Restaurant Database by Amit Deb

Made with Schema (MySQL v5.7)

> This database is used for restaurants. It keeps track of the amount of sales of different recipes, different menus, every recipe, all its inventory, chefs, and expenses.
> Using this information you can check the profit margins of different aspects of the restaurant.
> Such has if recipes lose or gain money on a sale and popularity of recipes on differ days of the week.

ERD

Relational Schema

Table: Menu

> This table shows different menus, what recipes are associated, and if the menus are active or not.

```
SELECT * FROM Menu;
```

MenuID	Menu_name	activeStatus	RecipeID
001	Sunday Brunch	0	000
001	Sunday Brunch	0	001
001	Sunday Brunch	0	002
001	Sunday Brunch	0	100
001	Sunday Brunch	0	101
001	Sunday Brunch	0	102
001	Sunday Brunch	0	202
002	Mid-Summer	1	003
002	Mid-Summer	1	004
002	Mid-Summer	1	006
002	Mid-Summer	1	103
002	Mid-Summer	1	104
002	Mid-Summer	1	105
002	Mid-Summer	1	200

```
create table Menu
(
MenuID          varchar(15),
Menu_name       varchar(45),
activeStatus    tinyint DEFAULT '0',
RecipeID    varchar(15),
foreign key (RecipeID) references Recipe (RecipeID) on delete cascade
);
```

Table: Recipe

> This table shows different recipes the restaurant has stored. It has its unique name, the time used to make it,
> what price it is, class whether it is Apps, Entree, Desert, or more, and what chef cooks it.

```
SELECT * FROM Recipe;
```

RecipeID	Recipe_Name	timeUsed	priceSold	class	ChefID
000	Roast Chicken	1.30	10.22	Entree	73791
001	Smoked Salmon	0.23	18.27	Entree	70183
002	Roast Rack of Lamb	0.45	24.49	Entree	73791
003	Lucy Burger	0.15	11.31	Entree	38192
004	Cider-Glazed Turkey	3.30	17.85	Entree	73791
005	Tennessee Pork Ribs	1.30	14.50	Entree	38192
006	Free Form Basil Lasagna	0.50	20.60	Entree	77777
100	Eggs and Toast	0.10	8.22	Apps	80122
101	Deviled Eggs	0.45	13.42	Apps	80122
102	Potato-Onion Soup	0.25	13.33	Apps	85541
103	Yellow Bean Salad	0.10	17.80	Apps	75119
104	Encrusted Tempura Shrimp	0.30	24.89	Apps	12381
105	Blue-Cheese Potato Salad	0.05	11.96	Apps	50213
106	Ham and Bean Stew	0.45	19.82	Apps	13571
200	Spiced Brandy Truffles	0.30	11.52	Dessert	91158
201	Double-Chocolate Raspberry Creams	0.15	25.64	Dessert	91158
202	Coconut Cashew Pralines	0.30	18.28	Dessert	91158

```
create table Recipe
(
RecipeID        varchar(15),
Recipe_Name     varchar(45) UNIQUE,
timeUsed   numeric(13,2),
priceSold  numeric(13,2),
class  varchar(25),
ChefID     varchar(15),
primary key (RecipeID),
  UNIQUE KEY `RecipeName_UNIQUE` (`Recipe_Name`),
  foreign key (ChefID) references Chef (ChefID) on delete cascade
  );
```

Table: Ingredients

> This table shows different ingredients by referencing the associated recipe, referecing the inventoryID for the ingredients,
> then the quantiy and units of the ingredients.

```
SELECT * FROM Ingredients;
```

InventoryID	RecipeID	quantity	unit
0001	000	1.00	lbs
0006	000	0.30	lbs
0007	000	0.12	lbs
0002	001	1.00	lbs
0005	002	0.30	lbs
0008	003	0.25	lbs
0009	003	0.01	lbs
0011	004	0.10	lbs
0012	004	0.30	fl. oz.
0019	005	0.41	lbs
0017	006	0.31	lbs
0016	006	0.23	lbs
0003	100	2.00	eggs
0004	100	0.05	loafs
0003	101	6.00	eggs
0006	102	0.30	lbs
0007	102	0.30	lbs
0010	103	1.04	lbs
0013	104	0.20	lbs
0014	104	2.21	lbs
0015	105	0.21	lbs
0006	105	0.32	lbs
0018	106	0.31	lbs

InventoryID	RecipeID	quantity	unit
0020	106	0.23	lbs
0028	200	1.09	lbs
0021	200	0.27	fl. oz.
0022	200	1.10	lbs
0023	201	0.34	lbs
0024	201	0.23	lbs
0025	201	0.11	fl. oz.
0028	201	0.91	lbs
0026	202	0.50	coconuts
0027	202	0.23	lbs
0028	202	0.31	lbs

```
create table Ingredients
(
    InventoryID     varchar(15),
    RecipeID    varchar(15),
    quantity     numeric(13,2),
    unit    varchar(20),
    foreign key (InventoryID) references Inventory (InventoryID) on delete cascade,
    foreign key (RecipeID) references Recipe (RecipeID) on delete cascade
);
```

Table: Chef

> This table shows the different chefs, thier ID, thier name, the payrate, and the type of chef.
> In this case we have French type chef so they title of thier job differ.

```
SELECT * FROM Chef;
```

ChefID	nameChef	typeChef	hourlyRate
12381	Oliver	Friturier	16.80
13571	Bourdain	Saucier	16.81
20000	Masaharu	Tournant	12.98
23589	Mario	Sous	22.97
29932	Marco	Tournant	12.76
36407	Ray	Entremetier	17.19
38192	Deen	Grillardin	16.06
50213	David	Garde Manger	15.28
59906	Raj	Head	31.41
70183	Guy	Poissonnier	16.05
73791	Alton	Rotisseur	16.25
75119	Giada	Legumier	26.83
77777	Curtis	Commis	14.58
78064	Bobby	Sous	23.59
80122	Lagasse	Boucher	16.33
83291	Cat	Tournant	13.97
83664	Gordon	Executive	35.67
85541	Child	Potager	16.36
91158	Wolfgang	Patissier	16.13

```
create table Chef
(
    ChefID      varchar(15),
    nameChef    varchar(45),
    typeChef    varchar(45),
    hourlyRate  numeric(13,2),
    primary key (ChefID)
);
```

Table: Inventory

> This table shows everything in inventory, its amount, units, and costPerUnit.

```
SELECT * FROM Inventory;
```

InventoryID	nameIngredient	amount	unit	costPerUnit
0001	Chicken Beast	19.00	lbs	7.38
0002	Salmon	17.00	lbs	5.49
0003	Egg	45.00	eggs	6.04
0004	Bread	30.00	loafs	8.16
0005	Lamb	25.00	lbs	7.92
0006	Potato	35.00	lbs	7.70
0007	Onion	23.00	lbs	4.05
0008	Ground Beef	13.00	lbs	8.35
0009	Cheddar	9.00	lbs	6.36
0010	Yellow Bean	40.00	lbs	2.51
0011	Turkey	21.00	lbs	2.15
0012	Cider	36.00	fl. oz.	7.86
0013	Tempura Batter	21.00	lbs	2.51
0014	Shrimp	41.00	lbs	4.38
0015	Blue Cheese	11.00	lbs	8.22
0016	Lasagna Pasta	21.00	lbs	6.52
0017	Mozzarella	26.00	lbs	6.52
0018	Pork Belly	14.00	lbs	5.76
0019	Pork Ribs	14.00	lbs	6.68
0020	Black Bean	41.00	lbs	6.90
0021	Brandy	23.00	fl. oz.	8.94
0022	Truffles	41.00	lbs	4.12
0023	Chocolate	21.00	lbs	6.67
0024	Raspberry	32.00	lbs	6.36

InventoryID	nameIngredient	amount	unit	costPerUnit
0025	Heavy Cream	13.00	fl. oz.	1.50
0026	Coconut	38.00	coconuts	8.17
0027	Cashew	41.00	lbs	4.13
0028	Sugar	39.00	lbs	2.26
0029	Raw Rats	200.00	rats	0.00

```
create table Inventory
(
    InventoryID     varchar(15),
    nameIngredient  varchar(50) UNIQUE,
    amount          numeric(13,2),
    unit            varchar(20),
    costPerUnit     numeric(13,2),
    primary key  (InventoryID),
    UNIQUE KEY `nameIngredient_UNIQUE` (`nameIngredient`)
);
```

Table: Sales

> This Table shows the sales of the restaurant with what recipe how many and when it was sold.

```
SELECT * FROM Sales;
```

SalesID	RecipeID	quantitySold	dateSold
100000	000	21.00	2021-01-01
100001	001	9.00	2021-01-02
100002	105	10.00	2021-02-12
100003	003	22.00	2021-02-20
100004	106	1.00	2021-03-13
100005	200	7.00	2021-03-30
100006	102	10.00	2021-04-01
100007	006	24.00	2021-06-07
100008	104	19.00	2021-09-11
100009	002	11.00	2021-10-21
100010	201	19.00	2021-11-13
100011	003	22.00	2021-12-09

```
create table Sales
(
    SalesID     varchar(15),
    RecipeID    varchar(15),
    quantitySold   numeric(13,2),
    dateSold    date,
    primary key  (SalesID),
    foreign key (RecipeID) references Recipe (RecipeID) on delete cascade
);
```

Table: Expense

> This Table shows the expenses of the restaurant with the name, the catagory, amount, and when it was paid.

```
SELECT * FROM Expenses;
```

ExpensesID	nameExpenses	catagory	amount	dateSpent
200000	Electric	Utilites	2000.99	2021-03-08
200001	Electric	Utilites	1092.79	2021-04-08
200002	Employees	Wages	700.99	2021-05-02
200003	Lawsuit	Law	90.99	2021-07-28
200004	Food Tax	Taxes	90.99	2021-12-25
200005	Food Tax	Taxes	90.99	2021-05-03
200006	Food Tax	Taxes	90.99	2021-10-03

```
create table Expenses
(
    ExpensesID      varchar(15),
    nameExpenses    varchar(50),
    catagory        varchar(50),
    amount          numeric(13,2),
    dateSpent       date,
    primary key (ExpensesID)
);
```

View: Recipe-less Chefs

> This View shows what chefs do not have recipe associated. We could cut down costs by firing these chefs.
> Although some of these chefs like Raj and Gordon are Chefs that mangage other chef so are important in a different way.

```
SELECT * FROM unassignedChefs AS `No Recipe Chefs`;
```

Names	Jobs	Wage
Masaharu	Tournant	12.98
Mario	Sous	22.97
Marco	Tournant	12.76
Ray	Entremetier	17.19
Raj	Head	31.41
Bobby	Sous	23.59
Cat	Tournant	13.97
Gordon	Executive	35.67

```
CREATE VIEW unassignedChefs AS
SELECT Chef.nameChef AS `Names`, Chef.typeChef AS `Jobs`, Chef.hourlyRate
AS `Wage`
FROM Chef
LEFT JOIN Recipe
ON Chef.ChefID = Recipe.ChefID
WHERE Recipe.ChefID IS NULL;
```

Function: Determine Profit/Loss of Entire Restaurant

> This Function shows the profit/loss of the restaurant given the date.
> It sums up all the sales and substracts it from the sum of expenses before the given date.

```
SELECT costOfRest(CURDATE()) AS `Restaurant Profit/Loss`;
```

Restaurant Profit/Loss
-1849.81

```
DELIMITER //
CREATE FUNCTION costOfRest
(
    dateToLookAt    date
)
RETURNS DECIMAL(13,2) SIGNED
BEGIN
    DECLARE gains DECIMAL(13,2) SIGNED;
    DECLARE losses DECIMAL(13,2) SIGNED;
    DECLARE total DECIMAL(13,2) SIGNED;

    SELECT SUM(s.quantitySold * Recipe.priceSold)
    INTO gains
    FROM Sales s
    JOIN Recipe
    ON Recipe.RecipeID = s.RecipeID
    WHERE s.dateSold <= dateToLookAt;

    SELECT SUM(amount)
    INTO losses
    FROM Expenses
    WHERE dateSpent <= dateToLookAt;

    SET total = gains - losses;

    RETURN total;

END //
DELIMITER ;
```

Procedure: Find if Taxes were Paid

> This Procedure checks the previous month of a given date and check taxes were paid. If not it tells you to pay taxes.

```
CALL taxEvasion(CURDATE(), @output);
SELECT @output AS `Taxes Done?`;
```

Taxes Done?
Taxes Paid month previous to: 2021-10-29

```
DELIMITER //
CREATE PROCEDURE taxEvasion
(
    IN dateToCheck DATE,
    OUT output VARCHAR(200)
)
BEGIN
    DECLARE taxesPaid BOOLEAN DEFAULT FALSE;

    SET taxesPaid = (SELECT EXISTS (
        SELECT *
        FROM Expenses
        WHERE catagory = 'Taxes'
        AND dateSpent BETWEEN DATE_ADD(dateToCheck, Interval -1 MONTH) AND
dateToCheck
    ));

    IF taxesPaid THEN
        SET output = CONCAT('Taxes Paid month previous to: ', dateToCheck);
    ELSEIF NOT taxesPaid THEN
        SET output = CONCAT('Taxes NOT Paid month previous to: ',
dateToCheck);
    END IF;
END //
DELIMITER ;
```

Query One: Recipes That Loss Money

> This Query calculates the profit of a recipe. It calculates the labor cost by multiplying the time used to make a recipe with the corresponding chef. Then the ingredient cost which multiplys the cost per unit of each ingredient. When adding thw labor and the ingredient cost we get a net value. Thus when we substract the price of which is it was sold at we get a profit or a loss. In this case we only want to see any negative value which are a loss. In business terms we want to either cut the serving, increase the price, change the chef, or use cheaper ingredients.

```
SELECT Distinct Recipe.Recipe_Name AS `Recipe`, SUM(Recipe.priceSold -
((Ingredients.quantity * Inventory.costPerUnit) + (Recipe.timeUsed *
Chef.hourlyRate))) AS `Money Loss`
FROM Recipe
JOIN Ingredients ON Ingredients.RecipeID = Recipe.RecipeID
JOIN Inventory ON Inventory.InventoryID = Ingredients.InventoryID
JOIN Chef ON Chef.ChefID = Recipe.ChefID
GROUP BY Recipe.Recipe_Name
HAVING SUM(Recipe.priceSold - ((Ingredients.quantity *
Inventory.costPerUnit) + (Recipe.timeUsed * Chef.hourlyRate))) < 0;
```

Recipe	Money Loss
Cider-Glazed Turkey	-74.1230
Deviled Eggs	-30.1685
Roast Chicken	-42.8910
Tennessee Pork Ribs	-9.1168

Query Two: Activate Sunday Brunch Menu

> This Query actives the specific Sunday Brunch Menu.

```
Update Menu
Set activeStatus = 1
Where Menu_name = 'Sunday Brunch' AND activeStatus = 0 AND MenuID = '001';
SELECT * FROM Menu;
```

MenuID	Menu_name	activeStatus	RecipeID
001	Sunday Brunch	1	000
001	Sunday Brunch	1	001
001	Sunday Brunch	1	002
001	Sunday Brunch	1	100
001	Sunday Brunch	1	101
001	Sunday Brunch	1	102
001	Sunday Brunch	1	202
002	Mid-Summer	1	003
002	Mid-Summer	1	004
002	Mid-Summer	1	006
002	Mid-Summer	1	103
002	Mid-Summer	1	104
002	Mid-Summer	1	105
002	Mid-Summer	1	200

Query Three: Grabs All Recipes and The Corresponding Menu

> This Query takes every recipe and attaches its menu to it. So it will show any recipe not in menu.

```
(SELECT Recipe.Recipe_Name AS `Recipe`, Menu.Menu_name AS `Menu`
From Recipe
RIGHT JOIN Menu
ON Menu.RecipeID = Recipe.RecipeID)
UNION
(SELECT Recipe.Recipe_Name, Menu.Menu_name
From Recipe
LEFT JOIN Menu
ON Menu.RecipeID = Recipe.RecipeID);
```

Recipe	Menu
Roast Chicken	Sunday Brunch
Smoked Salmon	Sunday Brunch
Roast Rack of Lamb	Sunday Brunch
Eggs and Toast	Sunday Brunch
Deviled Eggs	Sunday Brunch
Potato-Onion Soup	Sunday Brunch
Coconut Cashew Pralines	Sunday Brunch
Lucy Burger	Mid-Summer
Cider-Glazed Turkey	Mid-Summer
Free Form Basil Lasagna	Mid-Summer
Yellow Bean Salad	Mid-Summer
Encrusted Tempura Shrimp	Mid-Summer
Blue-Cheese Potato Salad	Mid-Summer
Spiced Brandy Truffles	Mid-Summer
Double-Chocolate Raspberry Creams	NULL
Ham and Bean Stew	NULL
Tennessee Pork Ribs	NULL

Query Four: Any Inventory Not Used in Recipes

> This Query will show any inventory item that is not being using in any recipe. For example rats are not used in any recipe.

```
SELECT  Inventory.nameIngredient,  Inventory.amount, Inventory.unit
FROM Inventory
WHERE Inventory.InventoryID NOT IN
(
    SELECT Ingredients.InventoryID
    FROM Ingredients
);
```

nameIngredient	amount	unit
Raw Rats	200.00	rats

Query Five: Any Recipes Not Used In Menu

> This Query specifically shows the any recipe not being used in Menu at the moment.

```
SELECT Recipe.Recipe_name, Recipe.RecipeID
FROM Recipe
WHERE NOT EXISTS
(
    SELECT Menu.RecipeID
    FROM Menu
    WHERE Menu.RecipeID = Recipe.RecipeID
);
```

Recipe_name	RecipeID
Double-Chocolate Raspberry Creams	201
Ham and Bean Stew	106
Tennessee Pork Ribs	005

View on DB Fiddle