Vending Machine DB System

- Fengyan Dong

Purpose

The purpose of this project is using SQL to manipulate and retrieve data in database to serve the purpose of business management and operations, in this case:

1) To manage the inventory and stock of each vending machine.
2) To evaluate the sales performance of each machine and each product.

Design

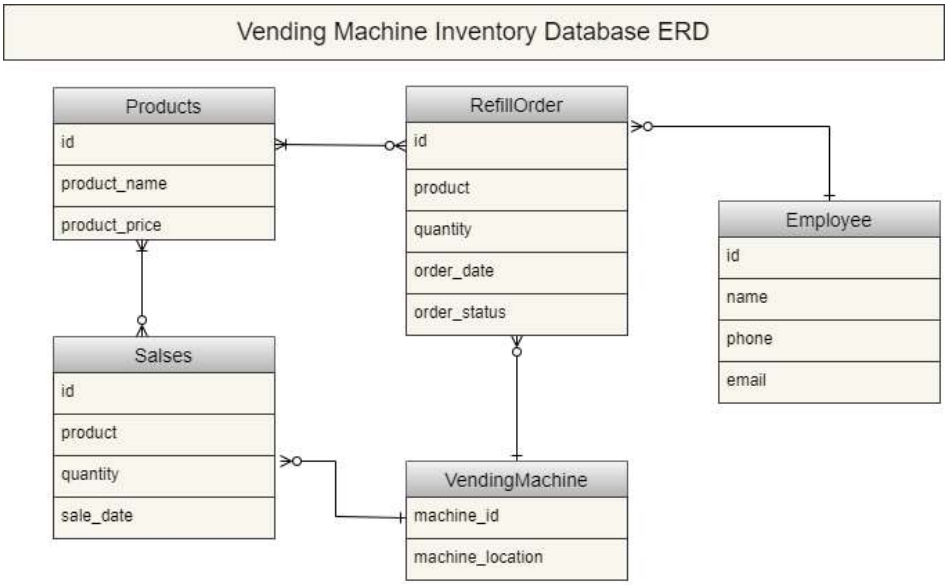
Based on the components of vending machine, this project has divided the whole system into 6 main categories:

1) Sales
2) Order
3) Stock
4) Machine
5) Product
6) Employee

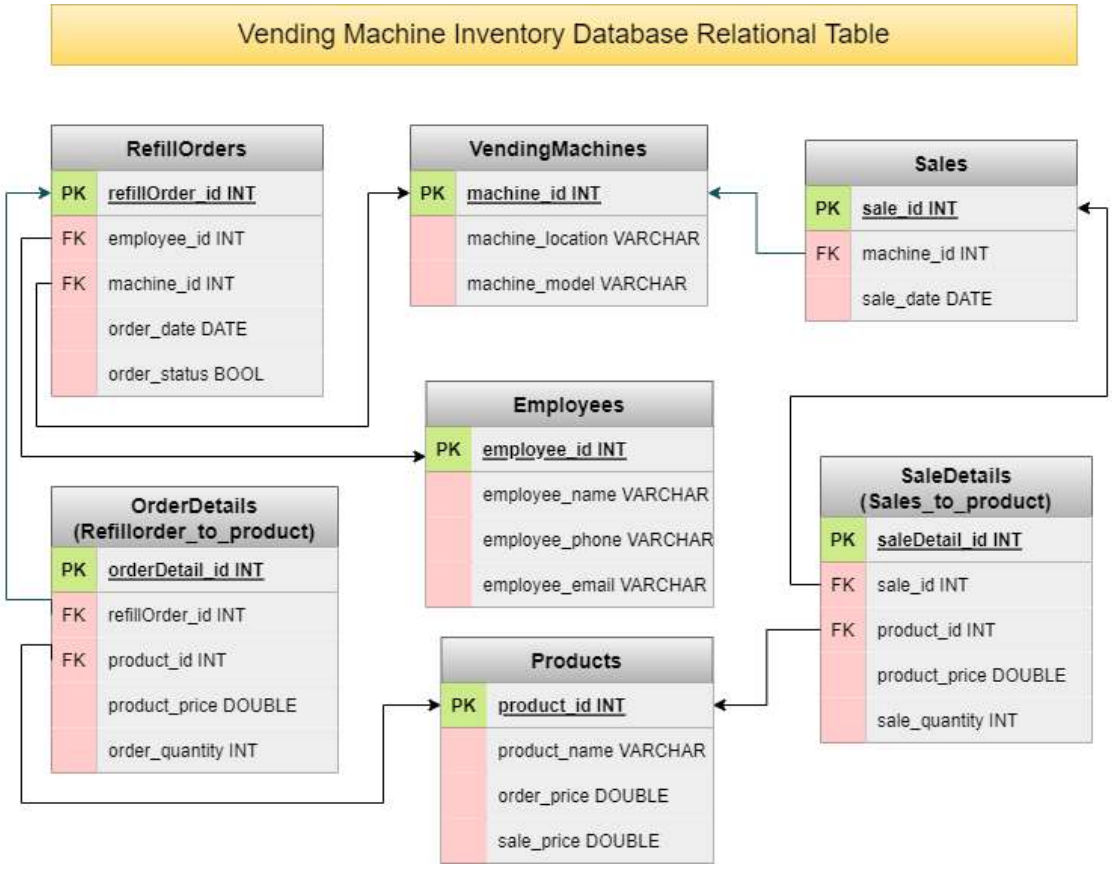
Under each category are main functions.

Category	**Functions**				
Sales	Add Sales	Show All Sales	SalesPerformence /Machine	SalesPerformance /Product	
Order	Add Order	Show All Orders	Update Order	Search Order	Delete Order
Stock	Show Stock	Auto Creating Order			
Machine	Add Machine	Show All Machines	Update Machine	Search Machine	Delete Machine
Product	Add Product	Show All Products	Update Product	Search Product	Delete Product
Employee	Add Employee	Show All Employees	Update Employee	Search Employee	Delete Employee

DB ER-diagram



Relational Table



Challenges

A. **Converting the *NULL* value to int 0.**

In database the value will be shown ass *NULL* where there is no data value. For example, in a sales table, the products' sale quantity value will be shown as *NULL* where the products have no sales:

product_id	product_name	sale_Qt
1	Julmust	20
2	Orange Juice	12
3	Protein Bars	13
4	Chocolate Bars	2
5	Mineral Water	2
6	Yogurt	7
7	Cok	1
8	Cookie	*NULL*
9	OREO	*NULL*
10	BEEF JERKY	*NULL*

It will affect later operation for example if sales money needs to be calculated (Total sales money = price * sale_Qt). The result will be also *NULL* where the sale_Qt is *NULL*.

using COALESCE to convert *NULL* value:

```
SELECT  p.product_id ,p.product_name,COALESCE(st.sale_Qt,0) AS sale_Qt
```

product_id	product_name	sale_Qt
1	Julmust	20
2	Orange Juice	12
3	Protein Bars	13
4	Chocolate Bars	2
5	Mineral Water	2
6	Yogurt	7
7	Cok	1
8	Cookie	0
9	OREO	0
10	BEEF JERKY	0

SQL Coalesce Function

The SQL server's Coalesce function is used to handle the Null values. The null values are replaced with user-defined values during the expression evaluation process. This function evaluates arguments in a particular order from the provided arguments list and always returns the first non-null value.

https://www.simplilearn.com/tutorials/sql-tutorial/coalesce-in-sql

B. Using WITH clause to get the stock.

In order to get the stock, first the total sales quantity need to be calculated by joining two tables.

```sql
SELECT saledetails.product_id, COALESCE(sum(saledetails.sale_quantity),0) AS Sqt
            FROM saledetails
            JOIN sales
            ON sales.sale_id = saledetails.sale_id
            WHERE sales.machine_id = 1
            GROUP BY saledetails.product_id;
```

Then using WITH Clause to store the Sqt(Total sale_ quantity), and the whole clause is given a new name as sold

```sql
WITH sold AS
(SELECT saledetails.product_id, COALESCE(sum(saledetails.sale_quantity),0) AS Sqt
            FROM saledetails
            JOIN sales
            ON sales.sale_id = saledetails.sale_id
            WHERE sales.machine_id = 1
            GROUP BY saledetails.product_id)
```

Then the total order quantity is calculated in another clause by joining tree tables. Here the Sqt is referred as sold.Sqt, because of the WITH clause used above.

Finally **LEFT OUTER JOIN** sold to get all the products including the ones that have not been sold.

Here **LEFT OUTER JOIN** is used because we want to see the stock of all the products that have been ordered on the inventory (in another word, all the products that have been refilled in the machines including the ones that have zero sales).

```sql

SELECT  orderdetails.product_id,products.product_name,sum(orderdetails.order_quantity)AS Oqt,COALESCE(sold.Sqt,0)AS
Sqt,(sum(orderdetails.order_quantity)-COALESCE(sold.Sqt,0)) AS Qt
        FROM orderdetails
        JOIN refillorders
        ON orderdetails.refillOrder_id = refillorders.refillOrder_id
        JOIN products ON orderdetails.product_id = products.product_id
        LEFT OUTER JOIN sold
        ON orderdetails.product_id  = sold.product_id
        WHERE refillorders.machine_id =1 AND refillorders.order_status=true = 1
        GROUP BY orderdetails.product_id;
```

Result:

product_id	product_name	Oqt	Sqt	Qt
1	Julmust	35	20	15
2	Orange Juice	32	12	20
3	Protein Bars	33	13	20
4	Chocolate Bars	22	2	20
5	Mineral Water	22	2	20
6	Yogurt	27	7	20
7	Cok	21	1	20
8	Cookie	20	0	20
9	OREO	20	0	20
10	BEEF JERKY	20	0	20
11	PopCorn	20	0	20
12	HealthyNut	20	0	20
13	Milk	21	1	20
16	Läkerol	26	6	20

Introduction to the SQL WITH Clause

The WITH clause in SQL was introduced in standard SQL to simplify complex long queries, especially those with `JOINs` and subqueries. Often interchangeably called CTE or subquery refactoring, a `WITH` clause defines a temporary data set whose output is available to be referenced in subsequent queries.

The best way to learn the `WITH` clause in SQL is through practice. I recommend LearnSQL.com's interactive Recursive Queries course. It contains over 100 exercises that teach the `WITH` clause starting with the basics and progressing to advanced topics like recursive `WITH` queries.

https://learnsql.com/blog/what-is-with-clause-sql/

Further Improvement

There will always be something can be improved in a project or work. It is the same in this project. There are several improvements can be done in the future, below are two of them:

- Logging in function

It would be more desirable if the system has login function which is true in all the cases. The employee would be required to login first in order to do any operations in the system.

```
public bool GetEmployeeLogInNameId(int idNr, string name)
{
    Open();
    string sql = $@"select * from employees where employee_name = '{name}' and employee_id=
    {idNr};";       You, 1 second ago • Uncommitted changes
    var result = connection.Query<Customer>(sql);
    if (result.Count() > 0)
        return true;
    else
        return false;
}
```

- Add date selection in SQL queries

Now orders and sales value are total values up to current time. It would be also more convenient if data values under certain period of time can be printed out. It will be solved simply by adding a date selection into SQL statement by using BETWEEN operator.

For example:

```
WITH st AS (SELECT sd.product_id,COALESCE(sum(sd.sale_quantity),0) AS Qt
        FROM saledetails sd
        INNER JOIN sales s ON s.sale_id = sd.sale_id
        WHERE s.machine_id = 1 AND s.sale_date BETWEEN '2020-01-01' AND '2023-01-01'
        GROUP BY sd.product_id )
```