QUERY A

```sql
/*First, let's create 2 CTE: one which contains all the not 'CanceledStatus' orders for the city GLV
on 01-11-2021, and another one just like that
but for the city PLY. I will consider the pickup_time datetime because I care about when orders were
bundled*/

WITH GLV AS
(SELECT * FROM orders
WHERE city_code = 'GLV' AND final_status <> 'CanceledStatus' AND CAST(pickup_time AS DATE) = '2021-
11-01'),

PLY AS
(SELECT * FROM orders
WHERE city_code = 'PLY' AND final_status <> 'CanceledStatus' AND CAST(pickup_time AS DATE) = '2021-
11-01'),

/*Now I want to know, for both cities GLV and PLY, how many (non-canceled) orders were pickedup on
01-11-2021. I defined 2 CTEs which
return these orders, so now I establish 2 new CTEs that will just tell me how many orders I have for
each city (with the mentioned characteristics)*/

G_TOTAL AS
(SELECT COUNT(*) AS gtotal FROM GLV),

P_TOTAL AS
(SELECT COUNT(*) AS ptotal FROM PLY),

/*Now, I am interested in finding out how many non-canceled orders were bundled on 01-11-2021 for
each city (GLV and PLY).
For this I joined the previous CTEs with the bundled_orders table, in order to obtain the order_id
for orders that were in a bundle,
and also that were not 'unbundled', which is a requirement of the exercise.*/

G_BUN AS
(SELECT COUNT(G.order_id) as b_ord_glv
FROM GLV AS G
INNER JOIN
bundled_orders AS bo
ON G.order_id = bo.order_id
WHERE is_unbundled = 'FALSE'),

P_BUN AS
(SELECT COUNT(P.order_id) as b_ord_ply
FROM PLY AS P
INNER JOIN
bundled_orders AS bo
ON P.order_id = bo.order_id
WHERE is_unbundled = 'FALSE')

/*Now with my CTEs I can access the total qty of orders maded & the qty of orders that were bundled
(for each city in 01-11-2021)
So for getting the ratio of orders (X% of total orders were bundled orders) for PLY & GLV, you would
write*/

SELECT (SELECT b_ord_ply FROM P_BUN)/(SELECT ptotal FROM P_TOTAL) AS per_bund_PLY, (SELECT b_ord_glv
FROM G_BUN)/(SELECT gtotal FROM G_TOTAL) AS per_bund_GLV
```

QUERY B

```
/*First, let's grab all the orders that were not canceled, which means final_status <>
'CanceledStatus' */

WITH ORD AS
(SELECT * FROM orders
WHERE final_status <> 'CanceledStatus'),

/*Now, I want all the orders that were ultimately bundled, and for this is_unbundled has to be
'FALSE'.*/

BUN AS
(SELECT * FROM bundled_orders
WHERE is_unbundled='FALSE'),

/*Let's make a left join so that we keep all the orders, and obtain new information for the orders
that were part of a bundle*/

ALLTS AS
(SELECT O.order_id, O.city_code, O.pickup_time, O.enters_delivery, O.pd_dist, B.bundle_id,
B.is_bundled
FROM ORD AS O
LEFT JOIN
BUN AS B
ON O.order_id = B.order_id),

/*Following that, for the orders that were indeed part of a bundle, we need to consider (for
calculating our average by city)
the order in each bundle with the smallest pd_dist value, and exclude the other orders from our
calculation.
So I groupby bundle_id, order_id and keep the order_id with the min pd_dist (for the orders part of
a bundle)*/

MIN_ORDER AS
(SELECT bundle_id, order_id, MIN(pd_dist) AS mindis
FROM ALLTS
WHERE is_bundled = 'TRUE'
GROUP BY bundle_id, order_id),

/*Now, I want to keep the order_ids from the CTE ALLTS that fulfill at least 1 of these 2
conditions:

-Either the is_bundled value (for the order) is <> 'TRUE' (which means the order is NOT part of a
bundle)
and therefore can be considered by itself in the average calculation.

-The order_id is in the MIN_ORDER CTE (this would imply 2 things, first that they are orders part of
a bundle
and secondly and more important is that they are the order in the bundle with the minimum pd_dist,
which we have been told it is the one we should consider for our calculation of the average */

FINAL AS
(SELECT order_id, city_code, pickup_time, enters_delivery, pd_dist, bundle_id, is_bundled
FROM ALLTS
WHERE (is_bundled <> 'TRUE') OR (order_id IN (SELECT order_id FROM MIN_ORDER)))

/*Finally, we will calculate an average in minutes (time between pickup and delivery)
by city for the orders that fulfill at least 1 of the 2 criteria explained above, and for the last
30 days
considering the date the moment the query is run.*/

SELECT city_code, AVG(DATEDIFF(minute, pickup_time, enters_delivery))
FROM FINAL
WHERE CAST(pickup_time AS DATE) BETWEEN (SELECT DATEADD(dd, -30, (SELECT GETDATE())))
AND
(SELECT GETDATE())
GROUP BY city_code
```