```sql
DROP DATABASE IF EXISTS JungleBook;
CREATE DATABASE JungleBook;

USE JungleBook;

-- Creat Tables
CREATE TABLE Author
(
    AuthorID  INT(10)     NOT NULL AUTO_INCREMENT,
    FirstName VARCHAR(32) NOT NULL,
    LastName  VARCHAR(32) NOT NULL,
    PRIMARY KEY (AuthorID)
);
CREATE TABLE Book
(
    ISBN                  VARCHAR(80)  NOT NULL,
    Title                 VARCHAR(80)  NOT NULL,
    Summary               VARCHAR(255) NOT NULL,
    Price                 FLOAT        NOT NULL,
    AdditionalInformation VARCHAR(80),
    PRIMARY KEY (ISBN)
);
CREATE TABLE Book_Author
(
    ISBN     VARCHAR(80) NOT NULL,
    AuthorID INT(10)     NOT NULL,
    PRIMARY KEY (ISBN, AuthorID)
);
CREATE TABLE Book_Order
(
    ISBN        VARCHAR(80) NOT NULL,
    OrderNumber INT(10)     NOT NULL,
    Quantity    INT(10)     NOT NULL,
    TotalPrice  FLOAT,
    PRIMARY KEY (ISBN, OrderNumber)
);
CREATE TABLE Category
(
    CategoryID  INT(10)     NOT NULL AUTO_INCREMENT,
    Description VARCHAR(80) NOT NULL,
    PRIMARY KEY (CategoryID)
);
CREATE TABLE Category_Book
(
    CategoryID INT(10)     NOT NULL,
    ISBN       VARCHAR(80) NOT NULL,
    PRIMARY KEY (CategoryID, ISBN)
);
CREATE TABLE Customer
(
    UserName        VARCHAR(32) NOT NULL,
    FirstName       VARCHAR(32) NOT NULL,
    LastName        VARCHAR(32) NOT NULL,
    BillingAddress  VARCHAR(80) NOT NULL,
    ShippingAddress VARCHAR(80),
    PhoneNumber     VARCHAR(80) NOT NULL,
    EmailAddress    VARCHAR(80) NOT NULL,
    Password        VARCHAR(32) NOT NULL,
    PRIMARY KEY (UserName)
);
CREATE TABLE `Order`
(
    OrderNumber INT(10)     NOT NULL AUTO_INCREMENT,
    DateOrdered DATE        NOT NULL,
    DateShipped DATE,
    StatusOrder VARCHAR(80) NOT NULL,
    SubTotal    FLOAT       NOT NULL,
    Tax         FLOAT       NOT NULL,
    TotalCost   FLOAT       NOT NULL,
    UserName    VARCHAR(32) NOT NULL,
    PRIMARY KEY (OrderNumber)
);

-- Adding foreign keys
ALTER TABLE Book_Author
    ADD CONSTRAINT FK__Book__ISBN FOREIGN KEY (ISBN) REFERENCES Book (ISBN);
ALTER TABLE Book_Author
    ADD CONSTRAINT FK__Author__AuthorID FOREIGN KEY (AuthorID) REFERENCES Author (AuthorID);
ALTER TABLE Category_Book
    ADD CONSTRAINT FK__Category__CategoryID FOREIGN KEY (CategoryID) REFERENCES Category (CategoryID);
ALTER TABLE Category_Book
    ADD CONSTRAINT FK__Book__ISBN__Category__Book FOREIGN KEY (ISBN) REFERENCES Book (ISBN);
ALTER TABLE `Order`
    ADD CONSTRAINT FK__Customer__UserName FOREIGN KEY (UserName) REFERENCES Customer (UserName);
ALTER TABLE Book_Order
```

```sql
    ADD CONSTRAINT FK__Book__ISBN__Book__Order FOREIGN KEY (ISBN) REFERENCES Book (ISBN);
ALTER TABLE Book_Order
    ADD CONSTRAINT FK__Order__OrderNumber__Book__Order FOREIGN KEY (OrderNumber) REFERENCES `Order` (OrderNumber);


-- Add three books of your choice, their information and their categories to the database.

INSERT INTO Book
    (ISBN, Title, Summary, Price, AdditionalInformation)
VALUES (10, 'Holy Bible', 'New International Version', 100.5, 'This is the most sold book in history'),
       (20, 'Dababase Fundamentals', 'Used in NSCC', 150.5, 'Great Introductory Book'),
       (30, 'Python Games', 'Pygame', 200.5, 'Fun and Educational');

INSERT INTO Category
    (CategoryID, Description)
VALUES (1111, 'IT'),
       (2222, 'Religion'),
       (1133, 'Database');

INSERT INTO Category_Book
    (CategoryID, ISBN)
VALUES (1111, 20),
       (1111, 30),
       (2222, 10),
       (1133, 20);

INSERT INTO Author
    (AuthorID, FirstName, LastName)
VALUES (01, 'Paul', 'et al.'),
       (02, 'Brian', 'Shewan'),
       (03, 'Ronan', 'Driscoll');

INSERT INTO Book_Author
    (ISBN, AuthorID)
VALUES (10, 01),
       (20, 02),
       (30, 03);

```

```sql

-- Create customer information for four fictitious people and add their information to the database.

INSERT INTO Customer
(UserName, FirstName, LastName, BillingAddress, ShippingAddress, PhoneNumber, EmailAddress, Password)
VALUES ('customer1', 'Jack', 'Bauer', '11 Anywhere', '22 Anywhere', '781-234-9349', 'jbauser@24.com', 'jpass'),
       ('customer2', 'Tom', 'Bauer', '11 Somewhere', '22 Somewhere', '782-234-9349', 'tbauser@24.com', 'tpass'),
       ('customer3', 'Kevin', 'Bauer', '11 Nowhere', '11 Nowhere', '783-234-9349', 'kbauser@24.com', 'kpass'),
       ('customer4', 'Sue', 'Bauer', '11 Here', '11 Here', '784-234-9349', 'sbauser@24.com', 'spass');

-- Create three new orders
-- Customer 1 buys one copy of all three books
-- Customer 2 buys just one of the books
-- Customer 3 buys five copies of one book, two copies of the second
-- and ten copies of the third

INSERT INTO `Order`
(OrderNumber, DateOrdered, DateShipped, StatusOrder, SubTotal, Tax, TotalCost, UserName)
VALUES (10000, '2019-11-21', NULL, 'Pending confirmation', 451.5, 45.15, 496.65, 'customer1'),
       (20000, '2019-10-21', '2019-10-27', 'In progress', 100.5, 10.05, 110.55, 'customer2'),
       (30000, '2019-09-01', '2019-09-01', 'Complete', 2807.6, 280.76, 3088.36, 'customer3');


INSERT INTO Book_Order
    (ISBN, OrderNumber, Quantity, TotalPrice)
VALUES (10, 10000, 1, 100.5),
       (20, 10000, 1, 150.5),
       (30, 10000, 1, 200.5),
       (10, 20000, 1, 100.5),
       (10, 30000, 5, 502.5),
       (20, 30000, 2, 301.0),
       (30, 30000, 10, 2005);


```

```sql
-- Delete the second order
DELETE
FROM Book_Order
WHERE OrderNumber = 20000;

DELETE
FROM `Order`
WHERE OrderNumber = 20000;

-- Change the status of the first order to complete.
UPDATE `Order`
SET StatusOrder = 'Complete'
WHERE OrderNumber = 10000;

-- Change the third order to add another copy of the second book.
-- Change prices both in book_order and order
UPDATE Book_Order
SET Quantity = 3
WHERE ISBN = 20
  AND OrderNumber = 30000;

UPDATE `Order`
SET SubTotal= 2959,
    Tax = 295.5,
    TotalCost = 3254.9
WHERE OrderNumber = 30000;


-- Display all customer information for customers that have no orders. (i.e.
-- the count of their order ids is zero)
-- display all customers with no orders.
SELECT C.UserName, COUNT(O.OrderNumber) AS ORDER_COUNT
FROM Customer C
         LEFT OUTER JOIN `Order` O
                        ON C.UserName = O.UserName
GROUP BY UserName
HAVING Order_COUNT = 0;


-- Display the title, author, ISBN and price of all books related to databases.
-- (i.e. contain the word "database" or are in the " database" category)
-- To solve the above problem, I am adding another DB book.

INSERT INTO Book
    (ISBN, Title, Summary, Price, AdditionalInformation)
VALUES (40, 'Advanced SQL', 'Intermediate DB Book', 200.5
, 'Great sql Book');

INSERT INTO Category_Book
    (CategoryID, ISBN)
VALUES (1133, 40);

INSERT INTO Author
    (AuthorID, FirstName, LastName)
VALUES (04, 'John', 'Joe');

INSERT INTO Book_Author
    (ISBN, AuthorID)
VALUES (40, 4);

-- Solutions
SELECT Title, A.FirstName, A.LastName, B.ISBN, B.Price
FROM Book B
         INNER JOIN Category C,
     Category_book CB,
     Author A,
     Book_author BA
WHERE (C.Description = 'Database' OR C.CategoryID = 1133)
  AND B.ISBN = CB.ISBN
  AND CB.CategoryID = C.CategoryID
  AND B.ISBN = BA.ISBN
  AND A.AuthorID = BA.AuthorID;



-- Display the email addresses of customers who have outstanding orders. ==> Working!!
-- (i.e. orders that have no ship date yet)
SELECT FirstName, LastName, EmailAddress
FROM Customer C
         INNER JOIN `Order` O
WHERE DateShipped IS NULL
  AND O.UserName = C.UserName;


```

```sql

-- Display just the order information (i.e. not the order details) on all orders
-- that have purchased more than one copy of any of the books. (e.g. two
-- copies of book one, three copies of book three, etc)
SELECT ISBN, OrderNumber, Quantity, TotalPrice
FROM Book_order
WHERE Book_Order.Quantity > 1;


-- Display the order number and the total number of books in each order.
-- This will be used to calculate shipping costs. ==> shipping cost?? Question asked => waiting reply...
-- I might need to calculate from scratch too ==> Recheck !
# Final solution. working!!!!
SELECT O.OrderNumber, SUM(Quantity)
FROM `Order` O
         INNER JOIN Book_order BO
WHERE O.OrderNumber = BO.OrderNumber
GROUP BY O.OrderNumber;


-- Display each order number, customer name and the total cost of of their
-- order by adding the cost of all the items. (Determine each cost by using
-- the quantity and price and adding 15% tax).0
-- Here, calculate from the scratch.
SELECT ROUND(SUM(Quantity * Price * 1.15),2) AS TotalCost, B.OrderNumber, C.FirstName, C.LastName
FROM Book
         INNER JOIN Book_Order B,
     `Order` O,
     Customer C
WHERE Book.ISBN = B.ISBN
  AND B.OrderNumber = O.OrderNumber
  AND O.UserName = C.UserName
GROUP BY B.OrderNumber;
```