Retail Store CS

Q1.1 Data type of all columns in the "customers" table.

Ans:

```
SELECT
  column_name, data_type
FROM
  `cs1.customers.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = `customers`
ORDER BY
  ordinal_position
```

Screenshot:

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

Insights: NA

Recommendation: NA

Q1.2 Get the time range between which the orders were

placed.

Ans:

```
SELECT
  MIN(order_purchase_timestamp) AS
`min_purchase_time`,
  MAX(order_purchase_timestamp) AS
`max_purchase_time`
FROM
  `cs1.orders`
```

Screenshot:

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

Insights:
1. For the current data set the orders are placed from 4th September, 2016 from 9:15 PM to 17th November 2018 until 5:30 PM.

Recommendation:
1. Its better to look at peak or min sales times to get better insights.

Q1.3 Count the Cities & States of customers who ordered during the given period.

Ans:

```
SELECT DISTINCT
  COUNT(DISTINCT customer_city) AS `city_count`,
  COUNT(DISTINCT customer_state) AS `state_count`
FROM
  `cs1.customers`
```

Screenshot:

Row	city_count	state_count
1	4119	27

Insights:
1. There are 27 states in total and it is not surprising that all 27 states made an order in that period. However there are a total of 8011 cities and approx half of them only seems to have made an order. So its safe to conclude we do not have any presence in those cities.
2. I cannot conclude on customer engagement in this but since the 4119 cities did place at least one order it is likely that they will place again so we can consider starting loyalty programs in these cities if its not already in place.

```
SELECT DISTINCT
  COUNT(DISTINCT geolocation_state) AS `state_count`,
  COUNT(DISTINCT geolocation_city) AS `city_count`
FROM
  `cs1.geolocation`
```

Recommendation:
1. We can try setting up new stores and expand our reach in the cities which did not make any orders through targeted marketing.

Q2.1 Is there a growing trend in the no. of orders placed over the past years?

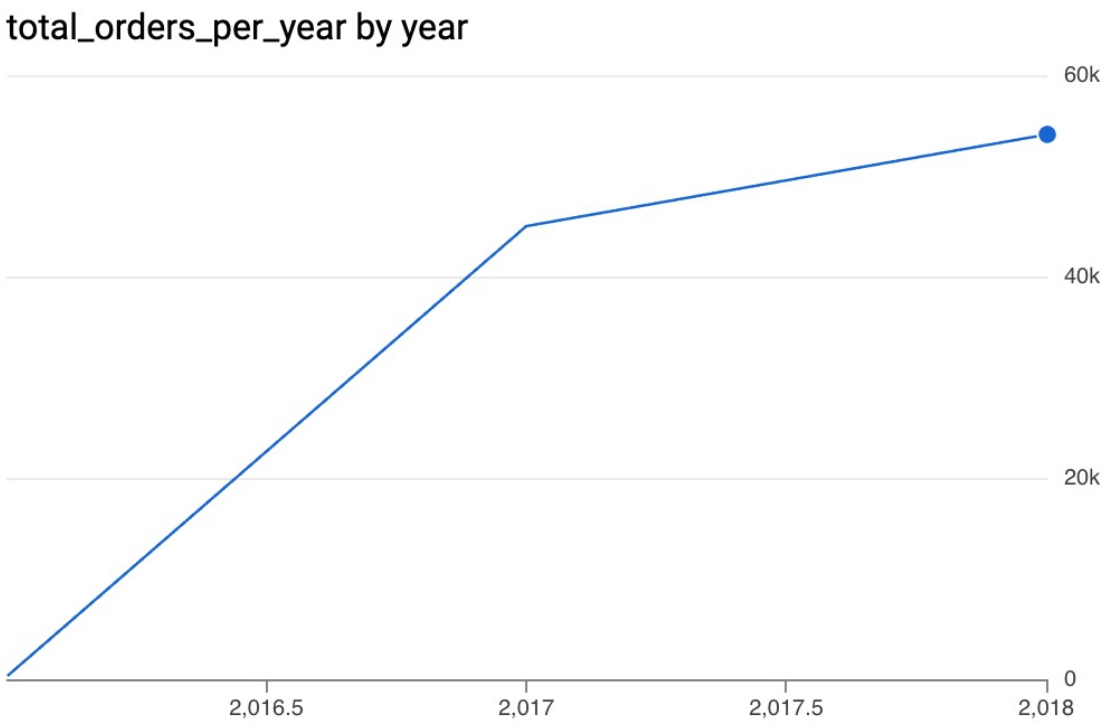
Ans:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS `year`,
  COUNT(order_id) AS `total_orders_per_year`
FROM
  `cs1.orders`
GROUP BY
  year
```

ORDER BY
year

Screenshot:

Row	year	total_orders_per_year
1	2016	329
2	2017	45101
3	2018	54011



Insights:
1. Yes there is a growing trend in the number of orders placed. A great spike from in sales from 2016 to 2017. Probably due to a marketing campaign or due to new store opening in new locations.
2. From the given data we can say there is a growth in the orders placed from 2017 to 2018. However the data is insufficient to predict continuous growth in the future. Sample size is too small.

Recommendation:
1. We should emulate the growth strategy of 2016-2017 to see if the spike is reproducible assuming its not a new store opening.

Q2.2 Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

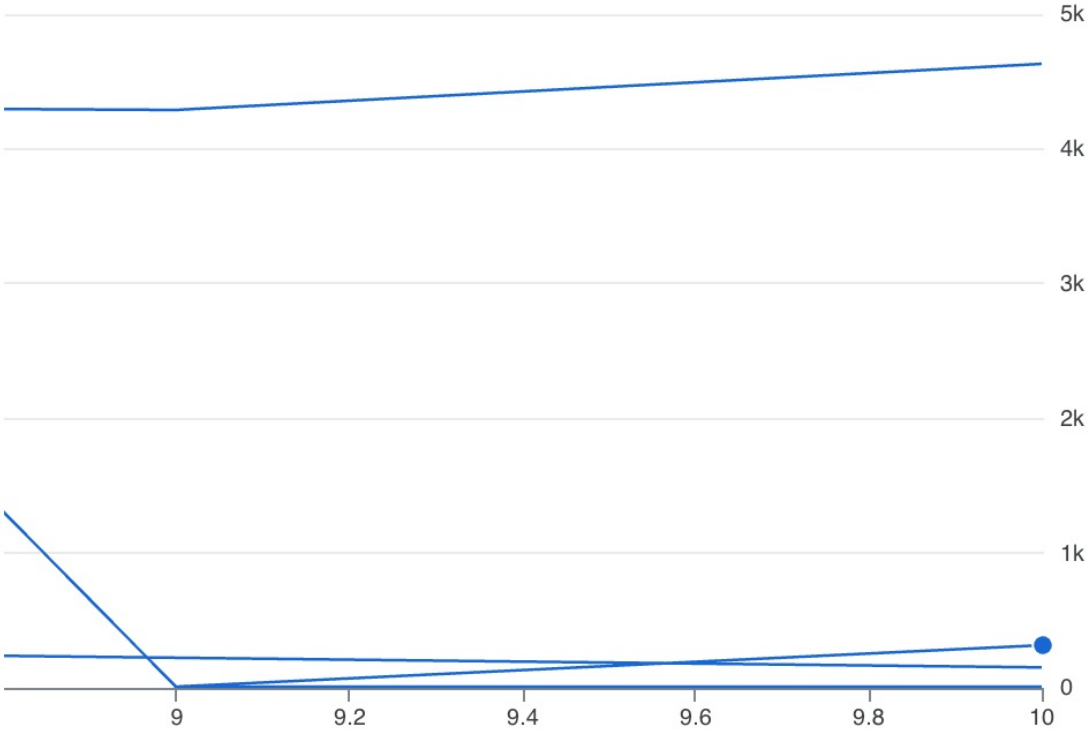
Ans:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS `year`,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS
`month`,
  COUNT(order_id) AS `total_orders_per_month`
FROM
  `cs1.orders`
GROUP BY
  month,
  year
ORDER BY
  year ASC,
  month ASC;
```

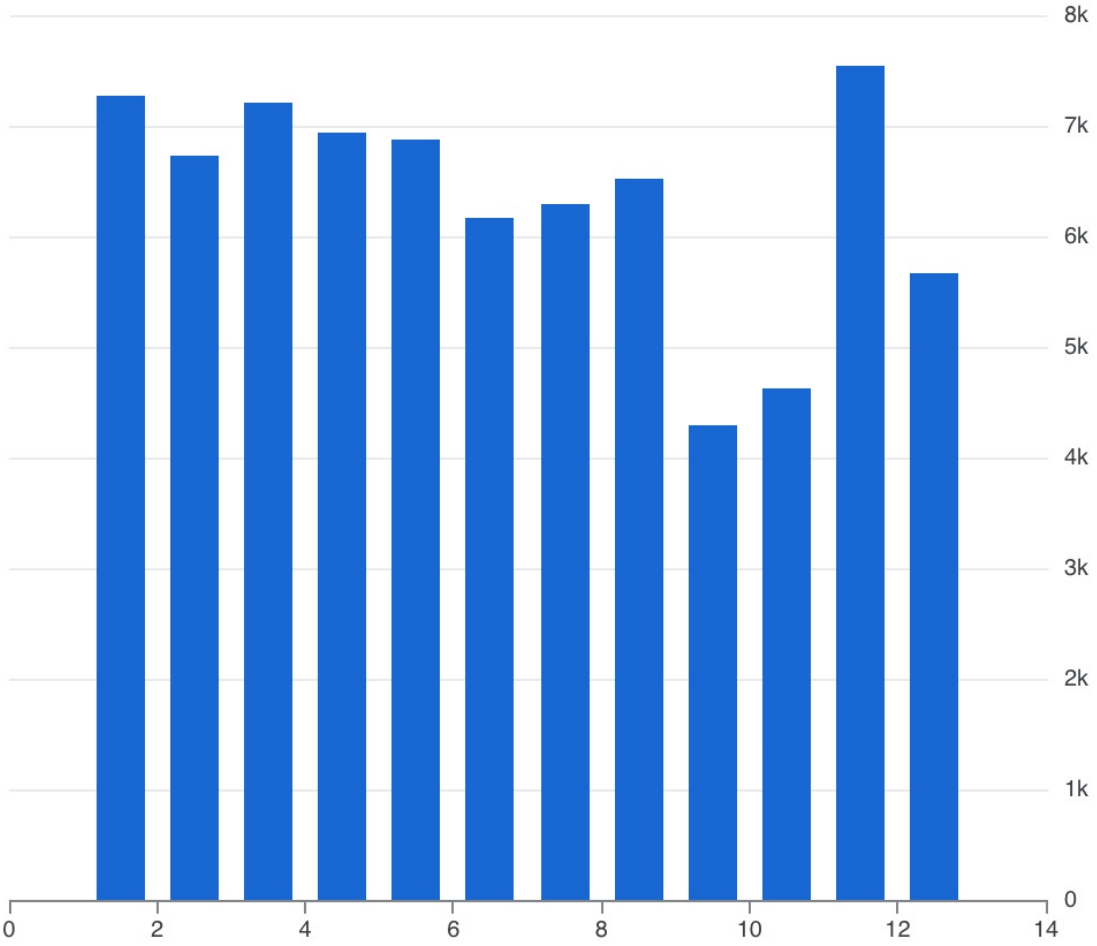
Screenshot:

Row	year	month	total_orders_per_month
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

total_orders_per_month by month



total_orders_per_month by month



Insights:
1. Yes there is seasonality. Especially there is a drastic drop in sales during September to December months probably due to festival season or winter break.

Recommendation:
1. Adjust the inventory during months 9 to 12 by either performing flash sale or bundled/up sale to exhaust the inventory. Or reduce the store working hours to avoid unnecessary store maintenance costs. Or stock up specific products that are related to the festival to increase sales.

Q2.3 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**

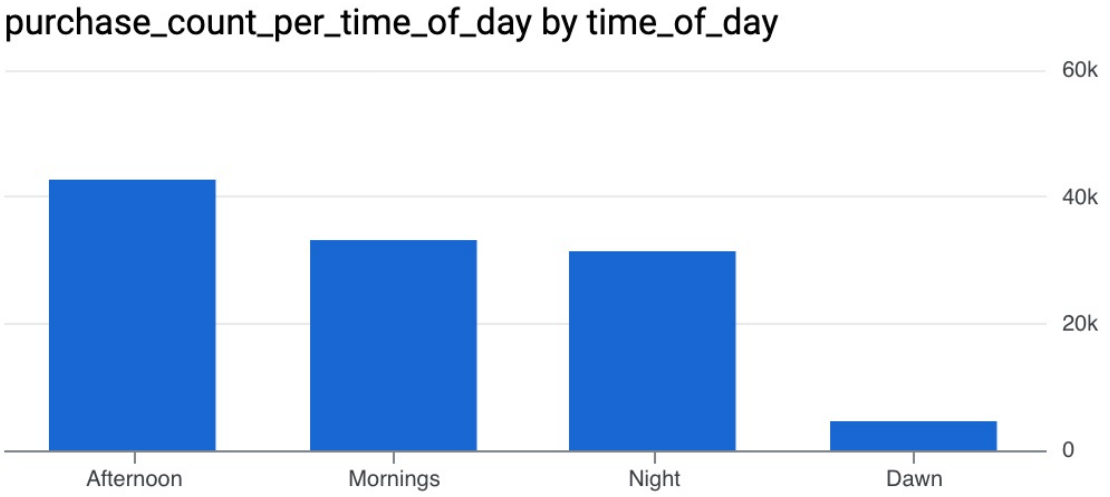
Ans:

```
WITH
  `T` AS (
    SELECT
      G.geolocation_city,
      C.customer_id,
      O.order_id,
      TIME(O.order_purchase_timestamp) AS
`order_purchase_timestamp`,
      (
        CASE
        WHEN EXTRACT(HOUR FROM
TIME(O.order_purchase_timestamp)) BETWEEN 0 AND 6 THEN
"Dawn"

        WHEN EXTRACT(HOUR FROM
```

```
TIME(O.order_purchase_timestamp)) BETWEEN 7 AND 12 THEN
"Mornings"

        WHEN EXTRACT(HOUR FROM
TIME(O.order_purchase_timestamp)) BETWEEN 13 AND 18 THEN
"Afternoon"

        WHEN EXTRACT(HOUR FROM
TIME(O.order_purchase_timestamp)) BETWEEN 19 AND 23 THEN
"Night"
        END
      ) AS `time_of_day`
    FROM
      `cs1.geolocation` AS `G` RIGHT JOIN `cs1.customers` AS `C`
      ON G.geolocation_zip_code_prefix =
C.customer_zip_code_prefix

      RIGHT JOIN `cs1.orders` AS `O`
      ON C.customer_id = O.customer_id
    WHERE
      LOWER(geolocation_city) LIKE "bra%"
)
SELECT
  time_of_day,
  COUNT(time_of_day) AS `purchase_count_per_time_of_day`,
FROM
  T
GROUP BY
  time_of_day
ORDER BY
  purchase_count_per_time_of_day DESC;
```

Screenshot:

Row	time_of_day	purchase_count_per_time_of_day
1	Afternoon	42836
2	Mornings	33074
3	Night	31559
4	Dawn	4432



Insights:
1. Brazilian customers mostly place their orders in the afternoon.

Recommendation:
1. Adjust store timings, increase staff capacity and stock up more products during afternoons to address more customers in Brazil location.

Q3.1 Get the month on month no. of orders placed in each state.

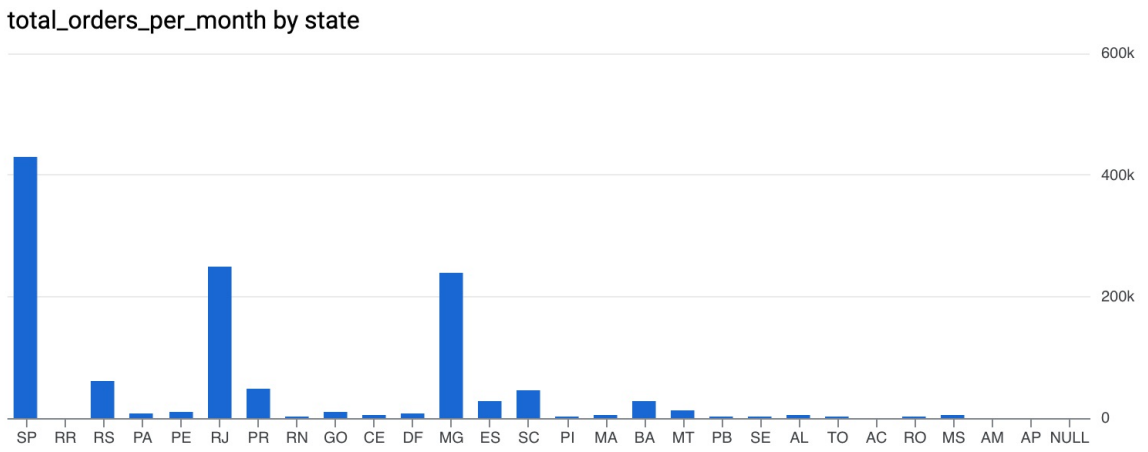
Ans:

```
WITH
  `T` AS (
    SELECT
```

```
      *,
      EXTRACT(YEAR FROM order_purchase_timestamp) AS `year`,
      EXTRACT(MONTH FROM order_purchase_timestamp) AS
`month`
    FROM
      `cs1.geolocation` AS `G` RIGHT JOIN `cs1.customers` AS `C`
      ON G.geolocation_zip_code_prefix =
C.customer_zip_code_prefix

      RIGHT JOIN `cs1.orders` AS `O`
      ON C.customer_id = O.customer_id
    ORDER BY
      year ASC,
      month ASC
  )
SELECT
  geolocation_state AS `state`,
  year,
  month,
  COUNT(order_id) AS `total_orders_per_month`
FROM
  T
GROUP BY
  geolocation_state,
  month,
  year
ORDER BY
  year ASC,
  month ASC;
```

Screenshot:

Row	state	year	month	total_orders_per_month
1	RS	2016	9	103
2	SP	2016	9	492
3	RR	2016	9	65
4	PE	2016	10	382
5	RJ	2016	10	12416
6	SP	2016	10	16277
7	PR	2016	10	2334
8	BA	2016	10	292
9	MG	2016	10	11756
10	GO	2016	10	367



Insights:
1. It appears that some states have low activity in November and December months while others dont. Unsurprisingly. So we can conclude that the the states with low sales during 9th and 10th months are either in winter break or some kind of festival season.

Recommendation:
1. We can adjust stock based on season in every location. Follow

Q2.3 recommendations.

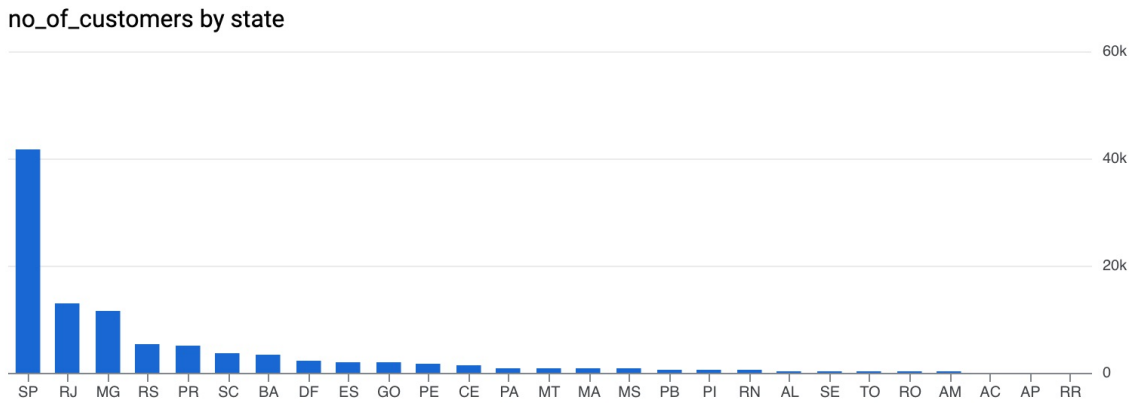
Q3.2 How are the customers distributed across all the states?

Ans:

```
SELECT
  customer_state AS `state`,
  COUNT(DISTINCT customer_id) AS `no_of_customers`
FROM
  `cs1.geolocation` AS `G` RIGHT JOIN `cs1.customers` AS `C`
  ON G.geolocation_zip_code_prefix = C.customer_zip_code_prefix
GROUP BY
  customer_state
ORDER BY
  no_of_customers DESC;
```

Screenshot:

Row	state	no_of_customers
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



Insights:
1. State SP has highest customers with 41746 while state RR has lowest with 46.

Recommendation:
1. We should focus more on SP state to drive more sales and set up loyalty programs to retain existing customers better. We should promote more in RR and other lesser states. Either our promotions are not effective or the price may be too high for this area.

Q4.1 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.

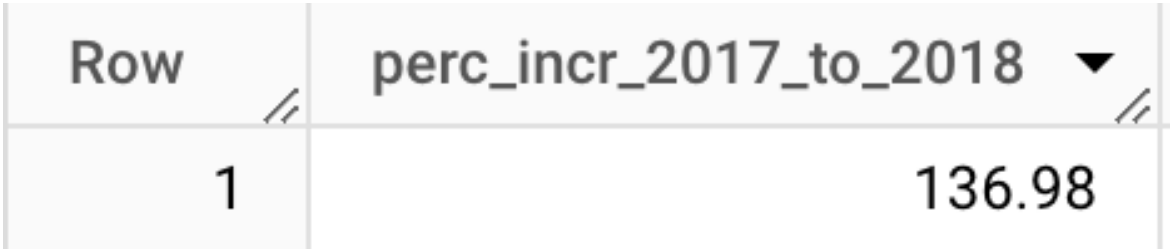
Algo:
1. Get cost of orders in Jan to Aug 2017 and add all of them.
2. Get cost of orders in Jan to Aug 2018 and add all of them.
3. Subtract both of them to get the increase.
4. To get % increase use this formula: 100 * ((final - initial) / initial)

Ans:

```
WITH
  `T` AS (
    SELECT DISTINCT
      EXTRACT(YEAR FROM order_purchase_timestamp) AS `year`,
      SUM(P.payment_value) OVER(PARTITION BY EXTRACT(YEAR
```

```
FROM order_purchase_timestamp)) AS `pay_jan_to_aug`
    FROM
      `cs1.orders` AS `O` RIGHT JOIN `cs1.payments` AS `P`
      ON O.order_id = P.order_id
    WHERE
      EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN
2017 AND 2018
      AND
      EXTRACT(MONTH FROM order_purchase_timestamp)
BETWEEN 1 AND 8
  )
SELECT
  ROUND(((MAX(pay_jan_to_aug) - MIN(pay_jan_to_aug)) /
MIN(pay_jan_to_aug)) * 100, 2) AS `perc_incr_2017_to_2018`
FROM
  T
```

Screenshot:



Row	perc_incr_2017_to_2018
1	136.98

Insights:
1. 5025712 increase in amount or 137 % increase in payments from 2017 to 2018.
2. January to August is the time when we had high sales as per the given data. Stretched over time we can get an insight in the customer spending behaviour.
3. It appears that there is demand for our products and customers are willing to either pay more or we can sell more.

Recommendation: NA

Q4.2 Calculate the Total & Average value of order price for each state. (NEED CLARIFICATION)

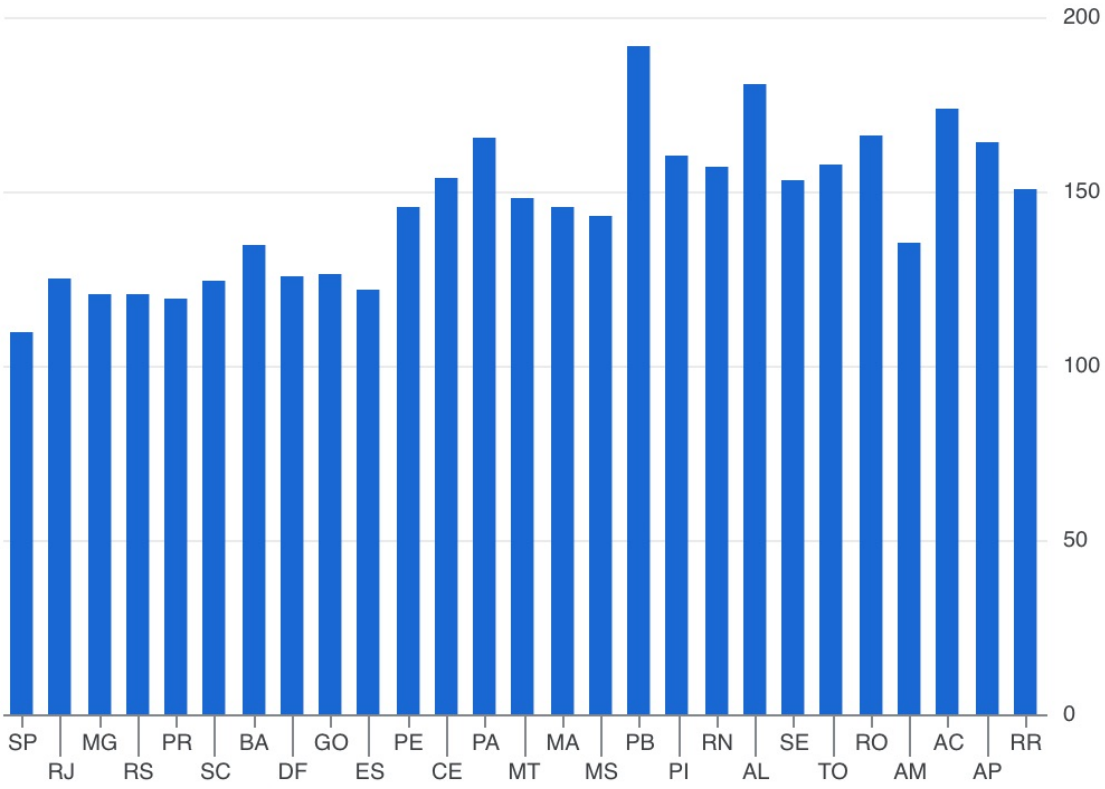
Ans:

```
SELECT
  IFNULL(C.customer_state, "Unknown") AS `state`,
  ROUND(SUM(I.price), 2) AS `total_price`,
  ROUND(AVG(I.price), 2) AS `avg_price`
FROM
  `cs1.orders` AS `O` RIGHT JOIN `cs1.order_items` AS `I`
  ON O.order_id = I.order_id

  LEFT JOIN `cs1.customers` AS `C`
  ON O.customer_id = C.customer_id
GROUP BY
  C.customer_state
ORDER BY
  total_price DESC;
```

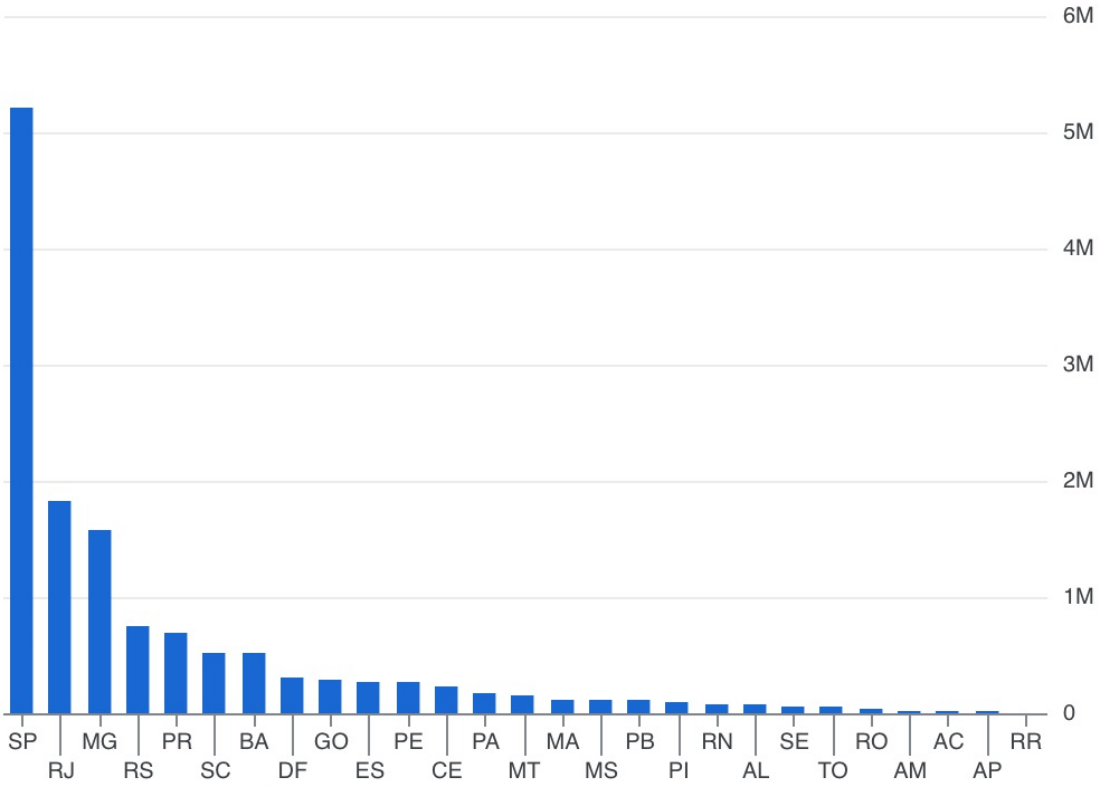
Screenshot:

Row	state	total_price	avg_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91

avg_price by state



total_price by state



Insights:

1. The highest revenue is from SP state at 5202955.05 while least

sale happened in RR state at 7829.43. However the highest average price is from PB state at 191.48 and least from SP at 109.65. So it appears there are much more wealthy customers or products are in demand in PB and SP states particularly in PB state with high average order price.

Recommendation:
1. Investing more in PB might give us higher returns in the future. We should prepare premium plans for these areas and perform a test run to see if it drives more revenue.

Q4.3 Calculate the Total & Average value of order freight for each state. (NEED CLARIFICATION)

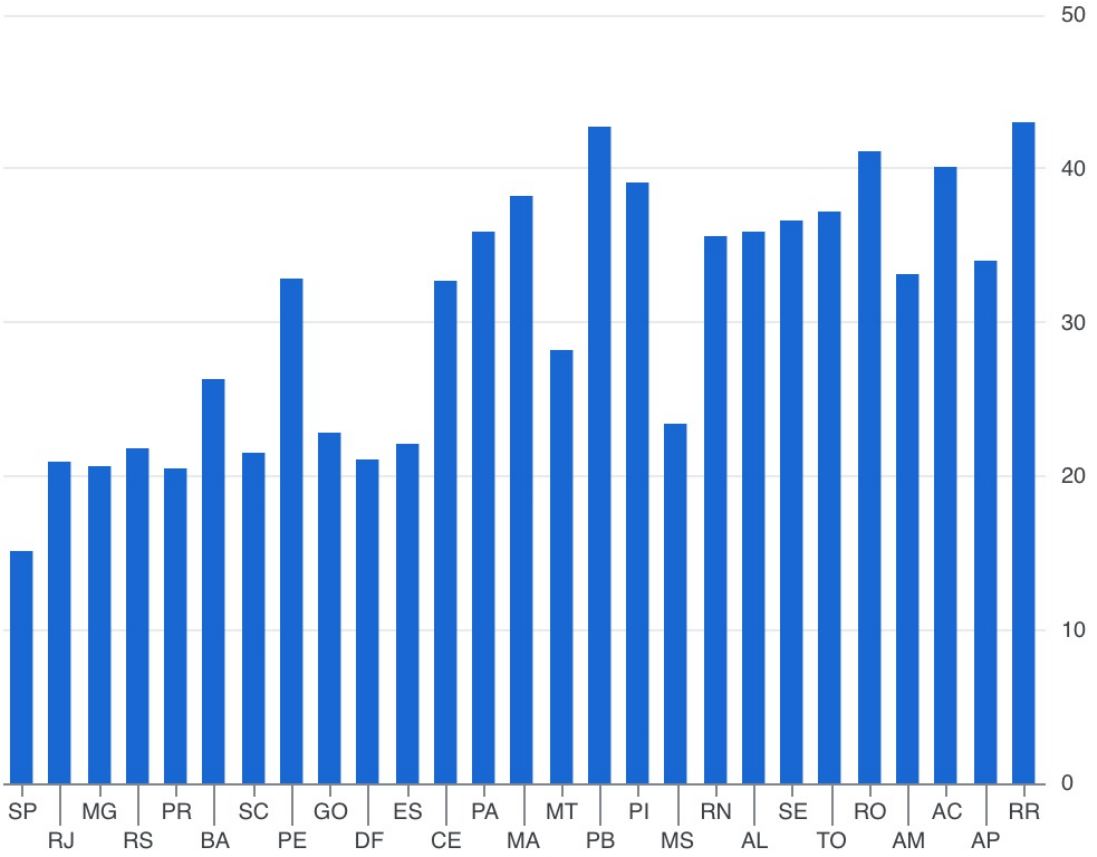
Ans:

```
SELECT
  IFNULL(C.customer_state, "Unknown") AS `state`,
  ROUND(SUM(I.freight_value), 2) AS `total_freight_value`,
  ROUND(AVG(I.freight_value), 2) AS `avg_freight_value`
FROM
  `cs1.orders` AS `O` LEFT JOIN `cs1.order_items` AS `I`
  ON O.order_id = I.order_id

  LEFT JOIN `cs1.customers` AS `C`
  ON O.customer_id = C.customer_id
GROUP BY
  C.customer_state
ORDER BY
  total_freight_value DESC;
```

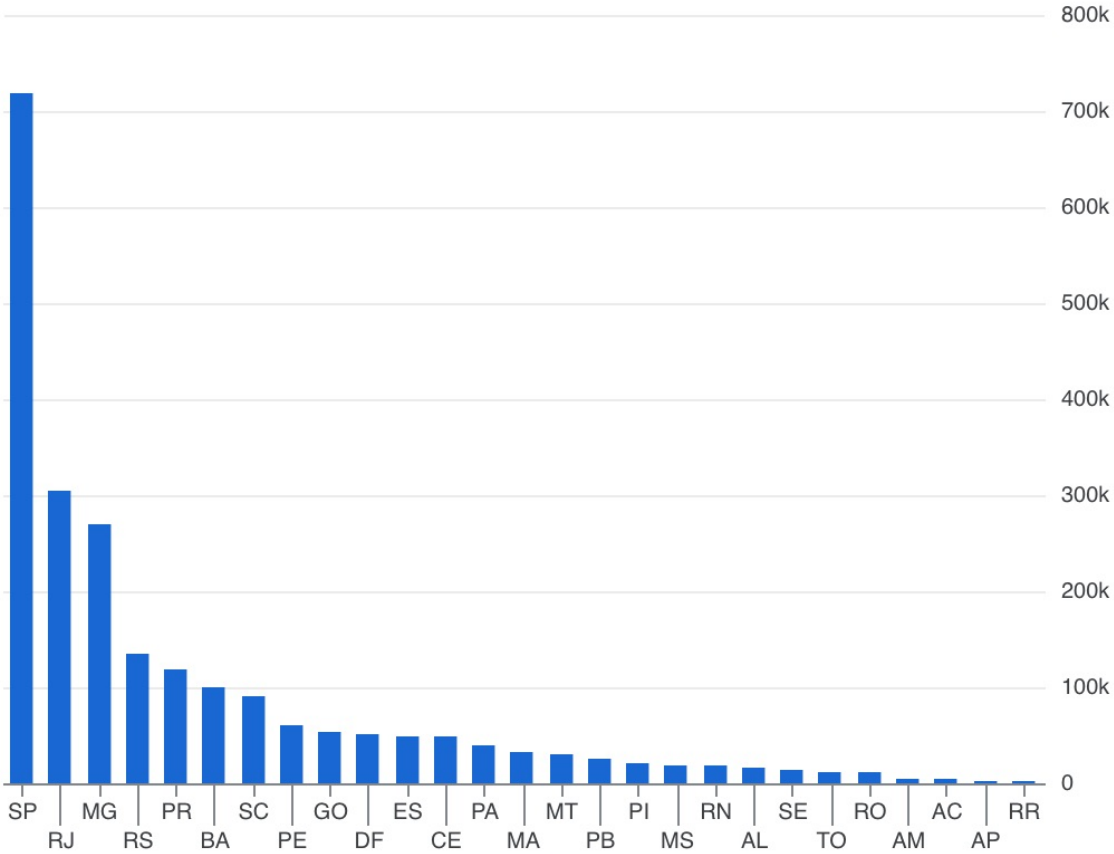
Screenshot:

Row	state	total_freight_value	avg_freight_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04

avg_freight_value by state



total_freight_value by state



Insights:
1. SP state has highest freight value at 718723.07 while RR seems to have lowest at 2235.19. Avg wise RR has highest at 42.98 while SP has lowest at 15.15.

Recommendation:
1. Find ways to reduce packaging size in area SP to reduce freight costs.
2. Find alternate transportation or routes.

Q5.1 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:

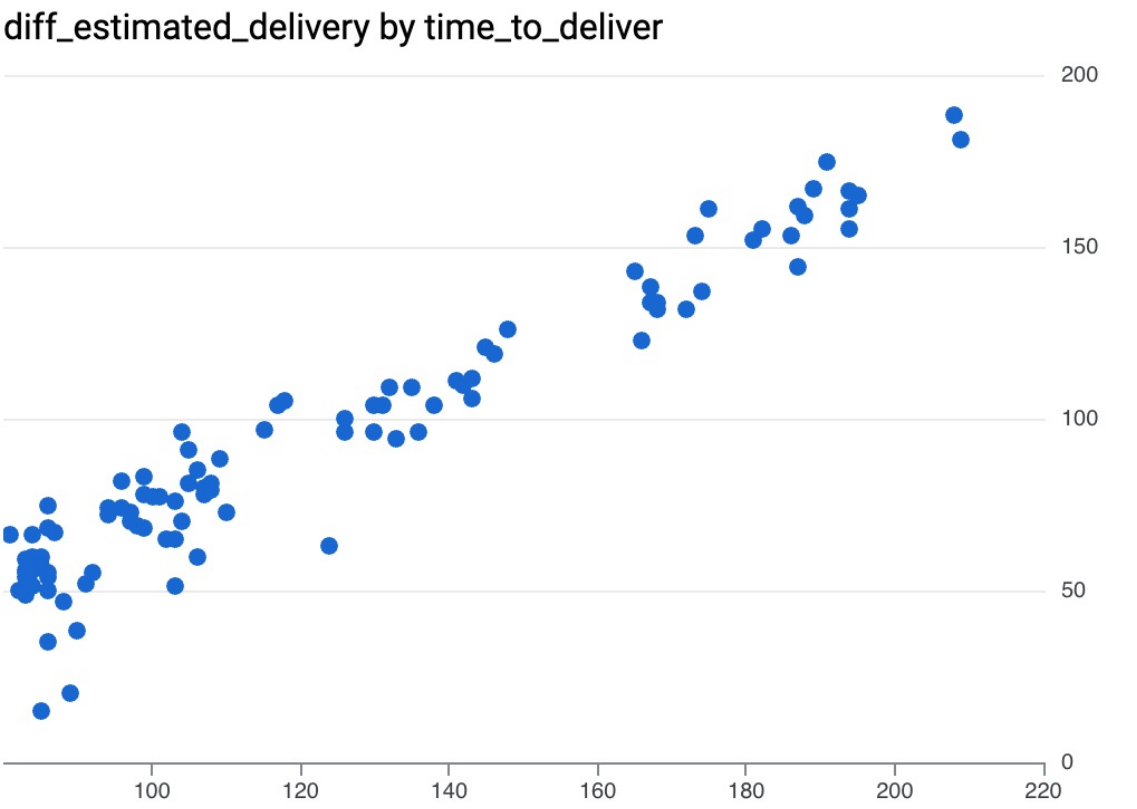
- **time_to_deliver = order_delivered_customer_date - order_purchase_timestamp**
- **diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date**

Ans:

```
SELECT
  O.order_id,
  ABS(DATE_DIFF(O.order_purchase_timestamp,
O.order_delivered_customer_date, DAY)) AS `time_to_deliver`,
  ABS(DATE_DIFF(O.order_estimated_delivery_date,
O.order_delivered_customer_date, DAY)) AS
`diff_estimated_delivery`
FROM
  `cs1.orders` AS `O`
ORDER BY
  time_to_deliver DESC;
```

Screenshot:

Row	order_id	time_to_deliver	diff_estimated_delivery
1	ca07593549f1816d26a572e06dc1eab6	209	181
2	1b3190b2dfa9d789e1f14c05b647a14a	208	188
3	440d0d17af552815d15a9e41abe49359	195	165
4	0f4519c5f1c541ddec9f21b3bddd533a	194	161
5	285ab9426d6982034523a855f55a885e	194	166
6	2fb597c2f772eca01b1f5c561bf6cc7b	194	155
7	47b40429ed8cce3aee9199792275433f	191	175
8	2fe324febf907e3ea3f2aa9650869fa5	189	167
9	2d7561026d542c8dbd8f0daeadf67a43	188	159
10	437222e3fd1b07396f1d9ba8c15fba59	187	144



Insights:
1. There seems to be a gap of min 20 days/hours in the expectations vs reality of delivery.
2. Inefficient route planning or poor algorithms or poor overall planning.

Recommendation:
1. Improve path finding algorithms to better plan the route if being

used to ensure timely order fulfilment.
2. Measure time at every checkpoint to identify delivery delays to optimise supply chain processes.
3. Find ways to reduce packaging size to reduce freight costs.
4. Find alternate transportation or routes.

Q5.2 Find out the top 5 states with the highest & lowest average freight value.

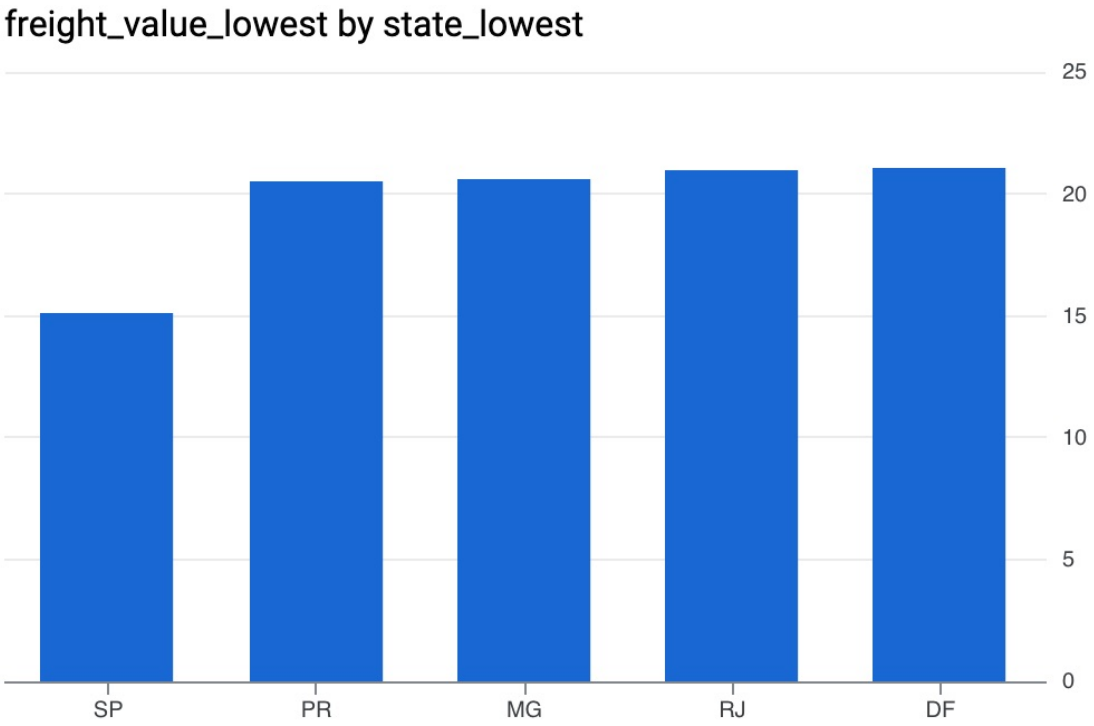
Ans:

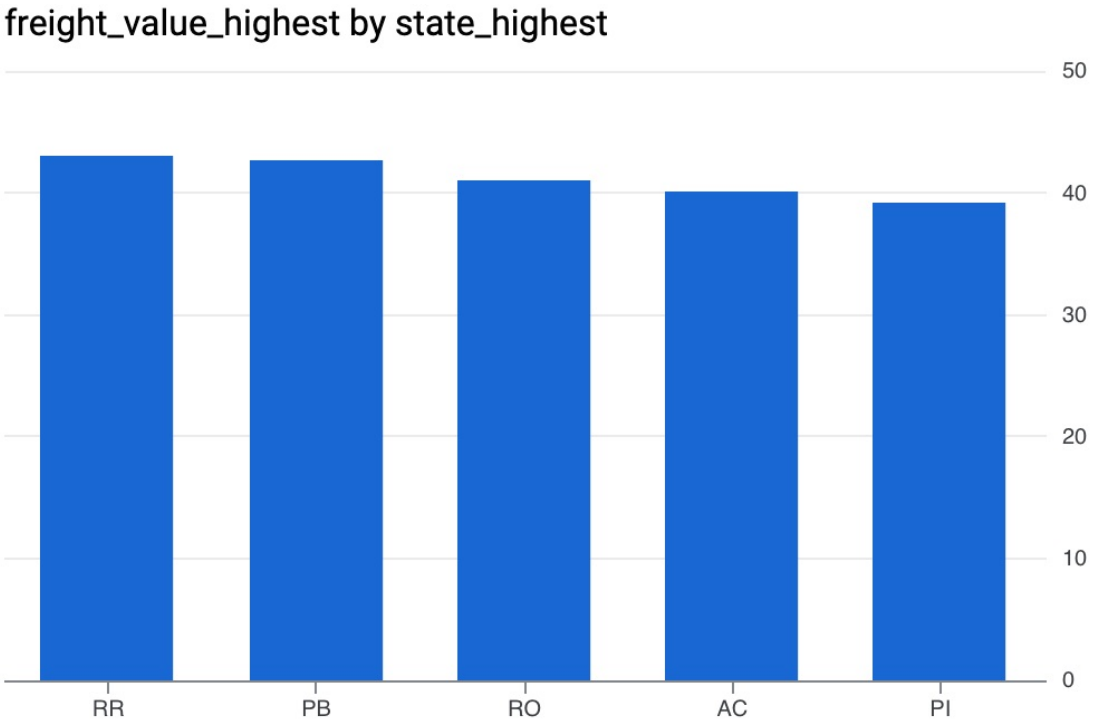
```
WITH
  `T` AS (
    SELECT
      C.customer_state,
      I.freight_value
    FROM
      `cs1.orders` AS `O` LEFT JOIN `cs1.order_items` AS `I`
      ON O.order_id = I.order_id

      LEFT JOIN `cs1.customers` AS `C`
      ON O.customer_id = C.customer_id
  ),
  `T1` AS (
    SELECT
      IFNULL(customer_state, "Unknown") AS `state`,
      ROUND(AVG(freight_value), 2) AS `avg_freight_value`
    FROM
      T
    GROUP BY
      customer_state
    ORDER BY
      avg_freight_value DESC
    LIMIT 5
  ),
  `T2` AS (
    SELECT
      IFNULL(customer_state, "Unknown") AS `state`,
```

```
    ROUND(AVG(freight_value), 2) AS `avg_freight_value`
  FROM
    T
  GROUP BY
    customer_state
  ORDER BY
    avg_freight_value ASC
  LIMIT 5
),
`T3` AS (
  SELECT
    ROW_NUMBER() OVER() AS `row`,
    *
  FROM
    T1
),
`T4` AS (
  SELECT
    ROW_NUMBER() OVER() AS `row`,
    *
  FROM
    T2
)
SELECT
  T3.state AS `state_highest`,
  T3.avg_freight_value AS `freight_value_highest`,
  T4.state AS `state_lowest`,
  T4.avg_freight_value AS `freight_value_lowest`
FROM
  T3 INNER JOIN T4 ON T3.row = T4.row
```

Screenshot:

Row	state_highest	freight_value_highest	state_lowest	freight_value_lowest
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04





Insights:
1. RR has highest freight value at 42.98 while SP has lowest at 15.15.

Recommendation:
1. Optimise packaging accordingly to increase the quantity

shipped.
2. Negotiate for lower freight rates.
3. Improve resource allocation.
4. Follow recommendations mentioned in Q5.1

Q5.3 Find out the top 5 states with the highest & lowest average delivery time.

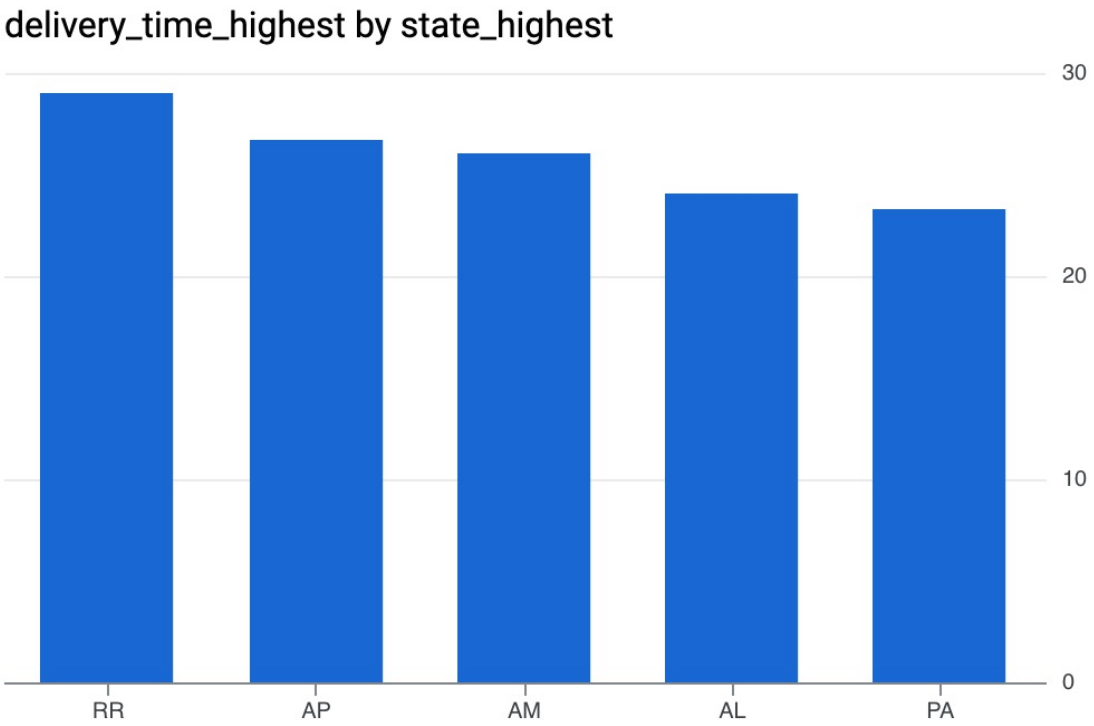
Ans:

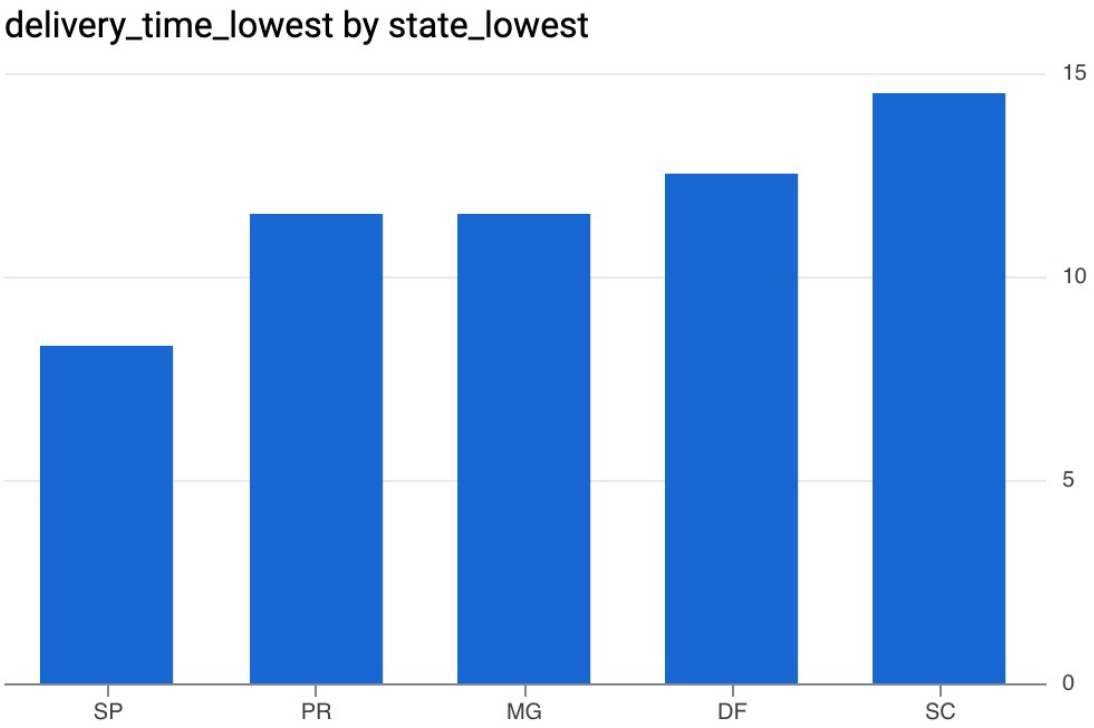
```
WITH
  `T` AS (
    SELECT
      C.customer_state,
      ABS(DATE_DIFF(O.order_purchase_timestamp,
O.order_delivered_customer_date, DAY)) AS `delivery_time`
    FROM
      `cs1.orders` AS `O` LEFT JOIN `cs1.customers` AS `C`
      ON O.customer_id = C.customer_id
  ),
  `T1` AS (
    SELECT
      IFNULL(customer_state, "Unknown") AS `state`,
      ROUND(AVG(delivery_time), 2) AS `avg_delivery_time`
    FROM
      T
    GROUP BY
      customer_state
    ORDER BY
      avg_delivery_time DESC
    LIMIT 5
  ),
  `T2` AS (
    SELECT
      IFNULL(customer_state, "Unknown") AS `state`,
      ROUND(AVG(delivery_time), 2) AS `avg_delivery_time`
    FROM
      T
```

```
    GROUP BY
      customer_state
    ORDER BY
      avg_delivery_time ASC
    LIMIT 5
  ),
  `T3` AS (
    SELECT
      ROW_NUMBER() OVER() AS `row`,
      *
    FROM
      T1
  ),
  `T4` AS (
    SELECT
      ROW_NUMBER() OVER() AS `row`,
      *
    FROM
      T2
  )
SELECT
  T3.state AS `state_highest`,
  T3.avg_delivery_time AS `delivery_time_highest`,
  T4.state AS `state_lowest`,
  T4.avg_delivery_time AS `delivery_time_lowest`
FROM
  T3 INNER JOIN T4 ON T3.row = T4.row
```

Screenshot:

Row	state_highest	delivery_time_highest	state_lowest	delivery_time_lowest
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48





Insights:
1. RR state has highest delivery time while SP has lowest. There could be many reasons for this. Dense population, poor transportation, product demand, state laws, etc.

Recommendation:
1. Scout the area to get more info and optimise departments that

are taking more time.
2. Delivery time and freight rates can sometimes have a correlation. Follow recommendations in Q5.2

Q5.4 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.

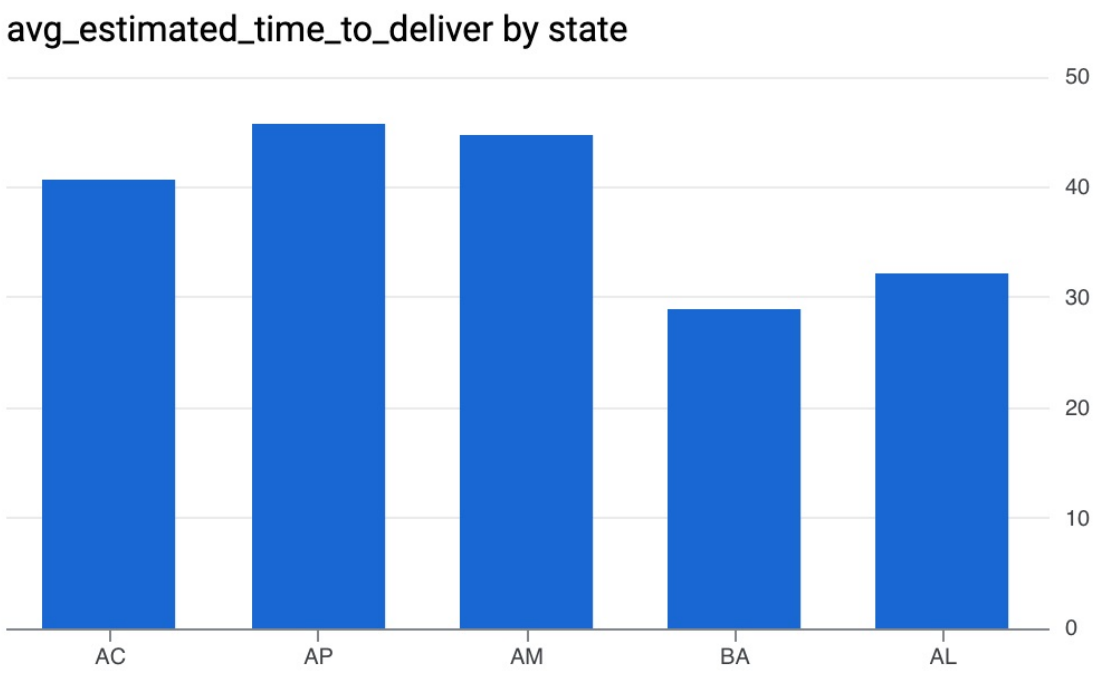
Ans:

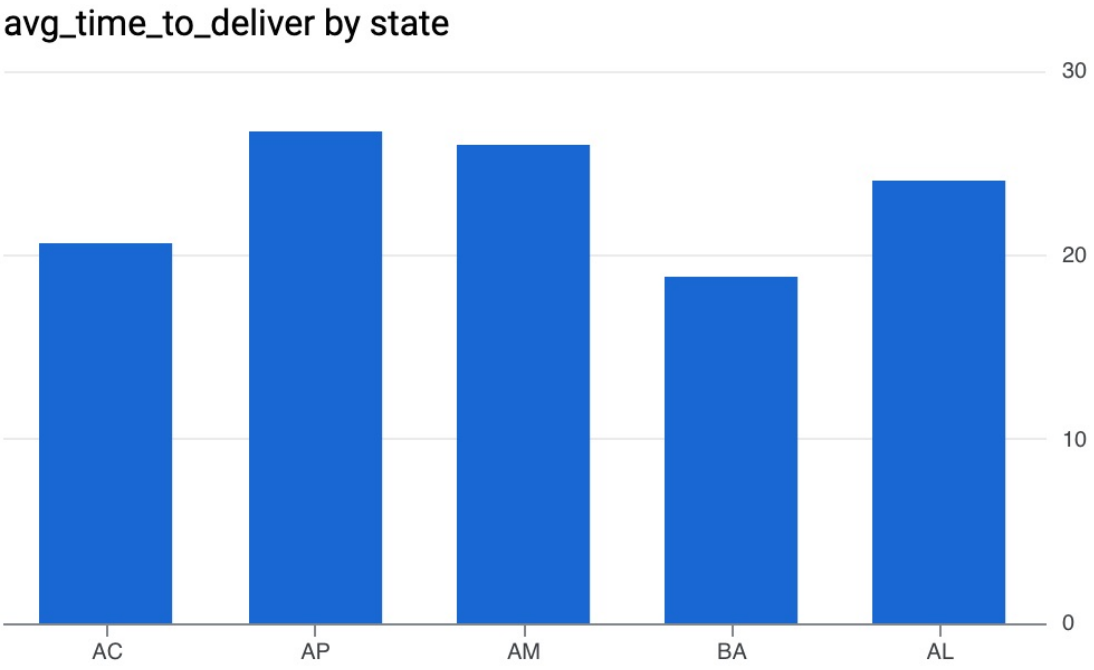
```
WITH
  `T` AS (
    SELECT
      C.customer_state,
      ABS(DATE_DIFF(O.order_purchase_timestamp,
O.order_delivered_customer_date, DAY)) AS `time_to_deliver`,
      ABS(DATE_DIFF(O.order_purchase_timestamp,
O.order_estimated_delivery_date, DAY)) AS
`estimated_time_to_deliver`
    FROM
      `cs1.orders` AS `O` LEFT JOIN `cs1.customers` AS `C`
      ON O.customer_id = C.customer_id
  )
SELECT DISTINCT
  customer_state AS `state`,
  ROUND(AVG(time_to_deliver), 2) AS `avg_time_to_deliver`,
  ROUND(AVG(estimated_time_to_deliver), 2) AS
`avg_estimated_time_to_deliver`,
  ROUND(ABS(AVG(time_to_deliver) -
AVG(estimated_time_to_deliver)), 2) AS `fast_by`
FROM
  T
WHERE
  customer_state IN (
      SELECT DISTINCT
        customer_state AS `state`,
```

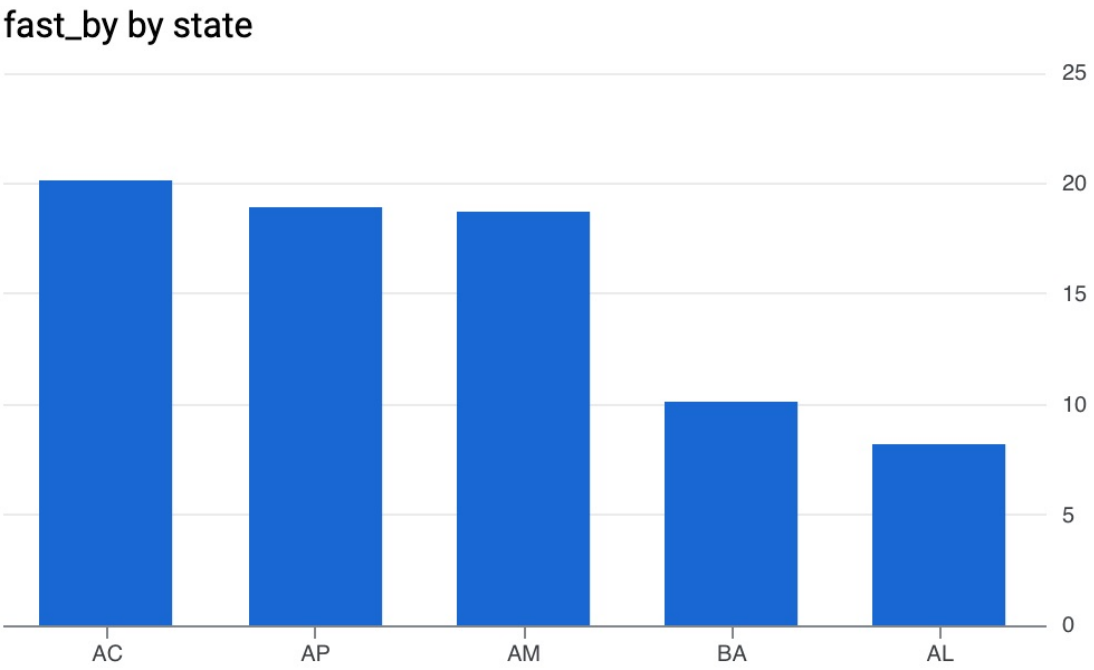
```
    FROM
      T
    WHERE
      time_to_deliver < estimated_time_to_deliver AND
      customer_state IS NOT NULL
    ORDER BY
      state ASC
    LIMIT 5
  )
GROUP BY
  customer_state
ORDER BY
  fast_by DESC;
```

Screenshot:

Row	state	avg_time_to_deliver	avg_estimated_time_to_deliver	fast_by
1	AC	20.64	40.77	20.13
2	AP	26.73	45.71	18.97
3	AM	25.99	44.76	18.77
4	BA	18.87	29.04	10.17
5	AL	24.04	32.23	8.18







Insights:
1. State AC is faster by 20.13 days from its estimated delivery date 40.77 days. ~1/2 the time.
2. Faster deliveries improve customer satisfaction.
3. Faster deliveries also shows a streamlined delivery process. The process can be used as a template to emulate the speed in other areas.

Recommendation:

1. Increase stock in AM state to make more deliveries.

Q6.1 Find the month on month no. of orders placed using different payment types.

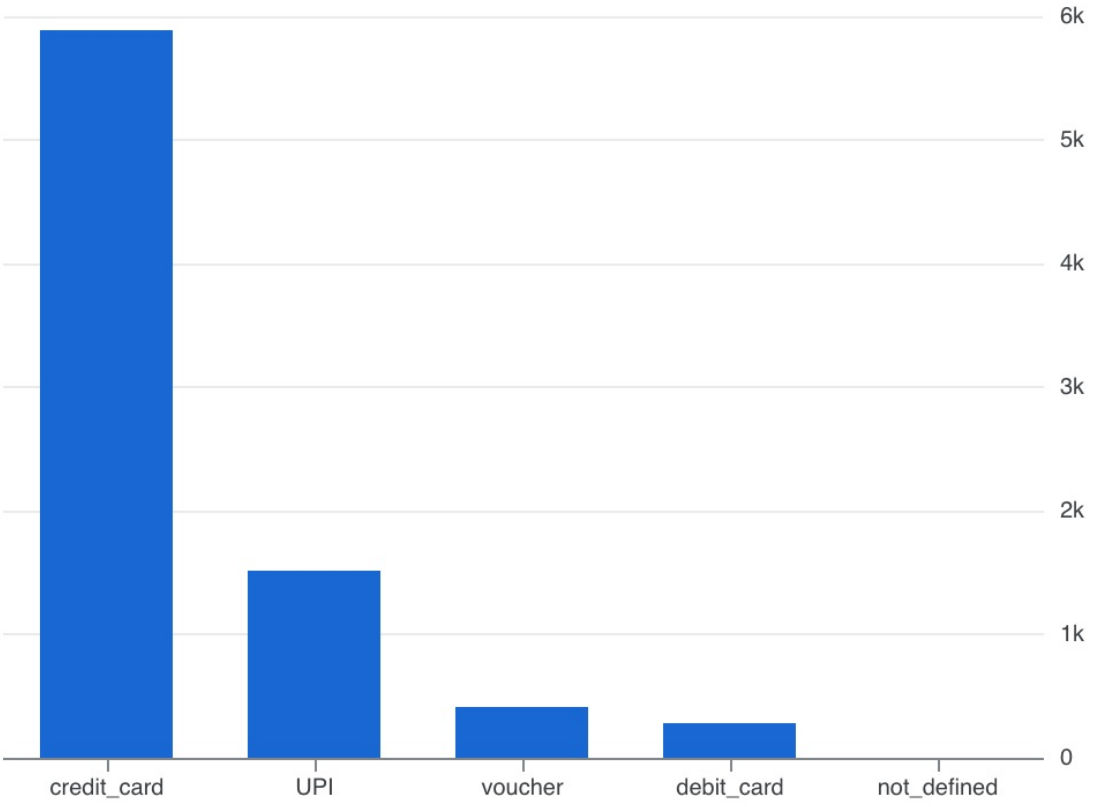
Ans:

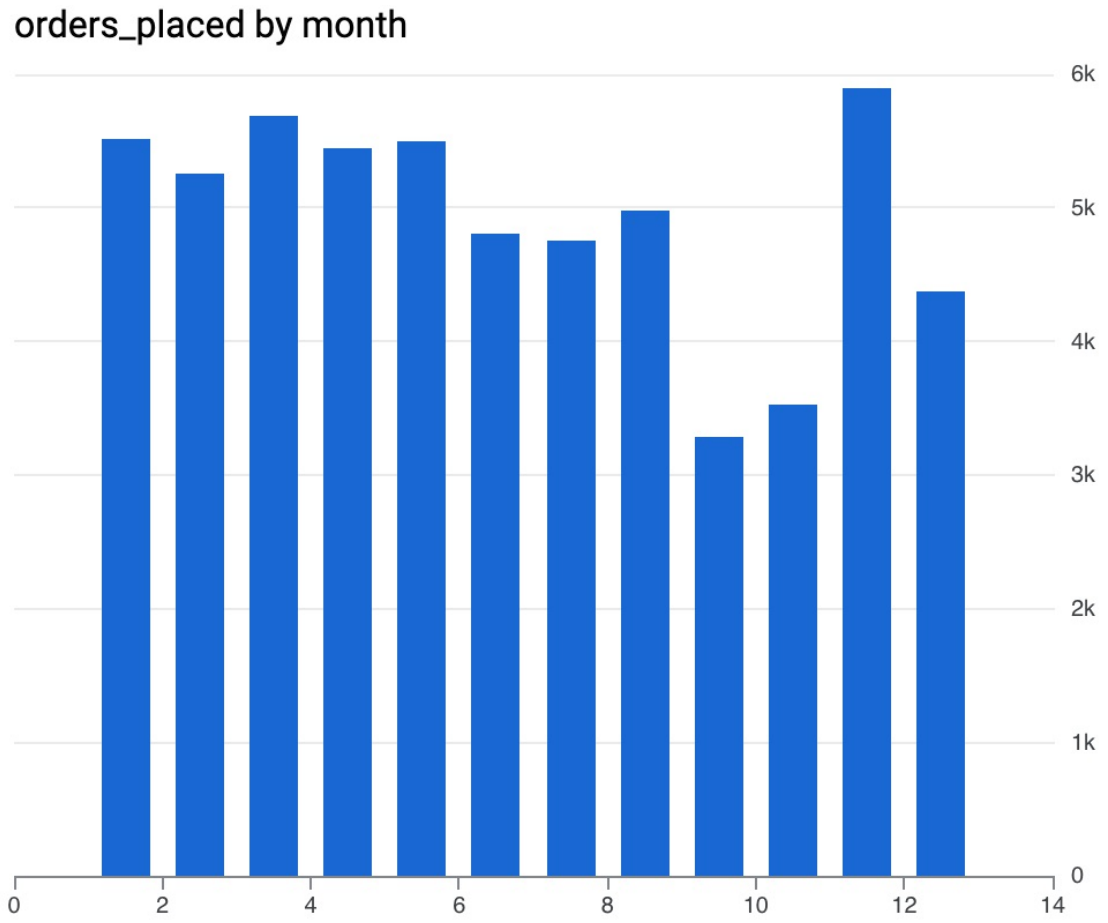
```
SELECT
  EXTRACT(YEAR FROM O.order_purchase_timestamp) AS `year`,
  EXTRACT(MONTH FROM O.order_purchase_timestamp) AS
`month`,
  P.payment_type,
  COUNT(O.order_id) AS `orders_placed`
FROM
  `cs1.orders` AS `O` LEFT JOIN `cs1.payments` AS `P`
  ON O.order_id = P.order_id
GROUP BY
  year,
  month,
  P.payment_type
HAVING
  P.payment_type IS NOT NULL
ORDER BY
  year ASC,
  month ASC;
```

Screenshot:

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

orders_placed by payment_type





Insights:
1. Credit card is clearly the most popular choice followed by UPI. Debit card being the least preferred.
2. We can observe seasonality in the usage of payment types.

Recommendation:
1. We should team up with banks to give low interest credit to those who signup with our scheme. Provide cash backs for using their credit card.

Q6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.

Ans:
SELECT

```
  COUNT(O.order_id) AS `orders_placed`,
  P.payment_sequential
FROM
  `cs1.orders` AS `O` LEFT JOIN `cs1.payments` AS `P`
  ON O.order_id = P.order_id
GROUP BY
  P.payment_sequential
ORDER BY
  orders_placed DESC;
```

Screenshot:

Row	orders_placed	payment_sequential
1	99360	1
2	3039	2
3	581	3
4	278	4
5	170	5
6	118	6
7	82	7
8	54	8
9	43	9
10	34	10

Insights:
1. It looks like most orders are placed in a single up-front payment. This is good.

Recommendation:
1. We should stick to the existing plan here.