Non-BCNF Relation:

CarPost

plateNumber	ownerEmail	model	bodyStyle	manufacturerName	mpg	fuel	carYear	mileage	city	carState	price	postDate	outerColor	bookmarkedBy
1PLT123	owner1@gmail.com	Prius	Hatchback	Toyota	45	hybrid	2020	160k	San Jose	CA	6500	2024/5/9	Navy	owner2@gmail.com
2PLT123	owner1@gmail.com	Camry	Sedan	Toyota	35	gas	2023	110k	San Francisc	CA	7500	2023/5/1	Black	
3PLT123	owner3@gmail.com	Prius	Hatchback	Toyota	55	gas	2024	100k	Fresno	CA	8500	2022/5/9	Blue	owner1@gmail.com
4PLT123	owner4@gmail.com	Camry	Sedan	Toyota	25	hybrid	2011	200k	Berkeley	CA	9000	2021/5/9	Grey	

BCNF Lossless Join Decomposition:

CarInfo

model	bodyStyle	manufacturerName
Prius	Hatchback	Toyota
Camry	Sedan	Toyota

Cities

city	carState
San Jose	CA
San Francisco	CA
Fresno	CA
Berkeley	CA

CarPost

plateNumber	ownerEmail	model	mpg	fuel	carYear	mileage	city	price	postDate	outerColor	bookmarkedBy
1PLT123	owner1@gmail.com	Prius	45	hybrid	2020	160k	San Jose	6500	2024/5/9	Navy	owner2@gmail.com
2PLT123	owner1@gmail.com	Camry	35	gas	2023	110k	San Francisco	7500	2023/5/1	Black	
3PLT123	owner3@gmail.com	Prius	55	gas	2024	100k	Fresno	8500	2022/5/9	Blue	owner1@gmail.com
4PLT123	owner4@gmail.com	Camry	25	hybrid	2011	200k	Berkeley	9000	2021/5/9	Grey	

Explanation for the Decomposition:

CarPost(plateNumber, ownerEmail, model, bodyStyle, manufacturerName, mpg, fuel, carYear, mileage, city, carState, price, postDate, outerColor, bookmarkedBy)

To normalize the CarPost relation into BCNF, we first identify all functional dependencies

1. All the functional dependencies:
   - FD1: plateNumber, ownerEmail -> model, mpg, fuel, carYear, mileage, city, price, postDate, outerColor, bookmarkedBy
   - FD2: model -> bodyStyle, manufacturerName
   - FD3: city -> carState
2. Check every functional dependency if they conform to normalization:
   - FD1: plateNumber, ownerEmail -> model, mpg, fuel, carYear, mileage, city, price, postDate, outerColor, bookmarkedBy **(Primary Key)/ VALID**
   - FD2: model -> bodyStyle, manufacturerName **(Transitive Dependency) AND (LHS does not contain a key) Thus, we should decompose.**
   - FD3: city -> carState **(Transitive Dependency) AND (LHS does not contain a key) Thus, we should decompose**.
   - We decompose and the following are the decomposed relations:
     - CarPost(plateNumber, ownerEmail, model, mpg, fuel, carYear, mileage, city, price, postDate, outerColor, bookmarkedBy)
     - CarInfo(model, bodyStyle, manufacturerName)
     - Cities(city, carState)
3. Check if the dependencies are preserved for BCNF:
   - R1: plateNumber, ownerEmail -> model, mpg, fuel, carYear, mileage, city, price, postDate, outerColor, bookmarkedBy
   - R2: model -> bodyStyle, manufacturerName
   - R3: city -> carState

   Now, let's check all the functional dependencies in each relation to validate they are in BCNF:

   1. R1(CarPost) FDs:

   plateNumber, ownerEmail -> model
   plateNumber, ownerEmail -> mpg
   plateNumber, ownerEmail -> fuel
   plateNumber, ownerEmail -> carYear
   plateNumber, ownerEmail -> mileage
   plateNumber, ownerEmail -> city
   plateNumber, ownerEmail -> price
   plateNumber, ownerEmail -> postDate
   plateNumber, ownerEmail -> outerColor
   plateNumber, ownerEmail -> bookmarkedBy

   As we can see, every non-candidate key attribute is fully functionally dependent on a candidate key. Therefore, R1 complies with 2NF.
   As we can see, no non-candidate key attribute is transitively dependent on a candidate key. Therefore, R1 complies with 3NF.
   As we can see, for all functional dependencies in R1, the left hand side (determinant) contains a candidate key. Therefore, R1 is in BCNF.

   2. R2(CarInfo) FDs

   model -> bodyStyle
   model -> manufacturerName

   As we can see, every non-candidate key attribute is fully functionally dependent on a candidate key. Therefore, R2 complies with 2NF.
   As we can see, no non-candidate key attribute is transitively dependent on a candidate key. Therefore, R2 complies with 3NF.
   As we can see, for all functional dependencies in R2, the left hand side (determinant) contains a candidate key. Therefore, R2 is in BCNF.

   3. R3(Cities) FDs:

   city -> carState

   As we can see, every non-candidate key attribute is fully functionally dependent on a candidate key. Therefore, R3 complies with 2NF.
   As we can see, no non-candidate key attribute is transitively dependent on a candidate key. Therefore, R3 complies with 3NF.
   As we can see, for all functional dependencies in R3, the left hand side (determinant) contains a candidate key. Therefore, R3 is in BCNF.

   Therefore, we have successfully made a Lossless Join Decomposition.

BCNF Validation of UserAccount Relation:

UserAccount(email, firstName, lastName, username, password)

1. All the functional dependencies:

   email -> firstName
   email -> lastName
   email -> username
   email -> password

As we can see, every non-candidate key attribute is fully functionally dependent on a candidate key. Therefore, UserAccount complies with 2NF.
As we can see, no non-candidate key attribute is transitively dependent on a candidate key. Therefore, UserAccount complies with 3NF.
As we can see, for all functional dependencies in UserAccount, the left hand side (determinant) contains a candidate key. Therefore, UserAccount is in BCNF.

BCNF Validation of Purchases Relation:

Purchases(purchaserEmail, plateNumber, purchaseDate, purchasePrice, sellerEmail)

1. All the functional dependencies:

   purchaserEmail, plateNumber -> purchaseDate
   purchaserEmail, plateNumber -> purchasePrice
   purchaserEmail, plateNumber -> sellerEmail

As we can see, every non-candidate key attribute is fully functionally dependent on a candidate key. Therefore, Purchases complies with 2NF.
As we can see, no non-candidate key attribute is transitively dependent on a candidate key. Therefore, Purchases complies with 3NF.
As we can see, for all functional dependencies in Purchases, the left hand side (determinant) contains a candidate key. Therefore, Purchases is in BCNF.

BCNF Validation of Sales Relation:

Purchases(sellerEmail, plateNumber, purchaserEmail, salesPrice, sellDate)

1. All the functional dependencies:

   sellerEmail, plateNumber -> sellDate
   sellerEmail, plateNumber -> purchaserEmail
   sellerEmail, plateNumber -> salesPrice

As we can see, every non-candidate key attribute is fully functionally dependent on a candidate key. Therefore, Sales complies with 2NF.
As we can see, no non-candidate key attribute is transitively dependent on a candidate key. Therefore, Sales complies with 3NF.
As we can see, for all functional dependencies in Sales, the left hand side (determinant) contains a candidate key. Therefore, Sales is in BCNF.

BCNF Validation of Issue Relation:

Issue(issuerEmail, issueId, issueDate, issueText)

1. All the functional dependencies:

   issuerEmail, issueId -> issueDate
   sellerEmail, plateNumber -> issueText

As we can see, every non-candidate key attribute is fully functionally dependent on a candidate key. Therefore, Issue complies with 2NF.
As we can see, no non-candidate key attribute is transitively dependent on a candidate key. Therefore, Issue complies with 3NF.
As we can see, for all functional dependencies in Issue, the left hand side (determinant) contains a candidate key. Therefore, Issue is in BCNF.

BCNF Validation of PhoneNumber Relation:

PhoneNumber(ownerEmail, phoneNumber)

1. All the functional dependencies:

   There is no non-key attribute

Since, the primary key is fully functionally dependent itself, PhoneNumber complies with 2NF.
Since there is no non-candidate key attribute at all in PhoneNumber, it complies with 3NF.
Since, the only functional dependency in PhoneNumber is that the candidate key determines itself, the left hand side (determinant) contains a candidate key. Therefore, PhoneNumber is in BCNF.

```
+--------------+---------------------+------------+-----+---------+---------+---------+----------------+-------+-----------+-----
-------+--------------+---------------+---------+
2000 rows in set (0.01 sec)

mysql> SELECT COUNT(*) FROM carpost;
+----------+
| COUNT(*) |
+----------+
|     2000 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM cities;
+----------+
| COUNT(*) |
+----------+
|       20 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM carpost A, cities B WHERE A.city = B.city;
+----------+
| COUNT(*) |
+----------+
|     2000 |
+----------+
1 row in set (0.03 sec)

mysql>
```

```
mysql> SELECT COUNT(*) FROM carpost;
+----------+
| COUNT(*) |
+----------+
|     2000 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM carinfo;
+----------+
| COUNT(*) |
+----------+
|       20 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM carpost A, carinfo B WHERE A.model = B.model;
+----------+
| COUNT(*) |
+----------+
|     2000 |
+----------+
1 row in set (0.00 sec)

mysql> 
```

54°

Search

7:44 AM
5/9/2024

```
mysql> SELECT COUNT(*) FROM carpost;
+----------+
| COUNT(*) |
+----------+
|     2000 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM carinfo;
+----------+
| COUNT(*) |
+----------+
|       20 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM cities;
+----------+
| COUNT(*) |
+----------+
|       20 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM carpost A, carinfo B, cities C WHERE A.model = B.model AND A.city = C.city;
+----------+
| COUNT(*) |
+----------+
|     2000 |
+----------+
1 row in set (0.00 sec)

mysql>
```