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

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

2) Get the time range between which the orders were placed.
ANSWER:

```
select
  MIN(order_purchase_timestamp) as minimum,
  MAX(order_purchase_timestamp) as maximum
from `targetSQL.orders`;
```

Job information | Results | Visualisation | JSON | Executio

ow	minimum	maximum
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

INSIGHT: The span of order purchases is between Septmenber2016 to October2018.

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

```
select
  COUNT(DISTINCT customer_city) as count_cities,
  COUNT(DISTINCT customer_state) as count_states
from `targetSQL.customers`
where customer_id in (
    select
      DISTINCT(customer_id)
    from `targetSQL.orders`
    );
```

Job information | Results | Visualisation | J

Row	count_cities	count_states
1	4119	27

INSIGHT: The Sales is happened in multiple cities related to multiple states.
RECOMMODATION: For Increase Sales, increase the marketing, so that prioritize the cities and states with high order purchased with demand.

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

ANSWER:

```
select
  extract(year from order_purchase_timestamp) as `ORDER_YEAR`,
  COUNT(order_id) as `ORDER_COUNTS`
from `targetSQL.orders`
group by 1
order by 1;
```

Row	ORDER_YEAR	ORDER_COUNTS
1	2016	329
2	2017	45101
3	2018	54011

INSIGHT: The Number of Orders got increased gradually year over year.
RECOMMODATION: Increase the support, improve the marketing for repeat purchasing and for increasing the new customers.

5) Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
ANSWER:

```
select
  extract(month from order_purchase_timestamp) as `ORDER_MONTH`,
  COUNT(order_id) as `ORDER_COUNTS`
from `targetSQL.orders`
group by 1
order by 1;
```

Row	ORDER_MONTH	ORDER_COUNTS
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
11	11	7544
12	12	5674

INSIGHT: Purchasing of orders got increased during monthly seasonality.
RECOMMODATION: After August, increase the offer of the products or do R&D for why order purchased decreased after August and introduce the high demand products.

6) 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

ANSWER:

```
select
  case
    when extract(hour from order_purchase_timestamp) between 0 and 6 then "DAWN(0-6)"
    when extract(hour from order_purchase_timestamp) between 7 and 12 then "MORNING(7-12)"
    when extract(hour from order_purchase_timestamp) between 13 and 18 then "AFTERNOON(13-18)"
    else "NIGHT(19-24)"
  end as `time_of_the_day`,
  count(order_id) as `orders_count`
from `targetSQL.orders`
group by 1
order by 2 desc;
```

Row	time_of_the_day	orders_count
1	AFTERNOON(13-18)	38135
2	NIGHT(19-24)	28331
3	MORNING(7-12)	27733
4	DAWN(0-6)	5242

INSIGHT: Mostly customer placing orders in Afternoon and Night.
RECOMMODATION: Do some promotions and ads, during high window using period.

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

ANSWER:

```
select
  c.customer_state as `state`,
  extract(month from o.order_purchase_timestamp) as `month`,
  count(o.order_id) as `count_orders`
from `targetSQL.customers` as c
inner join `targetSQL.orders` as o
on c.customer_id=o.customer_id
group by 1,2
order by 1,2;
```

Row	state	month	count_orders
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	1	39
14	AL	2	39
15	AL	3	40

INSIGHT: The number of orders purchased is differed by state to state.
RECOMMODATION: Use this to identify under-performing states and boost localized campaigns.

8) How are the customers distributed across all the states?
ANSWER:

```
select
  customer_state,
  count(customer_id) as `no_of_customers`
from `targetSQL.customers`
group by 1;
```

Row	customer_state	no_of_customers
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

INSIGHT: Volume of orders varies across each states.
RECOMMODATION: Focus retention and onboarding strategies where user density is highest.

9) 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.
ANSWER:

```
with base_table as (
  select
  extract(year from o.order_purchase_timestamp) as`year`,
```

```
  sum(p.payment_value) as `cost`
from `targetSQL.orders` as o
inner join `targetSQL.payments` as p
on o.order_id=p.order_id
where extract(year from o.order_purchase_timestamp) between 2017 and 2018
and extract(month from o.order_purchase_timestamp) between 1 and 8
group by 1),
actual_cost_table as(
          select
            *,
            lead(cost,1) over(order by year asc) as `next_cost`
          from base_table)
select
  year,
  (next_cost- cost)/cost * 100 as `percent`
from actual_cost_table;
```

Row	year	percent
1	2017	136.9768716466...
2	2018	null

INSIGHT: The total number of order got increased from 2017 to 2018.
RECOMMODATION: Investigate if this is driven by price hikes, higher basket size, or more orders.

10) Calculate the Total & Average value of order price for each state.
ANSWER:

```
select
  c.customer_state,
  sum(oi.price) as total_price,
  avg(oi.price) as avg_price
from `targetSQL.customers` as c
inner join `targetSQL.orders` as o1
on c.customer_id= o1.customer_id
inner join `targetSQL.order_items` as  oi
on o1.order_id=oi.order_id
group by 1
order by total_price desc;
```

Row	customer_state	total_price	avg_price
1	SP	5202955.050001...	109.6536291597...
2	RJ	1824092.669999...	125.1178180945...
3	MG	1585308.029999...	120.7485741488...
4	RS	750304.0200000...	120.3374530874...
5	PR	683083.7600000...	119.0041393728...
6	SC	520553.34000001	124.6535775862...
7	BA	511349.9900000...	134.6012082126...
8	DF	302603.9399999...	125.7705486284...
9	GO	294591.9499999...	126.2717316759...
10	ES	275037.3099999...	121.9137012411...
11	PE	262788.0299999...	145.5083222591...
12	CE	227254.7099999...	153.7582611637...
13	PA	178947.8099999...	165.6924166666...
14	MT	156453.5299999...	148.2971848341...
15	MA	119648.2199999...	145.2041504854...

INSIGHT: Some states have significant higher total and average price.
RECOMMODATION: Prioritize high-value states for premium offerings and loyalty programs.

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

ANSWER:

```
select
  c.customer_state,
  sum(oi.freight_value) as total_freight_value,
  avg(oi.freight_value) as avg_freight_value
from `targetSQL.customers` as c
inner join `targetSQL.orders` as o1
on c.customer_id= o1.customer_id
inner join `targetSQL.order_items` as  oi
on o1.order_id=oi.order_id
group by 1
order by total_freight_value desc;
```

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...
11	ES	49764.59999999...	22.05877659574...
12	CE	48351.58999999...	32.71420162381...
13	PA	38699.30000000...	35.83268518518...
14	MA	31523.77000000...	38.25700242718...
15	MT	29715.43000000...	28.16628436018...

INSIGHT: Due to logistics and geography, freight cost varies widely by states.
RECOMMODATION: Make a simple and easy in shipping partners in high freight cost states.

12) 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_estimated_delivery_date - order_delivered_customer_date

ANSWER:

```
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 `targetSQL.orders`
where order_delivered_customer_date is not null;
```

Row	order_id	time_to_deliver	diff_estimated_d...
1	65d1e226dfaeb8cdc42f665422...	35	16
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	1950d777989f6a877539f53795...	30	-12
4	bfbd0f9bdef84302105ad712db...	54	-36
5	98974b076b01553d49ee64679...	43	6
6	c4b41c36dd589e901f6879f25a...	36	14
7	d2292ff2201e74c5db154d1b7a...	29	20
8	95e01270fcbae986342340010...	30	19
9	ed8c7b1b3eb256c70ce0c7423...	44	5
10	5cc475c7c03290048eb2e742c...	68	-18
11	6b3ee7697a02619a0ace2b3f0a...	47	2
12	3b2ca3293a7ce539ea2379d70...	43	7
13	b2f92b2f7047cd8b35580d629d...	43	7
14	e2eaf909eb6ba881117aa4079...	40	10
15	90aea7c4e52538a18cb9bbfd16...	43	10

INSIGHT: Before the estimated date or on the delivery date, the orders got delivered.
RECOMMODATION: Use this metric to flag carriers causing frequent delays.

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

ANSWER:

```
with base as (
select
  c.customer_state,
  avg(oi.freight_value) as avg_freight_value
from `targetSQL.customers` as c
inner join `targetSQL.orders` as o1
on c.customer_id= o1.customer_id
inner join `targetSQL.order_items` as  oi
on o1.order_id=oi.order_id
```

```
group by 1),
high_ as (
  select
    customer_state,
    row_number() OVER(ORDER BY avg_freight_value desc) as `highest_freight_rn`
  from base
  limit 5),
low_ as (
  select
    customer_state,
    row_number() OVER(ORDER BY avg_freight_value asc) as `lowest_freight_rn`
  from base
  limit 5)
select
  h.customer_state as `hightest_5_avg_fv`,
  l.customer_state as `lowest_5_avg_fv`
from high_ as h
full outer join low_ as l
on h.highest_freight_rn=l.lowest_freight_rn
order by h.highest_freight_rn;
```

Row	hightest_5_avg_fv	lowest_5_avg_fv
1	RR	SP
2	PB	PR
3	RO	MG
4	AC	RJ
5	PI	DF

INSIGHT: Some states consistently incur high freight costs while others are very low.
RECOMMODATION: For high-cost states, explore regional warehouses or route optimization.

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

ANSWER:

```
with base as (
select
  c.customer_state,
  avg(DATETIME_DIFF(o1.order_delivered_customer_date,o1.order_purchase_timestamp,
SECOND)/86400.0) as avg_deliver_time
from `targetSQL.customers` as c
inner join `targetSQL.orders` as o1
on c.customer_id= o1.customer_id
where o1.order_delivered_customer_date is not null
group by 1),
high_ as(
  select
    customer_state,
    ROW_NUMBER() over(order by avg_deliver_time desc) as rn
```

```
  from base
  limit 5
),
low_ as(
  select
    customer_state,
    ROW_NUMBER() over(order by avg_deliver_time asc) as rn
  from base
  limit 5
)
select
  h.customer_state as `highest_5_dt`,
  l.customer_state as `lowest_5_dt`
from high_ as h
full outer join low_ as l
on h.rn=l.rn
order by h.rn asc;
```

Row	highest_5_dt	lowest_5_dt
1	RR	SP
2	AP	PR
3	AM	MG
4	AL	DF
5	PA	SC

INSIGHT: Long-delivery states show logistical inefficiencies.
RECOMMODATION: Improve routing, warehouse distribution, or carrier selection in slow states.

15) 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.
ANSWER:

```
with base as (
  select
  c.customer_state,
  (DATETIME_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp, SECOND)/86400.0)
as `delivered_time`,
  (DATETIME_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,
SECOND)/86400.0) as `estimated_delivert_time`
from `targetSQL.orders` as o
inner join `targetSQL.customers` as c
on c.customer_id=o.customer_id
where order_delivered_customer_date is not null),
base2 as (
select
```

```
  customer_state,
  (estimated_delivert_time-delivered_time) as `diff_`
from base
where delivered_time<estimated_delivert_time
order by 2 desc)
select DISTINCT(customer_state)
from base2
limit 5;
```

Row	customer_state
1	RJ
2	SP
3	MG
4	RS
5	PR

INSIGHT: These states consistently deliver earlier than expected.
RECOMMODATION: Use them as benchmarks for improving performance in other regions.

16) Find the month on month no. of orders placed using different payment types**.**
ANSWER:

```
select
  extract(month from o.order_purchase_timestamp) as `ordered_month`,
  p.payment_type,
  count(o.order_id) as `count_of_orders`
from `targetSQL.orders` as o
inner join `targetSQL.payments` as p
on o.order_id=p.order_id
group by 1,2
order by 1 asc,3 desc;
```

Row	ordered_month	payment_type	count_of_orders
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118
5	2	credit_card	6609
6	2	UPI	1723
7	2	voucher	424
8	2	debit_card	82
9	3	credit_card	7707
10	3	UPI	1942
11	3	voucher	591
12	3	debit_card	109
13	4	credit_card	7301
14	4	UPI	1783
15	4	voucher	572

INSIGHT: Payment preferences shift monthly, with some methods consistently leading.
RECOMMODATION: Promote the most-used payment options during peak months.

17) Find the no. of orders placed on the basis of the payment installments that have been paid.
ANSWER:

```
select
  p.payment_installments,
  count(distinct(o.order_id)) as `t_orders`
from `targetSQL.payments` as p
inner join `targetSQL.orders` as o
on o.order_id=p.order_id
group by 1
order by 2 desc;
```

Row	payment_installm...	t_orders
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	10	5315
6	5	5234
7	8	4253
8	6	3916
9	7	1623
10	9	644
11	12	133
12	15	74
13	18	27

INSIGHT: Most customers prefer specific installment ranges when paying.
RECOMMODATION: Offer flexible installment options aligned with top-used brackets.