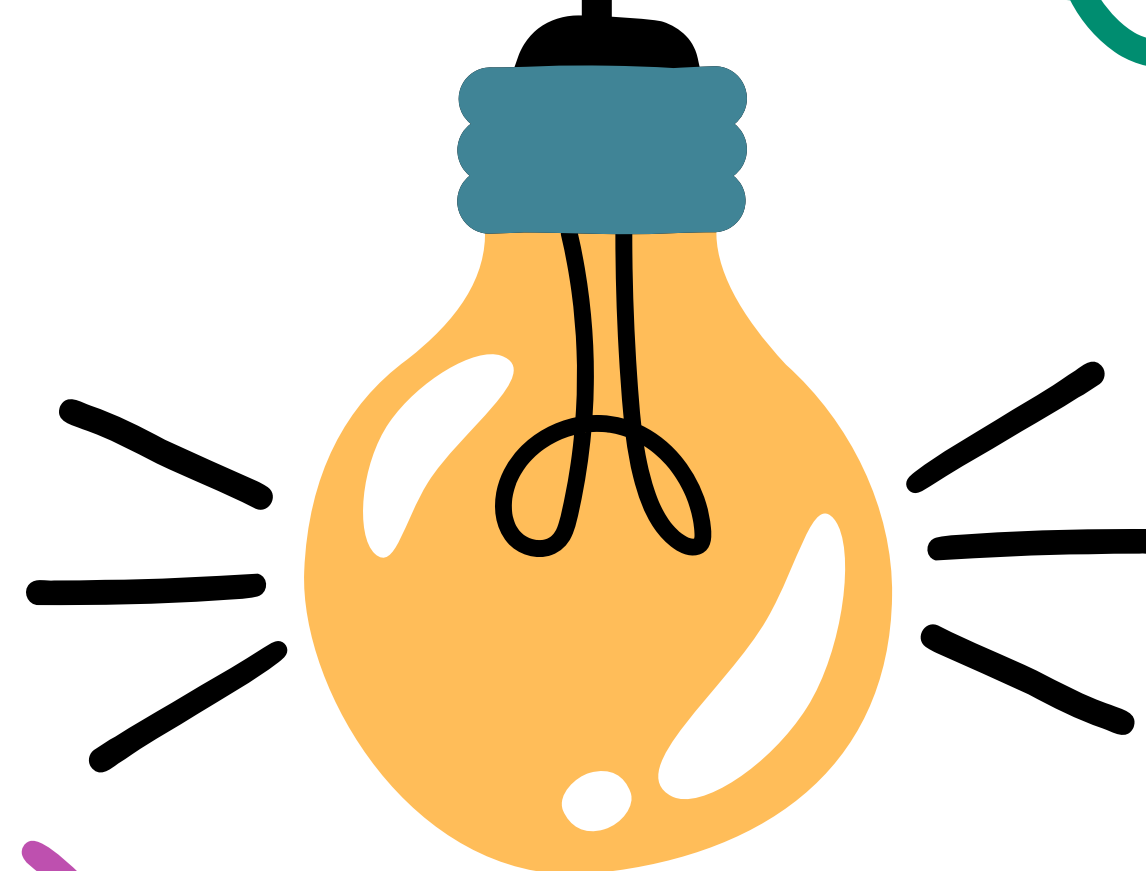
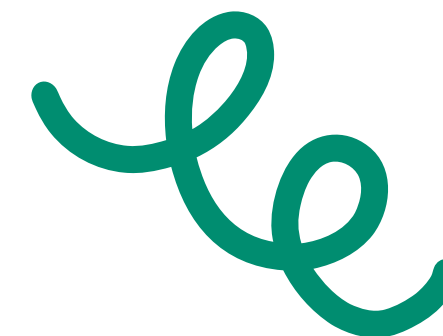




**DELIVR
EATERY**

**SQL
ANALYSIS**



By Andraine Wallace

A comprehensive executive report presenting the quarterly rankings of eateries based on the number of unique users who have placed orders with them.

```
-- Import tablefunc
CREATE EXTENSION IF NOT EXISTS tablefunc;

-- Pivot the previous query by quarter
SELECT * FROM CROSSTAB($$
  WITH eatery_users AS (
    SELECT
      eatery,
      -- Format the order date so "2018-06-01" becomes "Q2 2018"
      TO_CHAR(order_date, '"Q"Q YYYY') AS deliver_quarter,
      -- Count unique users
      COUNT(DISTINCT user_id) AS users
    FROM meals
    JOIN orders ON meals.meal_id = orders.meal_id
    GROUP BY eatery, deliver_quarter
    ORDER BY deliver_quarter, users)

    SELECT
      -- Select eatery and quarter
      eatery,
      deliver_quarter,
      -- Rank rows, partition by quarter and order by users
      RANK() OVER
        (PARTITION BY deliver_quarter
         ORDER BY users DESC) :: INT AS users_rank
    FROM eatery_users
    ORDER BY eatery, deliver_quarter;
  $$)

-- Select the columns of the pivoted table
SELECT (eatery TEXT,
       "Q2 2018" INT,
       "Q3 2018" INT,
       "Q4 2018" INT)
ORDER BY "Q4 2018";
```

Results:

eatery	Q1 2018	Q2 2018	Q4 2018
'The Moon Walk'	1	1	1
'Burgeria'	2	2	2
'Bacon Me Up Scotty'	3	3	3
'Leaning Tower of Pizza'	4	4	4
'Life of Pie'	5	5	5

Average monthly cost before September 2018

```
1  -- CTE named monthly_cost
2  WITH monthly_cost AS (
3      SELECT
4          DATE_TRUNC('month', stocking_date)::DATE AS deliver_month,
5          SUM(meal_cost * stocked_quantity) AS cost
6      FROM meals
7      JOIN stock ON meals.meal_id = stock.meal_id
8      GROUP BY deliver_month)
9
10 SELECT
11     -- Average monthly cost before September
12     AVG(cost)
13 FROM monthly_cost
14 WHERE deliver_month < '2018-09-01';
```

Results:

meal_id	cost
5	12248
4	10211.5
6	8219.75
13	6648.75
Showing 5 out of 5 rows	

Determined the profitability of each eatery to enhance their negotiation capabilities of the usiness Development team.

```
1  -- CTE named monthly_cost
2  WITH monthly_cost AS (
3      SELECT
4          DATE_TRUNC('month', stocking_date)::DATE AS deliver_month,
5          SUM(meal_cost * stocked_quantity) AS cost
6      FROM meals
7      JOIN stock ON meals.meal_id = stock.meal_id
8      GROUP BY deliver_month)
9
10 SELECT
11     -- Average monthly cost before September
12     AVG(cost)
13 FROM monthly_cost
14 WHERE deliver_month < '2018-09-01';
```

Results:

avg

3727.5833333333335

Tracked Delivr profits per month

```
1  -- Revenue CTE
2  WITH revenue AS (
3      SELECT
4          DATE_TRUNC('month', order_date) :: DATE AS deliver_month,
5          SUM(meal_price * order_quantity) AS revenue
6      FROM meals
7      JOIN orders ON meals.meal_id = orders.meal_id
8      GROUP BY deliver_month),
9  -- Cost CTE
10 cost AS (
11     SELECT
12         DATE_TRUNC('month', stocking_date) :: DATE AS deliver_month,
13         SUM(meal_cost * stocked_quantity) AS cost
14     FROM meals
15     JOIN stock ON meals.meal_id = stock.meal_id
16     GROUP BY deliver_month)
17 -- Calculate profit by joining CTEs
18 SELECT
19     revenue - cost AS profit,
20     revenue.delivr_month
21 FROM revenue
22 JOIN cost ON revenue.delivr_month = cost.delivr_month
23 ORDER BY revenue.delivr_month ASC;
```

Results:

profit	delivr_month
4073.5	2018-06-01
6575.5	2018-07-01
9974.25	2018-08-01
15339.5	2018-09-01
23087.5	2018-10-01
38743	2018-11-01
70300.5	2018-12-01

Tracked Delivr profits per month

```
1  -- Revenue CTE
2  WITH revenue AS (
3      SELECT
4          DATE_TRUNC('month', order_date) :: DATE AS deliver_month,
5          SUM(meal_price * order_quantity) AS revenue
6      FROM meals
7      JOIN orders ON meals.meal_id = orders.meal_id
8      GROUP BY deliver_month),
9  -- Cost CTE
10 cost AS (
11     SELECT
12         DATE_TRUNC('month', stocking_date) :: DATE AS deliver_month,
13         SUM(meal_cost * stocked_quantity) AS cost
14     FROM meals
15     JOIN stock ON meals.meal_id = stock.meal_id
16     GROUP BY deliver_month)
17 -- Calculate profit by joining CTEs
18 SELECT
19     revenue - cost AS profit,
20     revenue.delivr_month
21 FROM revenue
22 JOIN cost ON revenue.delivr_month = cost.delivr_month
23 ORDER BY revenue.delivr_month ASC;
```

Results:

profit	delivr_month
4073.5	2018-06-01
6575.5	2018-07-01
9974.25	2018-08-01
15339.5	2018-09-01
23087.5	2018-10-01
38743	2018-11-01
70300.5	2018-12-01

The comparison between the monthly active users (MAUs) of the previous and current month serves as a signal to the Product team. If the number of active users in the current month is lower than that of the previous month, it raises a concern that prompts the Product team's attention.

```
1 WITH mau AS (  
2   SELECT  
3     DATE_TRUNC('month', order_date) :: DATE AS deliver_month,  
4     COUNT(DISTINCT user_id) AS mau  
5   FROM orders  
6   GROUP BY deliver_month)  
7  
8   SELECT  
9     -- Select the month and the MAU  
10    deliver_month,  
11    mau,  
12    COALESCE(  
13      | LAG(mau) OVER(ORDER BY deliver_month ASC),  
14      0) AS last_mau  
15  FROM mau  
16  -- Order by month in ascending order  
17  ORDER BY deliver_month ASC;
```

Results:

deliver_month	mau	last_mau
2018-06-01	123	0
2018-07-01	226	123
2018-08-01	337	226
2018-09-01	489	337
2018-10-01	689	489
2018-11-01	944	689
2018-12-01	1267	944

The overall average revenue per user (ARPU)

```
1 SELECT
2   -- Select the user ID and calculate revenue
3   user_id,
4   SUM(m.meal_price * o.order_quantity) AS revenue
5 FROM meals AS m
6 JOIN orders AS o ON m.meal_id = o.meal_id
7 GROUP BY user_id
8 ORDER BY revenue DESC;
```

Results:

user_id	revenue
18	626
76	553.25
73	537
45	521.25
30	497.75
101	494.25
70	480.25

Showing 100 out of 1304 rows