

BSc (Hons) Artificial Intelligence and Data Science

Module: CM1603 Database Systems

Individual Coursework Report

Module Leader: Ms. Dileeka Alwis

RGU Student ID: 2330910

IIT Student ID: 20230374

Student Name: Setunge Mudalige Don Chirath Shamika Setunge

Acknowledgment

I would like to express my profound gratitude to Mrs. Dileeka Alwis for her tremendous collaboration and support in making my project achieve such a noteworthy outcome. Not to mention, I would want to thank my friends and supporters, without whom the work could not have been completed effectively and on time.

Table of content

ACKNOWLEDGMENT.....	2
TABLE OF CONTENT	3
TABLE OF FIGURES	4
SECTION 1.....	6
ENTITIES	6
PRIMARY KEYS	6
NON-KEY ATTRIBUTES.....	6
RELATIONSHIP BETWEEN ENTITIES	7
MULTIPLICITY CONSTRAINTS	7
ASSUMPTIONS.....	8
EERD (EXTENDED ENTITY RELATIONSHIP DIAGRAM)	9
SECTION 2.....	10
RELATIONAL SCHEMA DIAGRAM	10
RELATIONAL SCHEMA DIAGRAM ASSUMPTIONS.....	11
SECTION 3	12
CREATING THE DATABASE.....	12
TABLE CREATION	13
<i>Customer Table</i>	13
<i>CustomerContact Table</i>	14
<i>OnlineOrder Table</i>	15
<i>OnlinePayment Table</i>	16
<i>BankTransferPayment Table</i>	17
<i>Delivery Table</i>	18
<i>Supplier Table</i>	19
<i>SupplierContact Table</i>	20
<i>BookItem Table</i>	21
<i>StationeryItem Table</i>	22
<i>BookItemSupplier Table</i>	23
<i>StationeryItemSupplier Table</i>	24
<i>BookOrderItem Table</i>	25
<i>StationeryOrderItem Table</i>	26
INSERTING DATA TO EACH TABLE	27
<i>Customer Table</i>	28
<i>CustomerContact Table</i>	29
<i>BookItem Table</i>	30
<i>StationeryItem Table</i>	31
<i>Supplier Table</i>	32
<i>SupplierContact Table</i>	33
<i>BookItemSupplier Table</i>	34
<i>StationeryItemSupplier Table</i>	35
<i>Trigger 1</i>	36
<i>OnlineOrder Table</i>	37
<i>Trigger 2</i>	38
<i>BookOrderItem Table</i>	39
<i>StationeryOrderItem Table</i>	40
<i>OnlinePayment Table</i>	41
<i>BankTransferPayment Table</i>	42
<i>Delivery Table</i>	43
DATABASE DIAGRAM	44
SECTION 4.....	45
A).....	45
B)	46

Table of Figures

Figure 1 Extended Entity Relationship Diagram	9
Figure 2 Relational Schema Diagram	10
Figure 3 Query to create the Database.	12
Figure 4 Created Database.	12
Figure 5 Query to create Customer table.	13
Figure 6 Customer table created successfully.	13
Figure 7 Customer table column specifications	13
Figure 8 Query to create CustomerContact table.	14
Figure 9 CustomerContact table created successfully.	14
Figure 10 CustomerContact table Column Specifications.	14
Figure 11 Query to create OnlineOrder table.....	15
Figure 12 OnlineOrder table created successfully.	15
Figure 13 OnlineOrder table Column Specifications....	15
Figure 14 Query to create OnlinePayment table.	16
Figure 15 OnlinePayment table created successfully.	16
Figure 16 OnlinePayment table Column Specifications....	16
Figure 17 Query to create BankTransferPayment table.	17
Figure 18 BankTransferPayment table created successfully.	17
Figure 19 BankTransferPayment table Column Specifications....	17
Figure 20 Query to create Delivery table.	18
Figure 21 Delivery table created successfully.	18
Figure 22 Delivery table Column Specifications.	18
Figure 23 Query to create Supplier table.	19
Figure 24 Supplier table created successfully.	19
Figure 25 Supplier table Column Specifications.	19
Figure 26 Query to create SupplierContact table.....	20
Figure 27 SupplierContact table created successfully.	20
Figure 28 SupplierContact table Column Specifications....	20
Figure 29 Query to create BookItem table.	21
Figure 30 BookItem table created successfully.	21
Figure 31 BookItem table Column Specifications.	21
Figure 32 Query to create StationeryItem table.	22
Figure 33 StationeryItem table created successfully.....	22
Figure 34 StationeryItem table Column Specifications....	22
Figure 35 Query to create BookItemSupplier table.	23
Figure 36 BookItemSupplier table created successfully.	23
Figure 37 BookItemSupplier table Column Specifications.	23
Figure 38 Query to create StationeryItemSupplier table.....	24
Figure 39 StationeryItemSupplier table created successfully.	24
Figure 40 StationeryItemSupplier table Column Specifications....	24
Figure 41 Query to create BookOrderItem table.	25
Figure 42 BookOrderItem table created successfully.	25
Figure 43 BookOrderItem table Column Specifications....	25
Figure 44 Query to create StationeryOrderItem table.	26
Figure 45 StationeryOrderItem table created successfully.	26
Figure 46 StationeryOrderItem table Column Specifications....	26
Figure 47 Data entering query for Customer table.	28
Figure 48 Data entered successfully to Customer table.	28
Figure 49 Data entered Customer table.....	28
Figure 50 Data entering query for CustomerContact table.	29
Figure 51 Data entered successfully to CustomerContact table.	29
Figure 52 Data entered CustomerContact table.	29
Figure 53 Data entering query for BookItem table.	30
Figure 54 Data entered successfully to BookItem table.	30
Figure 55 Data entered BookItem tabl.	30

Figure 56 Data entering query for StationeryItem table	31
Figure 57 Data entered successfully to StationeryItem table.....	31
Figure 58 Data entered StationeryItem table.	31
Figure 59 Data entering query for Supplier table.	32
Figure 60 Data entered successfully to Supplier table.....	32
Figure 61 Data entered Supplier table.....	32
Figure 62 Data entering query for SupplierContact table.....	33
Figure 63 Data entered successfully to SupplierContact table.....	33
Figure 64 Data entered SupplierContact table.	33
Figure 65 Data entering query for BookItemSupplier table.	34
Figure 66 Data entered successfully to BookItemSupplier table.....	34
Figure 67 Data entered BookItemSupplier table.....	34
Figure 68 Data entering query for StationeryItemSupplier table.....	35
Figure 69 Data entered successfully to StationeryItemSupplier table.d	35
Figure 70 Data entered StationeryItemSupplier table.....	35
Figure 71 Trigger that update the totalPrice in OnlineOrder table(book).....	36
Figure 72 Trigger that update the totalPrice in OnlineOrder table(stationery).....	36
Figure 73 Data entering query for OnlineOrder table.....	37
Figure 74 Data entered successfully to OnlineOrder table.	37
Figure 75 Data entered OnlineOrder table.....	37
Figure 76 Trigger that calculate the price in StationeryOrderItem table.	38
Figure 77 Trigger that calculate the price in BookOrderItem table.	38
Figure 78 Data entering query for BookOrderItem table.....	39
Figure 79 Data entered successfully to BookOrderItem table.	39
Figure 80 Data entered BookOrderItem table.	39
Figure 81 Data entering query for StationeryOrderItem table.	40
Figure 82 Data entered successfully to StationeryOrderItem table.	40
Figure 83Data entered StationeryOrderItem table.	40
Figure 84 Data entering query for OnlinePayment table.	41
Figure 85 Data entered successfully to OnlinePayment table.....	41
Figure 86 Data entered OnlinePayment table.	41
Figure 87 Data entering query for BankTransferPayment table.	42
Figure 88 Data entered successfully to BankTransferPayment table.....	42
Figure 89 Data entered BankTransferPayment table.	42
Figure 90 Data entering query for Delivery table.	43
Figure 91 Data entered successfully to Delivery table.....	43
Figure 92 Data entered Delivery table.	43
Figure 93 Database diagram	44
Figure 94 DML query for question a).....	45
Figure 95 results for question a)	45
Figure 96 DML query for question b).....	46
Figure 97 results for question b)	46

Section 1

Entities

- Customer
- OnlineOrder
- Item
- Supplier
- Delivery
- BankTransferPayment
- OnlinePayment
- Book
- Stationery

Primary Keys

- Customer: customerId
- OnlineOrder: orderId
- Item: itemCode
- Supplier: supplierId
- Delivery: deliveryId
- BankTransferPayment: bankPaymentId
- OnlinePayment: onlinePaymentId

Non-Key Attributes

- Customer: name (f_name, l_name), eMail, address (stNo, street, city, postal_code), tellNo[1..3], password
- OnlineOrder: orderDate, orderStatus, totalPrice
- Item: price, stockLevel, reorderLevel
- Supplier: supplierType, name, eMail, address (stNo, street, city, postal_code), tellNo[1..3]
- Delivery: deliveryDate, deliveryAddress (stNo, street, city, postal_code), deliveryStatus
- BankTransferPayment: amount, paymentDate, bankName, accNo, accHolderName
- OnlinePayment: amount, paymentDate, transactionId, cardType, cardNumber, expiryDate, cvv
- Book: ISBN, title, genre, category, author, publisher, yearOfPublication
- Stationery: type, brand, color, size

Relationship Between Entities

- Customer
 - Customer *places* OnlineOrder
 - Customer *makes* OnlinePayment
 - Customer *makes* BankTransferPayment
 - Customer *receives* Delivery
- OnlineOrder
 - OnlineOrder *is_for* Item
 - OnlineOrder *Leads_to* Delivery
- Supplier
 - Supplier *supplies* Item
- OnlinePayment
 - OnlinePayment *confirms* OnlineOrder
- BankTransferPayment
 - BankTransferPayment *confirms* OnlineOrder

Multiplicity Constraints

- Each customer may place one or many online orders, while ensuring that each order is linked to exclusively to a single customer.
- Each customer may make one or many online payments, while ensuring that each online payment is linked to exclusively to a single customer.
- Each customer may make one or many bank transfer payments, while ensuring that each bank transfer payment is linked to exclusively to a single customer.
- An online payment can only confirm one online order, while each online order must be confirmed by one or many online payments.
- A bank transfer payment can only confirm one online order, while each online order must be confirmed by one or many bank transfer payments
- An online order is for one or many items, while each item may include in one or many online orders.
- Each supplier must supply at least a single item or supplier can supply many items; each item is supplied by one or many suppliers.
- At the end each online order may leads to a delivery, while each delivery is linked to exclusively to a single online order. This shop offer ‘store pickup’ facility that why online orders may leads to a delivery.

- Each customer may receive one or many deliveries, while ensuring that each delivery is linked to exclusively to a single customer.

Assumptions

- Customer may place one or many online orders and to confirms that one or many online orders customer may make one or many online or bank transfer payments.
- The Book Haven store supports for partial payments, allowing customers to pay for an online order in instalments. Due to expansive items but store doesn't release any items until store receives the full payment.
- If customer doesn't place an online order, then customer won't make payments and won't receive any deliveries.
- It is expected that a consumer will stick to a single payment method when confirming an online order.
- Items are specialized as books and stationeries, this online shop only sales books and stationeries.
- Only reading books are considered as books, while writing books are considered as stationeries.
- Supplier table entity supplier type attribute ensures whether the supplier is individual person or a company.
- In its operational boundaries, this online store only offers domestic shipping. The portal only serves local customers and engages with local suppliers.
- Customers send their online orders to different delivery address not only their private home address.
- orderStatus in OnlineOrder has these different statuses ('pending', 'processing', 'out for delivery', 'delivered', 'store pickup').
- deliveryStatus in Delivery has these different statuses ('pending', 'in transit', 'delivered').

EERD (Extended Entity Relationship Diagram)

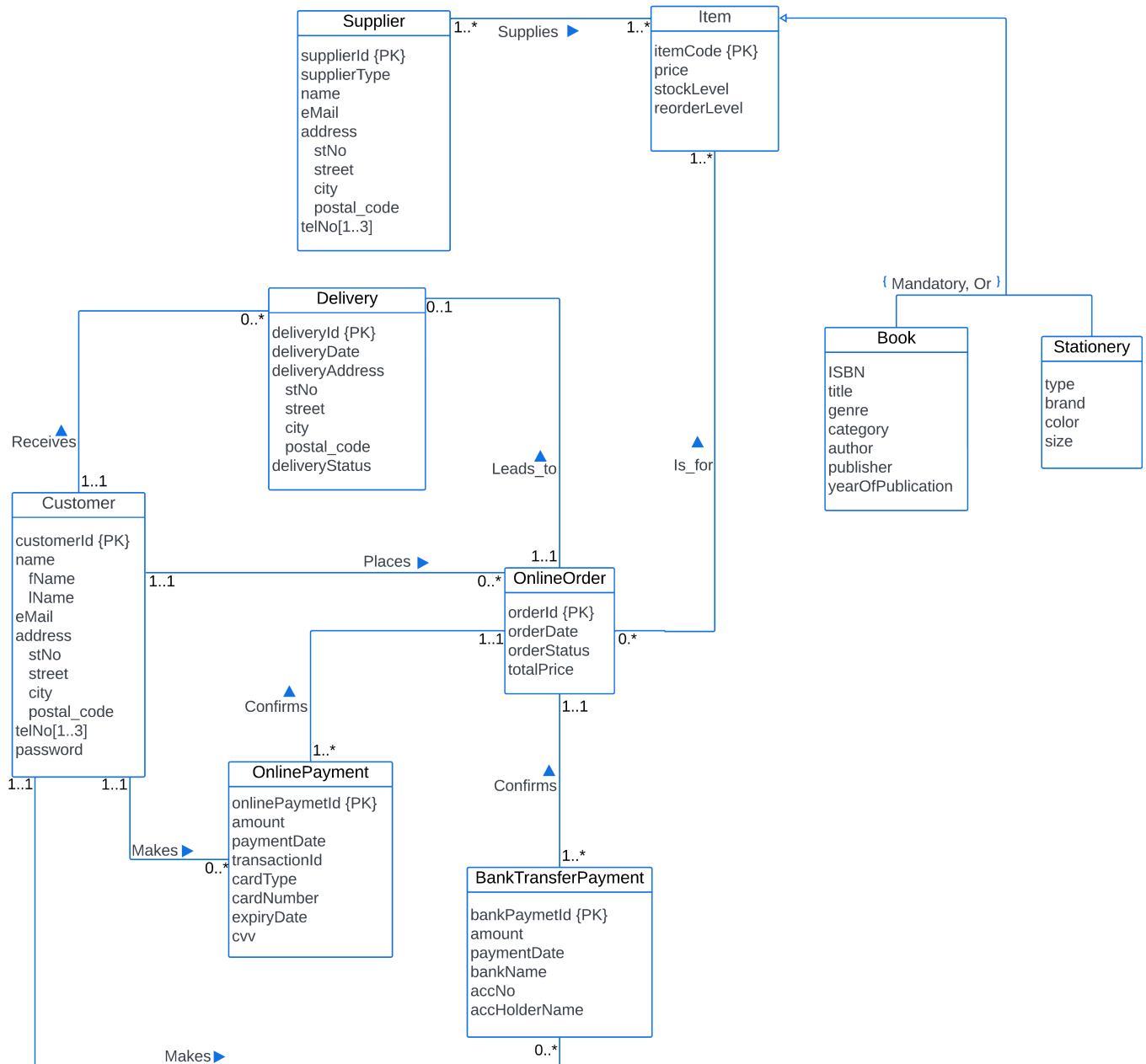


Figure 1 Extended Entity Relationship Diagram

Section 2

Relational Schema Diagram

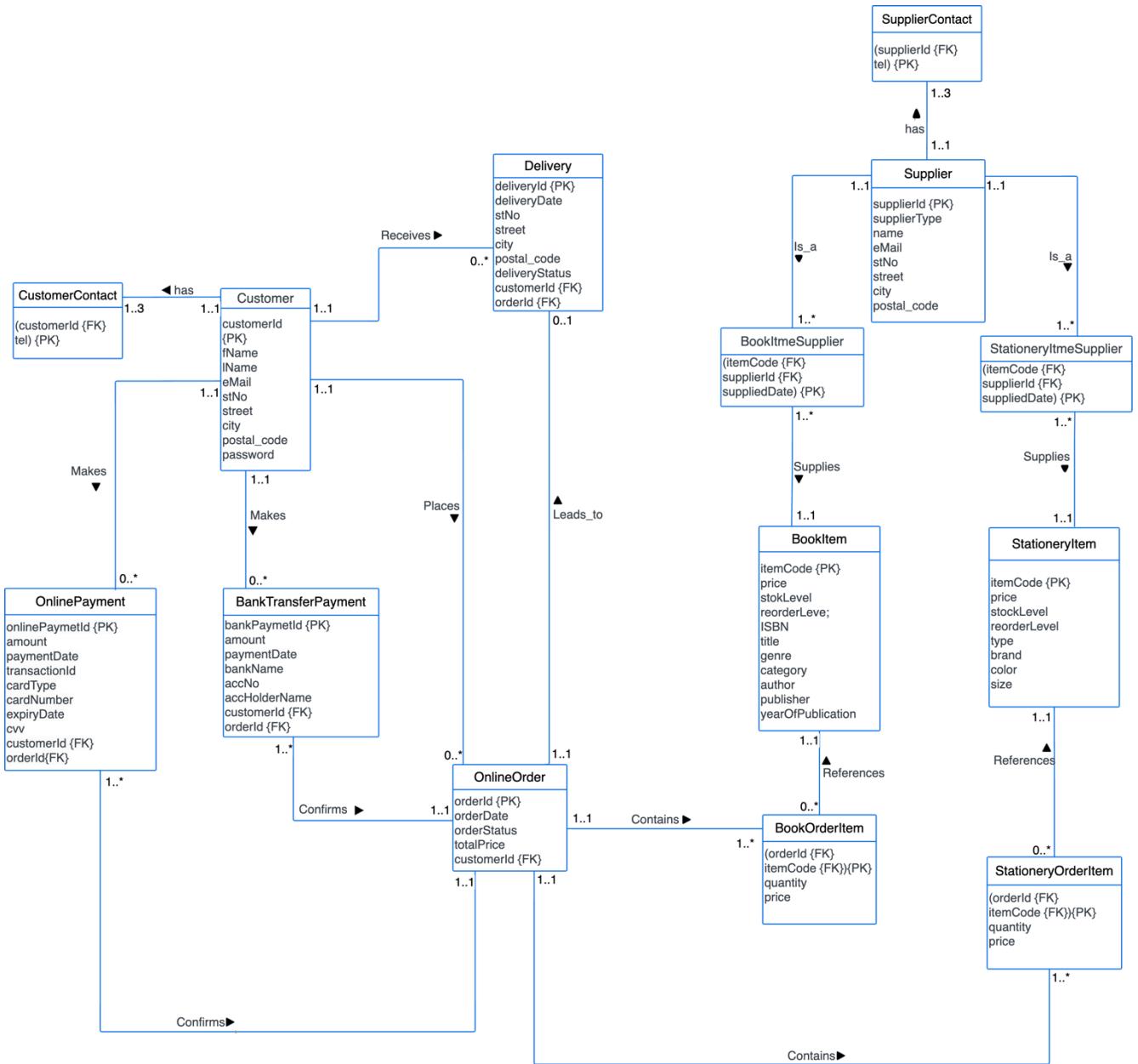


Figure 2 Relational Schema Diagram

Relational Schema Diagram Assumptions

- Created a new table called CustomerContact and assumed that the telNo in Customer table was a multi-valued attribute.
- Created a new table called SupplierContact and assumed that the telNo in Supplier was a multi-valued attribute.
- Created a new two table called BookItem and StationeryItem, assumed that the Item table is specialized as Book and Stationery.
- Created a new table called StationeryItemSupplier and assumed a many-to-many multiplicity constraint between Supplier and Item.
- Created a new table called BookItemSupplier and assumed a many-to-many multiplicity constraint between Supplier and Item.
- Created a new table called BookOrderItem and assumed a many-to-many multiplicity constraint between OnlineOrder and Item. Add quantity and price attributes to BookOrderItem.
- Created a new table called StationeryOrderItem and assumed a many-to-many multiplicity constraint between OnlineOrder and Item. Add quantity and price to StationeryOrderItem.
- customerId was added as a foreign key in OnlinePayment, assuming that the multiplicity constraint between Customer and OnlinePayment was one-to-many.
- customerId was added as a foreign key in BankTransferPayment, assuming that the multiplicity constraint between Customer and BankTransferPayment was one-to-many.
- customerId was added as a foreign key in OnlineOrder, assuming that the multiplicity constraint between Customer and OnlineOrder was one-to-many.
- customerId was added as a foreign key in Delivery, assuming that the multiplicity constraint between Customer and Delivery was one-to-many.
- orderId was added as a foreign key in OnlinePayment, assuming that the multiplicity constraint between OnlineOrder and OnlinePayment was one-to-many.
- orderId was added as a foreign key in BankTransferPayment, assuming that the multiplicity constraint between OnlineOrder and BankTransferPayment was one-to-many.
- orderId was added as a foreign key in Delivery, assuming that the multiplicity constraint between OnlineOrder and Delivery was one-to-many.

Section 3

Table Creation and Population of Data

Creating The Database

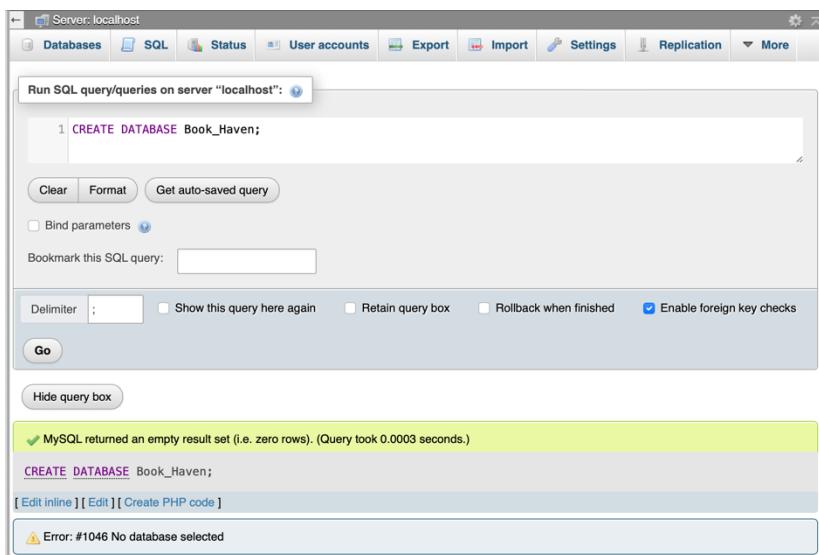


Figure 3 Query to create the Database.

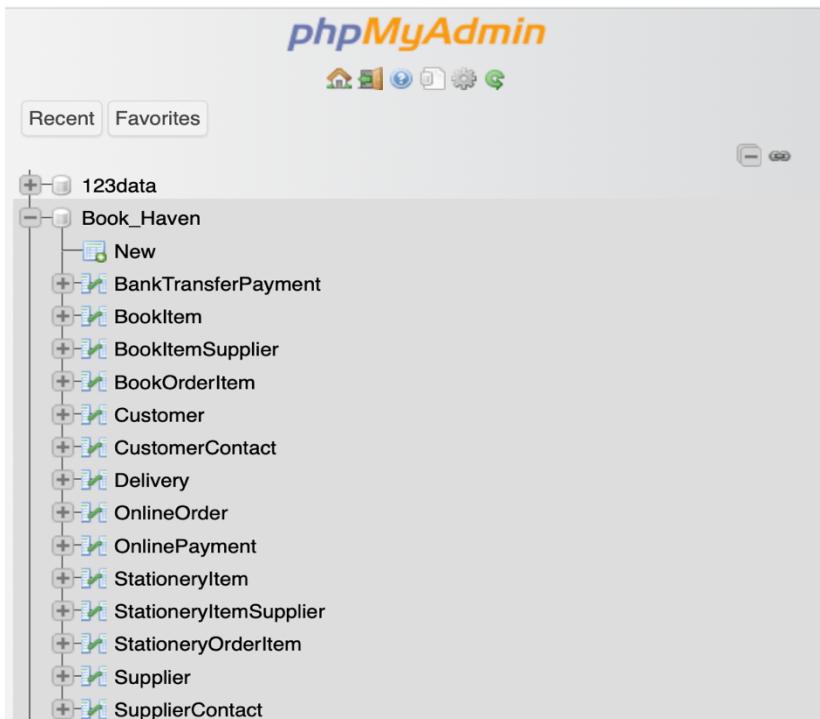
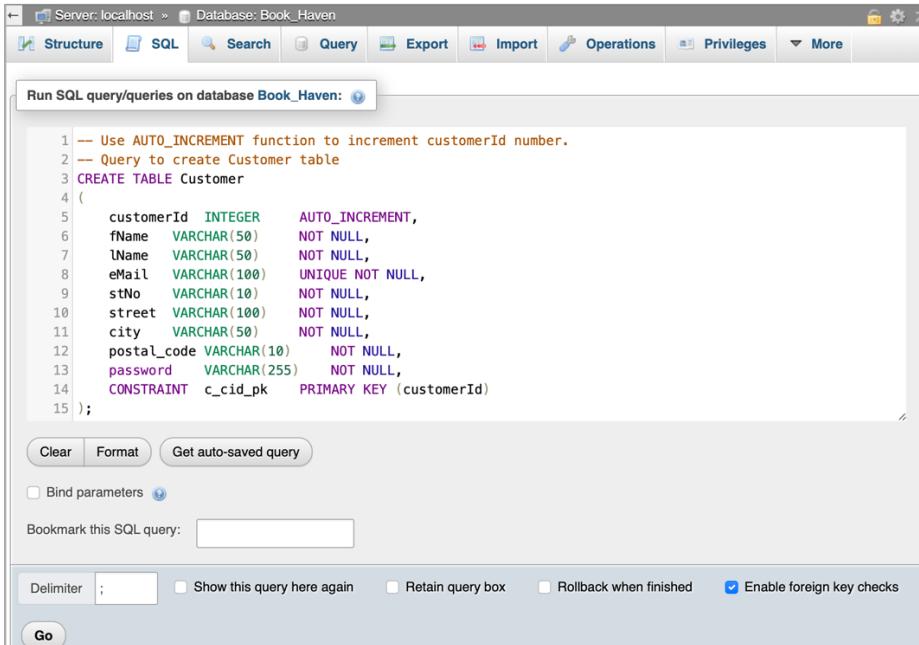


Figure 4 Created Database.

Table Creation

Customer Table

- Use AUTO_INCREMENT function to increment customerId number.



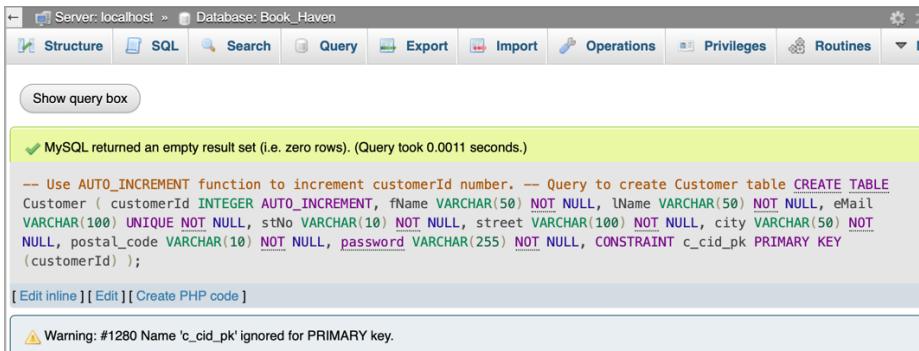
```

1 -- Use AUTO_INCREMENT function to increment customerId number.
2 -- Query to create Customer table
3 CREATE TABLE Customer
4 (
5     customerId INTEGER AUTO_INCREMENT,
6     fName VARCHAR(50) NOT NULL,
7     lName VARCHAR(50) NOT NULL,
8     eMail VARCHAR(100) UNIQUE NOT NULL,
9     stNo VARCHAR(10) NOT NULL,
10    street VARCHAR(100) NOT NULL,
11    city VARCHAR(50) NOT NULL,
12    postal_code VARCHAR(10) NOT NULL,
13    password VARCHAR(255) NOT NULL,
14    CONSTRAINT c_cid_pk PRIMARY KEY (customerId)
15 );

```

Clear Format Get auto-saved query
 Bind parameters
Bookmark this SQL query:
Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks

Figure 5 Query to create Customer table.



Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0011 seconds.)

```
-- Use AUTO_INCREMENT function to increment customerId number. -- Query to create Customer table CREATE TABLE Customer ( customerId INTEGER AUTO_INCREMENT, fName VARCHAR(50) NOT NULL, lName VARCHAR(50) NOT NULL, eMail VARCHAR(100) UNIQUE NOT NULL, stNo VARCHAR(10) NOT NULL, street VARCHAR(100) NOT NULL, city VARCHAR(50) NOT NULL, postal_code VARCHAR(10) NOT NULL, password VARCHAR(255) NOT NULL, CONSTRAINT c_cid_pk PRIMARY KEY (customerId) );
```

[Edit inline] [Edit] [Create PHP code]
Warning: #1280 Name 'c_cid_pk' ignored for PRIMARY key.

Figure 6 Customer table created successfully.

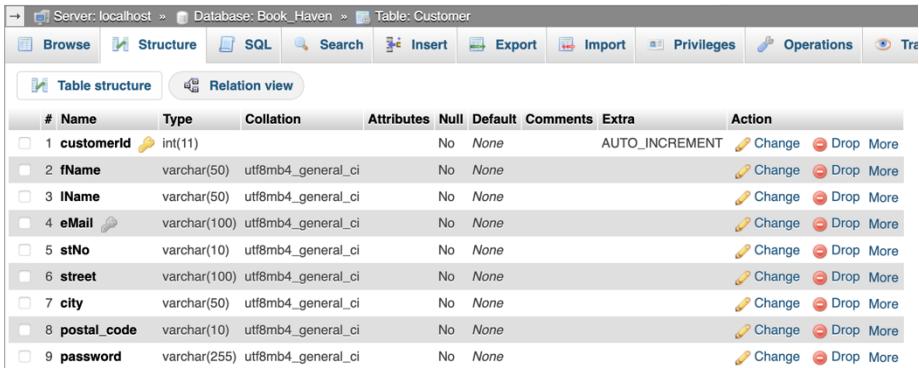
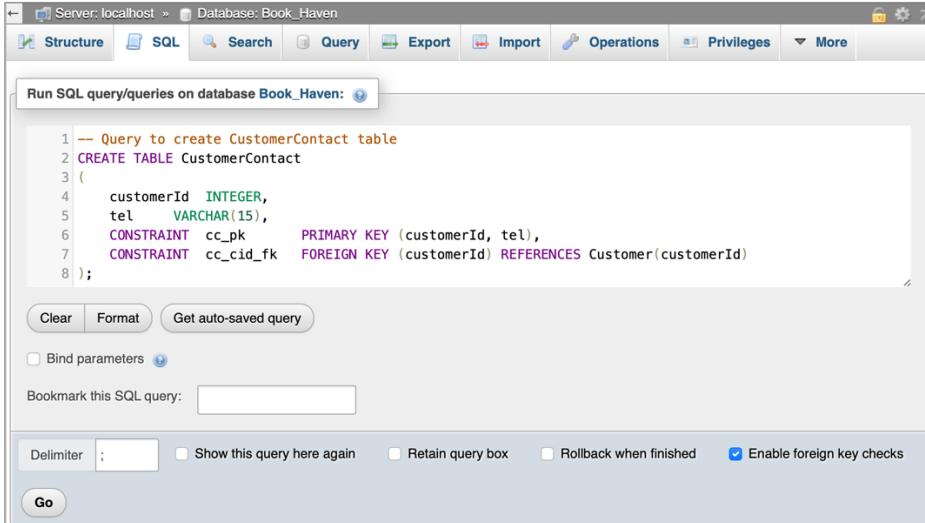


Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	customerId	int(11)			No	None	AUTO_INCREMENT		<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
2	fName	varchar(50)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
3	lName	varchar(50)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
4	eMail	varchar(100)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
5	stNo	varchar(10)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
6	street	varchar(100)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
7	city	varchar(50)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
8	postal_code	varchar(10)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
9	password	varchar(255)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>

Figure 7 Customer table column specifications

CustomerContact Table



```
1 -- Query to create CustomerContact table
2 CREATE TABLE CustomerContact
3 (
4     customerId INTEGER,
5     tel VARCHAR(15),
6     CONSTRAINT cc_pk PRIMARY KEY (customerId, tel),
7     CONSTRAINT cc_cid_fk FOREIGN KEY (customerId) REFERENCES Customer(customerId)
8 );
```

Clear Format Get auto-saved query
Bind parameters
Bookmark this SQL query:
Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks
Go

Figure 8 Query to create CustomerContact table.



Show query box
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
-- Query to create CustomerContact table CREATE TABLE CustomerContact (customerId INTEGER, tel VARCHAR(15), CONSTRAINT cc_pk PRIMARY KEY (customerId, tel), CONSTRAINT cc_cid_fk FOREIGN KEY (customerId) REFERENCES Customer(customerId));
[Edit inline] [Edit] [Create PHP code]
Warning: #1280 Name 'cc_pk' ignored for PRIMARY key.

Figure 9 CustomerContact table created successfully.

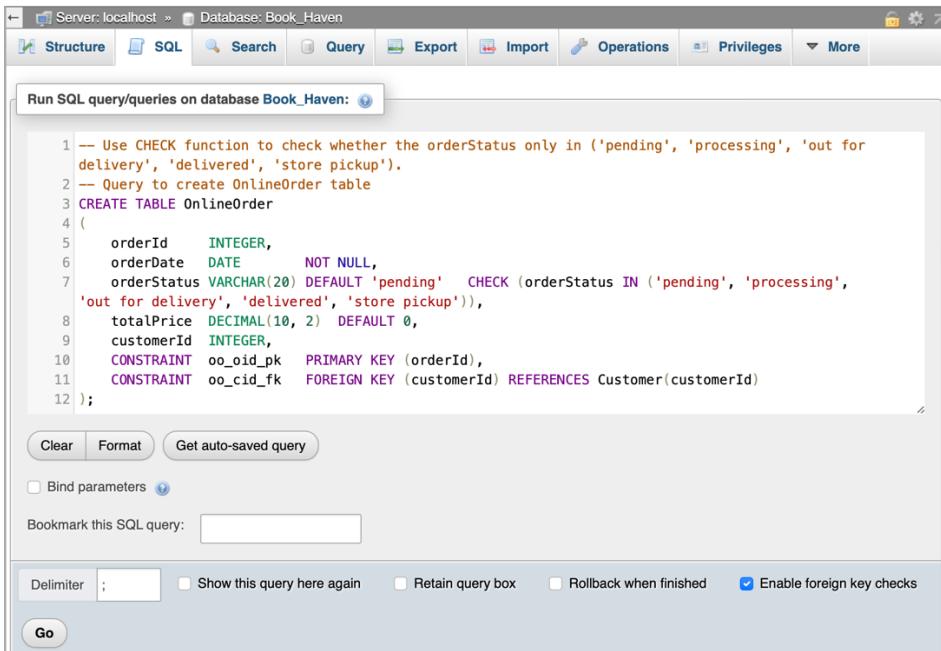


Server: localhost > Database: Book_Haven > Table: CustomerContact
Browse Structure SQL Search Insert Export Import Privileges Operations
Table structure Relation view
Name Type Collation Attributes Null Default Comments Extra Action
1 customerId int(11) No None Change Drop More
2 tel varchar(15) utf8mb4_general_ci No None Change Drop Primary Unique Index
Check all With selected: Browse Change Drop Primary Unique Index

Figure 10 CustomerContact table Column Specifications.

OnlineOrder Table

- Use CHECK function to check whether the orderStatus only in ('pending', 'processing', 'out for delivery', 'delivered', 'store pickup').



```

1 -- Use CHECK function to check whether the orderStatus only in ('pending', 'processing', 'out for
2 -- delivery', 'delivered', 'store pickup').
3 -- Query to create OnlineOrder table
4 CREATE TABLE OnlineOrder
5 (
6     orderId    INTEGER,
7     orderDate   DATE        NOT NULL,
8     orderStatus  VARCHAR(20) DEFAULT 'pending'  CHECK (orderStatus IN ('pending', 'processing',
9     'out for delivery', 'delivered', 'store pickup')),
10    totalPrice  DECIMAL(10, 2) DEFAULT 0,
11    customerId  INTEGER,
12    CONSTRAINT oo_oid_pk PRIMARY KEY (orderId),
13    CONSTRAINT oo_cid_fk FOREIGN KEY (customerId) REFERENCES Customer(customerId)
14 );

```

Clear Format Get auto-saved query

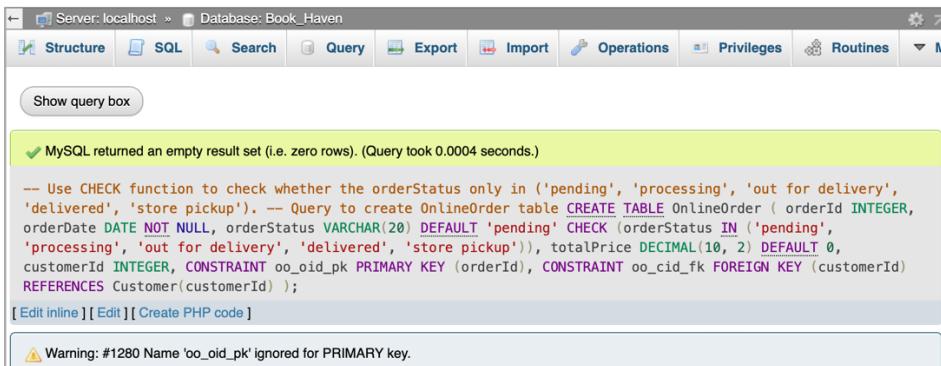
Bind parameters

Bookmark this SQL query:

Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks

Go

Figure 11 Query to create OnlineOrder table.



Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

```

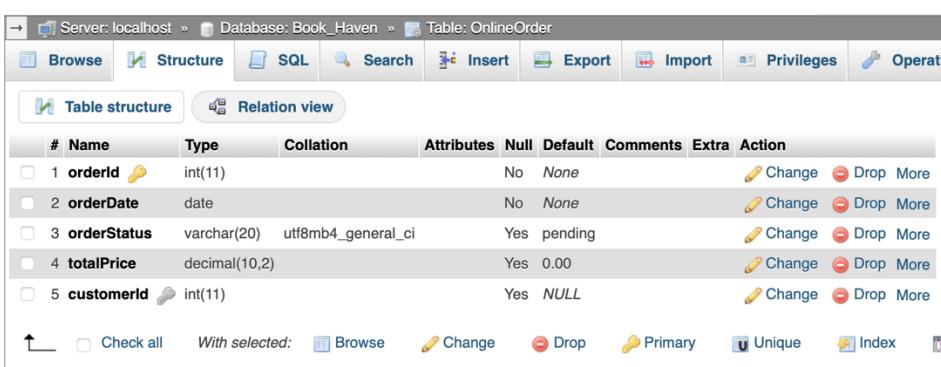
-- Use CHECK function to check whether the orderStatus only in ('pending', 'processing', 'out for delivery',
-- 'delivered', 'store pickup'). -- Query to create OnlineOrder table CREATE TABLE OnlineOrder ( orderId INTEGER,
orderDate DATE NOT NULL, orderStatus VARCHAR(20) DEFAULT 'pending' CHECK (orderStatus IN ('pending',
'processing', 'out for delivery', 'delivered', 'store pickup')), totalPrice DECIMAL(10, 2) DEFAULT 0,
customerId INTEGER, CONSTRAINT oo_oid_pk PRIMARY KEY (orderId), CONSTRAINT oo_cid_fk FOREIGN KEY (customerId)
REFERENCES Customer(customerId) );

```

[Edit inline] [Edit] [Create PHP code]

Warning: #1280 Name 'oo_oid_pk' ignored for PRIMARY key.

Figure 12 OnlineOrder table created successfully.



Browse Structure SQL Search Insert Export Import Privileges Operations

Table structure Relation view

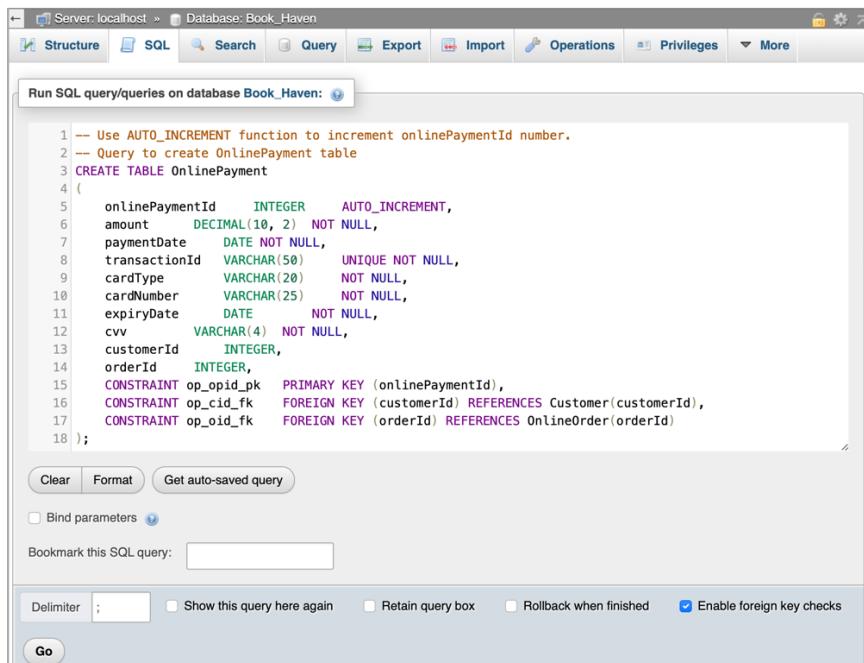
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	orderId	int(11)			No	None			<input type="button"/> Change <input type="button"/> Drop <input type="button"/> More
2	orderDate	date			No	None			<input type="button"/> Change <input type="button"/> Drop <input type="button"/> More
3	orderStatus	varchar(20)	utf8mb4_general_ci		Yes	pending			<input type="button"/> Change <input type="button"/> Drop <input type="button"/> More
4	totalPrice	decimal(10,2)			Yes	0.00			<input type="button"/> Change <input type="button"/> Drop <input type="button"/> More
5	customerId	int(11)			Yes	NULL			<input type="button"/> Change <input type="button"/> Drop <input type="button"/> More

Check all With selected: Browse Change Drop Primary Unique Index

Figure 13 OnlineOrder table Column Specifications.

OnlinePayment Table

- Use AUTO_INCREMENT function to increment onlinePaymentId number.



The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A SQL query is entered in the main pane:

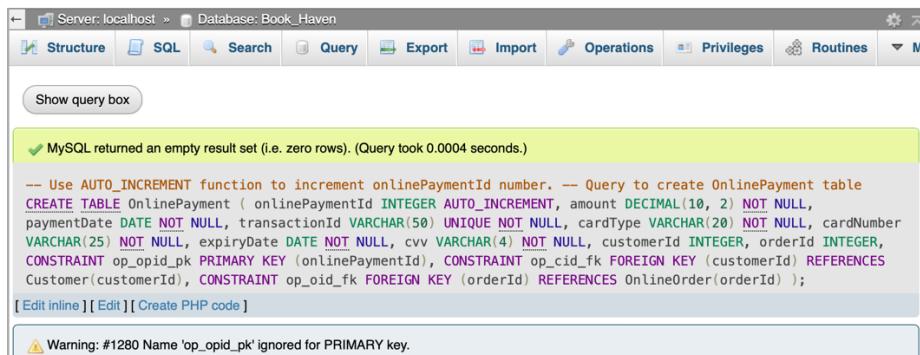
```

1 -- Use AUTO_INCREMENT function to increment onlinePaymentId number.
2 -- Query to create OnlinePayment table
3 CREATE TABLE OnlinePayment
4 (
5     onlinePaymentId      INTEGER      AUTO_INCREMENT,
6     amount        DECIMAL(10, 2)  NOT NULL,
7     paymentDate    DATE NOT NULL,
8     transactionId  VARCHAR(50)   UNIQUE NOT NULL,
9     cardType       VARCHAR(20)   NOT NULL,
10    cardNumber     VARCHAR(25)   NOT NULL,
11    expiryDate     DATE NOT NULL,
12    cvv           VARCHAR(4)    NOT NULL,
13    customerId    INTEGER,
14    orderId       INTEGER,
15    CONSTRAINT op_opid_pk PRIMARY KEY (onlinePaymentId),
16    CONSTRAINT op_cid_fk FOREIGN KEY (customerId) REFERENCES Customer(customerId),
17    CONSTRAINT op_oid_fk FOREIGN KEY (orderId) REFERENCES OnlineOrder(orderId)
18 );

```

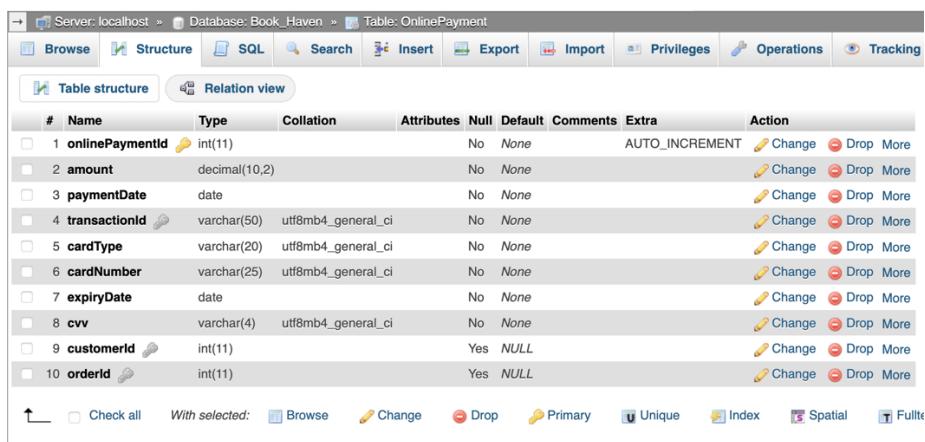
Below the query, there are several buttons: 'Clear', 'Format', 'Get auto-saved query', 'Bind parameters', 'Bookmark this SQL query:', 'Delimiter :', 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks'. A 'Go' button is at the bottom.

Figure 14 Query to create OnlinePayment table.



The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. The results pane displays a green message: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)'. Below the message is the same SQL query as in Figure 14. At the bottom of the results pane, there is a warning message: 'Warning: #1280 Name 'op_opid_pk' ignored for PRIMARY key.'.

Figure 15 OnlinePayment table created successfully.



The screenshot shows the MySQL Workbench interface with the 'Structure' tab selected for the OnlinePayment table. The table structure is displayed in a grid:

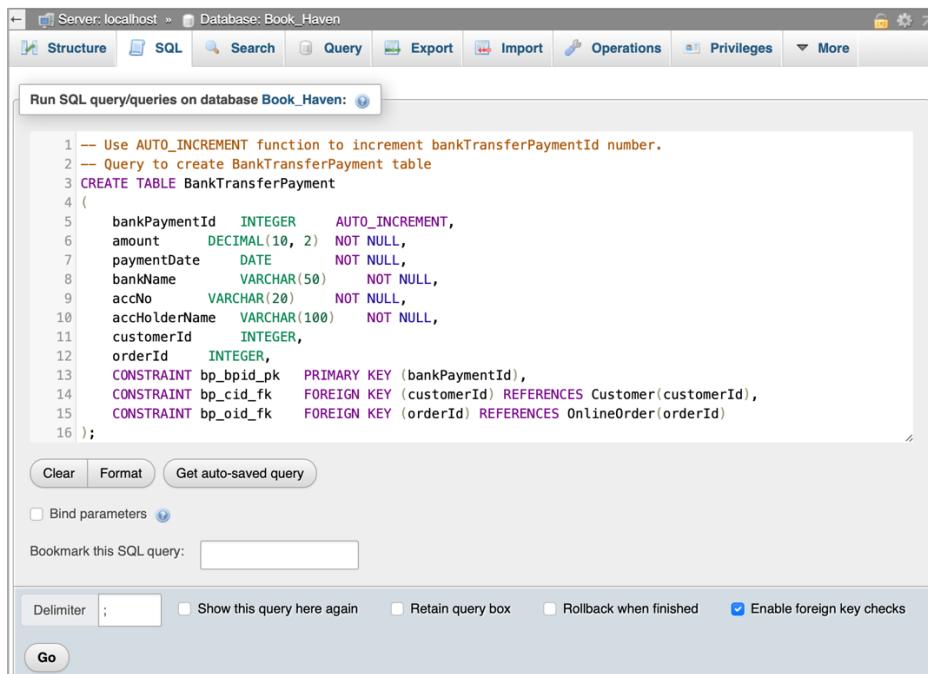
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	onlinePaymentId	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	amount	decimal(10,2)			No	None			Change Drop More
3	paymentDate	date			No	None			Change Drop More
4	transactionId	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
5	cardType	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
6	cardNumber	varchar(25)	utf8mb4_general_ci		No	None			Change Drop More
7	expiryDate	date			No	None			Change Drop More
8	cvv	varchar(4)	utf8mb4_general_ci		No	None			Change Drop More
9	customerId	int(11)			Yes	NULL			Change Drop More
10	orderId	int(11)			Yes	NULL			Change Drop More

At the bottom of the interface, there are buttons for 'Check all', 'With selected:', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'.

Figure 16 OnlinePayment table Column Specifications.

BankTransferPayment Table

- Use AUTO_INCREMENT function to increment bankTransferPaymentId number.



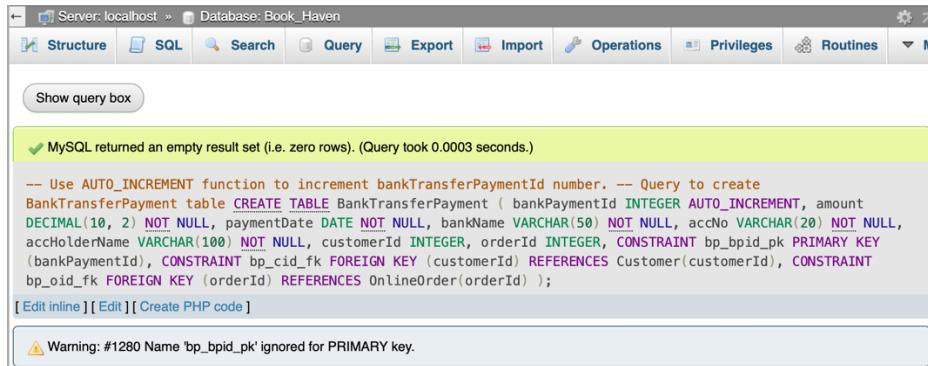
```

1 -- Use AUTO_INCREMENT function to increment bankTransferPaymentId number.
2 -- Query to create BankTransferPayment table
3 CREATE TABLE BankTransferPayment
4 (
5     bankPaymentId    INTEGER      AUTO_INCREMENT,
6     amount           DECIMAL(10, 2) NOT NULL,
7     paymentDate     DATE         NOT NULL,
8     bankName         VARCHAR(50)  NOT NULL,
9     accNo            VARCHAR(20)  NOT NULL,
10    accHolderName   VARCHAR(100) NOT NULL,
11    customerId      INTEGER,
12    orderId         INTEGER,
13    CONSTRAINT bp_bp_id_pk PRIMARY KEY (bankPaymentId),
14    CONSTRAINT bp_cid_fk FOREIGN KEY (customerId) REFERENCES Customer(customerId),
15    CONSTRAINT bp_oid_fk FOREIGN KEY (orderId) REFERENCES OnlineOrder(orderId)
16 );

```

Clear Format Get auto-saved query
 Bind parameters
Bookmark this SQL query:
Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks

Figure 17 Query to create BankTransferPayment table.



Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```

-- Use AUTO_INCREMENT function to increment bankTransferPaymentId number. -- Query to create
BankTransferPayment table CREATE TABLE BankTransferPayment (bankPaymentId INTEGER AUTO_INCREMENT, amount
DECIMAL(10, 2) NOT NULL, paymentDate DATE NOT NULL, bankName VARCHAR(50) NOT NULL, accNo VARCHAR(20) NOT NULL,
accHolderName VARCHAR(100) NOT NULL, customerId INTEGER, orderId INTEGER, CONSTRAINT bp_bp_id_pk PRIMARY KEY
(bankPaymentId), CONSTRAINT bp_cid_fk FOREIGN KEY (customerId) REFERENCES Customer(customerId), CONSTRAINT
bp_oid_fk FOREIGN KEY (orderId) REFERENCES OnlineOrder(orderId) );

```

[Edit inline] [Edit] [Create PHP code]
Warning: #1280 Name 'bp_bp_id_pk' ignored for PRIMARY key.

Figure 18 BankTransferPayment table created successfully.

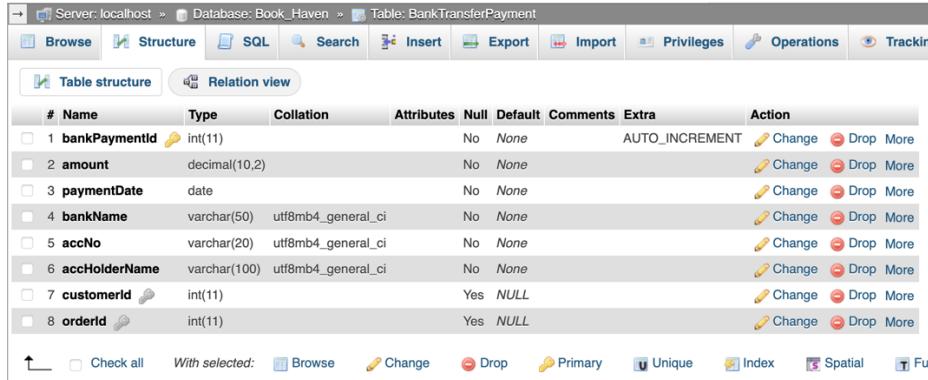


Table structure Relation view

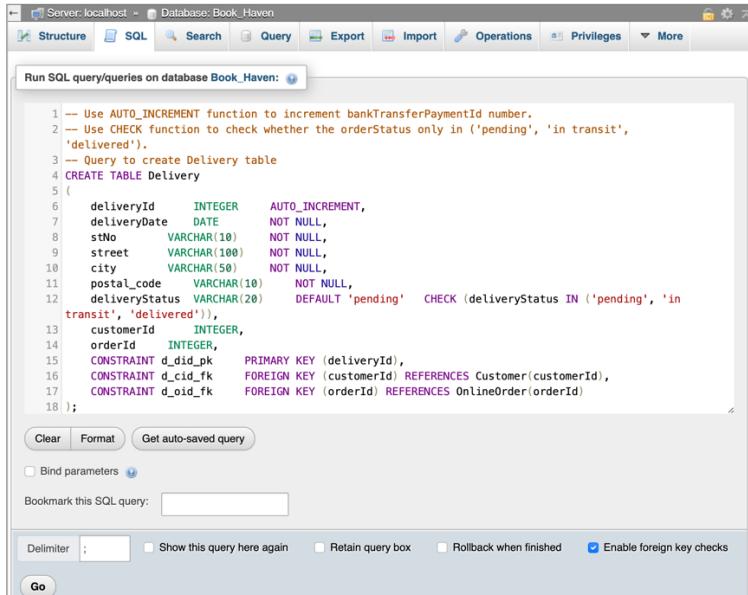
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	bankPaymentId	int(11)			No	None		AUTO_INCREMENT	<input type="button" value="Change"/> <input type="button" value="Drop"/> More
2	amount	decimal(10,2)			No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> More
3	paymentDate	date			No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> More
4	bankName	varchar(50)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> More
5	accNo	varchar(20)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> More
6	accHolderName	varchar(100)	utf8mb4_general_ci		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> More
7	customerId	int(11)			Yes	NULL			<input type="button" value="Change"/> <input type="button" value="Drop"/> More
8	orderId	int(11)			Yes	NULL			<input type="button" value="Change"/> <input type="button" value="Drop"/> More

With selected: Primary

Figure 19 BankTransferPayment table Column Specifications.

Delivery Table

- Use AUTO_INCREMENT function to increment bankTransferPaymentId number.
- Use CHECK function to check whether the orderStatus only in ('pending', 'in transit', 'delivered').



The screenshot shows the MySQL Workbench interface with the SQL tab selected. The query editor contains the following SQL code:

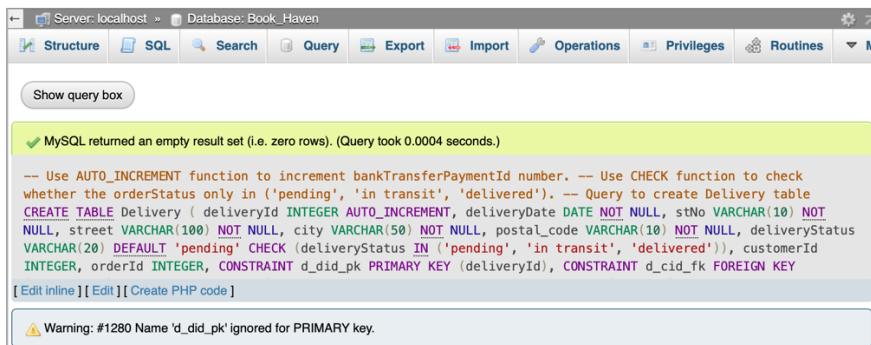
```

1 -- Use AUTO_INCREMENT function to increment bankTransferPaymentId number.
2 -- Use CHECK function to check whether the orderStatus only in ('pending', 'in transit',
3 -- 'delivered').
4 CREATE TABLE Delivery
5 (
6     deliveryId      INTEGER      AUTO_INCREMENT,
7     deliveryDate    DATE        NOT NULL,
8     stNo           VARCHAR(10)  NOT NULL,
9     street          VARCHAR(100) NOT NULL,
10    city            VARCHAR(50)  NOT NULL,
11    postal_code    VARCHAR(10)  NOT NULL,
12    deliveryStatus  VARCHAR(20)  DEFAULT 'pending'  CHECK (deliveryStatus IN ('pending', 'in
13    transit', 'delivered')),
14    customerId      INTEGER,
15    CONSTRAINT d_did_pk PRIMARY KEY (deliveryId),
16    CONSTRAINT d_cid_fk FOREIGN KEY (customerId) REFERENCES Customer(customerId),
17    CONSTRAINT d_oid_fk FOREIGN KEY (orderId) REFERENCES OnlineOrder(orderId)
18 );

```

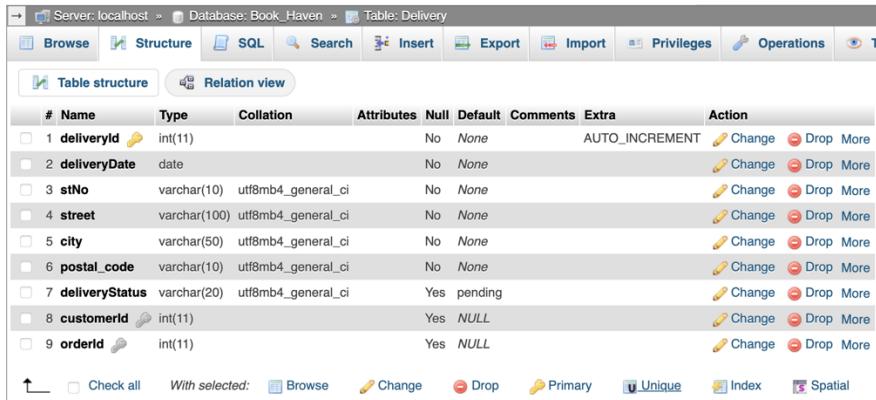
Below the query editor are several buttons: Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query:, Delimiter :, Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A 'Go' button is at the bottom left.

Figure 20 Query to create Delivery table.



The screenshot shows the MySQL Workbench interface with the SQL tab selected. The results pane displays a green message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)". Below the message is the same SQL code as in Figure 20. At the bottom of the results pane, there is a warning message: "Warning: #1280 Name 'd_did_pk' ignored for PRIMARY key." Buttons for Edit inline, Edit, and Create PHP code are also visible.

Figure 21 Delivery table created successfully.



The screenshot shows the MySQL Workbench interface with the Structure tab selected for the Delivery table. The table structure is displayed in a grid:

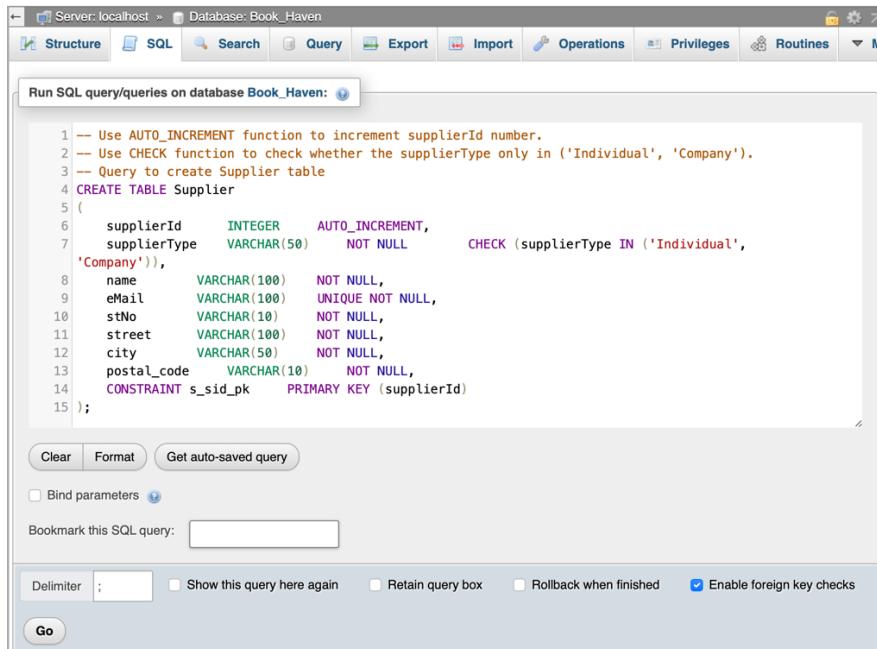
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	deliveryId	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	deliveryDate	date			No	None			Change Drop More
3	stNo	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
4	street	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
5	city	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
6	postal_code	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
7	deliveryStatus	varchar(20)	utf8mb4_general_ci		Yes	pending			Change Drop More
8	customerId	int(11)			Yes	NULL			Change Drop More
9	orderId	int(11)			Yes	NULL			Change Drop More

At the bottom of the interface, there are buttons for Check all, With selected: Browse, Change, Primary, Unique, Index, and Spatial.

Figure 22 Delivery table Column Specifications.

Supplier Table

- Use AUTO_INCREMENT function to increment supplierId number.
- Use CHECK function to check whether the supplierType only in ('Individual', 'Company').



```

1 -- Use AUTO_INCREMENT function to increment supplierId number.
2 -- Use CHECK function to check whether the supplierType only in ('Individual', 'Company').
3 -- Query to create Supplier table
4 CREATE TABLE Supplier
5 (
6     supplierId      INTEGER      AUTO_INCREMENT,
7     supplierType    VARCHAR(50)   NOT NULL           CHECK (supplierType IN ('Individual',
8         'Company')),
9     name            VARCHAR(100)  NOT NULL,
10    eMail           VARCHAR(100)  UNIQUE NOT NULL,
11    stNo            VARCHAR(10)   NOT NULL,
12    street          VARCHAR(100)  NOT NULL,
13    city            VARCHAR(50)   NOT NULL,
14    postal_code    VARCHAR(10)   NOT NULL,
15    CONSTRAINT s_sid_pk PRIMARY KEY (supplierId)
15 );

```

Clear Format Get auto-saved query

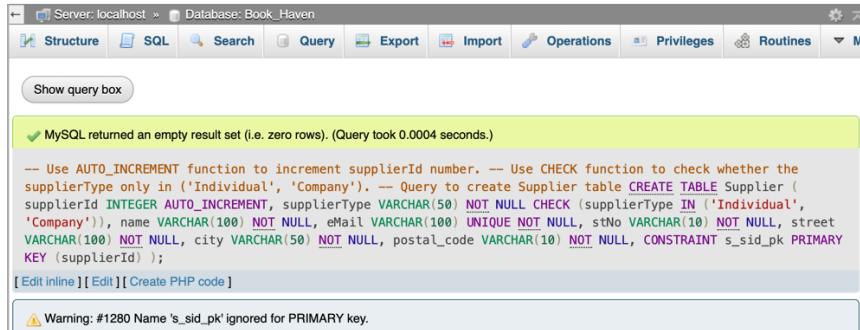
Bind parameters

Bookmark this SQL query:

Delimiter ; Show this query here again Retain query box Rollback when finished Enable foreign key checks

Go

Figure 23 Query to create Supplier table.



Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

```

-- Use AUTO_INCREMENT function to increment supplierId number. -- Use CHECK function to check whether the
supplierType only in ('Individual', 'Company'). -- Query to create Supplier table CREATE TABLE Supplier (
supplierId INTEGER AUTO_INCREMENT, supplierType VARCHAR(50) NOT NULL CHECK (supplierType IN ('Individual',
'Company')), name VARCHAR(100) NOT NULL, eMail VARCHAR(100) UNIQUE NOT NULL, stNo VARCHAR(10) NOT NULL, street
VARCHAR(100) NOT NULL, city VARCHAR(50) NOT NULL, postal_code VARCHAR(10) NOT NULL, CONSTRAINT s_sid_pk PRIMARY
KEY (supplierId) );

```

[Edit inline] [Edit] [Create PHP code]

Warning: #1280 Name 's_sid_pk' ignored for PRIMARY key.

Figure 24 Supplier table created successfully.

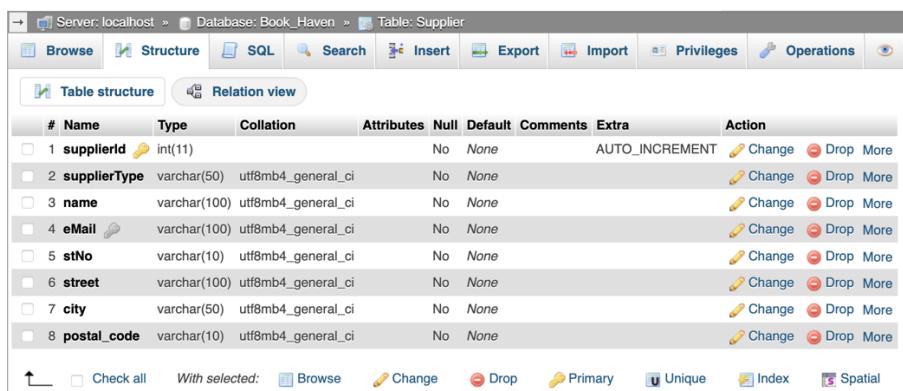


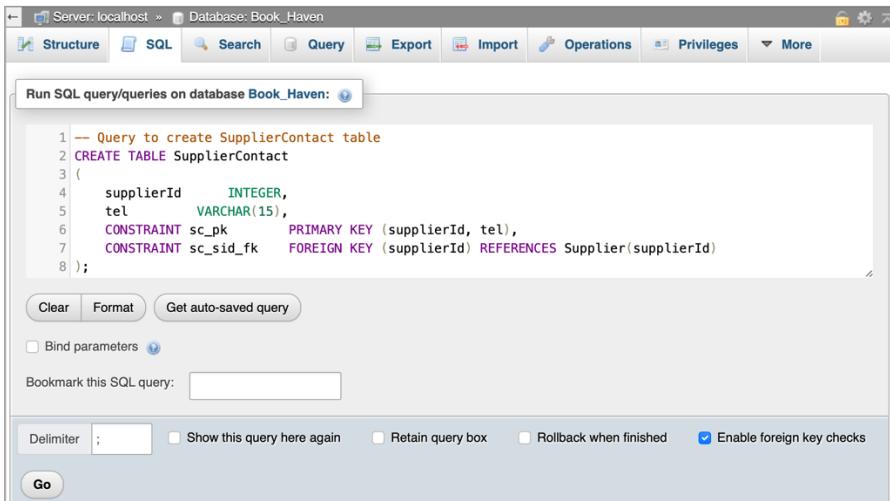
Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	supplierId	int(11)			No	None		AUTO_INCREMENT	More
2	supplierType	varchar(50)	utf8mb4_general_ci		No	None			More
3	name	varchar(100)	utf8mb4_general_ci		No	None			More
4	eMail	varchar(100)	utf8mb4_general_ci		No	None			More
5	stNo	varchar(10)	utf8mb4_general_ci		No	None			More
6	street	varchar(100)	utf8mb4_general_ci		No	None			More
7	city	varchar(50)	utf8mb4_general_ci		No	None			More
8	postal_code	varchar(10)	utf8mb4_general_ci		No	None			More

Check all With selected: Primary Unique Index Spatial

Figure 25 Supplier table Column Specifications.

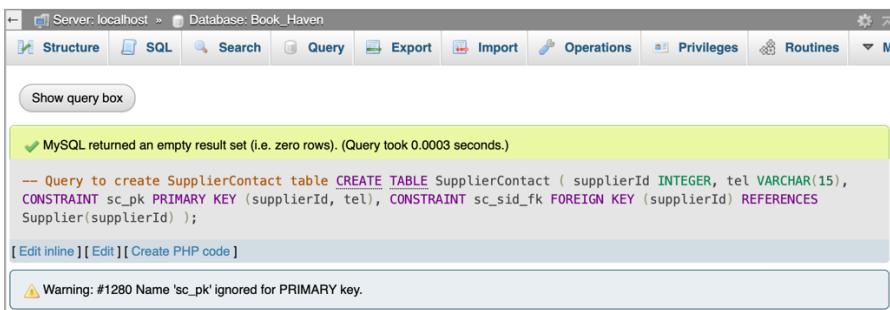
SupplierContact Table



```
1 -- Query to create SupplierContact table
2 CREATE TABLE SupplierContact
3 (
4     supplierId      INTEGER,
5     tel             VARCHAR(15),
6     CONSTRAINT sc_pk    PRIMARY KEY (supplierId, tel),
7     CONSTRAINT sc_sid_fk FOREIGN KEY (supplierId) REFERENCES Supplier(supplierId)
8 );
```

Clear Format Get auto-saved query
Bind parameters
Bookmark this SQL query:
Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks
Go

Figure 26 Query to create SupplierContact table.



Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
-- Query to create SupplierContact table CREATE TABLE SupplierContact ( supplierId INTEGER, tel VARCHAR(15), CONSTRAINT sc_pk PRIMARY KEY (supplierId, tel), CONSTRAINT sc_sid_fk FOREIGN KEY (supplierId) REFERENCES Supplier(supplierId) );
```

[Edit inline] [Edit] [Create PHP code]

Warning: #1280 Name 'sc_pk' ignored for PRIMARY key.

Figure 27 SupplierContact table created successfully.



Table structure Relation view

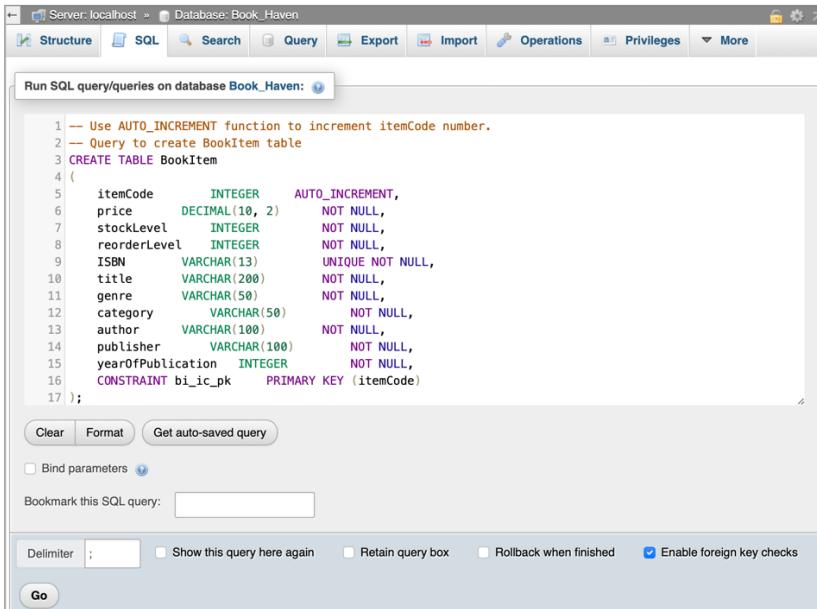
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
1	supplierId	int(11)			No	None					
2	tel	varchar(15)	utf8mb4_general_ci		No	None					

Check all With selected: Browse Change Primary Unique Index

Figure 28 SupplierContact table Column Specifications.

BookItem Table

- Use AUTO_INCREMENT function to increment itemCode number.



```

1 -- Use AUTO_INCREMENT function to increment itemCode number.
2 -- Query to create BookItem table
3 CREATE TABLE BookItem
4 (
5     itemCode      INTEGER      AUTO_INCREMENT,
6     price         DECIMAL(10, 2)    NOT NULL,
7     stockLevel   INTEGER      NOT NULL,
8     reorderLevel INTEGER      NOT NULL,
9     ISBN          VARCHAR(13)    UNIQUE NOT NULL,
10    title         VARCHAR(200)   NOT NULL,
11    genre         VARCHAR(50)    NOT NULL,
12    category      VARCHAR(50)    NOT NULL,
13    author        VARCHAR(100)   NOT NULL,
14    publisher     VARCHAR(100)   NOT NULL,
15    yearOfPublication INTEGER      NOT NULL,
16    CONSTRAINT bi_ic_pk PRIMARY KEY (itemCode)
17 );

```

Clear Format Get auto-saved query

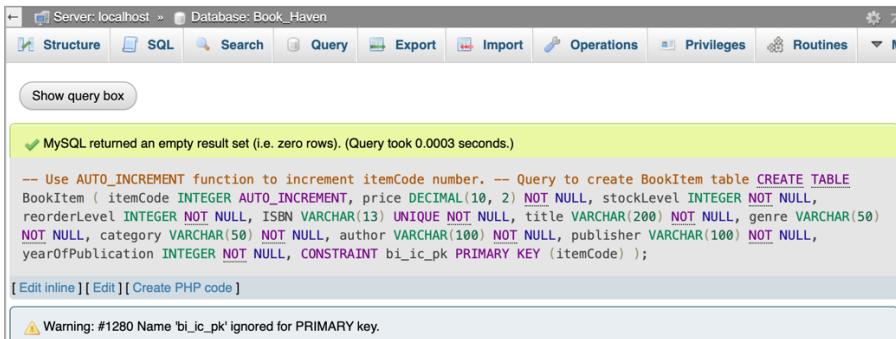
Bind parameters

Bookmark this SQL query:

Delimiter ; Show this query here again Retain query box Rollback when finished Enable foreign key checks

Go

Figure 29 Query to create BookItem table.



Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
-- Use AUTO_INCREMENT function to increment itemCode number. -- Query to create BookItem table
CREATE TABLE BookItem (itemCode INTEGER AUTO_INCREMENT, price DECIMAL(10, 2) NOT NULL, stockLevel INTEGER NOT NULL, reorderLevel INTEGER NOT NULL, ISBN VARCHAR(13) UNIQUE NOT NULL, title VARCHAR(200) NOT NULL, genre VARCHAR(50) NOT NULL, category VARCHAR(50) NOT NULL, author VARCHAR(100) NOT NULL, publisher VARCHAR(100) NOT NULL, yearOfPublication INTEGER NOT NULL, CONSTRAINT bi_ic_pk PRIMARY KEY (itemCode) );
```

[Edit inline] [Edit] [Create PHP code]

Warning: #1280 Name 'bi_ic_pk' ignored for PRIMARY key.

Figure 30 BookItem table created successfully.

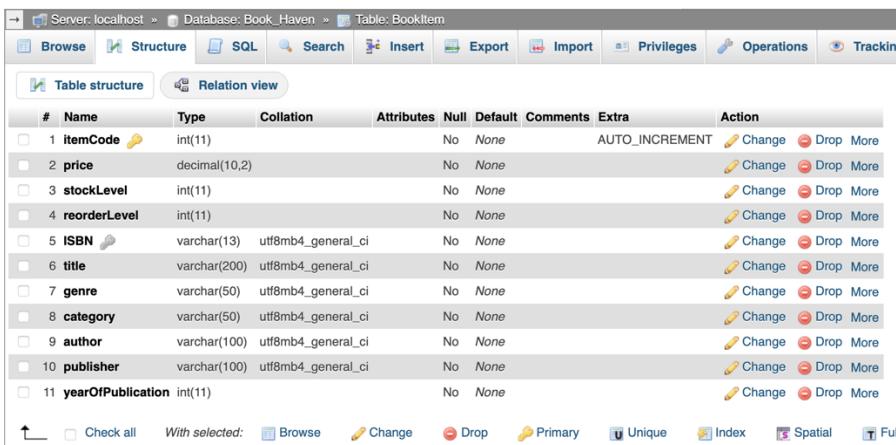


Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	itemCode	int(11)			No	None	AUTO_INCREMENT		More
2	price	decimal(10,2)			No	None			More
3	stockLevel	int(11)			No	None			More
4	reorderLevel	int(11)			No	None			More
5	ISBN	varchar(13)	utf8mb4_general_ci		No	None			More
6	title	varchar(200)	utf8mb4_general_ci		No	None			More
7	genre	varchar(50)	utf8mb4_general_ci		No	None			More
8	category	varchar(50)	utf8mb4_general_ci		No	None			More
9	author	varchar(100)	utf8mb4_general_ci		No	None			More
10	publisher	varchar(100)	utf8mb4_general_ci		No	None			More
11	yearOfPublication	int(11)			No	None			More

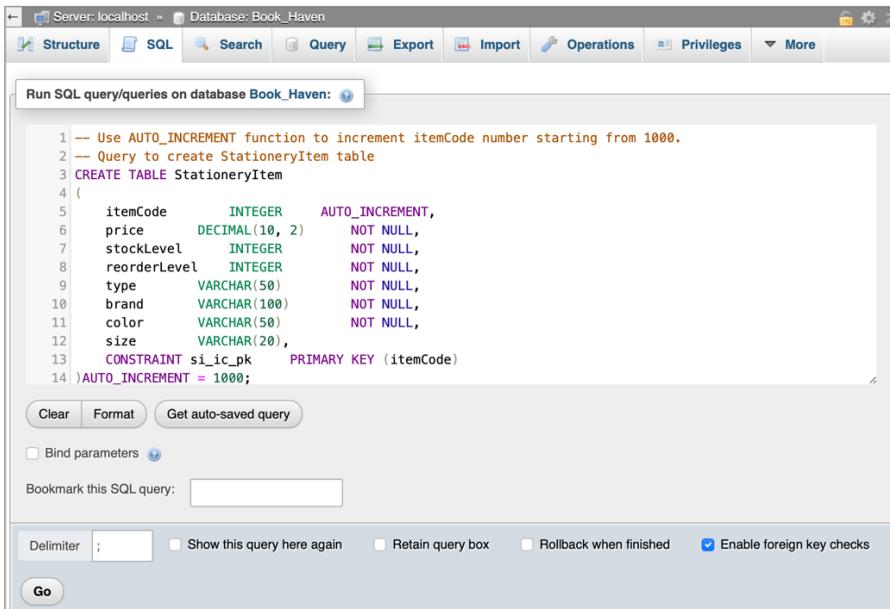
Check all With selected: Primary

Figure 31 BookItem table Column Specifications.

- This database can only hold up to 999 books because stationery itemCode starts from 1000.

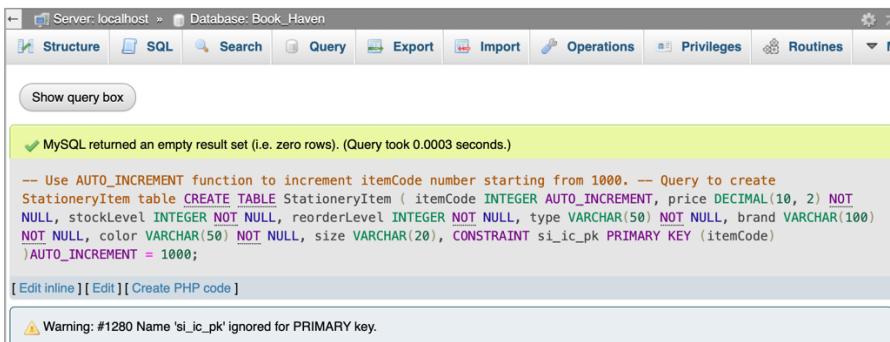
StationeryItem Table

- Use AUTO_INCREMENT function to increment itemCode number starting from 1000.



```
-- Use AUTO_INCREMENT function to increment itemCode number starting from 1000.
-- Query to create StationeryItem table
CREATE TABLE StationeryItem
(
    itemCode      INTEGER      AUTO_INCREMENT,
    price        DECIMAL(10, 2)  NOT NULL,
    stockLevel   INTEGER      NOT NULL,
    reorderLevel INTEGER      NOT NULL,
    type         VARCHAR(50)   NOT NULL,
    brand        VARCHAR(100)  NOT NULL,
    color        VARCHAR(50)   NOT NULL,
    size         VARCHAR(20),
    CONSTRAINT si_ic_pk PRIMARY KEY (itemCode)
)AUTO_INCREMENT = 1000;
```

Figure 32 Query to create StationeryItem table.



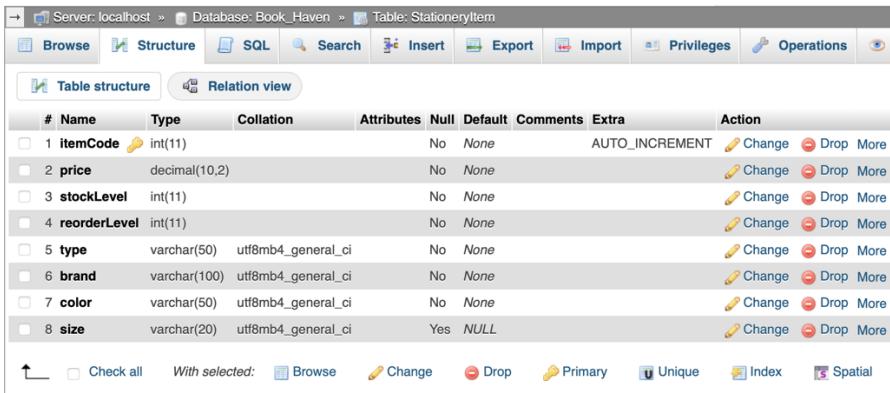
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
-- Use AUTO_INCREMENT function to increment itemCode number starting from 1000. -- Query to create
StationeryItem table CREATE TABLE StationeryItem ( itemCode INTEGER AUTO_INCREMENT, price DECIMAL(10, 2) NOT
NULL, stockLevel INTEGER NOT NULL, reorderLevel INTEGER NOT NULL, type VARCHAR(50) NOT NULL, brand VARCHAR(100)
NOT NULL, color VARCHAR(50) NOT NULL, size VARCHAR(20), CONSTRAINT si_ic_pk PRIMARY KEY (itemCode)
)AUTO_INCREMENT = 1000;
```

[Edit inline] [Edit] [Create PHP code]

⚠ Warning: #1280 Name 'si_ic_pk' ignored for PRIMARY key.

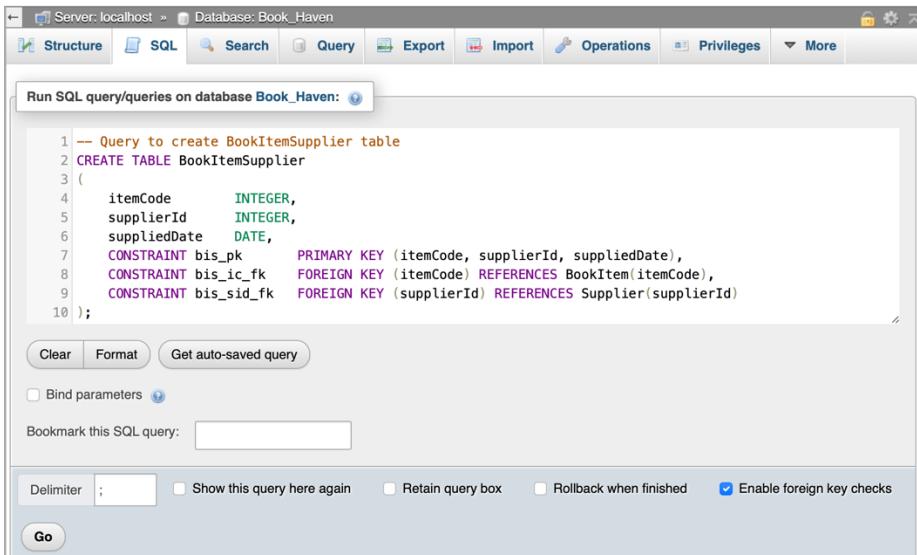
Figure 33 StationeryItem table created successfully.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	itemCode	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	price	decimal(10,2)			No	None			Change Drop More
3	stockLevel	int(11)			No	None			Change Drop More
4	reorderLevel	int(11)			No	None			Change Drop More
5	type	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
6	brand	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
7	color	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
8	size	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Figure 34 StationeryItem table Column Specifications.

BookItemSupplier Table

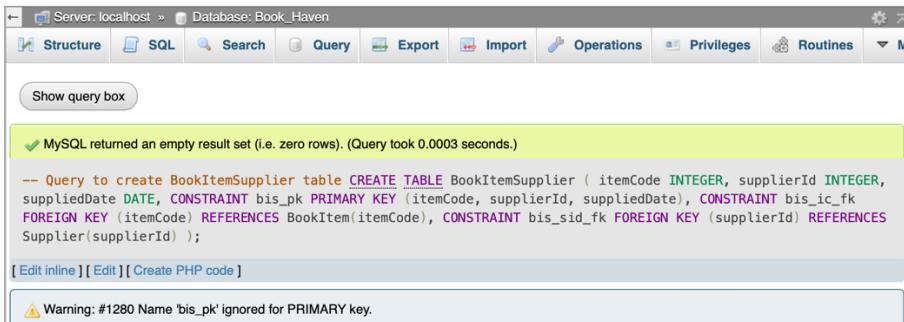


The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A SQL query is entered in the main pane:

```
-- Query to create BookItemSupplier table
CREATE TABLE BookItemSupplier (
    itemCode      INTEGER,
    supplierId   INTEGER,
    suppliedDate DATE,
    CONSTRAINT bis_pk PRIMARY KEY (itemCode, supplierId, suppliedDate),
    CONSTRAINT bis_ic_fk FOREIGN KEY (itemCode) REFERENCES BookItem(itemCode),
    CONSTRAINT bis_sid_fk FOREIGN KEY (supplierId) REFERENCES Supplier(supplierId)
);
```

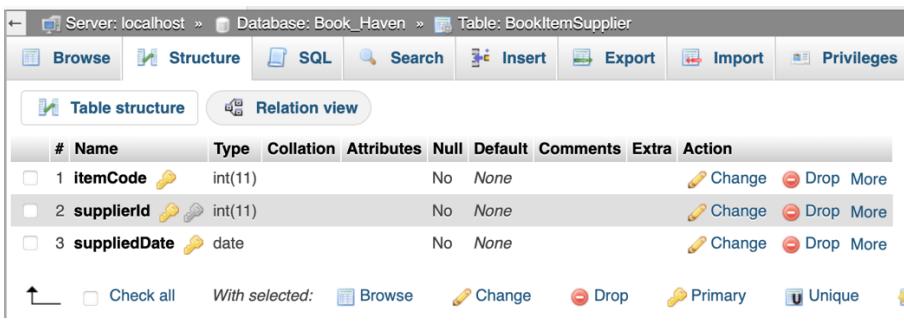
Below the query, there are several buttons: 'Clear', 'Format', 'Get auto-saved query', 'Bind parameters', 'Bookmark this SQL query:', 'Delimiter :', 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks'. A 'Go' button is at the bottom.

Figure 35 Query to create BookItemSupplier table.



The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. The results pane displays a green message: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)'. Below the message, the same CREATE TABLE query is shown. At the bottom of the results pane, a warning message reads: 'Warning: #1280 Name 'bis_pk' ignored for PRIMARY key.'

Figure 36 BookItemSupplier table created successfully.



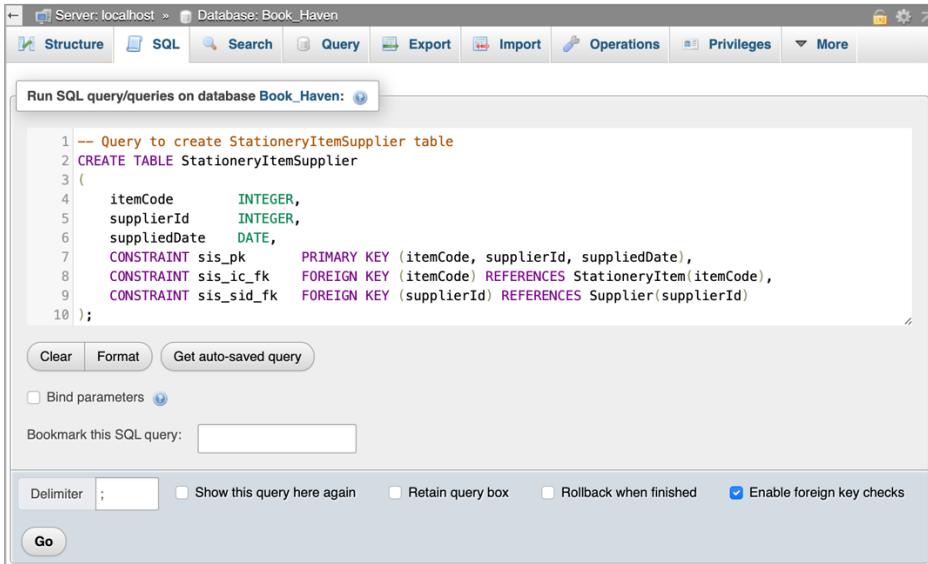
The screenshot shows the MySQL Workbench interface with the 'Structure' tab selected for the BookItemSupplier table. The 'Table structure' view is active, displaying the following column specifications:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	itemCode	int(11)			No	None			Change Drop More
2	supplierId	int(11)			No	None			Change Drop More
3	suppliedDate	date			No	None			Change Drop More

At the bottom of the table view, there are buttons for 'Check all', 'With selected:', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', and 'Import'.

Figure 37 BookItemSupplier table Column Specifications.

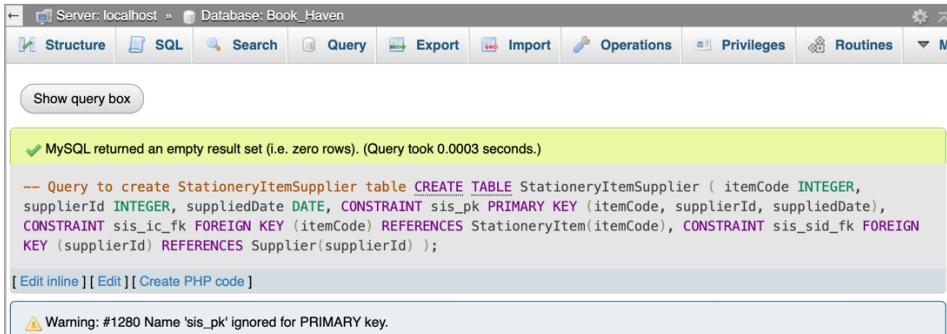
StationeryItemSupplier Table



```
1 -- Query to create StationeryItemSupplier table
2 CREATE TABLE StationeryItemSupplier
3 (
4     itemCode      INTEGER,
5     supplierId   INTEGER,
6     suppliedDate DATE,
7     CONSTRAINT sis_pk    PRIMARY KEY (itemCode, supplierId, suppliedDate),
8     CONSTRAINT sis_ic_fk FOREIGN KEY (itemCode) REFERENCES StationeryItem(itemCode),
9     CONSTRAINT sis_sid_fk FOREIGN KEY (supplierId) REFERENCES Supplier(supplierId)
10 );
```

Clear Format Get auto-saved query
Bind parameters
Bookmark this SQL query:
Delimiter ; Show this query here again Retain query box Rollback when finished Enable foreign key checks
Go

Figure 38 Query to create StationeryItemSupplier table.



Show query box

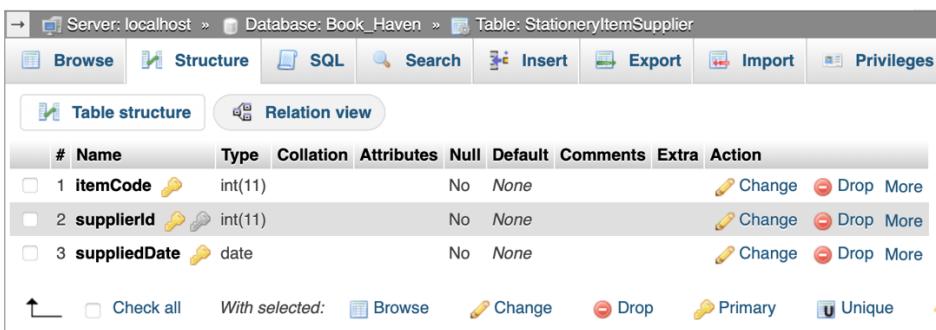
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
-- Query to create StationeryItemSupplier table CREATE TABLE StationeryItemSupplier ( itemCode INTEGER, supplierId INTEGER, suppliedDate DATE, CONSTRAINT sis_pk PRIMARY KEY (itemCode, supplierId, suppliedDate), CONSTRAINT sis_ic_fk FOREIGN KEY (itemCode) REFERENCES StationeryItem(itemCode), CONSTRAINT sis_sid_fk FOREIGN KEY (supplierId) REFERENCES Supplier(supplierId) );
```

[Edit inline] [Edit] [Create PHP code]

Warning: #1280 Name 'sis_pk' ignored for PRIMARY key.

Figure 39 StationeryItemSupplier table created successfully.



Server: localhost > Database: Book_Haven > Table: StationeryItemSupplier

Browse Structure SQL Search Insert Export Import Privileges

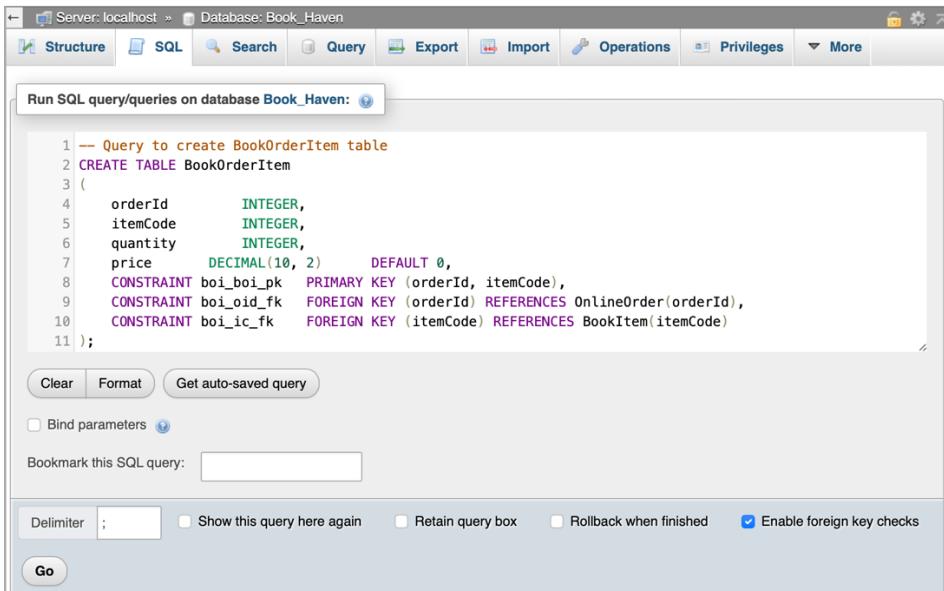
Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	itemCode	int(11)			No	None			Change Drop More
2	supplierId	int(11)			No	None			Change Drop More
3	suppliedDate	date			No	None			Change Drop More

Check all With selected: Browse Change Drop Primary Unique

Figure 40 StationeryItemSupplier table Column Specifications.

BookOrderItem Table

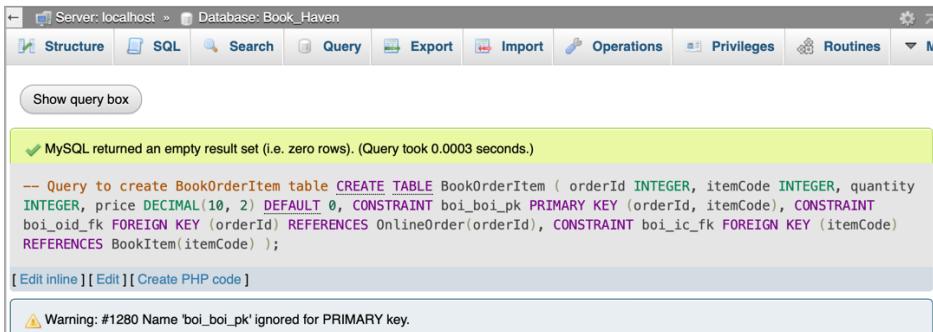


The screenshot shows the MySQL Workbench interface with the SQL tab selected. A query window displays the SQL code for creating the BookOrderItem table:

```
-- Query to create BookOrderItem table
CREATE TABLE BookOrderItem
(
    orderId      INTEGER,
    itemCode     INTEGER,
    quantity     INTEGER,
    price        DECIMAL(10, 2)      DEFAULT 0,
    CONSTRAINT boi_boi_pk PRIMARY KEY (orderId, itemCode),
    CONSTRAINT boi_oid_fk FOREIGN KEY (orderId) REFERENCES OnlineOrder(orderId),
    CONSTRAINT boi_ic_fk FOREIGN KEY (itemCode) REFERENCES BookItem(itemCode)
);
```

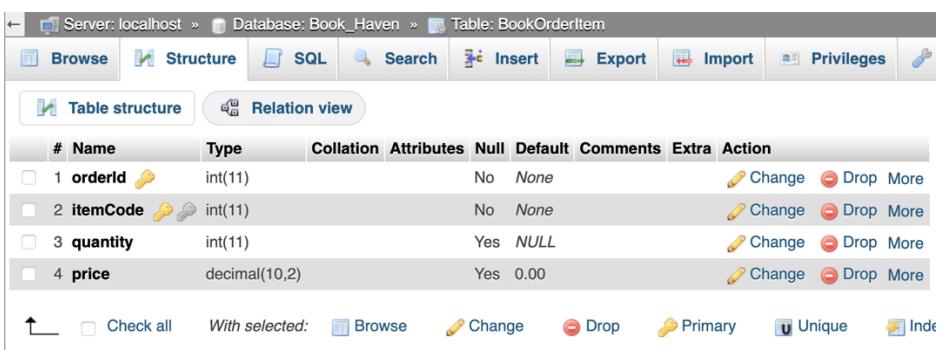
Below the query window, there are several buttons: Clear, Format, Get auto-saved query, Bind parameters, and a bookmark field. At the bottom, there are options for Delimiter, Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A "Go" button is also present.

Figure 41 Query to create BookOrderItem table.



The screenshot shows the MySQL Workbench interface with the SQL tab selected. The results pane indicates that the query was successful and returned zero rows. The query itself is the same as in Figure 41. Below the results, a warning message is displayed: "Warning: #1280 Name 'boi_boi_pk' ignored for PRIMARY key".

Figure 42 BookOrderItem table created successfully.



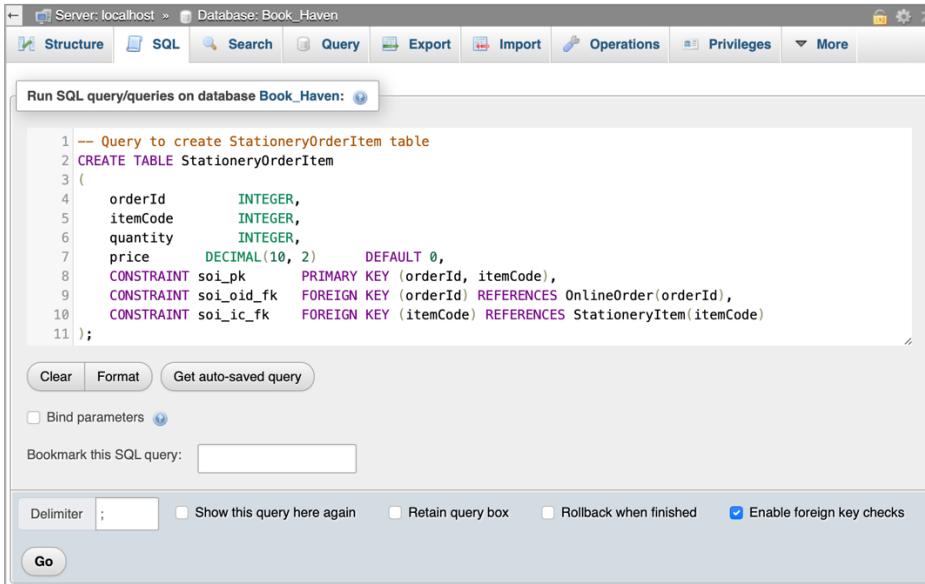
The screenshot shows the MySQL Workbench interface with the Structure tab selected for the BookOrderItem table. The table structure is defined as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	orderId	int(11)			No	None			Change Drop More
2	itemCode	int(11)			No	None			Change Drop More
3	quantity	int(11)			Yes	NULL			Change Drop More
4	price	decimal(10,2)			Yes	0.00			Change Drop More

At the bottom, there are buttons for Check all, With selected, and various table operations like Browse, Change, Drop, Primary, Unique, and Index.

Figure 43 BookOrderItem table Column Specifications.

StationeryOrderItem Table

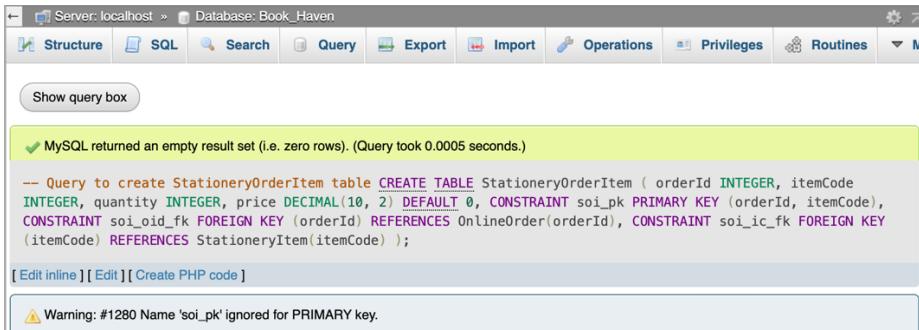


The screenshot shows the MySQL Workbench interface with the SQL tab selected. A SQL query is entered in the main pane:

```
-- Query to create StationeryOrderItem table
CREATE TABLE StationeryOrderItem (
    orderId      INTEGER,
    itemCode     INTEGER,
    quantity     INTEGER,
    price        DECIMAL(10, 2)      DEFAULT 0,
    CONSTRAINT soi_pk PRIMARY KEY (orderId, itemCode),
    CONSTRAINT soi_oid_fk FOREIGN KEY (orderId) REFERENCES OnlineOrder(orderId),
    CONSTRAINT soi_ic_fk FOREIGN KEY (itemCode) REFERENCES StationeryItem(itemCode)
);
```

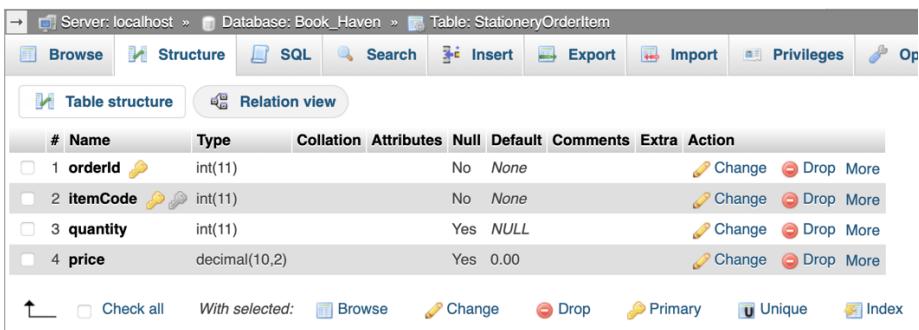
Below the query, there are several buttons: Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query:, Delimiter :, Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A 'Go' button is at the bottom.

Figure 44 Query to create StationeryOrderItem table.



The screenshot shows the MySQL Workbench interface with the SQL tab selected. The results pane displays a green message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)". Below the message is the same CREATE TABLE query as in Figure 44. At the bottom of the results pane, there is a warning message: "Warning: #1280 Name 'soi_pk' ignored for PRIMARY key." There is also a link to "Edit inline" and "Create PHP code".

Figure 45 StationeryOrderItem table created successfully.



The screenshot shows the MySQL Workbench interface with the Structure tab selected for the StationeryOrderItem table. The table structure is displayed in a grid:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	orderId	int(11)			No	None			Change Drop More
2	itemCode	int(11)			No	None			Change Drop More
3	quantity	int(11)			Yes	NULL			Change Drop More
4	price	decimal(10,2)			Yes	0.00			Change Drop More

At the bottom of the interface, there are buttons for Check all, With selected:, Browse, Change, Drop, Primary, Unique, and Index.

Figure 46 StationeryOrderItem table Column Specifications.

Inserting data to each table

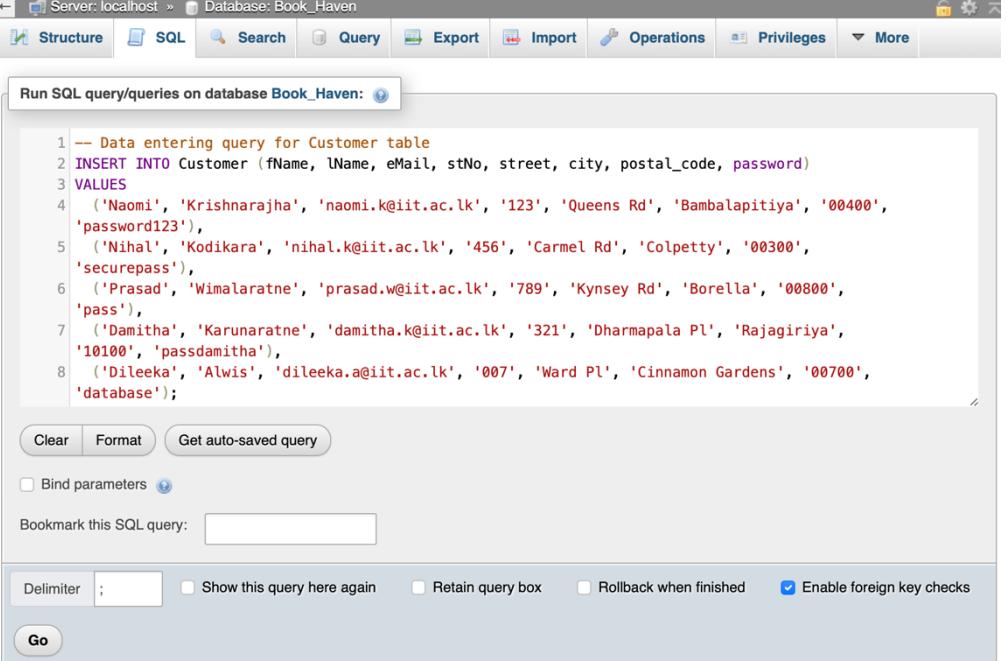
A crucial component in the 'OnlineOrder' table named 'orderId' keeps this database arranged nicely. It links to other tables such as [OnlinePayment](#), [BankTransferPayment](#), [BookOrderItem](#), [StationeryOrderItem](#) and [Delivery](#) in addition to serving as the primary identifier. As a result, information in 'OnlineOrder' must come before any linked tables. Because it is foreign key for [OnlinePayment](#), [BankTransferPayment](#), [BookOrderItem](#), [StationeryOrderItem](#) and [Delivery](#) and foreign keys can't be null

Data entry proceeds in an easily understood way: begin with the [Supplier](#)→[SupplierContact](#), [BookItem](#), [StationeryItem](#), and [Customer](#)→[CustomerContact](#) tables, then fill the intermediate [BookItemSupplier](#) and [StationeryItemSupplier](#), then proceed through [OnlineOrder](#) → [BookOrderItem](#) or [StationeryOrderItem](#) → [OnlinePayment](#) or [BankTransferPayment](#) → [Delivery](#).

Triggers automate certain activities to improve efficiency. The price in [BookOrderItem](#) and [StationeryOrderItem](#) is calculated by multiplying the quantity by the price relevant to the [itemCode](#). Additionally calculating the sum of the price in either [BookOrderItem](#) or [StationeryOrderItem](#) relevant to a single [orderId](#) to update the [totalPrice](#) in the [OnlineOrder](#) Table.

In summary, triggers automate crucial procedures for an efficient database structure, and this database is thoroughly structured to ensure that data is entered in an organized manner.

Customer Table



```

1 -- Data entering query for Customer table
2 INSERT INTO Customer (fName, lName, eMail, stNo, street, city, postal_code, password)
3 VALUES
4 ('Naomi', 'Krishnarajha', 'naomi.k@iit.ac.lk', '123', 'Queens Rd', 'Bambalapitiya', '00400',
5 'password123'),
6 ('Nihal', 'Kodikara', 'nihal.k@iit.ac.lk', '456', 'Carmel Rd', 'Colpetty', '00300',
7 'securepass'),
8 ('Prasad', 'Wimalaratne', 'prasad.w@iit.ac.lk', '789', 'Kynsey Rd', 'Borella', '00800',
9 'pass'),
10 ('Damitha', 'Karunaratne', 'damitha.k@iit.ac.lk', '321', 'Dharmapala Pl', 'Rajagiriya',
11 '10100', 'passdamitha'),
12 ('Dileeka', 'Alwis', 'dileeka.a@iit.ac.lk', '007', 'Ward Pl', 'Cinnamon Gardens', '00700',
13 'database');

```

Clear Format Get auto-saved query
 Bind parameters
Bookmark this SQL query:
Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks

Figure 47 Data entering query for Customer table.



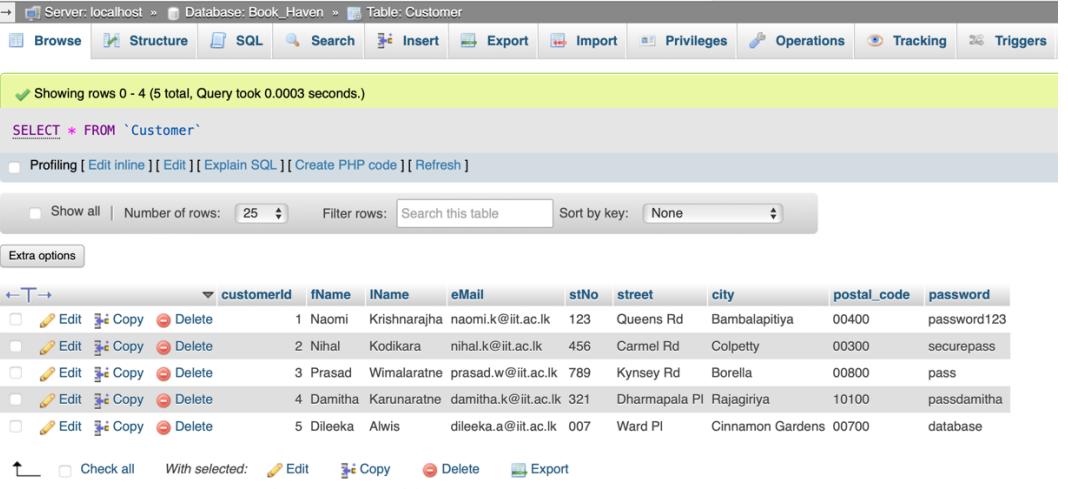
5 rows inserted.
Inserted row id: 5 (Query took 0.0086 seconds.)

```

-- Data entering query for Customer table INSERT INTO Customer (fName, lName, eMail, stNo, street, city,
postal_code, password) VALUES ('Naomi', 'Krishnarajha', 'naomi.k@iit.ac.lk', '123', 'Queens Rd',
'Bambalapitiya', '00400', 'password123'), ('Nihal', 'Kodikara', 'nihal.k@iit.ac.lk', '456', 'Carmel Rd',
'Colpetty', '00300', 'securepass'), ('Prasad', 'Wimalaratne', 'prasad.w@iit.ac.lk', '789', 'Kynsey Rd',
'Borella', '00800', 'pass'), ('Damitha', 'Karunaratne', 'damitha.k@iit.ac.lk', '321', 'Dharmapala Pl',
'Rajagiriya', '10100', 'passdamitha'), ('Dileeka', 'Alwis', 'dileeka.a@iit.ac.lk', '007', 'Ward Pl', 'Cinnamon
[Goto] [Edit] [Create PHP code]

```

Figure 48 Data entered successfully to Customer table.



Showing rows 0 - 4 (5 total, Query took 0.0003 seconds.)

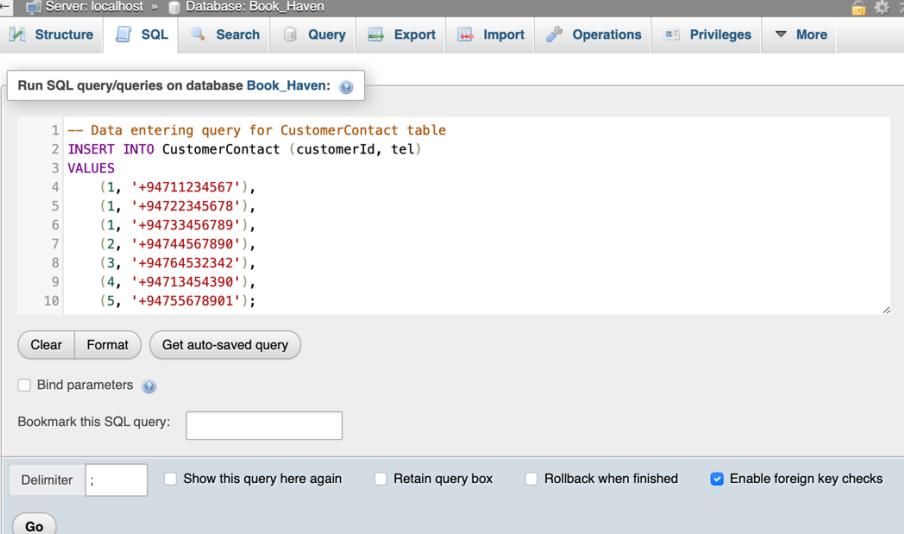
SELECT * FROM `Customer`

	customerID	fName	lName	eMail	stNo	street	city	postal_code	password
<input type="checkbox"/>	1	Naomi	Krishnarajha	naomi.k@iit.ac.lk	123	Queens Rd	Bambalapitiya	00400	password123
<input type="checkbox"/>	2	Nihal	Kodikara	nihal.k@iit.ac.lk	456	Carmel Rd	Colpetty	00300	securepass
<input type="checkbox"/>	3	Prasad	Wimalaratne	prasad.w@iit.ac.lk	789	Kynsey Rd	Borella	00800	pass
<input type="checkbox"/>	4	Damitha	Karunaratne	damitha.k@iit.ac.lk	321	Dharmapala Pl	Rajagiriya	10100	passdamitha
<input type="checkbox"/>	5	Dileeka	Alwis	dileeka.a@iit.ac.lk	007	Ward Pl	Cinnamon Gardens	00700	database

Check all With selected:

Figure 49 Data entered Customer table.

CustomerContact Table



The screenshot shows the MySQL Workbench interface with the following details:

- Server: localhost
- Database: Book_Haven
- Structure tab is selected.
- SQL tab is active.
- Query box contains the following SQL code:

```
1 -- Data entering query for CustomerContact table
2 INSERT INTO CustomerContact (customerId, tel)
3 VALUES
4     (1, '+94711234567'),
5     (1, '+94722345678'),
6     (1, '+94733456789'),
7     (2, '+94744567890'),
8     (3, '+94764532342'),
9     (4, '+94713454390'),
10    (5, '+94755678901');
```

Below the query box are several buttons: Clear, Format, Get auto-saved query, Bind parameters, and Bookmark this SQL query. At the bottom are settings for Delimiter (set to ;), Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A "Go" button is at the bottom right.

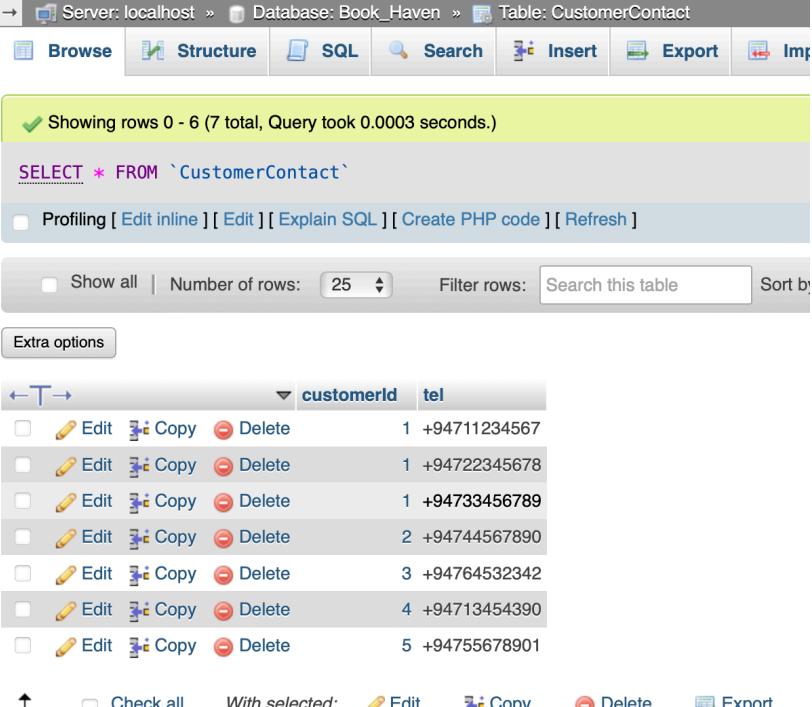
Figure 50 Data entering query for CustomerContact table.



The screenshot shows the MySQL Workbench interface with the following details:

- Server: localhost
- Database: Book_Haven
- Routines tab is selected.
- Show query box button is visible.
- Query result area displays: "7 rows inserted. (Query took 0.0006 seconds.)" followed by the same SQL insert statement as in Figure 50.
- Buttons at the bottom include Edit inline, Edit, Create PHP code, and Refresh.

Figure 51 Data entered successfully to CustomerContact table.



The screenshot shows the MySQL Workbench interface with the following details:

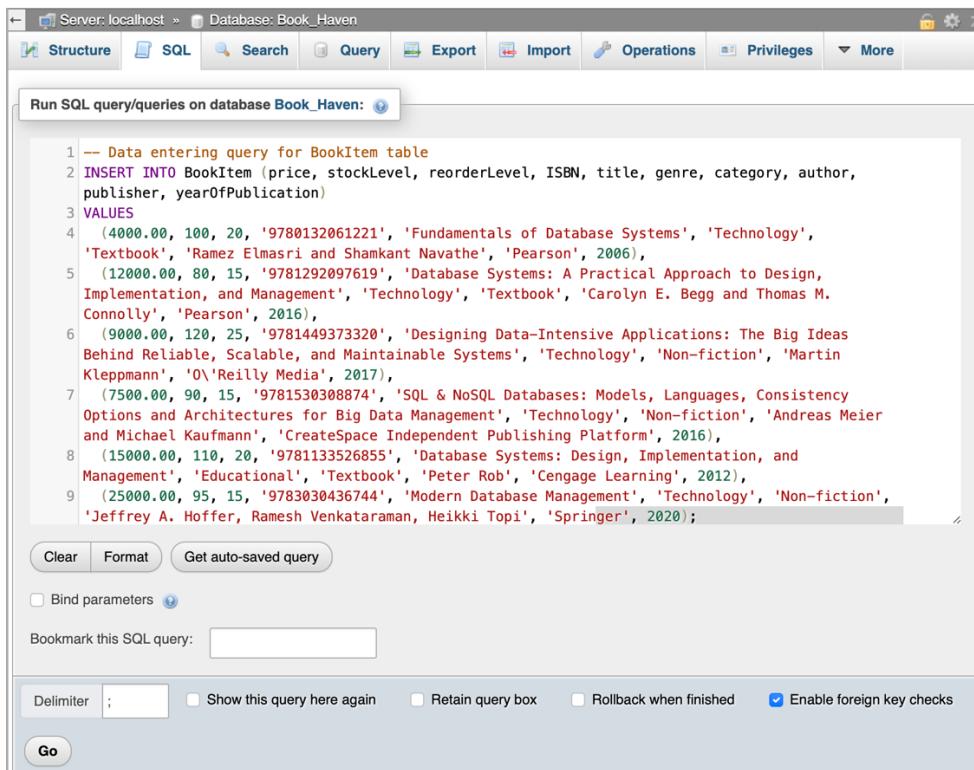
- Server: localhost
- Database: Book_Haven
- Table: CustomerContact
- Structure tab is selected.
- Table data view shows the following rows:

	customerId	tel
<input type="checkbox"/>	1	+94711234567
<input type="checkbox"/>	1	+94722345678
<input type="checkbox"/>	1	+94733456789
<input type="checkbox"/>	2	+94744567890
<input type="checkbox"/>	3	+94764532342
<input type="checkbox"/>	4	+94713454390
<input type="checkbox"/>	5	+94755678901

At the bottom, there are buttons for Check all, With selected: (Edit, Copy, Delete), Export, and Import.

Figure 52 Data entered CustomerContact table.

BookItem Table



The screenshot shows the MySQL Workbench interface with the following details:

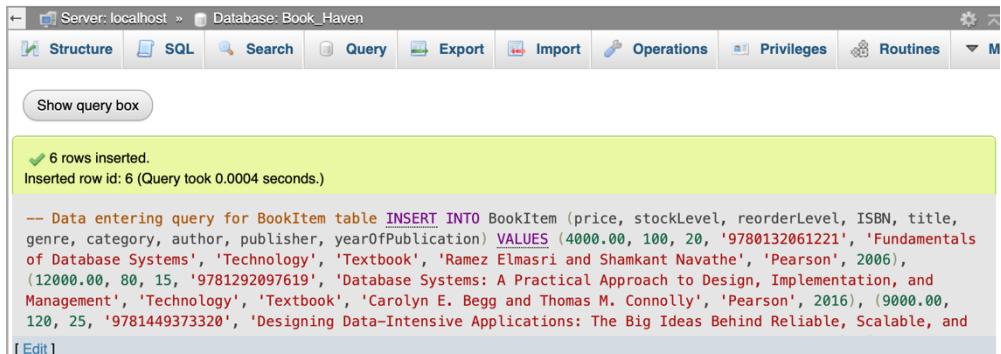
- Server:** localhost
- Database:** Book_Haven
- Tab:** SQL
- Query:**

```

1 -- Data entering query for BookItem table
2 INSERT INTO BookItem (price, stockLevel, reorderLevel, ISBN, title, genre, category, author, publisher, yearOfPublication)
3 VALUES
4     (4000.00, 100, 20, '9780132061221', 'Fundamentals of Database Systems', 'Technology', 'Textbook', 'Ramez Elmasri and Shamkant Navathe', 'Pearson', 2006),
5     (12000.00, 80, 15, '9781292097619', 'Database Systems: A Practical Approach to Design, Implementation, and Management', 'Technology', 'Textbook', 'Carolyn E. Begg and Thomas M. Connolly', 'Pearson', 2016),
6     (9000.00, 120, 25, '9781449373320', 'Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems', 'Technology', 'Non-fiction', 'Martin Kleppmann', 'O'Reilly Media', 2017),
7     (7500.00, 90, 15, '9781530308874', 'SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management', 'Technology', 'Non-fiction', 'Andreas Meier and Michael Kaufmann', 'CreateSpace Independent Publishing Platform', 2016),
8     (15000.00, 110, 20, '9781133526855', 'Database Systems: Design, Implementation, and Management', 'Educational', 'Textbook', 'Peter Rob', 'Cengage Learning', 2012),
9     (25000.00, 95, 15, '9783030436744', 'Modern Database Management', 'Technology', 'Non-fiction', 'Jeffrey A. Hoffer, Ramesh Venkataraman, Heikki Topi', 'Springer', 2020);

```
- Buttons:** Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query, Delimiter ;, Show this query here again, Retain query box, Rollback when finished, Enable foreign key checks.
- Buttons at bottom:** Go

Figure 53 Data entering query for BookItem table.



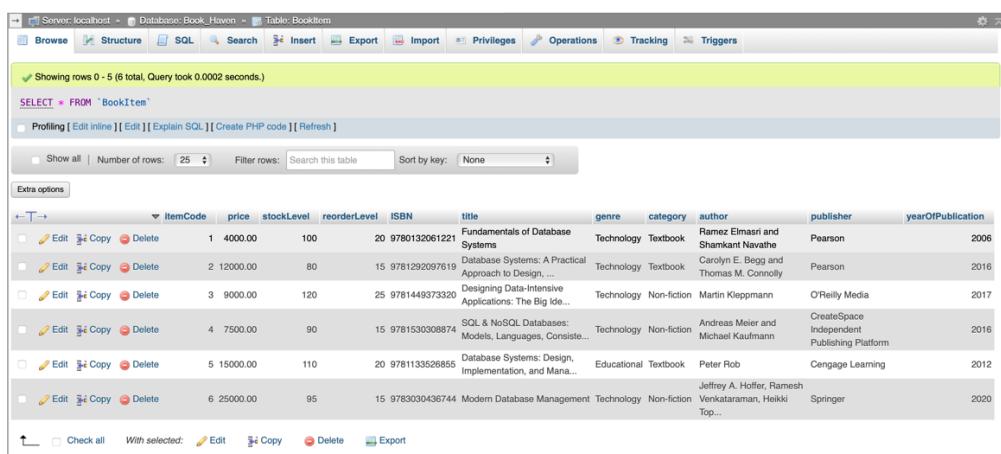
The screenshot shows the MySQL Workbench interface with the following details:

- Server:** localhost
- Database:** Book_Haven
- Tab:** SQL
- Message Bar:** ✓ 6 rows inserted. Inserted row id: 6 (Query took 0.0004 seconds.)
- Query:**

```
-- Data entering query for BookItem table
INSERT INTO BookItem (price, stockLevel, reorderLevel, ISBN, title, genre, category, author, publisher, yearOfPublication)
VALUES (4000.00, 100, 20, '9780132061221', 'Fundamentals of Database Systems', 'Technology', 'Textbook', 'Ramez Elmasri and Shamkant Navathe', 'Pearson', 2006),
(12000.00, 80, 15, '9781292097619', 'Database Systems: A Practical Approach to Design, Implementation, and Management', 'Technology', 'Textbook', 'Carolyn E. Begg and Thomas M. Connolly', 'Pearson', 2016),
(9000.00, 120, 25, '9781449373320', 'Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and

```
- Buttons at bottom:** [Edit]

Figure 54 Data entered successfully to BookItem table.



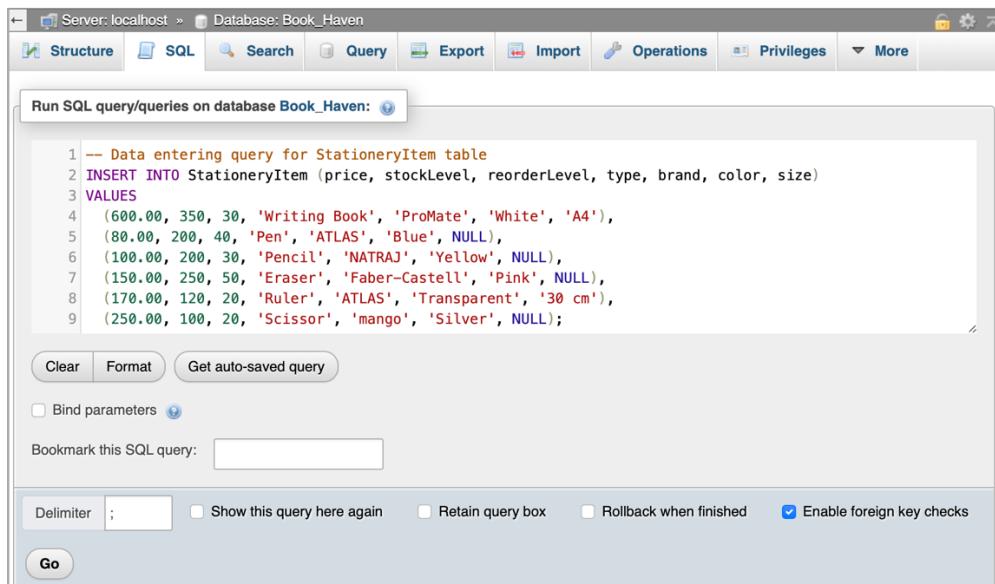
The screenshot shows the MySQL Workbench interface with the following details:

- Server:** localhost
- Database:** Book_Haven
- Table:** BookItem
- Grid:**

	itemCode	price	stockLevel	reorderLevel	ISBN	title	genre	category	author	publisher	yearOfPublication
<input type="checkbox"/>	1	4000.00	100	20	9780132061221	Fundamentals of Database Systems	Technology	Textbook	Ramez Elmasri and Shamkant Navathe	Pearson	2006
<input type="checkbox"/>	2	12000.00	80	15	9781292097619	Database Systems: A Practical Approach to Design, ...	Technology	Textbook	Carolyn E. Begg and Thomas M. Connolly	Pearson	2016
<input type="checkbox"/>	3	9000.00	120	25	9781449373320	Designing Data-Intensive Applications: The Big Ide...	Technology	Non-fiction	Martin Kleppmann	O'Reilly Media	2017
<input type="checkbox"/>	4	7500.00	90	15	9781530308874	SQL & NoSQL Databases: Models, Languages, Consiste...	Technology	Non-fiction	Andreas Meier and Michael Kaufmann	CreateSpace Independent Publishing Platform	2016
<input type="checkbox"/>	5	15000.00	110	20	9781133526855	Database Systems: Design, Implementation, and Mana...	Educational	Textbook	Peter Rob	Cengage Learning	2012
<input type="checkbox"/>	6	25000.00	95	15	9783030436744	Modern Database Management	Technology	Non-fiction	Jeffrey A. Hoffer, Ramesh Venkataraman, Heikki Topi	Springer	2020
- Buttons at bottom:** Check all, With selected: Edit, Copy, Delete, Export

Figure 55 Data entered BookItem tabl.

StationeryItem Table



```

1 -- Data entering query for StationeryItem table
2 INSERT INTO StationeryItem (price, stockLevel, reorderLevel, type, brand, color, size)
3 VALUES
4 (600.00, 350, 30, 'Writing Book', 'ProMate', 'White', 'A4'),
5 (80.00, 200, 40, 'Pen', 'ATLAS', 'Blue', NULL),
6 (100.00, 200, 30, 'Pencil', 'NATRAJ', 'Yellow', NULL),
7 (150.00, 250, 50, 'Eraser', 'Faber-Castell', 'Pink', NULL),
8 (170.00, 120, 20, 'Ruler', 'ATLAS', 'Transparent', '30 cm'),
9 (250.00, 100, 20, 'Scissor', 'mango', 'Silver', NULL);

```

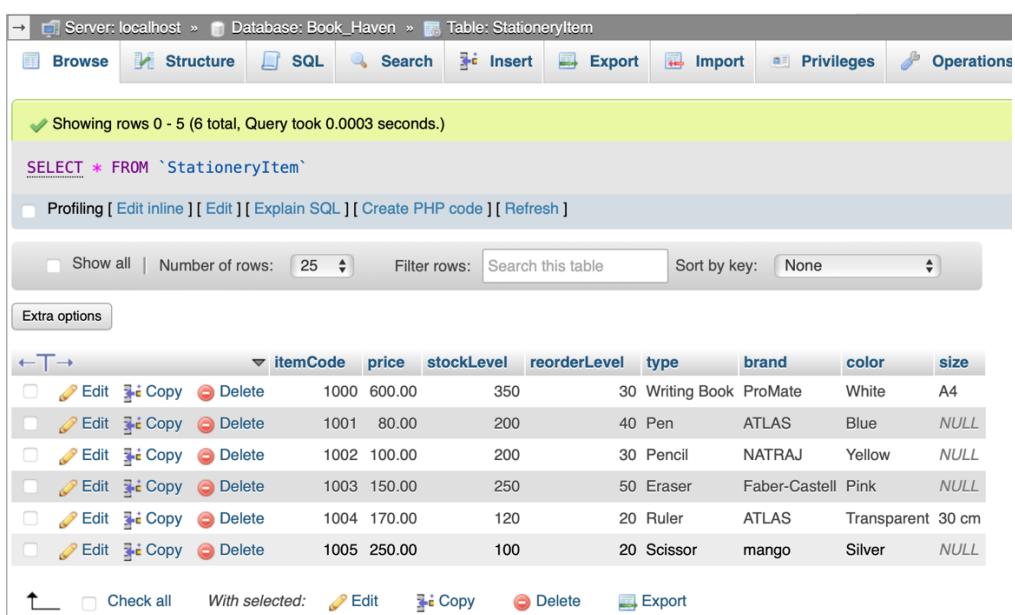
Clear Format Get auto-saved query
 Bind parameters
Bookmark this SQL query:
Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks

Figure 56 Data entering query for StationeryItem table.



Show query box
✓ 6 rows inserted.
Inserted row id: 1005 (Query took 0.0003 seconds.)
-- Data entering query for StationeryItem table `INSERT INTO StationeryItem (price, stockLevel, reorderLevel, type, brand, color, size) VALUES (600.00, 350, 30, 'Writing Book', 'ProMate', 'White', 'A4'), (80.00, 200, 40, 'Pen', 'ATLAS', 'Blue', NULL), (100.00, 200, 30, 'Pencil', 'NATRAJ', 'Yellow', NULL), (150.00, 250, 50, 'Eraser', 'Faber-Castell', 'Pink', NULL), (170.00, 120, 20, 'Ruler', 'ATLAS', 'Transparent', '30 cm'), (250.00, 100, 20, 'Scissor', 'mango', 'Silver', NULL);`
[Edit inline] [Edit] [Create PHP code]

Figure 57 Data entered successfully to StationeryItem table.



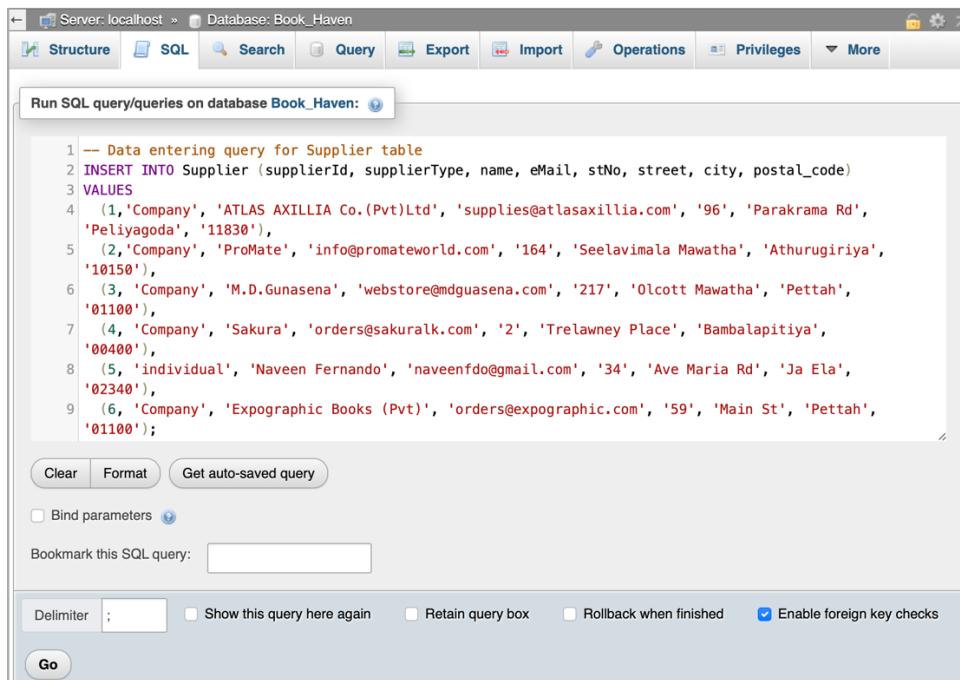
Browse Structure SQL Search Insert Export Import Privileges Operations
✓ Showing rows 0 - 5 (6 total, Query took 0.0003 seconds.)
SELECT * FROM `StationeryItem`
 Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
 Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	itemCode	price	stockLevel	reorderLevel	type	brand	color	size
<input type="checkbox"/>	1000	600.00	350	30	Writing Book	ProMate	White	A4
<input type="checkbox"/>	1001	80.00	200	40	Pen	ATLAS	Blue	NULL
<input type="checkbox"/>	1002	100.00	200	30	Pencil	NATRAJ	Yellow	NULL
<input type="checkbox"/>	1003	150.00	250	50	Eraser	Faber-Castell	Pink	NULL
<input type="checkbox"/>	1004	170.00	120	20	Ruler	ATLAS	Transparent	30 cm
<input type="checkbox"/>	1005	250.00	100	20	Scissor	mango	Silver	NULL

 Check all With selected:

Figure 58 Data entered StationeryItem table.

Supplier Table



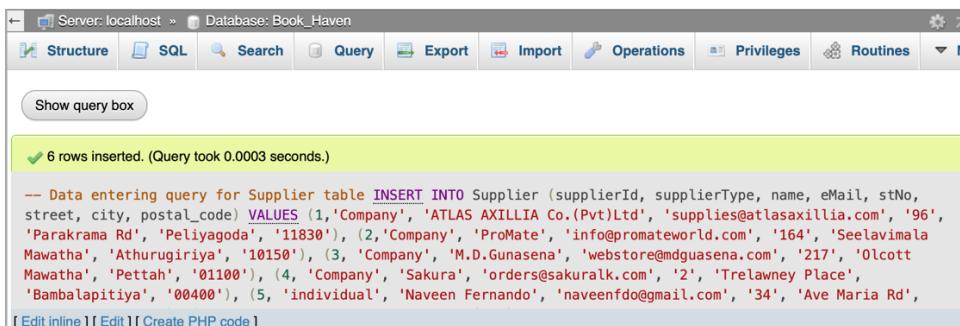
The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. The SQL tab is active, displaying the following SQL code:

```

1 -- Data entering query for Supplier table
2 INSERT INTO Supplier (supplierId, supplierType, name, eMail, stNo, street, city, postal_code)
3 VALUES
4 (1,'Company', 'ATLAS AXILLIA Co.(Pvt)Ltd', 'supplies@atlasaxillia.com', '96', 'Parakrama Rd',
5 'Peliyagoda', '11830'),
6 (2,'Company', 'ProMate', 'info@promateworld.com', '164', 'Seelavimala Mawatha', 'Athurugiriya',
7 '10150'),
8 (3, 'Company', 'M.D.Gunasena', 'webstore@mdguasena.com', '217', 'Olcott Mawatha', 'Pettah',
9 '01100'),
10 (4, 'Company', 'Sakura', 'orders@sakuralk.com', '2', 'Trelawney Place', 'Bambalapitiya',
11 '00400'),
12 (5, 'individual', 'Naveen Fernando', 'naveenfdo@gmail.com', '34', 'Ave Maria Rd', 'Ja Ela',
13 '02340'),
14 (6, 'Company', 'Expographic Books (Pvt)', 'orders@expographic.com', '59', 'Main St', 'Pettah',
15 '01100');
    
```

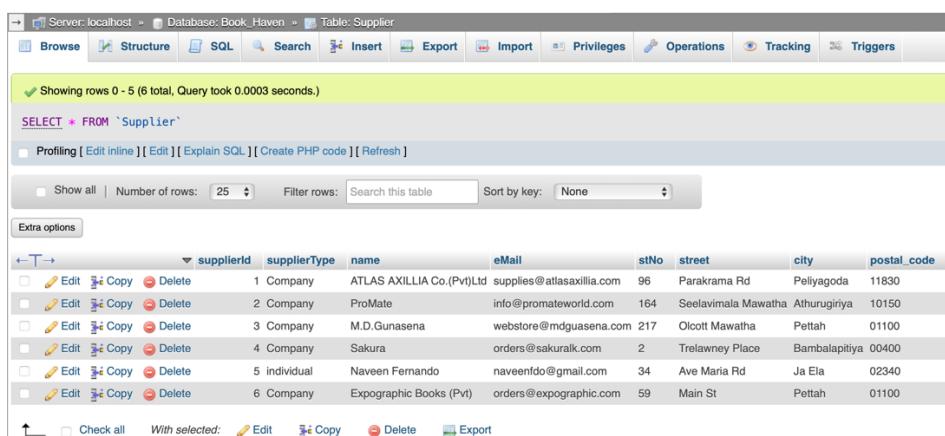
Below the code, there are several buttons: Clear, Format, Get auto-saved query, Bind parameters, and a bookmark input field. At the bottom, there are options for Delimiter (set to ;), Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A large 'Go' button is at the bottom left.

Figure 59 Data entering query for Supplier table.



The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. The SQL tab has a message: '6 rows inserted. (Query took 0.0003 seconds.)'. Below it, the same SQL insert query is shown. At the bottom, there are links for Edit inline, Edit, Create PHP code, Profiling, Explain SQL, and Refresh.

Figure 60 Data entered successfully to Supplier table.

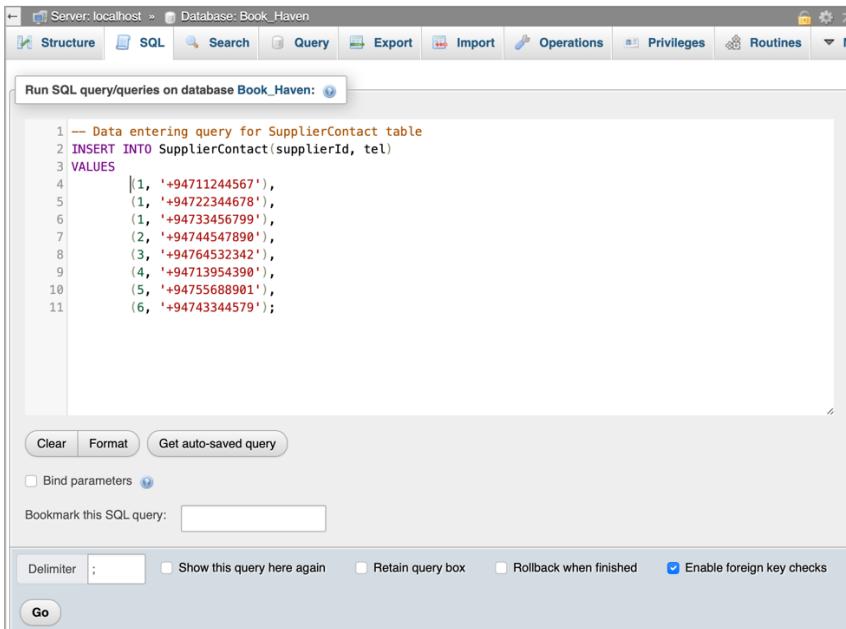


The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven' and the table selected as 'Supplier'. The Structure tab is active. The table data is displayed in a grid:

	supplierId	supplierType	name	eMail	stNo	street	city	postal_code
<input type="checkbox"/>	1	Company	ATLAS AXILLIA Co.(Pvt)Ltd	supplies@atlasaxillia.com	96	Parakrama Rd	Peliyagoda	11830
<input type="checkbox"/>	2	Company	ProMate	info@promateworld.com	164	Seelavimala Mawatha	Athurugiriya	10150
<input type="checkbox"/>	3	Company	M.D.Gunasena	webstore@mdguasena.com	217	Olcott Mawatha	Pettah	01100
<input type="checkbox"/>	4	Company	Sakura	orders@sakuralk.com	2	Trelawney Place	Bambalapitiya	00400
<input type="checkbox"/>	5	individual	Naveen Fernando	naveenfdo@gmail.com	34	Ave Maria Rd	Ja Ela	02340
<input type="checkbox"/>	6	Company	Expographic Books (Pvt)	orders@expographic.com	59	Main St	Pettah	01100

Figure 61 Data entered Supplier table.

SupplierContact Table

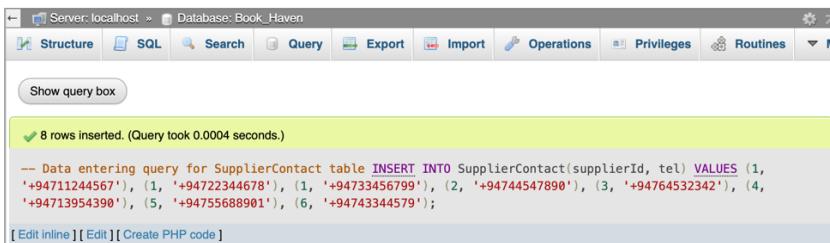


The screenshot shows the MySQL Workbench interface with the SQL tab selected. A query window titled "Run SQL query/queries on database Book_Haven:" contains the following SQL code:

```
1 -- Data entering query for SupplierContact table
2 INSERT INTO SupplierContact(supplierId, tel)
3 VALUES
4     (1, '+94711244567'),
5     (1, '+94722344678'),
6     (1, '+94733456799'),
7     (2, '+94744547890'),
8     (3, '+94764532342'),
9     (4, '+94713954390'),
10    (5, '+94755688901'),
11    (6, '+94743344579');
```

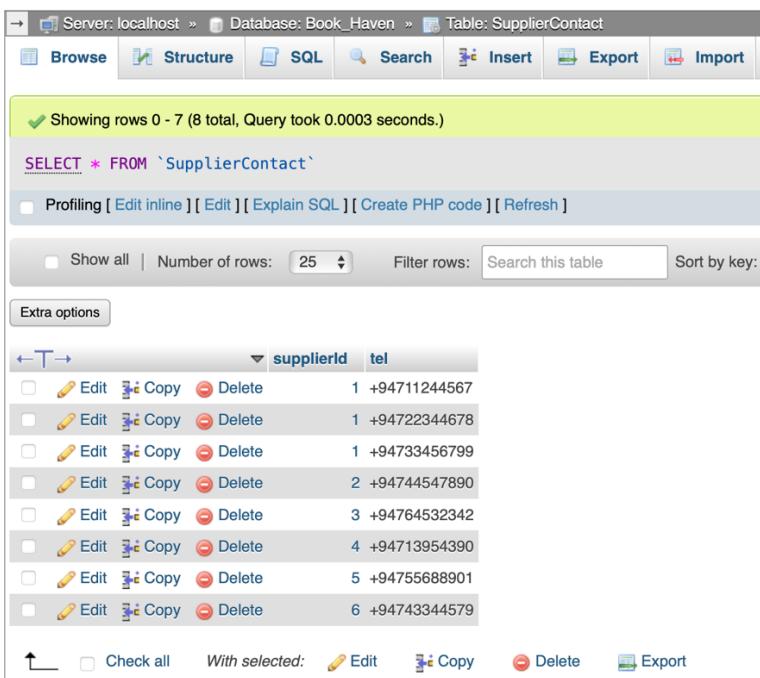
Below the query window, there are several buttons: Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query, Delimiter (set to ;), Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A "Go" button is at the bottom.

Figure 62 Data entering query for SupplierContact table.



The screenshot shows the MySQL Workbench interface with the SQL tab selected. The results pane displays a green message: "8 rows inserted. (Query took 0.0004 seconds)". Below the message is the same SQL insert query as in Figure 62. At the bottom of the results pane are links for [Edit inline], [Edit], and [Create PHP code].

Figure 63 Data entered successfully to SupplierContact table.

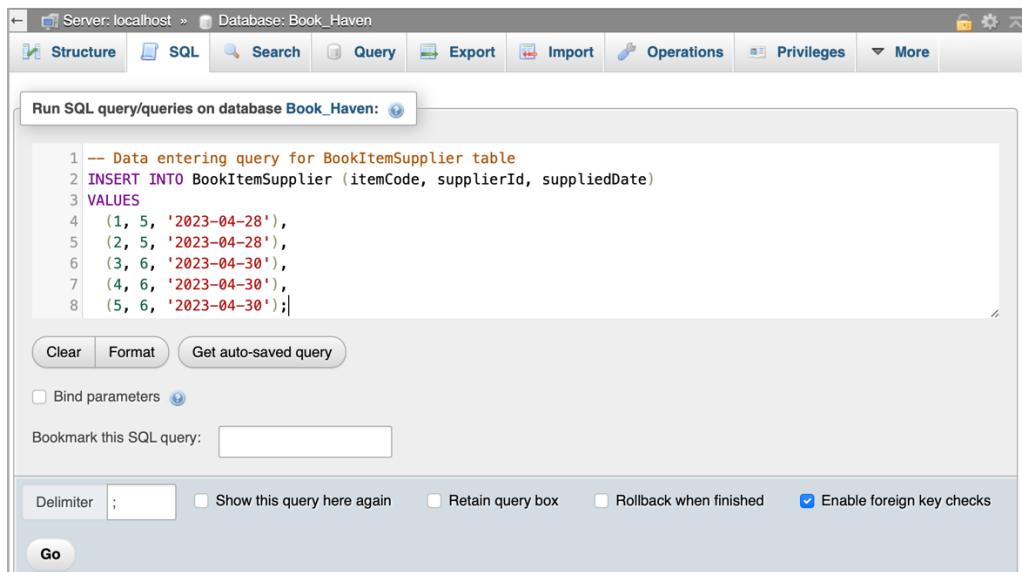


The screenshot shows the MySQL Workbench interface with the Structure tab selected. The table "SupplierContact" is displayed with the following data:

	supplierId	tel
<input type="checkbox"/>	1	+94711244567
<input type="checkbox"/>	1	+94722344678
<input type="checkbox"/>	1	+94733456799
<input type="checkbox"/>	2	+94744547890
<input type="checkbox"/>	3	+94764532342
<input type="checkbox"/>	4	+94713954390
<input type="checkbox"/>	5	+94755688901
<input type="checkbox"/>	6	+94743344579

Figure 64 Data entered SupplierContact table.

BookItemSupplier Table



The screenshot shows the MySQL Workbench interface with the following details:

- Server: localhost
- Database: Book_Haven
- Tab: SQL
- Query window:

```
1 -- Data entering query for BookItemSupplier table
2 INSERT INTO BookItemSupplier (itemCode, supplierId, suppliedDate)
3 VALUES
4 (1, 5, '2023-04-28'),
5 (2, 5, '2023-04-28'),
6 (3, 6, '2023-04-30'),
7 (4, 6, '2023-04-30'),
8 (5, 6, '2023-04-30');
```
- Buttons: Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query, Delimiter (set to ;), Show this query here again, Retain query box, Rollback when finished, Enable foreign key checks.
- Go button

Figure 65 Data entering query for BookItemSupplier table.



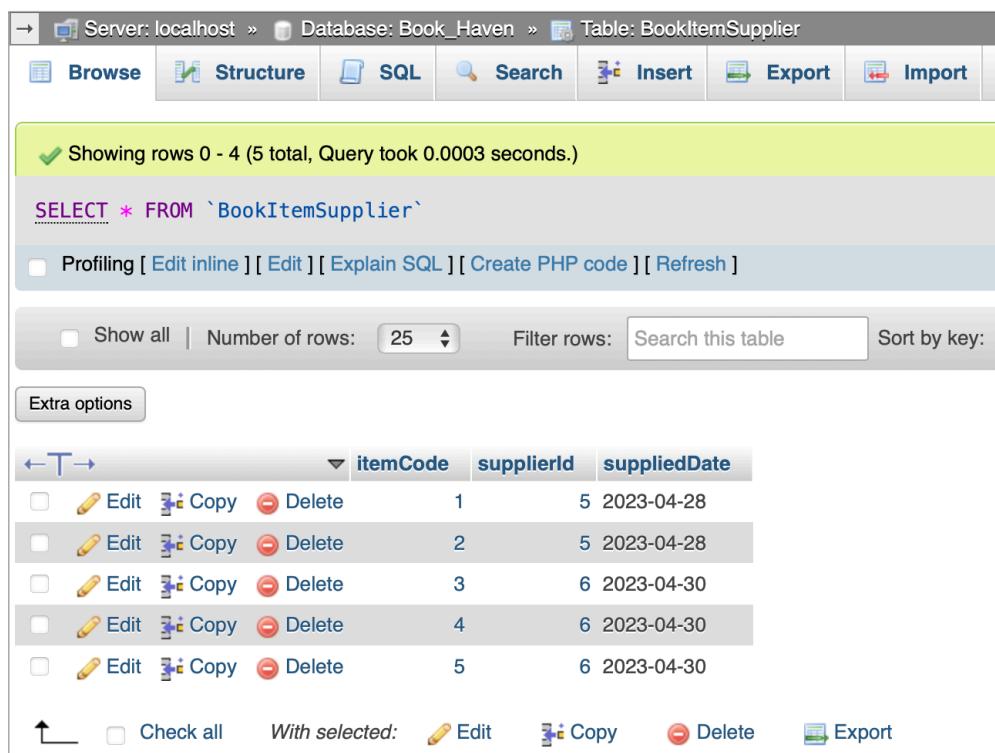
The screenshot shows the MySQL Workbench interface with the following details:

- Server: localhost
- Database: Book_Haven
- Tab: SQL
- Query window:

```
✓ 5 rows inserted. (Query took 0.0006 seconds.)
```

```
-- Data entering query for BookItemSupplier table INSERT INTO BookItemSupplier (itemCode, supplierId, suppliedDate) VALUES (1, 5, '2023-04-28'), (2, 5, '2023-04-28'), (3, 6, '2023-04-30'), (4, 6, '2023-04-30'), (5, 6, '2023-04-30');
```
- Buttons: Edit inline, Edit, Create PHP code.

Figure 66 Data entered successfully to BookItemSupplier table.



The screenshot shows the MySQL Workbench interface with the following details:

- Server: localhost
- Database: Book_Haven
- Table: BookItemSupplier
- Tab: Structure
- Query window:

```
SELECT * FROM `BookItemSupplier`
```
- Buttons: Profiling, Edit inline, Explain SQL, Create PHP code, Refresh.
- Table header: itemCode, supplierId, suppliedDate
- Data:

	itemCode	supplierId	suppliedDate
<input type="checkbox"/>	1	5	2023-04-28
<input type="checkbox"/>	2	5	2023-04-28
<input type="checkbox"/>	3	6	2023-04-30
<input type="checkbox"/>	4	6	2023-04-30
<input type="checkbox"/>	5	6	2023-04-30
- Buttons: Show all, Number of rows: 25, Filter rows: Search this table, Sort by key:.
- Extra options: Check all, With selected: Edit, Copy, Delete, Export.

Figure 67 Data entered BookItemSupplier table.

StationeryItemSupplier Table

The screenshot shows the phpMyAdmin interface with the SQL tab selected. A query window contains the following SQL code:

```

1 -- Data entering query for StationeryItemSupplier
2 INSERT INTO StationeryItemSupplier (itemCode, supplierId, suppliedDate)
3 VALUES
4     (1000, 2, '2023-04-15'),
5     (1001, 1, '2023-04-04'),
6     (1002, 3, '2023-04-04'),
7     (1003, 4, '2023-04-05'),
8     (1004, 1, '2023-04-07'),
9     (1005, 2, '2023-04-15');

```

Below the query window are several buttons: Clear, Format, Get auto-saved query, Bind parameters, and a bookmark input field. At the bottom are Delimiter, Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks checkboxes, with the latter checked.

Figure 68 Data entering query for StationeryItemSupplier table.

The screenshot shows the phpMyAdmin interface with the SQL tab selected. The query window displays the executed query and its result:

```

-- Data entering query for StationeryItemSupplier INSERT INTO StationeryItemSupplier (itemCode, supplierId, suppliedDate) VALUES (1000, 2, '2023-04-15'), (1001, 1, '2023-04-04'), (1002, 3, '2023-04-04'), (1003, 4, '2023-04-05'), (1004, 1, '2023-04-07'), (1005, 2, '2023-04-15');

```

A green message bar at the top indicates "6 rows inserted. (Query took 0.0003 seconds.)". Below the message, there are Edit, Create PHP code, and Refresh buttons.

Figure 69 Data entered successfully to StationeryItemSupplier table.

The screenshot shows the phpMyAdmin interface with the Browse tab selected. The table structure for StationeryItemSupplier is displayed, showing columns: itemCode, supplierId, and suppliedDate. The data grid lists the following entries:

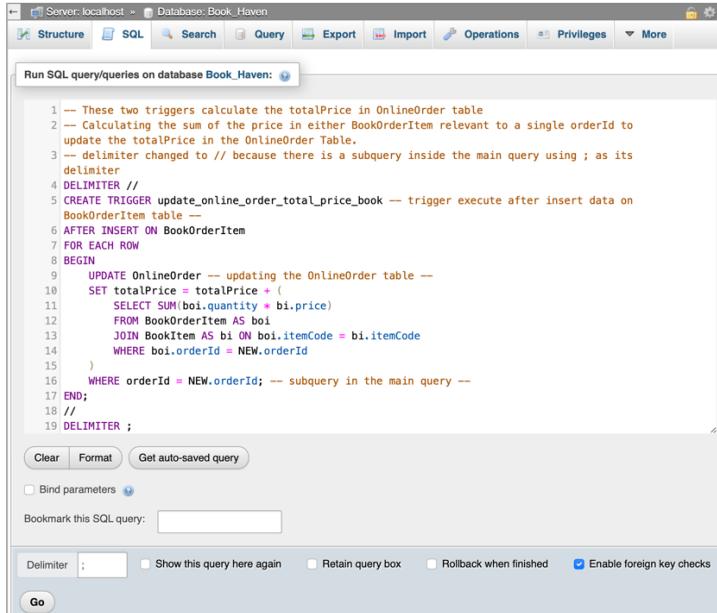
	itemCode	supplierId	suppliedDate
<input type="checkbox"/>	1000	2	2023-04-15
<input type="checkbox"/>	1001	1	2023-04-04
<input type="checkbox"/>	1002	3	2023-04-04
<input type="checkbox"/>	1003	4	2023-04-05
<input type="checkbox"/>	1004	1	2023-04-07
<input type="checkbox"/>	1005	2	2023-04-15

At the bottom of the table area are buttons for Check all, With selected: (Edit, Copy, Delete), and Export.

Figure 70 Data entered StationeryItemSupplier table.

Trigger 1

These two triggers calculate the totalPrice in OnlineOrder table.



The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A query window titled 'Run SQL query/queries on database Book_Haven:' contains the following SQL code:

```
1 -- These two triggers calculate the totalPrice in OnlineOrder table
2 -- Calculating the sum of the price in either BookOrderItem relevant to a single orderId to
3 -- update the totalPrice in the OnlineOrder Table.
4 DELIMITER //
5 CREATE TRIGGER update_online_order_total_price_book -- trigger execute after insert data on
BookOrderItem table --
6 AFTER INSERT ON BookOrderItem
7 FOR EACH ROW
8 BEGIN
9   UPDATE OnlineOrder -- updating the OnlineOrder table --
SET totalPrice = totalPrice +
10  SELECT SUM(boi.quantity * bi.price)
11    FROM BookOrderItem AS boi
12      JOIN BookItem AS bi ON boi.itemCode = bi.itemCode
13 WHERE boi.orderId = NEW.orderId
14 )
15 WHERE orderId = NEW.orderId; -- subquery in the main query --
16 END;
17 //
18 DELIMITER ;
```

Below the code, there are several buttons: 'Clear', 'Format', 'Get auto-saved query', 'Bind parameters', 'Bookmark this SQL query:', 'Delimiter :', 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks'. A 'Go' button is at the bottom.

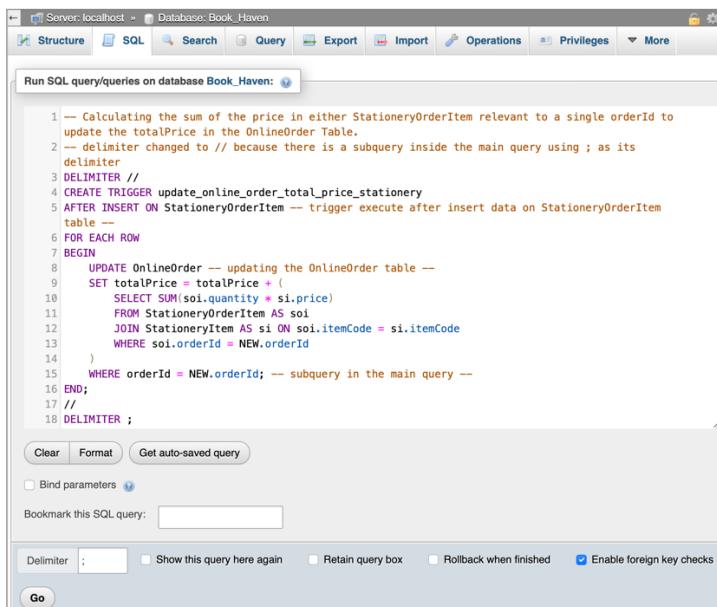


The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A query window titled 'Run SQL query/queries on database Book_Haven:' displays the results of the trigger creation query. It shows a green message: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0146 seconds.)' and the full trigger definition:

```
CREATE TRIGGER update_online_order_total_price_book -- trigger execute after insert data on BookOrderItem table
-- AFTER INSERT ON BookOrderItem FOR EACH ROW BEGIN UPDATE OnlineOrder -- updating the OnlineOrder table -- SET
totalPrice = totalPrice + ( SELECT SUM(boi.quantity * bi.price) FROM BookOrderItem AS boi JOIN BookItem AS bi
ON boi.itemCode = bi.itemCode WHERE boi.orderId = NEW.orderId ) WHERE orderId = NEW.orderId; -- subquery in the
main query -- END;
```

Below the code, there are buttons: 'Edit inline', 'Edit', and 'Create PHP code'.

Figure 71 Trigger that update the totalPrice in OnlineOrder table(book)



The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A query window titled 'Run SQL query/queries on database Book_Haven:' contains the following SQL code:

```
1 -- Calculating the sum of the price in either StationeryOrderItem relevant to a single orderId to
update the totalPrice in the OnlineOrder Table.
2 -- delimiter changed to // because there is a subquery inside the main query using ; as its
delimiter
3 DELIMITER //
4 CREATE TRIGGER update_online_order_total_price_stationery
5 AFTER INSERT ON StationeryOrderItem -- trigger execute after insert data on StationeryOrderItem
table --
6 FOR EACH ROW
7 BEGIN
8   UPDATE OnlineOrder -- updating the OnlineOrder table --
SET totalPrice = totalPrice +
9  SELECT SUM(soi.quantity * si.price)
10  FROM StationeryOrderItem AS soi
11    JOIN StationeryItem AS si ON soi.itemCode = si.itemCode
12 WHERE soi.orderId = NEW.orderId
13 )
14 WHERE orderId = NEW.orderId; -- subquery in the main query --
15 END;
16 //
17 DELIMITER ;
```

Below the code, there are buttons: 'Clear', 'Format', 'Get auto-saved query', 'Bind parameters', 'Bookmark this SQL query:', 'Delimiter :', 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks'. A 'Go' button is at the bottom.



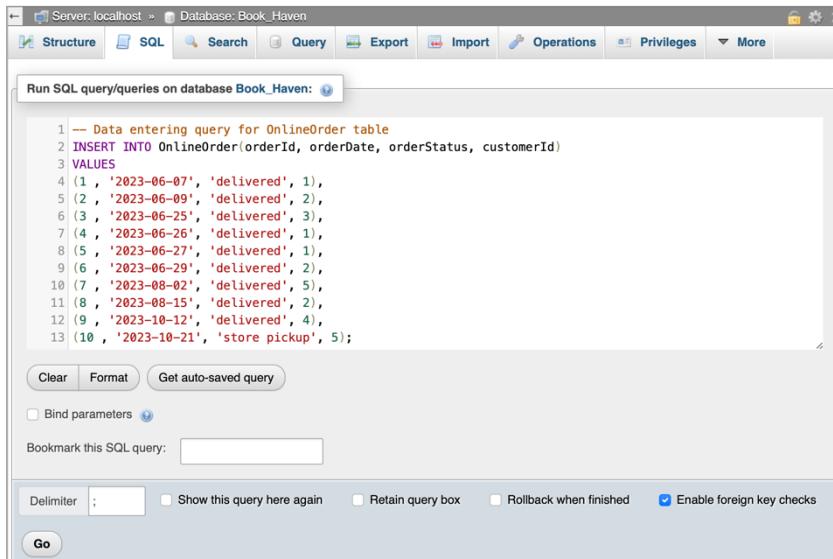
The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. A query window titled 'Run SQL query/queries on database Book_Haven:' displays the results of the trigger creation query. It shows a green message: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0100 seconds.)' and the full trigger definition:

```
CREATE TRIGGER update_online_order_total_price_stationery AFTER INSERT ON StationeryOrderItem -- trigger
execute after insert data on StationeryOrderItem table -- FOR EACH ROW BEGIN UPDATE OnlineOrder -- updating the
OnlineOrder table -- SET totalPrice = totalPrice + ( SELECT SUM(soi.quantity * si.price) FROM
StationeryOrderItem AS soi JOIN StationeryItem AS si ON soi.itemCode = si.itemCode WHERE soi.orderId =
NEW.orderId ) WHERE orderId = NEW.orderId; -- subquery in the main query -- END;
```

Below the code, there are buttons: 'Edit inline', 'Edit', and 'Create PHP code'.

Figure 72 Trigger that update the totalPrice in OnlineOrder table(stationery)

OnlineOrder Table



The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, More.
- Query Editor:** Run SQL query/queries on database Book_Haven. The query is:

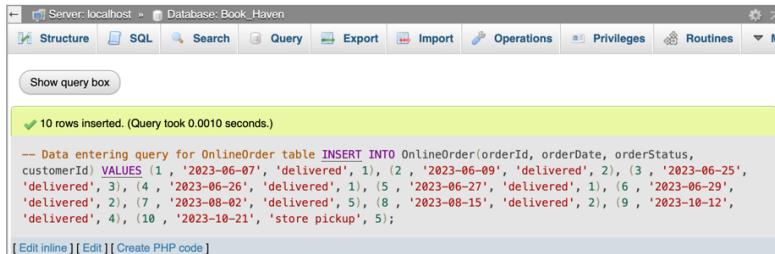
```

1 -- Data entering query for OnlineOrder table
2 INSERT INTO OnlineOrder(orderId, orderDate, orderStatus, customerId)
3 VALUES
4 (1 , '2023-06-07', 'delivered', 1),
5 (2 , '2023-06-09', 'delivered', 2),
6 (3 , '2023-06-25', 'delivered', 3),
7 (4 , '2023-06-26', 'delivered', 1),
8 (5 , '2023-06-27', 'delivered', 1),
9 (6 , '2023-06-29', 'delivered', 2),
10 (7 , '2023-08-02', 'delivered', 5),
11 (8 , '2023-08-15', 'delivered', 2),
12 (9 , '2023-10-12', 'delivered', 4),
13 (10 , '2023-10-21', 'store pickup', 5);

```

- Buttons:** Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query:, Delimiter :, Show this query here again, Retain query box, Rollback when finished, Enable foreign key checks.
- Buttons at the bottom:** Go, Save, Undo, Redo.

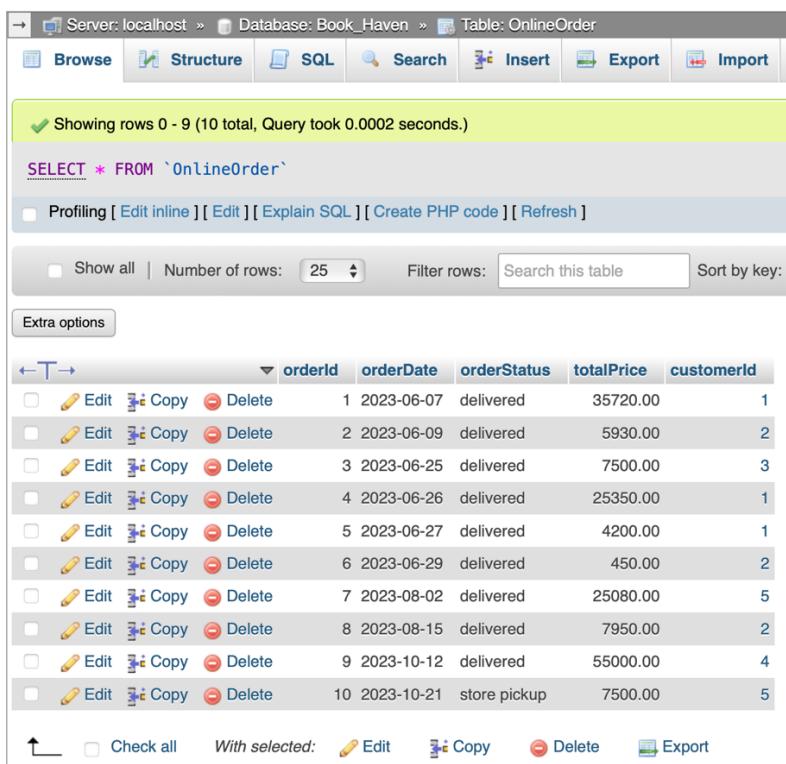
Figure 73 Data entering query for OnlineOrder table.



The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, M.
- Message Bar:** ✓ 10 rows inserted. (Query took 0.0010 seconds.)
- Text Area:** -- Data entering query for OnlineOrder table `INSERT INTO OnlineOrder(orderId, orderDate, orderStatus, customerId) VALUES (1 , '2023-06-07', 'delivered', 1), (2 , '2023-06-09', 'delivered', 2), (3 , '2023-06-25', 'delivered', 3), (4 , '2023-06-26', 'delivered', 1), (5 , '2023-06-27', 'delivered', 1), (6 , '2023-06-29', 'delivered', 2), (7 , '2023-08-02', 'delivered', 5), (8 , '2023-08-15', 'delivered', 2), (9 , '2023-10-12', 'delivered', 4), (10 , '2023-10-21', 'store pickup', 5);`
- Buttons:** Edit inline, Edit, Create PHP code.

Figure 74 Data entered successfully to OnlineOrder table.



The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Server: localhost, Database: Book_Haven, Table: OnlineOrder. Buttons: Browse, Structure, SQL, Search, Insert, Export, Import.
- Message Bar:** ✓ Showing rows 0 - 9 (10 total, Query took 0.0002 seconds.)
- Text Area:** `SELECT * FROM `OnlineOrder``
- Buttons:** Profiling, Edit inline, Edit, Explain SQL, Create PHP code, Refresh.
- Search Bar:** Show all, Number of rows: 25, Filter rows: Search this table, Sort by key:.
- Extra Options:** Extra options button.
- Data Grid:** A table with columns: orderId, orderDate, orderStatus, totalPrice, customerId. Rows (1-10):

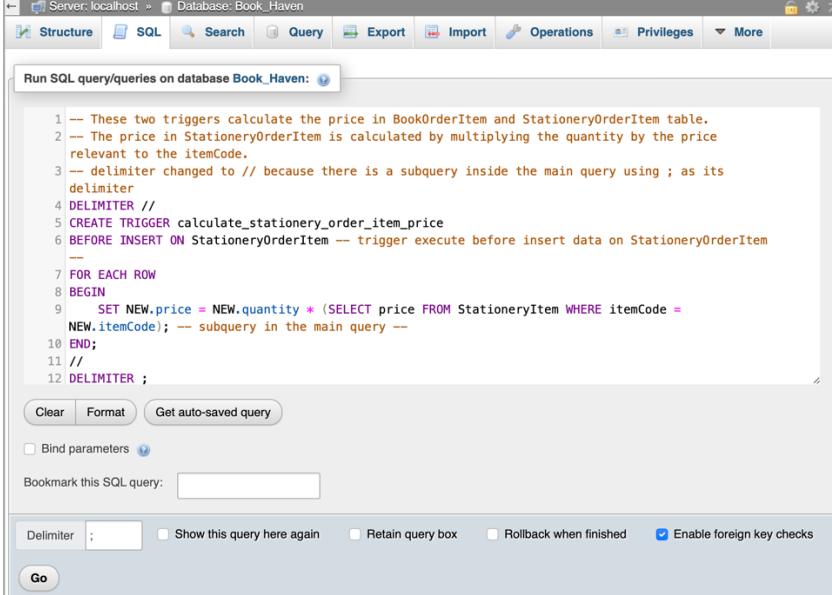
	orderId	orderDate	orderStatus	totalPrice	customerId
<input type="checkbox"/>	1	2023-06-07	delivered	35720.00	1
<input type="checkbox"/>	2	2023-06-09	delivered	5930.00	2
<input type="checkbox"/>	3	2023-06-25	delivered	7500.00	3
<input type="checkbox"/>	4	2023-06-26	delivered	25350.00	1
<input type="checkbox"/>	5	2023-06-27	delivered	4200.00	1
<input type="checkbox"/>	6	2023-06-29	delivered	450.00	2
<input type="checkbox"/>	7	2023-08-02	delivered	25080.00	5
<input type="checkbox"/>	8	2023-08-15	delivered	7950.00	2
<input type="checkbox"/>	9	2023-10-12	delivered	55000.00	4
<input type="checkbox"/>	10	2023-10-21	store pickup	7500.00	5

- Bottom Buttons:** Check all, With selected: Edit, Copy, Delete, Export.

Figure 75 Data entered OnlineOrder table.

Trigger 2

These two triggers calculate the price in BookOrderItem and StationeryOrderItem table.



The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. In the 'SQL' tab, a query is being run to create a trigger named 'calculate_stationery_order_item_price'. The trigger is defined to execute before an insert operation on the 'StationeryOrderItem' table. It uses a subquery to calculate the price by multiplying the quantity by the price from the 'StationeryItem' table where the item code matches the new item code. The code includes comments explaining the logic and delimiter changes.

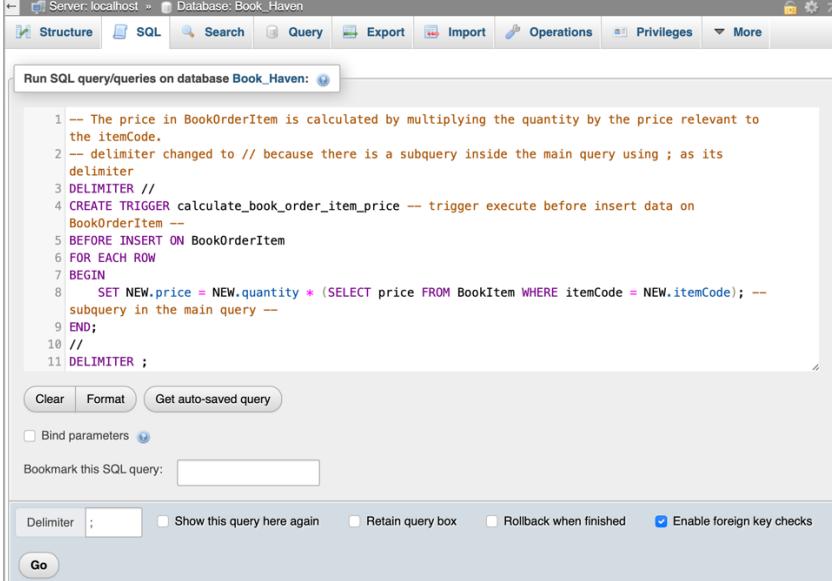
```
1 -- These two triggers calculate the price in BookOrderItem and StationeryOrderItem table.
2 -- The price in StationeryOrderItem is calculated by multiplying the quantity by the price
   relevant to the itemCode.
3 -- delimiter changed to // because there is a subquery inside the main query using ; as its
   delimiter
4 DELIMITER //
5 CREATE TRIGGER calculate_stationery_order_item_price
6 BEFORE INSERT ON StationeryOrderItem -- trigger execute before insert data on StationeryOrderItem
-- 
7 FOR EACH ROW
8 BEGIN
9     SET NEW.price = NEW.quantity * (SELECT price FROM StationeryItem WHERE itemCode =
10 NEW.itemCode); -- subquery in the main query --
11 END;
12 //
13 DELIMITER ;
```



The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. In the 'SQL' tab, a query is being run to create a trigger named 'calculate_stationery_order_item_price'. The trigger is defined to execute before an insert operation on the 'StationeryOrderItem' table. It uses a subquery to calculate the price by multiplying the quantity by the price from the 'StationeryItem' table where the item code matches the new item code. The code includes comments explaining the logic and delimiter changes.

```
CREATE TRIGGER calculate_stationery_order_item_price BEFORE INSERT ON StationeryOrderItem -- trigger execute before insert data on StationeryOrderItem -- FOR EACH ROW BEGIN SET NEW.price = NEW.quantity * (SELECT price FROM StationeryItem WHERE itemCode = NEW.itemCode); -- subquery in the main query -- END;
```

Figure 76 Trigger that calculate the price in StationeryOrderItem table.



The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. In the 'SQL' tab, a query is being run to create a trigger named 'calculate_book_order_item_price'. The trigger is defined to execute before an insert operation on the 'BookOrderItem' table. It uses a subquery to calculate the price by multiplying the quantity by the price from the 'BookItem' table where the item code matches the new item code. The code includes comments explaining the logic and delimiter changes.

```
1 -- The price in BookOrderItem is calculated by multiplying the quantity by the price relevant to
   the itemCode.
2 -- delimiter changed to // because there is a subquery inside the main query using ; as its
   delimiter
3 DELIMITER //
4 CREATE TRIGGER calculate_book_order_item_price -- trigger execute before insert data on
   BookOrderItem --
5 BEFORE INSERT ON BookOrderItem
6 FOR EACH ROW
7 BEGIN
8     SET NEW.price = NEW.quantity * (SELECT price FROM BookItem WHERE itemCode = NEW.itemCode); -- subquery in the main query --
9 END;
10 //
11 DELIMITER ;
```

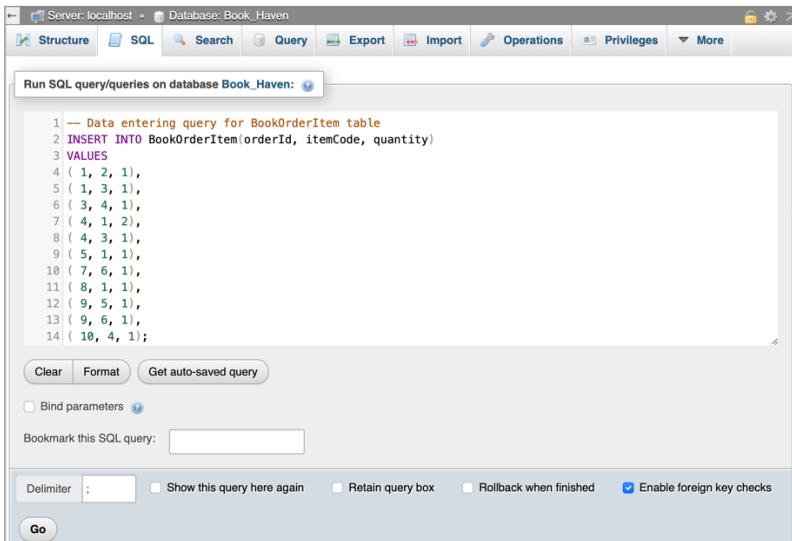


The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. In the 'SQL' tab, a query is being run to create a trigger named 'calculate_book_order_item_price'. The trigger is defined to execute before an insert operation on the 'BookOrderItem' table. It uses a subquery to calculate the price by multiplying the quantity by the price from the 'BookItem' table where the item code matches the new item code. The code includes comments explaining the logic and delimiter changes.

```
CREATE TRIGGER calculate_book_order_item_price -- trigger execute before insert data on BookOrderItem -- BEFORE
   INSERT ON BookOrderItem FOR EACH ROW BEGIN SET NEW.price = NEW.quantity * (SELECT price FROM BookItem WHERE
   itemCode = NEW.itemCode); -- subquery in the main query -- END;
```

Figure 77 Trigger that calculate the price in BookOrderItem table.

BookOrderItem Table



The screenshot shows the MySQL Workbench interface with the SQL tab selected. A query window displays the following SQL code:

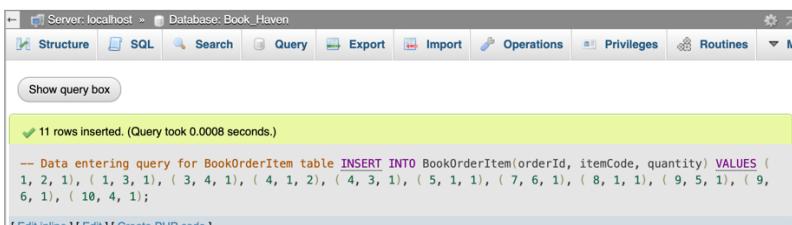
```

1 -- Data entering query for BookOrderItem table
2 INSERT INTO BookOrderItem(orderId, itemCode, quantity)
3 VALUES
4 ( 1, 2, 1),
5 ( 1, 3, 1),
6 ( 3, 4, 1),
7 ( 4, 1, 2),
8 ( 4, 3, 1),
9 ( 5, 1, 1),
10 ( 7, 6, 1),
11 ( 8, 1, 1),
12 ( 9, 5, 1),
13 ( 9, 6, 1),
14 ( 10, 4, 1);

```

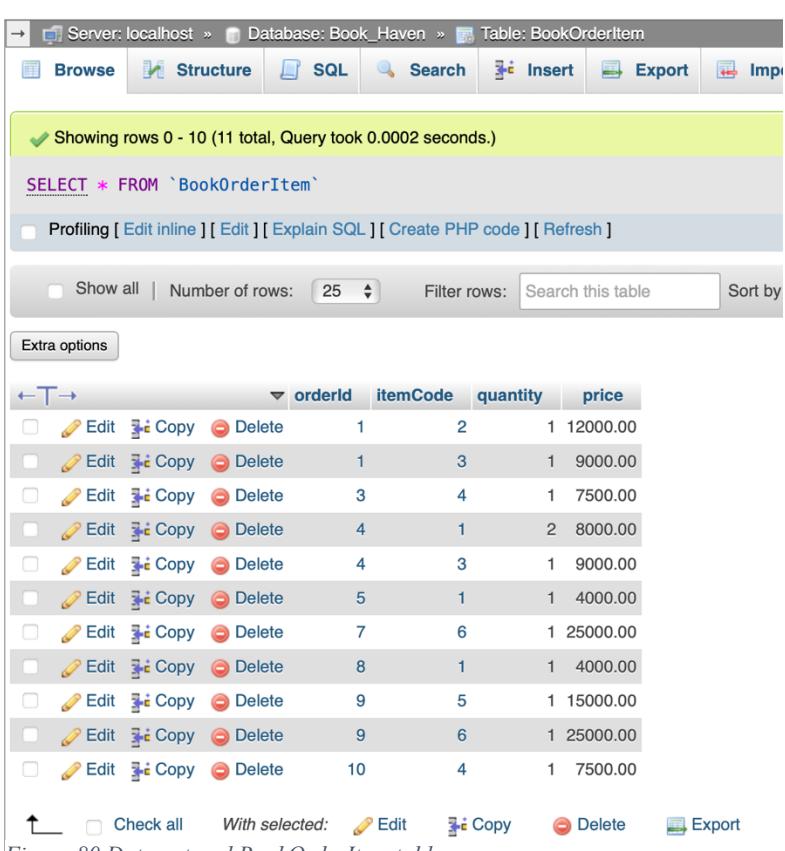
Below the query window, there are several buttons: Clear, Format, Get auto-saved query, Bind parameters, and a bookmark field. At the bottom, there are options for Delimiter, Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A 'Go' button is also present.

Figure 78 Data entering query for BookOrderItem table.



The screenshot shows the MySQL Workbench interface with the SQL tab selected. The query window now displays a success message: "✓ 11 rows inserted. (Query took 0.0008 seconds.)". Below the message, the same INSERT query is shown. At the bottom of the window, there are links for Edit inline, Edit, Create PHP code, and Refresh.

Figure 79 Data entered successfully to BookOrderItem table.



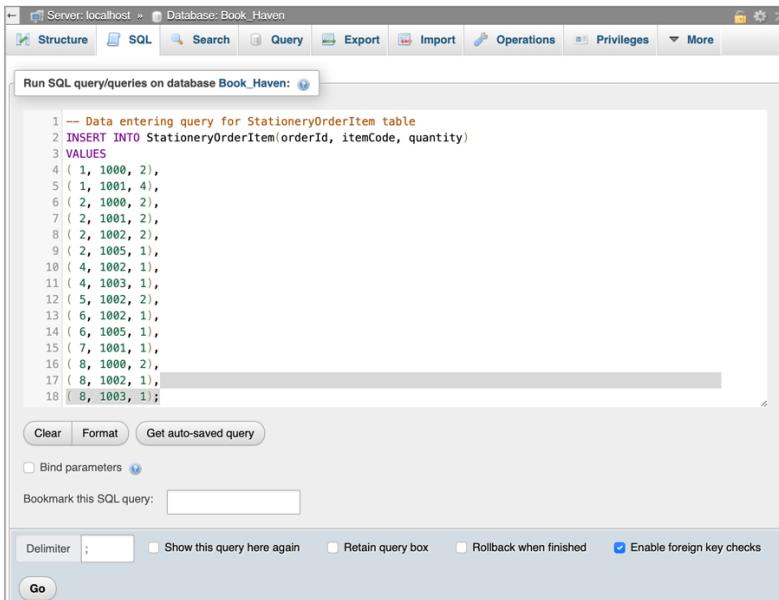
The screenshot shows the MySQL Workbench interface with the Table tab selected for the BookOrderItem table. The table has columns: orderId, itemCode, quantity, and price. The data grid contains the following rows:

	orderId	itemCode	quantity	price
<input type="checkbox"/>	1	2	1	12000.00
<input type="checkbox"/>	1	3	1	9000.00
<input type="checkbox"/>	3	4	1	7500.00
<input type="checkbox"/>	4	1	2	8000.00
<input type="checkbox"/>	4	3	1	9000.00
<input type="checkbox"/>	5	1	1	4000.00
<input type="checkbox"/>	7	6	1	25000.00
<input type="checkbox"/>	8	1	1	4000.00
<input type="checkbox"/>	9	5	1	15000.00
<input type="checkbox"/>	9	6	1	25000.00
<input type="checkbox"/>	10	4	1	7500.00

At the bottom of the table area, there are buttons for Check all, With selected, Edit, Copy, Delete, Export, and Import.

Figure 80 Data entered BookOrderItem table.

StationeryOrderItem Table



The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Structure, SQL, Search, Export, Import, Operations, Privileges, More.
- Query Editor:** Run SQL query/queries on database Book_Haven. The query is:

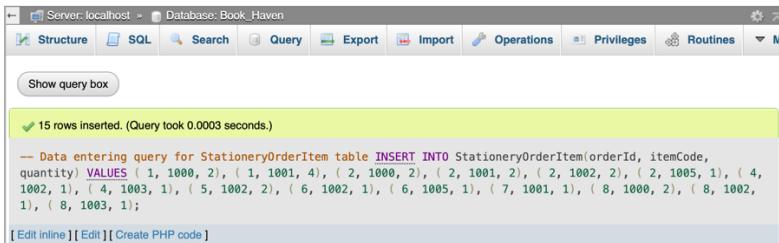
```

1 -- Data entering query for StationeryOrderItem table
2 INSERT INTO StationeryOrderItem(orderId, itemCode, quantity)
3 VALUES
4 ( 1, 1000, 2),
5 ( 1, 1001, 4),
6 ( 2, 1000, 2),
7 ( 2, 1001, 2),
8 ( 2, 1002, 2),
9 ( 2, 1005, 1),
10 ( 4, 1002, 1),
11 ( 4, 1003, 1),
12 ( 5, 1002, 2),
13 ( 6, 1002, 1),
14 ( 6, 1005, 1),
15 ( 7, 1001, 1),
16 ( 8, 1000, 2),
17 ( 8, 1002, 1),
18 ( 8, 1003, 1);

```

- Buttons:** Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query: [input field], Delimiter [dropdown], Show this query here again, Retain query box, Rollback when finished, Enable foreign key checks.
- Go Button:** A large blue button labeled "Go".

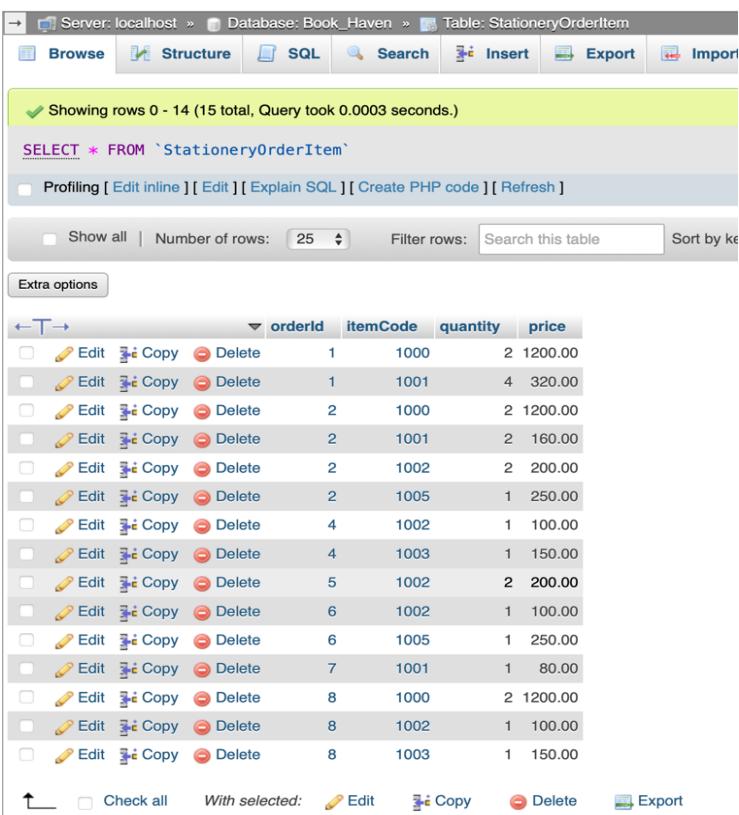
Figure 81 Data entering query for StationeryOrderItem table.



The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, More.
- Message Bar:** 15 rows inserted. (Query took 0.0003 seconds.)
- Query Editor:** The same SQL insert query as in Figure 81.
- Buttons:** Edit inline, Edit, Create PHP code.

Figure 82 Data entered successfully to StationeryOrderItem table.



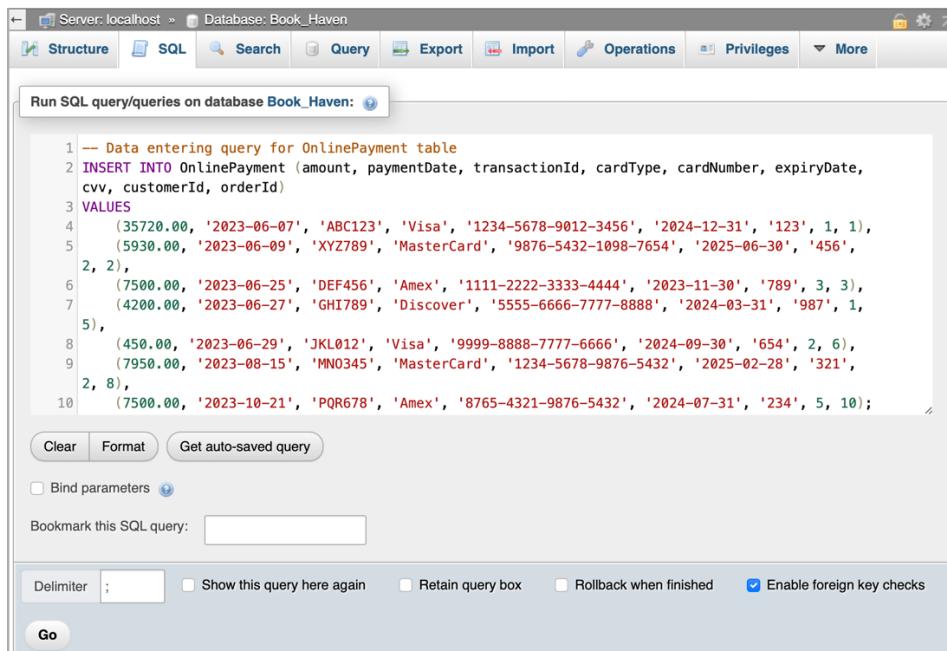
The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import.
- Message Bar:** Showing rows 0 - 14 (15 total, Query took 0.0003 seconds.)
- SQL Editor:** SELECT * FROM `StationeryOrderItem`.
- Table View:** A grid showing the data entered into the StationeryOrderItem table. The columns are orderId, itemCode, quantity, and price.

	orderId	itemCode	quantity	price
<input type="checkbox"/>	1	1000	2	1200.00
<input type="checkbox"/>	1	1001	4	320.00
<input type="checkbox"/>	2	1000	2	1200.00
<input type="checkbox"/>	2	1001	2	160.00
<input type="checkbox"/>	2	1002	2	200.00
<input type="checkbox"/>	2	1005	1	250.00
<input type="checkbox"/>	4	1002	1	100.00
<input type="checkbox"/>	4	1003	1	150.00
<input type="checkbox"/>	5	1002	2	200.00
<input type="checkbox"/>	6	1002	1	100.00
<input type="checkbox"/>	6	1005	1	250.00
<input type="checkbox"/>	7	1001	1	80.00
<input type="checkbox"/>	8	1000	2	1200.00
<input type="checkbox"/>	8	1002	1	100.00
<input type="checkbox"/>	8	1003	1	150.00

Figure 83 Data entered StationeryOrderItem table.

OnlinePayment Table



The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. The SQL tab is active, displaying the following SQL code:

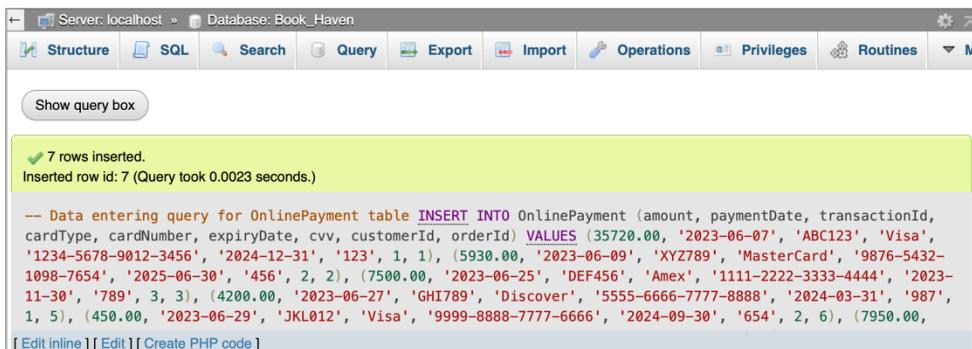
```

1 -- Data entering query for OnlinePayment table
2 INSERT INTO OnlinePayment (amount, paymentDate, transactionId, cardType, cardNumber, expiryDate,
3 cvv, customerId, orderId)
4 VALUES
5 (35720.00, '2023-06-07', 'ABC123', 'Visa', '1234-5678-9012-3456', '2024-12-31', '123', 1, 1),
6 (5930.00, '2023-06-09', 'XYZ789', 'MasterCard', '9876-5432-1098-7654', '2025-06-30', '456',
7 2),
8 (7500.00, '2023-06-25', 'DEF456', 'Amex', '1111-2222-3333-4444', '2023-11-30', '789', 3, 3),
9 (4200.00, '2023-06-27', 'GHI789', 'Discover', '5555-6666-7777-8888', '2024-03-31', '987', 1,
10 5),
10 (450.00, '2023-06-29', 'JKL012', 'Visa', '9999-8888-7777-6666', '2024-09-30', '654', 2, 6),
11 (7950.00, '2023-08-15', 'MNO345', 'MasterCard', '1234-5678-9876-5432', '2025-02-28', '321',
2, 8),
12 (7500.00, '2023-10-21', 'PQR678', 'Amex', '8765-4321-9876-5432', '2024-07-31', '234', 5, 10);

```

Below the code, there are several buttons: Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query, Delimiter (set to ;), Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A 'Go' button is at the bottom.

Figure 84 Data entering query for OnlinePayment table.

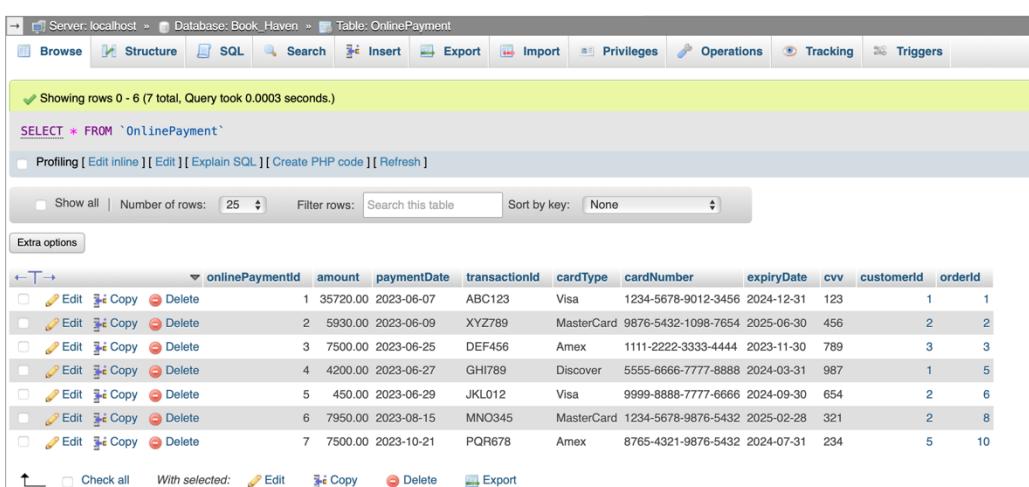


The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven'. The SQL tab is active, displaying the following message in a green box:

✓ 7 rows inserted.
Inserted row id: 7 (Query took 0.0023 seconds.)

Below the message, the same SQL code as in Figure 84 is shown. At the bottom of the SQL tab, there are buttons for [Edit inline], [Edit], and [Create PHP code].

Figure 85 Data entered successfully to OnlinePayment table.



The screenshot shows the MySQL Workbench interface with the database set to 'Book_Haven' and the table set to 'OnlinePayment'. The Structure tab is active, showing the table structure. Below it, the SQL tab shows the following output:

✓ Showing rows 0 - 6 (7 total). Query took 0.0003 seconds.

SELECT * FROM `OnlinePayment`

Below the SQL output, there are buttons for Profiling, Edit inline, Explain SQL, Create PHP code, and Refresh. There are also filters for Show all, Number of rows (set to 25), Filter rows, Search this table, Sort by key (set to None), and Extra options.

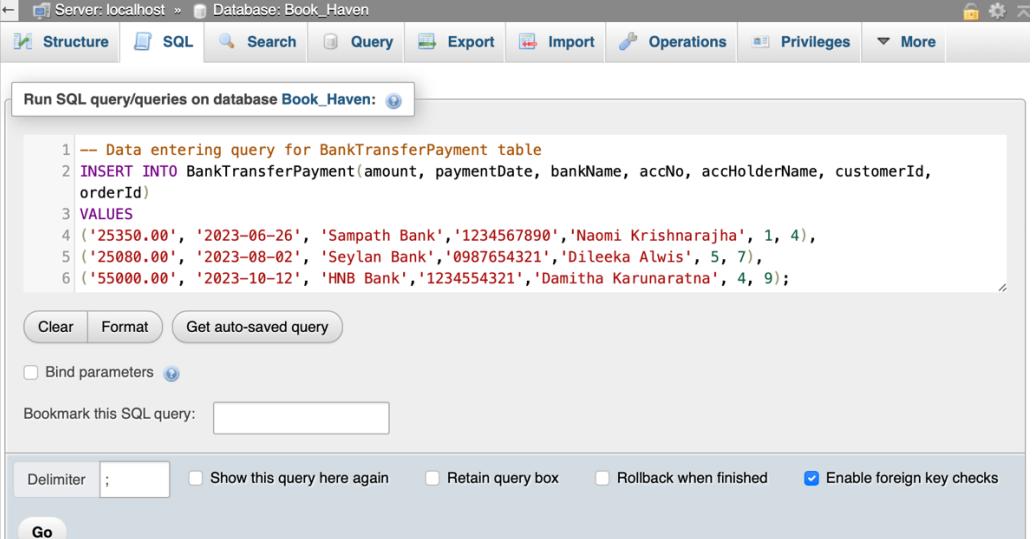
The main area displays the data grid for the OnlinePayment table:

	onlinePaymentId	amount	paymentDate	transactionId	cardType	cardNumber	expiryDate	cvv	customerId	orderId
<input type="checkbox"/>	1	35720.00	2023-06-07	ABC123	Visa	1234-5678-9012-3456	2024-12-31	123	1	1
<input type="checkbox"/>	2	5930.00	2023-06-09	XYZ789	MasterCard	9876-5432-1098-7654	2025-06-30	456	2	2
<input type="checkbox"/>	3	7500.00	2023-06-25	DEF456	Amex	1111-2222-3333-4444	2023-11-30	789	3	3
<input type="checkbox"/>	4	4200.00	2023-06-27	GHI789	Discover	5555-6666-7777-8888	2024-03-31	987	1	5
<input type="checkbox"/>	5	450.00	2023-06-29	JKL012	Visa	9999-8888-7777-6666	2024-09-30	654	2	6
<input type="checkbox"/>	6	7950.00	2023-08-15	MNO345	MasterCard	1234-5678-9876-5432	2025-02-28	321	2	8
<input type="checkbox"/>	7	7500.00	2023-10-21	PQR678	Amex	8765-4321-9876-5432	2024-07-31	234	5	10

At the bottom of the grid, there are buttons for Check all, With selected, Edit, Copy, Delete, and Export.

Figure 86 Data entered OnlinePayment table.

BankTransferPayment Table



```

1 -- Data entering query for BankTransferPayment table
2 INSERT INTO BankTransferPayment(amount, paymentDate, bankName, accNo, accHolderName, customerId,
3 orderId)
3 VALUES
4 ('25350.00', '2023-06-26', 'Sampath Bank','1234567890','Naomi Krishnarajha', 1, 4),
5 ('25080.00', '2023-08-02', 'Seylan Bank','0987654321','Dileeka Alwis', 5, 7),
6 ('55000.00', '2023-10-12', 'HNB Bank','1234554321','Damitha Karunaratna', 4, 9);
    
```

Clear Format Get auto-saved query

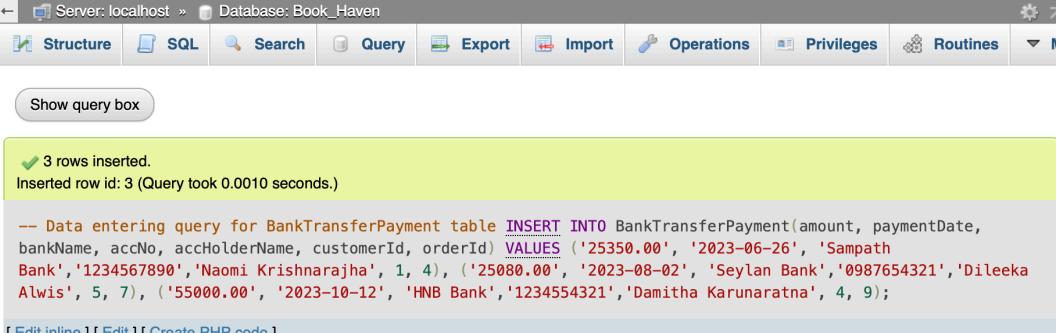
Bind parameters

Bookmark this SQL query:

Delimiter : Show this query here again Retain query box Rollback when finished Enable foreign key checks

Go

Figure 87 Data entering query for BankTransferPayment table.

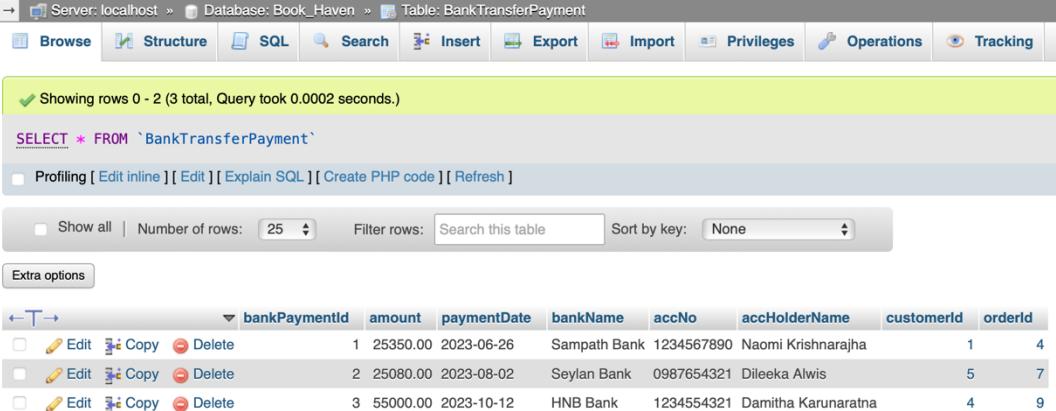


3 rows inserted.
Inserted row id: 3 (Query took 0.0010 seconds.)

```
-- Data entering query for BankTransferPayment table INSERT INTO BankTransferPayment(amount, paymentDate, bankName, accNo, accHolderName, customerId, orderId) VALUES ('25350.00', '2023-06-26', 'Sampath Bank','1234567890','Naomi Krishnarajha', 1, 4), ('25080.00', '2023-08-02', 'Seylan Bank','0987654321','Dileeka Alwis', 5, 7), ('55000.00', '2023-10-12', 'HNB Bank','1234554321','Damitha Karunaratna', 4, 9);
```

[Edit inline] [Edit] [Create PHP code]

Figure 88 Data entered successfully to BankTransferPayment table.



Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)

SELECT * FROM `BankTransferPayment`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

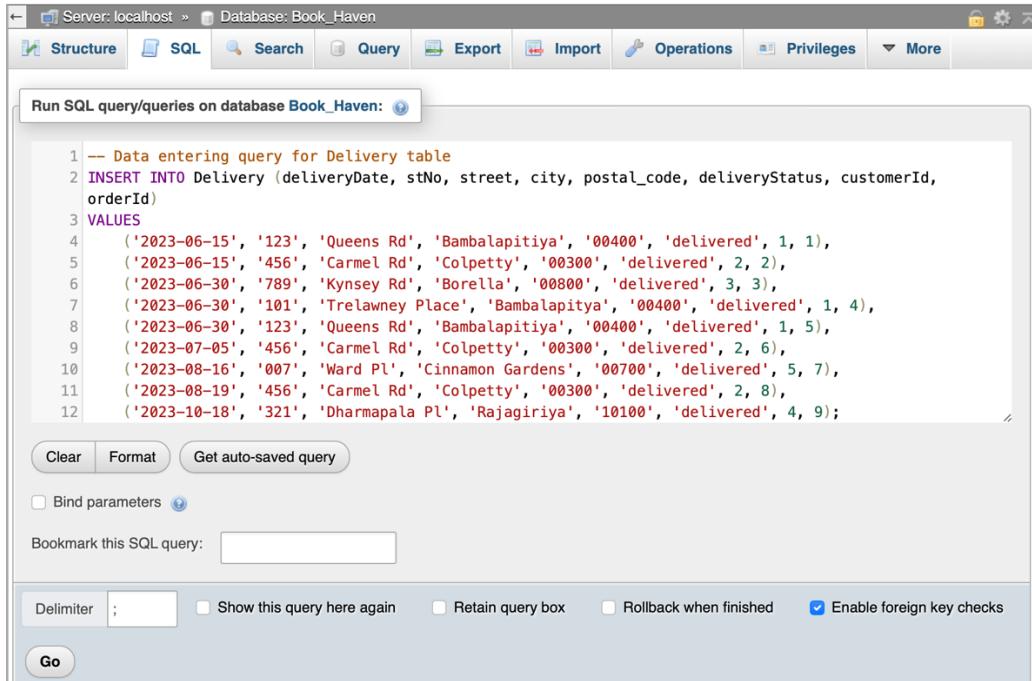
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	bankPaymentId	amount	paymentDate	bankName	accNo	accHolderName	customerId	orderId
<input type="checkbox"/>	1	25350.00	2023-06-26	Sampath Bank	1234567890	Naomi Krishnarajha	1	4
<input type="checkbox"/>	2	25080.00	2023-08-02	Seylan Bank	0987654321	Dileeka Alwis	5	7
<input type="checkbox"/>	3	55000.00	2023-10-12	HNB Bank	1234554321	Damitha Karunaratna	4	9

Figure 89 Data entered BankTransferPayment table.

Delivery Table



The screenshot shows the MySQL Workbench interface with the SQL tab selected. A query window titled "Run SQL query/queries on database Book_Haven:" contains the following SQL code:

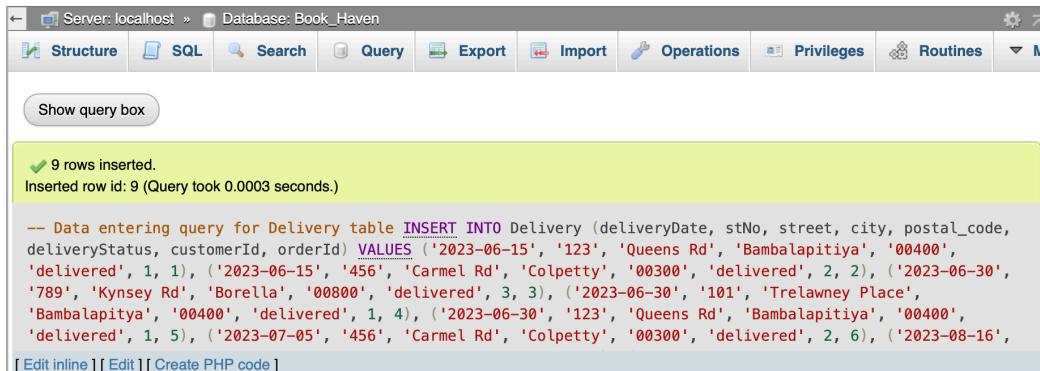
```

1 -- Data entering query for Delivery table
2 INSERT INTO Delivery (deliveryDate, stNo, street, city, postal_code, deliveryStatus, customerId, orderId)
3 VALUES
4 ('2023-06-15', '123', 'Queens Rd', 'Bambalapitiya', '00400', 'delivered', 1, 1),
5 ('2023-06-15', '456', 'Carmel Rd', 'Colpetty', '00300', 'delivered', 2, 2),
6 ('2023-06-30', '789', 'Kynsey Rd', 'Borella', '00800', 'delivered', 3, 3),
7 ('2023-06-30', '101', 'Trelawney Place', 'Bambalapitya', '00400', 'delivered', 1, 4),
8 ('2023-06-30', '123', 'Queens Rd', 'Bambalapitiya', '00400', 'delivered', 1, 5),
9 ('2023-07-05', '456', 'Carmel Rd', 'Colpetty', '00300', 'delivered', 2, 6),
10 ('2023-08-16', '007', 'Ward Pl', 'Cinnamon Gardens', '00700', 'delivered', 5, 7),
11 ('2023-08-19', '456', 'Carmel Rd', 'Colpetty', '00300', 'delivered', 2, 8),
12 ('2023-10-18', '321', 'Dharmapala Pl', 'Rajagiriya', '10100', 'delivered', 4, 9);

```

Below the code, there are several buttons: Clear, Format, Get auto-saved query, Bind parameters, Bookmark this SQL query, Delimiter (set to ;), Show this query here again, Retain query box, Rollback when finished, and Enable foreign key checks. A "Go" button is at the bottom.

Figure 90 Data entering query for Delivery table.

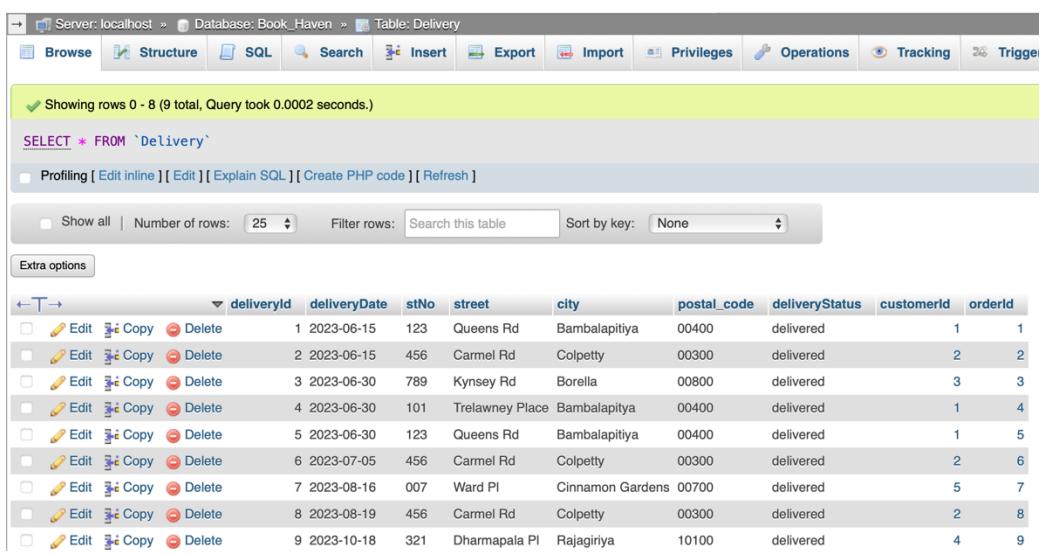


The screenshot shows the MySQL Workbench interface with the SQL tab selected. A message box displays the result of the query execution:

- 9 rows inserted.
- Inserted row id: 9 (Query took 0.0003 seconds.)

The same SQL code from Figure 90 is shown again, followed by the message. Below the message are three buttons: [Edit inline], [Edit], and [Create PHP code].

Figure 91 Data entered successfully to Delivery table.



The screenshot shows the MySQL Workbench interface with the Delivery table selected. The table has the following columns: deliveryId, deliveryDate, stNo, street, city, postal_code, deliveryStatus, customerId, and orderId. The data entered is as follows:

deliveryId	deliveryDate	stNo	street	city	postal_code	deliveryStatus	customerId	orderId
1	2023-06-15	123	Queens Rd	Bambalapitiya	00400	delivered	1	1
2	2023-06-15	456	Carmel Rd	Colpetty	00300	delivered	2	2
3	2023-06-30	789	Kynsey Rd	Borella	00800	delivered	3	3
4	2023-06-30	101	Trelawney Place	Bambalapitya	00400	delivered	1	4
5	2023-06-30	123	Queens Rd	Bambalapitiya	00400	delivered	1	5
6	2023-07-05	456	Carmel Rd	Colpetty	00300	delivered	2	6
7	2023-08-16	007	Ward Pl	Cinnamon Gardens	00700	delivered	5	7
8	2023-08-19	456	Carmel Rd	Colpetty	00300	delivered	2	8
9	2023-10-18	321	Dharmapala Pl	Rajagiriya	10100	delivered	4	9

Figure 92 Data entered Delivery table.

Database Diagram

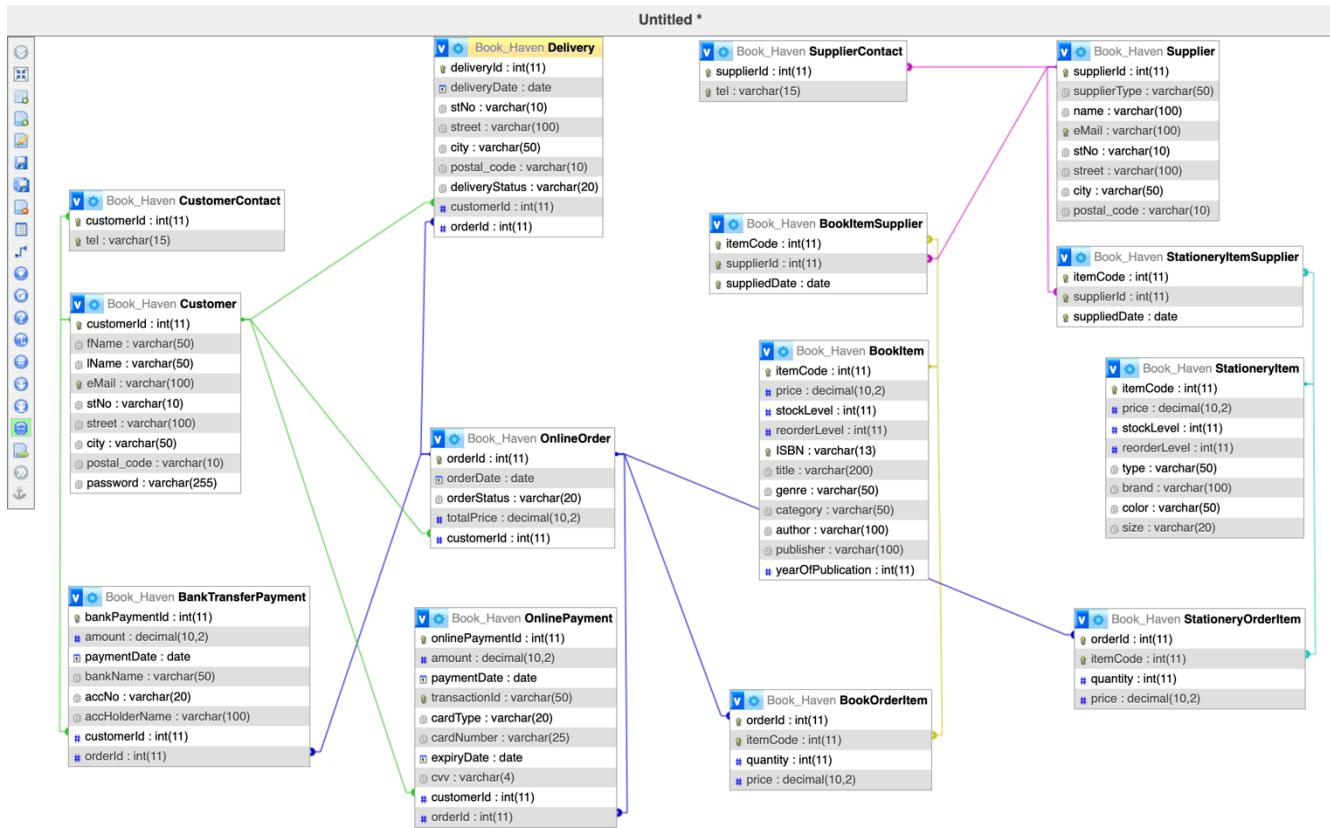
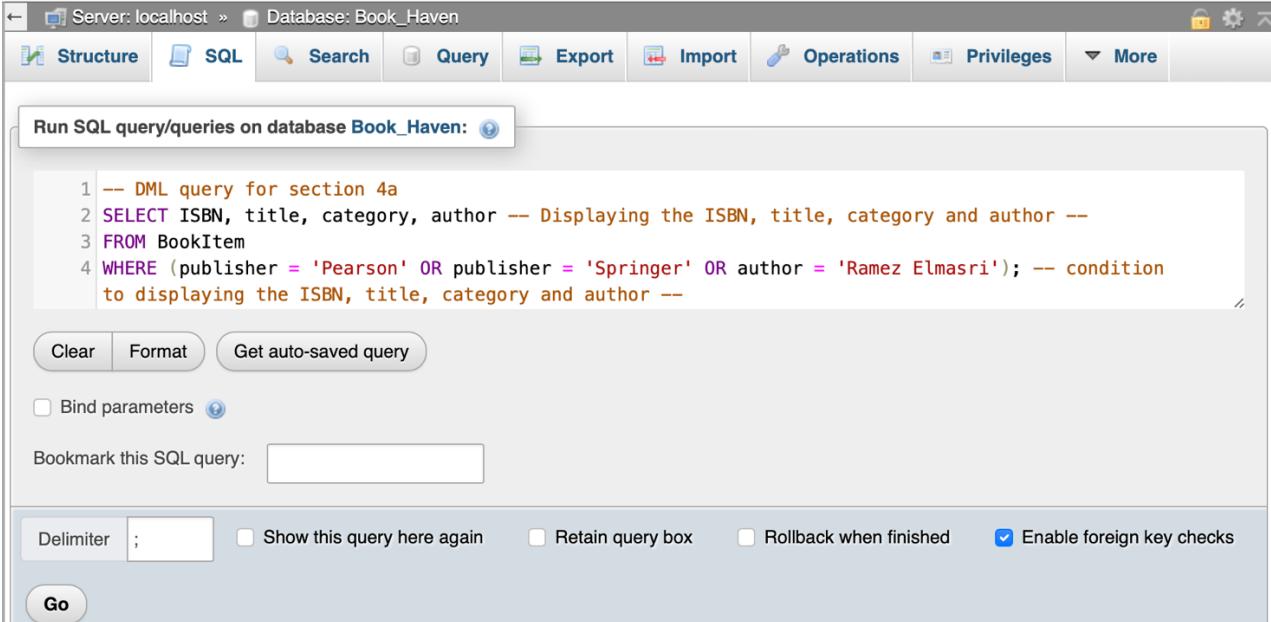


Figure 93 Database diagram

Section 4

Data manipulation with SQL

a)

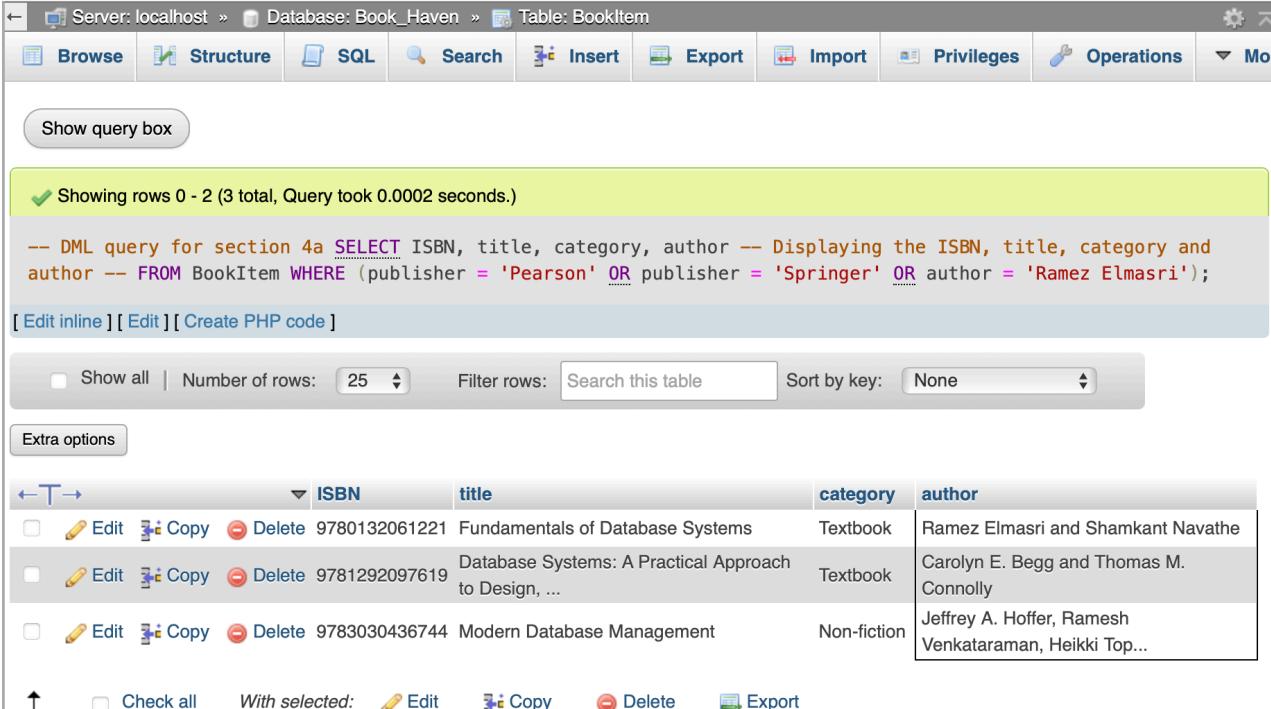


The screenshot shows the MySQL Workbench interface. The top menu bar has 'Server: localhost' and 'Database: Book_Haven'. The tabs at the top are 'Structure', 'SQL', 'Search', 'Query', 'Export', 'Import', 'Operations', 'Privileges', and 'More'. The 'SQL' tab is selected. Below it is a text area titled 'Run SQL query/queries on database Book_Haven:'. The query is:

```
1 -- DML query for section 4a
2 SELECT ISBN, title, category, author -- Displaying the ISBN, title, category and author --
3 FROM BookItem
4 WHERE (publisher = 'Pearson' OR publisher = 'Springer' OR author = 'Ramez Elmasri'); -- condition
      to displaying the ISBN, title, category and author --
```

Below the query are buttons for 'Clear', 'Format', 'Get auto-saved query', and a checkbox for 'Bind parameters'. There is also a field to 'Bookmark this SQL query:' with a placeholder box. At the bottom are options for 'Delimiter' (set to ;), checkboxes for 'Show this query here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks' (which is checked), and a 'Go' button.

Figure 94 DML query for question a)



The screenshot shows the MySQL Workbench interface with 'Table: BookItem' selected. The top menu bar has 'Server: localhost', 'Database: Book_Haven', and 'Table: BookItem'. The tabs at the top are 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', and 'More'. The 'Insert' tab is selected. Below it is a message: 'Showing rows 0 - 2 (3 total, Query took 0.00002 seconds.)'. The query is displayed again: -- DML query for section 4a SELECT ISBN, title, category, author -- Displaying the ISBN, title, category and author -- FROM BookItem WHERE (publisher = 'Pearson' OR publisher = 'Springer' OR author = 'Ramez Elmasri');

Below the message are buttons for '[Edit inline]', '[Edit]', and '[Create PHP code]'. There are filters for 'Show all' (checkbox), 'Number of rows: 25', 'Filter rows: Search this table', 'Sort by key: None', and 'Extra options' (button). The main area shows a table with columns: ISBN, title, category, and author. The data is:

	ISBN	title	category	author
<input type="checkbox"/>	Edit Copy Delete 9780132061221	Fundamentals of Database Systems	Textbook	Ramez Elmasri and Shamkant Navathe
<input type="checkbox"/>	Edit Copy Delete 9781292097619	Database Systems: A Practical Approach to Design, ...	Textbook	Carolyn E. Begg and Thomas M. Connolly
<input type="checkbox"/>	Edit Copy Delete 9783030436744	Modern Database Management	Non-fiction	Jeffrey A. Hoffer, Ramesh Venkataraman, Heikki Top...

At the bottom are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'.

Figure 95 results for question a)

b)

The screenshot shows the MySQL Workbench interface. The title bar indicates "Server: localhost » Database: Book_Haven". The tabs at the top include Structure, SQL, Search, Query, Export, Import, Operations, Privileges, and More. The SQL tab is selected. The main area contains the following SQL code:

```
1 -- DML query for section 4b
2 SELECT CONCAT(fName, ' ', lName) AS customer_name, eMail AS customer_email,
3 tel AS customer_contact,
4 COUNT(DISTINCT o.orderId) AS number_of_orders
5 FROM Customer AS c
6 JOIN
7 CustomerContact AS cc ON c.customerId = cc.customerId
8 JOIN
9 OnlineOrder AS o ON c.customerId = o.customerId
10 WHERE o.orderDate >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH) -- subtracting 6 months from current
date --
11 GROUP BY c.customerId
12 HAVING COUNT(DISTINCT o.orderId) > 2;-- condition to displaying customer only placed more than 2
order within 6 months of time --
```

Below the code are several buttons: Clear, Format, Get auto-saved query, Bind parameters, and a bookmark input field. At the bottom are settings for Delimiter (set to ;), Show this query here again (unchecked), Retain query box (unchecked), Rollback when finished (unchecked), and Enable foreign key checks (checked). A "Go" button is also present.

Figure 96 DML query for question b)

The screenshot shows the MySQL Workbench interface with the title bar "Server: localhost » Database: Book_Haven » Table: Customer". The tabs at the top include Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, and More. The SQL tab is selected. The main area displays the results of the previous query:

```
-- DML query for section 4b
SELECT CONCAT(fName, ' ', lName) AS customer_name, eMail AS customer_email, tel AS
customer_contact, COUNT(DISTINCT o.orderId) AS number_of_orders
FROM Customer AS c
JOIN CustomerContact AS cc
ON c.customerId = cc.customerId
JOIN OnlineOrder AS o
ON c.customerId = o.customerId
WHERE o.orderDate >=
DATE_SUB(CURDATE(), INTERVAL 6 MONTH) -- subtracting 6 months from current date --
GROUP BY c.customerId
HAVING COUNT(DISTINCT o.orderId) > 2;
```

A message bar at the top states "Showing rows 0 - 1 (2 total, Query took 0.0026 seconds.)". Below the message bar is a warning: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." The results table has columns: customer_name, customer_email, customer_contact, and number_of_orders. The data shows two rows:

customer_name	customer_email	customer_contact	number_of_orders
Naomi Krishnarajha	naomi.k@iit.ac.lk	+94711234567	3
Nihal Kodikara	nihal.k@iit.ac.lk	+94744567890	3

Figure 97 results for question b)

Reference list

- Vertabelo Data Modeler. (2014). *How to Create a Database Model From Scratch*. [online] Available at: <https://vertabelo.com/blog/how-to-create-a-database-model-from-scratch/> [Accessed 1 Dec. 2023].
- Vertabelo Data Modeler. (2015). *Tips for Better Database Design*. [online] Available at: <https://vertabelo.com/blog/9-tips-for-better-database-design/> [Accessed 27 Nov. 2023].
- Connolly, T.M. and Begg, C. (2015). *Database systems: a practical approach to design, implementation and management*. Harlow: Pearson Education Limited.
- w3schools (2019). *SQL Tutorial*. [online] W3schools.com. Available at: <https://www.w3schools.com/sql/>.

Triggers

- GeeksforGeeks. (2018). *SQL | Triggers*. [online] Available at: https://www.geeksforgeeks.org/sql-triggers/?ref=ml_lbp [Accessed 27 Nov. 2023].
- www.youtube.com. (n.d.). *Triggers | SQL | Tutorial 20*. [online] Available at: <https://www.youtube.com/watch?v=gpthfJnvzY8> [Accessed 27 Nov. 2023].
- www.youtube.com. (n.d.). *Triggers In SQL | Triggers In Database | SQL Triggers Tutorial For Beginners | Edureka*. [online] Available at: <https://www.youtube.com/watch?v=f6VWSInHGCE> [Accessed 27 Nov. 2023].