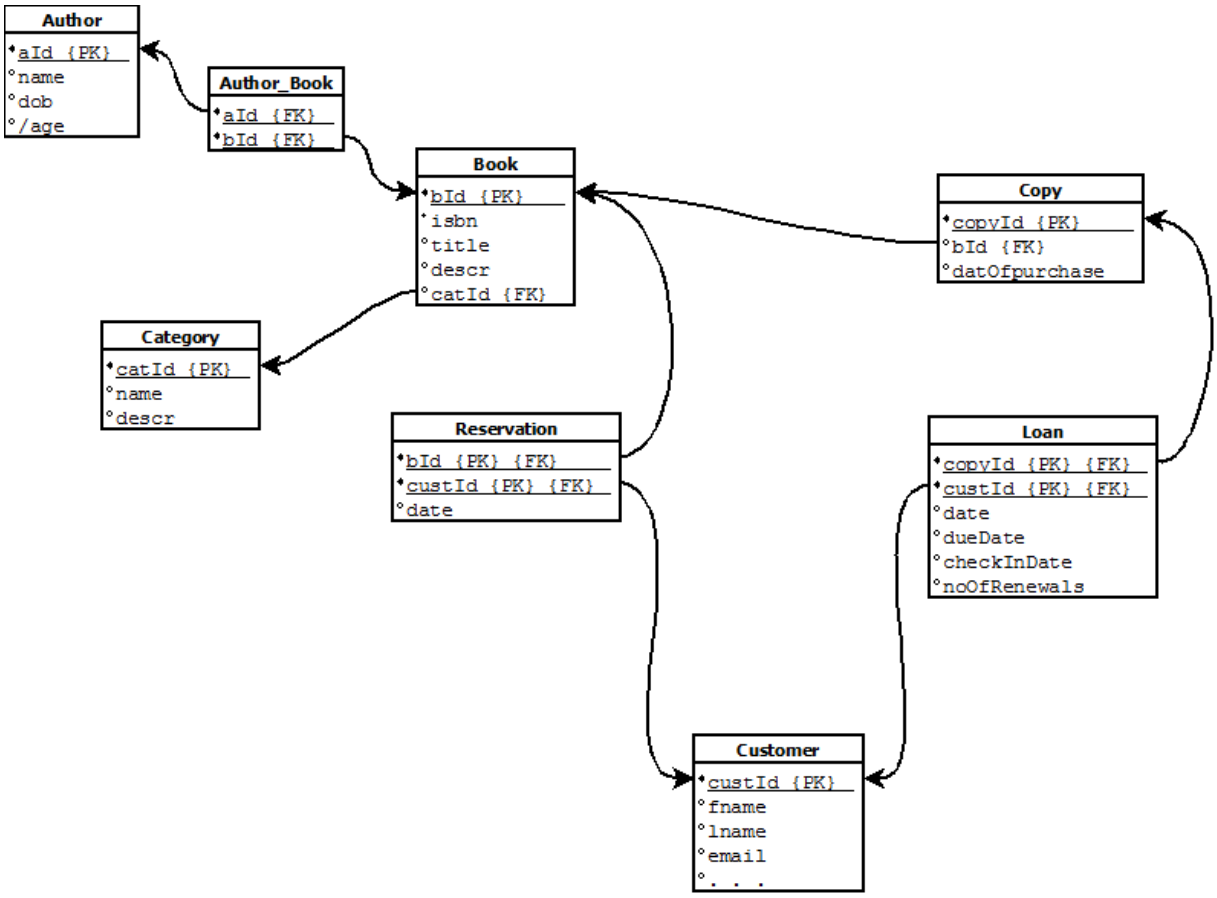
Övning 2
Förslag till schema för del av problemet, samt SQL-satser utifrån denna (förenklade lösning).

Schemat (för ER-modell, se kommentarer till övning 1):
Notera att vi behöver en sambandstabell för många-till-många-sambandet mellan bok och författare.



Alternativt kan schemat beskrivas som nedan.

Customer (custId, fname, lname, email, . . .)
Category (catId, name, descry)
Book (bId, isbn, title, descry, catId)
Copy (copyId, bId, dateOfPurchase)
Loan(copyId, custId, date, dueDate, checkInDate, noOfRenewals)
Reservation(bId, custId, date)
Author(aId, name, dob)
Author_Book(aId, bId)

Ett förtydligande av referensattributen (framgår annars av valda attributnamn):

Book.catId refererar Category.catId
Copy.bId refererar Book.bId
Loan.copyId refererar Copy.copyId och Loan.custId refererar Customer.custId
Reservation.bId refererar Book.bId och Reservation.custId refererar Customer.custId
Author_Book.aid refererar Author.aid och Author_Book.bid refererar Book.bid

SQL-satser

1. CREATE SEQUENCE Customer_custId_seq START 1;

```
CREATE TABLE T_Customer (
custId INT NOT NULL DEFAULT nextval('Customer_ custId _seq'),
fname VARCHAR(30) NOT NULL,
lname VARCHAR(30) NOT NULL,
email VARCHAR(30),
. . .
PRIMARY KEY(custId)
);
```

2. CREATE SEQUNCE Book_bId_seq START 1;

```
. . .

CREATE TABLE T_Book (
bId INT NOT NULL DEFAULT nextval('Book_bId_seq'),
isbn VARCHAR(20) NOT NULL,
title VARCHAR(50) NOT NULL,
publ VARCHAR(30),
descr VARCHAR(1000),              [möjligtvis CLOB?]
. . .
category INT NOT NULL,
UNIQUE(isbn),                     [alternativ nyckel]
PRIMARY KEY(bId),
FOREIGN KEY(catId) REFERENCES T_Category(cId) [förutsätter T_Category]
);

CREATE TABLE T_Copy (
copyId INT NOT NULL DEFAULT nextval('Copy_copyId_seq'),
bId INT NOT NULL,
datOfpurchase DATE,
PRIMARY KEY(copyId),
FOREIGN KEY(bId) REFERENCES T_Book(bId)
);

CREATE TABLE T_Loan (             [separat PK?]
custId INT NOT NULL,
copyId INT NOT NULL,
date DATE NOT NULL,
dueDate DATE NOT NULL,
checkInDate DATE,
noOfRenewals INT,
PRIMARY KEY(custId, copyId),
FOREIGN KEY(custId) REFERENCES T_Customer(custId),
FOREIGN KEY(copyId) REFERENCES T_Copy(copyId)
);
```

Notera att om tabellerna ovan definierats i annan ordning hade troligen fått lägga till visa referensattribut iefterhand med “ALTER TABLE”.

3. T_Loan kan betraktas som en sambandstabell mellan T_Copy och T_Customer. En eller flera böcker kan vara reserverade av en kund, vilket ger tabellen T_Reservation.

```
CREATE TABLE T_Reservation (                    [separat PK?]
bId INT NOT NULL,
custId INT NOT NULL,
date DATE NOT NULL,
PRIMARY KEY(custId, bId),
FOREIGN KEY(custId) REFERENCES T_Customer(custId),
FOREIGN KEY(bId) REFERENCES T_Book(bId)
);
```

4.
```
INSERT INTO T_Customer (fname, lname, email)
VALUES ('Anders', 'Lindström', 'anderslm@kth.se');

INSERT INTO T_Book (isbn, title, publ, descr)
VALUES ('112-222-333', '1Q84, 3e boken', 'Studentlitteratur',
'Fängslande');

INSERT INTO T_Copy (bId, datOfpurchase)
VALUES (1, '2011-10-25');
```

5.
```
INSERT INTO T_Loan (custId, copyId, date, dueDate, checkInDate,
noOfRenewals)
VALUES (1, 1, '2011-10-31', '2011-11-30', NULL, NULL);
```

6.
```
SELECT * FROM T_Book
WHERE publ LIKE 'Studentlitteratur';
```

7. T.ex. räkna antal (icke null) värden I tabellen som listar alla kopior med den givna titeln.
```
SELECT COUNT(c.copyId) AS noOfCopies
FROM (SELECT * FROM T_Copy, c, T_Book b
      WHERE c.bId = b.bId AND b.name LIKE 'Microserfs');
```

8.
```
SELECT cust.*
FROM T_Customer cust, T_Loan l, T_Copy copy, T_Book b
WHERE cust.custId = l.custId AND l.copyId = copy.copyId AND
copy.bId = b.bId
AND b.name LIKE '2666';
```

9.
```
SELECT T_Book.title, COUNT(T_Copy.cId)
FROM (SELECT * FROM T_Book b, T_Copy c
      WHERE b.bId = c.bId)
GROUP BY T_Book.title;
```

10.
```
CREATE TABLE T_Author (
aId INT NOT NULL DEFAULT nextval('Author_aId_seq'),
fname VARCHAR(30) NOT NULL,
lname VARCHAR(30) NOT NULL,
dob DATE,
descr VARCHAR(1000),
PRIMARY KEY(aId)
);

CREATE TABLE T_Author_Book (
aId INT NOT NULL,
bId INT NOT NULL,
PRIMARY KEY(aId, bId),
FOREIGN KEY(aId) REFERENCES T_Author(aId),
FOREIGN KEY(bId) REFERENCES T_Book(bId)
);
```

11.
```
SELECT DISTINCT cust.custId, T_Customer.name
FROM T_Customer cust, T_Loan l, T_Copy copy, T_Book b,
T_Author_Book ab, T_Author a
WHERE a.aId = ab.aId AND ab.bId = b.bId AND b.bId = copy.bId
AND copy.bId = l.bId AND l.custId = cust.custId
AND a.name LIKE 'Douglas Coupland';
```