CPT103 COURSEWORK REPORT

MUST READ

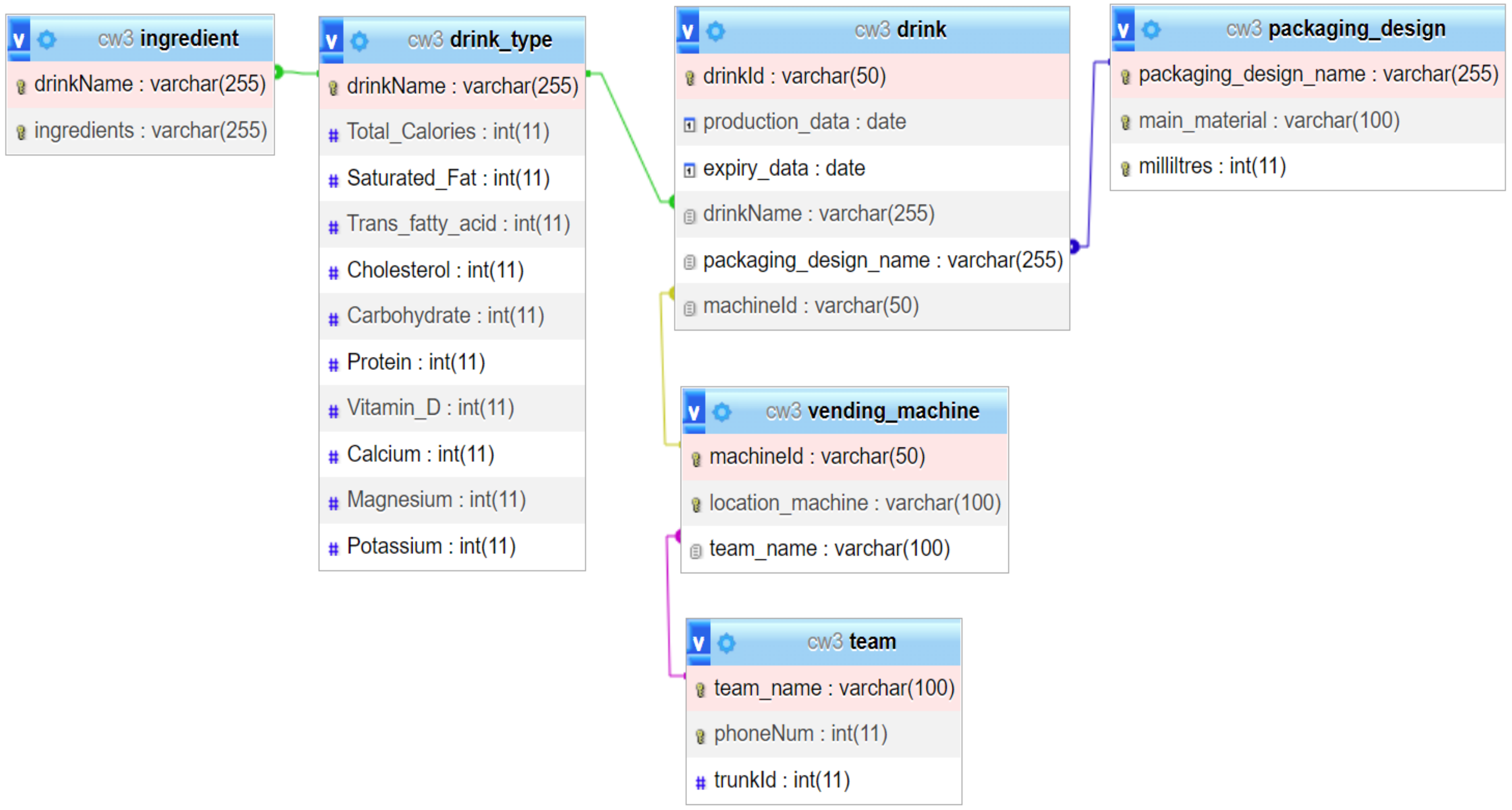
Your report and database design must be your own work. You should not copy any code from others or let anyone develop this database. *Plagiarism and collusion lead to a zero mark for this coursework*.

All tables must be in 3NF and the ER diagram should not contain any M:N or 1:1 relationships. ***Please strictly follow the structure of this template. Using your own template is not allowed.*** Remember to double-check grammar and wording errors so that the report could be understood correctly. *Any language other than English will be ignored when marking the report*.

Team information:

ID	Name	Email
2142112	Duanru Li	Duanru.Li21@student.xjtlu.edu.cn
2141522	Rongjie Mao	Rongjie.Mao21@student.xjtlu.edu.cn
2145668	Mingzhe Li	Mingzhe.Li21@student.xjtlu.edu.cn
2142261	Xinyao Chen	Xinyao.Chen21@student.xjtlu.edu.cn
2145618	Mingyuan Li	Mingyuan.Li21@student.xjtlu.edu.cn
2143873	Dongling Wang	Dongling.Wang21@student.xjtlu.edu.cn

Your ER Diagram for Task 1 here. Make sure this ER diagram fits one single page and is clear to read. Do not split the diagram into several pages.



DATABASE DESIGN DETAILS (TASK 1)

TABLES

Table name: drink_type

Table design explanation:

The drink_type table is used to store the information on the drink name and its corresponding nutrient data. The primary key is drinkName as it is unique to avoid repeating each drink name. Meantime, one drink type corresponds to one nutrient list(100ml), therefore the nutrient information is in the same table as the drink type. And all the nutrition items to be considered are constant, so each nutrient is a separate column. All nutrients default to 0.

Attributes:

Column Definition	Domain	Explanation
drinkName VARCHAR(255) PRIMARY KEY	All valid drink names are available. For example, “Pressed Coconut Milk”	The name of the drink. This is a unique identifying column for each type of drink. Once a drink type needs to be added, add information to this column. Inputs can only be checked manually
Total_Calories INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Total Calories. Default to 0 if not filled in. And input can only be checked manually
Saturated_Fat INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Saturated Fat. Default to 0 if not filled in. And input can only be checked manually
Trans_fatty_acid INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Trans-fatty acid. Default to 0 if not filled in. And input can only be checked manually
Cholesterol INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Cholesterol. Default to 0 if not filled in. And input can only be checked manually
Carbohydrate INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Carbohydrate. Default to 0 if not filled in. And input can only be checked manually

Protein INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Protein. Default to 0 if not filled in. And input can only be checked manually
Vitamin_D INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Vitamin D. Default to 0 if not filled in. And input can only be checked manually
Calcium INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Calcium. Default to 0 if not filled in. And input can only be checked manually
Magnesium INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Magnesium. Default to 0 if not filled in. And input can only be checked manually
Potassium INT DEFAULT 0	Any integer is allowed if it is valuable	Value of nutrition item for Potassium. Default to 0 if not filled in. And input can only be checked manually

Foreign keys and reasons:

No foreign keys in this table.

Table name: team

Table design explanation:

The table is used to store the information on names of the vending machine maintenance teams and the corresponding phone numbers and truck information. The primary key is the team_name as it is unique to avoid repeating each team name. And phone number should be unique because sharing of mobile phone numbers between teams is not allowed.

Attributes:

Column Definition	Domain	Explanation
team_name VARCHAR(100) PRIMARY KEY	All valid team names are available	The name of the team. This is a unique identifying column for each team. Once a team needs to be added, add information to this column. Inputs can only be checked manually

phoneNum INT UNIQUE NOT NULL	Any integer is allowed if number is valuable	Phone number of the team. Inputs can only be checked manually. And it can’t be null as every team must have a phone number
trunkId INT NOT NULL	Any integer is allowed.	The id of each trunk. Inputs can only be checked manually. This is related to the expansion of
		others. It can’t be null as every team must have a trunk

Foreign keys and reasons:

No foreign keys in this table.

Table name: vending_machine

Table design explanation:

The table is used to store the information on the vending machine ids, its locations and the name of the team managing it. The primary key is machineId as it is unique to avoid repeating each id. The team_name should come from team table, and this is to ensure that each team maintains a different list of machines. And team_name can be repeated because one team can maintain multiple machine.

Attributes:

Column Definition	Domain	Explanation
machineId VARCHAR(50) PRIMARY KEY	All valid ids are available	This is a unique identifying column for each machine. Inputs can only be checked manually
location_machine VARCHAR(100) UNIQUE NOT NULL	All valid location names are available	The location of the machine. The location of the machine must be different thus location_machine should be unique. And it can’t be null as every machine must have a location.
team_name VARCHAR(100) NOT NULL	All valid team names come from team table are available	The name of the team. It should come from the team table and therefore be consistent with that column in the team table.

Foreign keys and reasons:

The column team_name references team.team_name, this is to ensure that the table's team_name must come from the team_name column of the team table because the database user can't make up team_name.

Table name: packaging_design

Table design explanation:

The table is used to store the information on drink packaging design names, its main material used, and milliltres. The primary key is packaging_design_name as it is unique to avoid repeating each name of the packaging design. One main material with one millilitres to define a packaging name. In addition, to avoid duplication of combinations of packaging names, main materials, and milliltres, there exists a unique constraint of these three columns.

Attributes:

Column Definition	Domain	Explanation
packaging_design_name VARCHAR(255) PRIMARY KEY	All valid packaging design names are available	The name of packaging design. This is a unique identifying column for each packaging design. Inputs can only be checked manually
main_material VARCHAR(100) NOT NULL	All valid location names are available	The name of the main material of this packaging design. It can't be null as every packaging design must have a corresponding main material.
milliltres INT NOT NULL CHECK(milliltres>0)	Any natural number is allowed	The milliltres value of this packaging design. It can't be null as every packaging design must have a corresponding milliltres. And no package has a capacity of 0 or a negative number, so it must greater than 0.

Foreign keys and reasons:

No foreign keys in this table.

<u>Table name</u>:drink

Table design explanation:

The table is used to store the information on every drink produced by Ocean dew, which includes its production serial, the production date, the expiry date, the packaging design and the machine id which contains it. The primary key is drinkId as it is unique to avoid repeating each id of the drink. Multiple drinks can be of the same drink type and this relationship also applies to packaging design and machine ID, therefore, drinkName, packaging_design_name and machineId can be repeated.

Attributes:

Column Definition	Domain	Explanation
drinkId VARCHAR(50) PRIMARY KEY	All valid ids are available. For example, "A001"	The production serial of one drink. This is a unique identifying column for each drink. Inputs can only be checked manually.
production_date DATE NOT NULL	Satisfy the DATE type is allowed	This should match the DATE expression. It can't be null as every drink must have a corresponding production_date.

expiry_date DATE NOT NULL	Satisfy the DATE type is allowed	This should match the DATE expression. It can't be null as every drink must have a corresponding expiry_date.
drinkName VARCHAR(255) NOT NULL	All valid drink names come from the drink_type are available.	The drink type of this drink. It should come from the drink_type table and therefore be consistent with that column in the drink_type table.
packaging_design_name VARCHAR(255) NOT NULL	All valid packaging design names come from the packaging_design table are available	The packaging design of this drink. It should come from the packaging_design table and therefore be consistent with that column in the packaging_design table.
machineId VARCHAR(50)	All valid ids come from the vending_machine are available	The machine ID that contains this drink. It should come from the vending_machine table and therefore be consistent with that column in the vending_machine table. Besides, drinks are not always in the machine, thus machineId can be null.

Foreign keys and reasons:

The column drinkName references drink_type.drinkName, this is to ensure that the table's drinkName must come from the drinkName column of the drink_type table because the database user can't make up drinkName.

The column packaging_design_name references packaging_design.packaging_design_name, this is to ensure that the table's packaging_design_name must come from the packaging_design_name column of the packaging_design table because the database user can't make up packaging_design_name.

The column machineId references vending_machine.machineId, this is to ensure that the table's machineId must come from the machineId column of the vending_machine table because the database user can't make up machineId.

Table name: ingredient

Table design explanation:

The table is used to store the information on details of ingredients for each drink. One drink can contain one or more ingredients and one ingredient can be used in multiple drinks. Therefore, both column in this table can be repeated. To avoid repetition on two rows, there exists a unique constraint of these tow columns.

Attributes:

Column Definition	Domain	Explanation

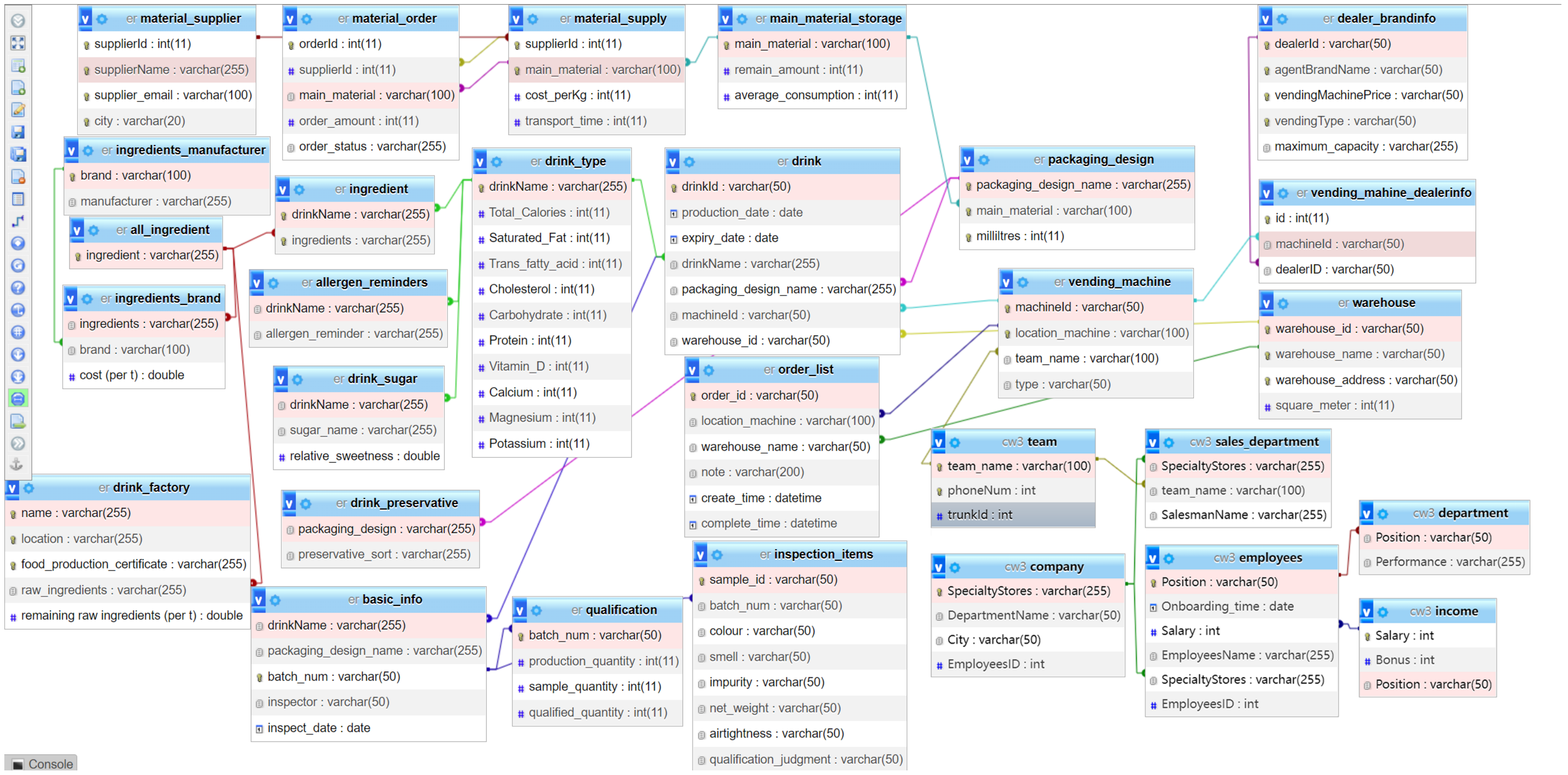
drinkName VARCHAR(255) NOT NULL	All valid drink names come from the drink_type are available.	The drink type of this drink. It should come from the drink_type table and therefore be consistent with that column in the drink_type table.
ingredients VARCHAR(255) NOT NULL	All valid ingredient names are available	The name of the ingredient. It can’t be null as every row must have a specific ingredient name.

Foreign keys and reasons:

The column drinkName references drink_type.drinkName, this is to ensure that the table's drinkName must come from the drinkName column of the drink_type table because the database user can't make up drinkName.

[add more blocks if needed]

Your ER Diagram for Task 2 here. Make sure this ER diagram fits one single page and is clear to read. Do not split the diagram into several page



er **material_supplier**
supplierId : int(11)
supplierName : varchar(255)
supplier_email : varchar(100)
city : varchar(20)

er **material_order**
orderId : int(11)
supplierId : int(11)
main_material : varchar(100)
order_amount : int(11)
order_status : varchar(255)

er **material_supply**
supplierId : int(11)
main_material : varchar(100)
cost_perKg : int(11)
transport_time : int(11)

er **main_material_storage**
main_material : varchar(100)
remain_amount : int(11)
average_consumption : int(11)

er **dealer_brandinfo**
dealerId : varchar(50)
agentBrandName : varchar(50)
vendingMachinePrice : varchar(50)
vendingType : varchar(50)
maximum_capacity : varchar(255)

er **ingredients_manufacturer**
brand : varchar(100)
manufacturer : varchar(255)

er **all_ingredient**
ingredient : varchar(255)

er **ingredient**
drinkName : varchar(255)
ingredients : varchar(255)

er **drink_type**
drinkName : varchar(255)
Total_Calories : int(11)
Saturated_Fat : int(11)
Trans_fatty_acid : int(11)
Cholesterol : int(11)
Carbohydrate : int(11)
Protein : int(11)
Vitamin_D : int(11)
Calcium : int(11)
Magnesium : int(11)
Potassium : int(11)

er **drink**
drinkId : varchar(50)
production_date : date
expiry_date : date
drinkName : varchar(255)
packaging_design_name : varchar(255)
machineId : varchar(50)
warehouse_id : varchar(50)

er **packaging_design**
packaging_design_name : varchar(255)
main_material : varchar(100)
millilitres : int(11)

er **vending_mahine_dealerinfo**
id : int(11)
machineId : varchar(50)
dealerID : varchar(50)

er **ingredients_brand**
ingredients : varchar(255)
brand : varchar(100)
cost (per t) : double

er **allergen_reminders**
drinkName : varchar(255)
allergen_reminder : varchar(255)

er **vending_machine**
machineId : varchar(50)
location_machine : varchar(100)
team_name : varchar(100)
type : varchar(50)

er **warehouse**
warehouse_id : varchar(50)
warehouse_name : varchar(50)
warehouse_address : varchar(50)
square_meter : int(11)

er **drink_sugar**
drinkName : varchar(255)
sugar_name : varchar(255)
relative_sweetness : double

er **order_list**
order_id : varchar(50)
location_machine : varchar(100)
warehouse_name : varchar(50)
note : varchar(200)
create_time : datetime
complete_time : datetime

cw3 **team**
team_name : varchar(100)
phoneNum : int
trunkId : int

cw3 **sales_department**
SpecialtyStores : varchar(255)
team_name : varchar(100)
SalesmanName : varchar(255)

er **drink_factory**
name : varchar(255)
location : varchar(255)
food_production_certificate : varchar(255)
raw_ingredients : varchar(255)
remaining raw ingredients (per t) : double

er **drink_preservative**
packaging_design : varchar(255)
preservative_sort : varchar(255)

cw3 **department**
Position : varchar(50)
Performance : varchar(255)

er **inspection_items**
sample_id : varchar(50)
batch_num : varchar(50)
colour : varchar(50)
smell : varchar(50)
impurity : varchar(50)
net_weight : varchar(50)
airtightness : varchar(50)
qualification_judgment : varchar(50)

cw3 **company**
SpecialtyStores : varchar(255)
DepartmentName : varchar(50)
City : varchar(50)
EmployeesID : int

cw3 **employees**
Position : varchar(50)
Onboarding_time : date
Salary : int
EmployeesName : varchar(255)
SpecialtyStores : varchar(255)
EmployeesID : int

er **basic_info**
drinkName : varchar(255)
packaging_design_name : varchar(255)
batch_num : varchar(50)
inspector : varchar(50)
inspect_date : date

er **qualification**
batch_num : varchar(50)
production_quantity : int(11)
sample_quantity : int(11)
qualified_quantity : int(11)

cw3 **income**
Salary : int
Bonus : int
Position : varchar(50)

Console

DATABASE EXTENSION DETAILS (TASK 2)

In this section, each team member needs to describe the new requirement in details and list a few related use cases. The new requirement and use cases should have a similar complexity level like requirement 2 and should have practical value. Remember to write down your name and ID. Each person has **a page limit of 5**. Please do not change the font size or line spacing or paragraph spacing.

REQUIREMENT 1 (NAME: DUANRU LI, ID: 2142112)

TABLES INVOLVED

Table name: basic_info

Table design explanation:

The basic_info table is used to store basic information about the beverage name and its production and inspection. The primary key is the batch_num, because it is unique and used to identify and trace each batch of products. Each batch of products has a corresponding production date. At the same time, the inspected product batch has a corresponding inspector. The inspector corresponding to the uninspected product can be blank, and the inspection date is the same.

Attributes:

Column Definition	Domain	Explanation
drinkName VARCHAR(255) NOT NULL	All valid drink names are available. For example, "Pressed Coconut Milk".	The name of the drink. Once a drink type needs to be added, add information to this column. Inputs can only be checked manually.
packaging_design_name VARCHAR(255) NOT NULL	All valid packaging design names are available.	The name of packaging design. Inputs can only be checked manually.
batch_num VARCHAR(50) PRIMARY KEY	All valid identification numbers are available.	The production number of each batch of product. This is a unique identifying column f for each batch of products. Inputs can only be checked manually.
inspector VARCHAR(50)	All valid names are available.	The name of the inspector. It can be null because some beverage batches have not been inspected.
inspect_date DATE	Satisfy the DATE type is allowed.	This should match the DATE expression. It can be null because some beverage batches have not been inspected.

Foreign keys and reasons:

The column drinkName references drink.drinkName, this is to ensure that the table's. drinkName must come from the drinkName column of the drink table because the database user can't make up drinkName.

[add more blocks if needed]

Table name: qualification

Table design explanation:

This table is used to store the quantity of beverage production, spot check and qualification. The primary key is the batch_num, because it is unique and used to identify and trace each batch of products. Each batch of products has a corresponding production quantity. At the same time, the inspected product batch has corresponding sampling quantity and qualified quantity. The spot check quantity and qualified quantity corresponding to uninspected products can be blank.

Attributes:

Column Definition	Domain	Explanation
batch_num VARCHAR(50) PRIMARY KEY	All valid identification numbers come from basic_info table are available.	The production number of each batch of product. This is a unique identifying column f for each batch of products. It should come from the basic_info table and therefore be consistent with that column in the basic_info table.
production_quantity INT NOT NULL	All valid figures are available.	Quantity of each batch of products. It can't be null as each batch of products must have a corresponding production quantity.
sample_quantity INT	All valid figures are available.	Number of products sampled for inspection. It can be null because some beverage batches have not been inspected.
qualified_quantity INT	All valid figures are available.	Quantity of qualified products sampled for inspection. It can be null because some beverage batches have not been inspected.

Foreign keys and reasons:

The column batch_num references basic_info.batch_num, this is to ensure that the table's batch_num must come from the batch_num column of the basic_info table because the database user can't make up batch_num.

Table name: inspection_items

Table design explanation:

This table is used to store the information of the drinks sampled for inspection and the corresponding inspection items. The primary key is sample_id, because it is unique to identify each drink sampled. Each bottle of beverage corresponds to an inspection item result and has an overall qualification judgment. All inspection items are qualified by default.

Attributes:

Column Definition	Domain	Explanation
sample_id VARCHAR(50) PRIMARY KEY	All valid ids are available.	The number of each drink sampled. This is a unique label for each drink. Inputs can only be checked manually.
batch_num VARCHAR(50) NOT NULL	All valid identification numbers come from basic_info table are available.	The production number of each batch of product. It should come from the basic_info table and therefore be consistent with that column in the basic_info table.

colour VARCHAR(50) DEFAULT 'qualified'	Qualified or unqualified is available.	Result of colour inspection item. Default to qualified if not filled in and input can only be checked manually.
smell VARCHAR(50) DEFAULT 'qualified'	Qualified or unqualified is available.	Result of smell inspection item. Default to qualified if not filled in and input can only be checked manually.
impurity VARCHAR(50) DEFAULT 'qualified'	Qualified or unqualified is available.	Result of impurity inspection item. Default to qualified if not filled in and input can only be checked manually.
net_weight VARCHAR(50) DEFAULT 'qualified'	Qualified or unqualified is available.	Result of net_weight inspection item. Default to qualified if not filled in and input can only be checked manually.
airtightness VARCHAR(50) DEFAULT 'qualified'	Qualified or unqualified is available.	Result of airtightness inspection item. Default to qualified if not filled in and input can only be checked manually.
qualification_judgment VARCHAR(50) DEFAULT 'qualified'	Qualified or unqualified is available.	Result of overall inspection of drink. Default to qualified if not filled in and input can only be checked manually.

Foreign keys and reasons:

The column batch_num references basic_info.batch_num, this is to ensure that the table's batch_num must come from the batch_num column of the basic_info table because the database user can't make up batch_num.

DESCRIPTION OF REQUIREMENT (150 WORDS MINIMUM)

The quality of the product affects the advancement of the subsequent links, and the enterprise that controls the quality of the product will have a good market and gain good profits. Quality inspection can effectively improve the product qualification rate, ensure product quality, reduce the adverse effects caused by unqualified products and increase corporate profits. Factories will produce a large number of beverages every day, and if the quality of beverages can be effectively recorded and analyzed, problems can be targeted for improvement and internal work can be optimized.

This database hopes to organize and record the quality inspection of beverages, implement the product inspection management system, and provide feedback to the relevant management departments. Adopt sampling inspection, that is, a certain amount of beverage samples are randomly selected for quality inspection, so as to determine whether the product is qualified or not, and promptly dispose of non-conforming products. When a problem occurs it can be traced back to the corresponding person in charge, providing evidence for selling qualified products to customers and disposing of unqualified products.

USE CASES (5 NEEDED)

Use case 1: List the production batch numbers with a qualification rate greater than 90%.

SQL query for use case 1: SELECT batch_num, CONCAT(ROUND(qualified_quantity/sample_quantity *100,2),"%") AS qualification_rate FROM qualification HAVING qualification_rate >'90.00%';

Use case 2: List the serial number and production batch number of beverages with unqualified air tightness.

SQL query for use case 2:

SELECT sample_id, batch_num FROM inspection_items WHERE airtightness='unqualified';

Use case 3: List the production batch number and beverage type tested by inspector "A".

SQL query for use case 3:

SELECT batch_num, drinkName, packaging_design_name FROM basic_info WHERE inspector='A';

Use case 4: Find out the production batch number that has not been checked.

SQL query for use case 4:

SELECT batch_num FROM basic_info WHERE inspector IS NULL;

Use case 5: List the production batch number and beverage type with production quantity exceeding 10000.

SQL query for use case 5:

SELECT DISTINCT batch_num, drinkName, packaging_design_name FROM basic_info WHERE batch_num= SOME (SELECT batch_num FROM qualification WHERE production_quantity>10000);

REQUIREMENT 2 (NAME: RONGJIE MAO, ID: 2141522)

TABLES INVOLVED

Table name: company

Table design explanation: The company table is used to store the information of company Information. The primary key is SpecialtyStores as different companies have different specialtystores.

Attributes:

Column Definition	Domain	Explanation
SpecialtyStores varchar(255) NOT NULL PRIMARY KEY	All company-owned specialty stores are available	The names of all sales stores owned by the company.
DepartmentName varchar(50) NOT NULL	All corporate divisions are available	The name of department. All departments in the company.
City varchar(50) NOT NULL	Any city that exists is available	This is the city in which the company is located, the company will have branches in different cities.
EmployeesID int NOT NULL	Any integer is allowed if it is valuable	The IDs of all employees.

Foreign keys and reasons:

No foreign keys in this table.

Table name: sales_department

Table design explanation: The sales department table is used to store the information about the company's internal sales department. This table records the personnel information of the company's internal sales department.

Attributes:

Column Definition	Domain	Explanation
SpecialtyStores varchar(255) NOT NULL	All company-owned specialty stores from Company are available	The names of all sales stores owned by the company.
team_name varchar(100) NOT NULL	All valid team names come from team table are available	The name of the team. It should come from the team table and therefore be consistent with that column in the team table.

SalesmanName varchar(255) NOT NULL	All valid names are acceptable. For example, ‘Mike’.	The name of salesman. The names of all salespeople are recorded in this table

Foreign keys and reasons:

The column team_name references team. team_name, this is to ensure that the table's team_name

must come from the team_name column of the team table because the database user can't make up

team_name.

The column SpecialtyStores references company. SpecialtyStores, this is to ensure that the table's SpecialtyStores must come from the SpecialtyStores column of the company table because the database user can't make up the name of specialtystores.

team_name.

Table name: employees

Table design explanation: The employees table is used to store the information of company employees. The primary key is Position as position is the best way to identify each employee

Attributes:

Column Definition	Domain	Explanation
Position varchar(50) NOT NULL PRIMARY KEY	All company jobs are available	The name of position. This column is the position of the company employee.
Onboarding_time date NOT NULL	Satisfy the DATE type is allowed	The time of onboarding time. This column is used foe recording employee's entry time.
Salary int NOT NULL	Any integer is allowed if number is valuable	The monthly salary of employee.
EmployeesName varchar(255) NOT NULL	All valid names are acceptable. For example, ‘Mike’.	The name of employees. The names of all s employees are recorded in this table.
SpecialtyStores varchar(255) NOT NULL	All company-owned specialty stores from Company are available	The names of all sales stores owned by the company.
EmployeesID int NOT NULL	Any integer is allowed if number is valuable	The IDs of all employees.

Foreign keys and reasons:

The column SpecialtyStores references company. SpecialtyStores, this is to ensure that the table's SpecialtyStores must come from the SpecialtyStores column of the team table because the database user can't make up specialtystores's name.

The column Salary references Income. Salary, this is to ensure that the table's Salary must come from the Salary column of the team table because the database user can't make up the salary of employees.

Table name: income

Table design explanation: The income table is used to store the information of employee income. The primary key is salary as employee salaries are different for everyone in the vast majority of cases.

Attributes:

Column Definition	Domain	Explanation
Salary int NOT NULL PRIMARY KEY	Any integer from table employees is allowed if number is valuable	The monthly salary of employee.
Bonus int DEFAULT 0	Any integer is allowed if number is valuable	The bonus of per employee.
Position varchar(50) NOT NULL	All company jobs from table employees are available	The name of position. This column is the position of the company employee.

Foreign keys and reasons:

No foreign keys in this table.

Table name: department

Table design explanation: The department table is used to store the information of departments. It contains the position and performance of employees, which can better record the situation of the department.

Attributes:

Column Definition	Domain	Explanation
Position varchar(50) NOT NULL	All company jobs from table employees are available	The name of position. This column is the position of the company employee.

Performance varchar(255) NOT NULL	any descriptions of performance are available	Some words to describe performance. This column describes the department's recent performance in a few words.

Foreign keys and reasons:

No foreign keys in this table.

DESCRIPTION OF REQUIREMENT (150 WORDS MINIMUM)

The following five requirements are studied for this database from the city where the sales store is located, the salary of the employee, the team name, and the time of entry. The first example is the city where the query is located, which reflects the geographic location of the company. The second and third examples both examine employees and can help company leaders grasp employee compensation. The fourth example lists the names of the teams in the sales department, which is convenient for the sales department to manage the teams below. The fifth example is the inspection of employees' entry time, which can distinguish new and old employees well and facilitate personnel management. In general, the following five examples can help leaders manage teams and employees well, so that the improvement of the database can be seen. At the same time, after inquiries, we can see the diversity of database data, and make statistics on employee information from various aspects.

USE CASES (5 NEEDED)

Use case 1: List all specialty stores in Shanghai

SQL query for use case 1:

SELECT SpecialtyStores FROM company WHERE City=Shanghai;

Use case 2: Set the software programmer salary to 2000

SQL query for use case 2:

INSERT INTO Income (Position, Salary) VALUES (' software programmer ', 2000);

Use case 3: Find employees with salary greater than 2000

SQL query for use case 3: SELECT EmployeesName FROM employees WHERE Salary>2000;

Use case 4: List the team names in the sales department

SQL query for use case 4:

SELECT team_name FROM sales_department;

Use case 5: List the names of employees who have been employed for 100 days

SQL query for use case 5:

```
SELECT EmployeesName FROM employees WHERE TO_DAYS(NOW()) - TO_DAYS(Onboarding_time) >
100;
```

REQUIREMENT 3 (NAME: MINGZHE LI, ID: 2145668)

TABLES INVOLVED

Table name: Table vending_machine_dealerInfo

Table design explanation:

Table vending_mahine_dealerInfo as an intermediate to link the table vending_machine and dealer_brandInfo, it contains two foreign keys linked to two primary keys from table vending_machine and table dealer_brandInfo.

Attributes:

Table vending_machine_dealerInfo

Column Definition	Domain	Explanation
id int auto_increment primary key	Only as a primary key	Setting as a primary key, doesn't really make sense.
machineId varchar(50)	All valid machines in Lukewarm Kingdom are available.	Corresponding to the id number of each vending machine
dealerID varchar(50)	All valid dealer's id is available.	Corresponding to the id number of each dealer.

Foreign keys and reasons:

alter table vending_mahine_dealerInfo ADD CONSTRAINT fk_delaer_brand foreign key (dealerID) references dealer_brandInfo(dealerId);

reasons: To link the dealer information and the intermediate table vending_machine_dealerInfo.

alter table vending_mahine_dealerInfo ADD CONSTRAINT fk_machine_dealer foreign key (machineId)references vending_machine(machineId);

reasons: To link the dealer information and the intermediate table vending_machine_dealerInfo.

Table name: Table dealer_brandInfo

Table design explanation:

Table vending_machine refers to the information of the vending machine, such as the machine id, location, etc. Table dealer_brandInfo refers the expanded information as mentioned before.

Column Definition	Domain	Explanation

dealerId varchar(50) not null	All valid dealer id is available.	Corresponding to the id number of each vending machine
agentBrandName varchar(50) not null	All valid brand name is available.	Corresponding to the name of the agent company behind each dealer.
vendingMachinePrice varchar(50)not null	All valid price of vending machine from each dealer is available.	Corresponding to the prices for different types of vending machines from different companies.
vendingType varchar(50) not null	All valid type of vending machine (online or offline type) is available.	Corresponding to the different type of the vending machine in different companies.
maximum_capacity varchar(255) not null	All valid maximum capacity form different type vending machine in different companies is available.	Corresponding to the maximum capacity the vending machine can hold.

Union primary key:

In table dealer_brandInfo set a union primary key from (dealerId,agentBrandName,vendingMachinePrice,vendingType)

Because the brand, vending machine type, price and dealer id can make up a variety of combinations, each of these different combinations can be represented as a primary key

Foreign key:

alter table vending_mahine_dealerInfo ADD CONSTRAINT fk_delaer_brand foreign key (dealerID) references dealer_brandInfo(dealerId);

reasons: To link the dealer information and the intermediate table vending_machine_dealerInfo. (Table vending_machine_dealerInfo is master table and dealer_brandInfo is slave table)

[add more blocks if needed]

DESCRIPTION OF REQUIREMENT (150 WORDS MINIMUM)

Different vending machines can be sold by different dealers, and different dealers can have different agents behind them. When business people decide to increase the use of input vending machines, in order to achieve the goal of obtaining the maximum revenue by using the lowest cost, it is first necessary to understand the prices of different brand types of vending machines and the attributes of different models of vending machines. Secondly, they should also understand the details of the vending machines that have been put into use and serve as a reference for the next investment decision.

Each dealer has a corresponding agent brand and each brand has many different types vending machine, for example, whether vending machines can be connected to the Internet for electronic payment, and different vending machine has different price. The aim of expansion is to simplify the information search process by integrating information about different brands of vending machines. Through these table can easily to find out or list the corresponding attribute.

USE CASES (5 NEEDED)

Use case 1: Find out the cheapest offline machine.

SQL query for use case 1:

select agentBrandName, vendingMachinePrice from dealer_brandInfo where vendingType = 'Offline type' order by vendingMachinePrice asc limit 1;

-- another way

select agentBrandName, min(vendingMachinePrice) from dealer_brandInfo where vendingType = 'offline type';

Use case 2: List the 'Offline Type' vending machine order by the capacity desc, price in delar_brandinfo.

SQL query for use case 2:

select maximum_capacity Maximum ,vendingMachinePrice 'Price', agentBrandName 'Brand' from dealer_brandinfo

where vendingType = 'offline type' order by maximum_capacity desc ,vendingMachinePrice asc ;

Use case 3: List the remaining capacity in the north vending machine.

SQL query for use case 3:

select distinct maximum_capacity - count(drinkName) 'RestNum', location_machine 'Location' from dealer_brandinfo dealer, vending_mahine_dealerInfo VD, drink , vending_machine V

where V.location_machine = 'north'

and VD.dealerID = dealer.dealerId

and V.type = dealer.vendingType

and V.machineId = VD.machineId

and drink.machineId = (select V.machineId where location_machine= 'north');

Use case 4:

SQL query for use case 4: How many drinks can the four vending machines hold in Lukewarm Kingdom.

```
select sum(maximum_capacity)  from dealer_brandinfo d, vending_mahine_dealerInfo info,
vending_machine v

where v.type = d.vendingType

and v.machineId = info.machineId

and d.dealerId = info.dealerID;
```

Use case 5: List the online vending machine where in the Lukewarm Kingdom type price from high to low.

SQL query for use case 5:

```
select  agentBrandName 'Brand', vendingType , vendingMachinePrice 'Price', location_machine
"Location" from dealer_brandInfo d,vending_mahine_dealerInfo info,vending_machine v

where type = 'Online Type'

and Info.dealerID = d.dealerId

and v.type = d.vendingType

and v.machineId = Info.machineId;
```

REQUIREMENT 4 (NAME: XINYAO CHEN, ID: 2142261)

TABLES INVOLVED

Table name: warehouse

Table design explanation:

The warehouse table is used to store the information of all Ocean Dew warehouse numbers, names, addresses, and areas in square meters. The primary key is warehouse_id as it is unique for every warehouse and avoids repeating each number. Each warehouse name and address are unique, so they need a UNIQUE key. Every warehouse has its own area data, so the value must not be null, but there may be duplicate area data. Ocean Dew can insert a new warehouse number, name, address, and area data in this table when Ocean Dew creates a new warehouse. In addition, if the company modifies the warehouse number, size, and migration address, they can all update the data in this table. (After that, foreign keys will be added, and if `warehouse_name` need to be modified, the related foreign key needs to be deleted first. Modifying `warehouse_id` does not require removing the foreign key, for reasons that will be explained later.)

Attributes:

Column Definition	Domain	Explanation
warehouse_id VARCHAR(50) PRIMARY KEY	Warehouse numbers. For example, 'W101'.	It is the individual unique number of each Ocean Dew warehouse. Every warehouse has its own unique number.
warehouse_name VARCHAR(50) UNIQUE NOT NULL	All valid warehouse names are acceptable. For example, 'Great Ocean Warehouse'.	The name of warehouses. Each warehouse has its own unique name.
warehouse_address VARCHAR(50) UNIQUE NOT NULL	All valid warehouse addresses are acceptable. For example, '37 Xueyuan Road, Fengye district'.	The address of the warehouse. The data cannot be checked by the database directly. As a result, manual checking is required when entering data. Every warehouse has its own unique address.
square_meter INT NOT NULL	Any integer is allowed if it is valuable.	The size of the Ocean Dew warehouse and its unit is square meters. Each warehouse has an area, and the size of the area may be duplicated. Therefore, the value cannot be NULL.

Foreign keys and reasons:

No foreign keys.

Table name: drink (a table created in TASK1)

Table design explanation:

No new table is created, but a column `warehouse_id` is added. In addition to being in the vending machine, the drinks may also be in the warehouse. (The related SQL syntax is also displayed in the script.sql file.)

Attributes:

Column Definition	Domain	Explanation
warehouse_id VARCHAR(50)	Warehouse numbers. For example, 'W101'.	It is the individual unique number of each Ocean Dew warehouse. The value should come from the `warehouse` table. Besides, drinks are not always in the warehouse, thus warehouse_id can be null.

Foreign keys and reasons:

The column warehouse_id references warehouse.warehouse_id, this is to ensure that the warehouse_id are valid numbers that reflect existing warehouse numbers of the warehouse where some drinks are located in. The foreign key is set to be 'NO ACTION' on delete considering if one warehouse stored in table `warehouse` is renumbered, the records in `drink` are still valid. (It is set to be `CASCADE` on update though.)

Table name: order_list

Table design explanation:

The order_list table is used to store the information of all warehouses' orders, including number, address, note, warehouse name, order creation time, and order completion time. The primary key is order_id as it is unique for every order and avoids repeating each number. The attribute 'location_machine' is intended to indicate the location of the vending machine so that it is easy to find the delivery location and find the team which is responsible for this order. It means by using a SELECT query, this table can also provide the information of the team which is in charge of the order. Similarly, the attribute 'warehouse_name' is intended to indicate the shipment warehouse, so that it is easy to find the shipping warehouse information including the address.

Attributes:

Column Definition	Domain	Explanation
order_id VARCHAR (50) PRIMARY KEY	order numbers. For example, 'W2022113001'.	It is the individual unique number of each order of logistics transportation. Every order has its own unique number.
location_machine VARCHAR(100) NOT NULL	All valid location names are available. For example, 'Ocean Amusement Park'	The location of the machine. It can't be null as any order must have a place of receipt and every machine must have a location.
warehouse_name VARCHAR(50) NOT NULL	All valid warehouse names are acceptable. For example, 'Great Ocean Warehouse'.	The name of warehouses. Each warehouse has its own unique name.

note VARCHAR(200)	Logistics order postscript. For example, ‘glass bottles should be handled with care.’	Notes in the logistics order are generally optional. It may be added when the drinks are valuable or fragile.
create_time DATETIME NOT NULL	Time String. For example, ‘2022-11-10 17:30:00’.	The time when the order is created. It could not be NULL because this table is used to store the creation time and once a new order is created, there must be a new creation time.
complete_time DATETIME	Time String. For example, ‘2022-11-10 17:30:00’.	The time when the drink is delivered, in the case of the drinks have not been delivered yet, could be NULL.

Foreign keys and reasons:

The column `warehouse_name` references warehouse.warehouse_name. This is to make sure that all warehouse names in the table are valid names that reflect the existing names of the warehouses.

The column `location_machine` references vending_machine.location_machine. This is to make sure that all location_machine in the table are valid locations that reflect the existing locations of the vending machine.

DESCRIPTION OF REQUIREMENT (150 WORDS MINIMUM)

When the number of bottles or cans left for each drink’s packaging in a vending machine is less than or equal to 2, staff members in the team that is hired by Ocean Dew need to replenish and maintain the vending machine. If replenishment is needed, the staff member who tracks the quantity of currently available drinks will send an order to the nearest warehouse. The staff member in charge of the vending machine drives a truck to the nearest warehouse to transport the drink’s packaging for replenishment. In a warehouse, the earlier the production date, the earlier the drink's packaging is shipped, provided that each of them has not expired.

Each warehouse of Ocean Dew has a unique number, a unique name, and a unique address. Each warehouse has only one floor. They all have detailed area data in square meters. For example, the warehouse numbered "W101" has 500 square meters. As the area gets larger, the capacity of the warehouse gets larger.

Each time a staff member delivers a shipment, a logistics order is generated with a delivery address (which is the vending machine location), a unique delivery order number, an order note (optional), the shipment warehouse information, the order creation time, and the order completion time, which is the time the order is delivered (if it is not delivered, the value is null). If the staff responsible for transportation is unable to deliver within 24 hours inclusive, the team which is hired for the vending machine will be subject to a fine from Ocean Dew.

USE CASES (5 NEEDED)

Use case 1:

Find the number(s) of the latest created logistics order.

SQL query for use case 1:

SELECT order_id FROM order_list WHERE create_time = (SELECT MAX(create_time) FROM order_list);

Use case 2:

In the warehouse numbered "W101", sort the serial numbers of the Ocean Dew Beer cans in the order of shipment from early to last (assuming today is November 27, 2022).

SQL query for use case 2:

SELECT drinkId FROM drink WHERE warehouse_id = 'W101' AND packaging_design_name = 'Ocean Dew Beer can' AND expiry_date> '2022-11-27' ORDER BY production_date;

Use case 3:

Find out the IDs of all vending machines that need to be replenished and maintained, the names of the drink's packages that are out of stock, and their current quantities of available drinks.

SQL query for use case 3:

SELECT machineId, packaging_design_name, COUNT(*) AS 'QUANTITY' FROM `drink` GROUP BY packaging_design_name, machineId HAVING QUANTITY <=2;

Use case 4: Find out warehouse numbers with less than average storage capacity, as Ocean Dew wants to expand its warehouse.

SQL query for use case 4:

SELECT warehouse_id FROM warehouse WHERE warehouse.square_meter < (SELECT AVG(square_meter) FROM warehouse);

Use case 5:

List all the fined teams and the corresponding order number.

SQL query for use case 5:

SELECT team_name,order_id FROM vending_machine, order_list

WHERE vending_machine.location_machine = order_list.location_machine

```
AND order_list.order_id IN (SELECT order_id FROM order_list WHERE(
   (UNIX_TIMESTAMP(complete_time) - UNIX_TIMESTAMP(create_time))/(60*60*24) )> 1
                        OR complete_time IS NULL
                        AND( (UNIX_TIMESTAMP(now()) - UNIX_TIMESTAMP(create_time))/(60*60*24) )>1);
```

REQUIREMENT 5 (NAME: MINGYUAN.LI, ID: 2145618)

TABLES INVOLVED

Table name: main_material_storage

Table design explanation:

The main_material_storage table is used to store the information on all the main materials to be used and their current surplus and average consumption. The primary key is main_material as it is unique to avoid repeating each material. If a new material is added, add a line with the message main_material. In addition, there will be remain_amount and average_sonsumption if and only if main_material appears.

Attributes:

Column Definition	Domain	Explanation
main_material VARCHAR(100) PRIMARY KEY	All valid names are available. For example, ” aluminum”	The name of the main materials. This is a unique identifying column for each material. It can be a word or a phrase and only manual inspections are allowed.
remain_amount INT DEFAULT 0 CHECK(remain_amount>=0)	All Natural numbers are allowed.	The amount of each material left over. It can’t be negative because materials that exist objectively cannot be overdrawn. If the database user is not specified it, there will be a default value, that is 0.
average_consumption INT NOT NULL CHECK(average_consumption>=0)	All Natural numbers are allowed.	Average daily usage of each material. It can’t be negative because the amount of material will not increase after consumption. There must be a value here because materials are either consumed or not consumed.

Foreign keys and reasons:

No foreign keys in this table.

Table name: material_supplier

Table design explanation:

The material_supplier table is used to store the information of all the material supplier and their e-mail, and the city they are in. The supplierId is a primary key to identifying each unique supplier. The supplierId column is used to be identifying column instead of supplierName because the company with the same name has branches in different cities. Meantime, to ensure that there are no duplicate supplierName and city combinations, there exists a unique constraint that contains both supplierName and city.

Attributes:

Column Definition	Domain	Explanation
supplierId INT PRIMARY KEY	Any integer is allowed	It's the identifying column of suppliers. Its type is up to the database consumer, as long as it is an integer to ensure legibility.
supplierName VARCHAR(255) NOT NULL	All valid names are available.	The name of supplier. It can be repeated because the same supplier has branches in different cities. But it can't be null as every supplierId must have a supplierName.
supplier_email VARCHAR(100) UNIQUE NOT NULL	Valid email addresses are allowed.	The email addresses cannot be checked by the database, so manual checking is required. It can't be null as every supplierId must have an email and it must be unique to ensure accurate contact.
city VARCHAR(20) NOT NULL	All valid city names are available.	The name of the city that the supplier is in. It can't be null as every supplierId must have a city. It can be repeated because you can have different suppliers in the same city

Foreign keys and reasons:

No foreign keys in this table.

Table name: material_supply

Table design explanation:

The material_supply is used to store the information on the unit price of a material supplied by a supplier and the corresponding shipping time. The primary key is the composite key of suppliers and main_material, and this is to limit the combination of the same material with the supplier not to be repeated. The same material may be supplied by multiple suppliers, and one supplier may supply multiple materials, but often the transportation time is different, and the price is different, this is the reason for setting the composite primary key.

Attributes:

Column Definition	Domain	Explanation
supplierId INT NOT NULL	Any integer from material_supplier table is allowed.	The name of the supplier. It should come from the material_supplier table and therefore be consistent with that column in the material_supplier table.
main_material VARCHAR(100) NOT NULL	Any names from main_material_storage table are available.	The name of the main material. It should come from the main_material_storage table and therefore be consistent with that column in the main_material_storage table.
cost_perKg INT NOT NULL CHECK(cost_perKg>0)	Natural numbers except 0 are allowed.	This column stores the unit price of this type of material supplied by this supplier. Store in round numbers for quick price comparison. It has to be greater than zero because there's no free material or a losing business.

		And it cannot be null because the primary key exists.
transport_time INT NOT NULL CHECK(transport_time>=0)	Natural numbers are allowed.	This column stores the transit time (days) of this type of material supplied by this supplier. It could be zero because it could be transported in the same city but it can't be negative because time can only flow forward. And it cannot be null because the primary key exists.

Foreign keys and reasons:

The column supplierId references material_supplier.supplierId, this is to ensure that the table's supplierId must come from the supplierId column of the material_supplier table because the database user can't make up supplierId. Ensure data integrity and consistency.

The column main_material references main_material_storage.main_material, this is to ensure that the table's main_material must come from the main_material column of the main_material_storage table because the database user can't make up main_material. Ensure data integrity and consistency

Table name: material_order

Table design explanation:

The material_order is used to store the information about details of current material orders to suppliers. The primary key is orderId as there needs a unique number to represent one specific order. This table is the core of the extension because it is the order that undertakes the task of increasing the material quantity. When purchasing materials, a drink company can place multiple orders from the same supplier or buy the same material from multiple suppliers, all depending on shipping time and current price. Better purchasing planning can be made by checking material_order.

Attributes:

Column Definition	Domain	Explanation
orderId INT PRIMARY KEY	Any rational number is allowed	The number represents the order. It is the identification column for the order.
supplierId INT NOT NULL	Any integer from material_supply table is allowed.	The name of the supplier in this order. It should come from the material_supply table and therefore be consistent with that column in the material_supply table.
main_material VARCHAR(100) NOT NULL	Any names from material_supply table are available.	The name of the main material in this order. It should come from the material_supply table and therefore be consistent with that column in the material_supply table.
order_amount INT NOT NULL CHECK(order_amount>0)	Any natural number is allowed	The quantity of materials specified in the order. This value must not be null after the order is created and it cannot be negative or zero because that would make the order meaningless.

order_status VARCHAR(255) NOT NULL	Any words valid are allowed	Displays the status of the order. Only human inspections are allowed, and it can insert any words that can represent status. It can be an exact time remaining for delivery or a macro description like in transit, not shipped

Foreign keys and reasons:

The columns supplierId and main_material reference material_supply.supplierId and material_supply.main_material, this is to ensure that the table's combination of supplierId and main_material must come from the combination of supplierId and main_material of the material_supply table because the database user can't make up a combination of supplierId and main_material. Ensure data integrity and consistency.

[add more blocks if needed]

DESCRIPTION OF REQUIREMENT (150 WORDS MINIMUM)

To save costs, Ocean Dew decided to order its own materials to produce its drink packaging. Since the same material can be supplied by multiple suppliers, the same supplier can supply multiple materials, and the transportation time of materials provided by different suppliers are different, it is necessary to record the kind of materials provided by suppliers, their unit price, and the transportation time. A material inventory table is also required to record material name, surplus, and average usage. In addition, the database user needs to see the current order status, which includes the order number, the amount of materials ordered from a supplier, and the order status.

Ocean Dew will be looking for new suppliers or will need to use new materials in the future, so suppliers and materials can be changed. By using the database, managers can develop better purchasing plans for materials

USE CASES (5 NEEDED)

Use case 1: Find out the name of the supplier that is located in city001

SQL query for use case 1:

```
        SELECT DISTINCT supplierName FROM material_supplier WHERE city = 'city001';
```

Use case 2: Find the amount of material left over after a day of use with an average consumption of more than 100 and rank it from highest to lowest

SQL query for use case 2:

```
        SELECT remain_amount-average_sonsumption AS OneDayuse FROM main_material_storage
WHERE average_sonsumption > 100 ORDER BY  OneDayuse DESC;
```

Use case 3: Find the names, ids and material prices of all the companies that supply aluminum and rank them from highest to lowest

SQL query for use case 3:

```
SELECT ly.supplierId,supplierName,cost_perKg FROM material_supplier AS er,material_supply AS ly
WHERE (er.supplierId = ly.supplierId) AND main_material = 'aluminum' ORDER BY cost_perKg DESC;
```

Use case 4: Find the status and shipping time of the material for order No. 003

SQL query for use case 4:

```
SELECT transport_time, order_status FROM material_supply AS s INNER JOIN material_order AS o ON
s.supplierId = o.supplierId AND s.main_material = o.main_material AND orderId = 003;
```

Use case 5: Select the material names of the various materials with a transit time of less than 4 and the corresponding total quantity ordered

SQL query for use case 5:

```
SELECT o.main_material,SUM(order_amount) AS Sum FROM material_order AS o, material_supply AS
s WHERE (o.main_material = s.main_material AND o.supplierId = s.supplierId) AND transport_time <=
4 GROUP BY o.main_material;
```

REQUIREMENT 6 (NAME: WANGDONGLING, ID: 2143873)

TABLES INVOLVED

Table name: drink_preservative

Table design explanation: the drink_preservative table is used to store different packaging of different types of drinks may correspond to the use of a variety of preservatives. The primary key is packaging_design as every drink with different packaging may use different preservatives, such as the bottle and the can use different preservative because they made in different material. A drink may without any preservative, which means the column `preservative sort` is null.

Attributes:

Column Definition	Domain	Explanation
Packaging_degisn varchar(255)	Packaging design (contain drink type)	Every name of packaging design contains corresponding drink name. As a result, checking preservatives of a drink or a packaging both acceptable.
Preservative_sort varchar(255)	All valid preservative types, such as "Sodium benzoate"	The preservatives for drink production. This column may null as for one drink there may not contain preservatives

Foreign keys and reasons:

The column packaging_design references packaging_design.packaging_design_name, this is to make mure that packaging_design are valid that reflect preservatives of every packaging.

Table name: allergen_reminder

Table design explanation: the allergen_reminder table is used to store the allergen of each drink. Allergies to customers may occur during raw ingredients and processing, so it is must be checked by the producer and customer. The primary key is drinkName.

Attributes:

Column Definition	Domain	Explanation
drinkName varchar(255)	6+ valid type of drink	Drink types.
Allergen_reminder varchar(255)	Allergen reminder, such as "This product may contain peanuts and soy products"	Remind to customer allergen of every drink, and can be checked directly by customer. This is possible null as one drink may not contain allergen.

Foreign keys and reasons:

The column drinkName references drink_type.drinkName, this is to make sure that drinkName are valid that reflect allergen reminder of every drink.

Table name: drink_sugar

Table design explanation: the drink_ sugar table is used to store the sweetener of each drink. sugar to customers may occur during raw ingredients and processing, so it is must be checked by the customer who control their daily sugar intake. The primary key is drinkName, and relative_sweetness is unique for each sugar.

Attributes:

Column Definition	Domain	Explanation
drinkName varchar(255)	6+ valid type of drink	Drink types.
sugar_name varchar(255)	All valid Sweetener names, such as "Maltose"	Remind to customer the sweetener of every drink, and can be checked directly by customer. This is possible null as one drink may not contain sweetener.
relative_sweetness double	Such as the relative sweetness of maltose is 0.5.	Corresponding to unique sugar (Sucrose 1.0)

Foreign keys and reasons:

The column drinkName references drink_type.drinkName, this is to make sure that drinkName are valid that reflect allergen reminder of every drink.

Table name: all_ingredient

Table design explanation: the all_ingredient table is used to store each ingredient. The primary key is ingredient.

Attributes:

Column Definition	Domain	Explanation
Ingredient varchar(255)	All valid preservative types, such as "Pure water"	The preservatives for drink production.

Foreign keys and reasons:

No foreign key. The ingredient.ingredients references all_ingredient.ingredient, this is make sure each ingredient can be checked.

Table name: ingredients_brand

Table design explanation: The ingredients_brand table is used to store the the brand of the material that makes up the composition and its unit price per ton, and used for drink ingredients purchasing department.

Attributes:

Column Definition	Domain	Explanation
Ingredients varchar(255)	All valid preservative types, such as "Pure water"	The preservatives for drink production.
Brand varchar(100)	Valid ingredient brand, such as "SUNTORY"	The brand of the raw ingredient of the ingredient before production. This column is not null because it is illegal factory use no brand ingredients to product the drink.
Cost (per t) double	Valid Unit price of each brand of ingredients, such as 30.1	Purchasing department use this column to purchase ingredients and their cost are comparable. This column is not unique because cost of two different brand may same.

Foreign keys and reasons:

The column ingredients references all_ingredient.ingredient, this is make sure every ingredient can be checked.

The column brand references ingredients_manufacturer.brand , this is make sure the only manufacturer corresponding to the same brand can be found.

Table name: ingredients_manufacturer

Table design explanation: the ingredients_manufacturer table is used to store the manufacturer of each ingredient brand. The primary key is brand.

Attributes:

Column Definition	Domain	Explanation
Brand varchar(100)	Valid ingredient brand, such as "SUNTORY"	The brand of the raw ingredient of the ingredient before production.

Manufacturer varchar(255)	Manufacturers name, such as “Toyo Beverage Co., Ltd.”	All valid manufacturer for each brand.

Foreign keys and reasons:

No foreign key.

Table name: drink_factory

Table design explanation: The table drink_factory is used to store all factory producing drink, this can be checked by manufacturers and consumers and some data printed on each drink packaging. The primary key is name and the column name, location and food_production_certificate is unique.

Attributes:

Column Definition	Domain	Explanation
Name varchar(255)	All valid factory name	The name of factorys.
Location varchar(255)	Building YY, ZZ Road, WW	The data can be checked by the database directly.
Food_production_certificate varchar(255)	Generally, is combination of uppercase letters and numbers, such as “SC10632058100011”	Record the food production license number of each manufacturer.
Raw_ingredients varchar(255)	All valid preservative types, such as “Pure water”	The preservatives for drink production. This column is not null because the if the ingredients remain equals zero, this column must be existed.
Remaining raw ingredients (per t) double	Record how much of the raw ingredient is left in this factory, such as 200.1	The data can be checked by the purchasing department using purchase ingredients conveniently.

Foreign keys and reasons:

The column raw_ingredients references all_ingredient.ingredient, this can be make sure that ingredients are valid name that reflect existing raw ingredients in factory.

[add more blocks if needed]

DESCRIPTION OF REQUIREMENT (150 WORDS MINIMUM)

The route of this expansion table is essentially the route of drink raw ingredients

1. Some customers may need to regulate their daily sugar intake, pay attention to the danger of excessive intake of preservatives to the human body, and personal allergic responses, therefore consumers may directly query these substances through the database.

2. Beverage producers and raw material brands are largely used by merchants and customers. Customers have the right to know about the maker. And Merchants can check the legality of a factory and the number of materials produced using the database, and purchase goods for factory.

3. The buying department can analyze the manufacturers with the most purchased brands to apply for a particular number of discounts to cut expenses.

4. All raw ingredients are provided individually for use as intermediary tables in these tables.

USE CASES (5 NEEDED)

Use case 1: List all drink with no sweetener and no preservative

SQL query for use case 1: SELECT drinkName FROM drink_sugar WHERE sugar_name = null AND drinkName= (SELECT drinkName FROM drink,drink_preservative WHERE (drink.packaging_design_name=drink_preservative.packaging_design) AND (preservative_sort=null));

Use case 2: Set the allerger reminder that is “This drink may has peanuts, soybeans” for the drink” Pressed Coconut Milk”

SQL query for use case 2: UPDATE allergen_reminders SET allergen_remider= 'This drink may has peanuts, soybeans' WHERE drinkName= 'Pressed Coconut Milk';

Use case 3: List the manufacturers with the greatest number of brands

SQL query for use case 3: SELECT manufacturer FROM ingredients_manufacturer GROUP BY manufacturer Having count(brand)=(SELECT COUNT(brand) FROM ingredients_manufacturer GROUP BY manufacturer ORDER BY 1 DESC LIMIT 1);

Use case 4: List all manufacturers without food production licenses in Mario Street

SQL query for use case 4: SELECT * FROM drink_factory WHERE food_production_certificate = null AND location LIKE '%Mario%';

Use case 5: Figure out how much cost when Suntory brand pure water can be imported into a factory called Fallos until its pure water reaches 2,000 tons

SQL query for use case 5: SELECT (SELECT `cost (per t)` FROM ingredients_brand WHERE ingredients='pure water' AND brand='Suntory') * 2000-(SELECT `remaining raw ingredients (per t)` FROM drink_factory WHERE raw_ingredients='pure water') AS cost FROM drink_factory, ingredients_brand;