

```
1
2 -- create the database
3 DROP DATABASE IF EXISTS JungleBooks;
4 CREATE DATABASE JungleBooks;
5
6 USE JungleBooks;
7
8
9 -- create the tables
10 DROP TABLE IF EXISTS Customer;
11 CREATE TABLE Customer (
12     CustomerID INT PRIMARY KEY,
13     FirstName VARCHAR(32) NOT NULL,
14     LastName VARCHAR(32) NOT NULL,
15     BillingAddress VARCHAR(255) NOT NULL,
16     ShippingAddress VARCHAR(255) NOT NULL,
17     PhoneNumber CHAR(10) NOT NULL,
18     EmailAddress VARCHAR(32) NOT NULL,
19     Username VARCHAR(32) NOT NULL,
20     Password VARCHAR(32) NOT NULL
21 );
22
23 DROP TABLE IF EXISTS CustomerOrder;
24 CREATE TABLE CustomerOrder(
25     CustomerOrderID INT PRIMARY KEY,
26     DateOrdered DATETIME NOT NULL,
27     DateShipped DATETIME NULL,
28     Status VARCHAR(32) NOT NULL,
29     Subtotal FLOAT NOT NULL,
30     Taxes FLOAT NOT NULL,
31     TotalCost FLOAT NOT NULL,
32     CustomerID INT NOT NULL
33 );
34
35 DROP TABLE IF EXISTS Book;
36 CREATE TABLE Book(
37     BookID INT PRIMARY KEY,
38     Title VARCHAR(255) NOT NULL,
39     Summary VARCHAR(255) NOT NULL,
40     ISBN VARCHAR(32) NOT NULL,
41     Price FLOAT NOT NULL
42 );
43
44 DROP TABLE IF EXISTS Author;
45 CREATE TABLE Author(
46     AuthorID INT PRIMARY KEY,
```

```
47     FirstName VARCHAR(32) NOT NULL,
48     LastName VARCHAR(32) NOT NULL
49 );
50
51 DROP TABLE IF EXISTS Category;
52 CREATE TABLE Category(
53     CategoryID INT PRIMARY KEY,
54     Description VARCHAR(32) NOT NULL
55 );
56
57 DROP TABLE IF EXISTS Book_CustomerOrder;
58 CREATE TABLE Book_CustomerOrder(
59     BookID INT NOT NULL,
60     CustomerOrderID INT NOT NULL,
61     Quantity INT NOT NULL,
62     Subtotal FLOAT NOT NULL
63 );
64
65 DROP TABLE IF EXISTS Book_Author;
66 CREATE TABLE Book_Author(
67     BookID INT NOT NULL,
68     AuthorID INT NOT NULL
69 );
70
71 DROP TABLE IF EXISTS Book_Category;
72 CREATE TABLE Book_Category(
73     BookID INT NOT NULL,
74     CategoryID INT NOT NULL
75 );
76
77
78 -- create relationships
79 ALTER TABLE CustomerOrder
80     DROP CONSTRAINT IF EXISTS FK__Customer__CustomerID;
81 ALTER TABLE CustomerOrder
82     ADD CONSTRAINT FK__Customer__CustomerID FOREIGN KEY (
83         CustomerID)
84     REFERENCES Customer (CustomerID);
85
86 ALTER TABLE Book_CustomerOrder
87     DROP CONSTRAINT IF EXISTS PK_Book_CustomerOrder;
88 ALTER TABLE Book_CustomerOrder
89     ADD CONSTRAINT PK_Book_CustomerOrder
90     PRIMARY KEY (BookID, CustomerOrderID);
91
```

```

92 ALTER TABLE Book_CustomerOrder
93     DROP CONSTRAINT IF EXISTS FK_Book1_BookID;
94 ALTER TABLE Book_CustomerOrder
95     ADD CONSTRAINT FK_Book1_BookID
96     FOREIGN KEY (BookID)
97     REFERENCES Book (BookID);
98
99 ALTER TABLE Book_CustomerOrder
100     DROP CONSTRAINT IF EXISTS
101     FK_CustomerOrder_CustomerOrderID;
101 ALTER TABLE Book_CustomerOrder
102     ADD CONSTRAINT FK_CustomerOrder_CustomerOrderID
103     FOREIGN KEY (CustomerOrderID)
104     REFERENCES CustomerOrder (CustomerOrderID);
105
106
107 ALTER TABLE Book_Author
108     DROP CONSTRAINT IF EXISTS PK_Book_Author;
109 ALTER TABLE Book_Author
110     ADD CONSTRAINT PK_Book_Author
111     PRIMARY KEY (BookID, AuthorID);
112
113 ALTER TABLE Book_Author
114     DROP CONSTRAINT IF EXISTS FK_Book2_BookID;
115 ALTER TABLE Book_Author
116     ADD CONSTRAINT FK_Book2_BookID
117     FOREIGN KEY (BookID)
118     REFERENCES Book (BookID);
119
120 ALTER TABLE Book_Author
121     DROP CONSTRAINT IF EXISTS FK_Author_AuthorID;
122 ALTER TABLE Book_Author
123     ADD CONSTRAINT FK_Author_AuthorID
124     FOREIGN KEY (AuthorID)
125     REFERENCES Author (AuthorID);
126
127
128 ALTER TABLE Book_Category
129     DROP CONSTRAINT IF EXISTS PK_Book_Category;
130 ALTER TABLE Book_Category
131     ADD CONSTRAINT PK_Book_Category
132     PRIMARY KEY (BookID, CategoryID);
133
134 ALTER TABLE Book_Category
135     DROP CONSTRAINT IF EXISTS FK_Book3_BookID;
136 ALTER TABLE Book_Category

```

```

137     ADD CONSTRAINT FK_Book3_BookID
138     FOREIGN KEY (BookID)
139     REFERENCES Book (BookID);
140
141 ALTER TABLE Book_Category
142     DROP CONSTRAINT IF EXISTS FK_Category_CategoryID;
143 ALTER TABLE Book_Category
144     ADD CONSTRAINT FK_Category_CategoryID
145     FOREIGN KEY (CategoryID)
146     REFERENCES Category (CategoryID);
147
148
149 -- INSERTing categories
150 INSERT INTO Category
151     (CategoryID, Description)
152 VALUES
153     (1, 'Classic Regency novel'), (2, 'Fantasy'), (3, '
Adventure'), (4, 'Science fiction'), (5, 'Comedy'), (6, '
Database');
154
155
156 -- INSERTing authors
157 INSERT INTO Author
158     (AuthorID, FirstName , LastName)
159 VALUES
160     (1, 'Jane', 'Austen'), (2, 'J. R. R.', 'Tolkien'), (3
, 'Douglas', 'Adams');
161
162
163 -- INSERTing books
164 INSERT INTO Book
165     (BookID, Title, Summary, ISBN, Price)
166 VALUES
167     (1, 'Pride and Prejudice and Database', 'Pride and
Prejudice has delighted generations of readers with its
unforgettable cast of characters, carefully choreographed
plot, and a hugely entertaining view of the world and its
absurdities.', '9780199535569', 13.5),
168     (2, 'Lord of the Rings: The Two Towers', 'Dispatched,
from the UK, within 48 hours of ordering. This book is in
good condition but will show signs of previous ownership.
Please expect some creasing to the spine and/or minor
damage to the cover.', '9780048231567', 9.9),
169     (3, 'The Hitchhikers Guide to the Galaxy', 'Campus
mission is to set online bookstore industry standards for
savings, selection, convenience and customer service as

```

```

169 expressed in the companys slogan Textbooks Easy. Fast.
    Cheap!', '9780345391803', 8.5);
170
171
172 -- INSERTing book categories
173 INSERT INTO Book_Category
174     (CategoryID, BookID)
175 VALUES
176     (1, 1), (2, 2), (3, 2), (4, 3), (5, 3), (6,3);
177
178
179 -- INSERTing book authors
180 INSERT INTO Book_Author
181     (BookID, AuthorID)
182 VALUES
183     (1, 1), (2, 2), (3, 3);
184
185
186 -- INSERTing customers
187 INSERT INTO Customer
188     (CustomerID, FirstName, LastName, BillingAddress,
    ShippingAddress, PhoneNumber, EmailAddress, Username,
    Password)
189 VALUES
190     (1, 'Reginaldo', 'Oliveira', 'Main Street 13 B3LN5C',
    'Main Street 13 B3LN5C', '9021012002', 'ReginaldoOliveira@
    gmail.com', 'Reginaldo121', 'R3g1nald0'),
191     (2, 'Anderson', 'Medina', 'South Street 521 E2LG3D', '
    South Street 521 E2LG3D', '9026035555', 'AndersonMedina@
    gmail.com', 'AndersonMDN', '4nd3rs0n'),
192     (3, 'Luiz', 'Souza', 'Young Street 1452 A3LN1C', '
    Young Street 1452 A3LN1C', '9021100555', 'LuizSouza@gmail.
    com', 'Luiz1987', 'Lu1z@2304'),
193     (4, 'Maria', 'Silva', 'North Street 322 N3LW5C', '
    North Street 322 N3LW5C', '9025023333', 'MariaSilva@gmail.
    com', 'Maria_864', 'M4r14!1221');
194
195
196 -- INSERTing customer`s order
197 INSERT INTO CustomerOrder
198     (CustomerOrderID, DateOrdered, DateShipped, Status,
    Subtotal, Taxes, TotalCost, CustomerID)
199 VALUES
200     (1, '2019-10-31T18:00:00', '2019-11-11T18:00:00', '
    Completed', 31.9, 1.15, 36.685, 1),
201     (2, '2019-11-20T22:00:00', '2019-11-26T18:00:00', '

```

```

201 Completed', 13.5, 1.15, 15.525, 2);
202
203 INSERT INTO CustomerOrder
204     (CustomerOrderID, DateOrdered, Status, Subtotal, Taxes
205     , TotalCost, CustomerID)
206 VALUES
207     (3, '2019-11-21T10:00:00', 'In progress', 172.3, 1.15
208     , 198.145, 3);
209
210 -- INSERTing books to a CustomerOrder
211 INSERT INTO Book_CustomerOrder
212     (CustomerOrderID, BookID, Quantity, Subtotal)
213 VALUES
214     (1, 1, 1, 13.50), (1, 2, 1, 9.90), (1, 3, 1, 8.50),
215     (2, 1, 1, 13.50),
216     (3, 1, 5, 67.5), (3, 2, 2, 19.8), (3, 3, 10, 85.0);
217
218 -- DELETEing the second order
219 DELETE FROM Book_CustomerOrder
220 WHERE CustomerOrderID = 2;
221
222 DELETE FROM CustomerOrder
223 WHERE CustomerOrderID = 2;
224
225
226 -- UPDATEing the status of the first order to complete
227 UPDATE CustomerOrder
228 SET Status = 'Complete'
229 WHERE CustomerOrderID = 1;
230
231
232 -- UPDATEing the third order to add another copy of the
233     second book
234 UPDATE Book_CustomerOrder BCO
235 RIGHT OUTER JOIN Book B
236 ON BCO.BookID = B.BookID
237 SET BCO.Quantity = (BCO.Quantity + 1), BCO.Subtotal = (BCO
238     .Subtotal + B.Price)
239 WHERE BCO.CustomerOrderID = 3 AND BCO.BookID = 2;
240
241 UPDATE CustomerOrder CO
242 INNER JOIN
243     (
244         SELECT BCO.CustomerOrderID, SUM(Subtotal) AS

```

```

242 SUBTOTAL_SUM
243     FROM Book_CustomerOrder BCO
244     WHERE BCO.CustomerOrderID = 3
245     GROUP BY BCO.CustomerOrderID
246 ) BCO
247 ON BCO.CustomerOrderID = CO.CustomerOrderID
248 SET CO.Subtotal = BCO.SUBTOTAL_SUM, CO.TotalCost = (CO.
    Subtotal * CO.Taxes)
249 WHERE CO.CustomerOrderID = 3;
250
251
252 -- SELECTing all customer information for customers that
    have no orders. (i.e. the count of their order ids is zero
    )
253 SELECT C.*, COUNT(CO.CustomerID) AS CUSTOMER_COUNT
254 FROM Customer C
255 LEFT OUTER JOIN CustomerOrder CO
256 ON C.CustomerID = CO.CustomerID
257 GROUP BY CustomerID
258 HAVING CUSTOMER_COUNT = 0;
259
260
261 -- SELECTing the title, author, ISBN and price of all
    books related to databases. (i.e. contain the word "
    database" or are in the "database" category)
262 SELECT B.Title, A.FirstName, A.LastName, B.ISBN, B.Price
263 FROM Book B
264 LEFT OUTER JOIN Book_Author BA
265 ON B.BookID = BA.BookID
266 LEFT OUTER JOIN Author A
267 ON BA.AuthorID = A.AuthorID
268 RIGHT OUTER JOIN Book_Category BC
269 ON B.BookID = BC.BookID
270 RIGHT OUTER JOIN Category C
271 ON BC.CategoryID = C.CategoryID
272 WHERE B.Title LIKE '%database%' OR C.Description = '
    database'
273 GROUP BY B.Title;
274
275
276 -- SELECTing the email addresses of customers who have
    outstanding orders. (i.e. orders that have no ship date
    yet)
277 SELECT C.EmailAddress, COUNT(CO.DateShipped) AS
    DATESHIPPED_COUNT, COUNT(CO.CustomerID) AS CUSTOMER_COUNT
278 FROM Customer C

```

```
279 LEFT OUTER JOIN CustomerOrder CO
280 ON C.CustomerID = CO.CustomerID
281 GROUP BY C.EmailAddress
282 HAVING DATESHIPPED_COUNT = 0 AND CUSTOMER_COUNT > 0;
283
284
285 -- SELECTing 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)
286 SELECT CO.CustomerOrderID, SUM(BCO.Quantity) AS
    QUANTITY_SUM, COUNT(BCO.Quantity) AS QUANTITY_COUNT
287 FROM CustomerOrder CO
288 LEFT OUTER JOIN Book_CustomerOrder BCO
289 ON CO.CustomerOrderID = BCO.CustomerOrderID
290 GROUP BY CO.CustomerOrderID
291 HAVING QUANTITY_SUM > QUANTITY_COUNT;
292 -- I'm sorry, but I didn't understand exactly what should
    be shown when you say "Display just the order information
    (ie not the order details)". I understood that it was to
    show only the order number
293
294
295 -- SELECTing the order number and the total number of
    books in each order.
296 SELECT CO.CustomerOrderID, SUM(BCO.Quantity) AS
    QUANTITY_SUM
297 FROM CustomerOrder CO
298 LEFT OUTER JOIN Book_CustomerOrder BCO
299 ON CO.CustomerOrderID = BCO.CustomerOrderID
300 GROUP BY CO.CustomerOrderID;
301
302
303 -- SELECTing each order number, customer name and the
    total cost of their order by adding the cost of all the
    items. (Determine each cost by using the quantity and
    price and adding 15% tax).
304 SELECT CO.CustomerOrderID, C.FirstName, C.LastName, SUM(B.
    Price * BCO.Quantity * CO.Taxes) AS TotalCost_SUM
305 FROM CustomerOrder CO
306 LEFT OUTER JOIN Customer C
307 ON CO.CustomerID = C.CustomerID
308 RIGHT OUTER JOIN Book_CustomerOrder BCO
309 ON CO.CustomerOrderID = BCO.CustomerOrderID
310 RIGHT OUTER JOIN Book B
311 ON BCO.BookID = B.BookID
```


312 **GROUP BY** C0.CustomerOrderID;

313

314