Virtual Spatula

Purpose of the Database

- The system aims to provide an end-to-end cooking experience of different cuisines for app users including recipe selection, grocery shopping and delivery

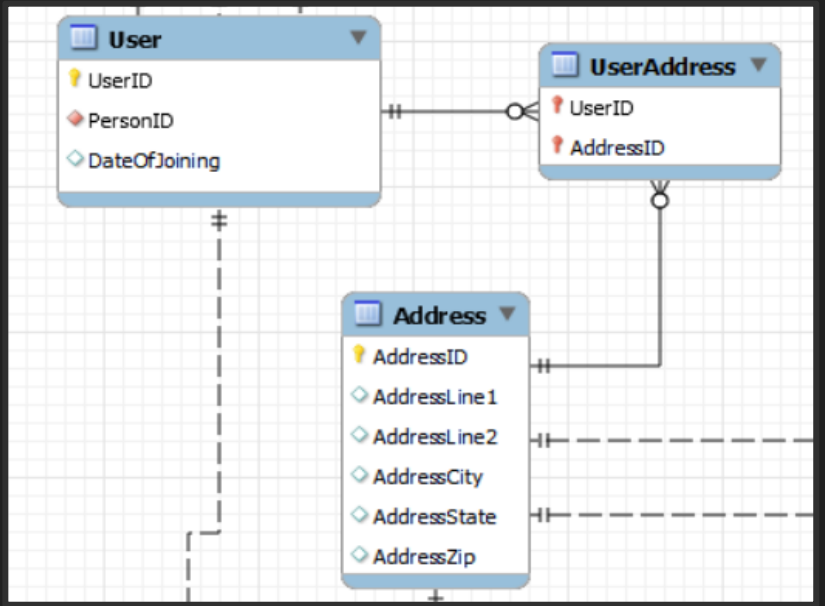
- By picking a recipe, purchasing the necessary ingredients, and opting for doorstep delivery, users may effortlessly cook at home

Mission statement & Business problem

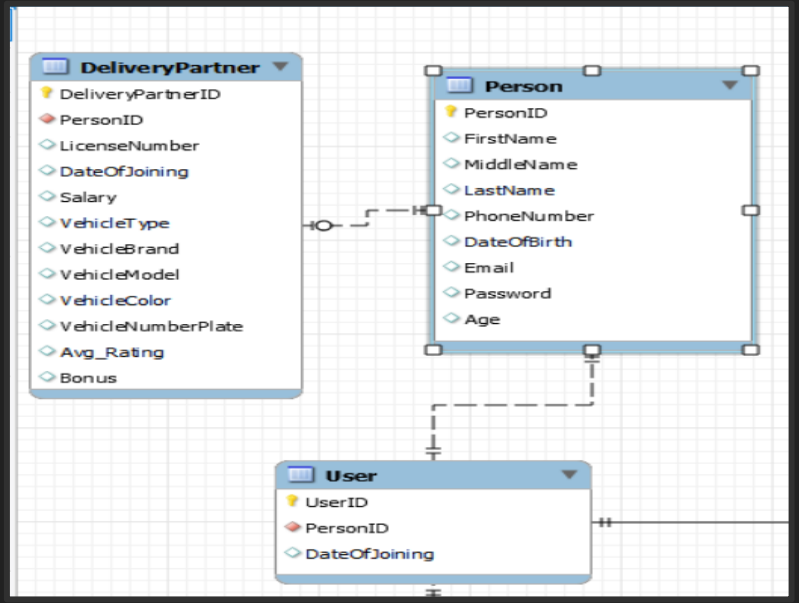
- Provide a list of recipes with ingredients required for the Virtual Spatula app
- Maintain a list of product availability for various shop inventories
- Increase delivery efficiency by designating the appropriate delivery partner based on location

Business Rules

1. One user can have multiple addresses. One address can be shared by multiple users

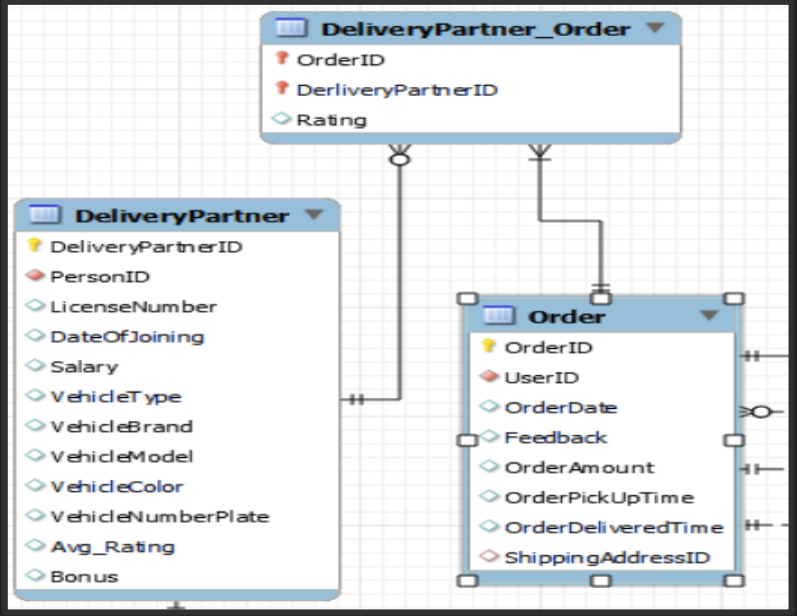


2. A person can be a user, a delivery partner or both

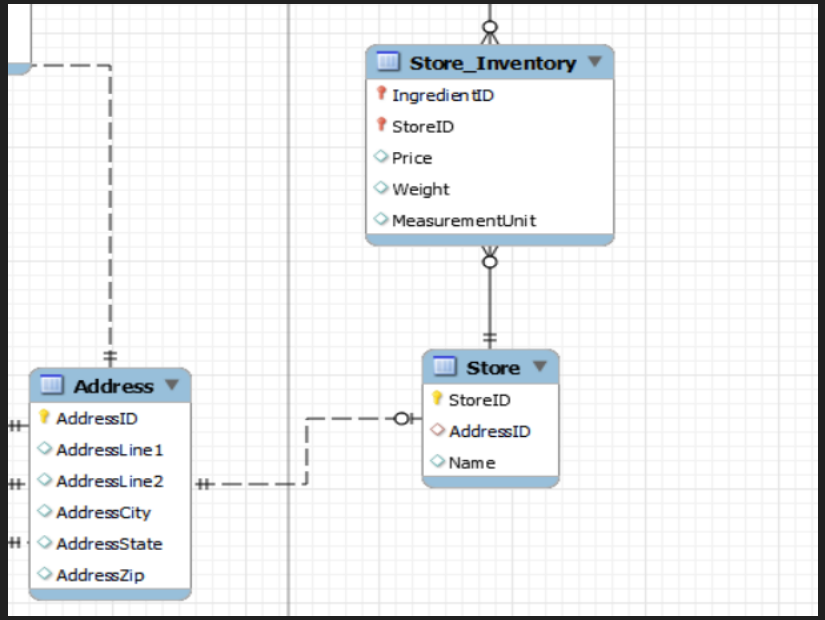


Business Rules

3. If a delivery partner faces issues during delivery, another delivery partner should be assigned to the order.



4. A non-operational store can be present in the database for future collaborations.



Computed Column - Bonus of Delivery Partner

Bonus of Delivery Partner is calculated as:

- 10% of Salary * Average Rating (Range 1-5)
- 5% of Salary if Monthly Goal is achieved

	DeliveryPartnerID	Salary	Avg_Rating	Bonus
1	1	1,000	4.5	450
2	2	1,200	5	600
3	3	11,500	4	4,600
4	4	1,200	5	600
5	5	1,100	3	330
6	6	1,000	2.5	250
7	7	1,100	5	550
8	8	1,000	5	500

```sql
---- Function for Bonus calculation
CREATE FUNCTION Payment.Bonus(@PartnerID INT , @GOAL INT , @Month INT)
RETURNS DECIMAL(20,2)
AS
BEGIN
    DECLARE @Avg_Rating DECIMAL(8,2) , @No_of_Orders INT , @Salary INT;
    DECLARE @Bonus Decimal(20,2);
    SELECT @Salary = Salary
    FROM Person.DeliveryPartner
    WHERE DeliveryPartnerID = @PartnerID;

    SELECT @Avg_Rating = AVG(CAST(Rating AS DECIMAL(8,2)))
    FROM [Order].DeliveryPartner_Order
    WHERE DeliveryPartnerID = @PartnerID;

    SELECT  @No_Of_Orders = COUNT(dpo.OrderID)
    FROM [Order].DeliveryPartner_Order dpo
    LEFT JOIN [Order].[Order] ord
        ON dpo.OrderID = ord.OrderID
    WHERE DATEPART(MONTH,ord.OrderDate) = @Month
    AND DATEPART(YEAR,GETDATE()) = 2021
    AND dpo.DeliveryPartnerID = @PartnerID ;

    DECLARE @OrderBonus DECIMAL(20,2);
    SET @OrderBonus = 0;
    IF(@No_Of_Orders > @Goal)
    BEGIN
        SET @OrderBonus = 0.05*@Salary;
    END

    SET @Bonus = @Avg_Rating*0.1*@Salary + @OrderBonus;
    RETURN @Bonus;
END
```

*Age Computed Column

Computed Column- Average Rating

- Average Rating is calculated as:

  --Each delivery partner is given a rating by the customer for each order.

  --We used this ratings to calculate the average rating.

```
CREATE FUNCTION Payment.Avg_Rating(@PartnerID INT )
RETURNS DECIMAL(20,2)
AS
BEGIN
    DECLARE @Avg_Rating DECIMAL(8,2)
    SELECT @Avg_Rating = AVG(CAST(Rating AS DECIMAL(8,2)))
    FROM [Order].DeliveryPartner_Order
    WHERE DeliveryPartnerID = @PartnerID;
    RETURN @Avg_Rating;
END;
```

OrderID	DeliveryPartnerID	Rating
1	1	4
2	1	5
3	2	5
4	3	4
5	3	4
6	4	5
7	5	3
8	6	1
9	6	4
10	7	5
11	8	5

DeliveryPartnerID	Avg_Rating
1	4.5
2	5
3	4
4	5
5	3
6	2.5
7	5
8	5

Encrypted Column - Password And Card Number

- Created encrypted columns to store password of the users to maintain personal information security
- To securely store payment card numbers, they are encrypted using symmetric key and AES 128 encryption algorithm

```
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Virtualpass';

CREATE CERTIFICATE VirtualCertificate
WITH SUBJECT = 'Virtual Spatula Certificate',
EXPIRY_DATE = '2025-11-30';

CREATE SYMMETRIC KEY PassCardKey
WITH ALGORITHM = AES_128
ENCRYPTION BY CERTIFICATE VirtualCertificate;

OPEN SYMMETRIC KEY PassCardKey
DECRYPTION BY CERTIFICATE VirtualCertificate;
```

```
-- Insert encrypted data
INSERT INTO Person.Person
    VALUES('John', 'Smith', 'Doe',
           '5362179108','1990-11-11',
           'john@gmail.com',
           EncryptByKey(Key_GUID(N'PassCardKey'), 'Password'));
```

```
INSERT INTO VirtualSpatula.Payment.Payment (PaymentMethod,NameOnCard,CardNumber,ExpireDate,
                                            PaymentDate,UserID,AddressID)
      VALUES (N'VISA',N'Wayra Ashkii',EncryptByKey(Key_GUID(N'PassCardKey'), '4539755523668190'),
              N'2022-05-31',{ts '2021-01-29 12:43:15.000'},1,1);
```

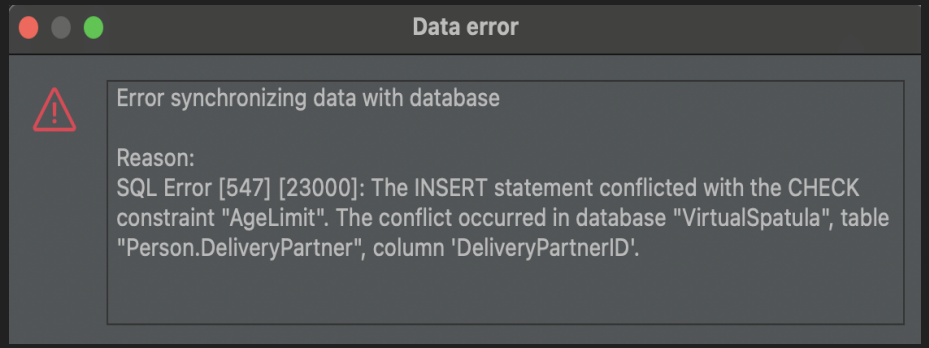
Table Level Check Constraint - Delivery Partner Age

- Table level check constraint on DeliveryPartner table to check if a Person is 18 years old

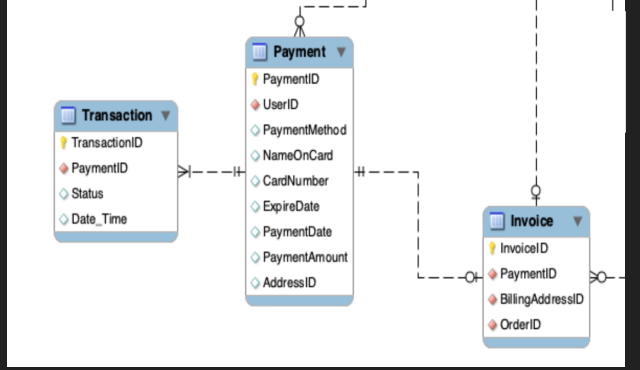
```
-- Table level check constraint

ALTER TABLE Person.DeliveryPartner
    ADD CONSTRAINT AgeLimit CHECK
        (Person.DeliveryPartnerAge(DeliveryPartnerID) >= 18);
```

```
CREATE FUNCTION Person.DeliveryPartnerAge(@PartnerID INT)
RETURNS INT
AS
BEGIN
    DECLARE @Age INT;
    SELECT @Age = Person.Calculate_Age(p.DateOfBirth)
    FROM Person.DeliveryPartner d
    JOIN Person.Person p
    ON d.PersonID = p.PersonID
    WHERE d.DeliveryPartnerID = @PartnerID
RETURN @Age;
END
```



Payment Method Promotion - View



- Promotions can be added to Payment methods that are used less frequently
- Provide cashback if a Payment method is used in less than or equal to 20% transactions
- Can be used as a report to negotiate promotions with credit card companies such as VISA, American Express, and etc.

```
CREATE VIEW CardPromtion AS
SELECT PaymentMethod,Times ,
CASE WHEN CAST(Times AS decimal(5,2))/(SELECT count(PaymentID) FROM Payment.Payment) <=0.2
     THEN '10% Cashback' ELSE 'No Offer' END AS 'Offer'
FROM (
SELECT p.PaymentMethod ,count( p.PaymentID)  AS Times
FROM Payment.Payment p
GROUP by p.PaymentMethod) AS a;
```

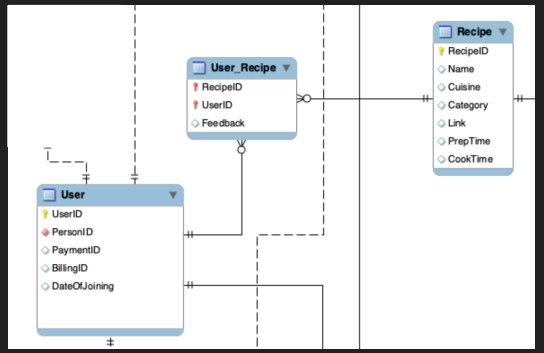
PaymentMethod	Times	Offer
American Express	2	10% Cashback
Discover	2	10% Cashback
VISA	6	No Offer

User Profile - View

- Displays the most preferred cuisine and total number of recipes per user.
- Can be used to provide promotions and recommend new recipes.

```
CREATE VIEW UserProfile AS
WITH temp AS
(SELECT p.UserID AS UserID1, rs.Cuisine AS CuisineName, COUNT(rs.Cuisine) AS Cuisine,
DENSE_RANK() OVER (PARTITION BY p.UserID ORDER BY COUNT(rs.Cuisine) DESC) AS [Rank of Cuisine]
FROM Person.[User] p
JOIN Recipe.User_Recipe r
ON p.UserID = r.UserID
JOIN Recipe.Recipe rs
ON r.RecipeID = rs.RecipeID
GROUP BY p.UserID,rs.Cuisine),
t2 AS(
SELECT UserID1, SUM(Cuisine) RecipeCount
FROM temp
GROUP BY UserID1
)
SELECT temp.UserID1,
       STRING_AGG(temp.CuisineName,', ') MostPreferredCuisine,
       MAX(t2.RecipeCount) TotalRecipes
FROM temp
JOIN t2
ON temp.UserID1 = t2.UserID1
WHERE [Rank of Cuisine]=1
GROUP BY temp.UserID1
```

	UserID1	FirstName	MostPreferredCuisine	TotalRecipes
1	1	John	Indian, Chinese	7
2	2	Shivani	Dessert	6
3	3	Sanchana	Indian	5
4	4	Dongliang	Dessert, Indian	2
5	5	Pulkit	Indian	3
6	6	Yuqi	Italian, Indian	2
7	7	Chris	Indian	2
8	8	Bennett	Indian	1

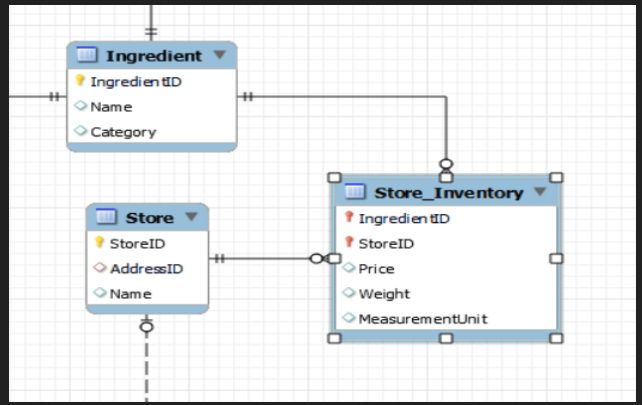


Store Inventory -View

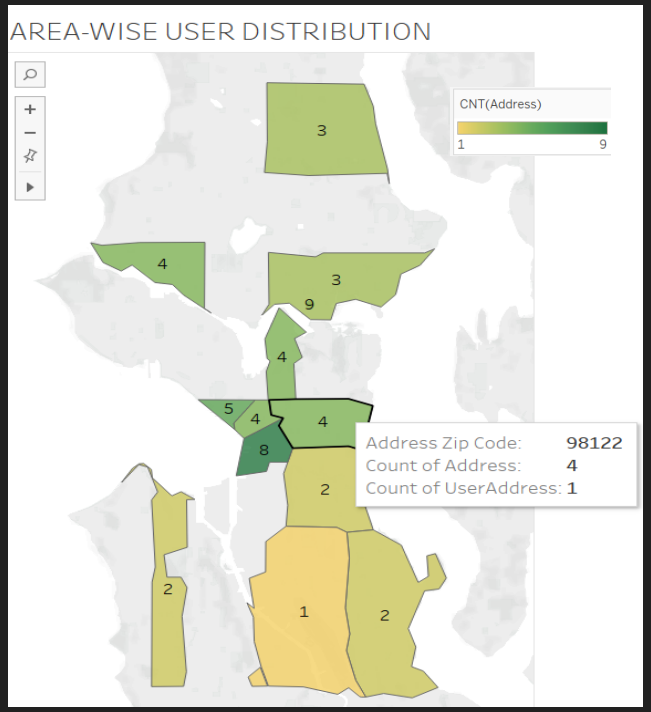
- Displays the availability of ingredients in stores

```
CREATE VIEW StoreView AS
WITH temp AS
(
SELECT DISTINCT t.IngredientID , t.StoreID,
CASE WHEN CONCAT(t.StoreID ,' ',t.IngredientID)  IN
    (SELECT concat(ssi.StoreID ,' ' ,ssi.IngredientID) FROM Store.StoreInventory ssi)
THEN 'Yes' ELSE 'No' END AS 'Availability'
FROM
    (
    SELECT ri.IngredientID ,ss.StoreID
    FROM Recipe.Ingredient ri
    CROSS JOIN Store.Store ss
    ) t
    CROSS JOIN Store.Store ss
    LEFT JOIN Store.StoreInventory ssi
        ON ss.StoreID = ssi.IngredientID
)
SELECT t.IngredientID,ri.Name,t.Store1, t.Store2, t.Store3
FROM (
SELECT IngredientID, [1] 'Store1', [2] 'Store2', [3]  'Store3'
FROM
    (SELECT IngredientID , StoreID, [Availability]
    FROM temp) vertical
PIVOT
(MAX([Availability])
FOR StoreID IN ([1],[2],[3])) horizontal
)  AS t
LEFT JOIN Recipe.Ingredient ri
    ON t.IngredientID = ri.IngredientID;
```

IngredientID	Name	Store1	Store2	Store3
10	Egg	No	No	No
11	Chicken	Yes	Yes	No
12	Mutton	Yes	No	No
13	Flour	Yes	No	Yes
14	Sugar	No	Yes	No
15	Rice	No	Yes	No
16	Mushroom	No	Yes	No
17	Bell Pepper	No	Yes	No
18	Cheese	Yes	No	Yes
20	Yeast	No	No	No
21	Onion	No	No	No



Distribution of Users in Seattle Area

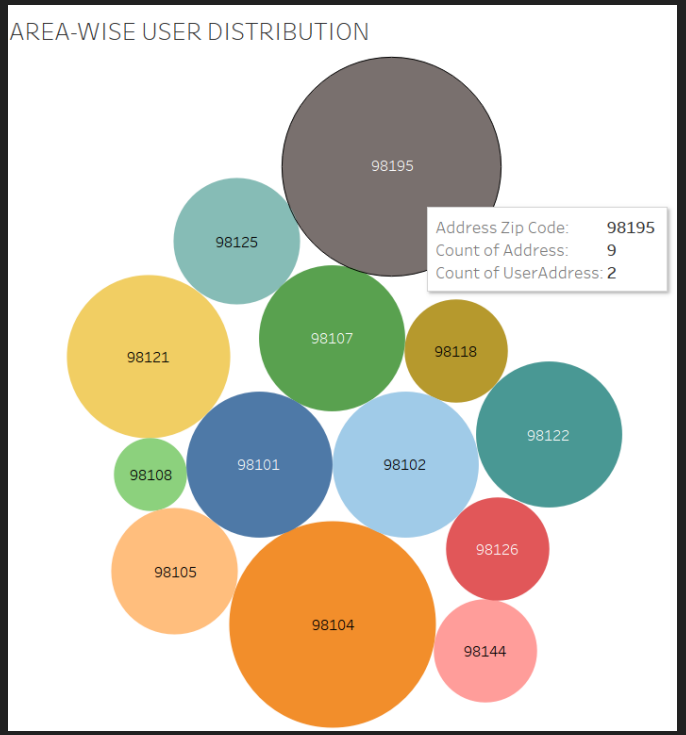


Above Geographical map reveals the concentration of users in different areas of Seattle

Uses of this report:

- Assignment of delivery partners to different areas based on the user concentration
- Collaboration with new stores in the user concentrated areas

Below Bubble chart shows the relational values. Size of bubble represents the count of users in each area



Recipe Word Cloud

The recipe word cloud visualisation helps us to see which recipe has been ordered the most.

This analysis/report can be useful in determining what type of new dish should be added to the database based on the most often ordered recipes and their relevant cuisines.

THANK YOU