Phase 8

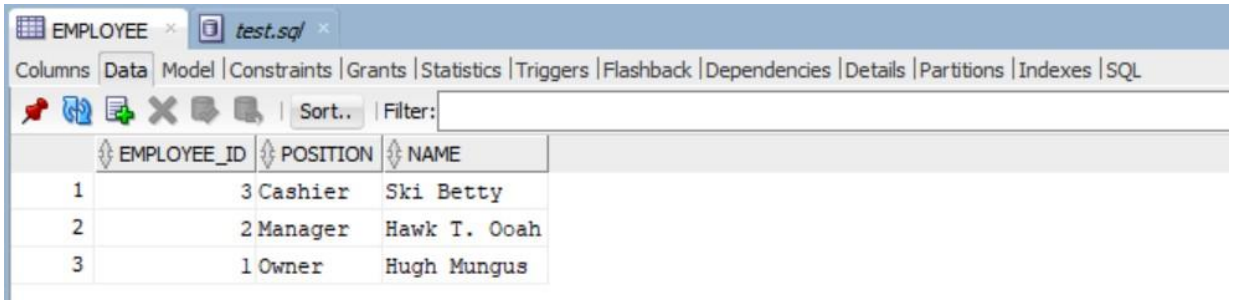
Toronto Metropolitan University

Simon Lin (501103322),
Dylan Ha (501056670),
Enes Polat (501061594)

CPS510 - Database Systems

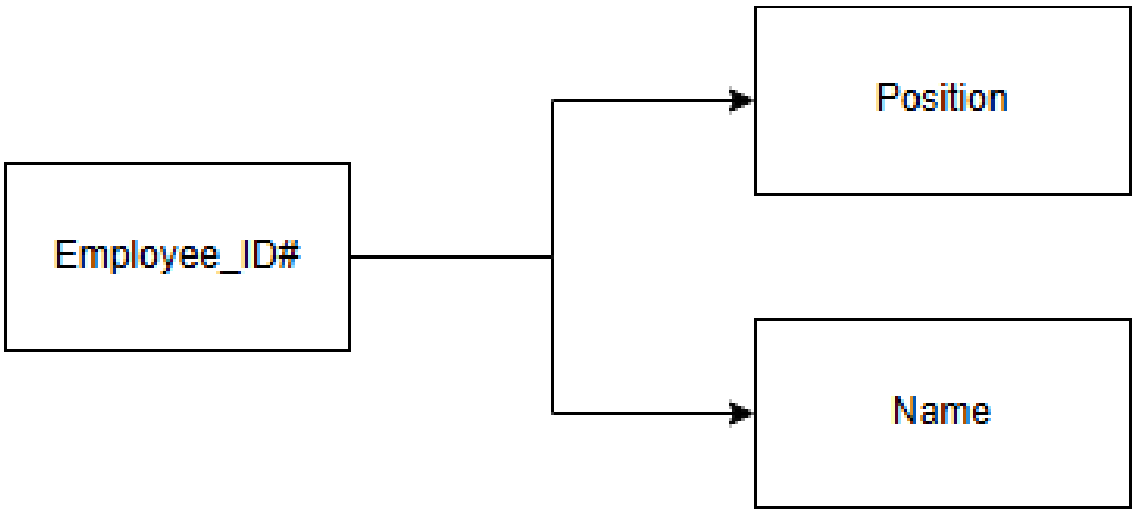
Point of Sale System for Shopper Drug Marts

Database Normalization/BCNF:

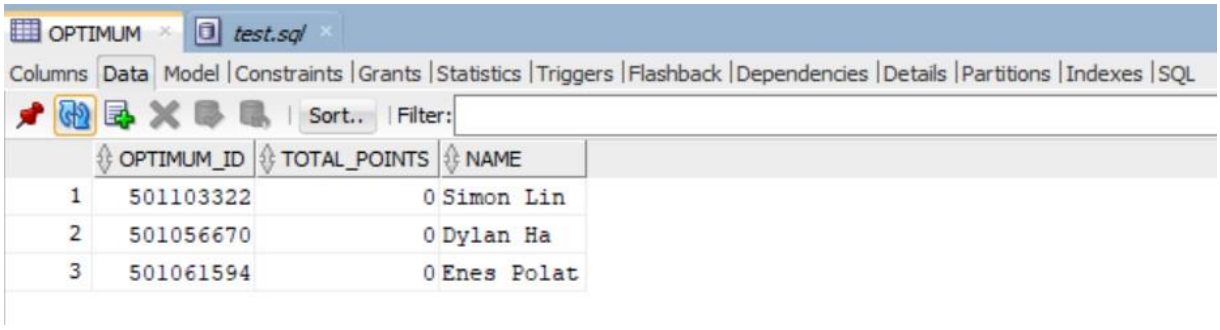
Employee (<u>Employee_ID#</u>, Position, Name)	
	
Functional Dependencies:	{Employee_ID} → Position {Employee_ID} → Name

Bernstein's Algorithm:

1. Minimize the list of FDs (FD X → Y, Z)
    a. FD1: {Employee_ID} → Position
    b. FD2: {Employee_ID} → Name
    c. Combine the FDs to form a single FD
        i. {Employee_ID} → Position, Name
2. Get rid of redundant FDs
    a. Since we combined FD1 and FD2 into a single dependency, there are no redundant dependencies to remove here.
3. Minimize left hand side
    a. The left hand side is just Employee_ID, which is already the sole candidate key and superkey in the relation. No further decomposition is needed.
4. Derive final schema
    a. We derive the final schema of
    Employee (<u>Employee_ID#</u>, Position, Name)
    Employee_ID is the candidate key and determines all other attributes in the relation, therefore the relation satisfies BCNF and 3NF.


Position
Employee_ID#
Name

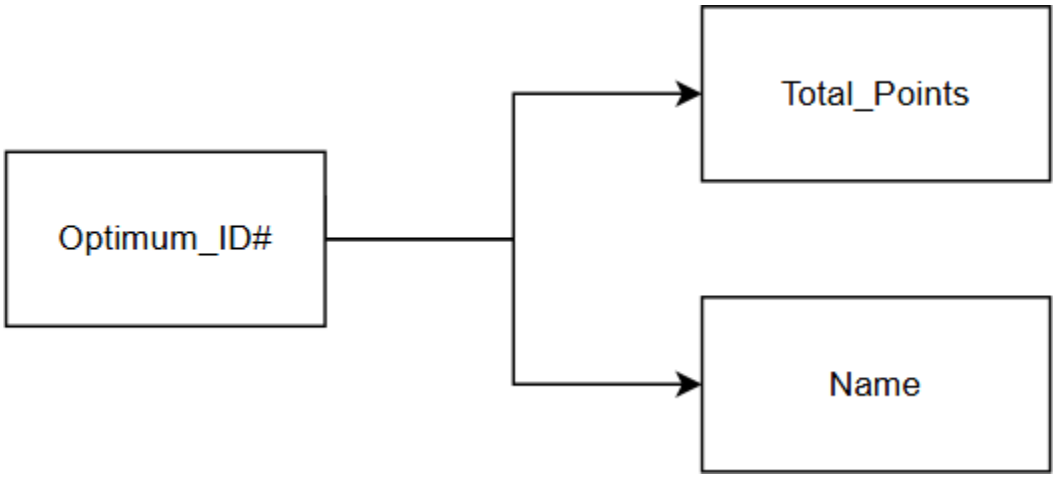
Optimum(<u>Optimum_ID#</u>, Total_Points, Name)

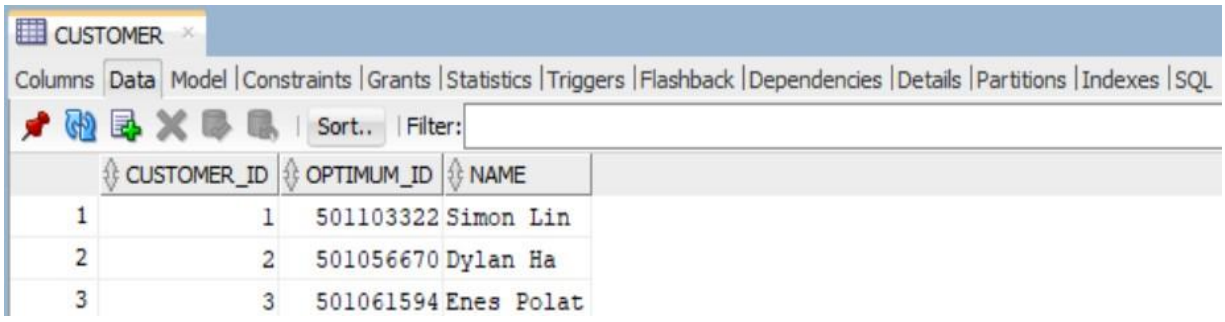


Functional Dependencies:	{Optimum_ID} → Total_Points {Optimum_ID} → Name

Bernstein's Algorithm:

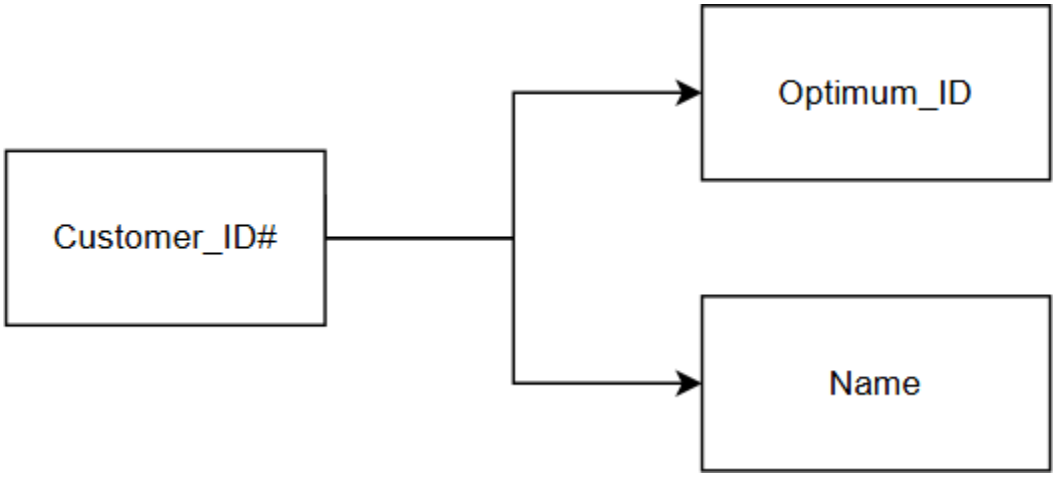
1. Minimize the list of FDs (FD X → Y, Z)
    b. FD1: {Optimum_ID} → Total_Points
    c. FD2: {Optimum_ID} → Name
    d. Combine the FDs to form a single FD
        i. {Optimum_ID} → Total_Points, Name
2. Get rid of redundant FDs
    e. Since we combined FD1 and FD2 into a single dependency, there are no redundant dependencies to remove here.
3. Minimize left hand side
    f. The left hand side is just Optimum_ID, which is already the sole candidate key and superkey in the relation. No further decomposition is needed.
4. Derive final schema
    g. We derive the final schema of
    Optimum(<u>Optimum_ID#</u>, Total_Points, Name)
    Optimum_ID is the candidate key and determines all other attributes in the relation, therefore the relation satisfies BCNF and 3NF.



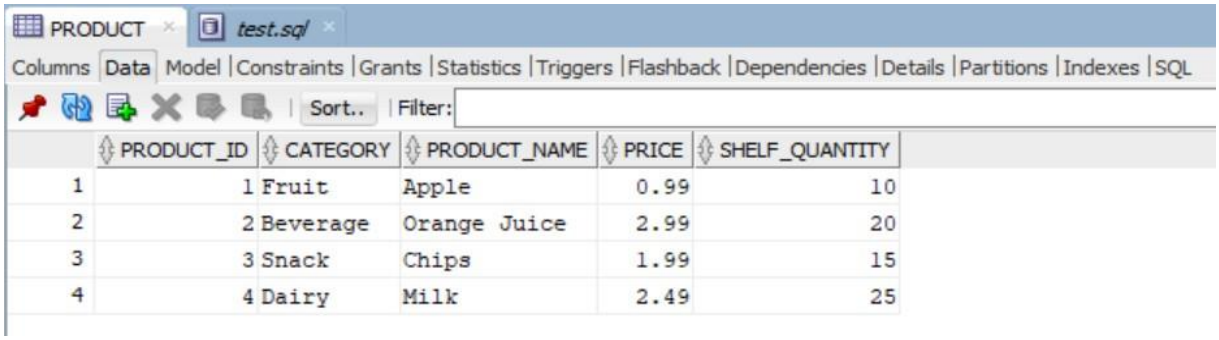
Customer(Customer_ID#, Optimum_ID, Name)	
	
Functional Dependencies:	{Customer_ID} → Optimum_ID {Customer_ID} → Name
Bernstein's Algorithm: 1. Minimize the list of FDs (FD X → Y, Z)	

a. FD1: {Customer_ID} → Optimum_ID
b. FD2: {Customer_ID} → Name
c. Combine the FDs to form a single FD
    i. {Customer_ID} → Optimum_ID, Name

2. Get rid of redundant FDs
    a. Since we combined FD1 and FD2 into a single dependency, there are no redundant dependencies to remove here.
3. Minimize left hand side
    a. The left hand side is just Customer_ID, which is already the sole candidate key and superkey in the relation. No further decomposition is needed.
4. Derive final schema
    a. We derive the final schema of
    Customer(Customer_ID#, Optimum_ID, Name)
    Customer_ID is the candidate key and determines all other attributes in the relation, therefore the relation satisfies BCNF and 3NF.



Product(<u>Product_ID#</u>, Category, Product_Name, Price, Shelf_Quantity)



	PRODUCT_ID	CATEGORY	PRODUCT_NAME	PRICE	SHELF_QUANTITY
1	1	Fruit	Apple	0.99	10
2	2	Beverage	Orange Juice	2.99	20
3	3	Snack	Chips	1.99	15
4	4	Dairy	Milk	2.49	25

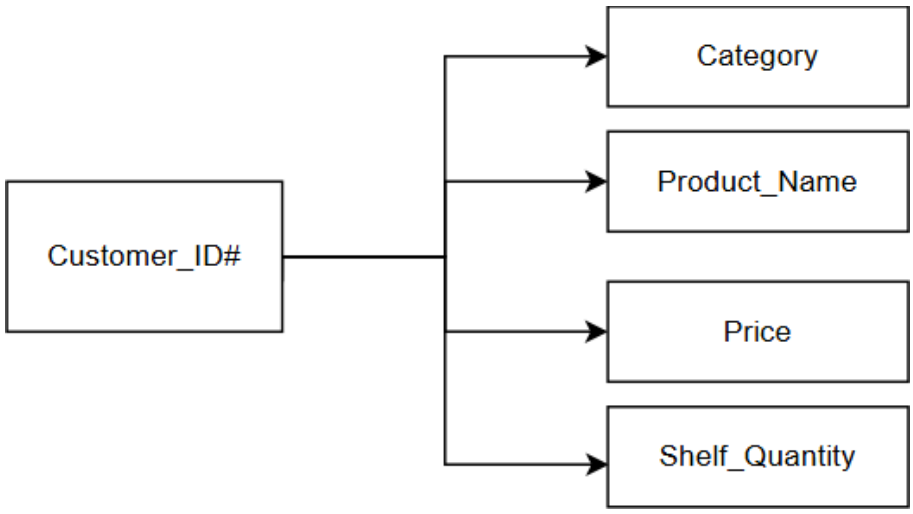
Functional Dependencies:	{Product_ID} → Category {Product_ID} → Product_Name {Product_ID} → Price {Product_ID} → Shelf_Quantity

Bernstein's Algorithm:

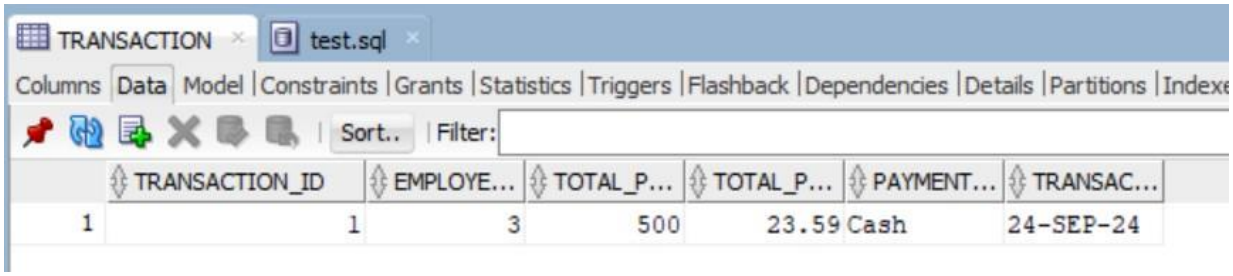
1. Minimize the list of FDs (FD X → Y, Z)
    a. FD1: {Product_ID} → Category
    b. FD2: {Product_ID} → Product_Name
    c. FD3: {Product_ID} → Price
    d. FD4: {Product_ID} → Shelf_Quantity
    e. Combine the FDs to form a single FD
        ii. {Product_ID} → Category, Product_Name, Price, Shelf_Quantity
2. Get rid of redundant FDs
    a. Since we combined FD1, FD2, FD3 and FD4 into a single dependency, there are no redundant dependencies to remove here.
3. Minimize left hand side
    a. The left hand side is just Product_ID, which is already the sole candidate key and superkey in the relation. No further decomposition is needed.
4. Derive final schema
    a. We derive the final schema of

    Product(<u>Product_ID#</u>, Category, Product_Name, Price, Shelf_Quantity)

    Product_ID is the candidate key and determines all other attributes in the relation, therefore the relation satisfies BCNF and 3NF.



Transaction(<u>Transaction_ID#</u>, <u>Employee_ID#</u>, Total_PointsSpent, Total_Price, Payment_Method, Transaction_Date)



Functional Dependencies:	{Transaction_ID} → Employee_ID {Transaction_ID} → Total_PointsSpent {Transaction_ID} → Total_Price {Transaction_ID} → Payment_Method {Transaction_ID} → Transaction_Date

Bernstein's Algorithm:

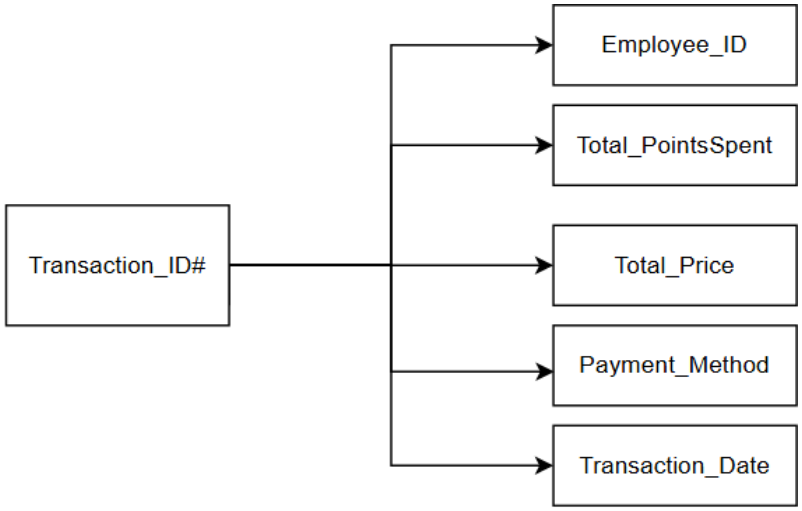
1. Minimize the list of FDs (FD X → Y, Z)
    a. FD1: {Transaction_ID} → Employee_ID
    b. FD2: {Transaction_ID} → Total_PointsSpent

c. FD3: {Transaction_ID} → Total_Price
d. FD4: {Transaction_ID} → Payment_Method
e. FD5: {Transaction_ID} → Transaction_Date
f. Combine the FDs to form a single FD
iii. {Transaction_ID} → Employee_ID, Total_PointsSpent, Total_Price, Payment_Method, Transaction_Date

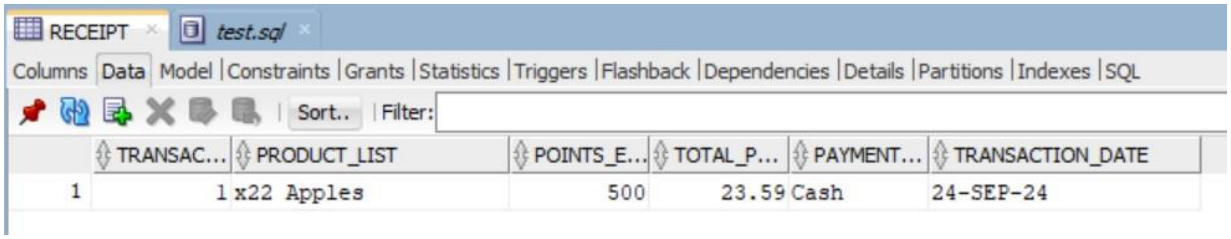
2. Get rid of redundant FDs
b. Since we combined FD1, FD2, FD3, FD4 and FD5 into a single dependency, there are no redundant dependencies to remove here.

3. Minimize left hand side
b. The left hand side is just Transaction_ID, which is already the sole candidate key and superkey in the relation. No further decomposition is needed.

4. Derive final schema
b. We derive the final schema of
Transaction(Transaction_ID#, Employee_ID#, Total_PointsSpent, Total_Price, Payment_Method, Transaction_Date)
Transaction_ID is the candidate key and determines all other attributes in the relation, therefore the relation satisfies BCNF and 3NF.



Receipt(Transaction_ID#, Product_List, Points_Earned, Total_Price, Payment_Method, Transaction_Date)



Functional Dependencies:	{Transaction_ID} → Product_List {Transaction_ID} → Points_Earned {Transaction_ID} → Total_Price {Transaction_ID} → Payment_Method {Transaction_ID} → Transaction_Date
Function Dependencies Removed Redundancy:	{Transaction_ID} → Product_List {Transaction_ID} → Points_Earned

Bernstein's Algorithm:

1. Minimize the list of FDs (FD X → Y, Z)
    a. FD1: {Transaction_ID} → Product_List
    b. FD2: {Transaction_ID} → Points_Earned
    c. FD3: {Transaction_ID} → Total_Price

d. FD4: {Transaction_ID} → Payment_Method
e. FD5: {Transaction_ID} → Transaction_Date
g. Combine the FDs to form a single FD
iv. {Transaction_ID} → Product_List, Points_Earned, Total_Price, Payment_Method, Transaction_Date

2. Get rid of redundant FDs

Find X+ in the reduced list, If X+ contains Y, then X, Y is redundant

a. We see that Total_Price, Payment_Method and Transaction_Date are repeated FDs from the transaction table, we eliminate these dependencies from the table.
b. This leaves {Transaction_ID} → Product_List, Points_Earned

3. Minimize left hand side

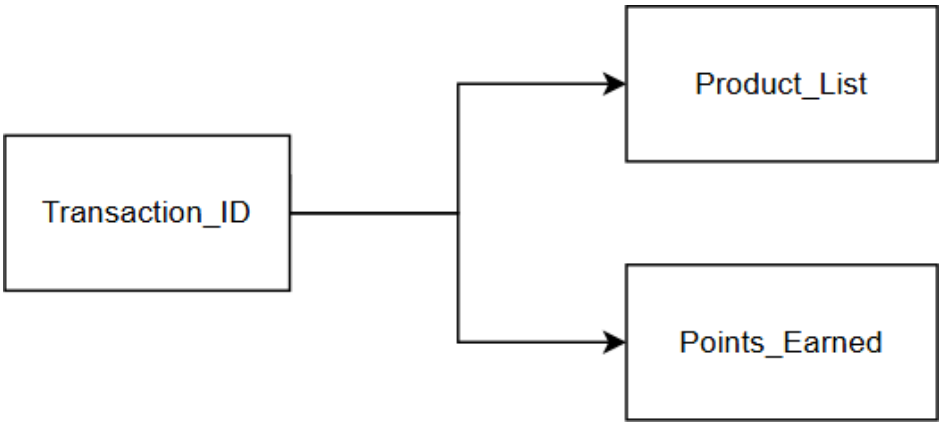
c. The left hand side is just Transaction_ID, which is already the sole candidate key and superkey in the relation. No further decomposition is needed.

4. Derive final schema

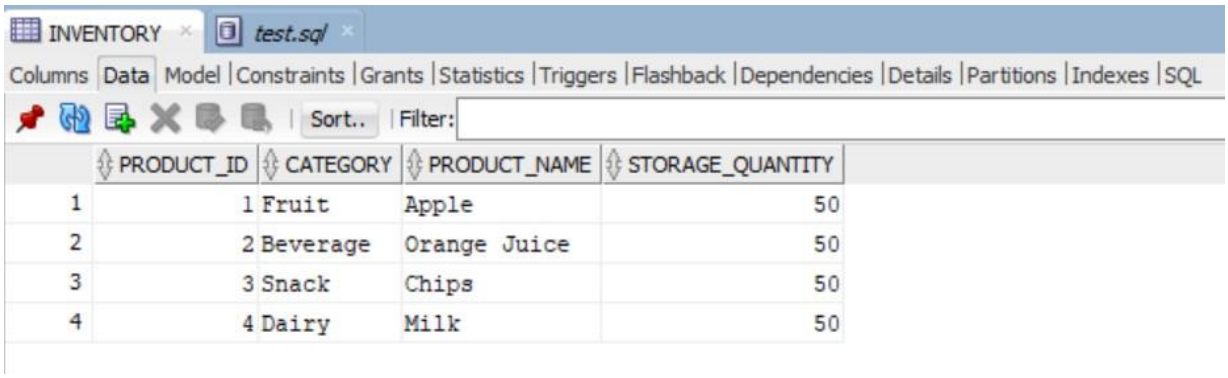
c. We derive the final schema of

Receipt(Transaction_ID#, Product_List, Points_Earned)

Transaction_ID is the candidate key and determines all other attributes in the relation, therefore the relation satisfies BCNF and 3NF.



Inventory(Product_ID#, Category, Product_Name, Storage_Quantity)



	PRODUCT_ID	CATEGORY	PRODUCT_NAME	STORAGE_QUANTITY
1	1	Fruit	Apple	50
2	2	Beverage	Orange Juice	50
3	3	Snack	Chips	50
4	4	Dairy	Milk	50

Functional Dependencies:	{Product_ID} → Category {Product_ID} → Product_Name {Product_ID} → Storage_Quantity
Function Dependencies Removed Redundancy:	{Product_ID} → Storage_Quantity

Bernstein's Algorithm:

1. Minimize the list of FDs (FD X → Y, Z)
    a. FD1: {Product_ID} → Category
    b. FD2: {Product_ID} → Product_Name

c. FD3: {Product_ID} → Storage_Quantity

d. Combine the FDs to form a single FD

i. {Product_ID} → Category, Product_Name, Storage_Quantity

2. Get rid of redundant FDs

Find X+ in the reduced list, If X+ contains Y, then X, Y is redundant

a. We see that Category and Product_Name are repeated FDs from the Product table, we eliminate these dependencies from the table.

c. This leaves {Product_ID} → Storage_Quantity

3. Minimize left hand side

a. The left hand side is just Product_ID, which is already the sole candidate key and superkey in the relation. No further decomposition is needed.

4. Derive final schema

d. We derive the final schema of

Inventory(Product_ID#, Storage_Quantity)

Product_ID is the candidate key and determines all other attributes in the relation, therefore the relation satisfies BCNF and 3NF.

The Inventory table relation is in **3NF**, all non-key values contained in the table are non-transitive and dependent on only the primary key.