// Match HighProtein nodes and all their relationships to Product

MATCH (hp:HighProtein)-[r:High\_Protein]-(p:Product)

// Remove all High\_Protein relationships

DELETE r

// Ensure that the HighProtein node has no other relationships

WITH hp

MATCH (hp)-[remaining]-()

DELETE remaining

// Delete the HighProtein node

DELETE hp

// 查找并删除具有特定描述的 KetoDiet 节点

MATCH (kd:DietaryClustering {name: "KetoDiet", description: "Carbohydrate: 5-10%, Protein: 10-20%, Fat: 70-80%"})

DETACH DELETE kd

// 删除 HighFat 节点及其与产品的关系

MATCH (hf:HighFat)

OPTIONAL MATCH (p)-[r:HIGH\_FAT]->(hf)

DELETE r, hf

CREATE (:DietaryClustering {name: "HighProtein"})

CREATE (:DietaryClustering {name: "HighCarbohydrate"})

CREATE (:DietaryClustering {name: "HighFat"})

CREATE (:DietaryClustering {name: "HighFiber"})

MATCH (node:DietaryClustering)

WHERE node.name = "HighProtein"

SET node.description = "High protein ratio products which over 20%"

RETURN node

UNION

MATCH (node:DietaryClustering)

WHERE node.name = "HighCarbohydrate"

SET node.description = "High Carbohydrate ratio products which over 40%"

RETURN node

UNION

MATCH (node:DietaryClustering)

WHERE node.name = "HighFat"

SET node.description = "High fat ratio products which over 30%"

RETURN node

UNION

MATCH (node:DietaryClustering)

WHERE node.name = "HighFiber"

SET node.description = "High fiber ratio products which over 5%"

RETURN node

// HighProtein 关系创建

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WHERE ni.Carbohydrate =~ '^[0-9.]+g$'

AND ni.Dietary\_Fibre =~ '^[0-9.]+g$'

AND ni.Fat\_Total =~ '^[0-9.]+g$'

AND ni.Protein =~ '^[0-9.]+g$'

WITH p,

toFloat(replace(ni.Carbohydrate, 'g', '')) AS carbohydrate,

toFloat(replace(ni.Dietary\_Fibre, 'g', '')) AS fiber,

toFloat(replace(ni.Fat\_Total, 'g', '')) AS fat,

toFloat(replace(ni.Protein, 'g', '')) AS protein

WITH p, carbohydrate, fiber, fat, protein, carbohydrate + fiber + fat + protein AS total

WHERE total > 0

MATCH (hp:DietaryClustering {name: "HighProtein"})

WHERE protein / total > 0.2

MERGE (hp)-[:High\_Protein]->(p)

如果四个元素任何一个存在，那么产品就会保留，其他不存在的值用0代替。

// 为 HighCarbohydrate 关系创建

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WITH p,

COALESCE(toFloat(replace(ni.Carbohydrate, 'g', '')), 0) AS carbohydrate,

COALESCE(toFloat(replace(ni.Dietary\_Fibre, 'g', '')), 0) AS fiber,

COALESCE(toFloat(replace(ni.Fat\_Total, 'g', '')), 0) AS fat,

COALESCE(toFloat(replace(ni.Protein, 'g', '')), 0) AS protein

// 将总量设为所有营养成分之和

WITH p, carbohydrate, fiber, fat, protein, carbohydrate + fiber + fat + protein AS total

WHERE total > 0

// 匹配 HighCarbohydrate 节点

MATCH (hc:DietaryClustering {name: "HighCarbohydrate"})

// 判断碳水化合物的比例是否超过 40%

WHERE carbohydrate / total > 0.4

MERGE (hc)-[:High\_Carbohydrate]->(p)

// 创建 HighKilojoules 节点

CREATE (:DietaryClustering {name: "HighKilojoules", description: "High Kilojoules Products which over 1000kJ"})

// 使用 WITH 分隔语句

WITH "HighKilojoules" AS clusterName

// 匹配符合要求的产品

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WHERE ni.Energy\_kJ =~ '^[0-9.]+kJ$'

WITH p, toFloat(replace(ni.Energy\_kJ, 'kJ', '')) AS energy\_kJ, clusterName

// 筛选出能量大于等于 1000kJ 的产品

WHERE energy\_kJ >= 1000

// 匹配 HighKilojoules 节点

MATCH (hk:DietaryClustering {name: clusterName})

// 建立关系

MERGE (hk)-[:High\_Kilojoules]->(p)

// 创建地中海饮食节点

CREATE (:DietaryClustering {name: "MediterraneanDiet", description: "Carbohydrate: 40-50%, Protein: 15-20%, Fat: 30-35%"})

// 使用 WITH 来分隔语句

WITH "MediterraneanDiet" AS clusterName

// 匹配符合要求的产品

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WHERE ni.Carbohydrate =~ '^[0-9.]+g$'

AND ni.Fat\_Total =~ '^[0-9.]+g$'

AND ni.Protein =~ '^[0-9.]+g$'

WITH p,

toFloat(replace(ni.Carbohydrate, 'g', '')) AS carbohydrate,

toFloat(replace(ni.Fat\_Total, 'g', '')) AS fat,

toFloat(replace(ni.Protein, 'g', '')) AS protein,

clusterName

WITH p, carbohydrate, fat, protein, carbohydrate + fat + protein AS total, clusterName

WHERE total > 0

MATCH (md:DietaryClustering {name: clusterName})

WHERE (carbohydrate / total) >= 0.4 AND (carbohydrate / total) <= 0.5

AND (protein / total) >= 0.15 AND (protein / total) <= 0.2

AND (fat / total) >= 0.3 AND (fat / total) <= 0.35

MERGE (md)-[:Mediterranean\_Diet]->(p)

User 1 = goes to the gym actively, wants a high protein diet. User 2 = follows a balanced diet, totaling 2000 calories per day User 3 = wants to try a keto diet.

User 4 = wants to shop for 1400 calories total

// 创建生酮饮食节点

CREATE (:DietaryClustering {name: "KetoDiet", description: "Carbohydrate: 0-10%, Protein: 10-100%, Fat: 10-100%"})

// 使用 WITH 分隔语句

WITH "KetoDiet" AS clusterName

// 匹配符合生酮饮食比例的产品

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WITH p,

COALESCE(toFloat(replace(ni.Carbohydrate, 'g', '')), 0) AS carbohydrate,

COALESCE(toFloat(replace(ni.Fat\_Total, 'g', '')), 0) AS fat,

COALESCE(toFloat(replace(ni.Protein, 'g', '')), 0) AS protein,

clusterName

// 将总量设为所有营养成分之和

WITH p, carbohydrate, fat, protein, carbohydrate + fat + protein AS total, clusterName

WHERE total > 0

// 匹配 KetoDiet 节点

MATCH (kd:DietaryClustering {name: clusterName})

// 判断营养成分比例是否符合生酮饮食的要求

WHERE (carbohydrate / total) >= 0 AND (carbohydrate / total) <= 0.1

AND (protein / total) >= 0.1 AND (protein / total) <= 1

AND (fat / total) >= 0.1 AND (fat / total) <= 1

MERGE (kd)-[:Keto\_Diet]->(p)

// 创建ZoneDiet节点

CREATE (zd:DietaryClustering {name: "ZoneDiet", description: "Carbohydrate: 30-50%, Protein: 20-40%, Fat: 20-40%"})

WITH zd

// 匹配符合ZoneDiet饮食比例的产品

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WITH p,

COALESCE(toFloat(replace(ni.Carbohydrate, 'g', '')), 0) AS carbohydrate,

COALESCE(toFloat(replace(ni.Fat\_Total, 'g', '')), 0) AS fat,

COALESCE(toFloat(replace(ni.Protein, 'g', '')), 0) AS protein,

zd

// 计算总量

WITH p, carbohydrate, fat, protein, carbohydrate + fat + protein AS total, zd

WHERE total > 0

// 计算各营养成分比例并继续传递变量

WITH p, carbohydrate / total AS carb\_ratio, protein / total AS protein\_ratio, fat / total AS fat\_ratio, zd

// 判断营养成分比例是否符合ZoneDiet的要求

WHERE carb\_ratio >= 0.3 AND carb\_ratio <= 0.5

AND protein\_ratio >= 0.2 AND protein\_ratio <= 0.4

AND fat\_ratio >= 0.2 AND fat\_ratio <= 0.4

// 创建关联

MERGE (zd)-[:Zone\_Diet]->(p)

CREATE (dashDiet:DietaryClustering {name: "DashDiet", description: "6–8 servings of grains or grain products, 4–5 servings of fruits, 4–5 servings of vegetables, 2–3 servings of low-fat dairy foods, 2 or fewer servings of fish"})

CREATE (grain:Category {name: "GrainProducts", description: "6–8 servings of grains or grain products"})

CREATE (fruit:Category {name: "Fruits", description: "4–5 servings of fruits"})

CREATE (veg:Category {name: "Vegetables", description: "4–5 servings of vegetables"})

CREATE (dairy:Category {name: "LowFatDairy", description: "2–3 servings of low-fat dairy foods"})

CREATE (fish:Category {name: "Fish", description: "2 or fewer servings of fish"})

WITH dashDiet, grain, fruit, veg, dairy, fish

MERGE (dashDiet)-[:Dash\_Diet]->(grain)

MERGE (dashDiet)-[:Dash\_Diet]->(fruit)

MERGE (dashDiet)-[:Dash\_Diet]->(veg)

MERGE (dashDiet)-[:Dash\_Diet]->(dairy)

MERGE (dashDiet)-[:Dash\_Diet]->(fish)

WITH grain, fruit, veg, dairy, fish

MATCH (p:Product)-[:STOCKED\_IN]->(:Category {name: "PASTA / RICE"})

MERGE (grain)-[:Suit\_For]->(p)

WITH grain, fruit, veg, dairy, fish

MATCH (p:Product)-[:STOCKED\_IN]->(:Category {name: "FRUIT"})

MERGE (fruit)-[:Suit\_For]->(p)

WITH grain, fruit, veg, dairy, fish

MATCH (p:Product)-[:STOCKED\_IN]->(:Category {name: "CANNED VEGETABLES"})

MERGE (veg)-[:Suit\_For]->(p)

WITH grain, fruit, veg, dairy, fish

MATCH (p:Product)-[:STOCKED\_IN]->(:Category {name: "DAIRY - MILK"})-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WHERE toFloat(replace(ni.`Fat Total Quantity Per 100g - ValueWord - NIP`, 'g', '')) < 3.1

MERGE (dairy)-[:Suit\_For]->(p)

WITH grain, fruit, veg, dairy, fish

MATCH (p:Product)-[:STOCKED\_IN]->(:Category {name: "CANNED FISH"})

MERGE (fish)-[:Suit\_For]->(p)

WITH grain, fruit, veg, dairy, fish

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WITH p, ni, toFloat(replace(ni.Energy\_kJ, 'kJ', '')) / 4.184 AS calories

SET p.Calorie = calories

RETURN p.name, p.Calorie

// 创建EliminationDiet节点

CREATE (ed:DietaryClustering {name: "EliminationDiet", description: "Products that do not contain egg or milk ingredients"})

WITH ed

// 查找并关联不含egg或milk的Product

MATCH (p:Product)-[:HAS\_RECIPE]->(r:Recipe)

WHERE NOT (

    toLower(r.ingredients\_orig) CONTAINS 'egg' OR

    toLower(r.ingredients\_orig) CONTAINS 'milk'

)

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)

WHERE NOT (

    toLower(ir.ingredient\_name) CONTAINS 'egg' OR

    toLower(ir.ingredient\_name) CONTAINS 'milk' OR

    toLower(ir.scientific\_name) CONTAINS 'egg' OR

    toLower(ir.scientific\_name) CONTAINS 'milk' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'egg' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'milk'

)

WITH p, ed

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(:IngredientResearch)-[:has\_ingredients]->(is:IngredientSource)

WHERE NOT (

    toLower(is.name) CONTAINS 'egg' OR

    toLower(is.name) CONTAINS 'milk'

)

WITH p, ed

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(:IngredientResearch)-[:has\_ingredients]->(:IngredientSource)-[:SOURCED\_FROM]->(t3ir:T3IngredientResearch)

WHERE NOT (

    toLower(t3ir.name) CONTAINS 'egg' OR

    toLower(t3ir.name) CONTAINS 'milk' OR

    toLower(t3ir.scientific\_name) CONTAINS 'egg' OR

    toLower(t3ir.scientific\_name) CONTAINS 'milk' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'egg' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'milk'

)

MERGE (ed)-[:EliminationDiet\_To]->(p)

RETURN p.name AS ProductName

CREATE (gfcf:DietaryClustering {name: "GlutenFreeAndCaseinFree", description: "Products that contain no gluten and no casein ingredients"})

WITH gfcf

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)

WHERE NOT (

    toLower(ir.ingredient\_name) CONTAINS 'gluten' OR

    toLower(ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten' OR

    toLower(ir.ingredient\_name) CONTAINS 'casein' OR

    toLower(ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'casein'

)

AND NOT EXISTS (

    (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: 'gluten'})

)

AND NOT EXISTS (

    (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: 'casein'})

)

WITH DISTINCT p, gfcf

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)-[:has\_ingredients]->(:IngredientSource)-[:SOURCED\_FROM]->(t3ir:T3IngredientResearch)

WHERE NOT (

    toLower(t3ir.name) CONTAINS 'gluten' OR

    toLower(t3ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten' OR

    toLower(t3ir.name) CONTAINS 'casein' OR

    toLower(t3ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'casein'

)

AND NOT EXISTS (

    (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: 'gluten'})

)

AND NOT EXISTS (

    (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: 'casein'})

)

MERGE (gfcf)-[:GlutenFreeAndCaseinFree\_To]->(p)

RETURN p.name AS ProductName

CREATE (gf:DietaryClustering {name: "GlutenFree", description: "Products that contain no gluten ingredients"})

WITH gf

// Finding products that don't have gluten ingredients or allergens

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)

WHERE NOT (

    toLower(ir.ingredient\_name) CONTAINS 'gluten' OR

    toLower(ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten'

)

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "Gluten"})

WITH DISTINCT p, gf

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)-[:has\_ingredients]->(:IngredientSource)-[:SOURCED\_FROM]->(t3ir:T3IngredientResearch)

WHERE NOT (

    toLower(t3ir.name) CONTAINS 'gluten' OR

    toLower(t3ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten'

)

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "Gluten"})

// Create relationship to mark the products as GlutenFree

MERGE (gf)-[:GlutenFree\_To]->(p)

RETURN p.name AS ProductName

CREATE (cf:DietaryClustering {name: "CaseinFree", description: "Products that contain no casein ingredients"})

WITH cf

// Find products that don't have casein ingredients or allergens

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)

WHERE NOT (

    toLower(ir.ingredient\_name) CONTAINS 'casein' OR

    toLower(ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'casein'

)

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "casein"})

WITH DISTINCT p, cf

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)-[:has\_ingredients]->(:IngredientSource)-[:SOURCED\_FROM]->(t3ir:T3IngredientResearch)

WHERE NOT (

    toLower(t3ir.name) CONTAINS 'casein' OR

    toLower(t3ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'casein'

)

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "casein"})

// Create relationship to mark the products as CaseinFree

MERGE (cf)-[:CaseinFree\_To]->(p)

RETURN p.name AS ProductName

CREATE (user1:DietaryClustering {name: "USER1", description: "User1 would like to shop 1400 calories products for total."})

WITH user1

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WITH p, ni, toFloat(replace(ni.Energy\_kJ, 'kJ', '')) / 4.184 AS calories, user1

SET p.Calorie = calories

WITH p, user1

WHERE p.Calorie < 1400

MERGE (user1)-[:Selected\_To]->(p)

RETURN p.name, p.Calorie

CREATE (user2:DietaryClustering {name: "USER2", description: "User would like to shop 2400 calories products for total based on the Keto Diet."})

WITH user2

MATCH (ketoDiet:DietaryClustering {name: "KetoDiet"})-[:Keto\_Diet]->(product:Product)

WITH user2, product

MATCH (product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WITH user2, product, ni, toFloat(replace(ni.Energy\_kJ, 'kJ', '')) / 4.184 AS calories

SET product.Calorie = calories

WITH user2, product

WHERE product.Calorie < 2400

MERGE (user2)-[:Selected\_To]->(product)

RETURN product.name, product.Calorie

CREATE (user3:DietaryClustering {name: "USER3", description: "User would like to shop 2000 calories products for total based on the Gluten-free casein-free diet."})

WITH user3

MATCH (gfDiet:DietaryClustering {name: "GlutenFreeAndCaseinFree"})-[GlutenFreeAndCaseinFree\_To

]->(product:Product)

WITH user3, product

MATCH (product)-[:HAS\_RECIPE]->(:Recipe)-[:HAS\_NUTRITIONAL\_INFO]->(ni:NutritionalInformation)

WITH user3, product, ni, toFloat(replace(ni.Energy\_kJ, 'kJ', '')) / 4.184 AS calories

SET product.Calorie = calories

WITH user3, product

WHERE product.Calorie < 2000

MERGE (user3)-[:Selected\_To]->(product)

RETURN product.name, product.Calorie

CREATE (glutenCaseinAllergic:CustomizedRecommendationEngine {name: "Gluten&CaseinAllergic", description: "Customized screening for those with Gluten&Casein allergies."})

WITH glutenCaseinAllergic

MATCH (gfDiet:DietaryClustering {name: "GlutenFreeAndCaseinFree"})-[:GlutenFreeAndCaseinFree\_To]->(product:Product)

MERGE (glutenCaseinAllergic)-[:Recommended\_To]->(product)

RETURN product.name

CREATE (diabetesPatient:CustomizedRecommendationEngine {name: "DiabetesPatient", description: "Food screening for diabetics.Fat <= 10g per 100g, Sugars <= 10g per 100g, Dietary Fibre >= 6g per 100g"})

WITH diabetesPatient

MATCH (diabetesDiet:DietaryClustering {name: "DiabetesDiet"})-[:Diabetes\_Diet]->(product:Product)

MERGE (diabetesPatient)-[:Recommended\_To]->(product)

RETURN product.name

CREATE (highProteinRatio:CustomizedRecommendationEngine {name: "HighProtenRatio", description: "Screening products with protein content of more than 20% for those with high protein needs."})

WITH highProteinRatio

MATCH (highProtein:DietaryClustering {name: "HighProtein"})-[:High\_Protein

]->(product:Product)

MERGE (highProteinRatio)-[:Recommended\_To]->(product)

RETURN product.name

CREATE (hypertensionPatient:CustomizedRecommendationEngine {name: "HypertensionPatient", description: "The DASH diet, screened for people with high blood pressure."})

WITH hypertensionPatient

MATCH (dashDiet:DietaryClustering {name: "DashDiet"})-[:Dash\_Diet]->(category:Category)

MERGE (hypertensionPatient)-[:Recommended\_To]->(category)

RETURN category.name

CREATE (cf:DietaryClustering {name: "CaseinFree", description: "Products that contain no casein ingredients"})

WITH cf

// Find products that don't have casein, milk, or dairy ingredients or allergens

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)

WHERE NOT (

    toLower(ir.ingredient\_name) CONTAINS 'casein' OR

    toLower(ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'casein' OR

    toLower(ir.ingredient\_name) CONTAINS 'milk' OR

    toLower(ir.scientific\_name) CONTAINS 'milk' OR

    toLower(ir.ingredient\_name) CONTAINS 'dairy' OR

    toLower(ir.scientific\_name) CONTAINS 'dairy'

)

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "casein"})

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "milk"})

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "dairy"})

WITH cf, p, ir

OPTIONAL MATCH (ir)-[:HAS\_COMPOSITION]->(comp:Composition)

WITH cf, p, COLLECT(comp) AS comps

WHERE ALL(comp IN comps WHERE NOT (

    toLower(comp.description) CONTAINS 'casein' OR

    toLower(comp.description) CONTAINS 'milk' OR

    toLower(comp.description) CONTAINS 'dairy'

))

WITH DISTINCT p, cf

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)-[:has\_ingredients]->(:IngredientSource)-[:SOURCED\_FROM]->(t3ir:T3IngredientResearch)

WHERE NOT (

    toLower(t3ir.name) CONTAINS 'casein' OR

    toLower(t3ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'casein' OR

    toLower(t3ir.name) CONTAINS 'milk' OR

    toLower(t3ir.scientific\_name) CONTAINS 'milk' OR

    toLower(t3ir.name) CONTAINS 'dairy' OR

    toLower(t3ir.scientific\_name) CONTAINS 'dairy'

)

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "casein"})

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "milk"})

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "dairy"})

WITH cf, p, t3ir

OPTIONAL MATCH (t3ir)-[:HAS\_COMPOSITION]->(comp:Composition)

WITH cf, p, COLLECT(comp) AS t3Comps

WHERE ALL(comp IN t3Comps WHERE NOT (

    toLower(comp.description) CONTAINS 'casein' OR

    toLower(comp.description) CONTAINS 'milk' OR

    toLower(comp.description) CONTAINS 'dairy'

))

// Create relationship to mark the products as CaseinFree

MERGE (cf)-[:CaseinFree\_To]->(p)

RETURN p.name AS ProductName

CREATE (gf:DietaryClustering {name: "GlutenFree", description: "Products that contain no gluten ingredients"})

WITH gf

// Finding products that don't have gluten ingredients or allergens

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)

WHERE NOT (

    toLower(ir.ingredient\_name) CONTAINS 'gluten' OR

    toLower(ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten'

)

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "Gluten"})

WITH gf, p, ir

OPTIONAL MATCH (ir)-[:HAS\_COMPOSITION]->(comp:Composition)

WITH gf, p, COLLECT(comp) AS comps

WHERE ALL(comp IN comps WHERE NOT toLower(comp.description) CONTAINS 'gluten')

WITH DISTINCT p, gf

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)-[:has\_ingredients]->(:IngredientSource)-[:SOURCED\_FROM]->(t3ir:T3IngredientResearch)

WHERE NOT (

    toLower(t3ir.name) CONTAINS 'gluten' OR

    toLower(t3ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten'

)

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "Gluten"})

WITH gf, p, t3ir

OPTIONAL MATCH (t3ir)-[:HAS\_COMPOSITION]->(comp:Composition)

WITH gf, p, COLLECT(comp) AS t3Comps

WHERE ALL(comp IN t3Comps WHERE NOT toLower(comp.description) CONTAINS 'gluten')

// Create relationship to mark the products as GlutenFree

MERGE (gf)-[:GlutenFree\_To]->(p)

RETURN p.name AS ProductName

CREATE (gfAndCf:DietaryClustering {name: "GlutenFreeAndCaseinFree", description: "Products that contain no gluten or casein ingredients"})

WITH gfAndCf

// Finding products that don't have gluten, casein, milk, or dairy ingredients or allergens

MATCH (p:Product)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)

WHERE NOT (

    toLower(ir.ingredient\_name) CONTAINS 'gluten' OR

    toLower(ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten' OR

    toLower(ir.ingredient\_name) CONTAINS 'casein' OR

    toLower(ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN ir.alternative\_names | s + name + ' ')) CONTAINS 'casein' OR

    toLower(ir.ingredient\_name) CONTAINS 'milk' OR

    toLower(ir.scientific\_name) CONTAINS 'milk' OR

    toLower(ir.ingredient\_name) CONTAINS 'dairy' OR

    toLower(ir.scientific\_name) CONTAINS 'dairy'

)

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "Gluten"})

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "casein"})

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "milk"})

AND NOT (ir)-[:HAS\_ALLERGEN]->(:Allergen {name: "dairy"})

WITH gfAndCf, p, ir

OPTIONAL MATCH (ir)-[:HAS\_COMPOSITION]->(comp:Composition)

WITH gfAndCf, p, COLLECT(comp) AS comps

WHERE ALL(comp IN comps WHERE NOT (

    toLower(comp.description) CONTAINS 'gluten' OR

    toLower(comp.description) CONTAINS 'casein' OR

    toLower(comp.description) CONTAINS 'milk' OR

    toLower(comp.description) CONTAINS 'dairy'

))

WITH DISTINCT p, gfAndCf

MATCH (p)-[:HAS\_RECIPE]->(:Recipe)-[:USES\_INGREDIENT]->(:Ingredient)-[:HAS\_RESEARCH]->(ir:IngredientResearch)-[:has\_ingredients]->(:IngredientSource)-[:SOURCED\_FROM]->(t3ir:T3IngredientResearch)

WHERE NOT (

    toLower(t3ir.name) CONTAINS 'gluten' OR

    toLower(t3ir.scientific\_name) CONTAINS 'gluten' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'gluten' OR

    toLower(t3ir.name) CONTAINS 'casein' OR

    toLower(t3ir.scientific\_name) CONTAINS 'casein' OR

    toLower(reduce(s = '', name IN t3ir.alternative\_names | s + name + ' ')) CONTAINS 'casein' OR

    toLower(t3ir.name) CONTAINS 'milk' OR

    toLower(t3ir.scientific\_name) CONTAINS 'milk' OR

    toLower(t3ir.name) CONTAINS 'dairy' OR

    toLower(t3ir.scientific\_name) CONTAINS 'dairy'

)

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "Gluten"})

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "casein"})

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "milk"})

AND NOT (t3ir)-[:HAS\_ALLERGEN\_INFO]->(:Allergen {name: "dairy"})

WITH gfAndCf, p, t3ir

OPTIONAL MATCH (t3ir)-[:HAS\_COMPOSITION]->(comp:Composition)

WITH gfAndCf, p, COLLECT(comp) AS t3Comps

WHERE ALL(comp IN t3Comps WHERE NOT (

    toLower(comp.description) CONTAINS 'gluten' OR

    toLower(comp.description) CONTAINS 'casein' OR

    toLower(comp.description) CONTAINS 'milk' OR

    toLower(comp.description) CONTAINS 'dairy'

))

// Create relationship to mark the products as GlutenFreeAndCaseinFree

MERGE (gfAndCf)-[:GlutenFreeAndCaseinFree\_To]->(p)

RETURN p.name AS ProductName

CREATE (newGlutenCaseinAllergic:CustomizedRecommendationEngine {name: "Gluten&CaseinAllergic", description: "Customized screening for those with Gluten&Casein allergies."})

WITH newGlutenCaseinAllergic

// Step 3: Connect the new node to all products connected to GlutenFreeAndCaseinFree

MATCH (gfAndCf:DietaryClustering {name: "GlutenFreeAndCaseinFree"})-[:GlutenFreeAndCaseinFree\_To]->(product:Product)

MERGE (newGlutenCaseinAllergic)-[:Recommended\_To]->(product)

RETURN product.name