Answers

1.a. Data type of columns in a table

```
SELECT column_name, data_type
FROM target-19087.casestudy.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers'
```

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

1.b. Get the time period for which the data is given

```
SELECT MIN(order_purchase_timestamp) AS first_order,
    MAX(order_purchase_timestamp) AS last_order
    from `target-19087.casestudy.orders`
```

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

1.c. Number of cities and states in our dataset

```
select count(distinct(geolocation_city)) as city_count,
count(distinct(geolocation_state)) as state_count
from `target-19087.casestudy.geolocations`;
```

Row	city_count	state_count
1	8011	27

2.a. Is there a growing trend in e-commerce in Brazil? How can we describe a complete scenario?

For that we will try and understand the trend in the data as how things have changed for the data over the course of time

```
SELECT Extract( year from order_purchase_timestamp) as year,
Extract( month from order_purchase_timestamp) as month,
COUNT(1)  as num_orders
FROM `target-19087.casestudy.orders`
GROUP BY year, month
ORDER BY year, month
```

Row	year	month	num_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
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544

b. Question: Can we see some seasonality with peaks at specific months?

```
SELECT  Extract( month from order_purchase_timestamp) as month, COUNT(1)  as num_orders
FROM `target-19087.casestudy.orders`
GROUP BY 1
ORDER BY 1
```

Row	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
11	11	7544
12	12	5674

In general we can see clearly that customers are increasingly started buying things online.

2.c. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or 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 'night'
end as time_of_day, count(distinct order_id) as counter
from `sqlfreetest.Ecommerce.orders`
group by time_of_day
order by counter desc
```

Row	time_of_day	counter
1	afternoon	38135
2	night	28331
3	morning	27733
4	dawn	5242

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

Now we'll try to understand data based on state or city level and see what variations are happening and how the people in various states order and receive deliveries.

3.a. Get month on month orders by states.

```
select Extract( month from order_purchase_timestamp) as month, c.customer_state, COUNT(1)
as num_orders
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.customers` c
on o.customer_id = c.customer_id
group by c.customer_state, month
order by num_orders desc
```

Row	month	customer_state	num_orders
1	8	SP	4982
2	5	SP	4632
3	7	SP	4381
4	6	SP	4104
5	3	SP	4047
6	4	SP	3967
7	2	SP	3357

3.b. Distribution of customers across the states in Brazil

```
select customer_state, COUNT(distinct(customer_unique_id)) as num_customers
from `target-19087.casestudy.customers`
group by customer_state
order by num_customers desc;
```

Row	customer_state	num_customers
1	SP	40302
2	RJ	12384
3	MG	11259
4	RS	5277
5	PR	4882
6	SC	3534
7	BA	3277
8	DF	2075
9	ES	1964

4. Impact on Economy:

Up until now, we just answered questions on the E-commerce scenario considering the number of orders received. We could see the volumetry on month, day of week, time of the day and even the geolocation states.

Now, **if analyze the money movement by e-commerce by looking at order prices, freight and others.**

4.a.Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) – For that we can use "payment_value" column in payments table

```
with base as
(
select * from `target-19087.casestudy.orders` a
inner join
```

```
`target-19087.casestudy.payments` b
on a.order_id = b.order_id
where
extract(year from a.order_purchase_timestamp) between 2017 and 2018
and
extract(month from a.order_purchase_timestamp) between 1 and 8
),
base_2 as
(
select extract(year from order_purchase_timestamp) as year, sum(payment_value) as cost
from base
group by 1
order by 1 asc
),
base_3 as
(
select *, lead(cost, 1) over (order by year) as next_year_cost from base_2
)
select *, (next_year_cost - cost)/ cost *100 as percent_increase from base_3
```

Row	year	cost	next_year_cost	percent_increas
1	2017	3669022.11...	8694733.83...	136.976871...
2	2018	8694733.83...	*null*	*null*

Breakdown and related queries:

- **Create CTE Table and new columns:**
  - **price_per_order = sum(price)/count(order_id)**
  - **freight_per_order= sum(freight_value)/count(order_id)**
  - **Group the data on yearly and monthly level**

```
with cte_table as (

select Extract( month from o.order_purchase_timestamp) as month,
Extract( year from o.order_purchase_timestamp) as year,
(sum(price)/count(o.order_id)) as price_per_order,
(sum(freight_value)/count(o.order_id)) as freight_per_order
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.order_items` i
on o.order_id= i.order_id
group by year,month
```

```
)
select (price_per_order), (freight_per_order), month , year
from cte_table
```

Row	price_per_order	freight_per_order	month	year
1	126.27005358150556	23.014778623801426	7	2018
2	117.92029939294153	20.505262141280323	8	2018
3	122.35762572534202	19.371327369438863	5	2017
4	122.22722661769379	22.259508335687997	6	2018
5	124.80635663285128	19.74688838782436	10	2017
6	110.03372132430309	18.604047184567456	2	2018
7	125.74355583596814	19.339323659305862	5	2018
8	124.78143333333442	19.234763333333277	3	2017
9	116.59219503751369	19.489024812464162	11	2017

4.a. Total amount sold in 2017 between Jan to august (Jan to Aug because data is available starting 2017 -01 to 2018- 08) and we can only compare cycles with cycles

```
with cte_table as (
select
Extract( month from order_purchase_timestamp) as month,
Extract( year from order_purchase_timestamp) as year,
sum(price) as total_price,
sum(freight_value) as total_freight
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.order_items` i
on o.order_id= i.order_id
group by year, month
)
select sum(total_price) as total_transaction_amt
from cte_table
where year =2017 and month between 1 and 8
```

Row	total_transaction_amt
1	3113000.3199994233

(3.9M)

4.a. Total amount sold in 2018 between Jan to august

```
with cte_table as (
select
Extract( month from order_purchase_timestamp) as month,
Extract( year from order_purchase_timestamp) as year,
sum(price) as total_price,
sum(freight_value) as total_freight
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.order_items` i
on o.order_id= i.order_id
group by year, month

)
select sum(total_price)
from cte_table
where year =2018 and month between 1 and 8
```

Row	f0_
1	7385905.8000043072

4.a. % increase from 2017 to 2018

Using another example (using orders and customers table)

```
select *, (orders-coalesce(lagger_orders,0))/coalesce(orders,1)*100 as difference from (

select *, lag (orders,1) over (order by year asc) as lagger_orders from (
select extract(year from a.order_purchase_timestamp) as year,
count(distinct a.order_id) as orders,
count(distinct b.customer_unique_id) as customers
from `sqlfreetest.Ecommerce.orders` a
left join `sqlfreetest.Ecommerce.customers` b
on a.customer_id=b.customer_id
group by 1
)base) base_2
```

```
order by year asc
```

Row	year	orders	customers	lagger_orders	difference
1	2016	329	326	*null*	100.0
2	2017	45101	43713	329	99.270526152413481
3	2018	54011	52749	45101	16.496639573420229

- **4.b. Sum and mean price by customer state**

```
with cte_table as (
select
c.customer_state as state,
sum(price) as total_price,
count(distinct(o.order_id)) as num_orders
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.order_items` i
on o.order_id= i.order_id
inner join `target-19087.casestudy.customers` c
on o.customer_id=c.customer_id
group by state

)
select state, total_price,  num_orders,(total_price/num_orders) as avg_price
from cte_table
order by total_price desc
```

Row	state	total_price	num_orders	avg_price
1	SP	5202955.0500027407	41375	125.75117945625959
2	RJ	1824092.6699996467	12762	142.93156793603251
3	MG	1585308.0299997134	11544	137.32744542617061
4	RS	750304.02000004181	5432	138.12666053019916
5	PR	683083.76000003726	4998	136.67142056823474
6	SC	520553.34000002244	3612	144.11775747508926
7	BA	511349.99000002112	3358	152.27813877308552
8	DF	302603.93999999622	2125	142.40185411764529
9	GO	294591.94999999512	2007	146.7822371699029 1
10	[illegible]	[illegible]	[illegible]	[illegible]

It's very interesting to see how the states which have a high total amount sold surprisingly have a low price per order. If we take SP (São Paulo), it's possible to see that it is the state with most valuable state for e-commerce (5202955 sold) but also where customers enjoyed the least price. (125.75 per order)

4.b. Sum and mean freight by customer state

```
with cte_table as (
select
c.customer_state as state,
sum(freight_value) as total_freight,count(distinct(o.order_id)) as num_orders
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.order_items` i
on o.order_id= i.order_id
inner join `target-19087.casestudy.customers` c
on o.customer_id=c.customer_id
group by state

)
select state, total_freight,  num_orders,(total_freight/num_orders) as avg_price
from cte_table
order by total_freight desc
```

Row	state	total_freight	num_orders	avg_price
1	SP	718723.0699999833	41375	17.370950332325879
2	RJ	305589.31000000035	12762	23.94525231154994
3	MG	270853.46000000357	11544	23.462704435204746
4	RS	135522.74000000212	5432	24.948958026509963
5	PR	117851.68000000139	4998	23.579767907163145
6	BA	100156.67999999883	3358	29.826289458010372
7	SC	89660.260000000431	3612	24.822884828350062
8	PE	59449.6599999999	1648	36.073822815533923
9	GO	53114.979999999865	2007	26.464862979571432
10	DF	50625.490000000011	2125	23.823764795283265

5. Analysis on sales, freight and delivery time

create new columns for time to delivery and difference in estimated vs actual delivery:

```
select order_id,TIMESTAMP_DIFF(
order_delivered_customer_date,order_purchase_timestamp, DAY) as time_to_dil,
TIMESTAMP_DIFF(  order_delivered_customer_date,order_estimated_delivery_date ,
DAY) as diff_estimated_dil
from `target-19087.casestudy.orders`
where order_status='delivered'
```

Row	order_id	time_to_dil	diff_estimated_dil
1	635c894d068ac37e6e03dc54eccb6189	30	-1
2	3b97562c3aee8bdedcb5c2e45a50d5e1	32	0
3	68f47f50f04c4cb6774570cfde3a9aa7	29	-1
4	276e9ec344d3bf029ff83a161c6b3ce9	43	4
5	54e1a3c2b97fb0809da548a59f64c813	40	4
6	fd04fa4105ee8045f6a0139ca5b49f27	37	1
7	302bb8109d097a9fc6e9cefc5917d1f3	33	5
8	66057d37308e787052a32828cd007e58	38	6
9	19135c945c554eebfd7576c733d5ebdd	36	2
10	4493e45e7ca1084efcd38ddebf174dda	34	0
11	70e77e51e0f179d75a64a614135afb6a	42	11

5.4.a Top 5 states with highest/lowest average time to delivery:

```
select g.geolocation_state as state,
SUM(TIMESTAMP_DIFF(  order_delivered_customer_date,order_purchase_timestamp,
DAY))/COUNT(ORDER_ID) AS avg_dil_time,
from `target-19087.casestudy.orders` o
inner join `sql free test-353004.Ecommerce.customers` c
on o.customer_id=c.customer_id
inner join `target-19087.casestudy.geolocations` g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
where order_status='delivered'
group by state
order by avg_dil_time
limit 5
```

Row	state	avg_dil_time
1	SP	8.4688929143521126
2	PR	11.038764047706067
3	MG	11.418216783727036
4	DF	12.496517892339362
5	SC	14.484084345800198

```
select
c.customer_state as state,
avg(freight_value) as total_freight
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.order_items` i
on o.order_id= i.order_id
inner join `target-19087.casestudy.customers` c
on o.customer_id=c.customer_id
group by state
order by total_freight desc
limit 5
```

Row	state	total_freight
1	RR	42.9844230...
2	PB	42.7238039...
3	RO	41.0697122...
4	AC	40.0733695...
5	PI	39.1479704...

5.4.c. Top 5 states where delivery is really fast/ not so fast compared to estimated date:

```
select geolocation_state,
SUM(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp,
DAY))/COUNT(ORDER_ID) AS average_time_for_del,
SUM(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID) AS average_est_dil_time,
from `target-19087.casestudy.orders` o
inner join `target-19087.casestudy.customers` c
on o.customer_id=c.customer_id
inner join `target-19087.casestudy.geolocations` g
```

```
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
where order_status='delivered'
group by geolocation_state
order by (average_time_for_del-average_est_dil_time)
```

Row	geolocation_state	average_time_for_del	average_est_dil_time
1	RR	24.520601336302896	45.259465478841868
2	AM	24.651196784213411	45.133382057372557
3	RO	18.654498235985887	37.63690709525676
4	AC	20.508373205741627	39.210260499734183
5	AP	27.991226237727179	46.568414455817837
6	MT	17.347533912348343	31.85192648785484
7	PA	22.550239824416469	36.410007274879469
8	RS	14.534751620280913	28.036358913604023
9	PR	19.486211631963776	32.490273169283?

6. Payment type analysis

6.1. Count of orders for different payment types

Row	payment_type	order_count	month	year
1	credit_card	3	9	2016
2	credit_card	254	10	2016
3	UPI	63	10	2016
4	voucher	23	10	2016
5	debit_card	2	10	2016
6	credit_card	1	12	2016
7	credit_card	583	1	2017
8	UPI	197	1	2017
9	voucher	61	1	2017
10	debit_card	9	1	2017
11	credit_card	1356	2	2017

6.3. Count of orders based on the no. of payment installments:

```
SELECT distinct(payment_installments) as installments, count(order_id) as Num_orders,
```

```
FROM `target-19087.casestudy.payments`
where payment_installments>1
GROUP BY payment_installments
order by Num_orders desc;
```

Row	installments	Num_orders
1	2	12413
2	3	10461
3	4	7098
4	10	5328
5	5	5239
6	8	4268
7	6	3920
8	7	1626
9	9	644
10	12	133
11	15	74

ADDITIONAL QUESTIONS

- **Rank payment_value partitioned by payment_type:**

```
select payment_type, payment_value, order_id,
rank() over(partition by payment_type order by payment_value desc ) as rank
from `target-19087.casestudy.payments`
```

Row	payment_type	payment_value	order_id	rank
1	voucher	3184.34	7813842ae95e8c497fc0233232ae815a	1
2	voucher	3184.34	03310aa823a66056268a3bab36e827fb	1
3	voucher	2266.61	afb61112fb99b07fe17e591c68c0c84c	3
4	voucher	1839.05	4df2b9c2c7b6eedea39ceb96eb54ad34	4
5	voucher	1522.42	a739bf4717343c5f9c84196b61a9c53f	5
6	voucher	1400.33	c3fbba8f64cc6b2e99abc0e00d5ade2b	6
7	voucher	1302.42	f86d7bc39aab05299691322044b63bb2	7
8	voucher	1224.1	99bc429ddaa03f67e02a463a20c2379f	8
9	voucher	1220.45	c08dd05931abd8cb08aad3d31e39bfed	9
10	voucher	1201.08	f7a8fae2d2d1ed95f1413db630a42dbe	10
11	voucher	1113.99	f398a143e0fe171d965db2096ef064ef	11

- **For each seller rank the items by price:**

```
select  order_id,product_id, seller_id, price,
rank() over(partition by seller_id order by price desc  ) as rank
from `target-19087.casestudy.order_items`
```

Row	order_id	product_id	seller_id	price	rank
1	8501926dd0837d694fc5af339c02a6b2	093cd981b714bcdff182b427d87fc8fc	001e6ad469a905060d959994f1b41e4f	250.0	1
2	1fc6f36a09a4afd9024b0cf47d52924e	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
3	77e18878827762954f8e0697901368de	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
4	bedbd4ec33763c46cc1043d118e96c27	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
5	f24616037f41893b85ebeff4018c6933	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
6	0629053d0e1e598c7a5048031e19e31b	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
7	06b54aee2ec1ca4bd7a460c7d256cce1	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
8	1c4a92d82c1b0dec18bef12da3fa7756	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
9	3c13ca3111b1b5f67cc6bae982e812d2	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
10	d55a3d635aefc15c1bf94b37a867c1c9	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5

- **What percentage of orders were canceled or unavailable:**

```
select counter/total*100  from (
select sum(case when order_status in ('canceled','unavailable') then 1 else 0 end) as counter,
count (1) as total from `target-19087.casestudy.orders`
);
```

Row	f0_
1	1.2409368369183738

- **Find customers with more than one order:**

```
SELECT customer_unique_id, count(1) as ordercount FROM  `target-19087.casestudy.customers` c
JOIN  `target-19087.casestudy.orders` o ON c.customer_id = o.customer_id
GROUP BY customer_unique_id
HAVING COUNT(order_purchase_timestamp) >1
ORDER BY COUNT(order_purchase_timestamp) DESC ;
```

Row	customer_unique_id	ordercount
1	8d50f5eadf50201ccdcedfb9e2ac8455	17
2	3e43e6105506432c953e165fb2acf44c	9
3	ca77025e7201e3b30c44b472ff346268	7
4	6469f99c1f9dfae7733b25662e7f1782	7
5	1b6c7548a2a1f9037c1fd3ddfed95f33	7
6	47c1a3033b8b77b3ab6e109eb4d5fdf3	6
7	f0e310a6839dce9de1638e0fe5ab282a	6
8	12f5d6e1cbf93dafd9dcc19095df0b3d	6
9	dc813062e0fc23409cd255f7f53c7074	6
10	[illegible]	6

- **Avg time for delivery:**

```
select SUM(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp, DAY))/COUNT(ORDER_ID) AS average_time_for_del
from `target-19087.casestudy.orders`where order_status='delivered' limit 1
```

Row	average_time_for_del
1	12.092601422085863