1st Question

1. ***<u>Data type of columns in a table</u>***

Ans. **<u>CUSTOMERS TABLE</u>**

Customer_id → STRING

Customer_unique_id → STRING

Customer_zip_code_prefix → INTEGER

Customer_city → STRING

Customer_state → STRING

<u>ORDER_ITEMS TABLE</u>

order_id → STRING

order_item_id → INTEGER

product_id → STRING

seller_id → STRING

shipping_limit_date → TIMESTAMP

price → FLOAT

freight_value → FLOAT

<u>ORDER_REVIEWS TABLE</u>

review_id → STRING

order_id → STRING

review_score → INTEGER

review_comment_title → STRING

review_creation_date → TIMESTAMP

review_answer_timestamp → TIMESTAMP

<u>ORDERS TABLE</u>

order_id → STRING

customer_id → STRING

order_status → STRING

order_purchase_timestamp → TIMESTAMP

order_approved_at → TIMESTAMP

order_delivered_carrier_date → TIMESTAMP

order_delivered_customer_date → TIMESTAMP

order_estimated_delivery_date → TIMESTAMP

PAYMENTS

order_id → STRING

payment_sequential → INTEGER

payment_type → STRING

payment_installments → INTEGER

payment_value → FLOAT

PRODUCTS

product_id → STRING

product_category →STRING

product_name_length → INTEGER

product_description_length → INTEGER

product_photos_qty → INTEGER

product_weight_g → INTEGER

product_length_cm → INTEGER

product_height_cm → INTEGER

product_width_cm → INTEGER

SELLERS

seller_id → STRING

seller_zip_code_prefix → INTEGER

seller_city → STRING

seller_state → STRING

2. *Time period for which the data is given*

Ans. SELECT min(order_purchase_timestamp) as 'start_date', max(order_purchase_timestamp) as 'end_date' FROM orders;

Result Grid | Filter Rows:

start_date	end_date
2016-09-04 21:15:19	2018-10-17 17:30:18

3. ***<u>Cities and States covered in the dataset</u>***

Ans. SELECT DISTINCT customer_city,customer_state FROM customers;

Result Grid | Filter Rows:

customer_city	customer_state
franca	SP
sao bernardo do campo	SP
sao paulo	SP
mogi das cruzes	SP
campinas	SP
jaragua do sul	SC
timoteo	MG
curitiba	PR
belo horizonte	MG
montes claros	MG
rio de janeiro	RJ
lencois paulista	SP
caxias do sul	RS

2nd Question

1. ***<u>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?</u>***

Ans. SELECT COUNT(1) as 'Total_No_of_transactions_done_In_This_Year',YEAR(order_purchase_timestamp) as 'Transaction_Year' FROM ORDERS GROUP BY YEAR(order_purchase_timestamp) ORDER BY YEAR(order_purchase_timestamp);

Result Grid | Filter Rows: | Export:

Total_No_of_transactions_done_In_This_Year	Transaction_Year
329	2016
45101	2017
54011	2018

SELECT COUNT(1) as 'Total_No_of_transactions_done_In_This_Year',YEAR(order_purchase_timestamp) as 'Transaction_Year',MONTH(order_purchase_timestamp) as 'Transaction_Month' FROM ORDERS GROUP BY YEAR(order_purchase_timestamp),MONTH(order_purchase_timestamp) ORDER BY YEAR(order_purchase_timestamp),MONTH(order_purchase_timestamp);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_No_of_transactions_done_In_This_Year	Transaction_Year	Transaction_Month
4	2016	9
324	2016	10
1	2016	12
800	2017	1
1780	2017	2
2682	2017	3
2404	2017	4
3700	2017	5
3245	2017	6
4026	2017	7
4331	2017	8
4285	2017	9
4631	2017	10

2. ***<u>What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?</u>***

Ans. WITH cte AS (SELECT

CASE

WHEN TIME(order_purchase_timestamp) BETWEEN '00:00:00.000000' AND '05:00:00.000000' THEN 'Dawn'

WHEN TIME(order_purchase_timestamp) BETWEEN '05:00:00.000001' AND '12:00:00.000000' THEN 'Morning'

WHEN TIME(order_purchase_timestamp) BETWEEN '12:00:00.000001' AND '19:00:00.000000' THEN 'Afternoon'

WHEN TIME(order_purchase_timestamp) BETWEEN '19:00:00.000001' AND '23:59:59.000000' THEN 'Night'

END AS 'Timing'

FROM orders)

SELECT count(*) as 'Total_transactions_done_in_that_time',Timing FROM cte GROUP BY Timing ORDER BY count(*) DESC;

Result Grid | Filter Rows: | Export: | Wr

Total_transactions_done_in_that_time	Timing
44130	Afternoon
28331	Night
22428	Morning
4552	Dawn

3rd Question

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

Ans. WITH m_o_m AS (SELECT customer_state,MONTH(order_purchase_timestamp) as mon FROM orders as o JOIN customers as c ON o.customer_id=c.customer_id)

SELECT customer_state,count(*) as 'no_of_transactions_in_that_month',mon,(count(*)-LAG(count(*),1,0) OVER(PARTITION BY customer_state ORDER BY mon))*100/LAG(count(*),1,0) OVER(PARTITION BY customer_state ORDER BY mon) as 'mon_on_mon_growth' FROM m_o_m GROUP BY mon;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_state	no_of_transactions_in_that_month	mon	mon_on_mon_growth
BA	9412	6	NULL
DF	9893	3	NULL
GO	10843	8	NULL
MG	10318	7	NULL
PR	7544	11	NULL
RJ	4305	9	NULL
RJ	5674	12	31.8002
RS	9343	4	NULL
SP	8069	1	NULL
SP	8508	2	5.4406
SP	10573	5	24.2713
SP	4959	10	-53.0975

2. ***How are customers distributed in Brazil***

Ans. SELECT customer_state,count(*) as 'No_of_customers_in_that_state' FROM customers GROUP BY customer_state ORDER BY count(*) DESC;

Result Grid | Filter Rows: | Export:

customer_state	No_of_customers_in_that_state
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020
PE	1652
CE	1336
PA	975

SELECT customer_city,count(*) as 'No_of_customers_in_that_city' FROM customers GROUP BY customer_city ORDER BY count(*) DESC;

Result Grid | Filter Rows: | Export:

customer_city	No_of_customers_in_that_city
sao paulo	15540
rio de janeiro	6882
belo horizonte	2773
brasilia	2131
curitiba	1521
campinas	1444
porto alegre	1379
salvador	1245
guarulhos	1189
sao bernardo do campo	938
niteroi	849
santo andre	797
osasco	746

4th Question

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

Ans. SELECT order_item_id,YEAR(order_purchase_timestamp) as 'Year',price,(price-LAG(price,1) OVER(PARTITION BY order_item_id ORDER BY YEAR(order_purchase_timestamp)))*100/LAG(price,1) OVER(PARTITION BY order_item_id ORDER BY YEAR(order_purchase_timestamp)) as 'percent_increase_in_price' FROM order_items as oi JOIN orders as o on oi.order_id=o.order_id WHERE MONTH(order_purchase_timestamp) BETWEEN 1 AND 8 GROUP BY YEAR(order_purchase_timestamp),order_item_id ORDER BY order_item_id,YEAR(order_purchase_timestamp);

Result Grid | Filter Rows: | Export:

order_item_id	Year	price	percent_increase_in_price
1	2017	109.9	NULL
1	2018	249.9	127.38853503184713
2	2017	179.9	NULL
2	2018	159	-11.61756531406337
3	2017	99	NULL
3	2018	39.99	-59.60606060606061
4	2017	99	NULL
4	2018	45	-54.54545454545455
5	2017	112.99	NULL
5	2018	45	-60.173466678467115
6	2017	112.99	NULL
6	2018	10	-91.14965926188158
7	2017	9.6	NULL

2. ***<u>Mean & Sum of price and freight value by customer state</u>***

Ans. SELECT AVG(price) as 'mean_price_state_wise',sum(price) as 'sum_of_price_state_wise',AVG(freight_value) as 'mean_freight_vale_state_wise',sum(freight_value) as 'sum_of_freight_value_state_wise',customer_state FROM order_items as oi JOIN orders as o ON oi.order_id=o.order_id JOIN customers as c ON o.customer_id=c.customer_id GROUP BY customer_state;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

mean_price_state_wise	sum_of_price_state_wise	mean_freight_vale_state_wise	sum_of_freight_value_state_wise	customer_state
109.65362915976209	5202955.050001551	15.147275390418875	718723.0699999852	SP
120.3374530874133	750304.0200000219	21.735804330393318	135522.74000000235	RS
164.32073170731692	13474.299999999988	34.00609756097561	2788.5	AP
124.65357758620901	520553.3400000088	21.470368773946397	89660.26000000015	SC
134.60120821268953	511349.9900000075	26.363958936562	100156.67999999903	BA
142.6283760683758	116812.63999999977	23.374884004884	19144.029999999995	MS
125.11781809450687	1824092.6699998158	20.960923931682416	305589.30999999796	RJ
160.35808118081195	86914.08000000007	39.14797047970483	21218.20000000002	PI
120.74857414882183	1585308.0299998817	20.630166806306864	270853.4600000028	MG
121.91370124113334	275037.3099999968	22.058776595744593	49764.5999999998	ES
165.97352517985644	46140.64000000009	41.06971223021582	11417.379999999997	RO
119.00413937282565	683083.7600000192	20.531651567944433	117851.68000000104	PR
125.77054862842789	302603.9399999975	21.041354945968347	50625.49999999984	DF
148.29718483412253	156453.52999999927	28.166284360189632	29715.430000000062	MT
126.27173167595221	294591.9499999965	22.766815259322733	53114.97999999994	GO
156.96593572778832	83034.98000000003	35.6523629489603	18860.1	RN
145.50832225913436	262788.02999999665	32.91786267995565	59449.659999999894	PE

5th Question

1. ***<u>Calculate days between purchasing, delivering and estimated delivery</u>***

Ans. SELECT order_id,DATEDIFF(order_delivered_customer_date,order_purchase_timestamp) as 'days_different_between_purchase_delhivery' FROM orders;

Result Grid | Filter Rows: | Export:

order_id	days_different_betw
e481f51cbdc54678b7cc49136f2d6af7	8
53cdb2fc8bc7dce0b6741e2150273451	14
47770eb9100c2d0c44946d9cf07ec65d	9
949d5b44dbf5de918fe9c16f97b45f8a	14
ad21c59c0840e6cb83a9ceb5573f8159	3
a4591c265e18cb1dcee52889e2d8acc3	17
136cce7faa42fdb2cefd53fdc79a6098	NULL
6514b8ad8028c9f2cc2374ded245783f	10
76c6e866289321a7c93b82b54852dc33	10
e69bfb5eb88e0ed6a785585b27e16dbf	18
e6ce16cb79ec1d90b1da9085a6118aeb	13
34513ce0c4fab462a55830c0989c7edb	6
82566a660a982b15fb86e904c8d32918	12

SELECT order_id,DATEDIFF(order_estimated_delivery_date,order_purchase_timestamp) as 'days_different_between_purchase_estimated_delhivery' FROM orders;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rov

order_id	days_different_between_purchase_estimated_delhivery
e481f51cbdc54678b7cc49136f2d6af7	16
53cdb2fc8bc7dce0b6741e2150273451	20
47770eb9100c2d0c44946d9cf07ec65d	27
949d5b44dbf5de918fe9c16f97b45f8a	27
ad21c59c0840e6cb83a9ceb5573f8159	13
a4591c265e18cb1dcee52889e2d8acc3	23
136cce7faa42fdb2cefd53fdc79a6098	28
6514b8ad8028c9f2cc2374ded245783f	22
76c6e866289321a7c93b82b54852dc33	42
e69bfb5eb88e0ed6a785585b27e16dbf	25
e6ce16cb79ec1d90b1da9085a6118aeb	22
34513ce0c4fab462a55830c0989c7edb	26
82566a660a982b15fb86e904c8d32918	41

SELECT order_id,DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'days_different_between_estimated_delhivery_delhivery' FROM orders;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch

order_id	days_different_between_estimated_delhivery_delhivery
e481f51cbdc54678b7cc49136f2d6af7	8
53cdb2fc8bc7dce0b6741e2150273451	6
47770eb9100c2d0c44946d9cf07ec65d	18
949d5b44dbf5de918fe9c16f97b45f8a	13
ad21c59c0840e6cb83a9ceb5573f8159	10
a4591c265e18cb1dcee52889e2d8acc3	6
136cce7faa42fdb2cefd53fdc79a6098	NULL
6514b8ad8028c9f2cc2374ded245783f	12
76c6e866289321a7c93b82b54852dc33	32
e69bfb5eb88e0ed6a785585b27e16dbf	7
e6ce16cb79ec1d90b1da9085a6118aeb	9
34513ce0c4fab462a55830c0989c7edb	20
82566a660a982b15fb86e904c8d32918	29

2. ***Create columns:***

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

Ans. WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery',

DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders)

SELECT time_to_delivery,diff_estimated_delivery FROM create_columns;

Result Grid | Filter Rows:

time_to_delivery	diff_estimated_delivery
-8	8
-14	6
-9	18
-14	13
-3	10
-17	6
NULL	NULL
-10	12
-10	32
-18	7
-13	9
-6	20
-12	29

Result 80

3. ***<u>Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery</u>***

Ans. WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery',

DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders),group_by_cte AS(

SELECT AVG(freight_value) as 'avg_freight_value',AVG(time_to_delivery) as 'avg_time_to_delivery',AVG(diff_estimated_delivery) as 'avg_diff_estimated_delivery' FROM create_columns as cc JOIN order_items as oi ON cc.order_id=oi.order_id JOIN customers as c ON cc.customer_id=c.customer_id

GROUP BY c.customer_state)

SELECT * FROM group_by_cte;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
22.76681525932273	-15.3355	12.2942
20.6301668063069	-11.9207	13.3426
15.147275390418883	-8.6623	11.2079
23.374884004884002	-15.4599	11.2293
20.960923931682412	-15.0748	12.0148
38.25700242718448	-21.5900	9.9063
26.363958936562	-19.1925	10.9826
32.91786267995565	-18.2245	13.4502
21.735804330393332	-15.1345	14.1342
20.531651567944436	-11.8931	13.4861
21.470368773946408	-14.9502	11.5727
22.05877659574458	-15.5874	10.6463
42.723803986711	-20.5461	13.0375

Result 81

4. Sort the data to get the following:

1. ***<u>Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5</u>***

Ans. WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery', DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders),group_by_cte AS(SELECT c.customer_state,AVG(freight_value) as 'avg_freight_value',AVG(time_to_delivery) as 'avg_time_to_delivery',AVG(diff_estimated_delivery) as 'avg_diff_estimated_delivery' FROM create_columns as cc JOIN order_items as oi ON cc.order_id=oi.order_id JOIN customers as c ON cc.customer_id=c.customer_id GROUP BY c.customer_state)

SELECT customer_state FROM group_by_cte ORDER BY avg_freight_value DESC LIMIT 5;

Result Grid | Filter Rows:

customer_state
RR
PB
RO
AC
PI

WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery',DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders),group_by_cte AS(SELECT c.customer_state,AVG(freight_value) as 'avg_freight_value',AVG(time_to_delivery) as 'avg_time_to_delivery',AVG(diff_estimated_delivery) as 'avg_diff_estimated_delivery' FROM create_columns as cc JOIN order_items as oi ON cc.order_id=oi.order_id JOIN customers as c ON cc.customer_id=c.customer_id GROUP BY c.customer_state)

SELECT customer_state FROM group_by_cte ORDER BY avg_freight_value LIMIT 5;

Result Grid | Filter Rows:

customer_state
SP
PR
MG
RJ
DF

2. ***<u>Top 5 states with highest/lowest average time to delivery</u>***

Ans. WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery',

DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders),group_by_cte AS(SELECT c.customer_state,AVG(freight_value) as 'avg_freight_value',AVG(time_to_delivery) as 'avg_time_to_delivery',AVG(diff_estimated_delivery) as 'avg_diff_estimated_delivery' FROM create_columns as cc JOIN order_items as oi ON cc.order_id=oi.order_id JOIN customers as c ON cc.customer_id=c.customer_id GROUP BY c.customer_state)

SELECT customer_state FROM group_by_cte ORDER BY avg_time_to_delivery;

Result Grid | Filter Rows:

customer_state
AP
RR
AM
AL
PA
MA
SE
CE
AC
PB
RO
PI
RN

WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery',

DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders),group_by_cte AS(SELECT c.customer_state,AVG(freight_value) as 'avg_freight_value',AVG(time_to_delivery) as 'avg_time_to_delivery',AVG(diff_estimated_delivery) as 'avg_diff_estimated_delivery' FROM create_columns as cc JOIN order_items as oi ON cc.order_id=oi.order_id JOIN customers as c ON cc.customer_id=c.customer_id GROUP BY c.customer_state)

SELECT customer_state FROM group_by_cte ORDER BY avg_time_to_delivery DESC;

Result Grid | Filter Rows:

customer_state
SP
PR
MG
DF
SC
RJ
RS
GO
MS
ES
TO
MT
PE

3. ***<u>Top 5 states where delivery is really fast/ not so fast compared to estimated date</u>***

Ans. WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery',

DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders),group_by_cte AS(SELECT c.customer_state,AVG(freight_value) as 'avg_freight_value',AVG(time_to_delivery) as 'avg_time_to_delivery',AVG(diff_estimated_delivery) as 'avg_diff_estimated_delivery' FROM create_columns as cc JOIN order_items as oi ON cc.order_id=oi.order_id JOIN customers as c ON cc.customer_id=c.customer_id GROUP BY c.customer_state)

SELECT customer_state FROM group_by_cte ORDER BY avg_diff_estimated_delivery LIMIT 5;

Result Grid | Filter Rows

customer_state
AL
MA
SE
ES
BA

WITH create_columns AS (SELECT *,DATEDIFF(order_purchase_timestamp,order_delivered_customer_date) as 'time_to_delivery',

DATEDIFF(order_estimated_delivery_date,order_delivered_customer_date) as 'diff_estimated_delivery' FROM orders),group_by_cte AS(SELECT c.customer_state,AVG(freight_value) as 'avg_freight_value',AVG(time_to_delivery) as 'avg_time_to_delivery',AVG(diff_estimated_delivery) as 'avg_diff_estimated_delivery' FROM create_columns as cc JOIN order_items as oi ON cc.order_id=oi.order_id JOIN customers as c ON cc.customer_id=c.customer_id GROUP BY c.customer_state)

SELECT customer_state FROM group_by_cte ORDER BY avg_diff_estimated_delivery DESC LIMIT 5;

Result Grid | Filter Rows:

customer_state
AC
RO
AM
AP
RR

6th Question

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

Ans. SELECT MONTH(order_purchase_timestamp) as 'Month',count(o.order_id) as 'no_of_orders',(count(o.order_id)-LAG(count(o.order_id),1) OVER(PARTITION BY payment_type ORDER BY MONTH(order_purchase_timestamp)))*100/LAG(count(o.order_id),1) OVER(PARTITION BY payment_type ORDER BY MONTH(order_purchase_timestamp)) as 'percent_increase',payment_type FROM orders as o LEFT JOIN payments as p on o.order_id=p.order_id GROUP BY MONTH(order_purchase_timestamp),payment_type ORDER BY MONTH(order_purchase_timestamp);

Result Grid | Filter Rows: | Export:

Month	no_of_orders	percent_increase	payment_type
1	6103	NULL	credit_card
1	1715	NULL	UPI
1	118	NULL	debit_card
1	477	NULL	voucher
2	6609	8.2910	credit_card
2	1723	0.4665	UPI
2	82	-30.5085	debit_card
2	424	-11.1111	voucher
3	7707	16.6137	credit_card
3	1942	12.7104	UPI
3	109	32.9268	debit_card
3	591	39.3868	voucher
4	124	13.7615	debit_card

2. ***Distribution of payment installments and count of orders***

Ans. SELECT payment_installments,count(*) as 'Total_no_of_transactions_for_no_installments' FROM payments GROUP BY payment_installments ORDER BY payment_installments;

Result Grid | Filter Rows: | Export: | Wrap

payment_installments	Total_no_of_transactions_for_no_installments
0	2
1	52546
2	12413
3	10461
4	7098
5	5239
6	3920
7	1626
8	4268
9	644
10	5328
11	23
12	133

7Th Question

ACTIONABLE INSIGHTS

More number of customers are located in the city sao Paulo,rio de janerio, belo horizonte etc

More number of customers are located in the state SP,RJ,MG,RS,PR.

Most of the item prices have decreases in 2018 when compared to 2017.

More number of orders are from the states RR,PB,RO,AC,PI and in the similar way less number of orders are from the states SP,PR,MG,RJ,DF.

Average Time of delivery is least in the states AP,RR,AM,Al,PA and highest in the states SP,PR,MG,DF,SC.

The delivery time is less in the states AL,MA,SE,ES,BA and in the similar way the delivery time is high in the states AC,RO,AM,Ap,RR.

Most of the people are preferring to pay in single instalment followed by two and three.

8Th Question

RECOMMENDATION

As more number of customers are from the city sao Paulo,rio de janerio, belo horizonte etc we should try to attract more customers by giving discount to the people living in that particular state and city.

As most number of orders are from the states RR,PB,RO,AC,PI the ecommerce should find a way to deliver the goods faster and also more number of delivery persons should be present at any point of time.

As we cannot see any trend in which card the people are using the most, we have to incentivise the digital payment so that a greater number of people will be attracted towards the digital transactions.

As the delivery of some of the states is too high we should fine some alternative ways to deliver the good in quick manner.