CS 145 – PSET 2 – Sample Answers – 2016

NOTE: there are more possible answers than those listed here

1a

```
SELECT *
FROM   R r1,
       R r2
WHERE (r1.A = r2.A AND r1.B = r2.B) <> (r1.C = r2.C);
```

```
SELECT *
FROM   R r1,
       R r2
WHERE  ((r1.A = r2.A AND r1.B = r2.B) AND (r1.C != r2.C))
        OR ((r1.C = r2.C) AND (r1.A <> r2.A OR r1.B <> r2.B));
```

```
SELECT *
FROM   R r1,
       R r2
WHERE (r1.A = r2.A AND r1.B = r2.B) <> (r1.C = r2.C);
```

1b

```
SELECT *
FROM   R r1,
       R r2
WHERE (r1.A = r2.A AND r1.B = r2.B AND r1.C = r2.C)
      AND (r1.D <> r2.D OR r1.E <> r2.E);
```

```
SELECT *
FROM   R r1,
       R r2
WHERE (r1.A = r2.A AND r1.B = r2.B AND r1.C = r2.C)
      AND (r1.D <> r2.D OR r1.E <> r2.E);
```

```
SELECT *
FROM   R r1,
       R r2
WHERE r1.A = r2.A
      AND r1.B = r2.B
      AND r1.C = r2.C
      AND (r1.D <> r2.D OR r1.E <> r2.E);
```

1c

```
SELECT *
FROM   S s1,
       S s2
WHERE (s1.A = s2.A AND (s1.B <> s2.B OR s1.C <> s2.C))
      OR (s1.B = s2.B AND (s1.A <> s2.A OR s1.C <> s2.C));
```

```
SELECT *
FROM   S s1,
       S s2
WHERE (s1.A = s2.A AND (s1.B <> s2.B OR s1.C <> s2.C))
      OR (s1.B = s2.B AND (s1.A <> s2.A OR s1.C <> s2.C));
```

```
SELECT *
FROM   S s1,
       S s2
WHERE (s1.A = s2.A AND (s1.B <> s2.B OR s1.C <> s2.C))
      OR (s1.B = s2.B AND (s1.A <> s2.A OR s1.C <> s2.C));
```

1d

```
SELECT *
FROM R t1, R t2
WHERE t1.A = t2.A AND
```

	NOT EXISTS (SELECT * FROM R t3 WHERE t3.A = t1.A AND t3.C = t1.C AND t3.E = t1.E AND t3.B = t2.B AND t3.D = t2.D);
	SELECT * FROM R t1, R t2 WHERE r1.A = r2.A AND NOT EXISTS (SELECT * FROM R r3 WHERE r3.A = r1.A AND r3.C = r1.C AND r3.E = r1.E AND r3.B = r2.B AND r3.D = r2.D);
	FROM R t1, R t2 WHERE r1.A = r2.A AND NOT EXISTS (SELECT * FROM R r3 WHERE r3.A = r1.A AND r3.C = r1.C AND r3.E = r1.E AND r3.B = r2.B AND r3.D = r2.D);
2a	answer = True explanation = \ """ Consider the original set of functional dependencies $f_1,f_2,\ldots,f_n$. The assumption that K is a superkey implies that the original set of FDs implies the FD $K \rightarrow {A}$ for any attribute A. Adding an additional functional dependency cannot cause any of the inferred conditions to become false, so all of the conditions for K to be a superkey will still hold. """
	answer = True explanation = "The addition of new functional dependencies does not remove any elements from the closure of the set of attributes that comprise K. Thus, K is still a superkey."
	answer = True explanation = "superkeys are not unique or minimal, any FD could only be more restrictive so a current superkey will still be valid"
2b	answer = False X = "X" Y = "Y" Z = "Z" K = set((X, Y)) FDs = [(set((X, Y)), set(Z))] new_FD = (set(X), set(Y))
	answer = False K = set(("X", "Y")) FDs = [(set(("X", "Y")), set("Z"))] new_FD = (set("X"), set("Y"))

	answer = False K = set(("X", "Y")) FDs = [(set(("X", "Y")), set("Z"))] new_FD = (set("X"), set("Y"))
2c	answer = False X1 = "A" X2 = "B" X3 = "C" X4 = "D" K = set((X1, X2)) FDs = [(set((X1, X2)), set((X3, X4))), (set((X3, X4)), set(X2)), (set(X1), set(X3))] new_FD = (set(X3), set(X4))
	answer = False K = set(("X", "Y")) FDs = [(set(("X", "Y")), set(("Z", "A"))), (set("X"), set("Z")), (set(("Z", "A")), set(("X", "Y"))] new_FD = (set("Z"), set("A"))
	answer = False K = set(("X", "Y")) FDs = [(set(("X", "Y")), set(("W", "Z"))), (set(("W", "Z")), set("Y")), (set("X"), set("W"))] new_FD = (set("W"), set("Z"))
3a	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int); INSERT INTO T VALUES (0, 0, 0, 0); INSERT INTO T VALUES (0, 1, 1, 1); INSERT INTO T VALUES (1, 0, 2, 2); INSERT INTO T VALUES (1, 1, 3, 3); INSERT INTO T VALUES (1, 2, 4, 5);
	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int); INSERT INTO T VALUES (0, 0, 0, 0); INSERT INTO T VALUES (0, 1, 1, 1); INSERT INTO T VALUES (0, 2, 2, 2); INSERT INTO T VALUES (1, 1, 3, 3);
	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int); INSERT INTO T VALUES (0, 0, 0, 0); INSERT INTO T VALUES (0, 1, 1, 1); INSERT INTO T VALUES (1, 0, 2, 2);
3b	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int); INSERT INTO T VALUES (0, 0, 0, 0);

	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int); INSERT INTO T VALUES (0, 0, 0, 0);
	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int); INSERT INTO T VALUES (0, 0, 0, 0);
3c	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int);
	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int);
	DROP TABLE IF EXISTS T; CREATE TABLE T(A int, B int, C int, D int);
4a	DROP TABLE IF EXISTS R; CREATE TABLE R (A int, B int, C int, D int); INSERT INTO R VALUES(0, 0, 0, 0); -- Violate {A, B, C} => {D} INSERT INTO R VALUES(0, 0, 0, 1); -- Nothing to violate for {A, B, D} -- Nothing to violate for {A, C, D} -- Violate {B, C, D} => {A} INSERT INTO R VALUES(2, 0, 0, 0); -- Violate {A, B} => {C, D} INSERT INTO R VALUES(0, 0, 3, 3); -- Violate {A, C} => {D} -- (already violated for {A, B, C} => {D}) -- Nothing to violate for {A, D} -- Violate {B, C} => {A, D} INSERT INTO R VALUES(4, 0, 0, 4); -- Violate {B, D} => {A} -- (Already violated by {B, C, D} => {A}) -- Violate {C, D} => {A, B} INSERT INTO R VALUES(5, 5, 0, 0); -- Violate {A} => {B, C, D} INSERT INTO R VALUES(0, 6, 6, 6); -- Violate {B} => {A, C, D} INSERT INTO R VALUES(7, 0, 7, 7); -- Violate {C} => {A, B, D} INSERT INTO R VALUES(8, 8, 0, 8); -- Violate {D} => {A, B} -- (Already violated by {C, D} => {A, B})
	DROP TABLE IF EXISTS R; CREATE TABLE R (A int, B int, C int, D int);

```
INSERT INTO R VALUES(0, 0, 0, 0);

-- Want to make sure the following functional dependencies U -> V do not hold

-- U = {A, B, C}, V = {D}
INSERT INTO R VALUES(0, 0, 0, 1);

-- U = {B, C, D}, V = {A}
INSERT INTO R VALUES(2, 0, 0, 0);

-- U = {A, B}, V = {C, D}
INSERT INTO R VALUES(0, 0, 3, 3);
-- This example also violates the FDs where U = {A, B}, V = {C} or V = {D}

-- U = {A, C}, V = {D}
INSERT INTO R VALUES(0, 0, 0, 4);

-- U = {B, C}, V = {A, D}
INSERT INTO R VALUES(5, 0, 0, 5);
-- This example also violates the FDs where U = {B, C}, V = {A} or V = {D}

-- U = {B, D}, V = {A}
INSERT INTO R VALUES(6, 0, 0, 0);

-- U = {C, D}, V = {A, B}
INSERT INTO R VALUES(7, 7, 0, 0);
-- This example also violates the FDs where U = {C, D}, V = {A} or V = {B}

-- U = {A}, V => {B, C, D}
INSERT INTO R VALUES(0, 8, 8, 8);

-- U = {B}, V => {A, C, D}
INSERT INTO R VALUES(9, 0, 9, 9);

-- U = {C}, V => {A, B, D}
INSERT INTO R VALUES(10, 10, 0, 10);

-- U = {D}, V => {A, B}
INSERT INTO R VALUES(11, 11, 0, 0);
```

```
INSERT INTO R
VALUES (0, 0, 0, 0);
INSERT INTO R
VALUES (0, 0, 0, 1);
INSERT INTO R
VALUES (1, 0, 0, 0);
INSERT INTO R
VALUES (0, 0, 2, 2);
INSERT INTO R
VALUES (1, 0, 0, 1);
INSERT INTO R
VALUES (4, 0, 0, 4);
INSERT INTO R
VALUES (5, 5, 0, 1);
INSERT INTO R
VALUES (1, 6, 1, 6);
```

4b

```
DROP TABLE IF EXISTS T;
CREATE TABLE S(A int, B int, C int, D int);

INSERT INTO S SELECT *, 0 FROM R);
INSERT INTO S SELECT *, 1 FROM R);
```

	DROP TABLE IF EXISTS T; CREATE TABLE S(A int, B int, C int, D int); INSERT INTO S SELECT *, 0 FROM R); INSERT INTO S SELECT *, 12 FROM R);
	DROP TABLE IF EXISTS T; CREATE TABLE S(A int, B int, C int, D int); INSERT INTO S SELECT *, 1 FROM R); INSERT INTO S SELECT *, 2 FROM R);