Growth Analyst - Data Task

Report considerations/requirements

1. Analysis would be conducted for the **previous month** (last closed month)
2. The report granularity level is **city**
3. The dataset was imported correctly, with the correct data type set for each field in the database

Total number of orders

```
WITH date_cte AS (
    -- Retrieve the previous month formatted as 'mm-yyyy'
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
)
SELECT
    city, COUNT(id) as TotalOrders
FROM orders
-- start_time field is the initiation timestamp of an order, while end_time is
delivery
WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT last_closed_month FROM date_cte)
GROUP BY city
ORDER BY 2 DESC;
```

Total number of orders coming from food partners

```
WITH date_cte AS (
    -- Retrieve the previous month formatted as 'mm-yyyy'
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
)
-- The food partners are participating food stores identified by the predicate
`is_food = "TRUE"`
SELECT
    stores.city, COUNT(orders.id) AS TotalFoodPartnerOrders
FROM stores
    JOIN orders ON stores.id = orders.store_id
        AND stores.is_food = "TRUE"
WHERE TO_CHAR(orders.start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
GROUP BY stores.city
ORDER BY 2 DESC;
```

Share of orders that were delivered in less than 45 minutes

```
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
), total_orders_cte AS (
    SELECT
        city, COUNT(id) as TotalOrders
    FROM orders
    WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    GROUP BY city
)
SELECT
    t1.city, ROUND(100 * t1.TotalQuickDeliveries / t2.TotalOrders, 2) AS
QuickDeliveryShare
FROM (
    SELECT
        city, COUNT(id) AS TotalQuickDeliveries
    FROM (SELECT id, city, start_time, end_time, (end_time - start_time) * 1440 as
mins_delivered
            FROM orders
            WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
                AND end_time IS NOT NULL
            ) as Deliveries
    WHERE Deliveries.mins_delivered < 45
    GROUP BY city
) t1 JOIN total_orders_cte t2
    ON t1.city = t2.city
ORDER BY 2 DESC;
```

Share of orders coming from top stores

```
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
), total_orders_cte AS (
    SELECT
        city, COUNT(id) as TotalOrders
    FROM orders
    WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    GROUP BY city
)
SELECT
```

```
    t1.city, ROUND(100 * t1.TotalStoresOrders / t2.TotalOrders, 2) AS
TopStoresShare
FROM (
    SELECT
        stores.city, COUNT(orders.id) AS TotalStoresOrders
    FROM stores
        JOIN orders ON stores.id = orders.store_id
            AND stores.top_store = "TRUE"
    WHERE TO_CHAR(orders.start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    GROUP BY stores.city
) t1 JOIN total_orders_cte t2
    ON t1.city = t2.city
ORDER BY 2 DESC;
```

Share of stores that received no orders

```
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
), total_stores_cte AS (
    SELECT
        city, COUNT(id) as TotalStores
    FROM stores
    GROUP BY city
)
-- We compare stores that received no orders to the total number of stores by city
SELECT
    t3.city, ROUND(100 * t3.NoOrderStores / t4.TotalStores, 2) AS NoOrderShare
FROM (
        -- Stores that received no orders, their id's do not exist in the orders
table
    SELECT
        t1.city, COUNT(t1.id) as NoOrderStores
    FROM stores t1
    WHERE NOT EXISTS (
        SELECT 1
        FROM orders t2
        WHERE t1.id = t2.store_id
            AND TO_CHAR(t2.start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
        )
    GROUP BY t1.city
) t3 JOIN total_stores_cte t4
    ON t3.city = t4.city
ORDER BY 2 DESC;
```

Average spend in euros

```
-- The average spend per order by city for the last closed month
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
)
SELECT
    city, ROUND(AVG(total_cost_eur), 2) AS avg_spend_eur
FROM orders
WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT last_closed_month FROM date_cte)
GROUP BY city
ORDER BY 2 DESC;
```

Difference in average spend in euros between prime and non prime users

```
-- Pivoted differences in average spend can be evaluated using case statements
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
)
SELECT
    t3.city, ROUND(t3.prime_users_avg - t3.non_prime_users_avg, 2) AS
diff_avg_spend_eur
FROM (
    SELECT
        t1.city,
            , AVG(CASE WHEN t2.is_prime = "TRUE" THEN total_cost_eur ELSE 0 END)
AS prime_users_avg
            , AVG(CASE WHEN t2.is_prime = "FALSE" THEN total_cost_eur ELSE 0 END)
AS non_prime_users_avg
    FROM orders t1
        JOIN customers t2 ON t1.customer_id = t2.id
    WHERE TO_CHAR(t1.start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    GROUP BY t1.city
) t3
ORDER BY 2 DESC;
```

Number of customers who made their first order

```
/* Remember we're building a report for the **previous month**. Given that there
are customers who signed up in the previous month also and made no orders, we'll
retrieve information for users who signed up in the previous month and made their
first orders.
```

```
*/
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
    FROM orders
)
SELECT
    t1.preferred_city as city, COUNT(id) AS CustomerCount
FROM customers t1
WHERE TO_CHAR(t1.sign_up_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    AND EXISTS (
        SELECT 1
        FROM orders t2
        WHERE t1.id = t2.customer_id
            AND TO_CHAR(t2.start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    )
GROUP BY t1.preferred_city
ORDER BY 2 DESC;
```

Average monthly orders by recurrent customer

```
-- with recurrent we mean they had also made an order the month before
-- We can conduct membership tests for each user in a subquery of orders in the
month before
-- The average is calculated within two months, i.e. the last closed month and
month before
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -2), 'mm-yyyy') as
the_month_before
    FROM orders
)
SELECT
    t1.customer_id, 0.5 * (t1.last_closed_month_orders +
t2.the_month_before_orders) AS avg_mthly_orders
FROM (
    SELECT
        customer_id, COUNT(id) AS last_closed_month_orders
    FROM orders
    WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    GROUP BY customer_id
) t1 INNER JOIN (
    SELECT
        customer_id, COUNT(id) AS the_month_before_orders
    FROM orders
```

```
    WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT the_month_before FROM date_cte)
    GROUP BY customer_id
) t2 ON t1.customer_id = t2.customer_id
ORDER BY 2 DESC;
```

Average monthly orders by recurrent customer and by city

```
-- with recurrent we mean they had also made an order the month before
-- We can conduct membership tests for each user in a subquery of orders in the
month before
-- The average is calculated within two months, i.e. the last closed month and
month before
-- It is likely that a customer made orders in different cities
WITH date_cte AS (
    SELECT
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -1), 'mm-yyyy') as
last_closed_month
        TO_CHAR(ADD_MONTHS(TRUNC(MAX(start_time), 'MONTH'), -2), 'mm-yyyy') as
the_month_before
    FROM orders
)
SELECT
    t1.city, t1.customer_id, 0.5 * (t1.last_closed_month_orders +
t2.the_month_before_orders) AS avg_mthly_orders
FROM (
    SELECT
        city, customer_id, COUNT(id) AS last_closed_month_orders
    FROM orders
    WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT last_closed_month FROM
date_cte)
    GROUP BY city, customer_id
) t1 INNER JOIN (
    SELECT
        city, customer_id, COUNT(id) AS the_month_before_orders
    FROM orders
    WHERE TO_CHAR(start_time, 'mm-yyyy') = (SELECT the_month_before FROM date_cte)
    GROUP BY city, customer_id
) t2 ON t1.customer_id || t1.city = t2.customer_id || t2.city
ORDER BY 3 DESC;
```