('0-023-179-4', 'The Winter Players',3,2,'1977', '29.99'),

('0-023-179-3', 'Tarzan and the forbidden city',4,3,'1914','28.99'),

('0-113-139-6', 'Tarzan of the Apes',4,3,'1999','19.99'),

('0-444-139-6', 'The Gemini Contenders',5,4,'2009','21.99'),

('0-433-439-6', 'Chancellor Manuscript',7,4,'1999','29.99'),

('0-430-131-6', 'Dragon Flight',3,5,'2007','16.95'),

('0-410-121-1', 'Summer Return',6,6,'2016', '16.99'),

('0-230-166-1', 'Tale of the Thief',7,6,'2016', '26.99'),

('0-090-881-1', 'Prince and the Pauper',8,7,'2011', '22.99'),

('0-22-121-1','Alien',3,8,'1996', '20.99'),

('0-413-331-1', 'Gone Girl',7,9,'2015', '29.99'),

('9-330-121-1', 'All the Missing Girls',9,10,'2016','29.99'),

('0-410-9921-1', 'Empire of Storms',1,11,'2006','28.99'),

('0-413-121-1', 'Hillbilly Elegy',10,12,'2012','19.99'),

('0-310-331-1', 'The Fire Man',1,13,'2016', '21.99'),

('5-110-121-1', 'Night School',9,11,'2015','29.99'),

('0-7303-1484-4', 'Chicken Soup for the Soul', '11', '14', '1997', '40.99');

INSERT INTO Books\_Authors (ISBN, AID) VALUES

('0-332-3233-4',1),

('0-3113-443-4',1),

('0-113-473-8',1),

('0-993-433-3',1),

('0-293-333-6',2 ),

('0-023-179-4',2),

('0-023-179-3',3),

('0-113-139-6',3),

('0-444-139-6',4 ),

('0-433-439-6',4),

('0-430-131-6',5 ),

('0-410-121-1',6 ),

('0-230-166-1',7 ),

('0-090-881-1',8 ),

('0-22-121-1', 9),

('0-413-331-1',5),

('9-330-121-1',10),

('0-410-9921-1',11 ),

('0-413-121-1',12),

('0-310-331-1',13 ),

('5-110-121-1',14 ),

('0-7303-1484-4',15 ),

('0-7303-1484-4',16 ),

('0-7303-1484-4',17 );

INSERT INTO WareHouse VALUES

('100','0888588599','1 Nguyen Hue');

INSERT INTO WareHouse VALUES

('101','0818093636','12 Dong Khoi');

INSERT INTO Books\_WareHouse VALUES

('100','0-332-3233-4',10),

('100','0-3113-443-4',15),

('100','0-113-473-8',20),

('100','0-993-433-3',11),

('100','0-293-333-6',12 ),

('100','0-023-179-4',20),

('100','0-023-179-3',30),

('100','0-113-139-6',31),

('100','0-444-139-6',4 ),

('100','0-433-439-6',40),

('100','0-430-131-6',55 ),

('100','0-410-121-1',67 ),

('100','0-230-166-1',75 ),

('100','0-090-881-1',81 ),

('100','0-22-121-1', 19),

('100','0-413-331-1',25),

('100','9-330-121-1',10),

('100','0-410-9921-1',1 ),

('100','0-413-121-1',120),

('100','0-310-331-1',13 ),

('100','5-110-121-1',141 ),

('101','0-7303-1484-4',150 );

INSERT INTO Books\_WareHouse VALUES

('100','0-7303-1484-4',160 );

INSERT INTO Orders (OID, Date, CusID, StatusID) VALUES

('201805041', '2018-05-04', '1', '1'),

('201906041', '2019-06-04', '2', '1'),

('202001021', '2020-01-02', '3', '3'),

('201907081', '2019-07-08', '4', '1'),

('201910121', '2019-10-12', '5', '1'),

('201801091', '2018-01-09', '6', '3'),

('201904081', '2019-04-08', '7', '2'),

('202002121', '2020-02-12', '8', '1'),

('202002122', '2020-02-12', '9', '1'),

('201910122', '2019-10-12', '10', '1');

INSERT INTO OrdersDetails (OID, ISBN, NumberOfBooks) VALUES

('201805041','0-332-3233-4' , '3'),

('201906041','0-3113-443-4' , '4'),

('202001021','0-410-9921-1', '3'),

('201907081','0-413-121-1' , '2'),

('201910121','5-110-121-1' , '3'),

('201801091','0-413-121-1' , '2'),

('201904081','0-293-333-6' , '1'),

('202002121','0-410-121-1', '3'),

('202002122','0-113-139-6' , '1'),

('201910122','0-433-439-6' , '1');

**c. Using SQL query to update, manipulate, query, retrieve data from the database**

**UPDATE:**

/\*UPDATE\*/

UPDATE Customers

SET Name = 'Alfred Schmidt', Address= 'Frankfurt'

WHERE Name='Jakob';

**DELETE:**

/\*DELETE\*/

DELETE FROM Books\_WareHouse WHERE Books\_WareHouse.CountB<2;

**MANIPULATE, RETRIEVE:**

/\*Display the Total Amount of each Order\*/

SELECT O.OID, (B.Price\*OD.NumberOfBooks) as Total

FROM Orders O, OrdersDetails OD, Books B

WHERE O.OID=OD.OID AND B.ISBN=OD.ISBN;

/\*If buy >= 2 books then customer will be discounted for each book from 3rd

Display order that will be discounted and the Amount of discount they receive\*/

SELECT O.OID,OD.NumberOfBooks,(NumberOfBooks-2)\*O.Discount AS Amt\_Discount

FROM Orders O, OrdersDetails OD, Books B

WHERE O.OID=OD.OID AND B.ISBN=OD.ISBN AND OD.NumberOfBooks>2;

/\*Display the list of books that has more than 100 books remaining\*/

SELECT B.Title, BW.CountB

FROM Books B, Books\_WareHouse BW

WHERE B.ISBN=BW.ISBN AND BW.CountB>100;

/\*INTERSECTION\*/

/\*B1 is the total list of books

B2 is that list of books that have Adventure Fiction Genre (GenreID=4)\*/

(SELECT \* FROM Books B1) INTERSECT (SELECT \* FROM Books B2 WHERE B2.GenreID=4)

/\*O1 is full list of orders

O2 is list of orders that be cancelled\*/

(SELECT \* FROM Orders O1) INTERSECT (SELECT \* FROM Orders O2 WHERE EXISTS(SELECT S.StatusName FROM Status S WHERE(O2.StatusID=S.StatusID AND S.StatusName='Cancelled')))

/\*UNION\*/

/\*B1 is the list of book that have Biography Genre (GenreID=10)

B2 is that list of books that have Adventure Fiction Genre (GenreID=4)\*/

(SELECT \* FROM Books B1 WHERE B1.GenreID=10) UNION (SELECT \* FROM Books R2 WHERE R2.GenreID=4)

/\*O1 is list of orders that be cancelled

O2 is list of orders that are pending\*/

(SELECT \* FROM Orders O1, Status WHERE O1.StatusID=Status.StatusID AND Status.StatusName='Cancelled') UNION (SELECT \*FROM Orders O2, Status WHERE O2.StatusID=Status.StatusID AND Status.StatusName='Pending')

/\*SELECTION\*/

/\*Order status is “Completed”:\*/

SELECT \* FROM Orders

WHERE EXISTS(

SELECT Status.StatusName

FROM Status

WHERE Orders.StatusID=Status.StatusID AND Status.StatusName='Completed');

/\* Books’ price is >20: \*/

SELECT \* FROM Books

WHERE Books.Price>20;

/\*PROJECTION\*/

/\*List of Order’s ID and Date \*/

SELECT Orders.OID, Orders.Date

FROM Orders;

/\*List of Authors’ Name and URL:\*/

SELECT Authors.Name, Authors.URL

FROM Authors;

/\*SET DIFFERENCE\*/

/\*O1 is full list of orders

O2 is list of orders that are cancelled

Display list of orders that are not cancelled\*/

(SELECT \* FROM Orders O1) EXCEPT (SELECT \* FROM Orders O2 WHERE EXISTS(SELECT S.StatusName FROM

Status S WHERE(O2.StatusID=S.StatusID AND S.StatusName='Cancelled')))

/\*CROSS-PRODUCT\*/

/\*Combine each Order with its Corresponding OrderDetail\*/

SELECT \*

FROM Orders

CROSS JOIN OrdersDetails

WHERE Orders.OID=OrdersDetails.OID;

/\*JOIN\*/

/\*Name of Books that are Horror fiction: \*/

SELECT B.Title, G.GenreName

FROM Books B

INNER JOIN Genres G ON B.GenreID=G.GenreID AND G.GenreName='Horror Fiction';

/\*DIVISION\*/

/\*Find Books that be contained in both WareHouse\*/

SELECT \* FROM Books\_WareHouse

WHERE ISBN not in ( SELECT ISBN FROM ( (SELECT ISBN, WID FROM (select WID from WareHouse) as p

cross join

(select distinct ISBN from Books\_WareHouse) as sp)

EXCEPT

(SELECT ISBN, WID FROM Books\_WareHouse)) AS r );

/\*RENAMING\*/

EXEC sp\_rename 'Orders.OID', 'OrderID', 'COLUMN';

**4. (15) Using relational algebra to perform data queries.**

**Intersection:**

* B1 is the total list of books

B2 is that list of books that have Adventure Fiction Genre (GenreID=4):

B1 = ∏­­­ISBN,Title,GenreID,PID,PublishYear,Price­(Books)

B2 = σGenreID=’4’ (Books)

B1 ⋂ B2

* O1 is full list of orders

O2 is list of orders that be cancelled:

O1 = ∏­­­OID,Date,CusID,StatusID,Discount (Orders)

O2 = ∏­­­OID,Date,CusID,StatusID,Discount(σOrders.StatusID=Status.StatusID ∧ StatusName=’Cancelled’(Orders X Status))

O1 ⋂ O2

**Union:**

* B1 is the list of book that have Biography Genre (GenreID=10)

B2 is that list of books that have Adventure Fiction Genre (GenreID=4)

B1 = σGenreID=’10’ (Books)

B2 = σGenreID=’4’ (Books)

B1 ∪ B2

* O1 is list of orders that be cancelled

O2 is list of orders that are pending

O1= ∏­­­OID,Date,CusID,StatusID,Discount(σOrders.StatusID=Status.StatusID ∧ StatusName=’Cancelled’(Orders X Status))

O2 = ∏­­­OID,Date,CusID,StatusID,Discount (σOrders.StatusID=Status.StatusID ∧ StatusName=’Pending’(Orders X Status))

O1 ∪ O2

**Selection:**

* Order status is “Completed”:

∏­­­OID,Date,CusID,StatusID,Discount (σOrders.StatusID=Status.StatusID ∧ StatusName=‘Completed’(Orders X Status))

* Books’ price is > 20:

σPrice>20(Books)

**Projection:**

* List of Order’s ID and Date:

∏OID,Date(Orders)

* List of Authors’ Name and URL:

∏Name,URL(Authors)

**Set difference:**

* O1 is full list of orders
* O2 is list of orders that are cancelled
* Display list of orders that are not cancelled

O1 = ∏­­­OID,Date,CusID,StatusID,Discount (Orders)

O2 = ∏­­­OID,Date,CusID,StatusID,Discount(σOrders.StatusID=Status.StatusID ∧ StatusName=’Cancelled’(Orders X Status))

O1 - O2

**Cross-product:**

* Combine each Order with its Corresponding OrderDetail:

σOrders.OID=OrdersDetails.OID (Orders X OrdersDetails)

**Join:**

* Name of Books that are Horror fiction:

∏Title,GenreName(σGenre.GenreName='Horror Fiction'(Books ⋈ Genres))

**Division:**

* Find Books that be contained in both WareHouse:

Books\_WareHouse/WareHouse = ∏ISBN(Books\_WareHouse) - ∏ISBN((∏ISBN(Books\_WareHouse) x WareHouse – Books\_WareHouse)

**Rename:**

* In table Orders, rename column OID to OrderID

ρ OrderID (OID)