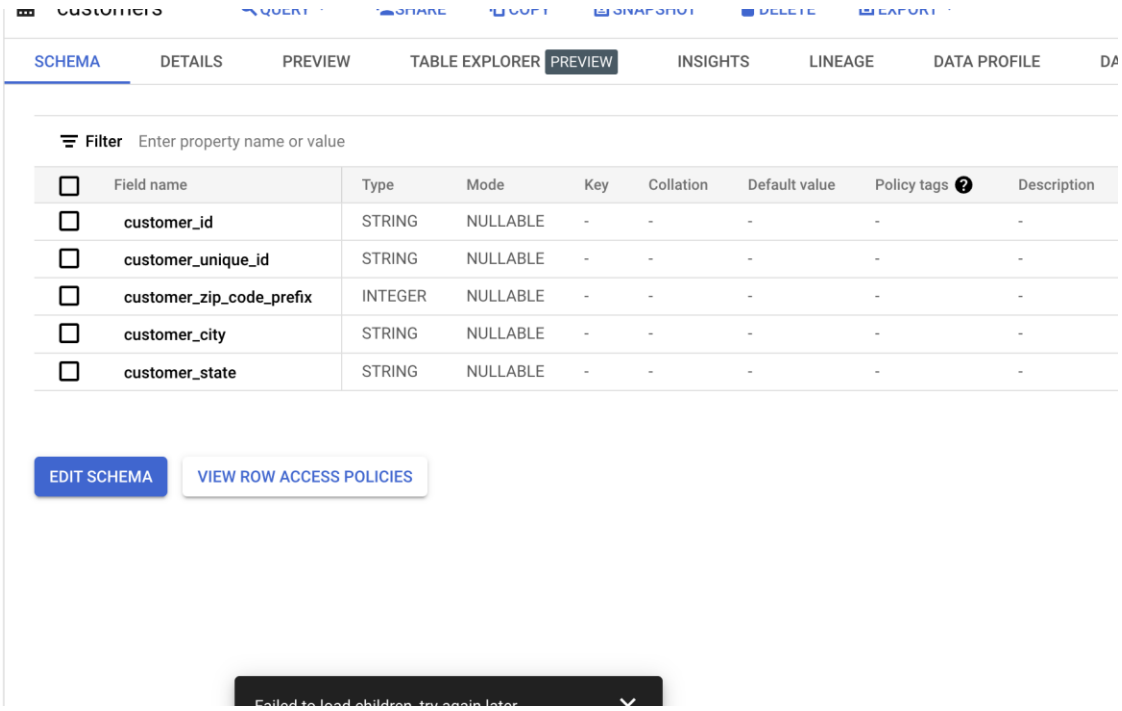
Business case study

Question-1 Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

Part A- Data type of all columns in the "customers" table.



Field name	Type	Mode	Key	Collation	Default value	Policy tags	Description
customer_id	STRING	NULLABLE	-	-	-	-	-
customer_unique_id	STRING	NULLABLE	-	-	-	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-	-
customer_city	STRING	NULLABLE	-	-	-	-	-
customer_state	STRING	NULLABLE	-	-	-	-	-

Part B - Get the time range between which the orders were placed.

Solution--
```
SELECT
MIN(order_purchase_timestamp) AS min,
MAX(order_purchase_timestamp) AS max
FROM `Target.orders`
```

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE OPEN IN

```
SELECT
    MIN(order_purchase_timestamp) AS min,
    MAX(order_purchase_timestamp) AS max
FROM `Target.orders`
```

Query results

SAVE RESULTS

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

Row	min	max
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Part C - Count the Cities & States of customers who ordered during the given period.

Solution-

```
SELECT
COUNT(DISTINCT customer_city) AS num_cities,
COUNT(DISTINCT customer_state) AS num_states
FROM `Target.customers`
```

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE OPEN IN

```

SELECT
    COUNT(DISTINCT customer_city) AS num_cities,
    COUNT(DISTINCT customer_state) AS num_states
FROM `Target.customers`
```

Query results

SAVE RESULTS

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

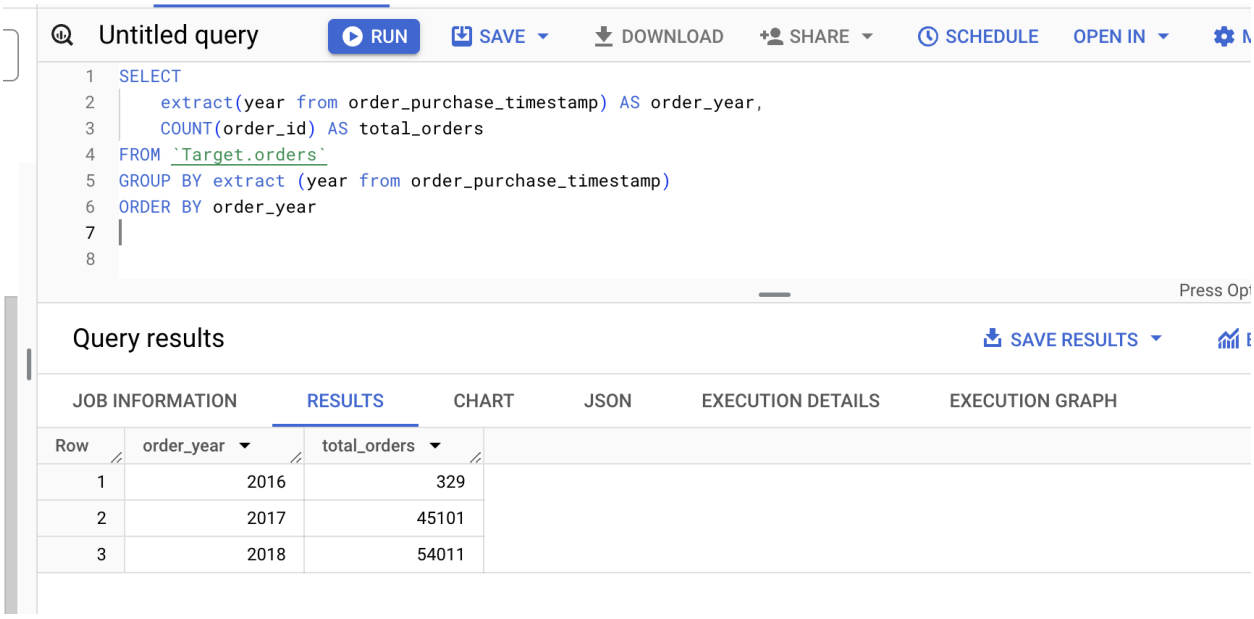
Row	num_cities	num_states
1	4119	27

Question 2 - In-depth Exploration.

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

Solution- SELECT

```
extract(year from order_purchase_timestamp) AS order_year,
COUNT(order_id) AS total_orders
FROM `Target.orders`
GROUP BY extract (year from order_purchase_timestamp)
ORDER BY order_year
```



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

Solution - SELECT

```
extract(YEAR from order_purchase_timestamp) AS order_year,
extract(month from order_purchase_timestamp) AS order_month,
COUNT(order_id) AS total_orders
FROM
`Target.orders`
GROUP BY
extract(YEAR from order_purchase_timestamp),
extract(month from order_purchase_timestamp)
ORDER BY
order_year,
order_month;
```

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE OPEN IN

```
SELECT
    extract(YEAR from order_purchase_timestamp) AS order_year,
```

Query results

SAVE RESULTS

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

Row	order_year	order_month	total_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Part C -- During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

. 0-6 hrs : Dawn

. 7-12 hrs : Mornings

. 13-18 hrs : Afternoon

. 19-23 hrs : Night

Solution-

```
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'
ELSE 'Night'
END AS time_of_day,
COUNT(order_id) AS total_orders
FROM
`Target.orders`
GROUP BY
time_of_day
ORDER BY
total_orders DESC
```

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE OPEN IN

```
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'
        ELSE 'Night'
    END AS time_of_day,
    COUNT(order_id) AS total_orders
FROM
    `Target.orders`
GROUP BY
```

Press Opt

Query results

SAVE RESULTS

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

Row	time_of_day	total_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

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

Part A -- Get the month-on-month no. of orders placed in each state.

Solution -

```
SELECT
customer_state,
EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
COUNT(order_id) AS total_orders
FROM
`Target.orders` as orders
JOIN
`Target.customers` as customers ON orders.customer_id = customers.customer_id
GROUP BY
customer_state,
order_year,
order_month
ORDER BY
customer_state, order_year, order_month;
```

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE OPEN IN MORE

```
SELECT
    customer_state,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
    COUNT(order_id) AS total_orders
FROM
    `Target.orders` as orders
```

Query results

SAVE RESULTS

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

Row	customer_state	order_year	order_month	total_orders
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6

Part B - How are the customers distributed across all the states?

Solution -

```
SELECT
customer_state,
COUNT(customer_id) AS total_customers
FROM
`Target.customers`
GROUP BY
customer_state
ORDER BY
total_customers DESC;
```

Untitled query RUN SAVE DOWNLOAD SHARE SCHEDULE OPEN IN MORE

```
SELECT
    customer_state,
    COUNT(customer_id) AS total_customers
FROM
    `Target.customers`
GROUP BY
    customer_state
ORDER BY
```

Query results SAVE RESULTS

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

Row	customer_state	total_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

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

Part A - Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Solution-
```
SELECT
extract (YEAR from o.order_purchase_timestamp) AS order_year,
extract (MONTH from o.order_purchase_timestamp) AS order_month,
SUM(p.payment_value) AS total_order_value
FROM
`Target.orders` o
JOIN
`Target.payments` p ON o.order_id = p.order_id
WHERE
extract (YEAR from o.order_purchase_timestamp) IN (2017, 2018)
AND extract (MONTH from o.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY
order_year,
order_month
ORDER BY
order_year, order_month;
```

```google_sql
SELECT
    extract (YEAR from o.order_purchase_timestamp) AS order_year,
    extract (MONTH from o.order_purchase_timestamp) AS order_month,
    SUM(p.payment_value) AS total_order_value
FROM
    `Target.orders` o
JOIN
    `Target.payments` p ON o.order_id = p.order_id
```

Query results

SAVE RESULTS

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

Row	order_year	order_month	total_order_value
1	2017	1	138488.0399999...
2	2017	2	291908.0099999...
3	2017	3	449863.6000000...
4	2017	4	417788.0300000...
5	2017	5	592918.8200000...
6	2017	6	511276.3800000...
7	2017	7	592382.9200000...
8	2017	8	674396.3200000...
9	2018	1	1115004.180000...
10	2018	2	992463.3400000...

Part B -- Calculate the Total & Average value of order price for each state.

Solution -

```google_sql
SELECT
c.customer_state,
AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS
avg_delivery_time,
AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY))
AS avg_estimated_diff
FROM
`Target.orders` AS o
JOIN
`Target.customers` AS c ON o.customer_id = c.customer_id
WHERE
o.order_delivered_customer_date IS NOT NULL
AND o.order_estimated_delivery_date IS NOT NULL
GROUP BY
c.customer_state
ORDER BY
avg_delivery_time DESC;
```

```sql
SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS avg_delivery_time,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS avg_estimated_diff
FROM
    `Target.orders` AS o
JOIN
    `Target.customers` AS c ON o.customer_id = c.customer_id
```

Query results

SAVE RESULTS

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

Row	customer_state	avg_delivery_time	avg_estimated_diff
1	RR	28.97560975609...	-16.4146341463...
2	AP	26.73134328358...	-18.7313432835...
3	AM	25.98620689655...	-18.6068965517...
4	AL	24.04030226700...	-7.94710327455...
5	PA	23.31606765327...	-13.1902748414...
6	MA	21.11715481171...	-8.76847977684...
7	SE	21.02985074626...	-9.17313432835...
8	CE	20.817826426896	-9.95777951524...
9	AC	20.63750000000...	-19.7625000000...
10	PB	19.95357833655...	-12.3713733075...

Part C - Calculate the Total & Average value of order freight for each state.

Solution -

```sql
SELECT
c.customer_state,
SUM(oi.freight_value) AS total_freight_value,
AVG(oi.freight_value) AS avg_freight_value
FROM
`Target.orders` AS o
JOIN
`Target.customers` AS c ON o.customer_id = c.customer_id
JOIN
`Target.order_items` AS oi ON o.order_id = oi.order_id
GROUP BY
c.customer_state
ORDER BY
total_freight_value DESC;
```

```sql
SELECT
    c.customer_state,
    SUM(oi.freight_value) AS total_freight_value,
    AVG(oi.freight_value) AS avg_freight_value
FROM
    `Target.orders` AS o
JOIN
    `Target.customers` AS c ON o.customer_id = c.customer_id
```

Query results

SAVE RESULTS

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

Row	customer_state	total_freight_value	avg_freight_value
1	SP	718723.0699999...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	MG	270853.4600000...	20.63016680630...
4	RS	135522.7400000...	21.73580433039...
5	PR	117851.6800000...	20.53165156794...
6	BA	100156.6799999...	26.36395893656...
7	SC	89660.26000000...	21.47036877394...
8	PE	59449.65999999...	32.91786267995...
9	GO	53114.97999999...	22.76681525932...
10	DF	50625.49999999...	21.04135494596...

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

Part A-- 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.

Solution --
```sql
SELECT
c.customer_state,
AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS
avg_delivery_time,
```

```
AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY))
AS avg_estimated_diff
FROM
`Target.orders` AS o
JOIN
`Target.customers` AS c ON o.customer_id = c.customer_id
WHERE
o.order_delivered_customer_date IS NOT NULL
AND o.order_estimated_delivery_date IS NOT NULL
GROUP BY
c.customer_state
ORDER BY
avg_delivery_time DESC;
```

```
SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS avg_delivery_time,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS avg_estimated_diff
FROM
    `Target.orders` AS o
JOIN
    `Target.customers` AS c ON o.customer_id = c.customer_id
```

Pr

Query results

SAVE RESULTS

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

Row	customer_state	avg_delivery_time	avg_estimated_diff
1	RR	28.97560975609...	-16.4146341463...
2	AP	26.73134328358...	-18.7313432835...
3	AM	25.98620689655...	-18.6068965517...
4	AL	24.04030226700...	-7.94710327455...
5	PA	23.31606765327...	-13.1902748414...
6	MA	21.11715481171...	-8.76847977684...
7	SE	21.02985074626...	-9.17313432835...
8	CE	20.817826426896	-9.95777951524...
9	AC	20.63750000000...	-19.7625000000...
10	PB	19.95357833655...	-12.3713733075...

Part B - Find out the top 5 states with the highest & lowest average freight value.

Solution-

```
SELECT
c.customer_state,
AVG(oi.freight_value) AS avg_freight_value
FROM
`Target.orders` AS o
JOIN
`Target.customers` AS c ON o.customer_id = c.customer_id
JOIN
`Target.order_items` AS oi ON o.order_id = oi.order_id
GROUP BY
```

```
c.customer_state
ORDER BY
avg_freight_value DESC
LIMIT 5;
```

```
SELECT
    c.customer_state,
    AVG(oi.freight_value) AS avg_freight_value
FROM
    `Target.orders` AS o
JOIN
    `Target.customers` AS c ON o.customer_id = c.customer_id
JOIN
    `Target.order_items` AS oi ON o.order_id = oi.order_id
GROUP BY
    c.customer_state
ORDER BY
    avg_freight_value DESC
LIMIT 5;
```

Pr

Query results

SAVE RESULTS

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

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

Part C- Find out the top 5 states with the highest & lowest average delivery time.

Solution-

```
SELECT
c.customer_state,
AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS
avg_delivery_time
FROM
`Target.orders` AS o
JOIN
`Target.customers` AS c ON o.customer_id = c.customer_id
WHERE
o.order_delivered_customer_date IS NOT NULL
GROUP BY
c.customer_state
ORDER BY
avg_delivery_time DESC
LIMIT 5;
```

```sql
SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS avg_delivery_time
FROM
    `Target.orders` AS o
JOIN
    `Target.customers` AS c ON o.customer_id = c.customer_id
WHERE
    o.order_delivered_customer_date IS NOT NULL
GROUP BY
    c.customer_state
ORDER BY
    avg_delivery_time DESC
LIMIT 5;
```

Query results

SAVE RESULTS

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

Row	customer_state	avg_delivery_time
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...

Part - D 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.

Solution-

```sql
SELECT
c.customer_state,
AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY))
AS avg_delivery_ahead
FROM
`Target.orders` AS o
JOIN
`Target.customers` AS c ON o.customer_id = c.customer_id
WHERE
o.order_delivered_customer_date IS NOT NULL
AND o.order_estimated_delivery_date IS NOT NULL
GROUP BY
c.customer_state
ORDER BY
avg_delivery_ahead DESC
LIMIT 5;
```

```sql
SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY)) AS avg_delivery_ahead
FROM
    `Target.orders` AS o
JOIN
    `Target.customers` AS c ON o.customer_id = c.customer_id
WHERE
    o.order_delivered_customer_date IS NOT NULL
    AND o.order_estimated_delivery_date IS NOT NULL
GROUP BY
    c.customer_state
ORDER BY
    avg_delivery_ahead DESC
LIMIT 5;
```

Pr

Query results

SAVE RESULTS

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

Row	customer_state	avg_delivery_ahead
1	AC	19.76250000000...
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

Question – 6 Analysis based on the payments:

Part – A Find the month-on-month no. of orders placed using different payment types.

Solution -

```sql
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
p.payment_type,
COUNT(o.order_id) AS total_orders
FROM
`Target.orders` AS o
JOIN
`Target.payments` AS p ON o.order_id = p.order_id
GROUP BY
order_year,
order_month,
p.payment_type
ORDER BY
order_year,
order_month,
total_orders DESC;
```

```sql
SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    p.payment_type,
    COUNT(o.order_id) AS total_orders
FROM
    `Target.orders` AS o
JOIN
```

Pr

Query results

SAVE RESULTS

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

Row	order_year	order_month	payment_type	total_orders
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9

Part – B Find the no. of orders placed on the basis of the payment installments that have been paid.

Solution-
```sql
SELECT
p.payment_installments,
COUNT(p.order_id) AS total_orders
FROM
`Target.payments` AS p
GROUP BY
p.payment_installments
ORDER BY
total_orders DESC
```

```sql
SELECT
    p.payment_installments,
    COUNT(p.order_id) AS total_orders
FROM
    `Target.payments` AS p
GROUP BY
    p.payment_installments
ORDER BY
```

Pr

Query results

SAVE RESULTS

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

Row	payment_installment	total_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644