SQL-05 | Window contd. Date and Time Functions

Lecture Queries

Resources

- PostgreSQL Window functions.
- Mode Window functions.

Question: As a farmer, you want to figure out which of your products were above the average product price on each market date?

```
SELECT
        vendor_id,
        market_date,
        product_id,
        original_price,
        AVG(original_price) OVER (PARTITION BY market_date ORDER BY
       market_date) AS average_cost_product_by_market_date
 FROM farmers_market.vendor_inventory
```

Question: Extract the farmer's products that have prices above the market date's average product cost.

Do it for vendor_id = 8

```
SELECT * FROM
        (
            SELECT
                vendor_id,
                    market_date,
                    product_id,
                    original_price,
                    ROUND(AVG(original_price) OVER (PARTITION BY market_date
        ORDER BY market_date), 2) AS average_cost_product_by_market_date
FROM farmers_market.vendor_inventory )x
WHERE x.vendor_id = 8
                    AND x.original_price > x.average_cost_product_by_market_date
```

Question: Count how many different products each vendor brought to market on each date, and display that count on each row.

```
SELECT
    vendor_id,
    market_date,
    product_id,
    original_price,
    COUNT(product_id) OVER (PARTITION BY market_date, vendor_id)
 vendor_product_count_per_market_date
    FROM farmers_market.vendor_inventory
 ORDER BY vendor_id, market_date, original_price DESC
```

Question: Calculate the running total of the cost of items purchased by each customer, sorted by the date and time and the product_id

```
SELECT customer_id,
    market_date,
    vendor_id,
    product_id,
    quantity * cost_to_customer_per_qty AS price,
    SUM(quantity * cost_to_customer_per_qty) OVER (PARTITION BY
 customer_id ORDER BY market_date, transaction_time, product_id)
AS
 customer_spend_running_total
 FROM farmers_market.customer_purchases
```

When you don't add order by, you get total sum instead of running total:

```
SELECT customer_id,
    market_date,
    vendor_id,
    product_id,
    ROUND(quantity *
cost_to_customer_per_qty, 2) AS price,
    ROUND(SUM(quantity *
cost_to_customer_per_qty) OVER
(PARTITION BY
 customer_id), 2) AS customer_spend_total
 FROM
farmers_market.customer_purchases
```

Question: Using the vendor_booth_assignments table in the Farmer's Market database, display each vendor's booth assignment for each market_date alongside their previous booth assignments.

Follow-up Question: The Market manager may want to filter these query results to a specific market date to determine which vendors are new or changing booths that day, so we can contact them and ensure setup goes smoothly.

Market_date: 2019-04-10

Question: Using the vendor_booth_assignments table in the Farmer's Market database, display each vendor's booth assignment for each market_date alongside their previous booth assignments.

```
SELECT
        market_date,
        vendor_id,
        booth_number,
        LAG(booth_number,1) OVER (PARTITION BY vendor_id ORDER BY market_date,
vendor_id) AS previous_booth_number
FROM farmers_market.vendor_booth_assignments
ORDER BY market_date, vendor_id, booth_number
```

Question: Let’s say you want to find out if the total sales on each market date are higher or lower than they were on the previous market date.

```
SELECT
    market_date,
    SUM(quantity * cost_to_customer_per_qty) AS market_date_total_sales,
    LAG(SUM(quantity * cost_to_customer_per_qty), 1) OVER (ORDER BY
market_date) AS previous_market_date_total_sales
FROM farmers_market.customer_purchases
GROUP BY market_date
ORDER BY market_date
```

Question: From each market_start_datetime, extract the following:

- day of week,
- month of year,
- year,
- hour and
- minute from the timestamp

```
SELECT
        market_start_datetime,
    EXTRACT(DAY FROM
market_start_datetime) AS start_day,
    EXTRACT(YEAR FROM
market_start_datetime) AS date_year,
    EXTRACT(MONTH FROM
market_start_datetime) AS month_of_year,
    EXTRACT(HOUR FROM
market_start_datetime) AS hour_of_day,
    EXTRACT(MINUTE FROM
market_start_datetime) AS minute_of_time
FROM farmers_market.datetime_demo;
```

Question: Let's say you want to calculate how many sales occurred within the first 30 minutes after the farmer's market opened, how would you dynamically determine what cutoff time to use?

```
SELECT
        market_start_datetime,
    DATE_ADD(market_start_datetime, INTERVAL 30
DAY) AS mkt_plus_30mins
FROM farmers_market.datetime_demo;
```

Question: Let’s say we wanted to get a profile of each farmer’s market customer’s habits over time.

```
SELECT customer_id,
    MIN(market_date) AS first_purchase,
    MAX(market_date) AS last_purchase,
    COUNT(DISTINCT market_date) AS
count_of_purchase_dates
 FROM farmers_market.customer_purchases
 WHERE customer_id = 1
 GROUP BY customer_id
```

Question: Write a query that gives us the days between each purchase a customer makes.

```
SELECT customer_id, market_date,
    RANK() OVER (PARTITION BY customer_id
ORDER BY market_date) AS
 purchase_number,
    LEAD(market_date,1) OVER (PARTITION BY
customer_id ORDER BY market_
 date) AS next_purchase
 FROM farmers_market.customer_purchases
```