
Group 4

Alejandro Suntay

James Talbott

Nirmalya Ghosh

Phuc To

Milestone 2

16th May 2021

OVERVIEW

Ready2Plate is an app designed to streamline the restaurant business. The app does this by analyzing every possible data point in the day to day operation. This data is created through optimization of the ordering process, and starts with the customer. The team chose to design this app in order to make our favorite restaurants more profitable AND more popular!

GOALS

1. Reduce latency and increase accuracy per order
2. Help restaurants establish better financial practices and optimize flow and purchases.
3. Make restaurants more profitable and popular!

SPECIFICATIONS

Ready2Plate contains three parts. The first part is a database that allows the restaurant owner to track all aspects of daily operation. The second part is an ordering system that allows staff to not have to take orders, which reduces the number of total staff members needed and reduces incorrect orders as the order is solely handled by the customer and the app. The third and final part of the app is allowing all parties involved in preparing the food or drink to check stock, submit orders to the Kitchen or Bar manager if stock is low, and see what ingredients are required for what dish the moment the customer submits an order.

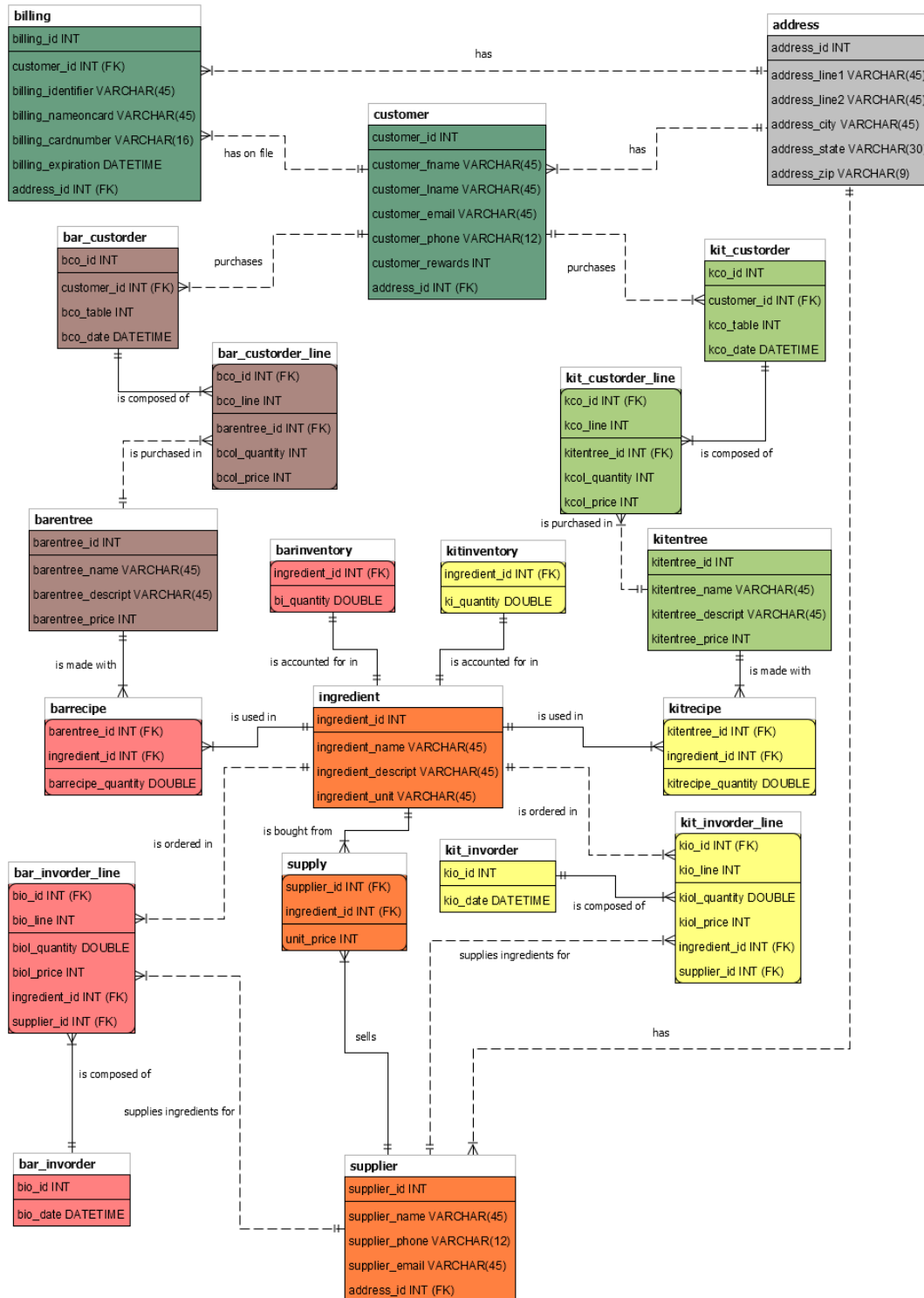
Once the customer has entered their information in, they view the menu and place an order. All aspects of the cooking process are tracked from Supplier to Ingredient, through the kitchen, and then when the plate is delivered to the customers table. Ready2Plate also does this for the bar, with both Kitchen and Bar receiving separate inventories so separate orders can be made by each Manager. This allows the kitchen staff and bar staff in maximum scaled applications to work independently of each other for optimal efficiency.

In the Kitchen, Ready2Plate allows the kitchen staff to see the order as soon as it is placed as well as all the ingredients that are required to prepare every dish. After the dish is prepared it goes to pickup, and is brought directly to the table with no guesswork (as the customers location is marked when they make their order). This will also cut down incorrect orders and food waste while increasing turnover rates.

Finally, Ready2Plate will decrease the amount of overhead restaurants need per month. The app accomplishes this by showing exactly which dishes were ordered, and which ingredients were used so the restaurant can waste less money on extra ingredients, and can create data models every week to track which entrees are popular so in turn they can order the appropriate items from the supplier.

MILESTONES

Internal Model



[High Resolution Image](#)

Queries

The following are the possible relevant queries that can be used for our app:

1. Customer Information

- This would be a basic query for pulling customer info.

SELECT

```
customer_fname,  
customer_lname,  
customer_email,  
customer_phone,  
address_id,  
customer_rewards
```

FROM

```
customer;
```

```
1 • SELECT customer_fname, customer_lname, customer_email, customer_phone, address_id, customer_rewards  
2 FROM customer;
```





	customer_fname	customer_lname	customer_email	customer_phone	address_id	customer_rewards
▶	Ashil	Clabburn	adlabburn0@goo.ne.jp	750 865 1873	1	5
	Valentijn	Copsey	vcopsey1@theforest.net	232 496 5316	2	97
	Bertrand	Boylan	bboylan2@tripadvisor.com	646 737 2964	3	70
	Ceil	Cecchi	cccecchi3@earthlink.net	380 899 4229	4	48
	Cal	Rechert	crechert4@tinyurl.com	312 268 1511	5	79
	Jaquith	Rosser	jrosser5@utexas.edu	827 389 3657	6	99
	Goldi	Camies	gcames6@delicious.com	728 587 6534	7	83
	Felix	Greatland	fgreatland7@mayoclinic.com	487 677 3087	8	39
	Chrisy	Lush	clush8@zdnnet.com	452 158 6351	9	26

2. Kitchen Inventory Ingredient Status

- This would allow us to check the ingredient inventory status -- If it's low.
- The limit is a dynamic parameter.

```
SET
    @low_limit = 100;
SELECT
    i.ingredient_id,
    i.ingredient_name,
    k.ki_quantity,
    ingredient_unit
FROM
    ingredient i
JOIN
    kitinventory k USING (ingredient_id)
WHERE
    i.ingredient_unit = "pounds"
    AND k.ki_quantity < @low_limit;
```

```
1 • SET @low_limit = 100;
2
3 • SELECT I.ingredient_id, I.ingredient_name, K.ki_quantity, ingredient_unit
4 FROM ingredient I
5 JOIN kitinventory K using (ingredient_id)
6 WHERE I.ingredient_unit = "pounds" and K.ki_quantity < @low_limit;
```

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 				
	ingredient_id	ingredient_name	ki_quantity	ingredient_unit
▶	6	garlic	96.84	pounds
	8	salt	52.9	pounds
	15	cherry tomatoes	94.42	pounds
	25	smoked sausage	57.47	pounds
	34	fresh shiitake mushrooms	80.95	pounds
	48	chicken drumsticks	21.66	pounds
	55	orange juice	32.15	pounds
	58	dried oregano	72.14	pounds

```

SET
    @low_limit = 100;
SELECT
    i.ingredient_id,
    i.ingredient_name,
    b.bi_quantity
FROM
    ingredient i
JOIN
    barinventory b USING (ingredient_id)
WHERE
    i.ingredient_unit = "ounces"
    AND b.bi_quantity < @low_limit;

```

```

1 • SET @low_limit = 100;
2 • SELECT I.ingredient_id, I.ingredient_name, B.bi_quantity
3     FROM ingredient I
4     JOIN barinventory B using (ingredient_id)
5     WHERE I.ingredient_unit = "ounces" and B.bi_quantity < @low_limit;

```

	ingredient_id	ingredient_name	bi_quantity
▶	202	vegetable oil	49.79
	204	banana liqueur	3.54
	207	mixed spice	66.71
	211	mixed spice	62.08
	214	sultana	64.72
	218	andouille sausage	73.99
	252	white vinegar	80.24
	257	water	33.84
	262	vegetable oil	13.68
	268	penne	30.14
	283	grated jack cheese	1.69
	289	hot chili powder	6.14
	291	egg whites	35.19
	300	chicken broth	5

3. Customer Reward Points

- The reward points required would be set by the owner (Restaurant Owner or Manager)
- The customer's first and last name would be input using the app.
- Will print out the points and whether the customer has enough points.

```
SET @point_required = 40;
SET @fname = "gay";
SET @lname = "touroto";
SELECT
    customer_id,
    customer_fname,
    customer_lname,
    customer_rewards,
    (customer_rewards >= @point_required) AS eligibility
FROM
    customer
WHERE
    customer_fname = @fname AND customer_lname = @lname;
```

```
1 • SET @point_required = 40;
2 • SET @fname = "Gay";
3   SET @lname = "Touroto";
4
5 • SELECT customer_id, customer_fname, customer_lname,
6         customer_rewards, (customer_rewards >= @point_required) as eligibility
7   FROM customer
8  WHERE customer_fname = @fname and customer_lname = @lname;
```

	customer_id	customer_fname	customer_lname	customer_rewards	eligibility
►	696	Gay	Touroto	100	1

4. Best Sellers

- Prints out the five 5 most ordered entrees from the kitchen.
- Can be done similarly with Bar entrees.

```
SELECT
    k.kitentree_id,
    k.kitentree_name,
    k.kitentree_descript,
    COUNT(kl.kitentree_id)
FROM
    kitentree k
JOIN
    kit_custorder_line kl USING (kitentree_id)
GROUP BY
    kl.kitentree_id
ORDER BY
    COUNT(kl.kitentree_id) DESC LIMIT 5;
```

```
1 • SELECT K.kitentree_id, K.kitentree_name, K.kitentree_descript, count(KL.kitentree_id) as 'amount of orders'
2 FROM kitentree K
3 JOIN kit_custorder_line KL using (kitentree_id)
4 GROUP BY KL.kitentree_id
5 ORDER BY count(KL.kitentree_id) DESC
6 LIMIT 5;
```

	kitentree_id	kitentree_name	kitentree_descript	amount of orders
▶	2	aliquam	Duis bibendum, felis sed interdum venenatis,	18
	49	aliquet massa id	Morbi non lectus. Aliquam sit amet diam in ma	18
	42	massa donec dapibus	Fusce posuere felis sed lacus. Morbi sem maur	16
	26	diam vitae quam	Maecenas tristique, est et tempus semper, est	16
	19	curae mauris	Praesent id massa id nisl venenatis lacinia.	15

5. Top 10 Customer Rank

- Rank customers based on how much money they have spent at the restaurant (kitchen and bar) for promotions to be sent out to their emails or app.

```
SELECT
  c.customer_id,
  c.customer_fname,
  c.customer_lname,
  c.customer_email,
  SUM(kcl.kcol_quantity*kcl.kcol_price) AS kitchenmoneyspent
FROM
  customer c
  JOIN
    kit_custorder kc
    ON c.customer_id = kc.customer_id
  JOIN
    kit_custorder_line kcl
    ON kc.kco_id = kcl.kco_id
GROUP BY
  c.customer_id
ORDER BY
  kitchenmoneyspent DESC LIMIT 10;
```

```
1 • SELECT c.customer_id, c.customer_fname, c.customer_lname, c.customer_email,
2         SUM(kcl.kcol_quantity*kcl.kcol_price) AS KitchenMoneySpent
3 FROM customer c
4 JOIN kit_custorder kc on c.customer_id = kc.customer_id
5 JOIN kit_custorder_line kcl on kc.kco_id = kcl.kco_id
6 GROUP BY c.customer_id
7 ORDER BY KitchenMoneySpent DESC LIMIT 10;
```

	customer_id	customer_fname	customer_lname	customer_email	KitchenMoneySpent
►	412	Lavina	Schubart	lschubartbf@wired.com	620
	927	Coop	Burness	cburnesspq@google.com.br	597
	620	Paddie	Eastcourt	peastcourth7@bandcamp.com	561
	354	Samuel	Jeger	sjeger9t@vk.com	509
	98	Josepha	Carding	jcarding2p@home.pl	500
	599	Pepillo	Sandell	psandellgm@vimeo.com	494
	101	Leena	Eustace	leustace2s@reference.com	480
	687	Maximo	Eliez	meliezj2@cafepress.com	471
	577	Misti	Obal	mobalg0@hc360.com	466
	826	Verile	Hearons	vhearonsmx@mapy.cz	446

```

SELECT
    c.customer_id,
    c.customer_fname,
    c.customer_lname,
    c.customer_email,
    SUM(bcl.bcol_quantity*bcl.bcol_price) AS barmoneyspent
FROM
    customer c
    JOIN
        bar_custorder bc
        ON c.customer_id = bc.customer_id
    JOIN
        bar_custorder_line bcl
        ON bc.bco_id = bcl.bco_id
GROUP BY
    c.customer_id
ORDER BY
    barmoneyspent DESC LIMIT 10;

```

```

1 • SELECT c.customer_id, c.customer_fname, c.customer_lname, c.customer_email,
2         SUM(bcl.bcol_quantity*bcl.bcol_price) AS BarMoneySpent
3 FROM customer c
4 JOIN bar_custorder bc on c.customer_id = bc.customer_id
5 JOIN bar_custorder_line bcl on bc.bco_id = bcl.bco_id
6 GROUP BY c.customer_id
7 ORDER BY BarMoneySpent DESC LIMIT 10;

```

	customer_id	customer_fname	customer_lname	customer_email	BarMoneySpent
▶	644	Sue	Lincey	slinceyhv@freewebs.com	1426
	434	Vivyan	Bautiste	vbautistec1@skyrock.com	1415
	175	Aurore	Stannis	astannis4u@princeton.edu	1387
	463	Glynn	Barthrup	gbarthrupcu@latimes.com	1321
	711	Orlando	Winchcombe	owinchcombejq@booking.com	1212
	515	Chrysa	Downse	cdownseea@networksolutions.com	1193
	186	Killy	Mussalli	kmussalli55@china.com.cn	1174
	916	Camellia	Willgoose	cwillgoosepf@godaddy.com	1147
	582	Fabe	Bohl	fbohlg5@whitehouse.gov	1104
	627	Angil	Chipchase	achipchasehe@networkadvertising.org	1047

6. Best Seat In The House

- Table use tracking - sorts most-used tables in a given date range:
- Good for preventative maintenance and pattern analysis

```
SELECT
    bco_table AS bar_table,
    COUNT(bco_table) AS times_used
FROM
    bar_custorder
WHERE
    bco_date BETWEEN '2021-06-01' AND '2021-06-07'
GROUP BY
    bco_table
ORDER BY
    times_used DESC;

SELECT
    kco_table AS kitchen_table,
    COUNT(kco_table) AS times_used
FROM
    kit_custorder
WHERE
    kco_date BETWEEN '2021-06-01' AND '2021-06-07'
GROUP BY
    kco_table
ORDER BY
    times_used DESC;
```

- 1 • Select bco_table as bar_table, count(bco_table) As times_used
- 2 From bar_custorder
- 3 Where bco_date between '2021-06-01' and '2021-06-07'
- 4 Group By bco_table
- 5 Order By times_used desc;
- 6 • Select kco_table as kitchen_table, count(kco_table) As times_used
- 7 From kit_custorder
- 8 Where kco_date between '2021-06-01' and '2021-06-07'
- 9 Group By kco_table
- 10 Order By times_used desc;

	bar_table	times_used
▶	18	4
	28	3
	30	3
	11	3
	24	3
	29	2
	16	2
	14	2
	9	2

Result 81 × Result 82

	kitchen_table	times_used
▶	25	4
	12	4
	14	4
	27	3
	19	3
	2	3
	30	3
	26	3
	21	2

Result 81 Result 82 ×

7. Loyal Customers' Favorites

- Calculates the entrees that appear on the most orders made by customers participating in the rewards points system.
- This is relevant to the Managerial staff and Owner to allow them to optimize to their most loyal customers.

```
SELECT
  c.customer_id,
  c.customer_fname,
  c.customer_lname,
  c.customer_rewards,
  barentree_name,
  COUNT(bcol_quantity) AS times_ordered
FROM
  ((bar_custorder AS bco
    INNER JOIN
      bar_custorder_line AS bcol
      ON bcol.bco_id = bco.bco_id)
    INNER JOIN
      barentree AS be
      ON be.barentree_id = bcol.barentree_id
  )
  INNER JOIN
    customer AS c
    ON c.customer_id = bco.customer_id
WHERE
  c.customer_rewards IS NOT NULL
GROUP BY
  bcol.barentree_id
ORDER BY
  times_ordered DESC;

SELECT
  c.customer_id,
  c.customer_fname,
  c.customer_lname,
  c.customer_rewards,
  kitentree_name,
  COUNT(kcol_quantity) AS times_ordered
FROM
  ((kit_custorder AS kco
    INNER JOIN
      kit_custorder_line AS kcol
      ON kcol.kco_id = kco.kco_id)
    INNER JOIN
      kitentree AS ke
      ON ke.kitentree_id = kcol.kitentree_id
  )
  INNER JOIN
    customer AS c
    ON c.customer_id = kco.customer_id
WHERE
  c.customer_rewards IS NOT NULL
GROUP BY
  kcol.kitentree_id
ORDER BY
  times_ordered DESC;
```

```

1 • Select c.customer_id, c.customer_fname, c.customer_lname, c.customer_rewards,
2     barentree_name, count(bcol_quantity) as times_ordered
3 From ((bar_custorder as bco
4     inner join bar_custorder_line as bcol on bcol.bco_id = bco.bco_id)
5     inner join barentree as be on be.barentree_id = bcol.barentree_id)
6     inner join customer as c on c.customer_id = bco.customer_id
7 Where c.customer_rewards is not null
8 Group By bcol.barentree_id
9 Order By times_ordered Desc;
10 • Select c.customer_id, c.customer_fname, c.customer_lname, c.customer_rewards,
11     kitentree_name, count(kcol_quantity) as times_ordered
12 From ((kit_custorder as kco
13     inner join kit_custorder_line as kcol on kcol.kco_id = kco.kco_id)
14     inner join kitentree as ke on ke.kitentree_id = kcol.kitentree_id)
15     inner join customer as c on c.customer_id = kco.customer_id
16 Where c.customer_rewards is not null
17 Group By kcol.kitentree_id
18 Order By times_ordered Desc;

```

	customer_id	customer_fname	customer_lname	customer_rewards	barentree_name	times_ordered
▶	277	Lauralee	Croydon	55	pharetra magna vestibulum	59
	468	Gayle	Blackesland	20	libero ut massa	58
	351	Inge	Hatchard	93	nunc rhoncus dui	56
	599	Pepillo	Sandell	75	neque libero	56
	398	Bryana	Peabody	72	consequat dui	52
	358	Lowell	Seekings	79	aenean lectus	51
	75	Timothea	Muller	31	sit	51
	236	Vin	Tokley	5	neque	51
	358	Lowell	Seekings	79	faucibus orci luctus	51

Result 79 × Result 80

	customer_id	customer_fname	customer_lname	customer_rewards	kitentree_name	times_ordered
▶	843	Laurette	Richen	53	aliquam	18
	70	Linzy	Rathjen	29	aliquet massa id	18
	387	Reba	Podbury	4	diam vitae quam	16
	851	Berkie	McKune	36	massa donec dapibus	16
	426	Maisey	O'Hearn	68	congue diam	15
	690	Arlyn	Chaffer	33	curae mauris	15
	139	Ogden	Querree	57	risus	14
	927	Coop	Burness	27	donec	14
	319	Rhodia	Bowlesworth	65	eget vulputate	14

Result 79 Result 80 ×

8. Entrees We're Short Ingredients To Make

- Displays all entrees for which necessary ingredients are not in inventory, which ingredients and how many units of it are missing.
- This is relevant to the Managerial staff so they can make orders immediately and restock.

```
SELECT kitentree_name,  
       ingredient_name AS missing_ingredient,  
       kitrecipe_quantity AS units_per_recipe,  
       IFNULL(ki_quantity, 0) AS units_on_hand,  
       (kitrecipe_quantity - IFNULL(ki_quantity, 0)) AS missing_units  
FROM ((kitentree AS ke  
       INNER JOIN kitrecipe AS kr ON kr.kitentree_id = ke.kitentree_id)  
       INNER JOIN ingredient AS i ON i.ingredient_id = kr.ingredient_id)  
       LEFT OUTER JOIN kitinventory AS ki ON ki.ingredient_id = i.ingredient_id  
WHERE kitrecipe_quantity - IFNULL(ki_quantity, 0) > 0  
ORDER BY kitentree_name;
```

```
SELECT kitentree_name,  
       ingredient_name AS missing_ingredient,  
       kitrecipe_quantity AS units_per_recipe,  
       IFNULL(ki_quantity, 0) AS units_on_hand,  
       (kitrecipe_quantity - IFNULL(ki_quantity, 0)) AS missing_units  
FROM ((kitentree AS ke  
       INNER JOIN kitrecipe AS kr ON kr.kitentree_id = ke.kitentree_id)  
       INNER JOIN ingredient AS i ON i.ingredient_id = kr.ingredient_id)  
       LEFT OUTER JOIN kitinventory AS ki ON ki.ingredient_id = i.ingredient_id  
WHERE kitrecipe_quantity - IFNULL(ki_quantity, 0) > 0  
ORDER BY kitentree_name;
```

```
SELECT barentree_name,  
       ingredient_name AS missing_ingredient,  
       barrecipe_quantity AS units_per_recipe,  
       IFNULL(bi_quantity, 0) AS units_on_hand,  
       (barrecipe_quantity - IFNULL(bi_quantity, 0)) AS missing_units  
FROM ((barentree AS be  
       INNER JOIN barrecipe AS br ON br.barentree_id = be.barentree_id)  
       INNER JOIN ingredient AS i ON i.ingredient_id = br.ingredient_id)  
       LEFT OUTER JOIN barinventory AS bi ON bi.ingredient_id = i.ingredient_id  
WHERE barrecipe_quantity - IFNULL(bi_quantity, 0) > 0  
ORDER BY barentree_name;
```

```

SELECT barentree_name,
       ingredient_name AS missing_ingredient,
       barrecipe_quantity AS units_per_recipe,
       IFNULL(bi_quantity, 0) AS units_on_hand,
       (barrecipe_quantity - IFNULL(bi_quantity, 0)) AS missing_units
FROM ((barentree AS be
      INNER JOIN barrecipe AS br ON br.barentree_id = be.barentree_id)
      INNER JOIN ingredient AS i ON i.ingredient_id = br.ingredient_id)
      LEFT OUTER JOIN barinventory AS bi ON bi.ingredient_id = i.ingredient_id
WHERE barrecipe_quantity - IFNULL(bi_quantity, 0) > 0
ORDER BY barentree_name;

```

	kitentree_name	missing_ingredient	units_per_recipe	units_on_hand	missing_units
▶	cursum vestibulum	sultana	3.8	0	3.8
	diam vitae quam	bananas	8.6	0	8.6
	eget vulputate	chopped cilantro fresh	5.2	0	5.2
	enim	taco seasoning mix	7.6	0	7.6
	in ante vestibulum	sour cream	6.3	0	6.3
	in ante vestibulum	cherries	2.2	0	2.2
	justo aliquam quis	garlic	8.2	0	8.2
	lacinia sapien quis	leeks	7.8	0	7.8
	lectus	large garlic cloves	2.2	0	2.2
	lectus	chicken broth	8	0	8

Result 23 × Result 24

	barentree_name	missing_ingredient	units_per_recipe	units_on_hand	missing_units
▶	dui	dark muscovado sugar	2.9	0	2.9
	dui	chicken seasoning mix	9.2	0	9.2
	faucibus orci luctus	soy sauce	6.1	0	6.1
	libero ut massa	grape juice	4.8	0	4.8
	libero ut massa	chicken broth	8.2	0	8.2
	lobortis vel	rum	7.2	0	7.2
	nascetur	beef tenderloin	6.3	0	6.3
	neque	chicken drumsticks	9.7	0	9.7
	neque libero	french bread	5	0	5
	nunc rhoncus dui	skim milk	7.8	6.62	1.1799999999999997

Result 23 Result 24 ×

9. Revenue and Expense Report

- Displays kitchen and bar revenues from customer orders and expenses from inventory orders during a given time period.
- This is relevant to the Managerial Staff and Owner so they can optimize the menu and waste less money ordering overhead on ingredients.

SELECT

```
total_of_kitchen_inventory_orders AS kitchen_order_expenses,  
total_of_bar_inventory_orders AS bar_order_expenses,  
total_of_kitchen_inventory_orders + total_of_bar_inventory_orders  
    AS inventory_order_expenses,  
total_of_kitchen_customer_orders AS kitchen_revenues,  
total_of_bar_customer_orders AS bar_revenues,  
total_of_kitchen_customer_orders + total_of_bar_customer_orders  
    AS customer_order_revenues,  
total_of_kitchen_customer_orders + total_of_bar_customer_orders  
    - total_of_kitchen_inventory_orders -  
total_of_bar_inventory_orders  
    AS net_revenue
```

FROM (

(SELECT

```
        SUM(kit_order_cost) AS total_of_kitchen_inventory_orders
```

FROM

```
    (SELECT SUM(kiol_price * kiol_quantity) AS kit_order_cost  
    FROM kit_invorder AS ki  
    INNER JOIN kit_invorder_line AS kil ON kil.kio_id = ki.kio_id  
    WHERE kio_date BETWEEN '2021-06-01' AND '2021-06-07'  
    GROUP BY ki.kio_id) AS kit_order_costs) AS kit_total_out,
```

(SELECT

```
        SUM(bar_order_cost) AS total_of_bar_inventory_orders
```

FROM

```
    (SELECT SUM(biol_price * biol_quantity) AS bar_order_cost  
    FROM bar_invorder AS bi  
    INNER JOIN bar_invorder_line AS bil ON bil.bio_id = bi.bio_id  
    WHERE bio_date BETWEEN '2021-06-01' AND '2021-06-07'  
    GROUP BY bi.bio_id) AS bar_order_costs) AS bar_total_out,
```

(SELECT

```
        SUM(kit_order_rev) AS total_of_kitchen_customer_orders
```

FROM

```
    (SELECT SUM(kcol_price * kcol_quantity) AS kit_order_rev  
    FROM kit_custorder AS kc  
    INNER JOIN kit_custorder_line AS kcl ON kcl.kco_id = kc.kco_id  
    WHERE kco_date BETWEEN '2021-06-01' AND '2021-06-07'  
    GROUP BY kc.kco_id) AS kit_order_revs) AS kit_total_in,
```

(SELECT

```
        SUM(bar_order_rev) AS total_of_bar_customer_orders
```

FROM

```
    (SELECT SUM(bcol_price * bcol_quantity) AS bar_order_rev  
    FROM bar_custorder AS bc  
    INNER JOIN bar_custorder_line AS bcl ON bcl.bco_id = bc.bco_id  
    WHERE bco_date BETWEEN '2021-06-01' AND '2021-06-07'  
    GROUP BY bc.bco_id) AS bar_order_revs) AS bar_total_in);
```



```

1 • Select total_of_kitchen_inventory_orders as kitchen_order_expenses,
2       total_of_bar_inventory_orders as bar_order_expenses,
3       total_of_kitchen_inventory_orders + total_of_bar_inventory_orders as inventory_order_expenses,
4       total_of_kitchen_customer_orders as kitchen_revenues,
5       total_of_bar_customer_orders as bar_revenues,
6       total_of_kitchen_customer_orders + total_of_bar_customer_orders as customer_order_revenues,
7       total_of_kitchen_customer_orders + total_of_bar_customer_orders
8       - total_of_kitchen_inventory_orders - total_of_bar_inventory_orders as net_revenue
9 From (
10      (Select Sum(kit_order_cost) as total_of_kitchen_inventory_orders
11
12      From (
13          Select Sum(kiol_price * kiol_quantity) as kit_order_cost
14          From kit_invorder as ki
15              inner join kit_invorder_line as kil on kil.kio_id = ki.kio_id
16          Where kio_date between '2021-06-01' and '2021-06-07'
17          Group By ki.kio_id) as kit_order_costs) as kit_total_out,
18      (Select Sum(bar_order_cost) as total_of_bar_inventory_orders
19      From (
20          Select Sum(biol_price * biol_quantity) as bar_order_cost
21          From bar_invorder as bi
22              inner join bar_invorder_line as bil on bil.bio_id = bi.bio_id
23          Where bio_date between '2021-06-01' and '2021-06-07'
24          Group By bi.bio_id) as bar_order_costs) as bar_total_out,
25      (Select Sum(kit_order_rev) as total_of_kitchen_customer_orders
26      From (
27          Select Sum(kcol_price * kcol_quantity) as kit_order_rev
28          From kit_custorder as kc
29              inner join kit_custorder_line as kcl on kcl.kco_id = kc.kco_id
30          Where kco_date between '2021-06-01' and '2021-06-07'
31          Group By kc.kco_id) as kit_order_revs) as kit_total_in,
32      (Select Sum(bar_order_rev) as total_of_bar_customer_orders
33      From (
34          Select Sum(bcol_price * bcol_quantity) as bar_order_rev
35          From bar_custorder as bc
36              inner join bar_custorder_line as bcl on bcl.bco_id = bc.bco_id
37          Where bco_date between '2021-06-01' and '2021-06-07'
38          Group By bc.bco_id) as bar_order_revs) as bar_total_in);

```

	kitchen_order_expenses	bar_order_expenses	inventory_order_expenses	kitchen_revenues	bar_revenues	customer_order_revenues	net_revenue
▶	6372.700000000001	5039.499999999999	11412.2	5296	12520	17816	6403.8

10. Customer Kitchen Order History

- This would be relevant for suggestions to the customer which will allow them to discover more menu items.

```
SELECT
    customer_fname,
    customer_lname,
    kc.kco_id,
    kco_date,
    SUM(kcol_quantity * kcol_price) AS order_total
FROM
    (
        customer AS c
        INNER JOIN
            kit_custorder AS kc
            ON kc.customer_id = c.customer_id
    )
    INNER JOIN
        kit_custorder_line AS kcl
        ON kcl.kco_id = kc.kco_id
WHERE
    c.customer_id =
    (
        SELECT
            customer_id
        FROM
            customer
        WHERE
            customer_fname = 'Selia'
            AND customer_lname = 'Snalom'
    )
GROUP BY
    kc.kco_id
ORDER BY
    kco_date DESC;
```

```
1 • Select customer_fname, customer_lname, kc.kco_id, kco_date, Sum(kcol_quantity * kcol_price) as order_total
2 From (customer as c
3     inner join kit_custorder as kc on kc.customer_id = c.customer_id)
4     inner join kit_custorder_line as kcl on kcl.kco_id = kc.kco_id
5 Where c.customer_id = (
6     Select customer_id
7     From customer
8     Where customer_fname = "Selia"
9     And customer_lname = "Snalom")
10 Group by kc.kco_id
11 Order by kco_date desc;
```

	customer_fname	customer_lname	kco_id	kco_date	order_total
▶	Selia	Snalom	351	2021-06-28 23:46:06	22
	Selia	Snalom	356	2021-05-29 14:32:43	105
	Selia	Snalom	284	2021-05-15 04:11:55	112

11.Lookup Entree by description

- Browsing aid

```
SET
  @pattern = " % garlic % ";
SELECT
  kitentree_name,
  kitentree_descript,
  kitentree_price
FROM
  (
    kitentree AS ke
    INNER JOIN
      kitrecipe AS kr
      ON kr.kitentree_id = kr.kitentree_id
  )
  INNER JOIN
    ingredient AS i
    ON i.ingredient_id = kr.ingredient_id
WHERE
  kitentree_name LIKE @pattern
  OR kitentree_descript LIKE @pattern
  OR ingredient_name LIKE @pattern
  OR ingredient_descript LIKE @pattern
ORDER BY
  kitentree_name;
```

```
Set @pattern = "% garlic %";
Select kitentree_name, kitentree_descript, kitentree_price
From (kitentree as ke
      Inner join kitrecipe as kr on kr.kitentree_id = kr.kitentree_id)
      Inner join ingredient as i on i.ingredient_id = kr.ingredient_id
Where kitentree_name Like @pattern
      Or kitentree_descript Like @pattern
      Or ingredient_name Like @pattern
      Or ingredient_descript Like @pattern
Order by kitentree_name;
```

	kitentree_name	kitentree_descript	kitentree_price
▶	aliquam	Duis bibendum, felis sed interdum venenatis,	69
	aliquam augue	Phasellus in felis. Donec semper sapien a lib	89
	aliquet massa id	Morbi non lectus. Aliquam sit amet diam in ma	33
	amet	Quisque id justo sit amet sapien dignissim ve	59
	bibendum imperdiet	Cras non velit nec nisi vulputate nonummy. Ma	71
	congue diam	Duis consequat dui nec nisi volutpat eleifend	65
	curae mauris	Praesent id massa id nisl venenatis lacinia.	69
	cursus vestibulum	Nullam porttitor lacus at turpis. Donec posue	83

12. Calculate Entree Margins

- Good candidate for a trigger to alert if margin falls below a threshold due to changing supply prices

```
SELECT kitentree_name AS entree,  
       kitentree_price AS entree_price,  
       Sum(unit_price * kitrecipe_quantity) AS entree_cost,  
       kitentree_price - Sum(unit_price * kitrecipe_quantity) AS margin  
FROM ((kitentree as ke  
      LEFT OUTER JOIN kitrecipe AS kr ON kr.kitentree_id = ke.kitentree_id)  
      LEFT OUTER JOIN ingredient AS i ON i.ingredient_id = kr.ingredient_id)  
      LEFT OUTER JOIN (  
        SELECT ingredient_id, Min(unit_price) AS unit_price  
        FROM supply AS s  
        GROUP BY ingredient_id  
      ) AS min_unit_prices ON min_unit_prices.ingredient_id =  
i.ingredient_id  
GROUP BY ke.kitentree_id  
ORDER BY kitentree_price - Sum(unit_price * kitrecipe_quantity) DESC;
```

```
Select kitentree_name As entree,  
       kitentree_price As entree_price,  
       Sum(unit_price * kitrecipe_quantity) As entree_cost,  
       kitentree_price - Sum(unit_price * kitrecipe_quantity) As margin  
From ((kitentree as ke  
      Left Outer Join kitrecipe as kr on kr.kitentree_id = ke.kitentree_id)  
      Left Outer Join ingredient as i on i.ingredient_id = kr.ingredient_id)  
      Left Outer Join (  
        Select ingredient_id, Min(unit_price) As unit_price  
        From supply as s  
        Group by ingredient_id  
      ) as min_unit_prices on min_unit_prices.ingredient_id = i.ingredient_id  
Group By ke.kitentree_id  
Order By kitentree_price - Sum(unit_price * kitrecipe_quantity) Desc;
```

```

SELECT
    barentree_name AS entree,
    barentree_price AS entree_price,
    SUM(unit_price * barrecipe_quantity) AS entree_cost,
    barentree_price - SUM(unit_price * barrecipe_quantity) AS margin
FROM
    (
        (barentree AS be
        LEFT OUTER JOIN
            barrecipe AS br
            ON br.barentree_id = be.barentree_id)
        LEFT OUTER JOIN
            ingredient AS i
            ON i.ingredient_id = br.ingredient_id
        )
    LEFT OUTER JOIN
        (
            SELECT
                ingredient_id,
                MIN(unit_price) AS unit_price
            FROM
                supply AS s
            GROUP BY
                ingredient_id
            )
        AS min_unit_prices
        ON min_unit_prices.ingredient_id = i.ingredient_id
GROUP BY
    be.barentree_id
ORDER BY
    barentree_price - SUM(unit_price * barrecipe_quantity) DESC;

```

```

Select barentree_name As entree,
    barentree_price As entree_price,
    Sum(unit_price * barrecipe_quantity) As entree_cost,
    barentree_price - Sum(unit_price * barrecipe_quantity) As margin
From ((barentree as be
    Left Outer Join barrecipe as br on br.barentree_id = be.barentree_id)
    Left Outer Join ingredient as i on i.ingredient_id = br.ingredient_id)
    Left Outer Join (
        Select ingredient_id, Min(unit_price) As unit_price
        From supply as s
        Group by ingredient_id
    ) as min_unit_prices on min_unit_prices.ingredient_id = i.ingredient_id
Group By be.barentree_id
Order By barentree_price - Sum(unit_price * barrecipe_quantity) Desc;

```

	entree	entree_price	entree_cost	margin
▶	pretium quis	92	46.2	45.8
	nam congue	57	68.2	-11.200000000000003
	sollicitudin vitae	65	90	-25
	quis turpis sed	82	110.39999999999999	-28.39999999999999
	mauris enim	21	57.6	-36.6
	aliquam augue	89	136.79999999999998	-47.79999999999998
	felis	41	132	-91
	porttitor pede justo	35	138	-103
	aliquam	69	176.4	-107.4
	id luctus	6	142.1	-136.1

Result 25 × Result 26

	entree	entree_price	entree_cost	margin
▶	pulvinar lobortis	88	16.099999999999998	71.9
	ligula	83	20.9	62.1
	sit	56	40	16
	vulputate nonummy maecenas	72	108.3	-36.3
	nascetur	66	126	-60
	phasellus sit amet	9	223.2	-214.2
	sapien non mi	56	376.4	-320.4
	aenean lectus	1	415	-414
	libero ut massa	43	471.4	-428.4
	nunc rhoncus dui	19	474	-455

Result 25 Result 26 ×

Stored Procedure

1. Check Supplier of Ingredient

- Lookup Supplier for an input ingredient name, sorted by unit price:

```
CREATE definer = `admin`@`%` `PROCEDURE` `lookup_suppliers_for` (item
VARCHAR(45))
BEGIN
    SELECT
        ingredient_name,
        ingredient_unit,
        supplier_name,
        unit_price,
        supplier_phone
    FROM
        (
            ingredient AS i
            INNER JOIN
                supply AS s
                ON s.ingredient_id = i.ingredient_id
        )
        INNER JOIN
            supplier AS v
            ON v.supplier_id = s.supplier_id
    WHERE
        ingredient_name = item
    ORDER BY
        unit_price;
END
CALL lookup_suppliers_for("olive oil");
```

```
1 • CREATE DEFINER=`admin`@`%` PROCEDURE `Lookup_Suppliers_For`(item Varchar(45))
2 Begin
3     Select ingredient_name, ingredient_unit, supplier_name, unit_price, supplier_phone
4     From (ingredient as i
5         inner join supply as s on s.ingredient_id = i.ingredient_id)
6         inner join supplier as v on v.supplier_id = s.supplier_id
7     Where ingredient_name = item
8     Order By unit_price;
9 End
```

	ingredient_name	ingredient_unit	supplier_name	unit_price	supplier_phone
▶	olive oil	ounces	Price and Sons	8	493-679-8359
	olive oil	cup	Goldner-Hahn	21	905-192-1223
	olive oil	pint	Grady Group	47	728-910-4234
	olive oil	cup	Wehner, Prohaska and Erdman	47	261-478-8232
	olive oil	pint	Balistreri Inc	77	416-591-5952
	olive oil	pint	Ankunding Inc	82	692-189-5865