1. What are ‘Assertions’ and ‘Triggers’ in SQL? Why do you need them? Explain with examples.

Assertions refer to semantic constraints and are meant to specify constraints that fall outside the scope of built-in relational model constraints. Assertions are useful so a user can ensure certain conditions will always be met or that some conditions cannot be violated in the database.

-specify queries that violate the desired outcome!

-Use a “NOT EXISTS” clause outside of query specifies the results of this query must be empty so that the condition will always be true  
EXAMPLE: an assertion created to make sure that the employee salary is not greater than their manager's salary

Text

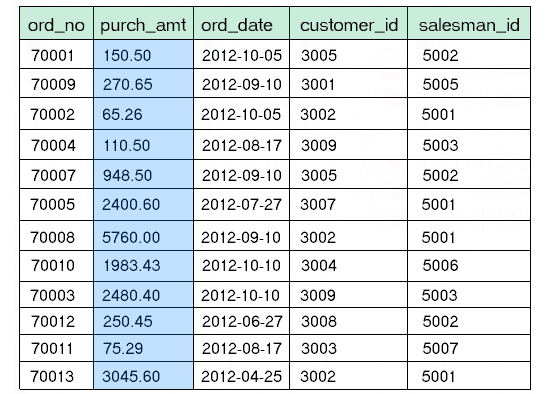
Description automatically generated

Triggers also refer to semantic constraints. They are stored procedures that are automatically executed when certain conditions are met. They monitor the database and have three important characteristics:

1. An event: usually database update operations **(insert, delete, modify)**.
2. Condition: determines whether the action should be executed or not. If it is unspecified, action will be executed as soon as the event occurs, otherwise, it will be evaluated as true or false and executed when it is true.
3. Action: sequence of SQL statements that are executed.

EXAMPLE: a trigger which in an online store when qt of an item falls below 10, the trigger alerts the admin to order more qt of the item that is low

(10 points)

(2) Consider the following **Orders** table. Write **one SQL query** that will compute the following from this table: (i) total number of orders, (ii) average purchase amount, (iii) maximum purchase amount, (iv) minimum purchase amount, and (v) total number of customers with purchase amounts greater than $1000.

1. SELECT COUNT(\*)

FROM Orders;

1. SELECT AVG(purch\_amt)

FROM Orders;

1. SELECT MAX(purch\_amt)

FROM Orders;

1. SELECT MIN(purch\_amt)

FROM Orders;

SELECT COUNT(customer\_id)

FROM Orders

WHERE purch\_amt > 1000;

(3)

(a) List the **four** informal design guidelines for relational databases.

1. Make semantics of relational attributes clear: create relations schema so it is easy to explain/understand its real-world meaning.

2. Reduce redundant information in tuples

3. Reduce NULL values in tuples: do

4. Disallow the possibility of creating **spurious(FAKE)** tuples

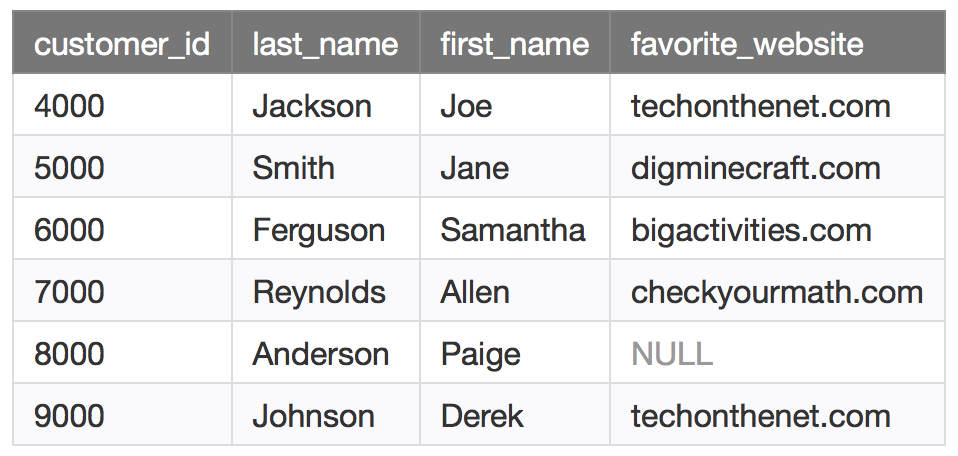
(b) Discuss the problem of **spurious(FAKE)** tuples and how you would prevent them:

Spurious tuples may be created when you create a relation schema that their matching attributes are not primary or foreign keys. This way, when performing join operations on relations, it will also output extra tuples that may not be required. We can prevent that from happening by creating relations that could be joined by matching attributes that are primary or foreign keys.

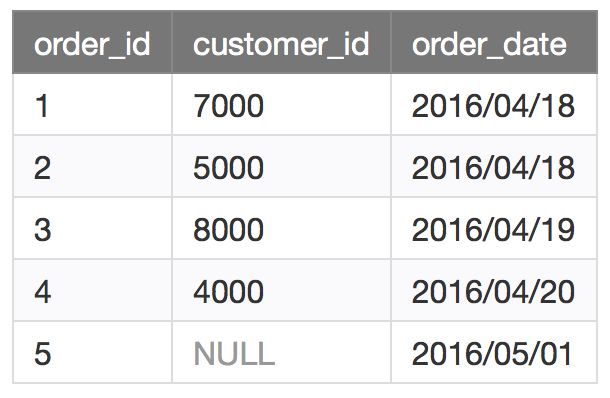
(10 points)

(4) Consider the following two relations in a database:

CUSTOMERS:



ORDERS:



Diagram

Description automatically generated

What are the **data output** generated for the following SQL retrieval commands?

(a) SELECT customers.customer\_id, orders.order\_id, orders.order\_date

FROM customers INNER JOIN orders

ON customers.customer\_id = orders.customer\_id

ORDER BY customers.customer\_id;

customer\_id order\_id order\_date

4000 4 2016/04/20

5000 2 2016/04/18

7000 1 2016/04/18

8000 3 2016/04/19

(b) SELECT customers.customer\_id, orders.order\_id, orders.order\_date

FROM customers LEFT OUTER JOIN orders

ON customers.customer\_id = orders.customer\_id

ORDER BY customers.customer\_id;

customer\_id order\_id order date

4000 4 2016/04/20

5000 2 2016/04/18

6000 NULL NULL

7000 1 2016/04/18

8000 3 2016/04/19

9000 NULL NULL

(c) SELECT customers.customer\_id, orders.order\_id, orders.order\_date

FROM customers FULL OUTER JOIN orders

ON customers.customer\_id = orders.customer\_id

ORDER BY customers.customer\_id;

customer\_id order\_id order date

NULL 5 2016/05/01

4000 4 2016/04/20

5000 2 2016/04/18

6000 NULL NULL

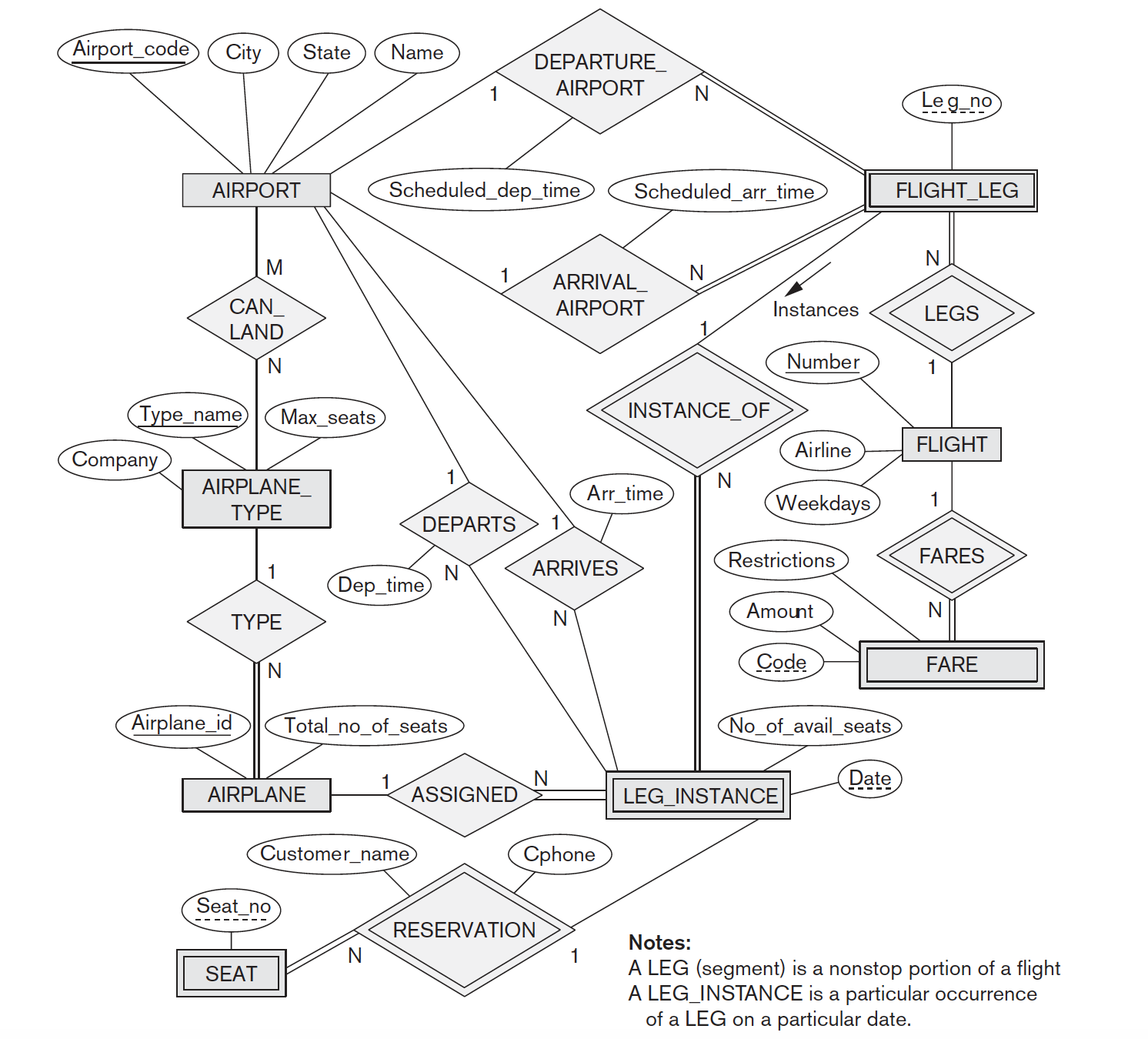
7000 1 2016/04/18

8000 3 2016/04/19

9000 NULL NULL

(5 points)

(5) Consider the ER diagram shown below, which shows a simplified schema for an airline reservations system. Extract from the ER diagram **the requirements (Entities, attributes, relationships, keys) and constraints** that produced this schema. Be as precise as possible in your requirements and constraints specification.



ENTITIES:

**AIRPORT (Entity)**

Airport\_code: PK

City: varchar

State: varchar

Name: varchar

**AIRPLANE TYPE (Entity)**  
Type\_name: PK

Company: varchar

Max\_seats: int

**AIRPLANE (Entity)**  
Airplane\_id: PK

Total\_no\_of\_seats: int

**SEAT (Weak Entity)**  
Seat\_no: FK??

**FLIGHT\_LEG (Weak Entity)**  
Leg\_no: FK???

**FLIGHT (Entity)**  
Number: PK

Airline: varchar

Weekdays: date

**FARE (Weak Entity)**  
Code: FK??

Amount: int

Restrictions: varchar

**LEG\_INSTANCE Weak Entity)**  
Date: FK???

No\_of\_avail\_seats: int

**RELATIONSHIPS:**

**DEPARTURE\_AIRPORT (1 departure airport : many flight legs)**

All FLIGHT\_LEG attributes must have total participation or in other words, each tuple in FLIGHT\_LEG must have a departure airport

Has Scheduled\_dep\_time

**ARRIVAL\_AIRPORT (1 arrival airport : many flight legs)**

All FLIGHT\_LEG attributes must have total participation or in other words, each tuple in FLIGHT\_LEG must have an arrival airport

Has Scheduled\_arr\_time **LEGS** (INDENTIFYING RELATIONSHIP, (Many flight legs : 1 flight))

**FARES** (IDENTIFYING RELATIONSHIP, (1 flight : many fares)) **INSTANCE\_OF** (IDENTIFYING RELATIONSHIP, (1 Flight leg : many leg instances)) **CAN\_LAND** (many airports : many airplane types) **TYPE** (1 airplane type : many airplanes) **DEPARTS** (1 airport : many leg instances)

Has Dep\_time **ARRIVES** (1 airport : many leg instances)

Has Arr\_time **ASSIGNED** (1 airplane : many leg instances) **RESERVATION** (1 leg instance : many seats)

Has Customer\_name, Cphone

Diagram

Description automatically generated Graphical user interface, diagram

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Text

Description automatically generated

(20 points)

(6) What are the 2 properties we need to keep in mind while doing decompositions to achieve a highest normal form for a given relation schema? Which one is more critical and why? Explain.

1. **ATTRIBUTE PRESERVATION:** also known as information loss, this occurs when we are decomposing a table for example R into R1 and R2. Once decomposed, if R1 U R2 =/ R there is evidence of failed attribute preservation. FOR EXAMPLE: if we are given a table with 3 attributes (name, price, category) \*shown below\* when trying to get a higher normal form from the table R we end up losing the information vital to determining which product is priced at what cost.

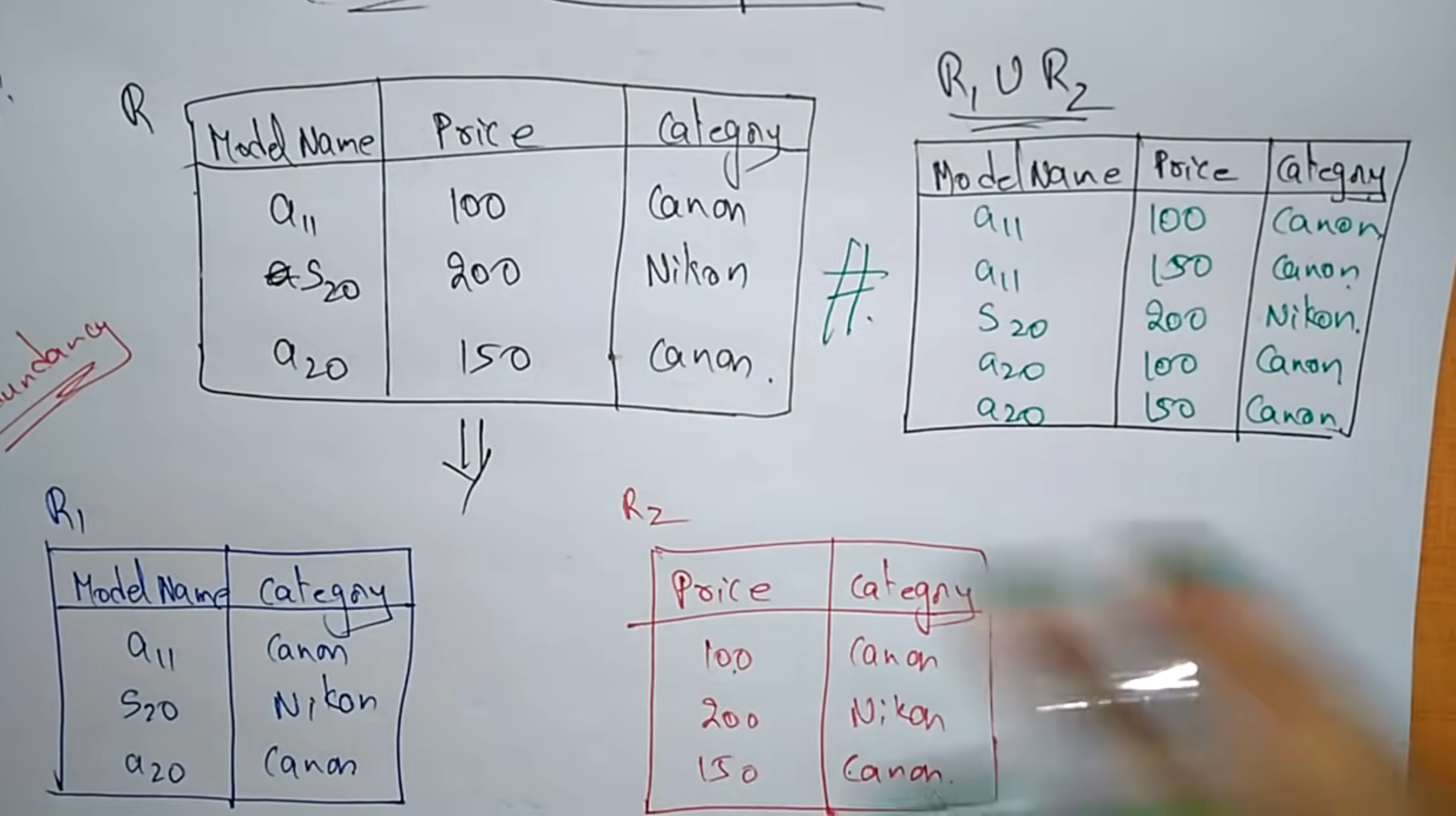
2. **DEPENDENCY PRESERVATION:**

* In the dependency preservation, at least one decomposed table must satisfy every dependency.
* If a relation R is decomposed into relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.
* For example, suppose there is a relation R (A, B, C, D) with functional dependency set (A->BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of relation R1(ABC).

ATTRIBUTE PRESERVATION

INFORMATION LOSS

R =/ R1 U R2



(10 points)

(7) Consider the following relation for published books:

**BOOK (Book\_title, Author\_name, Book\_type, List\_price, Author\_affiliation, Publisher)**

Following functional dependencies exist:

**Book\_title à Publisher, Book\_type**

**Book\_type à List\_price**

**Author\_name à Author\_affiliation**

What **normal form** is this relation in? Justify your answer.

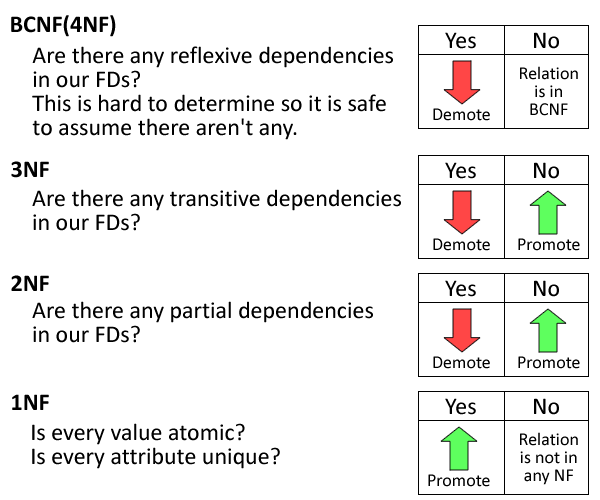
1NF Because not in 2NF since there no attributes that are fully functionally dependent on the key. Since it is not in 2NF then 3NF is also not possible.

Since list price has nothing to do with author\_name we have partial dependency

Apply normalization to decompose this relation further. Explain reasons for each decomposition. (**Hint:** you may need to do multiple decompositions to achieve the highest normal form.)

A piece of paper with writing on it

Description automatically generated



(8) You are the DBA at a Fortune 500 company in the ***financial sector***. What are the various aspects you would consider to ensure that your company’s databases are secure? Discuss all possible security challenges you may face, and explain how you would ensure that data are secure.

Aspects I would consider to ensure database is secure:

**account creation** and **user authentication** when logging in the database, **flow control**, **inference contro**l, and **transmission** and **storage of sensitive data**.

Major Database Security Issues:

– Privilege abuse

– Weak authentication

– Backup data exposure

– SQL injection

– DB platform vulnerabilities

Fixes:

– Encryption

– Levels of access control (Query, Content)

– Strong Authentication

– Firewall/IDS

– Patch management

(10 points)