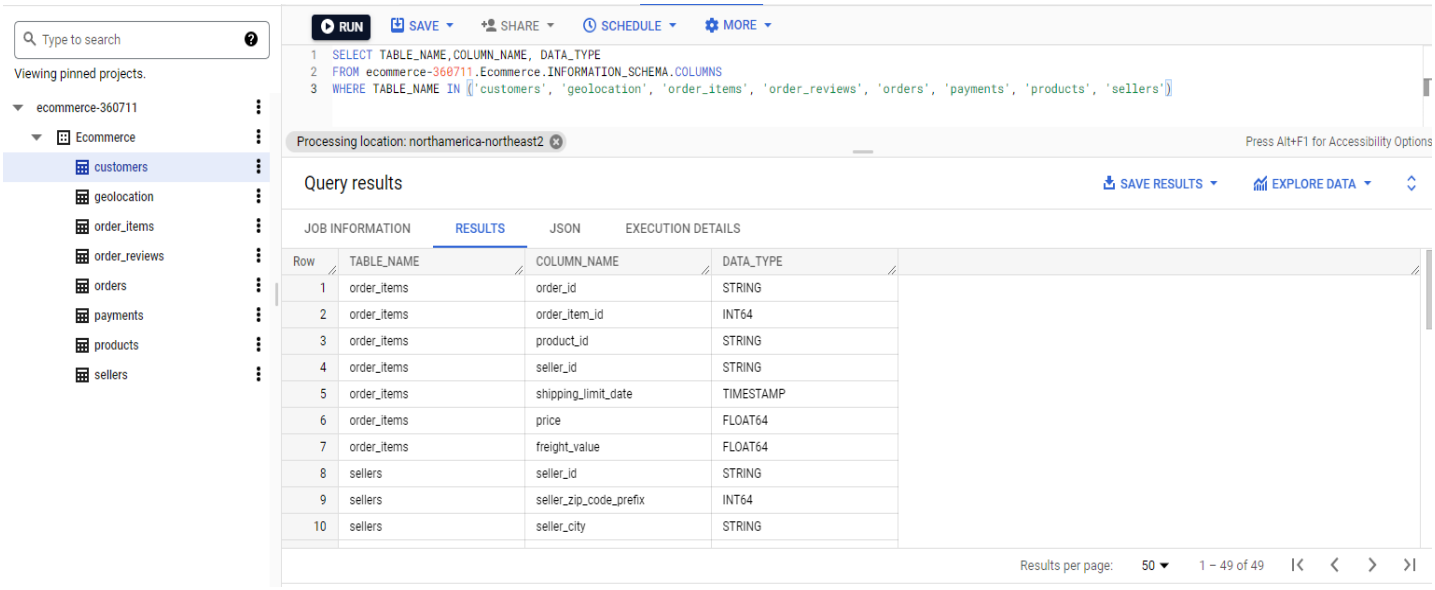
1. Initial exploration of dataset like checking the characteristics of data:

1. Data type of columns in a table:

Query used:

```
SELECT TABLE_NAME,COLUMN_NAME, DATA_TYPE
FROM ecommerce-360711.Ecommerce.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME IN ('customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'p
roducts', 'sellers')
```

Output:

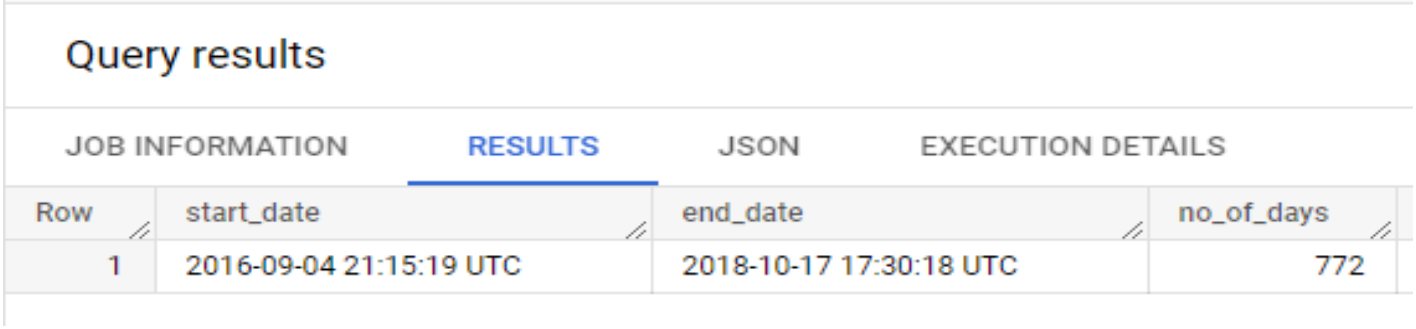


2. Time period for which the data is given:

Query used:

```
SELECT MIN(order_purchase_timestamp) AS start_date, MAX(order_purchase_timestamp) AS end_date, DATE_DIFF(
MAX(order_purchase_timestamp), MIN(order_purchase_timestamp), DAY) AS no_of_days FROM `Ecommerce.orders`
```

Output:



Conclusion: The time period for which data is given is from 2016-09-04 21:15:19 UTC to 2018-10-17 17:30:18 UTC i.e. data is given for 772 days.

3. Cities and States covered in the dataset:

Query used:

For city:

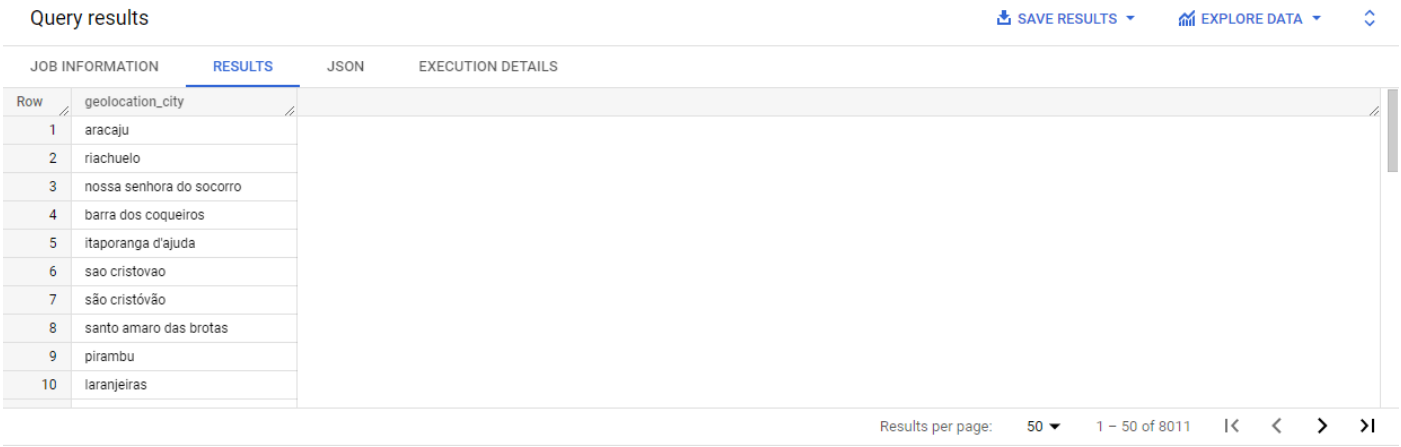
```
SELECT DISTINCT(geolocation_city)  from `Ecommerce.geolocation`
```

For state:

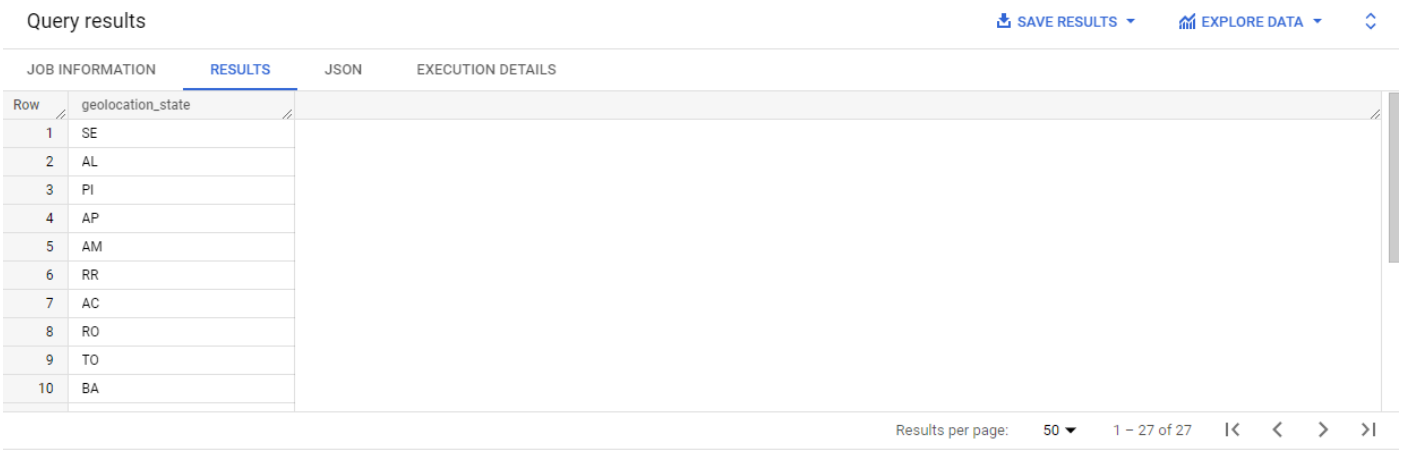
```
SELECT DISTINCT(geolocation_state)  from `Ecommerce.geolocation`
```

Output:

For city:



For state:



Conclusion: The dataset covers 8011 cities spanning across 27 states.

2. In-depth Exploration:

1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

Queries used:

For YoY trend increase:

```
SELECT order_year, COUNT(order_id) AS yearly_order_count FROM(
SELECT *,EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year
FROM `Ecommerce.orders`
)
GROUP BY order_year
ORDER BY order_year
```

Output:

Query results

JOB INFORMATION | RESULTS

Row	order_year	yearly_order...
1	2016	329
2	2017	45101
3	2018	54011

For MoM trend analysis:

```
SELECT order_year, order_month, COUNT(order_id) AS monthly_orders_count FROM(
SELECT *,EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year, EXTRACT(MONTH FROM order_purchase_tim
estamp) AS order_month
FROM `Ecommerce.orders`
)
GROUP BY order_year, order_month
ORDER BY order_year, order_month
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_year	order_month	monthly_orders_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780

Finding Month seasonality:

```
SELECT order_month, COUNT(order_id) AS monthly_orders_count FROM(
SELECT *,EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year, EXTRACT(MONTH FROM order_purchase_tim
estamp) AS order_month
FROM `Ecommerce.orders`
)
GROUP BY order_month
ORDER BY order_month
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_month	monthly_ord...
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

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

Queries used:

For hourly orders count:

```
SELECT hour_of_order, count(order_id) AS no_of_orders
FROM(
SELECT *,EXTRACT(HOUR FROM order_purchase_timestamp) AS hour_of_order
FROM `Ecommerce.orders`)
GROUP BY hour_of_order
ORDER BY no_of_orders DESC
```

Output:

Query results

JOB INFORMATION | RESULTS

Row	hour_of_order	no_of_orders
1	16	6675
2	11	6578
3	14	6569
4	13	6518
5	15	6454

Query results

JOB INFORMATION | RESULTS

Row	hour_of_order	no_of_orders
1	5	188
2	4	206
3	3	272
4	6	502
5	2	510

What time do Brazilians tend to buy in terms of time of the day?

```
SELECT time_of_the_day, SUM(no_of_orders) AS total_orders, SUM(no_of_orders)*100 / SUM(SUM(no_of_orders))
 OVER() AS order_pct FROM(
SELECT hour_of_order, count(order_id) AS no_of_orders,
CASE WHEN hour_of_order IN (4,5,6,7) THEN 'Dawn'
WHEN hour_of_order IN (8,9,10,11,12) THEN 'Morning'
WHEN hour_of_order IN (13,14,15,16,17) THEN 'Afternoon'
ELSE 'Night'
END AS time_of_the_day
 FROM(
SELECT *,EXTRACT(HOUR FROM order_purchase_timestamp) AS hour_of_order
FROM `Ecommerce.orders`)
GROUP BY hour_of_order
ORDER BY no_of_orders)
GROUP BY time_of_the_day
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	time_of_the_day	total_orders	order_pct
1	Afternoon	32366	32.5479430...
2	Morning	26502	26.6509789...
3	Night	38446	38.6621212...
4	Dawn	2127	2.13895676...

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

1. Get month on month orders by region, states.

Query used:

For states:

```
SELECT order_year,order_month, customer_state, COUNT(order_id) AS statewise_monthly_orders FROM(
SELECT o.order_id, o.order_purchase_timestamp, EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_yea
r, EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month, c.customer_id, c.customer_city, c.custo
mer_state
FROM `Ecommerce.orders` o
JOIN `Ecommerce.customers` c ON o.customer_id=c.customer_id
)
GROUP BY customer_state, order_year, order_month
ORDER BY order_year, order_month
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_year	order_month	customer_state	statewise_monthly_orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	SP	113
5	2016	10	RS	24
6	2016	10	RJ	56
7	2016	10	MT	3
8	2016	10	GO	9
9	2016	10	MG	40

For regions:

```
SELECT order_year,order_month, customer_city, COUNT(order_id) AS regionwise_monthly_orders FROM(
SELECT o.order_id, o.order_purchase_timestamp, EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_yea
r, EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month, c.customer_id, c.customer_city, c.custo
mer_state
FROM `Ecommerce.orders` o
JOIN `Ecommerce.customers` c ON o.customer_id=c.customer_id
)
GROUP BY customer_city, order_year, order_month
ORDER BY order_year, order_month
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_year	order_month	customer_city	regionwise_...
1	2016	9	boa vista	1
2	2016	9	passo fundo	1
3	2016	9	sao jose dos campos	1
4	2016	9	sao joaquim da barra	1
5	2016	10	itu	1
6	2016	10	porto alegre	7
7	2016	10	rio de janeiro	38
8	2016	10	cuiaba	2
9	2016	10	goiania	5

2. How are customers distributed in Brazil?

Query used:

For state-wise distribution:

```
SELECT customer_state, count(customer_id) AS statewse_customer_count, count(customer_id) * 100 / sum(coun
t(customer_id)) OVER() AS statewise_customer_percent
FROM `Ecommerce.customers`
GROUP BY customer_state
ORDER BY statewse_customer_count DESC
```

Output:

Top 5:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	statewse_customer_count	statewise_customer_percent
1	SP	41746	41.980671956235355
2	RJ	12852	12.924246538148248
3	MG	11635	11.700405265433774
4	RS	5466	5.496726702265665
5	PR	5045	5.0733600828632053

Bottom 5:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	statewse_customer_count	statewise_customer_percent
1	RR	46	0.046258585492905339
2	AP	68	0.0683822568155992
3	AC	81	0.081455335324463751
4	AM	148	0.14883197071630413
5	RO	253	0.25442222021097938

4. Impact on Economy:

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

Query used:

```
SELECT year_of_order, total_cost_of_order_yearwise, (total_cost_of_order_yearwise - LEAD(total_cost_of_order_yearwise) OVER (ORDER BY year_of_order DESC))*100/LEAD(total_cost_of_order_yearwise) OVER (ORDER BY year_of_order DESC) AS percentage_increase_YoY_from_jantosep FROM(
SELECT year_of_order, SUM(total_cost_of_order) AS total_cost_of_order_yearwise FROM(
SELECT order_id, EXTRACT(YEAR FROM shipping_limit_date) AS year_of_order, EXTRACT(MONTH FROM shipping_limit_date) AS month_of_order, ROUND(SUM(price+freight_value),2) AS total_cost_of_order FROM `Ecommerce.order_items`
WHERE EXTRACT(MONTH FROM shipping_limit_date) < 9 AND EXTRACT(YEAR FROM shipping_limit_date) IN (2017,2018)
GROUP BY order_id, shipping_limit_date)
GROUP BY year_of_order)
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year_of_order	total_cost_of_order_yearwise	percentage_increase_YoY_from_jantosep
1	2017	3455257.4899999527	*null*
2	2018	8769236.609999774	153.79401203468325

2. Mean & Sum of price and freight value by customer state

Query used:

```
SELECT DISTINCT c.customer_state,ROUND(SUM(oi.price) OVER (PARTITION BY c.customer_state),2) AS price_sum_by_state, ROUND(SUM(oi.freight_value) OVER (PARTITION BY c.customer_state),2) AS freight_value_sum_by_state, ROUND(AVG(oi.price) OVER (PARTITION BY c.customer_state),2) AS price_avg_by_state, ROUND(AVG(oi.freight_value) OVER (PARTITION BY c.customer_state),2) AS freight_value_avg_by_state, ROUND(SUM(oi.price+oi.freight_value) OVER (PARTITION BY c.customer_state),2) AS total_sum_by_state, ROUND(AVG(oi.price+oi.freight_value) OVER (PARTITION BY c.customer_state),2) AS total_avg_by_state
FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
```

Output:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	price_sum_by_state	freight_value_sum_by_state	price_avg_by_state	freight_value_avg_by_state	total_sum_by_state	total_avg_by_state
1	PB	115268.08	25719.73	191.48	42.72	140987.81	234.2
2	AL	80314.81	15914.59	180.89	35.84	96229.4	216.73
3	AC	15982.95	3686.75	173.73	40.07	19669.7	213.8
4	RO	46140.64	11417.38	165.97	41.07	57558.02	207.04
5	PA	178947.81	38699.3	165.69	35.83	217647.11	201.53
6	PI	86914.08	21218.2	160.36	39.15	108132.28	199.51
7	AP	13474.3	2788.5	164.32	34.01	16262.8	198.33
8	TO	49621.74	11732.68	157.53	37.25	61354.42	194.78
9	RR	7829.43	2235.19	150.57	42.98	10064.62	193.55

Results per page: 50 | 1 – 27 of 27

5 . Analysis on sales, freight and delivery time:

1. Computation of time to delivery and difference of time between estimated delivery

Query used:

```
SELECT c.customer_id, c.customer_state,o.order_id, TIMESTAMP_DIFF(o.order_delivered_customer_date, o.orde
r_purchase_timestamp, HOUR) AS time_to_delivery, DATE_DIFF(o.order_estimated_delivery_date, o.order_deliv
ered_customer_date, HOUR) AS diff_estimated_delivery, oi.price, oi.freight_value, SUM(oi.price+oi.freight
_value) OVER (PARTITION BY o.order_id) AS total_price FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
WHERE o.order_delivered_customer_date IS NOT NULL
ORDER BY time_to_delivery DESC
```

Output:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_id	customer_state	order_id	time_to_delivery	diff_estimated_delivery	price	freight_value	total_pri
1	75683a92331068e2d281b11a...	ES	ca07593549f1816d26a572e06...	5031	-4358	229.9	15.78	2
2	d306426abe5fca15e54b645e4...	RJ	1b3190b2dfa9d789e1f14c05b...	5000	-4535	144.99	17.26	1
3	7815125148cfa1e8c7fee1ff79...	PA	440d0d17af552815d15a9e41a...	4695	-3975	159.9	25.12	1
4	217906bc11a32c1e470eb7e08...	PI	2fb597c2f772eca01b1f5c561b...	4676	-3734	239.96	105.19	3
5	9cf2c3fa2632cee748e1a59ca9...	SE	285ab9426d6982034523a855f...	4671	-3998	429.9	27.75	4
6	1a8a4a30dc296976717f44e78...	PI	0f4519c5f1c541ddec9f21b3bd...	4657	-3878	231.27	27.88	259.150
7	cb2caaaead400c97350c37a3f...	SP	47b40429ed8cce3aee9199792...	4595	-4220	399.0	54.33	4
8	65b14237885b3972ebec28c0f...	SP	2fe324febf907e3ea3f2aa9650...	4556	-4025	39.9	16.05	

Results per page: 50 | 1 – 50 of 110196

2. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery.

1. Top 5 states with highest average freight value:

Query used:

```
SELECT customer_state, AVG(time_to_delivery) AS avg_time_to_delivery, AVG(diff_estimated_delivery) as avg
_diff_estimated_delivery, AVG(freight_value) AS avg_freight_value FROM(
```

```
SELECT c.customer_id, c.customer_state,o.order_id, TIMESTAMP_DIFF(o.order_delivered_customer_date, o.orde
r_purchase_timestamp, HOUR) AS time_to_delivery, DATE_DIFF(o.order_estimated_delivery_date, o.order_deliv
ered_customer_date, HOUR) AS diff_estimated_delivery, oi.price, oi.freight_value, SUM(oi.price+oi.freight
_value) OVER (PARTITION BY o.order_id) AS total_price FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
WHERE o.order_delivered_customer_date IS NOT NULL)
GROUP BY customer_state
ORDER BY avg_freight_value DESC
LIMIT 5
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to_delivery	avg_diff_estimated_delivery	avg_freight_value
1	PB	493.66894197952246	296.52730375426665	43.091689419795252
2	RR	676.97826086956513	422.43478260869563	43.088043478260865
3	RO	473.27106227106214	463.76923076923089	41.330549450549434
4	AC	496.67032967032958	487.59340659340654	40.047912087912081
5	PI	464.75143403441649	260.11663479923516	39.115086042064924

---

2. Top 5 states with lowest average freight value:

Query used:

```
SELECT customer_state, AVG(time_to_delivery) AS avg_time_to_delivery, AVG(diff_estimated_delivery) as avg
_diff_estimated_delivery, AVG(freight_value) AS avg_freight_value FROM(
SELECT c.customer_id, c.customer_state,o.order_id, TIMESTAMP_DIFF(o.order_delivered_customer_date, o.orde
r_purchase_timestamp, HOUR) AS time_to_delivery, DATE_DIFF(o.order_estimated_delivery_date, o.order_deliv
ered_customer_date, HOUR) AS diff_estimated_delivery, oi.price, oi.freight_value, SUM(oi.price+oi.freight
_value) OVER (PARTITION BY o.order_id) AS total_price FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
WHERE o.order_delivered_customer_date IS NOT NULL)
GROUP BY customer_state
ORDER BY avg_freight_value
LIMIT 5
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to_delivery	avg_diff_estimated_delivery	avg_freight_value
1	SP	208.86891458346574	251.89356846026288	15.114994078763218
2	PR	286.24039653035868	306.57408390865703	20.471816250663817
3	MG	287.11233258496662	302.91306030812041	20.6258372687155
4	RJ	363.06298600311231	271.04389933550141	20.909784391347358
5	DF	310.51847133757883	275.41953290870464	21.072161358811066

---

3. Top 5 states with highest average time to delivery:

Query used:

```
SELECT customer_state, AVG(time_to_delivery) AS avg_time_to_delivery, AVG(diff_estimated_delivery) as avg
_diff_estimated_delivery, AVG(freight_value) AS avg_freight_value FROM(
SELECT c.customer_id, c.customer_state,o.order_id, TIMESTAMP_DIFF(o.order_delivered_customer_date, o.orde
r_purchase_timestamp, HOUR) AS time_to_delivery, DATE_DIFF(o.order_estimated_delivery_date, o.order_deliv
ered_customer_date, HOUR) AS diff_estimated_delivery, oi.price, oi.freight_value, SUM(oi.price+oi.freight
_value) OVER (PARTITION BY o.order_id) AS total_price FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
WHERE o.order_delivered_customer_date IS NOT NULL)
GROUP BY customer_state
ORDER BY avg_time_to_delivery DESC
LIMIT 5
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to_delivery	avg_diff_est...	avg_freight_...
1	RR	676.97826086956513	422.434782...	43.0880434...
2	AP	676.45679012345681	426.012345...	34.1604938...
3	AM	632.8466257668706	460.969325...	33.3106134...
4	AL	587.22716627634679	193.133489...	35.8706557...
5	PA	569.60056925996287	325.289373...	35.6290132...

4. Top 5 states with lowest average time to delivery:

Query used:

```
SELECT customer_state, AVG(time_to_delivery) AS avg_time_to_delivery, AVG(diff_estimated_delivery) as avg
_diff_estimated_delivery, AVG(freight_value) AS avg_freight_value FROM(
SELECT c.customer_id, c.customer_state,o.order_id, TIMESTAMP_DIFF(o.order_delivered_customer_date, o.orde
r_purchase_timestamp, HOUR) AS time_to_delivery, DATE_DIFF(o.order_estimated_delivery_date, o.order_deliv
ered_customer_date, HOUR) AS diff_estimated_delivery, oi.price, oi.freight_value, SUM(oi.price+oi.freight
_value) OVER (PARTITION BY o.order_id) AS total_price FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
WHERE o.order_delivered_customer_date IS NOT NULL)
GROUP BY customer_state
ORDER BY avg_time_to_delivery
LIMIT 5
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to_delivery	avg_diff_est...	avg_freight_...
1	SP	208.86891458346574	251.893568...	15.1149940...
2	PR	286.24039653035868	306.574083...	20.4718162...
3	MG	287.11233258496662	302.913060...	20.6258372...
4	DF	310.51847133757883	275.419532...	21.0721613...
5	SC	359.52562225475691	260.550024...	21.5066276...

5. Top 5 states where delivery is really fast compared to estimated date:

Query used:

```
SELECT customer_state, AVG(time_to_delivery) AS avg_time_to_delivery, AVG(diff_estimated_delivery) as avg
_diff_estimated_delivery, AVG(freight_value) AS avg_freight_value FROM(
SELECT c.customer_id, c.customer_state,o.order_id, TIMESTAMP_DIFF(o.order_delivered_customer_date, o.orde
r_purchase_timestamp, HOUR) AS time_to_delivery, DATE_DIFF(o.order_estimated_delivery_date, o.order_deliv
ered_customer_date, HOUR) AS diff_estimated_delivery, oi.price, oi.freight_value, SUM(oi.price+oi.freight
_value) OVER (PARTITION BY o.order_id) AS total_price FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
WHERE o.order_delivered_customer_date IS NOT NULL)
GROUP BY customer_state
ORDER BY avg_diff_estimated_delivery
LIMIT 5
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to...	avg_diff_estimated_delivery	avg_freight_...
1	AL	587.227166...	193.13348946135841	35.8706557...
2	MA	519.058750...	221.11750000000012	38.4927125...
3	SE	514.722666...	223.46399999999988	36.5731733...
4	ES	375.080000...	238.414831460674	22.0289797...
5	BA	461.451262...	246.57344556068469	26.4875563...

6. Top 5 states where delivery is not so fast compared to estimated date:

Query used:

```
SELECT customer_state, AVG(time_to_delivery) AS avg_time_to_delivery, AVG(diff_estimated_delivery) as avg
_diff_estimated_delivery, AVG(freight_value) AS avg_freight_value FROM(
SELECT c.customer_id, c.customer_state,o.order_id, TIMESTAMP_DIFF(o.order_delivered_customer_date, o.orde
r_purchase_timestamp, HOUR) AS time_to_delivery, DATE_DIFF(o.order_estimated_delivery_date, o.order_deliv
ered_customer_date, HOUR) AS diff_estimated_delivery, oi.price, oi.freight_value, SUM(oi.price+oi.freight
_value) OVER (PARTITION BY o.order_id) AS total_price FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON c.customer_id = o.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
WHERE o.order_delivered_customer_date IS NOT NULL)
GROUP BY customer_state
ORDER BY avg_diff_estimated_delivery DESC
LIMIT 5
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to...	avg_diff_estimated_delivery	avg_freight_...
1	AC	496.670329...	487.59340659340654	40.0479120...
2	RO	473.271062...	463.76923076923089	41.3305494...
3	AM	632.846625...	460.96932515337443	33.3106134...
4	AP	676.456790...	426.01234567901241	34.1604938...
5	RR	676.978260...	422.43478260869563	43.0880434...

6. Payment type analysis:

1. Month over Month count of orders for different payment types

Query used:

```
SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year_of_purchase, EXTRACT(MONTH FROM o.order_purc
hase_timestamp) AS month_of_purchase, p.payment_type,  COUNT(o.order_id) AS order_count
FROM `Ecommerce.orders` o
JOIN `Ecommerce.payments`p ON o.order_id = p.order_id
GROUP BY year_of_purchase, month_of_purchase,p.payment_type
ORDER BY payment_type, year_of_purchase, month_of_purchase
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year_of_pur...	month_of_p...	payment_type	order_count
1	2016	10	UPI	63
2	2017	1	UPI	197
3	2017	2	UPI	398
4	2017	3	UPI	590
5	2017	4	UPI	496
6	2017	5	UPI	772
7	2017	6	UPI	707
8	2017	7	UPI	845
9	2017	8	UPI	938

2. Distribution of payment instalments and count of orders

Query used:

```
SELECT payment_installments, COUNT(order_id) AS orders_count, count(order_id) * 100 / sum(count(order_id)
) OVER() AS payment_installments_percent
FROM `Ecommerce.payments`
GROUP BY payment_installments
ORDER BY orders_count DESC
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_installments	orders_count	payment_in...
1	1	52546	50.5804439...
2	2	12413	11.9486745...
3	3	10461	10.0696917...
4	4	7098	6.83248945...
5	10	5328	5.12869876...
6	5	5239	5.04302793...
7	8	4268	4.10834953...
8	6	3920	3.77336695...
9	7	1626	1.56517721...

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_in...	orders_count	payment_in...
1	22	1	0.00096259...
2	23	1	0.00096259...
3	0	2	0.00192518...
4	21	3	0.00288778...
5	16	5	0.00481296...
6	17	8	0.00770074...
7	14	15	0.01443890...
8	13	16	0.01540149...
9	20	17	0.01636409...

3. Distribution of payment type in the data:

Query used:

```
SELECT payment_type, COUNT(order_id) AS orders_count, count(order_id) * 100 / sum(count(order_id)) OVER()
 AS payment_type_percent
FROM `Ecommerce.payments`
GROUP BY payment_type
ORDER BY orders_count DESC
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	orders_count	payment_type_percent
1	credit_card	76795	73.922376451109869
2	UPI	19784	19.043952024334367
3	voucher	5775	5.5589781106212577
4	debit_card	1529	1.4718056330978189
5	not_defined	3	0.0028877808366863677

7 . Product analysis:

1. Top 5 product categories:

Query used:

```
SELECT p.product_category, COUNT(oi.order_id) AS product_categorywise_orders
FROM `Ecommerce.products` p
JOIN `Ecommerce.order_items`oi ON p.product_id = oi.product_id
GROUP BY p.product_category
ORDER BY COUNT(oi.order_id) DESC
LIMIT 5
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	product_category	product_categorywise_orders
1	bed table bath	11115
2	HEALTH BEAUTY	9670
3	sport leisure	8641
4	Furniture Decoration	8334
5	computer accessories	7827

---

2. Best reviewed product categories with review count > 50:

Query used:

```
SELECT p.product_category, AVG(r.review_score) AS avg_review_score, COUNT(r.review_id) AS review_count
FROM `Ecommerce.order_reviews` r
JOIN `Ecommerce.order_items`oi ON r.order_id = oi.order_id
JOIN `Ecommerce.products` p ON oi.product_id = p.product_id
GROUP BY p.product_category
HAVING review_count > 50
ORDER BY avg_review_score DESC
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	product_category	avg_review_score	review_count
1	General Interest Books	4.4462659380692147	549
2	Construction Tools Tools	4.4444444444444429	99
3	Imported books	4.4	60
4	technical books	4.3684210526315788	266
5	Drink foods	4.3154121863799286	279
6	Bags Accessories	4.3152573529411793	1088
7	HOUSE PASTALS OVEN AND C...	4.3026315789473664	76
8	Fashion Calcados	4.2337164750957861	261
9	foods	4.21818181818182	495

---

3. Customer-wise favourite product categories:

Query used:

```
SELECT c.customer_id, p.product_category, COUNT(o.order_id) AS no_of_orders
FROM `Ecommerce.customers` c
JOIN `Ecommerce.orders` o ON o.customer_id = c.customer_id
JOIN `Ecommerce.order_items`oi ON o.order_id = oi.order_id
JOIN `Ecommerce.products`p ON oi.product_id = p.product_id
GROUP BY c.customer_id, p.product_category
ORDER BY no_of_orders DESC
```

Output:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_id	product_category	no_of_orders
1	fc3d1daec319d62d49bfb5e1f8...	HEALTH BEAUTY	21
2	bd5d39761aa56689a265d95d...	automotive	20
3	be1b70680b9f9694d8c70f41fa...	computer accessories	20
4	adb32467ecc74b53576d9d13a...	Garden tools	15
5	10de381f8a8d23fff822753305...	Furniture Decoration	15
6	d5f2b3f597c7ccafbb5cac0bcc...	Garden tools	14
7	a7693fba2ff9583c78751f2b66...	telephony	14
8	7d321bd4e8ba1caf74c4c1aab...	telephony	13
9	3b54b5978e9ace64a63f90d17...	housewares	12

---

8. Actionable Insights:

1. The e-commerce sail is increasing in Brazil year on year. The total orders made in 2108 are significantly higher than that in 2016 and 2017.
2. There is some seasonality in orders made when we talk in terms of months. May, July and August have higher sales, while sales drop down significantly in the month of September, October and December.
3. Most number of orders are made at 4PM, followed by 11AM, 2PM, 1PM. The least orders are made at 5AM, followed by 4AM, 3AM, 6AM.
4. Most Brazilians tend to order at night i.e., 5PM to 4AM (~38% of the orders are made at night). The least orders are made in dawn time (2.1%).
5. MoM sales is increasing in most parts of Brazil.
6. Top 5 states with most customers are SP, RJ, MG, RS, PR. While bottom 5 states with least customers are RR, AP, AC, AM, RO respectively.
7. Total cost of order has significantly increased from about 8M in 2017 to 34M in 2018 which is nearly 154% rise.
8. PB has highest average order price value followed by AL, AC, RO, PA (all above 200) while SP has lowest average order price (~124).
9. 5 states with highest average freight value are PB, RR, RO, AC, PI whereas that with lowest are SP, PR, MG, RJ, DF.
10. States with maximum average delivery time are RR, AP, AM, AL, PA while the fastest deliveries take place in SP, PR, MG, DF, SC.
11. States where delivery is fast compared to estimated delivery time are AL, MA, SE, ES, BA. States where delivery is not so fast compared to estimated delivery time are AC, RO, AM, AP, RR.
12. 73% of the order payments are made through credit card, while only 1.5% are made through debit card.
13. 50% of orders are paid in 1 payment instalments, while very few are made in instalments>15 (less than 1% each).
14. Bed table bath is the most selling product category, followed by health beauty, sport leisure, furniture decoration, computer accessories.
15. General interest books have best reviews where review count is > 50. The next few highly rated product categories are construction tools, imported books, drink foods.
16. Customer-wise favourite product categories are found so that we can recommend more such products to those customers.

9. Recommendations:

1. Can we have sale in the month of September, October and December in order to increase sales in those months.
2. Is it possible to have offers during dawn time to increase the sales in those hours.
3. The customers in RR, AP, AC states are lower. Can we have more advertisements and other type of marketing done in order to increase sales in those areas.
4. Does including more products in recommendations increase the average order price in a state. It is always good to increase the average order price in order to become profitable. Can average order price be increased in those states where it is lagging.
5. States with higher delivery time are mostly the ones where there are least customers. Can we increase the delivery speed in order to capture the market In those states which would also act as a promotion in that state.

6. Can we use delivery time as the parameter to market our products in order to increase sales in the states where delivery time is low.
7. Most of the payments are made through credit cards while least through debit cards. Can we have offers on the orders where payments are made through debit cards.
8. We have very few orders where number of instalments are greater than 15. So can we reduce the EMI interest rate so that more people buy on higher instalments.
9. Can we have offers on least selling product categories, and keep the most sold products in front page of the website/app.
10. Can we try having the most reviewed items for a particular category on the top when someone searches for that product category.
11. Can we send recommendations to the customers that have previously bought many items of the same product category.