Part 1 - English to Schema

A grocery store needs to track an inventory of products for sale. It has zero or more of each type of product for sale, and needs to track the quantity and price for each product. A product has a name and a "stock keeping unit" (SKU)

```
Product [__id__ (integer), SKU (string), name (string)]

Inventory [__productId__ (integer), quantity (integer), price (real)]
```

Consider the grocery store database from the previous problem, but with a few differences: Now we don't care about tracking quantity, but we do want to track which aisle(s) the product is to be displayed on. Sometimes a product is displayed on more than one aisle in special display racks, but the product can not have multiple display cases per aisle.

```
Product [__id__ (integer), SKU (string), name (string)]

Aisle [__id__ (integer), name (string)]

Display [__productId__ (integer), __aisleID__ (integer)]
```

A car has a make, model, year, color, and VIN (vehicle identification number). A salesperson has a name and a social security number, and is responsible for trying to sell zero or more cars. A car dealership has an inventory of cars and a set of salespeople. It needs to keep track of which car(s) each salesperson is trying to sell. More than one salesperson can be assigned to any given car, but a car does not necessarily have any salespeople assigned to it.

```
Car [__id__ (integer), VIN (string), make (string),
     model (string), year (integer), color (string)]

Salesperson [__id__ (integer), SSN (string), name (string)]

Dealership [__carId__ (integer), __salespersonId__ (integer)]
```

Part 2 - SQL Table Declarations

Library Database:

```
CREATE DATABASE Library;

USE Library;

CREATE TABLE Patrons (
    id INT AUTO_INCREMENT PRIMARY KEY,
    CardNum VARCHAR(20) UNIQUE,
    Name VARCHAR(100)
);

CREATE TABLE Phones (
    id INT AUTO_INCREMENT PRIMARY KEY,
    CardNum VARCHAR(20),
    Phone VARCHAR(8),
    FOREIGN KEY (CardNum) REFERENCES Patrons(CardNum)
);

CREATE TABLE Titles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    ISBN VARCHAR(13) UNIQUE,
    Title VARCHAR(100),
    Author VARCHAR(100)
);

CREATE TABLE Inventory (
    id INT AUTO_INCREMENT PRIMARY KEY,
    Serial INT UNIQUE,
    ISBN VARCHAR(13),
    FOREIGN KEY (ISBN) REFERENCES Titles(ISBN)
);

CREATE TABLE CheckedOut (
    CardNum VARCHAR(20),
    Serial INT,
    PRIMARY KEY (CardNum, Serial),
    FOREIGN KEY (CardNum) REFERENCES Patrons(CardNum),
    FOREIGN KEY (Serial) REFERENCES Inventory(Serial)
);
```

Part 3 - Fill in Tables

Table 1: Car

id	VIN	make	model	year	color
1	ABC123456789	Toyota	Tacoma	2008	Red
2	DEF987654321	Toyota	Tacoma	1999	Green
3	GHI456789012	Tesla	Model 3	2018	White
4	JKL321098765	Subaru	WRX	2016	Blue
5	MNO567890123	Ford	F150	2004	Red

Table 2: Salesperson

id	SSN	name
1	123-45-6789	Arnold
2	987-65-4321	Hannah
3	456-78-9012	Steve

Table 3: Dealership

carId	salespersonId
1	1
1	2
5	2
5	1
3	3

Part 4 - Keys and Superkeys

Ignore empty sets when looking at the keys and superkeys

Attribute Sets	Superkey?	Proper Subsets	Key?
{A1}	No		No
{A2}	No		No
{A3}	No		No
{A1, A2}	Yes	{A1}, {A2}	Yes
{A1, A3}	No	{A1}, {A3}	No
{A2, A3}	No	{A2}, {A3}	No
{A1, A2, A3}	Yes	{A1}, {A2}, {A3}, {A1, A2}, {A1, A3}, {A2, A3}	No

Part 5 - Abstract Reasoning

If {x} is a superkey, then any set containing x is also a superkey.

True.

Since {x} is a superkey, it guarantees that the values of x are unique for each tuple. When we add more attributes to form the set S, it does not change the uniqueness of the values of x. Therefore, S still satisfies the condition of uniquely identifying each tuple in the relation.

If {x} is a key, then any set containing x is also a key.

False.

A set of fields is a key if:

- it is a superkey
- no proper subset of its fields is a superkey

If {x} is a key, then {x, y} has a proper subset {x} is a superkey, which makes {x, y} not a key.

If {x} is a key, then {x} is also a superkey.

True.

A key itself is a superkey, its the definition.

If {x, y, z} is a superkey, then one of {x}, {y}, or {z} must also be a superkey.

False.

Consider this example:

x	y	z
1	2	3
1	2	4

Obviously we have duplicated values in {x} and {y} so those two are not superkeys.

If an entire schema consists of the set {x, y, z}, and if none of the proper subsets of {x, y, z} are keys, then {x, y, z} must be a key.

True.

If none of the proper subsets of {x, y, z} are keys, then {x, y, z} must be a key because it is the only set of attributes that uniquely identifies each tuple in the schema.