








Lab 3: Generation of Normalized Database Schema

SC2207 Introduction to Databases

Dr. W. K. Ng

March 7, 2023

Team 1

Full Name	Individual Contribution to Lab 3 Submission	Percentage of Contribution	Signature
Anusree Sreekumar	Analyzed viability of FDs using potential queries + propose relevant edits	14.3%	
Aruna D/O Vadivelu	Construction of multiple components	14.3%	
Ho Wei Hao	Group Leader Directed the logic of the relation schema to allow for smooth flow of the relation schema	14.3%	
Kevin Yok	Helped to prepare part of the relation schema	14.3%	
Liv Tan Ker Jin	Participated in the development of raw form of schemas and proposed relevant edits required for the functional dependencies	14.3%	
Ng Si En	Helped with development of different aspects of relation schema and answered queries raised by members	14.3%	
Shi Xinyu	Represented relationships between different entity sets	14.3%	

Contents

1 Introduction	3
2 Relation Schema	4
3 Summary	12
4 Appendix A	13

1. Introduction

This report presents the set of relational schema converted from the ER diagram from lab 1. Our group's revised ER diagram is detailed in Appendix A.

For subclass relations, we followed the ER approach as it avoids having NULL values and avoids having an excessive number of tables.

If a relation schema is not in 3NF, we first attempt to perform BCNF decomposition. If the BCNF decomposition is functional dependency preserving, we will stop there. Else, we proceed to perform 3NF decomposition. This is due to BCNF decomposition being less redundant than 3NF decomposition making BCNF decomposition preferable as the primary approach.

2. Relation Schema

We will be using the following format for each relation schema:

R1(A, B, C, D)
Keys: AB, AD
Primary Key: AB
FDs: $(AB) \rightarrow CD$, $A \rightarrow D$

The relation is in BCNF/3NF else perform decomposition until all tables are in BCNF/3NF:

- a) Identify an FD that violates BCNF
- b) Compute the closure of this FD
- c) Decompose the table into 2 subtables

Thought process:

- Justification of why we include or exclude certain keys
- Explanation of FDs: To explain why our FDs and choice of keys make sense.

In addition, each relation schema is capitalised and each attribute is in bold. We now present the generation of our database schema.

1. PUBLICATION(PubID, Publisher, Year)

Key(s): PubID

Primary Key: PubID

FDs: PubID \rightarrow Publisher, Year

The relation is in BCNF in addition to being in 3NF, because the LHS of every non-trivial FD contains a key.

Thought process:

- Explanation of FDs: It is logical to say that if we know the **PubID**, we know the **Publisher** and the **Year** of publication.

2. BOOKS(PubID, Title)

Key(s): PubID

Primary Key: PubID

FDs: PubID \rightarrow Title

The relation is in BCNF in addition to being in 3NF, because the LHS of every non-trivial FD contains a key.

Thought process:

- Explanation of FDs: It is logical to say that if we know the **PubID**, we know the **Title** of the book.

3. MAGAZINES(PubID, Title, Issue)

Key(s): PubID

Primary Key: PubID

FDs: PubID \rightarrow Title, Issue

The relation is in BCNF in addition to being in 3NF, because the LHS of every non-trivial FD contains a key.

Thought process:

- Explanation of FDs: It is logical to say that If we know the **PubID**, we know the **Title** and the **Issue** of the magazines.

4. STOCKS-IN-BOOKSTORE(PubID, BookstoreID, StockID, Stock-Qty, Stock-Price)
Key(s): (PubID, Bookstore ID), (StockID, BookstoreID)
FDs: (PubID, BookstoreID) \rightarrow StockID. StockID \rightarrow Stock-Qty, Stock-Price. (StockID, BookstoreID) \rightarrow PubID.

The relation is not in 3NF. We first attempt a BCNF decomposition.

The BCNF decomposition is as follows:

- a) Identify an FD that violates BCNF:
StockID \rightarrow Stock-Qty, Stock-Price
- b) Compute the closure of this FD:
 $\{\text{StockID}\}^+ = \{\text{Stock ID, Stock-Qty, Stock-Price}\}$
- c) Break the table into SB1 and SB2:
SB1(StockID, Stock-Qty, Stock-Price)
Key(s): StockID
Primary Key: StockID
FDs: StockID \rightarrow Stock-Qty, Stock-Price. Thus SB1 is in BCNF.

SB2(PubID, BookstoreID, StockID)
Key(s): (PubID, BookstoreID), (StockID, BookstoreID)
Primary Key: (PubID, BookstoreID)
FDs: (PubID, BookstoreID) \rightarrow StockID and (StockID, BookstoreID) \rightarrow PubID. Thus SB2 is in BCNF.

The original functional dependencies are preserved through BCNF decomposition of STOCKS-IN-BOOKSTORE into SB1 and SB2. Hence, 3NF decomposition is not needed.

Thought Process:

- **StockID** is not included as a key as it is only unique within the particular Bookstore.
- **PubID** and **BookstoreID** are added as keys of STOCKS-IN-BOOKSTORE as STOCKS-IN-BOOKSTORE is a weak entity set supported by PUBLICATION and BOOKSTORE, and is identifiable by the keys of PUBLICATION (**PubID**), BOOKSTORE (**BookstoreID**) and **StockID**.
- (**StockID, BookstoreID**) is also a key because each **PubID** is uniquely identified by **StockID** within the **BookstoreID**.
- Explanation of FDs: It is logical to say that if we know the publication ID and the bookstore ID, we know the stock ID for the stocks in the Bookstore. Furthermore, It is also logical to say that if we know the stock ID, we know the stock quantity and stock price for the stocks in Bookstore.

5. PRICE-HISTORY(Start-date, End-date, StockID, Price)

Key(s): (Start-date, End-date, StockID)

Primary Key: (Start-date, End-date, StockID)

FDs: (Start-date, End-date, StockID) \rightarrow Price

The relation is in BCNF, as the LHS of all non trivial FDs contain the key.

Thought Process:

- **StockID** is added as a key of PRICE-HISTORY since PRICE-HISTORY is a weak entity set supported by STOCKS-IN-BOOKSTORE. This means that attributes of price history (**Price**) are identifiable by the key of STOCKS-IN-BOOKSTORE (which is **StockID**), **Start-date** and **End-date**.
- We exclude **Price** as a key since given the **Start-date**, **End-date** and **StockID**, we definitely can find out the price of the particular stock in a bookstore.

6. ITEMS-IN-ORDERS(ItemID, OrderID, Item-Price, Item-Qty, Delivery-date, StockID, Comment, Rating, Date-time, CustomerID)

Key(s): (ItemID, OrderID)

Primary Key: (ItemID, OrderID)

FDs: (ItemID, OrderID) \rightarrow Item-Price, Item-Qty, Delivery-Date, StockID, Comment, Rating, Date-time, CustomerID

OrderID \rightarrow CustomerID

The relation is not in BCNF, as the LHS of OrderID \rightarrow CustomerID does not contain the key (ItemID, OrderID).

The BCNF decomposition is as follows:

- d) Identify an FD that violates BCNF:

OrderID \rightarrow CustomerID

- e) Compute the closure of this FD:

$\{\text{OrderID}\}^+ = \{\text{OrderID}, \text{CustomerID}\}$

- f) Break the table into IO1 and IO2:

IO1(OrderID, CustomerID)

Key(s): OrderID

FDs: OrderID \rightarrow CustomerID. Thus IO1 is in BCNF.

IO2(OrderID, ItemID, StockID, Item-Price, Item-Qty, Delivery-date, Comment, Rating, Date-time)

Key(s): (ItemID, OrderID)

FDs: (ItemID, OrderID) \rightarrow Item-Price, Item-Qty, Delivery-Date, StockID, Comment, Rating, Date-time. Thus IO2 is in BCNF.

The original functional dependencies are preserved through BCNF decomposition of Items-In-Orders into IO1 and IO2. Hence, 3NF decomposition is not needed.

Thought Process:

- **OrderID** is added as a key of ITEMS-IN-ORDER since ITEMS-IN-ORDERS is a weak entity set supported by STOCKS-IN-BOOKSTORE. This means that entities of ITEMS-IN-ORDERS (which are the items) are identifiable by the key of ORDERS (which is **OrderID**), and the **ItemID** itself.
- **Comment**, **Rating**, **Date-time** and **CustomerID** are included as attributes, representing the Many-to-One relationship "Feedback"
- Take note that ITEMS-IN-ORDERS is not a weak entity set of STOCKS-IN-BOOKSTORE and instead, is a many-to-one relationship. It is considered that items in orders are identifiable by (**OrderID**, **ItemID**). We do not need to include **StockID** into the composite key.
- Explanation of FDs: It is logical to say that If we know the (**ItemID**, **OrderID**), we know the **Item-Price**, **Item-Qty**, **Delivery-Date**, **StockID**, **Comment**, **Rating**, **Date-time**, and **CustomerID** of the items in orders. Since ORDER and CUSTOMER are in a many-to-one relationship, it is logical to say that If we know the **OrderID**, we will know the **CustomerID**.

7. ORDERSTATUS(OrderID, Date, ItemID, State)

Key(s): (OrderID, Date, ItemID)

Primary Key: (OrderID, Date, ItemID)

FDs: (OrderID, Date, ItemID) → State

This relation is in BCNF as the LHS of the non-trivial FD (OrderID, Date, ItemID) → State contains the key (OrderID, Date, ItemID). Since the relation is in BCNF, it is in 3NF as well.

Thought Process:

- **OrderID** and **ItemID** are added as a key of ORDERSTATUS as ORDERSTATUS is a weak entity set supported by ITEMS-IN-ORDERS, while ITEMS-IN-ORDERS is a weak entity set supported by ORDERS. ORDERSTATUS is identifiable by the key of ITEMS-IN-ORDERS (**ItemID**), key of ORDERSTATUS (**OrderID**) and **Date**.
- Explanation of FD: It is logical to say that if we know the **Date**, **OrderID** and **ItemID**, we know the **State** of the status of the items in the orders.

8. BOOKSTORE(BookstoreID)

Key(s): BookstoreID

Primary key: BookstoreID

FD: BookstoreID \rightarrow BookstoreID

This relation is in BCNF as there is no non-trivial FD.

Thought Process:

- **BookstoreID** is the only key.
- Explanation of FD: By the axiom of reflexivity, it is logical to say that **BookstoreID** determines itself.

9. COMPLAINTS(ComplaintID, Text, Filed-date-time, CustomerID, EmployeeID, Handled-date-time)

Key(s): ComplaintID

Primary key: ComplaintID

FD: ComplaintID \rightarrow Text, Filed-date-time, CustomerID, EmployeeID, Handled-date-time

The relation is in BCNF, because the LHS of all non trivial FDs contains the key.

Because the relation is in BCNF, the relation is in 3NF.

Thought Process:

- **Text**, **Filed-date-time** and **ComplaintID** are attributes of COMPLAINTS
- Additionally, it has a many-to-one relationship with CUSTOMERS. Hence, the key of customers, which is **CustomerID**, has to be included in the attributes of COMPLAINTS as well.
- COMPLAINTS has also a many-to-one relationship with EMPLOYEES. The key of EMPLOYEES i.e. **EmployeeID**, should be included in the attribute as well.
- Since 'HANDLED' is a relationship between COMPLAINTS and EMPLOYEES, the attribute of 'HANDLED', which is 'Handled-date-time', should be included as an attribute under COMPLAINTS as well.
- Explanation of ID: Since **ComplaintID** is the only key, it is logical to say that **ComplaintID** determines all other attributes.

10. COMPLAINTS-ON-BOOKSTORE(ComplaintID, BookstoreID)

Key(s): ComplaintID

Primary Key: ComplaintID

FD: ComplaintID \rightarrow BookstoreID

This relation is in BCNF as the LHS of the non-trivial FD, **ComplaintID** \rightarrow **BookstoreID** is the key **ComplaintID**. Since the relation is in BCNF, it is in 3NF.

Thought Process:

- Since COMPLAINTS-ON-BOOKSTORE is a subclass of COMPLAINTS, the key of COMPLAINTS, **ComplaintID**, is the key for COMPLAINTS-ON-BOOKSTORE as well.
- Since COMPLAINTS-ON-BOOKSTORE and BOOKSTORE is a many-to-one relation, the key of BOOKSTORE, **BookstoreID**, is an attribute of COMPLAINTS-ON-BOOKSTORE.
- Explanation of FDs: It is logical to say that if we know the **ComplaintID**, we will know the bookstore that the complaint is addressed to as COMPLAINTS-ON-BOOKSTORE is a subclass of COMPLAINTS.

11. COMPLAINTS-ON-ORDERS(ComplaintID, OrderID)

Key(s): ComplaintID

Primary Key: ComplaintID

FD: ComplaintID \rightarrow OrderID

This relation is in BCNF as the LHS of the non-trivial FD, **ComplaintID** \rightarrow **OrderID** is the key **ComplaintID**. Since the relation is in BCNF, it is in 3NF.

Thought Process:

- Since COMPLAINTS-ON-ORDERS is a subclass of COMPLAINTS, the key of COMPLAINTS, which is **ComplaintID**, should be a key for the subclass as well.
- Since COMPLAINTS-ON-ORDERS is a many-to-one relation with ORDERS, the key of ORDERS, **OrderID**, is an attribute for COMPLAINTS-ON-ORDERS.
- Explanation of FD: It is logical to say that if we know the **ComplaintID**, we will know the **OrderID** of the order placed on the book as COMPLAINTS-ON-ORDERS is a subclass of COMPLAINTS.

12. EMPLOYEES(EmployeeID, Name, Salary)

Key: EmployeeID

Primary Key: EmployeeID

FD: EmployeeID \rightarrow Name, Salary

Explanation: The relation is in BCNF as LHS contains the key, hence the relation is in 3NF.

Thought Process:

- **EmployeeID**, **Name** and **Salary** are all attributes of EMPLOYEES. Since **EmployeeID** is the key, **EmployeeID** determines **Name** and **Salary**.
- Explanation of FD: It is logical to say that if we know the **EmployeeID**, we will know the **Name** and **Salary** of the employees.

13. COMPLAINTSTATUS(Date, ComplaintID, State)

Key(s): (Date, ComplaintID)

Primary Key: (Date, ComplaintID)

FDs: (Date, ComplaintID) \rightarrow State

This relation is in BCNF as the LHS of all non-trivial FDs contain the key. Since the relation is in BCNF, it is in 3NF.

Thought process:

- **State** is the attribute of STATUS, and **Date** is the key of STATUS
- Since STATUS is a weak entity of COMPLAINTS, the key of COMPLAINTS, **ComplaintID**, should be included as a key in STATUS
- Explanation of FDs: With both the **Date** and **ComplaintID**, it makes sense to say the **State** of the complaint can be obtained.

14. ORDERS(OrderID, Date-time, Shipping-address, CustomerID)

Key(s): OrderID

Primary Key: OrderID

FDs: OrderID \rightarrow Date-time, Shipping-address, CustomerID

This relation is in BCNF as the LHS of all non-trivial FDs contain the key. Since the relation is in BCNF, it is in 3NF.

Thought Process:

- **CustomerID** is added as an attribute, representing the Many-to-One relationship
- **OrderID** will determine the date and time of the order and the shipping address of the order itself.
- In addition, the **OrderID** is able to trace the **CustomerID** due to the many to one relationship.
- A customer may have more than one **Shipping-address**, hence the FD **CustomerID** \rightarrow **Shipping-address** is invalid

15. CUSTOMERS(CustomerID, Name)

Key(s): CustomerID

Primary key: CustomerID

FDs: CustomerID \rightarrow Name

This relation is in BCNF as the LHS of all non-trivial FDs contain the key. Since the relation is in BCNF, it is in 3NF.

Thought Process:

- It is logical to say that if we know **CustomerID** we can find the customer's **Name** as these are all attributes of the CUSTOMERS.

3 Summary

IO1(OrderID, CustomerID) is a subset of Orders (OrderID, Date-time, Shipping-address, CustomerID). Hence IO1 is not needed. We then rename IO2(OrderID, StockID, Item-Price, Item-Qty, Delivery-date, Comment, Rating, Date-time) to Items-in-Orders(OrderID, StockID, Item-Price, Item-Qty, Delivery-date, Comment, Rating, Date-time)

The summary of final normalised tables, with primary keys underlined is as follows.

1. Publication (PubID, Publisher, Year)

```
CREATE TABLE Publication (  
    PubID int NOT NULL,  
    Publisher VarChar(30) NOT NULL,  
    Year int NOT NULL,  
    PRIMARY KEY (PubID)  
)
```

2. Books (PubID, Title)

```
CREATE TABLE Books(  
    PubID int foreign key (PubID) references Publication (PubID) ON DELETE  
    CASCADE ,  
    Title VarChar(30) NOT NULL  
)
```

3. Magazines (PubID, Title, Issue)

```
CREATE TABLE Magazines (  
    PubID int foreign key (PubID) references Publication (PubID) ON DELETE  
    CASCADE,  
    Title VarChar(30) NOT NULL,  
    Issue int NOT NULL  
)
```

4. SB1(StockID, Stock-Qty, Stock-Price)

```
- CREATE TABLE SB1(  
    StockID int NOT NULL,  
    Stock_Qty int NOT NULL,  
    Stock_Price int NOT NULL,  
    PRIMARY KEY(StockID)  
)
```

5. SB2(PubID, BookstoreID, StockID)

```
- CREATE TABLE SB2 (  
    PubID int FOREIGN KEY (PubID) REFERENCES Publication(PubID) NOT NULL,  
    BookstoreID int FOREIGN KEY (BookstoreID) REFERENCES  
    Bookstore(BookstoreID) NOT NULL,  
    StockID int FOREIGN KEY (StockID) REFERENCES SB1(StockID) NOT NULL,  
    PRIMARY KEY (PubID, BookstoreID)  
)
```

6. Price-History (Start-date, End-date, StockID, Price)

```
CREATE TABLE PriceHistory(  
    Startdate date NOT null,  
    Enddate date not null,  
    StockID int foreign key (StockID) references SB1(StockID) NOT NULL,  
    Price int NOT NULL,  
    PRIMARY KEY (Startdate, Enddate, StockID)  
)
```

7. Items-in-Orders(OrderID, ItemID, StockID, Item-Price, Item-Qty, Delivery-date, Comment, Rating, Date-time)

```
CREATE TABLE ItemsInOrders(  
    OrderID int foreign key (OrderID) references Orders (OrderID) NOT NULL,  
    StockID int foreign key (StockID) references SB1(StockID) NOT NULL,  
    ItemID int not null,  
    ItemPrice int NOT NULL,  
    ItemQty int NOT NULL,  
    Deliverydate date,  
    Comment varchar(200),  
    Rating int,  
    DateTime datetime NOT NULL,  
    PRIMARY KEY (OrderID, ItemID)  
)
```

8. Orderstatus(OrderID, Date, ItemID, currentstatus)

```
CREATE TABLE OrderStatus (  
    OrderDate date,
```

```

itemID int NOT null,
currentstatus varchar(30) DEFAULT 'Being Processed' NOT null,
orderID int foreign key (orderID) references Orders (OrderID),
PRIMARY KEY (orderdate, itemID, orderID)
)

```

9. **Bookstore(BookstoreID)**

```

- CREATE TABLE (
    BookstoreID int NOT NULL,
    PRIMARY KEY (BookstoreID)
)

```

10. **Complaints(ComplaintID, Text, Filed-date-time, CustomerID, EmployeeID, Handled-date-time)**

```

- CREATE TABLE Complaints (
    ComplaintID int primary key, textcontent varchar(500) NOT NULL,
    filedatetime datetime not null,
    customerID int foreign key(customerID) references customers(customerID) not null,
    employeeID int foreign key(employeeID) references employees(employeeID) not
    null,
    Handledatetime datetime not null
)

```

11. **Complaints-On-Bookstore (ComplaintID, BookstoreID)**

```

- CREATE TABLE ComplaintsOnBookstore (
    ComplaintID int foreign key (ComplaintID) references complaints(ComplaintID),
    BookstoreID int foreign key (BookstoreID) references Bookstore(BookstoreID),
    PRIMARY KEY (ComplaintID))

```

12. **Complaints-On-Orders (ComplaintID, OrderID)**

```

- CREATE TABLE ComplaintsOnOrders (
    ComplaintID int foreign key (ComplaintID) references Complaints(ComplaintID),
    OrderID int foreign key (OrderID) references Orders(OrderID),
    primary key(complaintID))

```

13. **Employees(EmployeeID, Name, Salary)**

```

CREATE TABLE Employees(
    EmployeeID int NOT NULL,
    Name varchar(50) NOT NULL,

```

```

Salary int NOT NULL
PRIMARY KEY (EmployeeID)
)

```

14. ComplaintStatus(Date, State, ComplaintID)

```

CREATE TABLE ComplaintStatus(
    ComDate date NOT NULL,
    Complaintstatus varchar(50) NOT NULL DEFAULT 'Pending',
    ComplaintID int FOREIGN KEY (ComplaintID) REFERENCES
Complaints(ComplaintID)
    PRIMARY KEY (ComplaintID, ComDate)
)

```

15. Orders (OrderID, Date-time, Shipping-address, CustomerID)

```

CREATE TABLE Orders (
    OrderID int primary key NOT NULL,
    Date_time datetime NOT NULL,
    Shipping_address varchar(80) NOT NULL,
    CustomerID int FOREIGN KEY(CustomerID) REFERENCES
Customers(CustomerID) ON UPDATE CASCADE
)

```

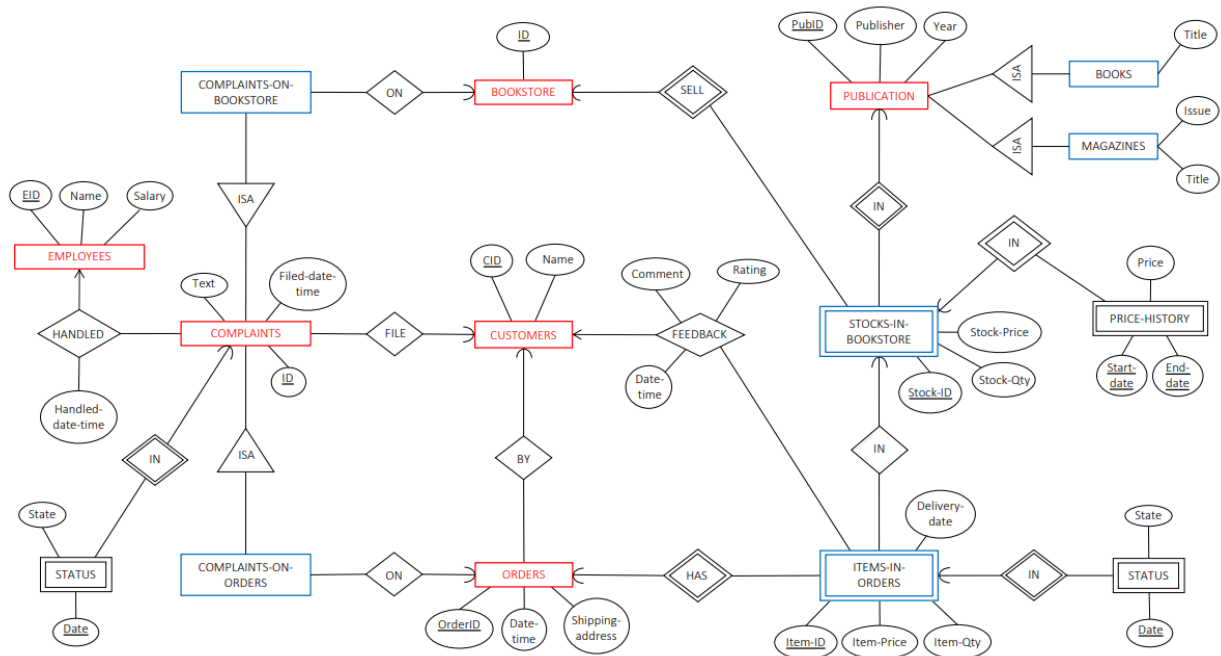
16. Customers(CustomerID, Name)

```

- CREATE TABLE Customers(
    CustomerID int NOT NULL,
    Name varchar(50) NOT NULL,
    PRIMARY KEY (CustomerID)
)

```

4 Appendix A



INSERT INTO SB2(PubID, BookstoreID, StockID)

VALUES

(76854921, 197442,653241),
 (36485721, 197442,354681),
 (76854921, 223296, 767281),
 (36485721, 223296, 123456),
 (64587236, 290976, 345276),
 (98541276, 290976, 546387),
 (64587236, 353629, 837261),
 (98541276, 353629, 657482),
 (46548732, 457705, 456789),
 (76854921, 457705,123459);

197442
 223296
 290976
 353629
 457705

76854921
 36485721

64587236
98541276
46548732

Pub ID, BookstoreID, StockID

34567890	197442	653241
34567890	223296	123456
54876953	353629	657482
54876953	457705	123459
97772093	197442	354681
97772093	223296	767281
23154879	290976	345276
23154879	353629	837261
35468921	290976	546387
24587692	457705	456789

INSERT INTO ItemsInOrders(OrderID, ItemID, StockID, ItemPrice, ItemQty, Deliverydate, Comment, Rating, DateTime)

VALUES

(96525998, 03034022, 653241, 25, 1, '2022-06-28', 'Amazing Book!', 8, '2022-06-28 10:22:03'),
(96525998, 123456, 20, 1, '2022-06-28', 'Slightly Damaged:', 3, '2022-06-28 10:27:03')
(34569002, 49030321, 354681, 18, 1, '2022-06-30', 'Perfect as per usual. Thanks!', 10, '2022-06-30 10:27:03')
(36912158, 04049495, 653241, 25, 1, '2022-08-05', 'Ok', 7, '2022-08-06 10:27:03')
(36912158, 99303494, 837261, 15, 2, '2022-08-05', 'Not Bad', 7, '2022-08-06 10:33:03')
(36912158, 48494949, 767281, 13, 1, '2022-08-05', 'Can be better', 4, '2022-08-06 10:35:03')
(20220020, 03938434, 657482, 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03')
(20220020, 29393939, 767281, 28, 1, '2022-08-01', 'Great book!', 9, '2022-08-01 10:38:03')
(20494850, 39494505, 345276, 9, 1, '2022-07-01', 'Nice!', 10, '2022-07-01 10:38:03')

INSERT INTO OrderStatus (OrderDate, itemID, currentstatus, orderID)

*VALUES ('2022-06-12', 03034022, 'Out for delivery', 96525998),
('2022-06-12', 39940203, 'Delivered', 34569002),
('2022-05-30', 49030321, 'Delivered', 36912158),
('2022-05-28', 39494505, 'Delivered', 20494850),
('2022-06-10', 27817208, 'Out for delivery', 98392332),
('2022-06-05', 29393939, 'Delivered', 20220020)*

Orders (OrderID, Date-time, Shipping-address, CustomerID)

~~(96525998, 2022-06-23 10:22:03, '5 Tyersall Park', 13254769)~~
~~(34569002, 2022-06-25 15:22:03, '29 Bukit Panjang Ring Road', 25146358)~~
~~(20494850, 2022-06-25 13:21:03, '31 Jurong East Road', 32154769)~~
~~(20220020, 2022-07-21 09:21:03, '3 Pasir Panjang Lane', 34126587)~~
~~(17872689, 2022-08-28 22:21:03, '550 Jurong West', 41754398)~~
~~(24681011, 2022-08-15 13:34:03, '33 Dempsey Hill', 54876213)~~
~~(36912158, 2022-07-31 18:57:03, '10 Marina Boulevard', 65412879)~~
~~(13579113, 2022-08-15 03:21:03, '33 Dempsey Hill', 75632148)~~
~~(87829188, 2022-08-15 05:21:03, '33 Woodlands Grove', 86214357)~~
~~(28494905, 2022-08-15 23:21:03, '20 Admiralty Road', 98541275)~~

13254769	Myles Henson
25146358	Jaylyn Bernard
32154769	Jaycee Alvarez
34126587	Paula Cox
41754398	Natasha Freeman
54876213	Levi Raymond
65412879	Gabriela Gomez
75632148	Derek Wang
86214357	Jane Tate
98541275	Leroy Edwards
12349045	Jesse Lee

INSERT INTO ItemsInOrders
 Values

(20220020, 111111111,653241 , 22, 1, '2022-08-01', 'Soso',5, '2022-08-01 10:35:03'),
 (20220020, 222222222,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03'),
 (20220020, 333333333,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03'),
 (20220020, 444444444,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03'),
 (20220020, 555555555,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03'),
 (20220020, 666666666,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03'),
 (20220020, 777777777,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03'),
 (20220020, 999999999,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03'),
 (20220020, 656666667,653241 , 22, 1, '2022-08-01', 'Soso', 5, '2022-08-01 10:35:03')

```
SELECT PubID  
FROM SB2 as S, ItemsInOrders as I  
WHERE
```