1. 如果分解是一個 lossless join decomposition，則矩陣 S 中應至少有一列是全由"a"組成的。

舉例說明：

X	Y	Z
x1	y1	z1
x1	y2	z2

R={X, Y, Z}

R1={X, Y}

R2={X, Z}

F={Y→X, Z→X}

Matrix S:

X	Y	Z
a1	a2	b13
a1	b22	a3

上述的分解得出的 matrix S 不包含整列都為 a 的列

因此上述的分解方法不通過 **Testing for Lossless Join Property**

以下透過重組展示：

R1：

X	Y	Z
x1	y1	-
x1	y2	-

R2：

X	Y	Z
x1	-	z1
x1	-	z2

通過自然連接重組 R1 跟 R2：

X	Y	Z
x1	y1	z1
x1	y1	z2
x1	y2	z1
x1	y2	z2

其中第二列及第三列（灰色列）及為產生的 Spurious Tuples

2.

(1) 是，joined views 可能存在部分依賴也可能存在傳遞依賴。因為當 base tables 被連接形成一個 view 時，新的屬性之間可能引入依賴關係，而這些依賴關係在原先的 base tables 內部可能並不存在。

(2) 是，由於 view 是多個表的聯接，insert / delete / update 需要確保 data 能夠正確地分配到各個 base tables 中。例如 view 中的某些欄位可能對應多個 base tables，這會導致 data 不一致。

解決方法：

- 將 insert / delete / update 分解為對 base tables 的操作，確保所有 base tables 中的 data 保持一致
- 以 insert 為例：

```
-- view中的insert操作
INSERT INTO view_name (StudentID, Name, ClassID, ClassName)
VALUES (1, '張三', 101, '一班');

-- 分解為base tables的insert操作
INSERT INTO Students (StudentID, Name, ClassID) VALUES (1, '張三', 101);
INSERT INTO Classes (ClassID, ClassName) VALUES (101, '一班');
```

3.

(1) 尋找部分依賴的算法

I. 第一正規化包含 primary_key，primary_key 可能有多個 attribute 組成。

II. 初始化一個空列表 partial_dependencies 來存儲部分依賴關係。

III. 獲取表格中非主鍵屬性的列表 non_primary_key_attributes。

IV. 生成主鍵的所有子集 primary_key_parts。

V. 對每個主鍵子集計算其閉包 closure。

VI. 對每個閉包中的屬性進行檢查，如果該屬性是非主鍵屬性且不在當前子集中，則將其作為部分依賴添加到 partial_dependencies 列表中。

VII. 返回 partial_dependencies 列表。

```
FUNCTION Find_Partial_Dependencies(table, primary_key)
    partial_dependencies = []
    non_primary_key_attributes = Get_Non_Primary_Key_Attributes(table, primary_key)
    primary_key_parts = Generate_Subsets(primary_key)

    FOR subset in primary_key_parts DO
        closure = Compute_Closure(subset, table)
        FOR attribute in closure DO
            IF attribute in non_primary_key_attributes and attribute not in subset then
                partial_dependencies.append((subset, attribute))
            END IF
        END FOR
    END FOR
    RETURN partial_dependencies
END FUNCTION
```

(2) 尋找傳遞依賴的算法

I. 初始化一個空列表 transitive_dependencies 來存儲傳遞依賴關係。

II. 獲取表格中的所有屬性列表 attributes。

III. 生成所有屬性的子集 subsets。

IV. 對每個子集計算其閉包 closure。

V. 對每個閉包中的中間屬性 intermediate 進行檢查，如果該屬性不在當前子集中，則計算該屬性的閉包 intermediate_closure。

VI. 對每個中間閉包中的屬性進行檢查，如果該屬性不在當前子集中但在初始閉包中，則為傳遞依賴添加到 transitive_dependencies。

VII. 返回 transitive_dependencies 列表。

```
FUNCTION Find_Transitive_Dependencies(table)
    transitive_dependencies = []
    attributes = Get_All_Attributes(table)

    FOR subset in Generate_Subsets(attributes) DO
        closure = Compute_Closure(subset, table)
        FOR intermediate in closure DO
            IF intermediate not in subset then
                intermediate_closure = Compute_Closure([intermediate], table)
                FOR attribute in intermediate_closure DO
                    IF attribute not in subset and attribute in closure then
                        transitive_dependencies.append((subset, intermediate, attribute))
                    END IF
                END FOR
            END IF
        END FOR
    END FOR

    RETURN transitive_dependencies
END FUNCTION
```

4.

I. 前提假設

(a.) 每個站點具有相同的計算和通訊能力。
(b.) 不同站點之間的資料傳輸速度比在同一站點內慢約 50 倍。
(c.) 需要強制執行多個函數依賴和引用約束，例如 CASCADE 操作。

II. 策略

(a.) 將具有函數依賴關係和引用約束的表格分組放在同一站點。
- 舉例說明：如果表 A 和表 B 經常在查詢中一起使用，並且表 B 引用了表 A，則將它們放在同一站點，避免站點之間的傳輸時間成本。

(b.) 對於讀取頻繁但更新不頻繁的表格複製到多個站點。減少不同站點之間的傳輸成本。
- 舉例說明：產品信息表可能讀取頻繁，但較少更新，因此將其複製到多個站點以加快查詢速度。

(c.) 確保涉及引用約束的表格（尤其是存在 CASCADE 操作的表格）位於同一站點，以便在本地強制執行約束。
- 舉例說明：當 A 表和 B 表之間存在 ON DELETE CASCADE 操作時，將它們放在同一站點可以確保操作的快速執行。

(d.) 使用 Two-phase Commit 來確保跨站點操作時的一致性。
- 舉例說明：當需要跨多個站點進行複雜的操作時，Two-phase Commit 確保所有相關操作要麼全部完成，要麼全部 rollback。

III. 實例

(a.) 假設我們有一個學校系統需要管理以下數據：
- 學生表（學生 ID、姓名、班級 ID）。
- 成績表（學生 ID、課程 ID、成績）。
- 班級表（班級 ID、班級名稱、年級、導師 ID）。
- 課程表（課程 ID、課程名稱、授課教師）。

(b.) 函數依賴：
- 班級表（班級 ID -> 班級名稱, 年級, 導師 ID）
- 學生表（學生 ID -> 姓名, 班級 ID）
- 成績表（學生 ID, 課程 ID -> 成績）
- 課程表（課程 ID -> 課程名稱, 授課教師）

(c.) 引用約束：

- 學生表中的班級 ID 參考班級表中的班級 ID（CASCADE DELETE）

  當刪除某個班級時，該班級的所有學生記錄也會自動刪除。

- 成績表中的學生 ID 參考學生表中的學生 ID（CASCADE DELETE）

  當刪除某個學生時，該學生的所有成績記錄也會自動刪除。

(d.) 應用策略：

- 站點 1：班級表、學生表、課程表

  班級表與學生表引用約束，因此放在同一站點，課程表複製到每個站點以提高查詢速度。

- 站點 2：成績表、學生表、課程表

  成績表與學生表引用約束，因此放在同一站點，課程表複製到每個站點以提高查詢速度。