TARGET – SQL ANALYSIS

By Bharath krishna

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table
    2. Time period for which the data is given
    3. Cities and States covered in the dataset

Sol:

1.1

Cutomers table

Field name	Type	Mode
customer_id	STRING	NULLABLE
customer_unique_id	STRING	NULLABLE
customer_zip_code_prefix	INTEGER	NULLABLE
customer_city	STRING	NULLABLE
customer_state	STRING	NULLABLE

```
SELECT *
FROM `dsml-target-business-case-1.target.customers` LIMIT 5
```

Press Alt+F1 for Accessibility Opti

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_id	customer_unique_id	customer_zip_c	customer_city	customer_state
1	0735e7e4298a2ebbb4664934...	fcb003b1bdc0df64b4d065d9b...	59650	acu	RN
2	903b3d86e3990db01619a4eb...	46824822b15da44e983b021d...	59650	acu	RN
3	38c97666e962d4fea7fd6a83e...	b6108acc674ae5c99e29adc10...	59650	acu	RN
4	77c2f46cf580f4874c9a5751c2...	402cce5c0509000eed9e77fec...	63430	ico	CE
5	4d3ef4cfffb8ad4767c199c36a...	6ba00666ab7eada5ceec279b2...	63430	ico	CE

Orders table

Field name	Type	Mode
order_id	STRING	NULLABLE
customer_id	STRING	NULLABLE
order_status	STRING	NULLABLE
order_purchase_timestamp	TIMESTAMP	NULLABLE
order_approved_at	TIMESTAMP	NULLABLE
order_delivered_carrier_date	TIMESTAMP	NULLABLE
order_delivered_customer_date	TIMESTAMP	NULLABLE
order_estimated_delivery_date	TIMESTAMP	NULLABLE

RUN SAVE SHARE SCHEDULE MORE Query completed.

```
SELECT *
FROM `dsml-target-business-case-1.target.orders` LIMIT 5
```

Press Alt+F1 for Accessibility Options

Query results SAVE RESULTS EXPLORE DATA

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

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at
1	7a4df5d8cff4090e541401a20a...	725e9c75605414b21fd8c8d5a...	created	2017-11-25 11:10:33 UTC	null
2	35de4050331c6c644cddc86f4...	4ee64f4bfc542546f422da0aeb...	created	2017-12-05 01:07:58 UTC	null
3	b5359909123fa03c50bdb0cfe...	438449d4af8980d107bf04571...	created	2017-12-05 01:07:52 UTC	null
4	dba5062fbda3af4fb6c33b1e04...	964a6df3d9bdf60fe3e7b8bb69...	created	2018-02-09 17:21:04 UTC	null
5	90ab3e7d52544ec7bc3363c82...	7d61b9f4f216052ba664f22e9c...	created	2017-11-06 13:12:34 UTC	null

Order_items table

Filter Enter property name or value

Field name	Type	Mode	Co
order_id	STRING	NULLABLE	
order_item_id	INTEGER	NULLABLE	
product_id	STRING	NULLABLE	
seller_id	STRING	NULLABLE	
shipping_limit_date	TIMESTAMP	NULLABLE	
price	FLOAT	NULLABLE	
freight_value	FLOAT	NULLABLE	

```
SELECT *
FROM `dsml-target-business-case-1.target.order_items` LIMIT 5
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

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

Row	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price
1	f09e36e258656850b92657ac5...	1	44d53f1240d6332232e4393c0...	b64d51f0435e884e8de603b16...	2018-07-09 13:31:36 UTC	
2	f9ccaff7267fd0cf076e795b1fa...	1	44d53f1240d6332232e4393c0...	b64d51f0435e884e8de603b16...	2018-08-14 14:04:44 UTC	
3	c79bdf061e22288609201ec60...	1	5304ff3fa35856a156e1170a60...	cf6f6bc4df3999b9c6440f124f...	2017-05-12 19:05:20 UTC	
4	37193e64eb9a46b7f3197762f...	1	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-28 01:30:49 UTC	
5	95d6357ffe41aa6d2998852a7...	1	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-12 19:15:14 UTC	

1.2.

Query

```
SELECT  min(order_purchase_timestamp) as start_time,
        max(order_purchase_timestamp) as end_time,
        date_diff(max(order_purchase_timestamp),min(order_purchase_timestamp),day) as no_of_days

FROM `dsml-target-business-case-1.target.orders`
```

Output

```
SELECT  min(order_purchase_timestamp) as start_time,
        max(order_purchase_timestamp) as end_time,
        date_diff(max(order_purchase_timestamp),min(order_purchase_timestamp),day) as no_of_days

FROM `dsml-target-business-case-1.target.orders`
```

Query results

SAVE RESULTS

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

Row	start_time	end_time	no_of_days
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

1.3

Query:

```
SELECT distinct
       customer_city,
       customer_state
FROM `dsml-target-business-case-1.target.customers` c
join `dsml-target-business-case-1.target.orders`  o
   on o.customer_id=c.customer_id
```

output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DI

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP

2.In-depth Exploration:

1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

Soln:

2.1

Query:

```
SELECT extract(month from date(o.order_purchase_timestamp)) as month,
       FORMAT_DATETIME("%B", DATETIME(o.order_purchase_timestamp)) as monthname,
       extract(year from o.order_purchase_timestamp) as year,
       count(*) as no_of_orders
 FROM `dsml-target-business-case-1.target.orders`  o
left join `dsml-target-business-case-1.target.order_items` oi
  on oi.order_id=o.order_id
group by month,monthname,year
order by year,month
```

sample output:

month	monthname	year	no_of_orders
9	September	2016	7
10	October	2016	379
12	December	2016	1
1	January	2017	966
2	February	2017	1998
3	March	2017	3041
4	April	2017	2697
5	May	2017	4176
6	June	2017	3611
7	July	2017	4576

2.2

Query:

```
select
duration,
count(*) as no_of_sales
from
(SELECT time(order_purchase_timestamp) as time,
      case
      when time(order_purchase_timestamp) between '04:00:00' and '06:00:00'
      then 'Dawn'
      when time(order_purchase_timestamp) between '06:00:00' and '12:00:00'
      then 'Morning'
      when time(order_purchase_timestamp) between '12:00:00' and '18:00:00'
      then 'Afternoon'
```

```
       else
      'Night'
      end as duration
FROM `dsml-target-business-case-1.target.orders` ) as x
group by duration
order by count(duration)
```

output:

Query results

JOB INFORMATION | RESULTS | JSON

Row	duration	no_of_sales
1	Dawn	394
2	Morning	22240
3	Afternoon	38365
4	Night	38442

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

1.Get month on month orders by region, states

2.How are customers distributed in Brazil

Query:

3.1

```
SELECT extract(month from order_purchase_timestamp) as month,
       FORMAT_DATETIME("%B", DATETIME(o.order_purchase_timestamp)) as monthname,
       extract(year from order_purchase_timestamp) as year,
       c.customer_state as state,
       count(order_id) as no_of_orders,
FROM `dsml-target-business-case-1.target.orders` o
join `dsml-target-business-case-1.target.customers` c
    on o.customer_id=c.customer_id
group by year,month,monthname,customer_state,c.customer_state
order by year,month
```

output:

Query results

SAVE RESULTS

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

Row	month	monthname	year	state	no_of_orders
1	9	September	2016	RR	1
2	9	September	2016	RS	1
3	9	September	2016	SP	2
4	10	October	2016	SP	113
5	10	October	2016	RS	24
6	10	October	2016	RJ	56
7	10	October	2016	MT	3
8	10	October	2016	GO	9
9	10	October	2016	MG	40
10	10	October	2016	CE	8
11	10	October	2016	SC	11

3.2

Query:

```
SELECT  customer_city,
        customer_state,
       count(distinct customer_id) as no_of_customers
FROM `dsml-target-business-case-1.target.customers`
group by customer_state,customer_city
```

sample output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUT

Row	customer_city	customer_state	no_of_customer
1	acu	RN	3
2	ico	CE	8
3	ipe	RS	2
4	ipu	CE	4
5	ita	SC	3
6	itu	SP	136
7	jau	SP	74
8	luz	MG	2
9	poa	SP	85
10	uba	MG	53
11	una	BA	5

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

4.1 Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

4.2 Mean & Sum of price and freight value by customer state

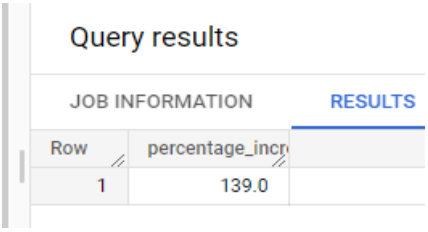
Soln:

Query:

4.1

```
select round((max(total_cost)-
min(total_cost))/min(total_cost),2) * 100 as percentage_increase
from
(select extract(year from order_date) as year,
       sum(cost) as total_cost
from
(SELECT o.order_id,
     date(o.order_purchase_timestamp) as order_date,
      oi.price,
       oi.freight_value,
       oi.price + oi.freight_value as cost
 FROM `dsml-target-business-case-1.target.order_items` as oi
 join `dsml-target-business-case-1.target.orders` as o
 on o.order_id=oi.order_id
where (date(o.order_purchase_timestamp) between '2017-01-01' and '2017-08-
31') or (date(o.order_purchase_timestamp) between '2018-01-01' and '2018-08-31')
order  by order_date) as x
group by year)
```

output:



4.2

Query:

```
SELECT c.customer_state,
     round(sum(oi.price),2) as total_price,
      round(avg(oi.price),2) as avg_price,
      round(sum(oi.freight_value),2) as total_freight_val,
      round(avg(oi.freight_value),2) as avg_freight_val,

  FROM `dsml-target-business-case-1.target.order_items` oi
  join `dsml-target-business-case-1.target.orders` o
      on o.order_id=oi.order_id
  join `dsml-target-business-case-1.target.customers` c
      on o.customer_id=c.customer_id
  group by c.customer_state
  limit 10
```

output:

Query results

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

Row	customer_state	total_price	avg_price	total_freight_val	avg_freight_val
1	MT	156453.53	148.3	29715.43	28.17
2	MA	119648.22	145.2	31523.77	38.26
3	AL	80314.81	180.89	15914.59	35.84
4	SP	5202955.05	109.65	718723.07	15.15
5	MG	1585308.03	120.75	270853.46	20.63
6	PE	262788.03	145.51	59449.66	32.92
7	RJ	1824092.67	125.12	305589.31	20.96
8	DF	302603.94	125.77	50625.5	21.04
9	RS	750304.02	120.34	135522.74	21.74
10	SE	58920.85	153.04	14111.47	36.65

5.Analysis on sales, freight and delivery time

5.1 Calculate days between purchasing, delivering and estimated delivery
5.2 Create columns
5.3 time_to_delivery = order_purchase_timestamp-order_delivered_customer_datediff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
5.4 Sort the data to get the following:
    5.4.1 Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
    5.4.2 Top 5 states with highest/lowest average time to delivery
    5.4.3 Top 5 states where delivery is really fast/ not so fast compared to estimated date

Soln:

5.1

Query:

```
SELECT
        date(order_purchase_timestamp) as purchase_date,
        date(order_delivered_customer_date) as delivered_date,
        date(order_estimated_delivery_date) as est_delivery_date,
        abs(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as  deli
very_days,
         abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as
  est_delivery_days,

 FROM `dsml-target-business-case-1.target.orders`
 where order_status='delivered'
 LIMIT 10
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECU

Row	purchase_date	delivered_date	est_delivery_dat	delivery_days	est_delivery_day
1	2017-04-14	2017-05-08	2017-05-18	23	9
2	2017-05-10	2017-05-23	2017-05-18	12	5
3	2017-04-22	2017-05-05	2017-05-18	12	12
4	2017-05-09	2017-05-16	2017-05-18	7	1
5	2017-04-26	2017-05-08	2017-05-18	12	9
6	2017-05-10	2017-05-12	2017-05-18	1	5
7	2017-05-10	2017-05-17	2017-05-18	6	0
8	2017-04-18	2017-05-10	2017-05-18	21	7
9	2017-05-10	2017-05-18	2017-05-18	7	0
10	2017-04-15	2017-05-16	2017-05-18	30	1

5.2

Query:

```
SELECT
         order_purchase_timestamp,
        order_delivered_customer_date,
        order_estimated_delivery_date,
        timestamp_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_
to_delivery,
       abs(date_diff(date(order_delivered_customer_date),date(order_estimated_delivery_date
),day)) as diff_estimated_delivery
FROM `dsml-target-business-case-1.target.orders`
where  order_status='delivered'
LIMIT 10
```

Output:

Query results

SAVE RESULTS | EXPLOR

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

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_c
1	2017-04-14 22:06:32 UTC	2017-05-08 11:10:26 UTC	2017-05-18 00:00:00 UTC	23	10
2	2017-05-10 14:03:27 UTC	2017-05-23 13:12:27 UTC	2017-05-18 00:00:00 UTC	12	5
3	2017-04-22 15:50:30 UTC	2017-05-05 13:27:50 UTC	2017-05-18 00:00:00 UTC	12	13
4	2017-05-09 17:42:45 UTC	2017-05-16 23:22:20 UTC	2017-05-18 00:00:00 UTC	7	2
5	2017-04-26 01:01:39 UTC	2017-05-08 08:54:36 UTC	2017-05-18 00:00:00 UTC	12	10
6	2017-05-10 20:47:02 UTC	2017-05-12 17:00:05 UTC	2017-05-18 00:00:00 UTC	1	6
7	2017-05-10 15:34:59 UTC	2017-05-17 11:14:40 UTC	2017-05-18 00:00:00 UTC	6	1
8	2017-04-18 21:20:40 UTC	2017-05-10 11:50:00 UTC	2017-05-18 00:00:00 UTC	21	8
9	2017-05-10 22:02:40 UTC	2017-05-18 17:09:46 UTC	2017-05-18 00:00:00 UTC	7	0
10	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	2
11	2017-04-22 13:55:16 UTC	2017-05-12 13:55:55 UTC	2017-05-18 00:00:00 UTC	20	6

5.3

Query:

```
select z.customer_state,
       round(avg(z.time_to_delivery),2) as avg_time_to_delivery,
```

```
       round(avg(z.diff_estimated_delivery),2) as avg_est_del,
       round(avg(z.freight_value),2) as freight_avg
from
(select y.order_id,
       y.time_to_delivery,
       y.diff_estimated_delivery,
       oi.freight_value,
       c.customer_state
from
(select  x.order_id,
        x.customer_id,
        x.purchase_date,
        x.delivered_date,
        x.est_delivery_date,
        abs(date_diff(x.delivered_date,x.purchase_date,day)) as time_to_delivery,
        abs(date_diff(x.delivered_date,x.est_delivery_date,day)) as diff_estimated_delivery
from
(SELECT order_id,
       customer_id,
       extract(date from order_purchase_timestamp) as purchase_date,
       extract(date from order_delivered_customer_date) as delivered_date,
       extract(date from order_estimated_delivery_date) as est_delivery_date
 FROM `dsml-target-business-case-1.target.orders`
 where order_status='delivered') as x) as y
 left join `dsml-target-business-case-1.target.order_items` as oi
    on y.order_id=oi.order_id
  left join `dsml-target-business-case-1.target.customers` c
   on y.customer_id=c.customer_id) as z
  group by z.customer_state
```

output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	customer_state	avg_time_to_del	avg_est_del	freight_avg
1	GO	15.34	13.82	22.56
2	SP	8.66	11.93	15.12
3	RS	15.13	15.39	21.61
4	BA	19.19	13.79	26.49
5	MG	11.92	14.08	20.63
6	MT	17.91	15.82	28.0
7	RJ	15.07	15.09	20.91
8	SC	14.95	13.02	21.51
9	SE	21.42	14.74	36.57
10	PE	18.22	15.59	32.69
11	TO	17.4	13.61	37.44

5.4.1

Query:

```
select z.customer_state,
       round(avg(z.freight_value),2) as freight_avg
from
(select y.order_id,
       y.time_to_delivery,
       y.diff_estimated_delivery,
       oi.freight_value,
       c.customer_state
from
(select  x.order_id,
        x.customer_id,
        x.purchase_date,
        x.delivered_date,
        x.est_delivery_date,
        abs(date_diff(x.delivered_date,x.purchase_date,day)) as time_to_delivery,
        abs(date_diff(x.delivered_date,x.est_delivery_date,day)) as diff_estimated_delivery
from
(SELECT order_id,
       customer_id,
       extract(date from order_purchase_timestamp) as purchase_date,
       extract(date from order_delivered_customer_date) as delivered_date,
       extract(date from order_estimated_delivery_date) as est_delivery_date
 FROM `dsml-target-business-case-1.target.orders`
 where order_status='delivered') as x) as y
 left join `dsml-target-business-case-1.target.order_items` as oi
    on y.order_id=oi.order_id
  left join `dsml-target-business-case-1.target.customers` c
   on y.customer_id=c.customer_id) as z
  group by z.customer_state
 order by freight_avg
 limit 5
```

output:

Query results

JOB INFORMATION | RESULTS | JSON | EXE

Row	customer_state	freight_avg
1	SP	15.12
2	PR	20.47
3	MG	20.63
4	RJ	20.91
5	DF	21.07

5.4.2

Query:

```
select z.customer_state,
       round(avg(z.time_to_delivery),2) as avg_time_to_delivery
from
(select y.order_id,
       y.time_to_delivery,
       y.diff_estimated_delivery,
       oi.freight_value,
       c.customer_state
from
(select  x.order_id,
        x.customer_id,
        x.purchase_date,
        x.delivered_date,
        x.est_delivery_date,
        abs(date_diff(x.delivered_date,x.purchase_date,day)) as time_to_delivery,
        abs(date_diff(x.delivered_date,x.est_delivery_date,day)) as diff_estimated_delivery
from
(SELECT order_id,
       customer_id,
       extract(date from order_purchase_timestamp) as purchase_date,
       extract(date from order_delivered_customer_date) as delivered_date,
       extract(date from order_estimated_delivery_date) as est_delivery_date
 FROM `dsml-target-business-case-1.target.orders`
 where order_status='delivered') as x) as y
 left join `dsml-target-business-case-1.target.order_items` as oi
    on y.order_id=oi.order_id
  left join `dsml-target-business-case-1.target.customers` c
   on y.customer_id=c.customer_id) as z
  group by z.customer_state
  order by avg_time_to_delivery desc
  limit 5
```

output:

Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	avg_time_to_del
1	AP	28.22
2	RR	28.17
3	AM	26.34
4	AL	24.45
5	PA	23.7

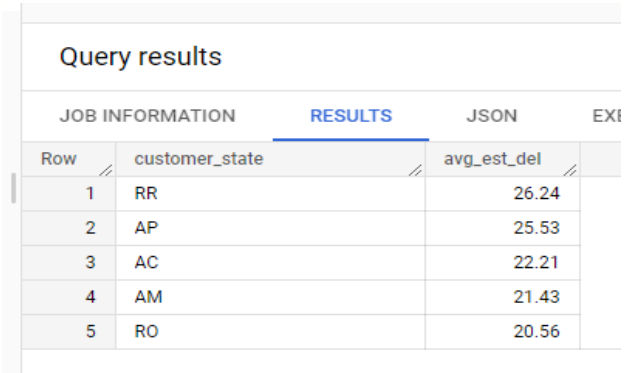
5.4.3

Query:

```
select z.customer_state,
       round(avg(z.diff_estimated_delivery),2) as avg_est_del
from
(select y.order_id,
       y.time_to_delivery,
       y.diff_estimated_delivery,
       oi.freight_value,
       c.customer_state

from
(select  x.order_id,
        x.customer_id,
        x.purchase_date,
        x.delivered_date,
        x.est_delivery_date,
        abs(date_diff(x.delivered_date,x.purchase_date,day)) as time_to_delivery,
        abs(date_diff(x.delivered_date,x.est_delivery_date,day)) as diff_estimated_delivery
from
(SELECT order_id,
       customer_id,
       extract(date from order_purchase_timestamp) as purchase_date,
       extract(date from order_delivered_customer_date) as delivered_date,
       extract(date from order_estimated_delivery_date) as est_delivery_date
 FROM `dsml-target-business-case-1.target.orders`
 where order_status='delivered') as x) as y
 left join `dsml-target-business-case-1.target.order_items` as oi
    on y.order_id=oi.order_id
  left join `dsml-target-business-case-1.target.customers` c
   on y.customer_id=c.customer_id) as z
  group by z.customer_state
  order by avg_est_del desc
```

output:



Query results

JOB INFORMATION | RESULTS | JSON | EXE

Row	customer_state	avg_est_del
1	RR	26.24
2	AP	25.53
3	AC	22.21
4	AM	21.43
5	RO	20.56

6.Payment type analysis:

6.1 Month over Month count of orders for different payment types
6.2 Distribution of payment installments and count of orders

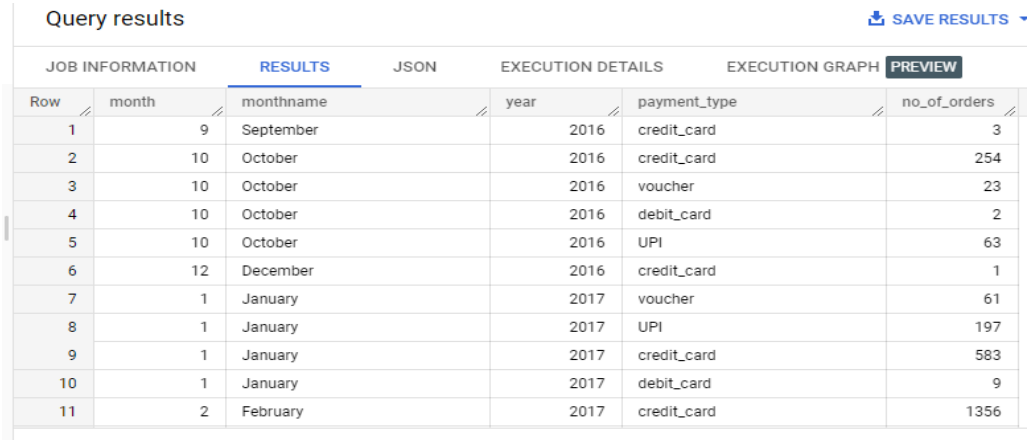
Soln:

6.1

Query:

```
SELECT extract(month from o.order_purchase_timestamp) as month,
      format_date("%B",datetime(o.order_purchase_timestamp)) as monthname,
       extract(year from o.order_purchase_timestamp) as year,
       p.payment_type,
        count(p.order_id) as no_of_orders
FROM `dsml-target-business-case-1.target.payments`  p
 join `dsml-target-business-case-1.target.orders`  o
 on o.order_id=p.order_id
 group by monthname,month,p.payment_type,year
 order by year,month
```

output:



Query results

SAVE RESULTS

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

Row	month	monthname	year	payment_type	no_of_orders
1	9	September	2016	credit_card	3
2	10	October	2016	credit_card	254
3	10	October	2016	voucher	23
4	10	October	2016	debit_card	2
5	10	October	2016	UPI	63
6	12	December	2016	credit_card	1
7	1	January	2017	voucher	61
8	1	January	2017	UPI	197
9	1	January	2017	credit_card	583
10	1	January	2017	debit_card	9
11	2	February	2017	credit_card	1356

6.2

Query:

```
SELECT
       payment_installments,
       count(*) as no_of_orders
 FROM `dsml-target-business-case-1.target.payments`
 group by payment_installments
LIMIT 1000
```

Output:

Query results

JOB INFORMATION | RESULTS

Row	payment_installr	no_of_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328