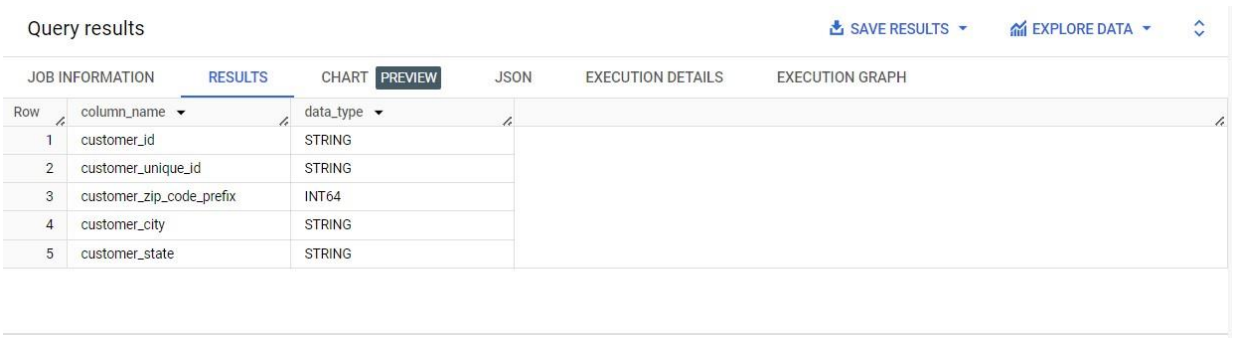
Q1. Business Case: Target SQL

1. : Data type of all columns in the "customers" table.

ANS.

```
SELECT
  column_name,
  data_type
FROM
  `business-case-1-413214.123.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```



Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

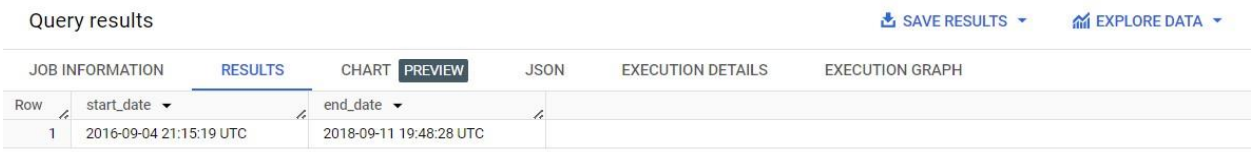
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

Inference: here I used where filter and selected column name and data type from business case.information_schema.columns

2. Get the time range between which the orders were placed.

Ans

```
SELECT
  MIN(order_purchase_timestamp
) AS start_date,
  MAX(order_delivered_carrier_date
) AS end_date
FROM
  `business-case-1-413214.123.orders`;
```



Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	start_date	end_date
1	2016-09-04 21:15:19 UTC	2018-09-11 19:48:28 UTC

Inference: I used min and max function in order_purchase_timestamp and order_delivered_carrier_date and finaly used timestamp diff(from timestamp,)

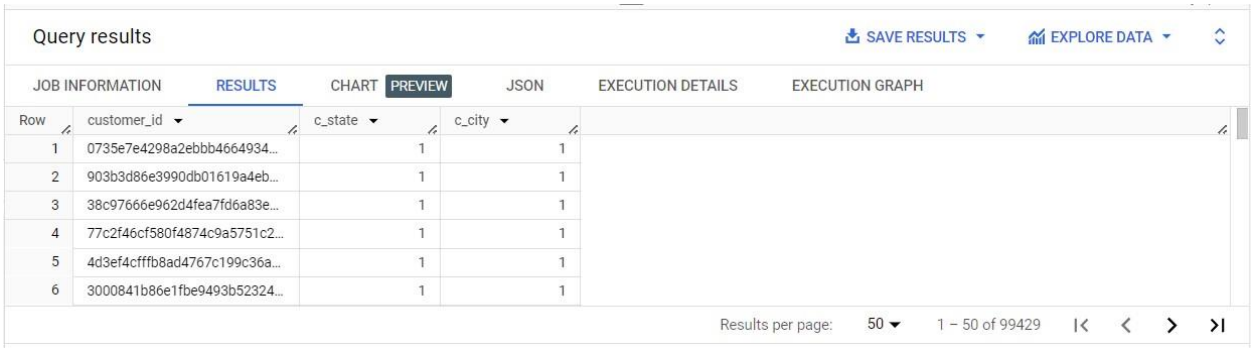
To find out days between above dates I used another query as follows

```
SELECT
  TIMESTAMP_DIFF(TIMESTAMP('2018-09-11 19:48:28'), TIMESTAMP('2016-09-04 21:15:19'),
DAY) AS days_between;
```

Sample out put

3 .Count the Cities & States of customers who ordered during the given period.

```
select customer_id,
      count(customer_state) as c_state,
      count(customer_city) as c_city
from `business-case-1-413214.123.customers`
where customer_id in ( select customer_id
from `business-case-1-413214.123.orders`
where order_purchase_timestamp
BETWEEN '2016-09-04' AND   '2018-09-11'
)
group by 1
```

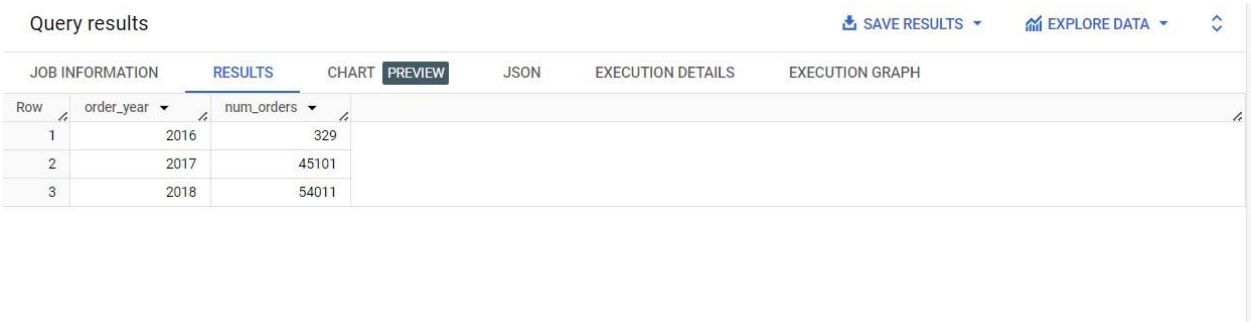


Inference: here I used count function in city and state and I used where filter with given time stamp

2. In-depth Exploration:

1. Is there a growing trend in the no. of orders placed over the past years?

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
  COUNT(*) AS num_orders
FROM
  `business-case-1-413214.123.orders`
GROUP BY
  order_year
ORDER BY
  order_year;
```

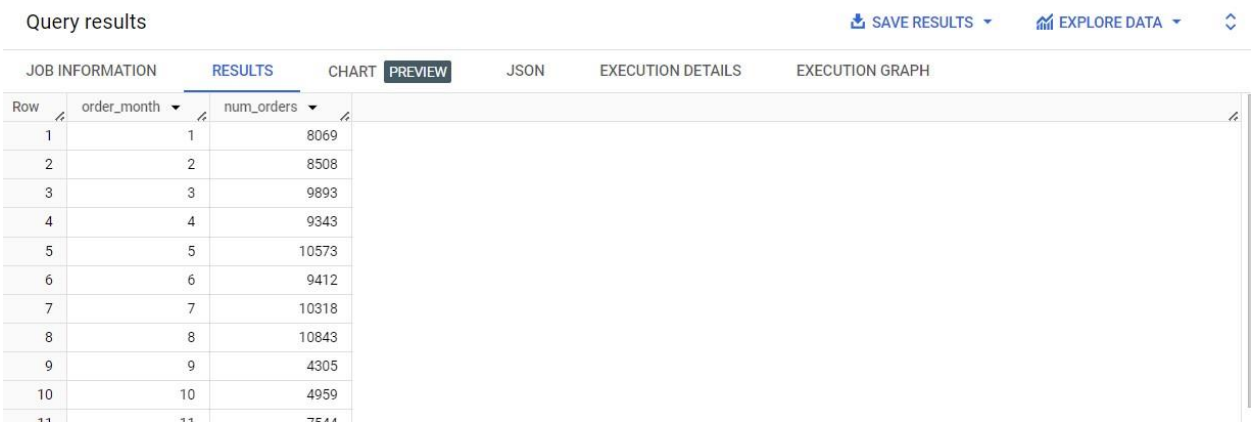


Row	order_year	num_orders
1	2016	329
2	2017	45101
3	2018	54011

Inference : here I used extract function for years in order purchase timestamp after that I used count

2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```
SELECT
  EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
  COUNT(*) AS num_orders
FROM
  `business-case-1-413214.123.orders`
GROUP BY
  order_month
ORDER BY
  order_month;
```



Row	order_month	num_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

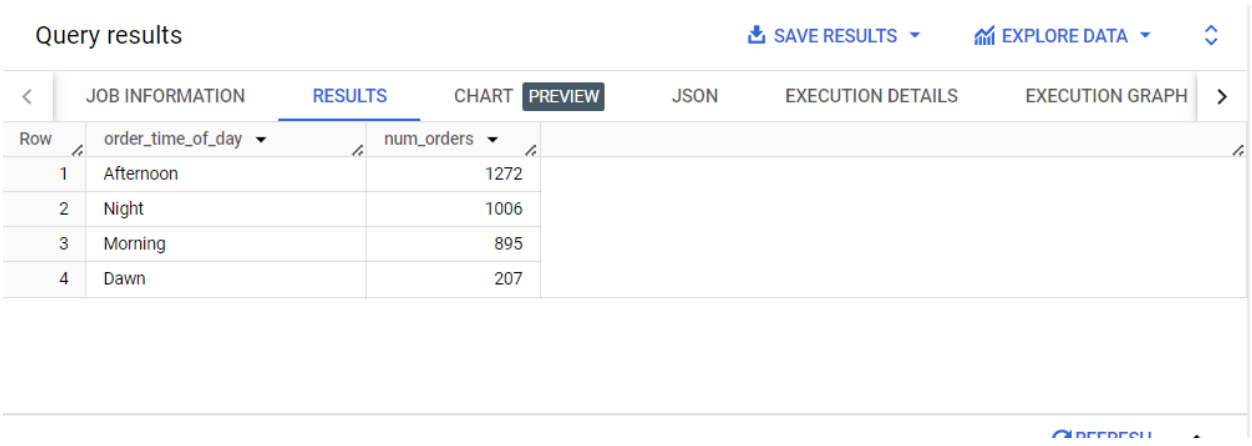
Inference: extracted month from order_purchase_timestamp and used count functions.

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    1. 0-6 hrs : Dawn
    2. 7-12 hrs : Mornings
    3. 13-18 hrs : Afternoon
    4. 19-23 hrs : Night

    ANS:

```
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'Afternoon'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
    ELSE 'Unknown'
  END AS order_time_of_day,
  COUNT(*) AS num_orders
FROM
  `business-case-1-413214.123.orders` AS a

join `business-case-1-413214.123.customers` as b
on a.customer_id = b.customer_id
WHERE
  customer_state = 'BA'
GROUP BY
  1, customer_state
ORDER BY
  num_orders DESC;
```



Row	order_time_of_day	num_orders
1	Afternoon	1272
2	Night	1006
3	Morning	895
4	Dawn	207

Inference: join on customer_id of orders and customer table along with case when inside extract hours used

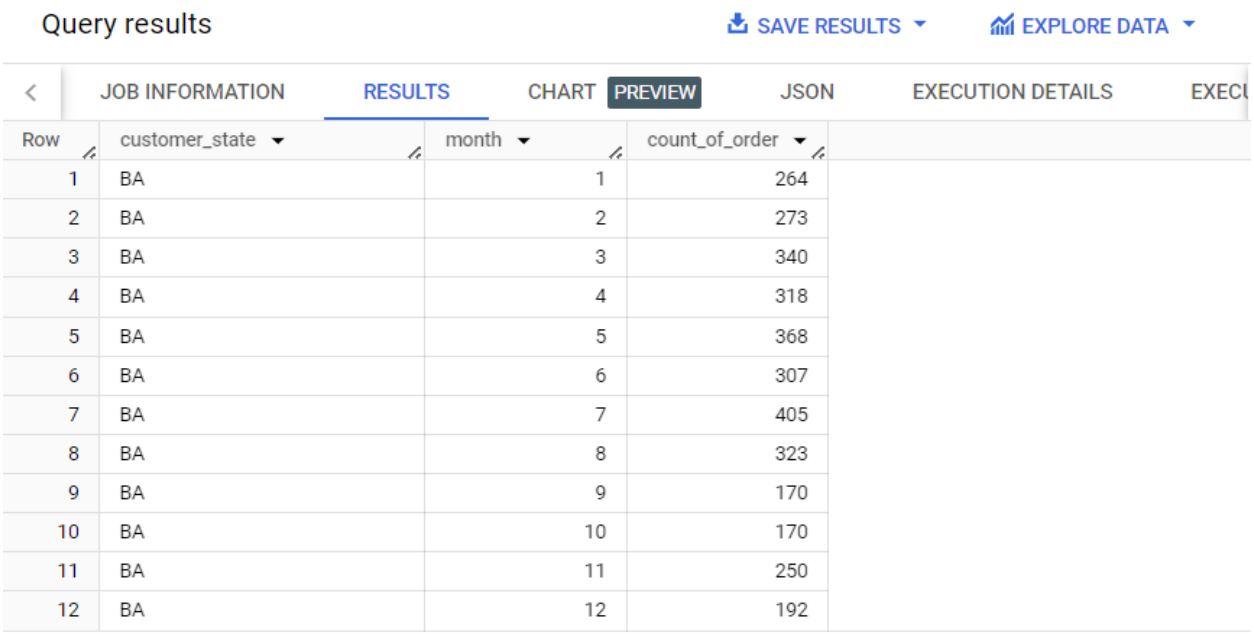
3. Evolution of E-commerce orders in the Brazil region:

1. Get the month on month no. of orders placed in each state.

ANS

```
select b.customer_state,
 EXTRACT(month FROM order_purchase_timestamp) as month,
 count(a.order_id) as count_of_order


from `business-case-1-413214.123.orders` as a
join `business-case-1-413214.123.customers` as b
on a.customer_id = b.customer_id
where b.customer_state = 'BA'
group by 1,2
order by 2
```



Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS | EXECU

Row	customer_state	month	count_of_order
1	BA	1	264
2	BA	2	273
3	BA	3	340
4	BA	4	318
5	BA	5	368
6	BA	6	307
7	BA	7	405
8	BA	8	323
9	BA	9	170
10	BA	10	170
11	BA	11	250
12	BA	12	192

Inference: I joined order with customer along with I extracted no of months with respected to no of order.

2. How are the customers distributed across all the states?

```
SELECT customer_state,
      count(*) as num_customer
from `business-case-1-413214.123.customers`
group by 1
order by 1
```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECU

Row	customer_state	num_customer
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635

Inference: the count functions has used for this question

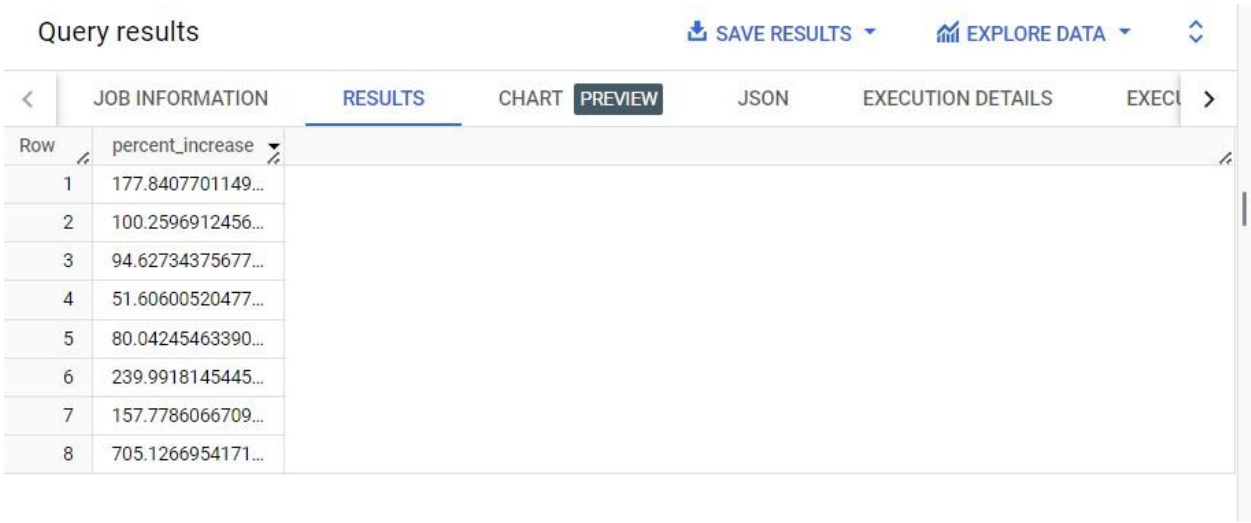
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
   You can use the "payment_value" column in the payments table to get the cost of orders.

```
WITH OrderCost AS (
  SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
    SUM(payment_value) AS total_payment_value
  FROM
    `business-case-1-413214.123.orders` as a
  join `business-case-1-413214.123.payments` as b
  on a.order_id = b.order_id
  WHERE
    EXTRACT(YEAR FROM order_purchase_timestamp) IN (2017, 2018)
    AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY
    order_year, order_month
  order by 1,2
)

SELECT
  (o18.total_payment_value - o17.total_payment_value) / o17.total_payment_value *
100 AS percent_increase
```

```
FROM
  OrderCost o17
JOIN
  OrderCost o18
ON
  o17.order_month = o18.order_month
WHERE
  o17.order_year = 2017
  AND o18.order_year = 2018;
```



Inference: CTE is used for find out the percentage increase of previous year month to next year month.

2. Calculate the Total & Average value of order price for each state.

ANS:

```
select customer_state,
     sum(payment_value) as total_sum,
     avg(payment_value) as total_avg

from `business-case-1-413214.123.customers` as a
join `business-case-1-413214.123.orders` as b
on a.customer_id = b.customer_id
join `business-case-1-413214.123.payments` as c
on b.order_id = c.order_id
group by 1
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	total_sum	total_avg
1	RJ	2144379.689999...	158.5258882235...
2	RS	890898.5399999...	157.1804057868...
3	SP	5998226.959999...	137.5046297739...
4	DF	355141.0800000...	161.1347912885...
5	PR	811156.3799999...	154.1536259977...
6	MT	187029.2900000...	195.2289039665...
7	MA	152523.0200000...	198.8566101694...
8	AL	96962.05999999...	227.0774238875...
9	MG	1872257.260000...	154.7064336473...
10	PE	324850.4400000...	187.9921527777...
11	SE	75246.25	208.4383656509...

Inference: here I used inner join of customer,order and payments along with sum and average of common states.

3. Calculate the Total & Average value of order freight for each state

ANS:

```
select a.customer_id,b.customer_state,
        round(sum(payment_value),2) as total_sum_for_state,
        round(avg(payment_value),2) as total_avg_for_state


FROM `business-case-1-413214.123.orders` as a
join `business-case-1-413214.123.customers` as b
on a.customer_id = b.customer_id
join `business-case-1-413214.123.payments` as c
on a.order_id = c.order_id
where a.order_status = 'shipped'
group by 1,2
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_id	customer_state	total_sum_for_state	total_avg_for_state
1	0735e7e4298a2ebbb4664934...	RN	196.14	196.14
2	285195a5b585842e25bd1ef90...	SP	87.53	87.53
3	79298f6a8720081178c7741e3...	SP	157.81	157.81
4	7cd85ff9a069e8822b0f0b328e...	MA	220.86	220.86
5	eec3d506a23070bb0ffad2fea9...	RJ	141.23	141.23
6	e8d32260f2ebace5f1b80c9b2...	RJ	654.41	654.41
7	cbdf66401d733d5f09fb411e49...	RJ	23.1	23.1
8	494fe6ed11aa9695f8fd1fddd3...	SP	93.56	93.56
9	c5d2a092d72c266cc8edae2b5...	BA	145.14	145.14
10	0026d252429f669d454d726e5...	SP	139.15	139.15
11	31cb868903a0743d096e5996...	SP	121.74	121.74

Inference: here I used sum and average of payment column with respect to customer state and customer id that is inner joined with customer and orders coulumn.

5.Analysis based on sales, freight and delivery time.

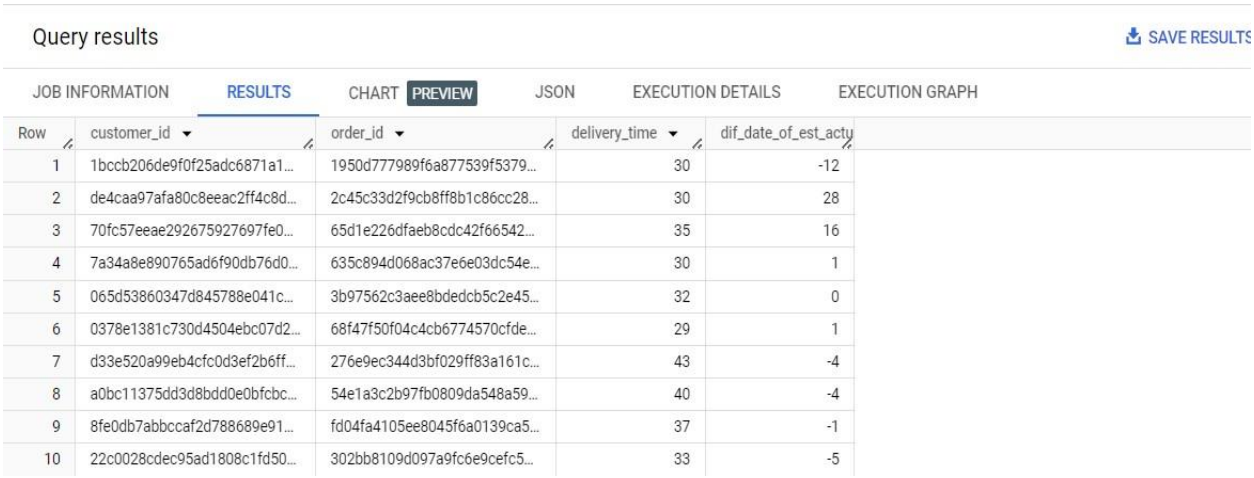
1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
   Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
   Do this in a single query.

   You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
   - **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
   - **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date

ANS:

```
select customer_id,order_id,
       date_diff(order_delivered_customer_date, order_purchase_timestamp, DAY) as
delivery_time,
       date_diff(order_estimated_delivery_date, order_delivered_customer_date, DAY )
AS dif_date_of_est_actual

FROM `business-case-1-413214.123.orders`
```



Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_id	order_id	delivery_time	dif_date_of_est_actu
1	1bccb206de9f0f25adc6871a1...	1950d777989f6a877539f5379...	30	-12
2	de4caa97afa80c8eeac2ff4c8d...	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	70fc57eeae292675927697fe0...	65d1e226dfaeb8cdc42f66542...	35	16
4	7a34a8e890765ad6f90db76d0...	635c894d068ac37e6e03dc54e...	30	1
5	065d53860347d845788e041c...	3b97562c3aee8bdedcb5c2e45...	32	0
6	0378e1381c730d4504ebc07d2...	68f47f50f04c4cb6774570cfde...	29	1
7	d33e520a99eb4cfc0d3ef2b6ff...	276e9ec344d3bf029ff83a161c...	43	-4
8	a0bc11375dd3d8bdd0e0bfcbc...	54e1a3c2b97fb0809da548a59...	40	-4
9	8fe0db7abbccaf2d788689e91...	fd04fa4105ee8045f6a0139ca5...	37	-1
10	22c0028cdec95ad1808c1fd50...	302bb8109d097a9fc6e9cefc5...	33	-5

Inference: I used date_diff() functions in order table column.

2. Find out the top 5 states with the highest & lowest average freight value.

```
select a.customer_state,
```

```
     avg(freight_value) as avg_of_freight
from `business-case-1-413214.123.customers` as a
join `business-case-1-413214.123.orders` as b
on a.customer_id = b.customer_id
join `business-case-1-413214.123.order_items` as c
on b.order_id = c.order_id
group by 1
order by 2 desc
limit 5
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	avg_of_freight
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

Code for least 5 avg

```
select a.customer_state,
     avg(freight_value) as avg_of_freight
from `business-case-1-413214.123.customers` as a
join `business-case-1-413214.123.orders` as b
on a.customer_id = b.customer_id
join `business-case-1-413214.123.order_items` as c
on b.order_id = c.order_id
group by 1
order by 2
limit 5
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECU

Row	customer_state	avg_of_freight
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

Inference: I used average of freight by joining of order, customer and orde_item. Both descending and ascending.

3. Find out the top 5 states with the highest & lowest average delivery time.

```
SELECT
  customer_state,
  MAX(Date_diff_delivery_dispatch) AS max_of_delivery,
  MIN(Date_diff_delivery_dispatch) AS min_of_delivery
FROM
  (
    SELECT customer_state,
      DATE_DIFF(order_delivered_customer_date, order_delivered_carrier_date, DAY)
AS Date_diff_delivery_dispatch
    FROM
      `business-case-1-413214.123.orders` AS a
    JOIN
      `business-case-1-413214.123.customers` AS b
    ON
      a.customer_id = b.customer_id
  ) AS cc
GROUP BY
  customer_state
ORDER BY
  max_of_delivery DESC,
  customer_state ASC

limit 5
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	customer_state	max_of_delivery
1	RJ	205
2	ES	195
3	SE	194
4	PI	190
5	PA	188

Lowest avg delivery time

```
SELECT
```

```
  customer_state,

  MIN(Date_diff_delivery_dispatch) AS min_of_delivery
FROM
  (
    SELECT customer_state,
      DATE_DIFF(order_delivered_customer_date, order_delivered_carrier_date, DAY) AS
Date_diff_delivery_dispatch
    FROM
      `business-case-1-413214.123.orders` AS a
    JOIN
      `business-case-1-413214.123.customers` AS b
    ON
      a.customer_id = b.customer_id
  ) AS cc
GROUP BY
  customer_state
ORDER BY 2
limit 5
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	min_of_delivery
1	MG	-16
2	SP	-7
3	RS	-1
4	SC	-1
5	PR	0

Inference: in this I used subquery method in which I find out the temporary tables of dates by date_diff() functions, then it is treated for main functions.

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
   You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

ANS:

```
select customer_state,
      date_diff(order_delivered_customer_date,order_estimated_delivery_date, day) as
top_5
```

```
from `business-case-1-413214.123.orders` as a
join `business-case-1-413214.123.customers` as b
on a.customer_id = b.customer_id
order by 2 desc
limit 5
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	top_5
1	RJ	188
2	ES	181
3	SP	175
4	SP	167
5	SE	166

Inference: I used date_diff() function after inner joining of orders and customers tables

6. Analysis based on the payments:

1. Find the month on month no. of orders placed using different payment types.

ANS:

```
SELECT

  c.payment_type,
  EXTRACT(MONTH FROM order_delivered_customer_date) AS month,
  count(payment_type)
FROM
  `business-case-1-413214.123.orders` AS a
JOIN `business-case-1-413214.123.payments` as c

ON
  a.order_id = c.order_id
group by 1,2
ORDER BY
   month
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	payment_type	month	f0_
12	UPI	2	1425
13	debit_card	2	79
14	credit_card	3	7086
15	voucher	3	535
16	UPI	3	1899
17	debit_card	3	90
18	credit_card	4	7588
19	voucher	4	523
20	UPI	4	1844
21	debit_card	4	134
22	credit_card	5	8552
23	voucher	5	662

Inference : extracted months and count of orders after inner joining of orders and payments.

2. Find the no. of orders placed on the basis of the payment installments that have been paid.

```
SELECT distinct payment_installments, count(e.order_id) as no_of_order

from `business-case-1-413214.123.payments` as d
join `business-case-1-413214.123.orders` as e

on d.order_id =e.order_id
group by 1
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	payment_installment	no_of_order
1	1	52546
2	7	1626
3	10	5328
4	6	3920
5	2	12413
6	4	7098
7	3	10461
8	8	4268
9	9	644
10	5	5239
11	12	133
12	20	17
13	15	74

Inference: here I used inner joining of order and payments table after that I selected distinct payment installment and corresponding orders.