

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

```
select *  
from `Target.customers`  
limit 10
```

Row	customer_id	customer_unique_id	customer_zip_code	customer_city	customer_state
1	0735e7e4298a2ebbb46649346...	fc003b1bdc0df64b4d065d9b...	59650	acu	RN
2	903b3d86e3990db01619a4ebe...	46824822b15da44e983b021d...	59650	acu	RN
3	38c97666e962d4fea7fd6a83e...	b6108acc674ae5c99e29adc10...	59650	acu	RN
4	77c2f46cf580f4874c9a5751c2...	402cce5c0509000eed9e77fec...	63430	ico	CE
5	4d3ef4cfff8ad4767c199c36a...	6ba00666ab7eada5ceec279b2...	63430	ico	CE
6	3000841b8e1f9e9493b52324...	796a0b1a21f597704057184a1...	63430	ico	CE
7	3c325415ccc7e622c66dec4bc...	05d1d2d9f0161c5f397ce7fc77...	63430	ico	CE
8	04f3a7b250e3be964f01bf22bc...	c34585a0276ecc5e4fb03de75...	63430	ico	CE
9	894202b8ef01f4719a4691e79...	01a4fe5fc00bbdb0b0a4af5a53...	63430	ico	CE
10	9d715b9fb75a9d081c14126c0...	8f399f3b7ace8e6245422c9e1f...	63430	ico	CE

B) Get the time range between which the orders were placed.

```
SELECT MIN(order_purchase_timestamp) AS first_order_date,  
MAX(order_purchase_timestamp) AS last_order_date  
FROM `Target.orders`
```

Query results				
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	first_order_date	last_order_date		
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC		

C) Count the number of Cities and States in our dataset.

```
select  
count(distinct geolocation_city ) as no_of_cities,  
count(distinct geolocation_state) as no_of_states  
from `Target.geolocation`
```

Query results			
JOB INFORMATION		RESULTS	EXECUTION DETAILS
Row	no_of_cities	no_of_states	
1	8011	27	

2. A) 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,  
EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,  
COUNT(*) AS order_count  
FROM  
  `Target.orders`  
GROUP BY  
  order_year, order_month  
ORDER BY  
  order_year, order_month  
limit 20
```

Query results				
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	order_year	order_month	order_count	
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	

B. 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 order_count  
FROM  
  `Target.orders`  
GROUP BY  
  order_month  
order by order_count desc  
limit 5
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION
Row	order_month	order_count		
1	8	10843		
2	5	10573		
3	7	10318		
4	3	9893		
5	6	9412		

C. 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

```
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 "Evening"
end as time_of_the_day,
count(*) as order_count
from `Target.orders` as o
left join `Target.customers` as cu
on o.customer_id=cu.customer_id
where cu.customer_city like 'bras%'
group by time_of_the_day
order by order_count asc
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	time_of_the_day	order_count		
1	Dawn	99		
2	Morning	614		
3	Evening	615		
4	Afternoon	822		

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

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

```
select distinct(cu.customer_state),
extract(month from order_purchase_timestamp) as months,
count(cu.customer_state) as order_count
from `Target.orders` as o
left join `Target.customers` as cu
on o.customer_id=cu.customer_id
where cu.customer_city like 'bras%'
group by cu.customer_state,months
order by order_count desc,months asc
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	customer_state ▼	months ▼	order_count ▼	
1	DF	7	241	
2	DF	8	231	
3	DF	6	217	
4	DF	5	208	
5	DF	3	204	
6	DF	2	196	
7	DF	4	183	
8	DF	11	168	
9	DF	1	151	
10	DF	12	131	

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

```
select
customer_state,
count(distinct customer_id) as unique_customer
from `Target.customers`
group by customer_state
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION
Row	customer_state	unique_customer		
1	RN	485		
2	CE	1336		
3	RS	5466		
4	SC	3637		
5	SP	41746		
6	MG	11635		
7	BA	3380		
8	RJ	12852		
9	GO	2020		
10	MA	747		

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

```
select
(sum(case when extract (year from order_purchase_timestamp)=2018 then
p.payment_value end)-sum(case when extract (year from
order_purchase_timestamp)=2017 then p.payment_value end))/ sum(case when extract
(year from order_purchase_timestamp)=2017 then p.payment_value end) * 100 as
percentage_increase
from `Target.orders` as o
left join `Target.payments` as p
on o.order_id=p.order_id
where
extract (year from order_purchase_timestamp) in (2017,2018)
and extract(month from order_purchase_timestamp) between 1 and 8
```

Query results		
JOB INFORMATION		RESULTS
Row	percentage_increase	
1	136.9768716466...	

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

```
select
se.seller_state,
round(sum(price),2) as tot_price,
round(avg(price),2) as avg_price
from `Target.order_items` as o
left join `Target.sellers` as se
on o.seller_id=se.seller_id
group by se.seller_state
```

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	seller_state	tot_price	avg_price	
1	SP	8753396.21	108.95	
2	MG	1011564.74	114.6	
3	PR	1261887.21	145.53	
4	SC	632426.07	155.2	
5	RS	378559.54	172.15	
6	DF	97749.48	108.73	
7	ES	47689.61	128.2	
8	RJ	843984.22	175.17	
9	GO	66399.21	127.69	
10	PA	1238.0	154.75	

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

```
SELECT
  s.seller_state,
  round(SUM(freight_value),2) AS total_freight,
  round(AVG(freight_value),2) AS average_freight
FROM
  `Target.order_items` as oi
  left join `Target.sellers` as s
  on oi.seller_id=s.seller_id
GROUP BY
  s.seller_state
```

Query results				
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	seller_state	total_freight	average_freight	
1	SP	1482487.67	18.45	
2	MG	212595.06	24.08	
3	PR	197013.52	22.72	
4	SC	106547.06	26.15	
5	RS	57243.09	26.03	
6	DF	18494.06	20.57	
7	ES	12171.13	32.72	
8	RJ	93829.9	19.47	
9	GO	12565.5	24.16	
10	PA	155.11	19.39	

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

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


```

select
order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
date_diff( order_estimated_delivery_date, order_delivered_customer_date, day) as
diff_estimated_delivery
from `Target.orders`

```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION TIME
Row	order_id	time_to_deliver	diff_estimated_delivery		
1	1950d777989f6a877539f5379...	30	-12		
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28		
3	65d1e226dfaeb8cdc42f66542...	35	16		
4	635c894d068ac37e6e03dc54e...	30	1		
5	3b97562c3aee8bdedcb5c2e45...	32	0		
6	68f47f50f04c4cb6774570cfde...	29	1		
7	276e9ec344d3bf029ff83a161c...	43	-4		
8	54e1a3c2b97fb0809da548a59...	40	-4		
9	fd04fa4105ee8045f6a0139ca5...	37	-1		
10	302bb8109d097a9fc6e9cefc5...	33	-5		

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

```

(SELECT

```

```

s.seller_state,
round(AVG(freight_value),2) AS average_freight
FROM
`Target.order_items` as oi
left join `Target.sellers` as s
on oi.seller_id=s.seller_id
GROUP BY
s.seller_state
order by average_freight desc
limit 5)
union all
(SELECT
s.seller_state,
round(AVG(freight_value),2) AS average_freight
FROM
`Target.order_items` as oi
left join `Target.sellers` as s
on oi.seller_id=s.seller_id
GROUP BY
s.seller_state
order by average_freight asc
limit 5)
order by average_freight asc

```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTED
Row	seller_state ▼	average_freight ▼		
1	SP	18.45		
2	PA	19.39		
3	RJ	19.47		
4	DF	20.57		
5	PR	22.72		
6	AC	32.84		
7	PI	36.94		
8	PB	39.19		
9	CE	46.38		
10	RO	50.91		

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

```

(SELECT
  s.customer_state,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
  time_to_deliver,
FROM
  `Target.orders` as oi
  left join `Target.customers` as s
  on oi.customer_id=s.customer_id
  order by time_to_deliver desc
  limit 5)
union all
(SELECT
  s.customer_state,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
  time_to_deliver,
FROM
  `Target.orders` as oi
  left join `Target.customers` as s
  on oi.customer_id=s.customer_id
  order by time_to_deliver asc
  limit 5)
  order by time_to_deliver asc

```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUT
Row	customer_state	time_to_deliver		
1	RJ	null		
2	RS	null		
3	SP	null		
4	DF	null		
5	PR	null		
6	PI	194		
7	SE	194		
8	PA	195		
9	RJ	208		
10	ES	209		

D. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

```

SELECT cu.customer_state,
round(AVG(date_diff(order_delivered_customer_date,order_estimated_delivery_date,
DAY)),2) AS delivery_days_difference
FROM `Target.orders` as o
left join `Target.customers` as cu
on o.customer_id=cu.customer_id
WHERE order_delivered_customer_date IS NOT NULL
GROUP BY cu.customer_state
HAVING delivery_days_difference < 0
ORDER BY delivery_days_difference
LIMIT 5;

```

Query results			
JOB INFORMATION		RESULTS	JSON
Row	customer_state	delivery_days_difference	
1	AC	-19.76	
2	RO	-19.13	
3	AP	-18.73	
4	AM	-18.61	
5	RR	-16.41	

6. Analysis based on the payments:

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

```

SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
Extract(month from o.order_purchase_timestamp) as month,
p.payment_type,
COUNT(*) AS order_count
FROM `Target.orders` as o
left join `Target.payments` as p
on o.order_id=p.order_id
GROUP BY year,month, p.payment_type
ORDER BY year asc,month asc;

```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	year ▼	month ▼	payment_type ▼	order_count ▼	
1	2016	9	credit_card	3	
2	2016	9	<i>null</i>	1	
3	2016	10	credit_card	254	
4	2016	10	UPI	63	
5	2016	10	voucher	23	
6	2016	10	debit_card	2	
7	2016	12	credit_card	1	
8	2017	1	credit_card	583	
9	2017	1	UPI	197	
10	2017	1	voucher	61	

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

```
SELECT payment_installments,  
COUNT(*) AS order_count  
FROM `Target.payments`  
WHERE payment_installments > 0  
GROUP BY payment_installments;
```

Query results

JOB INFORMATION		RESULTS	JSON
Row	payment_installment	order_count ▼	
1	1	52546	
2	2	12413	
3	3	10461	
4	4	7098	
5	5	5239	
6	6	3920	
7	7	1626	
8	8	4268	
9	9	644	
10	10	5328	

7. Actionable Insights & Recommendations:

Valuable insights:

1. States with Consistently Faster and Slower Delivery: Target can determine the states whose order delivery is consistently faster or slower than the anticipated dates by analysing the order delivery data. Target may use the best practises from quicker states or concentrate on increasing delivery efficiency in slower states as a result of this understanding.
2. Analysis of Order Numbers Based on Payment Installments: This analysis can reveal information about client preferences for various forms of payment. Target can determine how popular installment payments are and adjust its payment options accordingly.
3. Order Trends from Month to Month: Analysing order trends from month to month for orders made with various payment methods might show seasonal or trend tendencies. Target can pinpoint peak demand periods or times for particular payment methods and adjust its marketing and inventory strategy in line with those findings.

Potential Action Items:

1. Improve Delivery Speed in Slower States
2. Expand Payment Options
3. Promote Preferred Payment Types

4. Monitor and Address Seasonal Demand
5. Enhance Communication and Transparency