

Target_SQL case study

Q.1

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

The screenshot shows a database interface with an Explorer on the left and a table schema view on the right. The Explorer lists resources under 'targetsql-430317' > 'target_sql', including 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', and 'products'. The 'customers' table is selected, and its schema is displayed in the main panel. The schema table has columns: Field name, Type, Mode, Key, Collation, Default Value, and Policy Tags. The data is as follows:

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags
customer_id	STRING	NULLABLE	-	-	-	-
customer_unique_id	STRING	NULLABLE	-	-	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-
customer_city	STRING	NULLABLE	-	-	-	-
customer_state	STRING	NULLABLE	-	-	-	-

2. Time range between orders placed

The screenshot shows a database interface with an Explorer on the left and a query editor on the right. The Explorer lists resources under 'kalpana-sql1', including 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'orders' table is selected. The query editor shows an 'Untitled query' with the following SQL:

```
SELECT
  min(order_purchase_timestamp) as first_date_of_purchase,
  max(order_purchase_timestamp) as last_date_of_purchase
FROM `targetsql-430317.target_sql.orders`
```

The query results are displayed in a table with columns: Row, first_date_of_purchase, and last_date_of_purchase. The results are as follows:

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

Inference: The orders placed time is nothing but the order purchase timestamp. The minimum date-time and the max date-time will give the range in which orders get placed.

3. Count the Cities & States of customers

```
1 with states as
2   (select count(*) as no_of_states
3    from
4      (select distinct c.customer_state as states
5       from `targetsql-430317.target_sql.customers` as c
6       join `targetsql-430317.target_sql.orders` as o on
7         c.customer_id = o.customer_id
8       group by customer_state
9      )
10  ),
11 cities as
12   (select count(*) as no_of_cities
13    from
14      (select distinct c.customer_city as cities
15       from `targetsql-430317.target_sql.customers` as c
16       join `targetsql-430317.target_sql.orders` as o on
17         c.customer_id = o.customer_id
18       group by customer_city
19      )
20  ),
21 final_data as
22   (
23     select no_of_states, no_of_cities from states, cities
24   )
25 select * from final_data
```

Query results

Row	no_of_states	no_of_cities
1	27	4119

Inference: Count of Cities & States of customers who ordered between min and max range of order_purchase_timestamp. So all orders will be considered.

Q.2

1.

```
1 select extract(year from order_purchase_timestamp ) as Year,
2        count(*) as no_of_orders from targetsql-430317.target_sql.orders
3 group by extract(year from order_purchase_timestamp)
4 order by Year
```

Query results

Row	Year	no_of_orders
1	2016	329
2	2017	45101
3	2018	54011

Inference: There has been a growing trend in the no. of orders placed over the past years as the orders are increasing year by year.

2.

The screenshot shows the TargetSQL interface. At the top, there's a search bar and a 'Search' button. Below that, a banner for 'Sandbox' mentions upgrading to the full BigQuery experience. The main area is divided into an 'Explorer' on the left and a query editor on the right. The Explorer shows a list of resources: customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers. The query editor shows a SQL query:

```
1 select format_datetime("%m %Y", order_purchase_timestamp) as date_format,
2 > count(*) as no_of_orders
3 from orders
4 group by format_datetime("%m %Y", order_purchase_timestamp)
5 order by date_format
6
```

The query results are displayed in a table with columns 'date_format' and 'no_of_orders'. The results show a peak in orders during January and February, followed by a decline.

Row	date_format	no_of_orders
1	01 2017	800
2	01 2018	7269
3	02 2017	1780
4	02 2018	6728
5	03 2017	2682
6	03 2018	7211
7	04 2017	2404
8	04 2018	6939
9	05 2017	3700
10	05 2018	6873
11	06 2017	2245

Inference: There are adequate number of orders placed in the month from January to August but those get reduced from September to December.

3.

The screenshot shows the TargetSQL interface with a more complex SQL query. The query is:

```
1 select order_period, count(*) as no_of_orders_during_period from
2 (select c.customer_city, c.customer_state, o.order_purchase_timestamp,
3 extract(hour from o.order_purchase_timestamp) as hr,
4 case when extract(hour from o.order_purchase_timestamp) between 0 and 6
5 then 'Dawn'
6 when extract(hour from o.order_purchase_timestamp) between 7 and 12
7 then 'Mornings'
8 when extract(hour from o.order_purchase_timestamp) between 13 and 18
9 then 'Afternoon'
10 else 'Night'
11 End as order_period
12 from 'targetsql-430317.target_sql.customers' as c
13 join 'targetsql-430317.target_sql.orders' as o on
14 c.customer_id = o.customer_id
15 ) as orders_placed_period
16 group by order_period
```

The query results are displayed in a table with columns 'order_period' and 'no_of_orders_during'. The results show that the highest number of orders are placed during the 'Afternoon' period.

Row	order_period	no_of_orders_during
1	Night	28331
2	Afternoon	38135
3	Mornings	27733
4	Dawn	5242

Inference: From the output it is observed that they placed maximum orders in afternoon i.e. between 7 to 12.

Q.3

1.

The screenshot shows a BigQuery interface with a query editor and a results table. The query is as follows:

```
1 select distinct purchase_month, customer_state,
2 count(order_id) over(partition by purchase_month, customer_state)as no_of_orders
3 from
4 (select c.customer_id, c.customer_state, o.order_purchase_timestamp,o.order_id,
5 format_date("%B %Y",o.order_purchase_timestamp) as purchase_month
6 from `targetsql-430317.target_sql.orders` o
7 join `targetsql-430317.target_sql.customers` c
8 on o.customer_id = c.customer_id
9 )
10 order by purchase_month
```

The results table shows the following data:

Row	purchase_month	customer_state	no_of_orders
1	April 2017	SC	105
2	April 2017	CE	43
3	April 2017	PR	114
4	April 2017	MG	275
5	April 2017	DF	35
6	April 2017	TO	14
7	April 2017	MT	27

Results per page: 50 1 - 50 of 565

Inference: Maximum orders placed in Jan 2018 as 7269

2.

The screenshot shows a BigQuery interface with a query editor and a results table. The query is as follows:

```
1 select customer_state, count(*) as no_of_customers from `targetsql-430317.target_sql.customers`
2 group by customer_state
3 order by no_of_customers desc
```

The results table shows the following data:

Row	customer_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
11	PE	1652

Results per page: 50 1 - 27 of 27

Inference: The maximum customers are from SP state.

Q.4

1.

TargetSQL Search (/) for resources, docs, products, and more Search

ling to upgrade to the full BigQuery experience. [Learn more](#) DISMISS UP

+ ADD K

sources ?

Untitled query RUN SAVE DOWNLOAD SHARE SCHEDULE MORE This query will process 8.14 ME

```
1 with cost_2017 as
2   (select sum(p.payment_value) as payment_cost_2017
3     from `targetsql-438317.target_sql.orders` as o
4     join `targetsql-438317.target_sql.payments` p
5       on o.order_id = p.order_id
6     where o.order_purchase_timestamp between "2017-01-01" and "2017-08-31"
7   ),
8   cost_2018 as
9     (select sum(p.payment_value) as payment_cost_2018
10       from `targetsql-438317.target_sql.orders` as o
11       join `targetsql-438317.target_sql.payments` p
12         on o.order_id = p.order_id
13       where o.order_purchase_timestamp between "2018-01-01" and "2018-08-31"
14     ),
15   percent_change as
16     (select cost_2017, cost_2018,
17       ((payment_cost_2018 - payment_cost_2017)*100/payment_cost_2017) as percentage_change
18     from cost_2017, cost_2018
19   )
20
21 select * from percent_change
```

Query results SAVE RESULTS EXPLORE DATA

Row	payment_cost_2017	payment_cost_2018	percentage_change
1	3645107.270000...	8694669.949999...	138.5298787105...

Inference: The payment value from 2017 to 2018 is increased by 138% .

2.

TargetSQL Search (/) for resources, docs, products, and more Search

illing to upgrade to the full BigQuery experience. [Learn more](#) DI

+ ADD K

sources ?

Untitled query RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

```
1 select distinct customer_state, Total_payment_value, Avg_payment_value
2 from
3   (select c.customer_id, c.customer_state, o.order_purchase_timestamp, p.payment_value,
4     sum(payment_value) over(partition by customer_state) as Total_payment_value,
5     round(avg(payment_value) over(partition by customer_state),4) as Avg_payment_value
6     from `targetsql-438317.target_sql.customers` as c
7     join `targetsql-438317.target_sql.orders` o
8       on c.customer_id = o.customer_id
9     join `targetsql-438317.target_sql.payments` p
10       on o.order_id = p.order_id
11   )
12 order by customer_state
```

Query results SAVE RESULTS EX

Row	customer_state	Total_payment_value	Avg_payment_value
1	AC	19680.62	234.2931
2	AL	96962.06	227.0774
3	AM	27966.93	181.6034
4	AP	16262.8	232.3257
5	BA	616645.82	170.816
6	CE	279464.03	199.9027

Results per page: 50 1 - 27 of 27

Inference: Maximum order cost is for the state SP.

3.

TargetSQL Search (/) for resources, docs, products, and more Search

ling to upgrade to the full BigQuery experience. [Learn more](#)

Untitled query

```
1 select distinct customer_state, Total_freight_value, Avg_freight_value
2 from
3 (select c.customer_id, c.customer_state, o.order_purchase_timestamp, oi.freight_value,
4      sum(freight_value) over(partition by customer_state) as Total_freight_value,
5      round(avg(freight_value) over(partition by customer_state),4) as Avg_freight_value
6      from `targetsql-430317.target_sql.customers` as c
7      join `targetsql-430317.target_sql.orders` o
8      on c.customer_id = o.customer_id
9      join `targetsql-430317.target_sql.order_items` oi
10     on o.order_id = oi.order_id
11 )
12 order by customer_state
```

Query results

Row	customer_state	Total_freight_value	Avg_freight_value
1	AC	3686.75	40.0734
2	AL	15914.59	35.8437
3	AM	5478.89	33.2054
4	AP	2788.5	34.0061
5	BA	100156.68	26.364
6	CE	48351.59	32.7142

Inference: Max. fright value is for the state SP.

Q.5

1.

TargetSQL Search (/) for resources, docs, products, and more Search

illing to upgrade to the full BigQuery experience. [Learn more](#)

Untitled query

```
1 select order_purchase_timestamp,
2        order_delivered_customer_date, order_estimated_delivery_date,
3        date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as Delivery_time,
4        date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as Diff_in_estimated_and_actual_delivery
5 from `targetsql-430317.target_sql.orders`
```

Query results

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	Delivery_time	Diff_in_estimated_and_actual_delivery
1	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	12
2	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	-28
3	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	-16
4	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	-1
5	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	-1
7	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	4
8	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	4
9	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	1
10	2017-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	5

Inference: If the difference between the estimated and the actual delivery time (last column in the result) is negative, it means the order is delivered before the stimated time.

Q.5

2. i) Top 5 states with highest average freight value

TargetSQL Search (/) for resources, docs, products, and more Search

billing to upgrade to the full BigQuery experience. [Learn more](#)

Untitled query RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

```
1 select * from
2 (select distinct customer_state, Avg_freight_value,
3    dense_rank() over(order by Avg_freight_value desc) as rank_
4   from
5    (select c.customer_id, c.customer_state, o.order_purchase_timestamp, oi.freight_value,
6       round(avg(freight_value) over(partition by customer_state),4) as Avg_freight_value,
7       from `targetsql-438317.target_sql.customers` as c
8       join `targetsql-438317.target_sql.orders` o
9       on c.customer_id = o.customer_id
10      join `targetsql-438317.target_sql.order_items` oi
11      on o.order_id = oi.order_id
12     )
13  ) x
14 where x.rank_ <=5
15 order by x.rank_
```

Query results

JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EXECUTION GRAPH

Row	customer_state	Avg_freight_value	rank_
1	RR	42.9844	1
2	PB	42.7238	2
3	RO	41.0697	3
4	AC	40.0734	4
5	PI	39.148	5

2. ii) Top 5 states with lowest average freight value

TargetSQL Search (/) for resources, docs, products, and more Search

ing to upgrade to the full BigQuery experience. [Learn more](#)

Untitled query RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

```
1 select * from
2 (select distinct customer_state, Avg_freight_value,
3    dense_rank() over(order by Avg_freight_value desc) as rank_
4   from
5    (select c.customer_id, c.customer_state, o.order_purchase_timestamp, oi.freight_value,
6       round(avg(freight_value) over(partition by customer_state),4) as Avg_freight_value,
7       from `targetsql-438317.target_sql.customers` as c
8       join `targetsql-438317.target_sql.orders` o
9       on c.customer_id = o.customer_id
10      join `targetsql-438317.target_sql.order_items` oi
11      on o.order_id = oi.order_id
12     )
13  ) x
14 where x.rank_ <=5
15 order by x.rank_
```

Query results

JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EXECUTION GRAPH

Row	customer_state	Avg_freight_value	rank_
1	RR	42.9844	1
2	PB	42.7238	2
3	RO	41.0697	3
4	AC	40.0734	4
5	PI	39.148	5

3. i) Top 5 states with highest average delivery time

TargetSQL Search (/) for resources, docs, products, and more

ling to upgrade to the full BigQuery experience. [Learn more](#)

DISMISS UPGRADE

Untitled query

```
1 select *from
2 (select distinct customer_state, avg_delivery_time,
3  dense_rank() over(order by avg_delivery_time desc) as rank_
4  from(select c.customer_id,c.customer_state, o.order_purchase_timestamp, o.order_delivered_customer_date,
5   o.order_estimated_delivery_date,
6   date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day) as Delivery_time,
7   round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)) over(partition by c.customer_state),4)
8   as avg_delivery_time
9  from `targetsql-438317.target_sql.customers` c
10 join `targetsql-438317.target_sql.orders` o
11 on c.customer_id = o.customer_id) X
12 where x.rank_ <= 5
13 order by x.rank_
```

Query results

JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EXECUTION GRAPH

Row	customer_state	avg_delivery_time	rank_
1	RR	28.9756	1
2	AP	26.7313	2
3	AM	25.9862	3
4	AL	24.0403	4
5	PA	23.3161	5

ii) Top 5 states with lowest average delivery time

TargetSQL Search (/) for resources, docs, products, and more

billing to upgrade to the full BigQuery experience. [Learn more](#)

DISMISS UPGRADE

Untitled query

```
1 select *from
2 (select distinct customer_state, avg_delivery_time,
3  dense_rank() over(order by avg_delivery_time asc) as rank_
4  from(select c.customer_id,c.customer_state, o.order_purchase_timestamp, o.order_delivered_customer_date,
5   o.order_estimated_delivery_date,
6   date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day) as Delivery_time,
7   round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)) over(partition by c.customer_state),4)
8   as avg_delivery_time
9  from `targetsql-438317.target_sql.customers` c
10 join `targetsql-438317.target_sql.orders` o
11 on c.customer_id = o.customer_id) X
12 where x.rank_ <= 5
13 order by x.rank_
```

Query results

JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EXECUTION GRAPH

Row	customer_state	avg_delivery_time	rank_
1	SP	8.2981	1
2	PR	11.5267	2
3	MG	11.5438	3
4	DF	12.5091	4
5	SC	14.4796	5

Q.5

4. The estimated delivery time will be the duration from the order placed date to the expected delivery date.

The delivery time will be the duration from the order placed date to the actual order delivery date.


```
1 select * from
2     (select customer_state, dense_rank() over(order by delivery_diff desc) as fast_delivery_rank
3      from
4          (select customer_state, (avg_estimated_delivery_time - avg_delivery_time) as delivery_diff from
5           (select distinct customer_state, avg_estimated_delivery_time, avg_delivery_time
6            from
7                (select c.customer_id, c.customer_state, o.order_delivered_customer_date, o.order_estimated_delivery_date,
8                 round(avg(date_diff(o.order_estimated_delivery_date, o.order_purchase_timestamp, day)) over(partition by c.customer_state), 4) as
9                  avg_estimated_delivery_time,
10                 round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) over(partition by c.customer_state), 4) as avg_delivery_time
11                  from "targetsql-430317.target_sql.customers" c
12                  join "targetsql-430317.target_sql.orders" o
13                  on c.customer_id = o.customer_id
14                 )
15             )
16          )x
17 where x.fast_delivery_rank <= 5
18 order by x.fast_delivery_rank
```

Q.6

+ ADD

IK

< d query

✕

Q *Untitled query

✕

Q *Untitled query

✕

Q *Untitled query

✕

Q *Untitled query

✕

Q *Untitled query

✕

Q *Untitled query

✕

>

+

Untitled query

▶ RUN

▶ SAVE

▶ DOWNLOAD

▶ SHARE

▶ SCHEDULE

▶ MORE

✔ This query will pro

```

1 select distinct purchase_month,payment_type,
2 count(order_id) over(partition by payment_type) as no_of_orders
3 from
4 (select p.order_id, p.payment_type, p.payment_installments,
5 format_date("%B %Y",o.order_purchase_timestamp) as purchase_month from targetsql-430317.target_sql.payments p
6 join targetsql-430317.target_sql.orders o
7 on p.order_id = o.order_id
8 )
9 order by purchase_month

```

Press Alt+F

Query results

▶ SAVE RESULTS

▶ EXPL

JOB INFORMATION

RESULTS

CHART

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	purchase_month	payment_type	no_of_orders
1	April 2017	UPI	19784
2	April 2017	credit_card	76795
3	April 2017	voucher	5775
4	April 2017	debit_card	1529
5	April 2018	voucher	5775
6	April 2018	debit_card	1529
7	April 2018	UPI	19784

Results per page: 50 1 - 50 of 90

2.

Untitled query

Run

Save

Download

Share

Schedule

More

Query completed

```
1 select count(*) as no_of_orders_with_payment_installments
2   from
3     (select * from
4       (select p.order_id, p.payment_type, p.payment_installments,
5        from targetsql-438317.target_sql.payments p
6        join targetsql-438317.target_sql.orders o
7          on p.order_id = o.order_id
8       )
9      where payment_installments > 1
10     )
11
```

Press Alt+F1 for Accessibility O

Query results

Save Results

Explore Data

JOB INFORMATION

RESULTS

CHART

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	no_of_orders_with_p
1	51338

Inference: Maximum orders are placed on EMI (payment_installment) basis.

Detailed Report:

Tha analysis report of Target company which provides E-commerce service in Brazil from September 2016 to October 2018 has following highlighted points:

- Target company provides E-commerce service in Brazil across 27 states and 4119 cities from September 2016.
- The number of orders are increasing year on year so there is a growing trend.
- Maximum orders are placed in afternoon i.e. from 13:00hr to 18:00hr
- Maximum orders are placed in SP state as 41746 and minimum orders placed as 46 in RR state.
- Freight value is more than the product price which is not preferred. So the company has to take necessary action in order to reduce freight value to increase the sales.
- As the state SP has lowest average delivery time around 8 days, so there are maximum customers and orders, consequently the sale.
- But in the RR state the average delivery time is highest which is around 29 days which is too high. So this can be one of the reason to have less sale in RR state. Company has to increase the logistic network in this state in order to shorten the delivery timelines.
- Around 50% customers are using the installment facility.

Company has to take below actions in order to increase revenue:

- Reducing freight value
- Reducing delivery time

