<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 1a | SELECT *  
|    | FROM R r1,  
|    |    R r2  
|    | SELECT *  
|    | FROM R r1,  
|    |    R r2  
|    | SELECT *  
|    | FROM R r1,  
|    |    R r2  
| 1b | SELECT *  
|    | FROM R r1,  
|    |    R r2  
|    | AND (r1.D <> r2.D OR r1.E <> r2.E);  |
|    | SELECT *  
|    | FROM R r1,  
|    |    R r2  
|    | 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  |
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.

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.

Superkeys are not unique or minimal, any FD could only be more restrictive so a current superkey will still be valid.
answer = False
K = set(('X', 'Y'))
FDs = [(set(('X', 'Y')), set('Z'))]
new_FD = (set('X'), set('Y'))

answer = False
X1 = "A"
X2 = "B"
X3 = "C"
X4 = "D"
K = set((X1, X2))
FDs = [(set((X1, X2)), set((X3, X4)))
new_FD = (set(X3), set(X4))

answer = False
K = set(('X', 'Y'))
FDs = [
    (set(('X', 'Y')), set(('Z', 'A'))),
    (set('Y'), set('Z'))
]
new_FD = (set('Z'), set('A'))

answer = False
K = set(('X', 'Y'))
FDs = [
    (set(('X', 'Y')), set(('W', 'Z'))),
    (set('W'), set('Z'))
]
new_FD = (set('W'), set('Z'))

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

INSERT INTO T VALUES (0, 0, 0, 0):
<table>
<thead>
<tr>
<th>3c</th>
</tr>
</thead>
<tbody>
<tr>
<td>DROP TABLE IF EXISTS T;</td>
</tr>
<tr>
<td>CREATE TABLE T(A int, B int, C int, D int);</td>
</tr>
<tr>
<td>INSERT INTO T VALUES (0, 0, 0, 0);</td>
</tr>
<tr>
<td>DROP TABLE IF EXISTS T;</td>
</tr>
<tr>
<td>CREATE TABLE T(A int, B int, C int, D int);</td>
</tr>
<tr>
<td>INSERT INTO T VALUES (0, 0, 0, 0);</td>
</tr>
<tr>
<td>DROP TABLE IF EXISTS T;</td>
</tr>
<tr>
<td>CREATE TABLE T(A int, B int, C int, D int);</td>
</tr>
<tr>
<td>DROP TABLE IF EXISTS T;</td>
</tr>
<tr>
<td>CREATE TABLE T(A int, B int, C int, D int);</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4a</th>
</tr>
</thead>
<tbody>
<tr>
<td>DROP TABLE IF EXISTS R;</td>
</tr>
<tr>
<td>CREATE TABLE R (A int, B int, C int, D int);</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(0, 0, 0, 0);</td>
</tr>
<tr>
<td>-- Violate {A, B, C} =&gt; {D}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(0, 0, 0, 1);</td>
</tr>
<tr>
<td>-- Nothing to violate for {A, B}</td>
</tr>
<tr>
<td>-- Nothing to violate for {A, C}</td>
</tr>
<tr>
<td>-- Violate {B, C, D} =&gt; {A}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(2, 0, 0, 0);</td>
</tr>
<tr>
<td>-- Violate {A, B} =&gt; {C, D}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(0, 0, 3, 3);</td>
</tr>
<tr>
<td>-- Violate {A, C} =&gt; {D}</td>
</tr>
<tr>
<td>-- (already violated for {A, B, C} =&gt; {D})</td>
</tr>
<tr>
<td>-- Nothing to violate for {A, D}</td>
</tr>
<tr>
<td>-- Violate {B, C} =&gt; {A, D}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(4, 0, 0, 4);</td>
</tr>
<tr>
<td>-- Violate {B, D} =&gt; {A}</td>
</tr>
<tr>
<td>-- (Already violated by {B, C, D} =&gt; {A})</td>
</tr>
<tr>
<td>-- Violate {C, D} =&gt; {A, B}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(5, 5, 0, 0);</td>
</tr>
<tr>
<td>-- Violate {A} =&gt; {B, C, D}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(0, 6, 6, 6);</td>
</tr>
<tr>
<td>-- Violate {B} =&gt; {A, C, D}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(7, 0, 7, 7);</td>
</tr>
<tr>
<td>-- Violate {C} =&gt; {A, B, D}</td>
</tr>
<tr>
<td>INSERT INTO R VALUES(8, 8, 0, 8);</td>
</tr>
<tr>
<td>-- Violate {D} =&gt; {A, B}</td>
</tr>
<tr>
<td>-- (Already violated by {C, D} =&gt; {A, B})</td>
</tr>
</tbody>
</table>

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);

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);