Gretel Rajamoney
rajamong@oregonstate.edu
933188305

Kaavya Subramanian
subramka@oregonstate.edu
933291513

Tables for Problem 1

Orders

oid	title	price
Ord1	“Big Data”	30
Ord2	“SQL”	35
Ord3	“Logic”	50

Pay

cid	orderName
c1	Ord1
c2	-
c3	Ord3
c4	Ord4

Customer

cid	Name
c1	“john”
c2	“mary”
c4	“emily”
-	“mark”

1. Incomplete Information

   a. An example query is one that looks for Orders that don't have a customer: **SELECT O.title FROM orders O WHERE NOT EXISTS (SELECT * FROM customer C, pay P WHERE C.cid = P.cid AND P.orderName = O.oid);** . This query will return "Logic", even though the answer is none. All orders have a customer associated with them, even though we don't know all their names.

   b. Outer joins can return false positives or false negatives, such as an left outer join. For example, if we apply **SELECT pay.cid FROM pay LEFT JOIN orders ON pay.orderName = orders.oid;** we get c1,c2,c3,c4**.**We are looking for customers who have orders and we get c2 back even though c2 does not have an associated order.

   c. A query that gives us a false negative is one where we check customers haven't paid for an order: S**ELECT C.Name from customer C WHERE C.cid NOT IN (SELECT cid FROM pay);** This returns nothing, even though Mark has not paid for the order. We can get that answer back with this transformation: **SELECT C.Name from customer C WHERE C.cid IS NULL OR C.cid IS NOT NULL AND C.cid NOT IN (SELECT cid FROM pay);**

   d.The biggest subset of SQL queries that would not return false positives would be those that do not contain negations, set differences, subqueries, or inequalities. This is because this subset of queries will not cause comparisons with null values, and do not need to support their three-way logic (use IS NULL or IS NOT NULL to guarantee a correct answer).

2. Constraint Inference
   A. Provided with the relation R(X, Y, W, Z), with functional dependencies of X->Y and YW-> Z. Through utilizing the pseudo-transitive rule of Armstrong's axioms, if X->Y and YW->Z then XW->Z.
   B. For any given attribute in the relation "R", using the union of Armstrong's axioms, if X->Y and X->Z then X->YZ.
3. Schema Decomposition

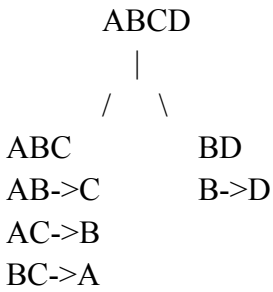
   A.
   A+ = A
   B+ = BD
   C+ = C
   D+ = D
   AB+ = ABCD
   AC+ = ACBD
   AD+ = AD
   BC+ = BDCA
   BD+ = BD

CD+ = CD
ABD+ = ABCD
BCD+ = BCAD

Therefore, our keys for R are: AB, AC, BC, ABD, BCD

B.

X must be a super key in all functional dependencies.
When B->D, B is not considered to be a super key.



Since all X's are considered super keys, R(ABC) and R(BD) are in BCNF.

C. It is considered to be in 3NF as for all functional dependencies X->Y to be in 3NF, X is either a super key or Y is a prime attribute. A relation is also considered to be in 3NF form if there is no transitive dependency for non-prime attributes. In all provided functional dependencies, Y are considered to be prime attributes, therefore R is in 3NF.

D. A relation is in BCNF if and only if it is also in 3NF and for any dependency A->B, A should be a superkey. If there is only one key in relation R, then both A and B will be part of the superkey. If A is a superkey, then the only other condition required for R to be in BCNF is that it must also be in 3NF.

4. Information Preservation

A. The decomposition is dependency preserving as S1 maintains the dependency B-> C and S2 maintains the dependency D->A. It is not lossless because the intersection of S1 and S2 is null, which means that if the two sets were naturally joined, it would not look like the original relation R.