# Part (1)

There are two sample tables in this small test, one table called "vanorder" and one table called "vaninterest".

Logic of the sample system: A record is created in van-order when user places an order. Each order starts at order\_subset A, and each order can be accepted by multiple drivers. There will be a new record in the vaninterest table when a driver accepts an order. After accepting order, the driver has the option to reject an order, which will go back to the pool and allow other drivers to match this order again.

1) vanorder

- Contains the final status of the orders information

- idvanOrder: The order ID, which is the primary key

- order\_status: The final status of the order.

-- 2: 'Completed'

-- 3: 'Cancelled'

-- 5: 'Expired'

- order\_subset: The final order subset of the order

- requestor\_client\_id: The user account ID

- servicer\_auth: The driver ID

- total\_price: The price of the order

- order\_datetime: The order time of the order

- txCreate: The record creation time, which is the order placed time by the user

2) vaninterest

- Contains the order information for each subset

- idvanInterest: The primary key of this table

- idvanOrder: The order ID associated to this record

- order\_subset\_assigned: The order subset that is associated to this record

- servicer\_auth: The driver ID for this subset

- txCreate: The record creation time, which is the time that this driver accepted the order

We are using MySQL for the tables. If you have a MySQL database available, you can run the SQL file, p1\_data.sql, to create two tables in the database. The two files, vaninterest.csv and vanorder.csv, contain the same data.

Now we are write a few SQL statement to understand more about our data, users, and drivers. Unless specified otherwise, use all data available in the sample tables. Please use one SQL statement for finding each of the following:

### a) For hours with orders, how many orders are there each hour based on order time?

Please refer to attached Part1-a.csv

SQL Code:

SELECT

FORMAT\_TIMESTAMP('%Y-%m-%d %H:00:00',order\_datetime) AS Hour,

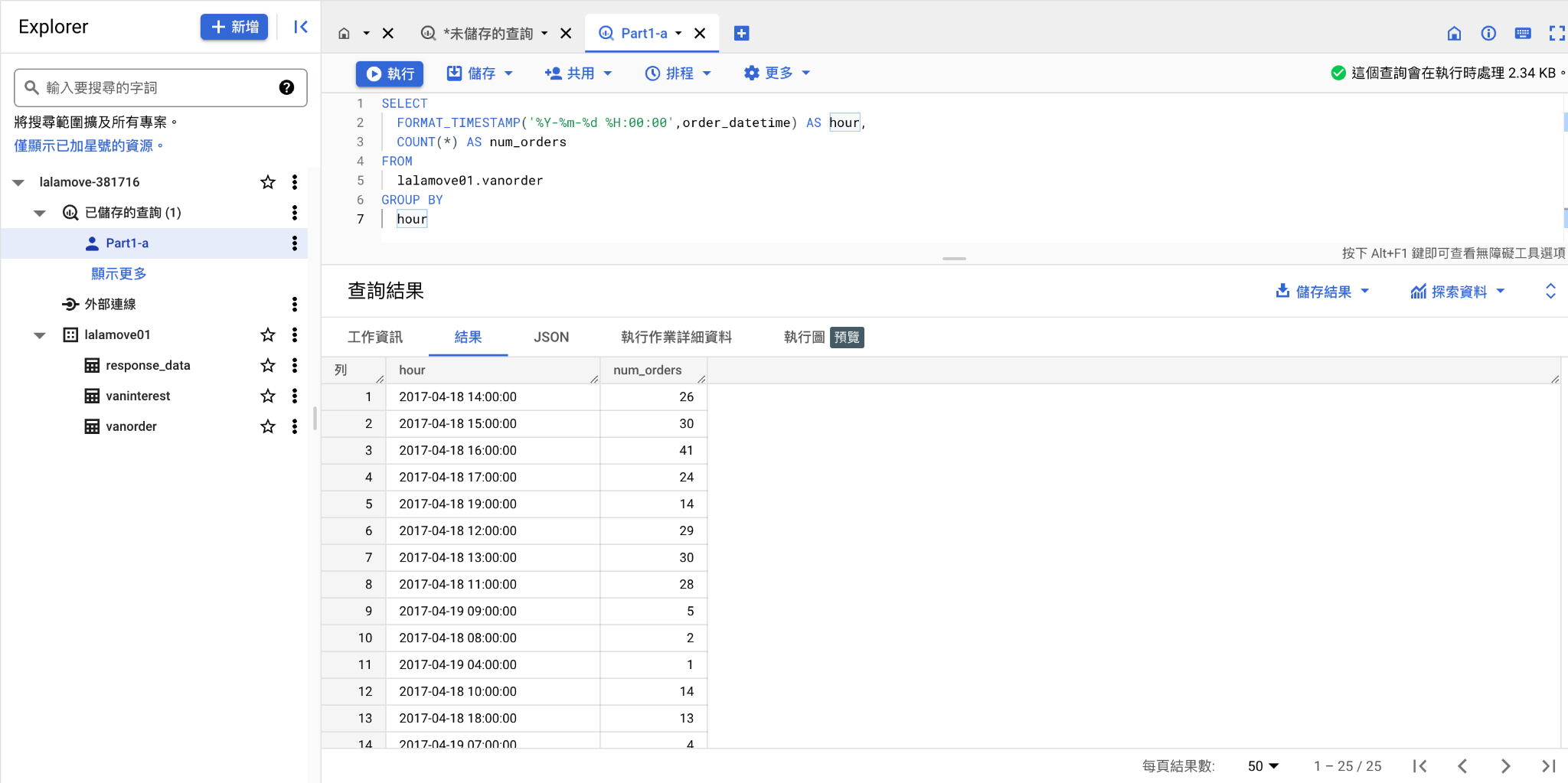
COUNT(idvanOrder) AS num\_orders

FROM

lalamove01.vanorder

GROUP BY

Hour



### b) What is the percentage of money spent for each of the following group of clients?

### - Clients who completed 1 order

### - Clients who completed more than 1 order

Please refer to attached Part1-b.csv

SQL Code:

WITH order\_summary AS (

SELECT

requestor\_client\_id,

COUNT(\*) AS count\_orders,

SUM(total\_price) AS total\_price

FROM

lalamove01.vanorder

WHERE

order\_status = 2

GROUP BY

requestor\_client\_id

)

SELECT

(

CASE

WHEN count\_orders = 1 THEN '1 order'

ELSE 'more than 1 order'

END

) AS order\_group,

SUM(total\_price) AS total\_spent,

(

SUM(total\_price) /

(

SELECT

SUM(total\_price)

FROM

lalamove01.vanorder)

) \* 100

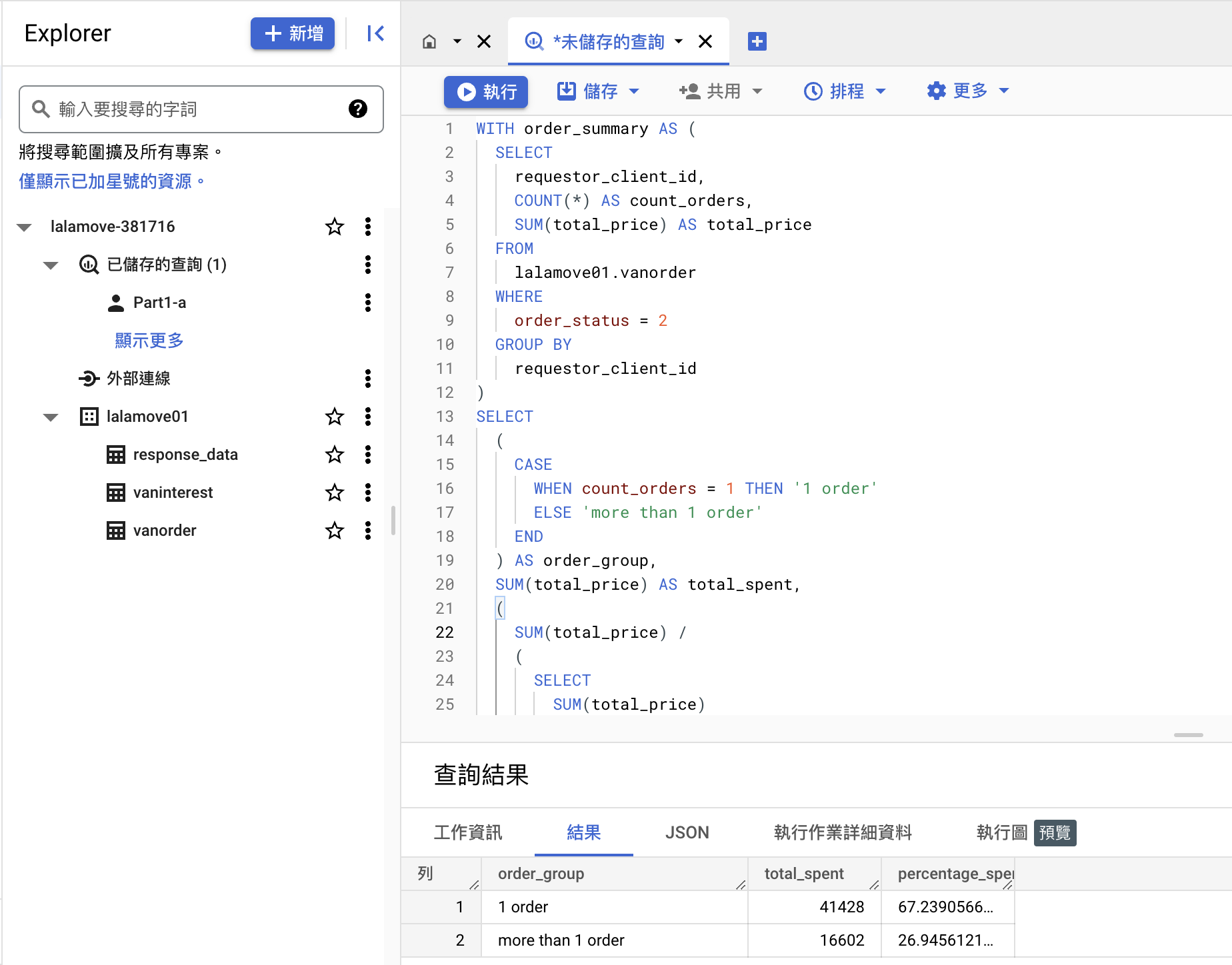
AS percentage\_spent

FROM

order\_summary

GROUP BY

order\_group



### c) List of unique Client ID who completed at least one order, also show each client's total money spent, and the total order(s) completed. Order the list by total money spent (descending), then by total order(s) completed (descending)

Please refer to attached Part1-c.csv

SQL Code:

SELECT

requestor\_client\_id,

COUNT(\*) AS total\_orders\_completed,

SUM(total\_price) AS total\_spent

FROM

lalamove01.vanorder

WHERE

order\_status = 2

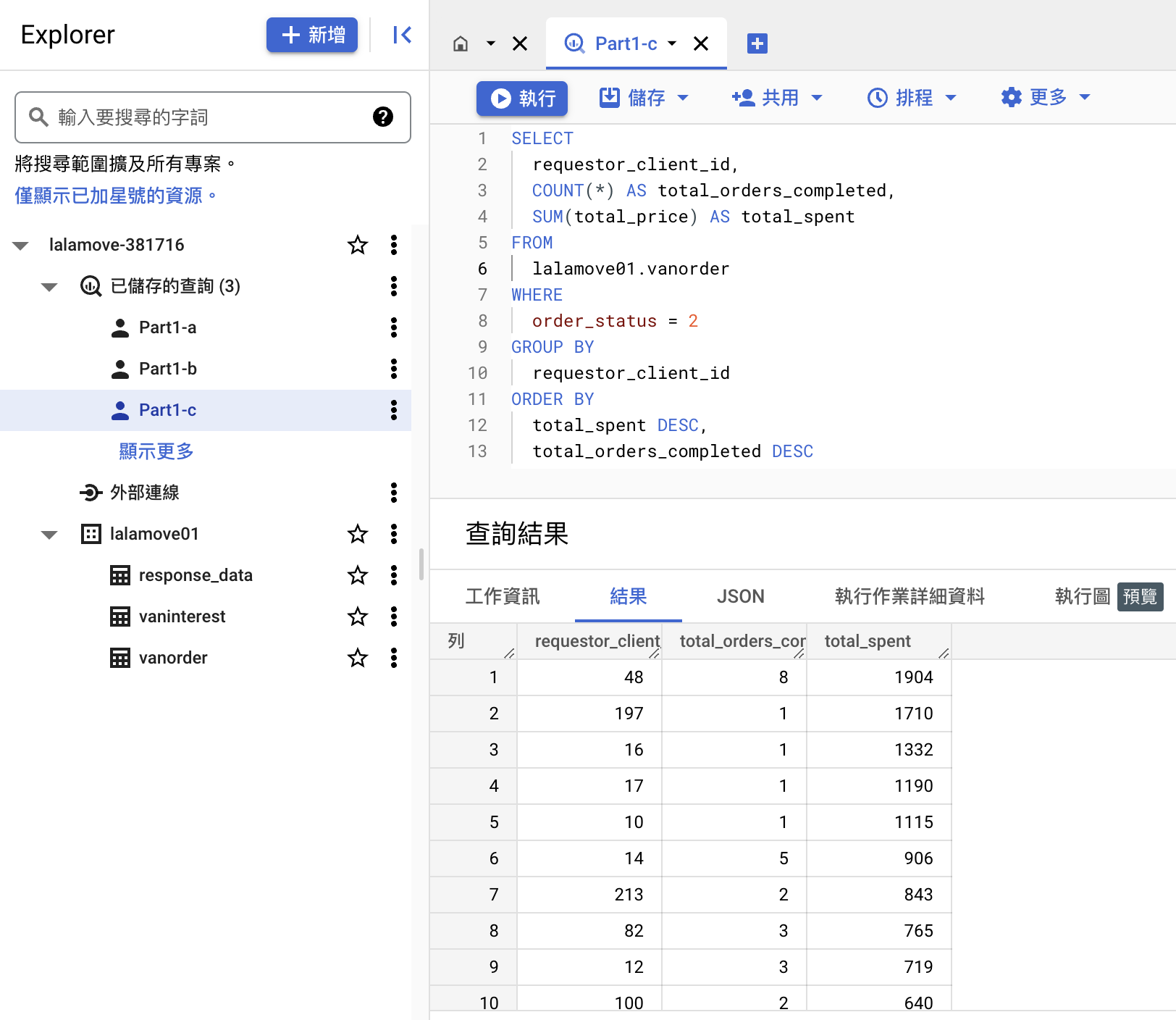
GROUP BY

requestor\_client\_id

ORDER BY

total\_spent DESC,

total\_orders\_completed DESC



### d) List of all drivers who took order(s) (regardless of whether they eventually complete the order), also show each driver's total income and total order(s) completed. Order the list by total income (descending), then by total order(s) completed

Please refer to attached Part1-d.csv

SQL Code:

SELECT

servicer\_auth,

COUNT(DISTINCT idvanOrder) AS total\_orders\_completed,

SUM(total\_price) AS total\_income

FROM

lalamove01.vanorder

WHERE

servicer\_auth IS NOT NULL

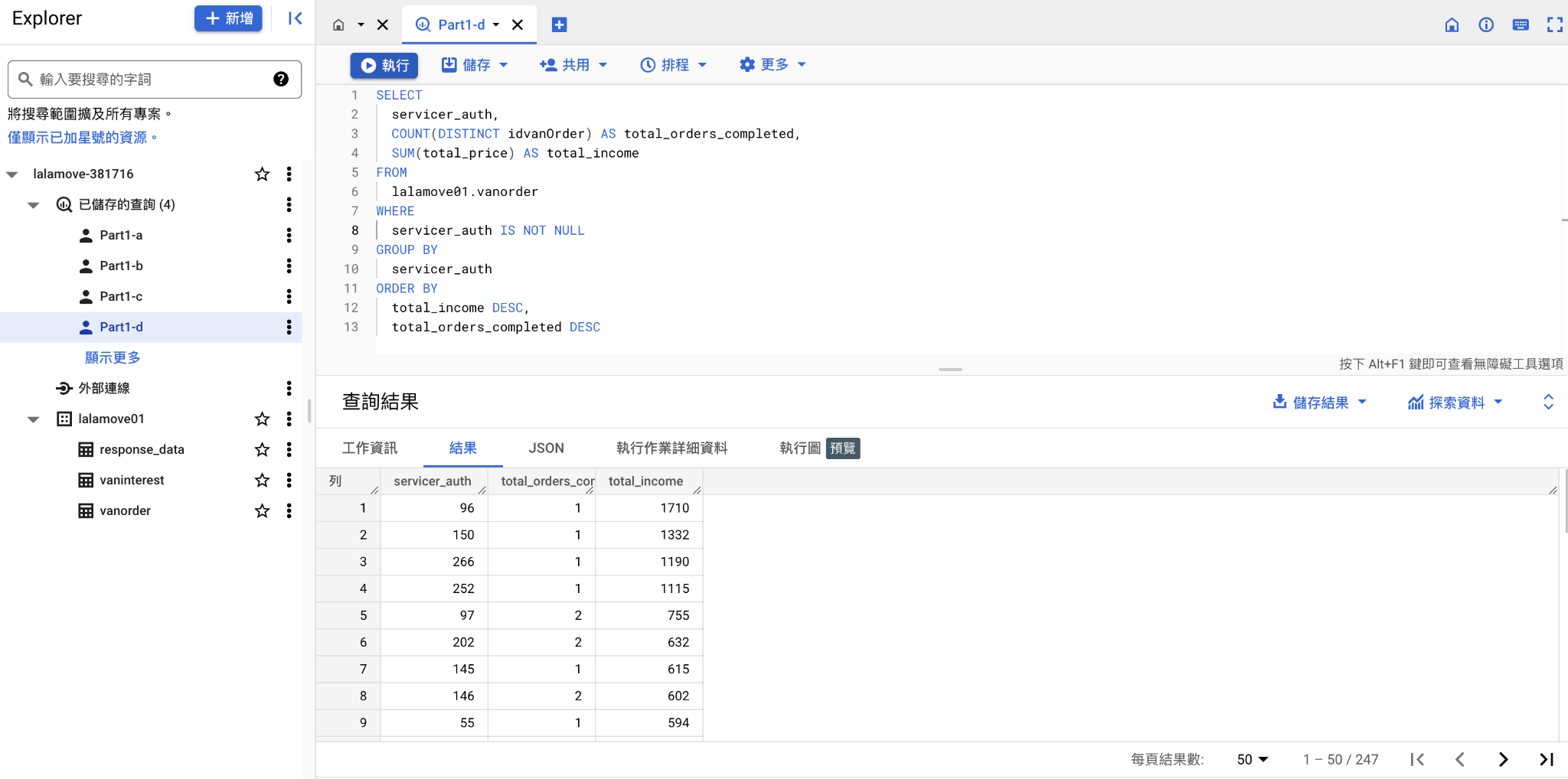
GROUP BY

servicer\_auth

ORDER BY

total\_income DESC,

total\_orders\_completed DESC



### e) List of driver ID who took orders, but never complete an order?

Please refer to attached Part1-e.csv

SQL Code:

SELECT DISTINCT

servicer\_auth

FROM

lalamove01.vaninterest

WHERE

servicer\_auth NOT IN (

SELECT DISTINCT

servicer\_auth

FROM

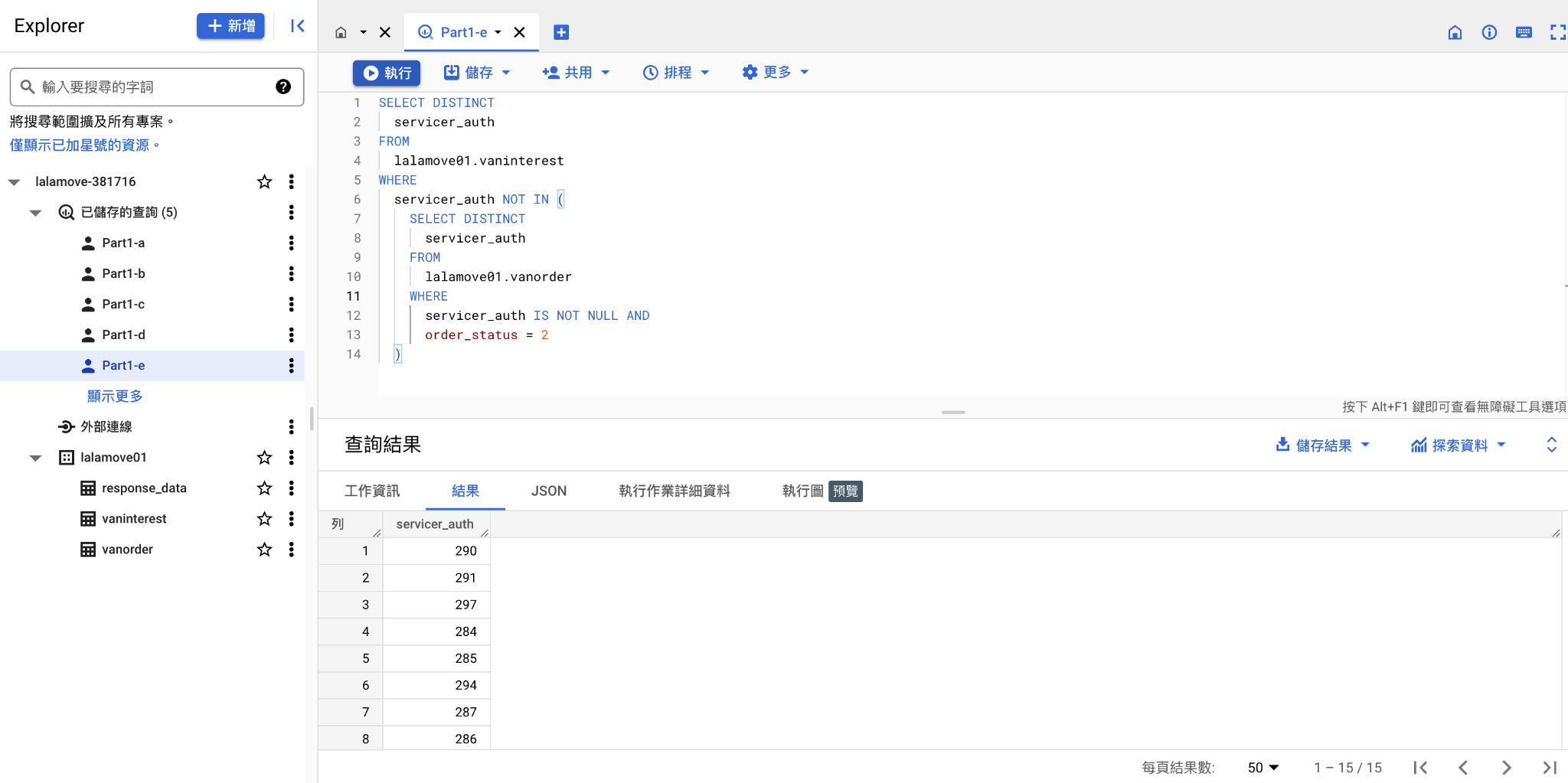
lalamove01.vanorder

WHERE

servicer\_auth IS NOT NULL AND

order\_status = 2

)



# Part (2)

We would like to test the impact of a change introduced at ‘2017-03-30 12:00:00 AM’ on our Order Allocation system. The objective of the change is to improve order Match Time, which is the duration since the time an order was created to the time the order was accepted by a driver. The file response\_data.csv contains the order records you’d need for this part of the challenge.

Feel free to derive your answers to the following questions using any techniques and softwares of your choice:

a) What is the resulting impact from this change?

b) Any additional insights or observations you think are worth noting?

c) Any observations regarding the testing process or environment? How would you improve or redesign this experiment?

Your answer will be evaluated based on the following aspects (you can provide your answers in any form you like):

- Your thought process and how you get to your results

- Your visualization and statistical knowledge

- Your presentation and delivery of results

### a) What is the resulting impact from this change?

Order match time before and after the change:

From Histogram:

* Distribution of the match time after change skewed to the right side slightly which means the overall matching time has slightly improved.

From Box plot diagram:

* Match time before change has a long tail indicating that there are some orders that take much longer to match. The distribution for orders placed after the change date has a slightly shorter tail.

Hypothesis test :

* The t-test result shows that the t-statistic is negative (-0.3586) and the p-value is greater than 0.05 (0.71987). This indicates that we fail to reject the null hypothesis that there is no significant difference in the match time for orders placed before and after the change date.

Based on this analysis, we can conclude that the change introduced at ‘2017-03-30 12:00:00 AM’ did not have a significant impact on the order match time.

Python Code:

**import** pandas as pd

order\_data **=** pd.read\_csv('/Users/dolphinchan/Downloads/Data\_Analyst\_Test/response\_data.csv')

order\_data['Order Create Timestamp'] **=** pd.to\_datetime(order\_data['Order Create Timestamp'])

order\_data['Driver Response Timestamp'] **=** pd.to\_datetime(order\_data['Driver Response Timestamp'])

before\_change = order\_data[order\_data['Order Create Timestamp'] **<** '2017-03-30 12:00:00 AM']

after\_change = order\_data[order\_data['Order Create Timestamp'] **>=** '2017-03-30 12:00:00 AM']

before\_change['before\_match\_time'] **=** before\_change['Driver Response Timestamp'] - before\_change['Order Create Timestamp']

after\_change['after\_match\_time'] **=** after\_change['Driver Response Timestamp'] - after\_change['Order Create Timestamp']

**import** matplotlib.pyplot **as** plt

before\_change['before\_match\_time'].astype('timedelta64[s]').plot.hist(bins **=** **60**, histtype **=** u'step', range **=** [**0,200**], color **=** 'blue', label **=** 'Before Change')

after\_change['after\_match\_time'].astype('timedelta64[s]').plot.hist(bins **=** **60**, histtype **=** u'step', range **=** [**0,200**], color **=** 'Orange', label **=** 'After Change')

plt.legend(title **=** 'Change Time')

plt.xlabel('Match Time (seconds)')

plt.ylabel('Count')

plt.show()

**import** matplotlib.pyplot **as** plt

data **=** [before\_change['before\_match\_time'].astype('timedelta64[s]'), after\_change['after\_match\_time'].astype('timedelta64[s]')]

fig **=** plt.figure(figsize =(10, 7))

ax **=** fig.add\_subplot(111)

ax.set\_xticklabels(['before', 'after'])

plt.boxplot(data)

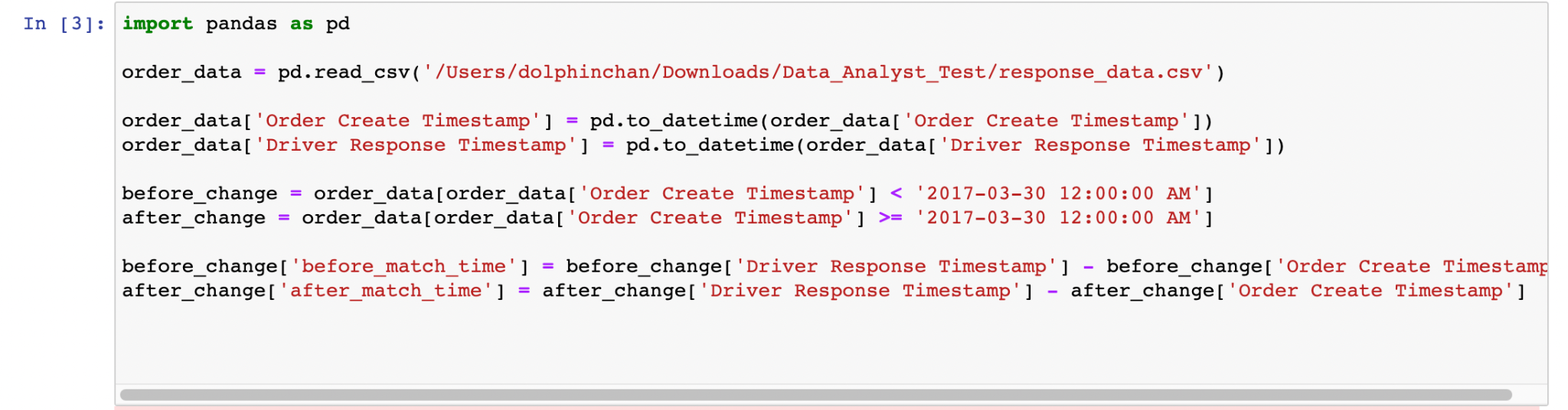
plt.show()

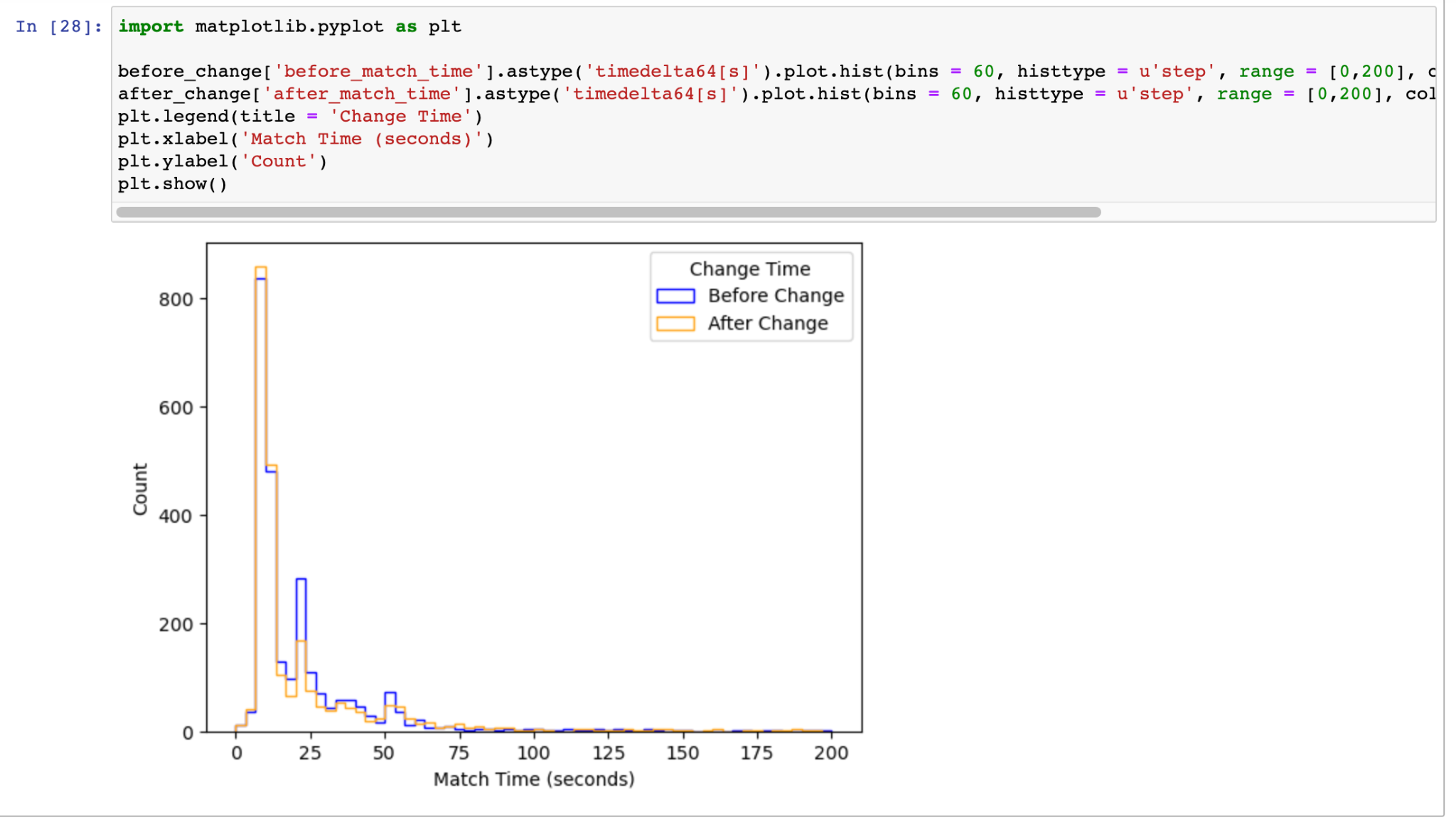
**import** scipy.stats **as** stats

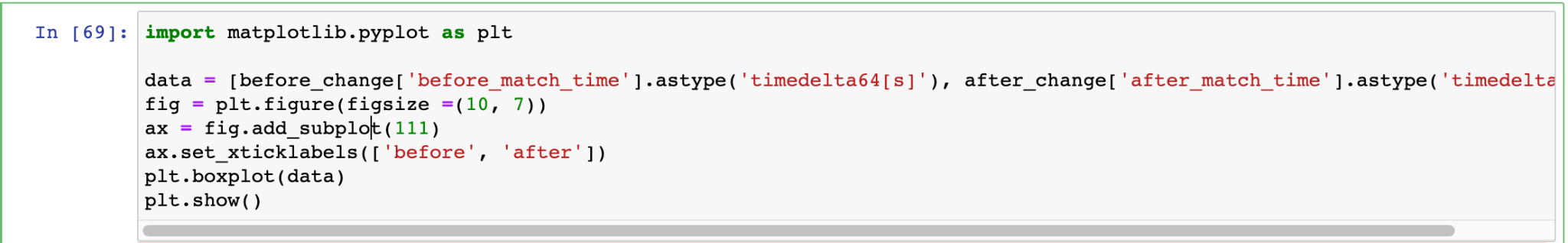
t\_stat, p\_value **=** stats.ttest\_ind(before\_change['before\_match\_time'].astype('timedelta64[s]').dropna(), after\_change['after\_match\_time'].astype('timedelta64[s]').dropna(), equal\_var **=** **False**)

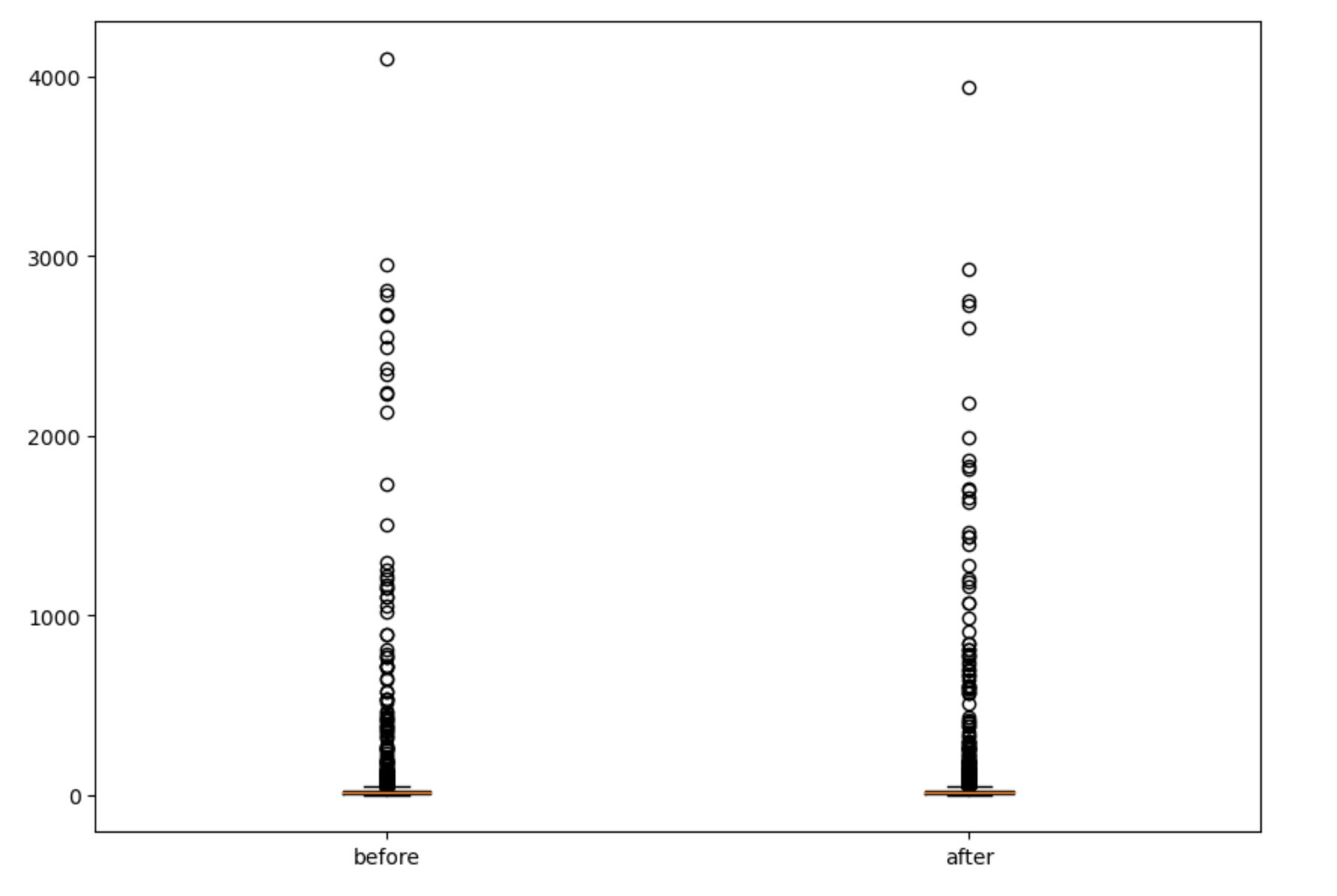
**print**('T-statistic:', t\_stat)

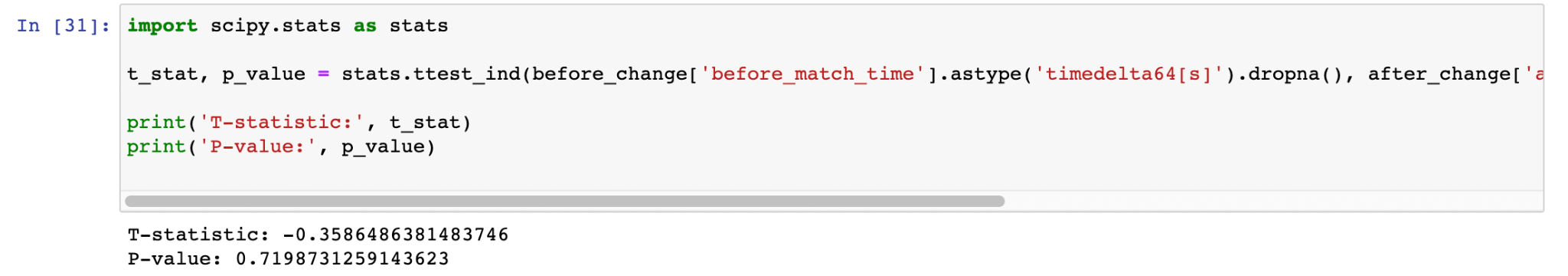
**print**('P-value:', p\_value)











### b) Any additional insights or observations you think are worth noting?

The distribution of the match time is skewed to the right, indicating that the majority of the orders are matched relatively quickly, but there are some orders that take a long time to match.

The change introduced may have been targeted at improving the match time for orders that take a long time.

### c) Any observations regarding the testing process or environment? How would you improve or redesign this experiment?

1. Control for any other factors that may have impacted the order match time, such as changes in demand or driver availability, special date/ event.
2. Running the experiment for a longer period of time to ensure that the results are stable over time.