Target is Brazil based e-commerce website which provides best shopping experience to its customers, Target has provided data for 100k orders which includes order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

Business Problem : Target needs our help in summarising the data, understand the behaviour of its customers, need and demand of its customer and extract useful insights form the data.

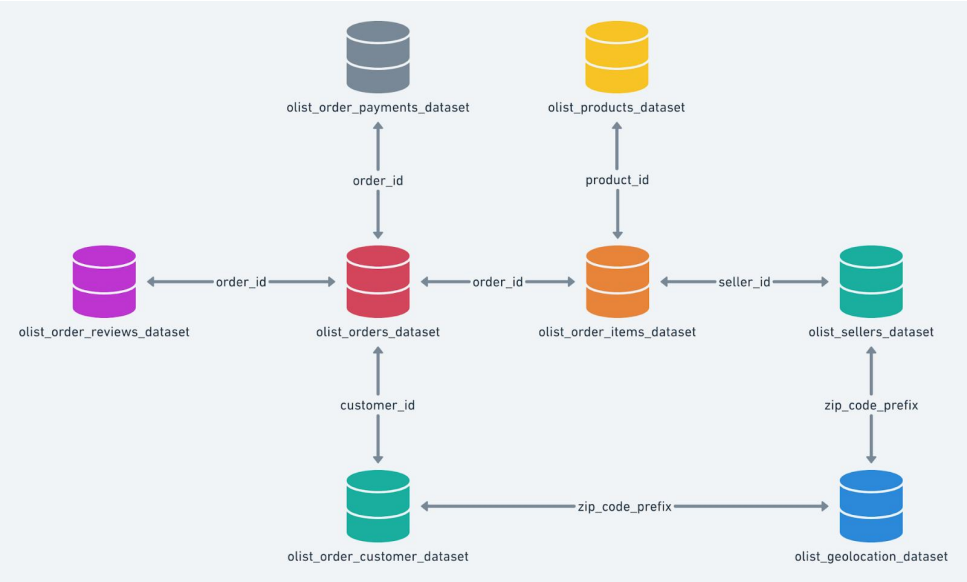

Fig: Schema of the data

1.Importing data set to find structure and characteristics of the dataset :

1.1. Data type of columns in a table

Query:

```
--Data type of columns in a table

SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'customer'
```

Output :

Row	table_catalog	table_schema	table_name
1	pro-core-360407	ecommerce	customer
2	pro-core-360407	ecommerce	customer
3	pro-core-360407	ecommerce	customer
4	pro-core-360407	ecommerce	customer
5	pro-core-360407	ecommerce	customer

Query:

```
SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'geolocation'
```

Output:

Row	table_catalog	table_schema	table_name
1	pro-core-360407	ecommerce	geolocation
2	pro-core-360407	ecommerce	geolocation
3	pro-core-360407	ecommerce	geolocation
4	pro-core-360407	ecommerce	geolocation
5	pro-core-360407	ecommerce	geolocation

Query:

```
SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'order_items'
```

Output:

Row	table_catalog	table_schema	table_name
1	pro-core-360407	ecommerce	order_items
2	pro-core-360407	ecommerce	order_items
3	pro-core-360407	ecommerce	order_items
4	pro-core-360407	ecommerce	order_items
5	pro-core-360407	ecommerce	order_items
6	pro-core-360407	ecommerce	order_items
7	pro-core-360407	ecommerce	order_items

Query:

```
SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'order_reviews'
```

Output:

Row	table_catalog	table_schema	table_name
1	pro-core-360407	ecommerce	order_reviews
2	pro-core-360407	ecommerce	order_reviews
3	pro-core-360407	ecommerce	order_reviews
4	pro-core-360407	ecommerce	order_reviews
5	pro-core-360407	ecommerce	order_reviews
6	pro-core-360407	ecommerce	order_reviews

Query:

```
SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'orders'
```

Output:

Row	table_catalog	table_schema	table_name
1	pro-core	ecommerce	orders
2	pro-core-360407	ecommerce	orders
3	pro-core-360407	ecommerce	orders
4	pro-core-360407	ecommerce	orders
5	pro-core-360407	ecommerce	orders
6	pro-core-360407	ecommerce	orders
7	pro-core-360407	ecommerce	orders
8	pro-core-360407	ecommerce	orders

Query:

```
SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'payments'
```

Output:

Row	table_catalog	table_schema	table_name
1	pro-core-360407	ecommerce	payments
2	pro-core-360407	ecommerce	payments
3	pro-core-360407	ecommerce	payments
4	pro-core-360407	ecommerce	payments
5	pro-core-360407	ecommerce	payments

Query:

```
SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'products'
```

Output:

Row	table_catalog	table_schema	table_name
1	pro-core-360407	ecommerce	products
2	pro-core-360407	ecommerce	products
3	pro-core-360407	ecommerce	products
4	pro-core-360407	ecommerce	products
5	pro-core-360407	ecommerce	products
6	pro-core-360407	ecommerce	products
7	pro-core-360407	ecommerce	products
8	pro-core-360407	ecommerce	products
9	pro-core-360407	ecommerce	products

Query:

```
SELECT
  * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
  `pro-core-360407.ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'sellers'
```

Output:

Row	table_catalog	table_schema	table_name
1	pro-core-360407	ecommerce	sellers
2	pro-core-360407	ecommerce	sellers
3	pro-core-360407	ecommerce	sellers
4	pro-core-360407	ecommerce	sellers

1.2 Time period for which data is given :

```
SELECT
  EXTRACT(month FROM order_purchase_timestamp) Month,
  EXTRACT(year FROM order_purchase_timestamp) Year,

FROM `pro-core-360407.ecommerce.orders`
GROUP BY Year,Month
ORDER BY Year,Month
```

Output:

Row	Month	Year
1	9	2016
2	10	2016
3	12	2016
4	1	2017
5	2	2017
6	3	2017
7	4	2017
8	5	2017
9	6	2017
10	7	2017

1.3 Cities and States covered in the dataset

```
--Citites
SELECT
  DISTINCT geolocation_city ,
FROM `pro-core-360407.ecommerce.geolocation`
ORDER BY geolocation_city
```

Output:

Row	geolocation_city
1	* cidade
2	...arraial do cabo
3	4o. centenario
4	4º centenario
5	abadia de goias
6	abadia dos dourados
7	abadiania
8	abadiânia
9	abaete
10	abaetetuba

```
--Which city belongs to which state?
SELECT
  DISTINCT geolocation_state ,
  geolocation_city
FROM `pro-core-360407.ecommerce.geolocation`
ORDER BY geolocation_state
```

Output:

Row	geolocation_state	geolocation_city
1	AC	sena madureira
2	AC	rio branco
3	AC	feijo
4	AC	senador guiomard
5	AC	cruzeiro do sul
6	AC	xapuri
7	AC	feijó
8	AC	manoel urbano
9	AC	santa rosa do purus
10	AC	placido de castro

```sql
--How many cities are there in particular state?
SELECT
  geolocation_state ,
  COUNT(DISTINCT geolocation_city) count
FROM `pro-core-360407.ecommerce.geolocation`
GROUP BY geolocation_state
ORDER BY geolocation_state
```

Output:

Row	geolocation_state	count
1	AC	34
2	AL	130
3	AM	74
4	AP	17
5	BA	652
6	CE	260
7	DF	28
8	ES	160
9	GO	384
10	MA	299

2. In-depth Exploration :

2.1 Trend in Brazil, seasonality etc.

```sql
SELECT
  EXTRACT(month FROM order_approved_at)  month,
  COUNT(`pro-core-360407.ecommerce.orders`.order_id) order_counts
FROM `pro-core-360407.ecommerce.orders` JOIN `pro-core-360407.ecommerce.order_items` ON `pro-core-360407.ecommerce.orders`.order_id = `pro-core-360407.ecommerce.order_items`.order_id
GROUP BY month
ORDER BY order_counts DESC
```

Output:

Row	month	order_counts
1	8	12375
2	5	12289
3	7	11424
4	3	11330
5	6	10670
6	4	10440
7	2	9589
8	1	9008
9	11	8508
10	12	6505

2.2 At what time does Brazallians buy?

```
--Which time does Brazallians buy?
SELECT x.time_period, COUNT(x.order_id) Freqeuncy
FROM
(SELECT
  order_id,
  EXTRACT (hour FROM order_purchase_timestamp),
  CASE
    WHEN EXTRACT (hour FROM order_purchase_timestamp) BETWEEN 6 AND 12 THEN 'morning
'
    WHEN EXTRACT (hour FROM order_purchase_timestamp) BETWEEN 12 AND 17 THEN 'aftern
oon'
    WHEN EXTRACT (hour FROM order_purchase_timestamp) BETWEEN 17 AND 22 THEN 'evenin
g'
    ELSE 'night' END AS time_period
FROM `pro-core-360407.ecommerce.orders`
) AS x
GROUP BY x.time_period
```

Row	time_period	Freqeuncy
1	morning	28235
2	night	8863
3	afternoon	32366
4	evening	29977

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

3.1 Month on month order by states

```
SELECT
  c.customer_state State,
  EXTRACT(month FROM order_approved_at)  Month,
  EXTRACT(year FROM order_approved_at) Year,
  COUNT(o.order_id) order_counts
FROM `pro-core-360407.ecommerce.orders` o
  JOIN `pro-core-360407.ecommerce.order_items` ON o.order_id = `pro-core-
360407.ecommerce.order_items`.order_id
  JOIN `pro-core-360407.ecommerce.customer` c ON c.customer_id = o.customer_id
WHERE EXTRACT(month FROM order_approved_at) IS NOT NULL
GROUP BY State, Month, Year
ORDER BY State,Year,Month
```

Output:

Row	State	Month	Year	order_counts
1	AC	1	2017	4
2	AC	2	2017	6
3	AC	3	2017	2
4	AC	4	2017	5
5	AC	5	2017	8
6	AC	6	2017	4
7	AC	7	2017	5
8	AC	8	2017	4
9	AC	9	2017	5
10	AC	10	2017	6

```sql
--Order status count
SELECT order_status,count(order_status) as order_count
FROM `pro-core-360407.ecommerce.orders`
GROUP BY order_status
ORDER BY order_count
```

Row	order_status	order_count
1	approved	2
2	created	5
3	processing	301
4	invoiced	314
5	unavailable	609
6	canceled	625
7	shipped	1107
8	delivered	96478

3.2 Customer distribution in Brazil

```sql
--Which city?
SELECT
  customer_city,
  count(customer_unique_id) customer_count
FROM `pro-core-360407.ecommerce.customer`
GROUP BY customer_city
ORDER BY customer_count DESC
```

Row	customer_city	customer_c...
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245
9	guarulhos	1189
10	sao bernardo do campo	938

```sql
--Which state?
SELECT
  customer_state,
  count(customer_unique_id) customer_count
FROM `pro-core-360407.ecommerce.customer`
GROUP BY customer_state
ORDER BY customer_count DESC
```

Row	customer_state	customer_c...
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

customer_state

```
--Active customers from each city
SELECT
  customer_city,
  COUNT(DISTINCT o.customer_id) count
FROM
  `pro-core-360407.ecommerce.orders` o
  join `pro-core-360407.ecommerce.customer` c on o.customer_id = c.customer_id
GROUP BY customer_city
ORDER BY COUNT DESC
```

Output:

Row	customer_city	count
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245
9	guarulhos	1189
10	sao bernardo do campo	938

4. Analyze the money movement

4.1 Percent change in order amount

```
SELECT
  EXTRACT (YEAR FROM o.order_approved_at) as Year,
  EXTRACT (MONTH FROM o.order_approved_at) as Month,
  ROUND(SUM(p.payment_value),2) Total_amount_per_month,
FROM `pro-core-360407.ecommerce.orders` o  JOIN
  `pro-core-360407.ecommerce.payments` p ON
  o.order_id = p.order_id
WHERE
  (EXTRACT (YEAR FROM o.order_approved_at) BETWEEN 2017 and 2018) and
```

```
  (EXTRACT (MONTH FROM o.order_approved_at) BETWEEN 1 and 8)
GROUP BY Year, Month
ORDER BY Year, Month
```

Row	Year	Month	Total_amou...
1	2017	1	131835.87
2	2017	2	291836.22
3	2017	3	446020.61
4	2017	4	413537.47
5	2017	5	593119.02
6	2017	6	515293.57
7	2017	7	585260.3
8	2017	8	672772.79
9	2018	1	1106076.67
10	2018	2	984422.01

Percent Increase month wise in following year

```
WITH sales_monthwise_2017 AS
(
SELECT
  EXTRACT (MONTH FROM o.order_approved_at) as Month,
  ROUND(SUM(p.payment_value),2) Total_amount_per_month,
FROM `pro-core-360407.ecommerce.orders` o join
  `pro-core-360407.ecommerce.payments` p on
  o.order_id = p.order_id
WHERE
  (EXTRACT (YEAR FROM o.order_approved_at) = 2017) and
  (EXTRACT (MONTH FROM o.order_approved_at) BETWEEN 1 and 8)
GROUP BY Month
ORDER BY Month
),

sales_monthwise_2018 AS
(
SELECT
  EXTRACT (MONTH FROM o.order_approved_at) as Month,
  ROUND(SUM(p.payment_value),2) Total_amount_per_month,
FROM `pro-core-360407.ecommerce.orders` o join
  `pro-core-360407.ecommerce.payments` p on
  o.order_id = p.order_id
WHERE
  (EXTRACT (YEAR FROM o.order_approved_at) = 2018) and
  (EXTRACT (MONTH FROM o.order_approved_at) BETWEEN 1 and 8)
GROUP BY Month
ORDER BY Month
)

SELECT
  s18.Month Month,
```

```
  ROUND((s18.Total_amount_per_month -
 s17.Total_amount_per_month) / s17.Total_amount_per_month * 100,2) as percent_increa
se
FROM sales_monthwise_2017 s17 JOIN sales_monthwise_2018 s18 ON
  s17.Month = s18.Month
ORDER BY Month
```

Output:

Row	Month	percent_incr...
1	1	738.98
2	2	237.32
3	3	162.41
4	4	175.11
5	5	98.97
6	6	99.5
7	7	78.23
8	8	53.91

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

```
SELECT
  c.customer_state,
  ROUND(SUM(price),2) total_product_purchase_price,
  ROUND(SUM(freight_value),2) total_freight_value,
  ROUND(AVG(price),2) avg_product_purchase_price,
  ROUND(AVG(freight_value),2) avg_freight_value
FROM `pro-core-360407.ecommerce.order_items` oi
  JOIN `pro-core-360407.ecommerce.orders` o ON oi.order_id = o.order_id
  JOIN `pro-core-360407.ecommerce.customer` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY total_product_purchase_price DESC
```

Output:

Row	customer_state	total_produc...	total_freight...	avg_product...	avg_freight_...
1	SP	5202955.05	718723.07	109.65	15.15
2	RJ	1824092.67	305589.31	125.12	20.96
3	MG	1585308.03	270853.46	120.75	20.63
4	RS	750304.02	135522.74	120.34	21.74
5	PR	683083.76	117851.68	119.0	20.53
6	SC	520553.34	89660.26	124.65	21.47
7	BA	511349.99	100156.68	134.6	26.36
8	DF	302603.94	50625.5	125.77	21.04
9	GO	294591.95	53114.98	126.27	22.77
10	ES	275037.31	49764.6	121.91	22.06

5. Analysis on sales, freight and delivery time

```
--Days between purchasing, delivering and estimated delivery
SELECT
  DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) delivered_i
n_days ,
  DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) estimated_
days,
  COUNT(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)) freq
uency
FROM `pro-core-360407.ecommerce.orders`
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY delivered_in_days,estimated_days
ORDER BY delivered_in_days DESC
```

Output:

Row	delivered_in...	estimated_d...	frequency
1	125	26	1
2	107	26	1
3	104	13	1
4	66	26	1
5	62	50	1
6	61	50	1
7	55	24	1
8	55	94	1
9	54	43	1
10	53	18	1

```
--Weeks taken to deliver the order:
WITH weeks_to_deliver as
(SELECT
  CASE
    WHEN DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) <= 7
 then '1 week'
    WHEN DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) <= 1
4 then '2 weeks'
    WHEN DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) <= 2
1 then '3 weeks'
    ELSE '3+ weeks' END AS no_of_weeks_to_deliver,
  COUNT(date_diff(order_delivered_carrier_date, order_purchase_timestamp, day)/7) fr
equency_for_delivery
FROM `pro-core-360407.ecommerce.orders`
GROUP BY no_of_weeks_to_deliver
ORDER BY frequency_for_delivery
), weeks_estimated as
(
SELECT
CASE
  WHEN DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) <= 7
then '1 week'
```

```
  WHEN DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) <= 14
 then '2 weeks'
  WHEN DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) <= 21
 then '3 weeks'
  ELSE '3+ weeks' END AS no_of_weeks_estimated,
COUNT(date_diff(order_estimated_delivery_date, order_purchase_timestamp, day)/7) fre
quency_for_estimated
FROM `pro-core-360407.ecommerce.orders`
GROUP BY no_of_weeks_estimated
ORDER BY frequency_for_estimated
)

SELECT x1.no_of_weeks_to_deliver, x1.frequency_for_delivery,x2.frequency_for_estimat
ed
FROM weeks_to_deliver x1 join weeks_estimated x2 on x1.no_of_weeks_to_deliver = x2.n
o_of_weeks_estimated
```

Row	no_of_weeks_to_deliver	frequency_f...	frequency_f...
1	3+ weeks	511	57109
2	1 week	91662	2147
3	2 weeks	4515	12326
4	3 weeks	970	27859

```
--columns for estimated days and days required to deliver the product
SELECT
  order_id,
  DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) time_to_del
iver ,
  DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) diff_estim
ated_delivery
FROM `pro-core-360407.ecommerce.orders`
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
```

Output:

Row	order_id	time_to_deli...	diff_estimat...
1	f88aac7ebccb37f19725a0753...	9	50
2	790cd37689193dca0d00d2feb...	2	6
3	49db7943d60b6805c3a41f547...	6	44
4	063b573b88fc80e516aba87df...	22	54
5	a68ce1686d536ca72bd2dadc4...	33	56
6	45973912e490866800c0aea8f...	18	54
7	cda873529ca7ab71f677d5ec1...	39	56
8	ead20687129da8f5d89d831bb...	1	41
9	6f028ccb7d612af251aa442a1f...	1	3
10	8733c8d440c173e524d2fab80...	0	3

5.3 Grouping data state-wise

```
--finding average freight_value,time_to_deliver,diff_estimated_delivery
SELECT
  c.customer_state,
  ROUND(AVG(ot.freight_value),2) avg_freight_value,
  ROUND(AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),
2) avg_time_to_deliver ,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)),
2) avg_diff_estimated_delivery
FROM `ecommerce.orders` o
  join `ecommerce.customer` c on  o.customer_id = c.customer_id
  join `ecommerce.order_items` ot on o.order_id = ot.order_id
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY c.customer_state
ORDER BY avg_freight_value DESC
```

Output:

Row	customer_state	avg_freight_...	avg_time_to...	avg_diff_est...
1	RR	43.32	4.63	45.9
2	PB	42.82	3.14	32.53
3	RO	41.33	2.34	38.7
4	AC	40.07	2.84	40.7
5	PI	39.04	2.77	29.86
6	MA	38.31	3.4	30.47
7	TO	37.31	3.01	28.81
8	SE	36.69	3.25	30.36
9	AL	35.92	3.15	32.09

```
--Highest freight value
SELECT
  c.customer_state,
  ROUND(AVG(ot.freight_value),2) avg_freight_value,
  ROUND(AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),
2) avg_time_to_deliver ,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)),
2) avg_diff_estimated_delivery
FROM `ecommerce.orders` o
  join `ecommerce.customer` c on  o.customer_id = c.customer_id
  join `ecommerce.order_items` ot on o.order_id = ot.order_id
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY c.customer_state
ORDER BY avg_freight_value DESC
LIMIT 5
```

Output:

Row	customer_state	avg_freight_...	avg_time_to...	avg_diff_est...
1	RR	43.32	4.63	45.9
2	PB	42.82	3.14	32.53
3	RO	41.33	2.34	38.7
4	AC	40.07	2.84	40.7
5	PI	39.04	2.77	29.86

```sql
--Lowest freight value
SELECT
  c.customer_state,
  ROUND(AVG(ot.freight_value),2) avg_freight_value,
  ROUND(AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),
2) avg_time_to_deliver ,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)),
2) avg_diff_estimated_delivery
FROM `ecommerce.orders` o
  join `ecommerce.customer` c on  o.customer_id = c.customer_id
  join `ecommerce.order_items` ot on o.order_id = ot.order_id
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY c.customer_state
ORDER BY avg_freight_value ASC
LIMIT 5
```

Row	customer_state	avg_freight_...	avg_time_to...	avg_diff_est...
1	SP	15.12	2.72	18.87
2	PR	20.47	2.85	24.38
3	MG	20.61	2.79	24.27
4	RJ	20.93	2.9	26.09
5	DF	21.07	2.83	24.17

```sql
--Highest avg_delivery time
SELECT
  c.customer_state,
  ROUND(AVG(ot.freight_value),2) avg_freight_value,
  ROUND(AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),
2) avg_time_to_deliver ,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)),
2) avg_diff_estimated_delivery
FROM `ecommerce.orders` o
  join `ecommerce.customer` c on  o.customer_id = c.customer_id
  join `ecommerce.order_items` ot on o.order_id = ot.order_id
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY c.customer_state
ORDER BY avg_time_to_deliver DESC
LIMIT 5
```

Row	customer_state	avg_freight_...	avg_time_to...	avg_diff_est...
1	RR	43.32	4.63	45.9
2	MA	38.31	3.4	30.47
3	SE	36.69	3.25	30.36
4	RN	35.65	3.2	32.2
5	AL	35.92	3.15	32.09

```sql
--Lowest avg_delivery time
SELECT
  c.customer_state,
  ROUND(AVG(ot.freight_value),2) avg_freight_value,
```

```
  ROUND(AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),
2) avg_time_to_deliver ,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)),
2) avg_diff_estimated_delivery
FROM `ecommerce.orders` o
  join `ecommerce.customer` c on  o.customer_id = c.customer_id
  join `ecommerce.order_items` ot on o.order_id = ot.order_id
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY c.customer_state
ORDER BY avg_time_to_deliver ASC
LIMIT 5
```

Row	customer_state	avg_freight_...	avg_time_to...	avg_diff_est...
1	AM	33.21	2.29	45.21
2	RO	41.33	2.34	38.7
3	GO	22.52	2.62	26.63
4	MS	23.36	2.72	25.69
5	MT	28.06	2.72	31.5

```
--State where delivery is very fast compared to estimated time
SELECT
  c.customer_state,
  ROUND(AVG(ot.freight_value),2) avg_freight_value,
  ROUND(AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),
2) avg_time_to_deliver ,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)),
2) avg_diff_estimated_delivery
FROM `ecommerce.orders` o
  join `ecommerce.customer` c on  o.customer_id = c.customer_id
  join `ecommerce.order_items` ot on o.order_id = ot.order_id
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY c.customer_state
ORDER BY (avg_diff_estimated_delivery - avg_time_to_deliver) ASC
LIMIT 5
```

Row	customer_state	avg_freight_...	avg_time_to...	avg_diff_est...
1	SP	15.12	2.72	18.87
2	DF	21.07	2.83	24.17
3	MG	20.61	2.79	24.27
4	PR	20.47	2.85	24.38
5	ES	22.05	2.96	25.23

```
--State where delivery is not so fast compared to estimated time
SELECT
  c.customer_state,
  ROUND(AVG(ot.freight_value),2) avg_freight_value,
  ROUND(AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),
2) avg_time_to_deliver ,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)),
2) avg_diff_estimated_delivery,
```

```
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day))
 -
 AVG(DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day)),2) as d
iff_esti_to_actual
FROM `ecommerce.orders` o
  join `ecommerce.customer` c on  o.customer_id = c.customer_id
  join `ecommerce.order_items` ot on o.order_id = ot.order_id
WHERE DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) is not
null
GROUP BY c.customer_state
ORDER BY diff_esti_to_actual DESC
LIMIT 5
```

Row	customer_state	avg_freight_...	avg_time_to...	avg_diff_est...	diff_esti_to_...
1	AM	33.21	2.29	45.21	42.92
2	AP	34.16	3.15	45.62	42.47
3	RR	43.32	4.63	45.9	41.27
4	AC	40.07	2.84	40.7	37.86
5	RO	41.33	2.34	38.7	36.36

6. Payment type analysis:

```
--Month by month count of orders for different payment types
SELECT
  *,
  ROUND((x.count / SUM(x.count) OVER(PARTITION BY x.Year, x.Month))* 100,2) percent_
transaction
FROM
(SELECT
  payment_type,
  EXTRACT (MONTH FROM o.order_approved_at) as Month,
  EXTRACT (YEAR FROM o.order_approved_at) as Year,
  COUNT(payment_type) count
FROM `ecommerce.orders` o
  join `ecommerce.payments` p on o.order_id = p.order_id
WHERE EXTRACT (MONTH FROM o.order_approved_at) IS NOT NULL
GROUP BY
  Year,
  Month,
  payment_type
ORDER BY Year, Month
) as x
ORDER BY x.Year, x.Month
```

Output:

Row	payment_type	Month	Year	count	percent_tra...
1	credit_card	10	2016	253	74.85
2	UPI	10	2016	61	18.05
3	voucher	10	2016	22	6.51
4	debit_card	10	2016	2	0.59
5	credit_card	12	2016	1	100.0
6	credit_card	1	2017	580	71.69
7	UPI	1	2017	161	19.9
8	voucher	1	2017	59	7.29
9	debit_card	1	2017	9	1.11
10	credit_card	2	2017	1354	72.33

```sql
--Distribution of payment installments and count of orders:
SELECT
  payment_installments Duration,
  ROUND(COUNT(order_id)/(SELECT COUNT(order_id) FROM `ecommerce.payments`
) * 100,5) Distribution,
FROM `ecommerce.payments`
GROUP BY Duration
```

Output:

Row	Duration	Distribution
1	0	0.00193
2	1	50.58044
3	2	11.94867
4	3	10.06969
5	4	6.83249
6	5	5.04303
7	6	3.77337
8	7	1.56518
9	8	4.10835
10	9	0.61991