1. Question 1 – Identifying Functional Dependencies

* Identify Simple Dependencies
  + Determining if one attribute determines another attribute.
  + Examples:
    - `pname` ⟶ `price`
      * SELECT pname, COUNT(DISTINCT price)

FROM MySales

GROUP BY pname

HAVING COUNT(DISTINCT price) > 1;

* + - * + The result was empty, this would mean `pname` uniquely determines `price`
    - `month` ⟶ `discount`
      * SELECT month, COUNT(DISTINCT discount)

FROM MySales

GROUP BY month

HAVING COUNT(DISTINCT discount) > 1;

* + - * + The result is empty meaning `month` uniquely determines `discount`
    - `price` ⟶ `pname`
      * SELECT price, COUNT(DISTINCT pname)

FROM MySales

GROUP BY price

HAVING COUNT(DISTINCT pname) > 1;

* + - * + The result of the query was NOT empty meaning that `price` does not uniquely determine `pname`
    - `discount` ⟶ `month`
      * SELECT discount, COUNT(DISTINCT month)

FROM MySales

GROUP BY discount

HAVING COUNT(DISTINCT month) > 1;

* + - * + The result of the query NOT empty meaning that `discount` does not uniquely determine `month`
* Identifying Complex Dependencies
  + Check if combinations of attributes determine other attributes.
  + Examples:
    - (`pname, month`) ⟶ `price`
      * SELECT pname, month, COUNT(DISTINCT price)

FROM MySales

GROUP BY pname, month

HAVING COUNT(DISTINCT price) > 1;

* + - * + The result is empty, so `(name, month)` uniquely determines `price`
    - (`pname, discount`) ⟶ `month`
      * SELECT pname, discount, COUNT(DISTINCT month)

FROM MySales

GROUP BY pname, discount

HAVING COUNT(DISTINCT month) > 1;

* + - * + The result is Not empty, this means that `(pname, discount)` does not uniquely determine `month`
    - (`month, price`) ⟶ `pname`
      * SELECT month, price, COUNT(DISTINCT pname)

FROM MySales

GROUP BY month, price

HAVING COUNT(DISTINCT pname) > 1;

* + - * + The result is Not empty, this means that `(month, price)` does not uniquely determine `pname`
    - (`discount, price`) ⟶ `month`
      * SELECT discount, price, COUNT(DISTINCT month)

FROM MySales

GROUP BY discount, price

HAVING COUNT(DISTINCT month) > 1;

* + - * + The result is Not empty, this means that `(discount, price)` does not uniquely determine `month`

Decompose the Table to BCNF Relations

Functional Dependencies Identified:

* `pname`⟶ `price`
* `month`⟶ `discount`
* `(pname, month)`⟶ `price`

1. Decompose the Table to BCNF Relations
   1. Identifying Keys and Candidate Keys:
      1. `pname, month` is a composite key
   2. Decompose `MySales`:
      1. Goal, crate separate tables removing partial dependencies so each relation is in BCNF
   3. Deposition plan
      1. Table 1 - `ProductPrices` to hold the relationship between `pname` and `price`
      2. Table 2 - `MonthlyDiscounts` to hold the relationship between `month` and `discount`
      3. Table 3 - `Sales` to hold the main sales data with foreign keys referencing other tables.
         1. SQL Commands for Creating the necessary Tables:
            1. -- Table for Product Prices

CREATE TABLE ProductPrices (

pname VARCHAR(50) PRIMARY KEY,

price NUMERIC

);

-- Table for Monthly Discounts

CREATE TABLE MonthlyDiscounts (

month VARCHAR(20) PRIMARY KEY,

discount NUMERIC

);

-- Main Sales Table

CREATE TABLE Sales (

pname VARCHAR(50),

month VARCHAR(20),

discount NUMERIC,

price NUMERIC,

PRIMARY KEY (pname, month),

FOREIGN KEY (pname) REFERENCES ProductPrices(pname),

FOREIGN KEY (month) REFERENCES MonthlyDiscounts(month)

);

* + - * INSERT INTO ProductPrices (pname, price)

SELECT DISTINCT pname, price FROM MySales;

* + - * INSERT INTO MonthlyDiscounts (month, discount)

SELECT DISTINCT month, discount FROM MySales;

* + - * INSERT INTO Sales (pname, month, discount, price)

SELECT pname, month, discount, price FROM MySales;

* + - * SELECT COUNT(\*) FROM ProductPrices; - 36

SELECT COUNT(\*) FROM MonthlyDiscounts; - 12

SELECT COUNT(\*) FROM Sales; - 426

Question 2 – BCNF Decomposition

1. Relation `R(A,B,C,D,E,F)` with FDs `A`⟶ `BC`, D⟶ `AF`
2. Identify the minimal key(s) for the relation

* `A` determines `BC`
* `D` determines `AF`
* The candidate key would be `D` since `D` can determine `A` and `A` can determine `BC`

1. Check BCNF:

* The relation would not be BCNF because `A` is not a superkey for the dependency `A`⟶ `BC`

1. Decompose to BCNF:

* Getting `R` into two relations:
  + CREATE TABLE R1 (

A VARCHAR PRIMARY KEY,

B VARCHAR,

C VARCHAR

);

CREATE TABLE R2 (

D VARCHAR PRIMARY KEY,

A VARCHAR,

F VARCHAR,

FOREIGN KEY (A) REFERENCES R1(A)

);

1. Preservation of Functional Dependencies:

* `A`⟶ `BC` is preserved in `R1`
* `D`⟶ `AF` is preserved in `R2`

1. Relation `S(A,B,C,D)` with FDs `ABC ⟶ D, D ⟶ A`
   1. Identify the minimal key(s) for the relation
   * `ABC` is a candidate key because `ABC` determines `D`
   * `D` determines `A`
   1. Check BCNF:
   * The relation would not in BCNF because `D` is not a superkey for the dependency `D`⟶ `A`
   1. Decompose to BCNF:
   * Getting `S` into two relations:
     + CREATE TABLE S1 (

D VARCHAR PRIMARY KEY,

A VARCHAR

);

CREATE TABLE S2 (

A VARCHAR,

B VARCHAR,

C VARCHAR,

D VARCHAR,

PRIMARY KEY (A, B, C),

FOREIGN KEY (D) REFERENCES S1(D)

);

* 1. Preservation of Functional Dependencies:
  + `D`⟶ `A` is preserved in `S1`
  + `ABC`⟶ `D` is preserved in `S2`