Isha Chaturvedi – ic1018@nyu.edu

HW 3

Problem 2:

a. Candidate Key: {AE}
b. Canonical cover: Fc = {E → BG, A → BCDFG}
c. It is not in BCNF because there is nontrivial functional dependency CD → F, and CD is not a superkey.
   Decomposing relation R into BCNF:
   Because E → BG and E is not a superkey: R1 = {E,B,G} and R2 = {ACDEF}.
   Because A → CDF and A is not a superkey, decomposing relation R2 into BCNF:
   R2 = {A,C,D,F} and R3 = {A,E}.

   Thus the BCNF form is: R1 = {E,B,G}, R2 = {A,C,D,F} and R3 = {A,E}.
d. No. The BCNF form in (c) is not dependency preserving. We cannot check A → BG in the result of (c).
   3NF form: R1 ={E,B,G}, R2 ={A,B,C,D,F,G}, R2={A,E}.

Problem 3:

a. This is not a good design for following reasons:

   1) Current schema does not reflect a lot of functional dependencies. For instance, as per assumptions in the question, *{cid → cname, email, phone}* holds, but since the *{cid}* is not a candidate key, we will end up repeating this information in several tuples.

   2) Inserting values for the separate entities in one table like customer or theater will lead to storing NULL and duplicate values at several places. This will increase the storage and our data will be inconsistent.

   3) We have to access this one table for querying even minimal data, for instance, information for one single customer, this will be time consuming as everything *{theater, ticket}* is under this table.

   4) Lastly, this schema does not adhere to the normalization rules. As everything is stored in one table, it will make maintenance and querying from the database difficult and it will be hard to see relation between each attribute. The relational database system works better with several small tables than a big table.

b. Following are all non-trivial functional dependencies:

It Is assumed here that the input value of discount would be 1 if there is no discount.

{cid} → {cname, email, phone}
{tid} → {tname, taddress}
{tid, screennum} → {capacity}
{tid, sucreennum, mtime} → {mid,mname, mtype, mgenre}
{mid} → {mname, mtype, mgenre}
{cid, tid, sucreennum, mtime, ticketnum} → {price, discount}

c. Candidate Keys: {cid, tid, mid, mtime, screennum, ticketnum}

d. The canonical cover for the functional dependencies in F is:

{cid} → {cname, email, phone}
{tid} → {tname, taddress}
{tid, screennum} → {capacity}
{tid, sucreennum, mtime} → {mid}
{mid} → {mname, mtype, mgenre}
{cid, tid, sucreennum, mtime, ticketnum} → {price, discount}

e. No, it is not in BCNF. Because for schema to be BCNF for each non-trivial dependency A->B, A should be a superkey, which is not the case here.

Convert into BCNF form:

Step 1: Relation TicketInformation can be decomposed into CUSTOMER and R2 according to the violation by dependency {cid} → {cname, email, phone}, we can get relation Customer:
Customer (cid, cname, email, phone)
R2= {cid, tid, tname, taddress, mid, mname, mtype, mgenre, mtime, screennum, capacity, ticketnum, price, discount}

Step 2: According to the violation by functional dependency {tid} → {tname, taddress}, We can get relation Theater:
Theater (tid, tname, taddress)
R3={cid, tid, mid, mname, mtype, mgenre, mtime, screennum, capacity, ticketnum, price, discount}

Step3: According to the violation by functional dependency {tid, screennum} → {capacity}, We can get relation Capacity:
Capacity(tid, screennum, capacity)
R4={cid, tid, mid, mname, mtype, mgenre, mtime, screennum, ticketnum, price, discount}

Step4: According to the violation by functional dependency {tid, sucreennum, mtime} → {mid}, We can get Showing:
Showing (tid, sucreennum, mtime,mid)
R5 ={cid, tid, mid, mname, mtype, mgenre, mtime, screennum, ticketnum, price, discount}

Step 5: According to the violation by functional dependency {mid} → {mname, mtype, mgenre} , we can get relation Movie:
Movie (mid, mname, mtype, mgenre)
R6 ={cid, tid, mid, mtime, screennum, ticketnum, price, discount}

Step 6: According to the violation by functional dependency {cid, tid, sucreennum, mtime, ticketnum} → {price, discount} , we can get relation Ticket:
Ticket (cid, tid, sucreennum, mtime, ticketnum, price, discount)
R7 ={cid, tid, mid, mtime, screennum, ticketnum}

Thus, the BCNF form is:

Customer (cid, cname, email, phone)
Theater(tid, tname, taddress)
Capacity (tid, screennum, capacity)
Showing (tid, sucreennum, mtime,mid)
Movie (mid, mname, mtype, mgenre)
Ticket (cid, tid, sucreennum, mtime, ticketnum, price, discount)
TicketInformation (cid, tid, mid, mtime, screennum, ticketnum)

f. Yes, the schema in section e is dependency preserving because each relation carries one of the relationships in canonical cover.

g. If we add an additional constraint ---

b. non-trivial functional dependencies:

{cid} → {cname, email, phone}
{tid} → {tname, taddress}
{tid, screennum} → {capacity}
{tid, sucreennum, mtime} → {mid,mname, mtype, mgenre}
{mid} → {mname, mtype, mgenre}
{cid, tid, sucreennum, mtime, ticketnum} → {price, discount}
{tid, sucreennum, mtime} → {price}

c. candidate keys: {cid, tid, mid, mtime, screennum, ticketnum}

d. canonical cover:

{cid} → {cname, email, phone}
{tid} → {tname, taddress}
{tid, screennum} → {capacity}
{tid, sucreennum, mtime} → {mid}
{mid} → {mname, mtype, mgenre}
{cid, tid, sucreennum, mtime, ticketnum} → {price, discount}
{tid, sucreennum, mtime} → {price}

e. No, it is not in BCNF.
Decomposition (BCNF form):

Customer (cid, cname, email, phone)
Theater (tid, tname, taddress)
Capacity (tid, screennum, capacity)
Showing (tid, sucreennum, mtime,mid)
Movie (mid, mname, mtype, mgenre)
Ticket (cid, tid, sucreennum, mtime, ticketnum, price, discount)
TIcketPrice (tid, sucreennum, mtime, price)
TicketInformation (cid, tid, mid, mtime, screennum, ticketnum)

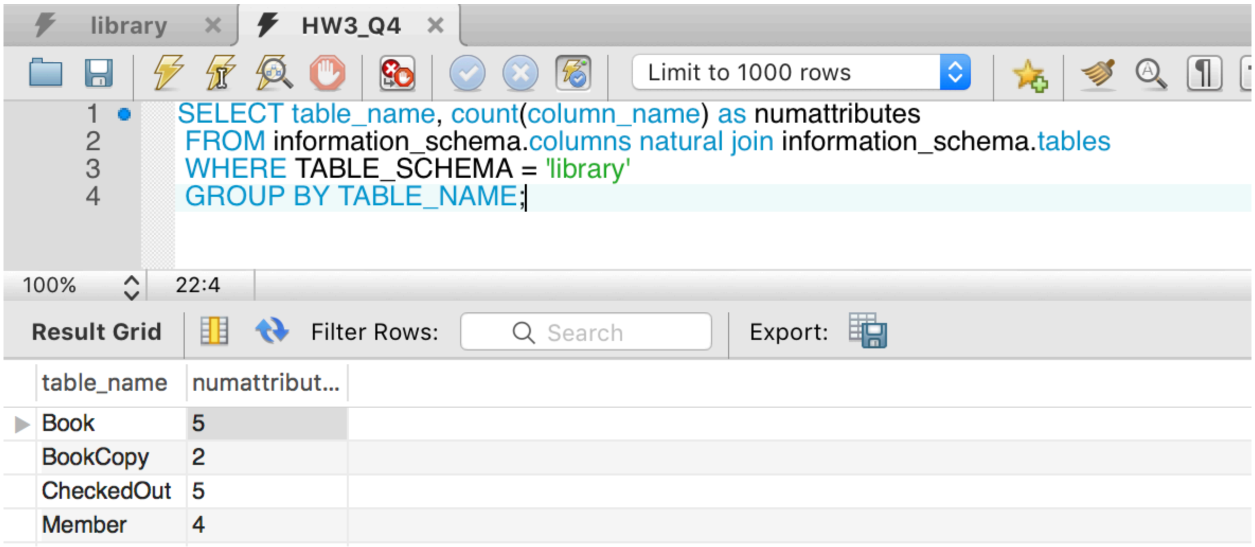
f. No, the schema in e. is not dependency preserving because the functional dependency {date, mgenre} → {discount} cannot be checked in the BCNF form. Thus, the 3NF form is:

Customer (cid, cname, email, phone)
Theater(tid, tname, taddress)
Capacity (tid, screennum, capacity)
Showing (tid, sucreennum, mtime,mid)
Movie (mid, mname, mtype, mgenre)
Ticket (cid, tid, sucreennum, mtime, ticketnum, price)
Discount (date, mgenre, discount)
TicketInformation (cid, tid, mid, date, time, screennum, ticketnum)

Problem 4:

Using MySQL

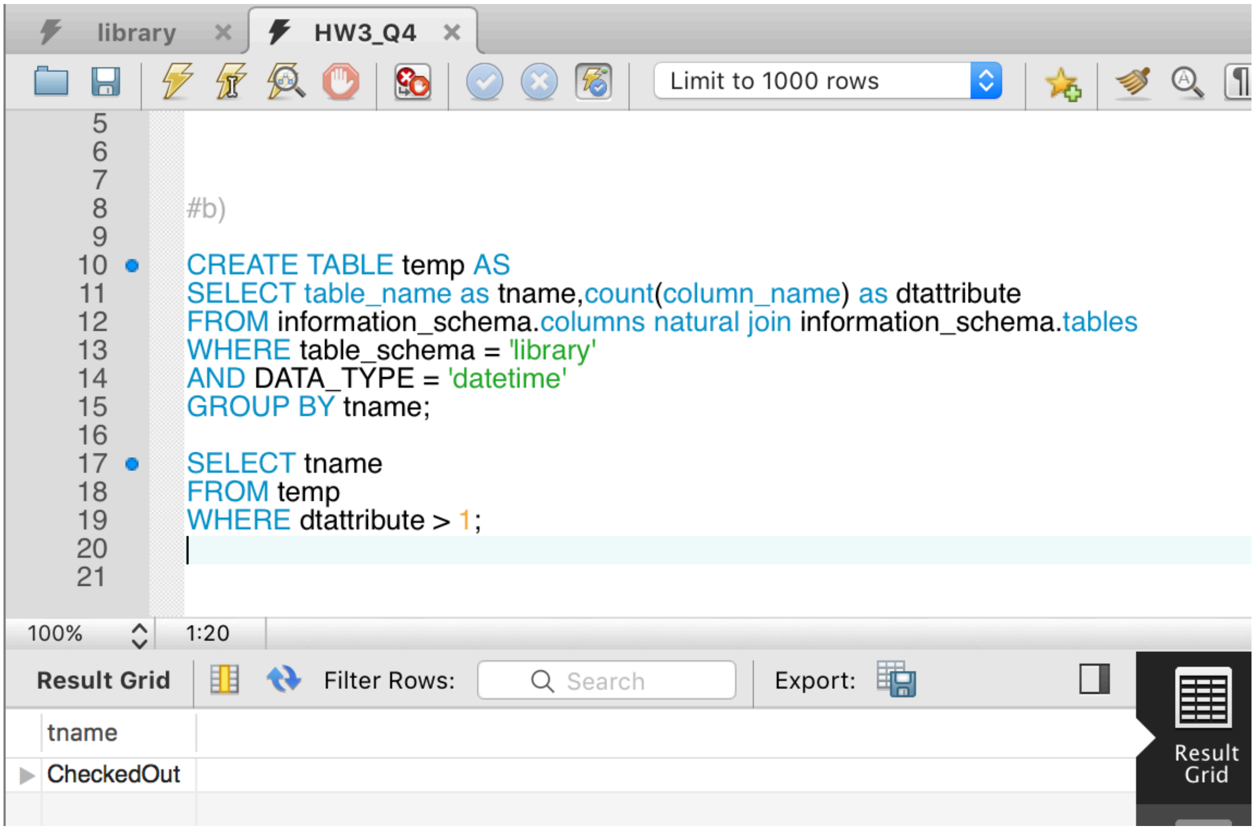
a.



```
SELECT table_name, count(column_name) as numattributes
 FROM information_schema.columns natural join information_schema.tables
 WHERE TABLE_SCHEMA = 'library'
 GROUP BY TABLE_NAME;
```

table_name	numattribut...
Book	5
BookCopy	2
CheckedOut	5
Member	4

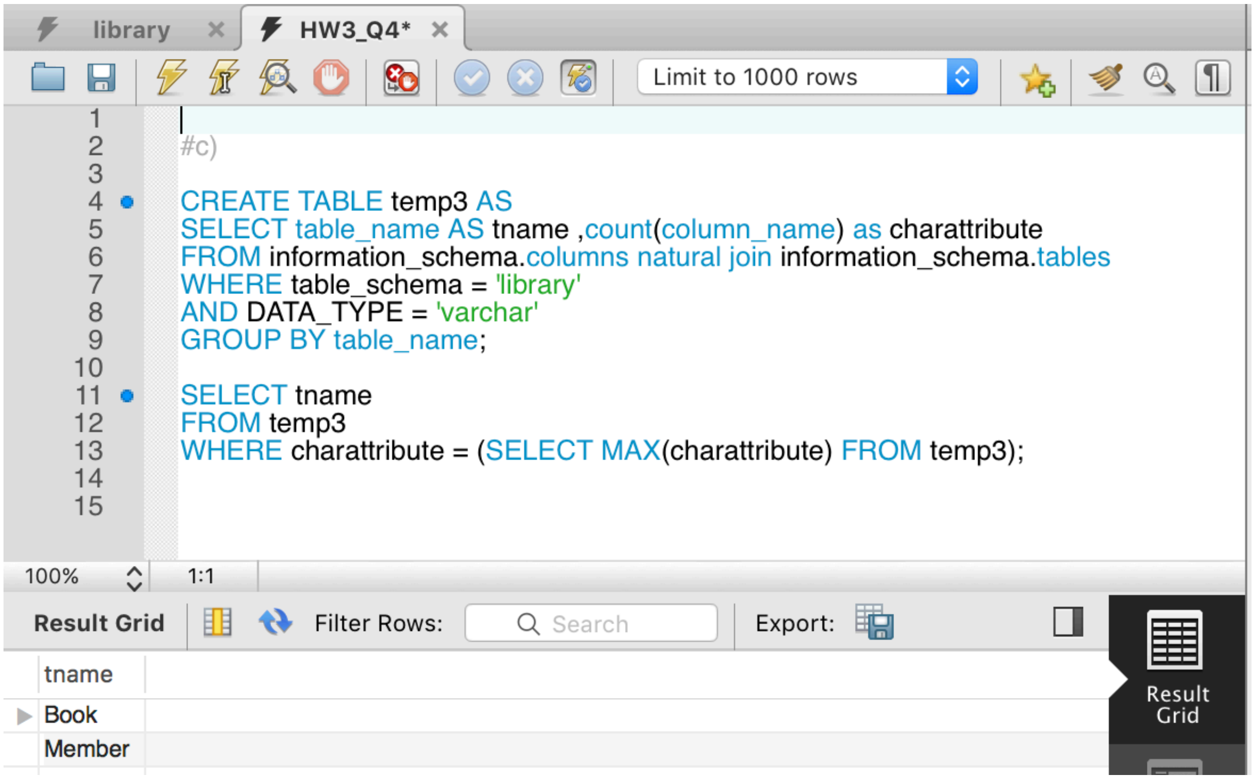
b.



```



#b)

CREATE TABLE temp AS
SELECT table_name as tname,count(column_name) as dtattribute
FROM information_schema.columns natural join information_schema.tables
WHERE table_schema = 'library'
AND DATA_TYPE = 'datetime'
GROUP BY tname;

SELECT tname
FROM temp
WHERE dtattribute > 1;

```

tname
CheckedOut

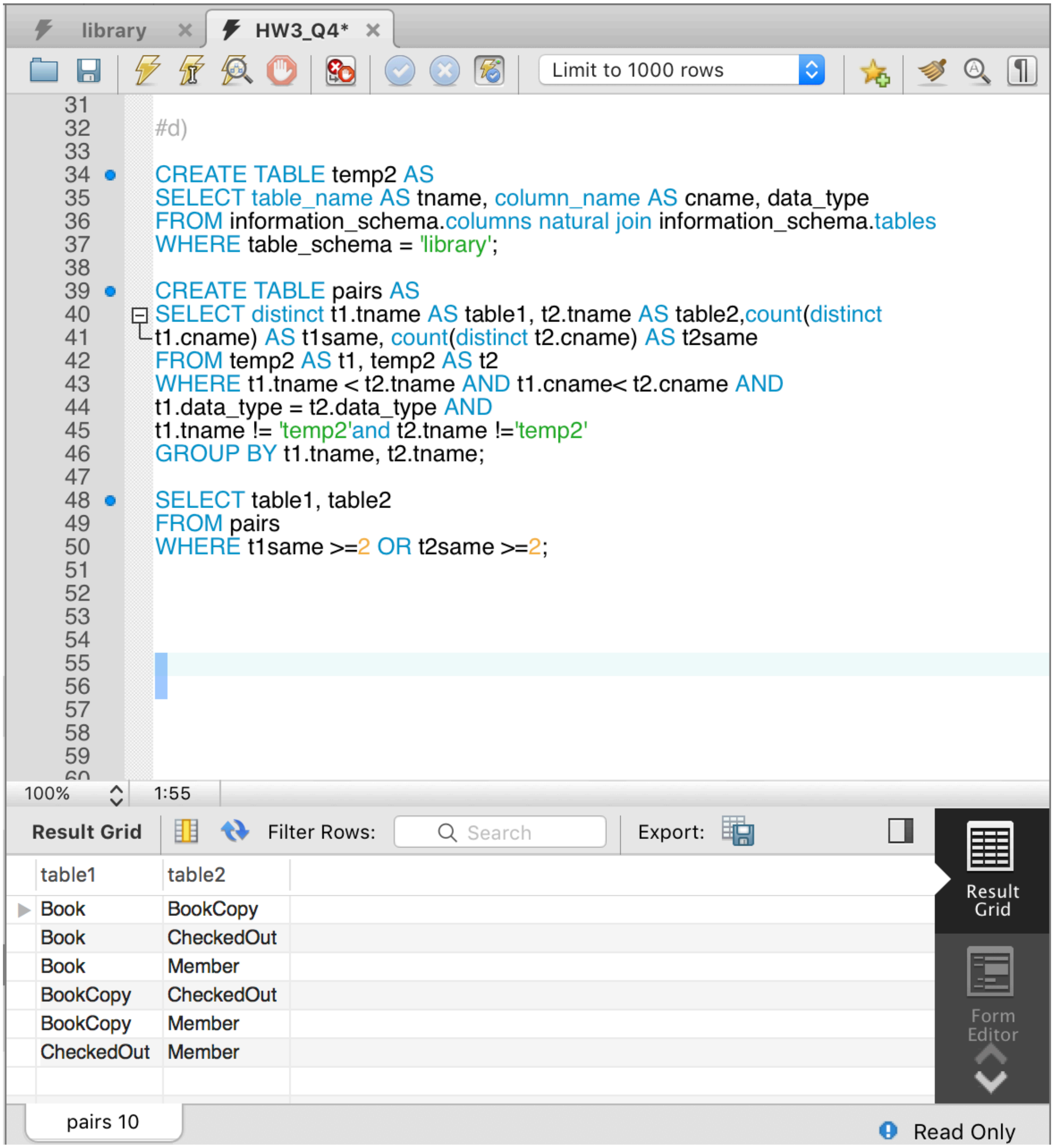
c.



```

#c)

CREATE TABLE temp3 AS
SELECT table_name AS tname ,count(column_name) as charattribute
FROM information_schema.columns natural join information_schema.tables
WHERE table_schema = 'library'
AND DATA_TYPE = 'varchar'
GROUP BY table_name;

SELECT tname
FROM temp3
WHERE charattribute = (SELECT MAX(charattribute) FROM temp3);
```

tname
Book
Member

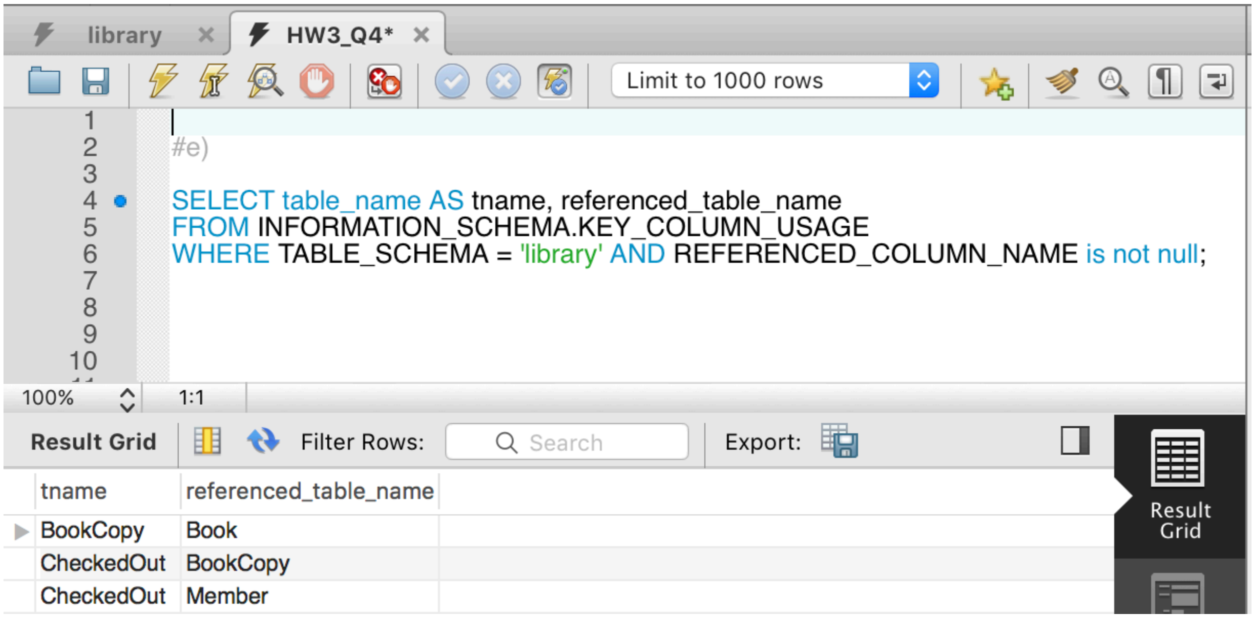
d.



```
#d)

CREATE TABLE temp2 AS
SELECT table_name AS tname, column_name AS cname, data_type
FROM information_schema.columns natural join information_schema.tables
WHERE table_schema = 'library';

CREATE TABLE pairs AS
SELECT distinct t1.tname AS table1, t2.tname AS table2,count(distinct
t1.cname) AS t1same, count(distinct t2.cname) AS t2same
FROM temp2 AS t1, temp2 AS t2
WHERE t1.tname < t2.tname AND t1.cname< t2.cname AND
t1.data_type = t2.data_type AND
t1.tname != 'temp2'and t2.tname !='temp2'
GROUP BY t1.tname, t2.tname;

SELECT table1, table2
FROM pairs
WHERE t1same >=2 OR t2same >=2;
```

table1	table2
Book	BookCopy
Book	CheckedOut
Book	Member
BookCopy	CheckedOut
BookCopy	Member
CheckedOut	Member

e.



Limit to 1000 rows

```
#e)

SELECT table_name AS tname, referenced_table_name
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE TABLE_SCHEMA = 'library' AND REFERENCED_COLUMN_NAME is not null;
```

tname	referenced_table_name
BookCopy	Book
CheckedOut	BookCopy
CheckedOut	Member