

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
 1. Data type of columns in a table

```
select * from `Project.orders`
```

Query results

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date
1	7a4df52b0cf14090e541401a20a...	725e9c75605414021f8c3d5a...	created	2017-11-25 11:10:33 UTC	null	null	null
2	35de4050381cfc0544cdd86f4...	4ee64f4bfc542546422da0aeb...	created	2017-12-05 01:07:58 UTC	null	null	null
3	b5399909123f6c3503db0cfe...	43944954f6980d107b764371...	created	2017-12-05 01:07:52 UTC	null	null	null
4	dba5062bda3af4fbcc33b1e04...	964a6ff349bdf60f63a7682b69...	created	2018-02-06 17:21:04 UTC	null	null	null
5	90ab3a7d50544ec7bc3363c82...	7661b94f216052b6a64f2269c...	created	2017-11-06 13:12:34 UTC	null	null	null
6	f6c5dad1b0e1f8c3cc5c90e3...	9f2372a1e49340278e7c1ef9...	shipped	2017-04-20 12:45:34 UTC	2017-04-22 09:10:13 UTC	2017-04-24 11:31:17 UTC	null
7	1d02775799eecd9d08903425...	1240c3e54601d686b6a3a37...	shipped	2017-07-13 11:03:05 UTC	2017-07-13 11:10:22 UTC	2017-07-18 18:17:30 UTC	null
8	6190a94657e1012983a27408...	5fc4c97dc36393f996714524...	shipped	2017-07-11 13:36:30 UTC	2017-07-11 13:45:15 UTC	2017-07-13 17:55:46 UTC	null
9	58ce513e55c740a3a81e8c807...	53004104769d9a9cc951d856...	shipped	2017-07-29 18:05:07 UTC	2017-07-29 18:15:17 UTC	2017-07-31 16:41:59 UTC	null
10	088683795a3d303d61152c4f...	58d891d1863819f9b040734f...	shipped	2017-07-13 10:02:47 UTC	2017-07-14 02:25:54 UTC	2017-07-20 20:02:58 UTC	null

I have used one table name just to show the data inside it

2. Time period for which the data is given

The time period for which the data is available in the orders.csv file

```
SELECT MIN(order_purchase_timestamp) as first_order , MAX(order_purchase_timestamp) as last_order
```

```
FROM `Project.orders`
```

Query results

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

3. Cities and States of customers ordered during the given period

```
SELECT DISTINCT c.customer_city, c.customer_state
FROM `Project.orders` as o join `Project.customers` as c
on o.customer_id=c.customer_id
WHERE order_purchase_timestamp BETWEEN '2016-09-04 21:15:19 UTC' AND '2018-10-17 17:30:18 UTC'
```

The screenshot shows the Google Cloud BigQuery console. The query results table has the following data:

Row	customer_city	customer_state
1	acu	RN
2	roo	CE
3	ipe	RS
4	ipu	CE
5	ita	AC
6	itu	SP
7	jau	SP
8	luz	MS
9	poa	SP
10	uba	MS

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?

```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
COUNT(*) AS num_orders
FROM Project.orders
GROUP BY year, month
ORDER BY year, month
```

The screenshot shows the Google Cloud BigQuery console with the following query results table:

Row	year	month	num_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	990
5	2017	2	1790
6	2017	3	2680
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245

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

```
SELECT
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 5 THEN 'Dawn'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 AND 11 THEN 'Morning'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 12 AND 17 THEN 'Afternoon'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 18 AND 23 THEN 'Night'
ELSE 'Unknown'
```

```

END AS purchase_time,
COUNT(DISTINCT order_id) AS total_orders
FROM
Project.orders
GROUP By purchase_time

```

The screenshot shows the Google Cloud BigQuery console. The query editor contains the following SQL:

```

1 SELECT
2 CASE
3   WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 5 THEN 'Dawn'
4   WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 AND 11 THEN 'Morning'
5   WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 12 AND 17 THEN 'Afternoon'
6   WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 18 AND 23 THEN 'Night'
7   ELSE 'Unknown'
8 END AS purchase_time,
9 COUNT(DISTINCT order_id) AS total_orders
10 FROM
11 Project.orders
12 GROUP BY purchase_time

```

The query results are displayed in a table with the following data:

Row	purchase_time	total_orders
1	Morning	22240
2	Dawn	4740
3	Afternoon	38361
4	Night	34100

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

1. Get month on month orders by states

```

with a as (select customer_state, extract (year from order_purchase_timestamp) years,
extract (month from order_purchase_timestamp) months, count(order_id) counts
from `Project.orders` o
join Project.customers c on o.customer_id=c.customer_id
group by customer_state, order_purchase_timestamp,
extract(month from order_purchase_timestamp)
order by customer_state, years asc, months asc)
SELECT customer_state, years, months, counts-lag(counts,1)
over(partition by customer_state order by years asc, months asc) count
from a
order by customer_state asc, years asc, months asc

```

The screenshot shows the Google Cloud BigQuery console. The query editor contains the following SQL:

```

1 with a as (select customer_state, extract (year from order_purchase_timestamp) years,
2 extract (month from order_purchase_timestamp) months, count(order_id) counts
3 from `Project.orders` o
4 join Project.customers c on o.customer_id=c.customer_id
5 group by customer_state, order_purchase_timestamp,
6 extract(month from order_purchase_timestamp)
7 order by customer_state, years asc, months asc)
8 SELECT customer_state, years, months, counts-lag(counts,1)
9 over(partition by customer_state order by years asc, months asc) count
10 from a
11 order by customer_state asc, years asc, months asc
12

```

The query results are displayed in a table with the following data:

Row	customer_state	years	months	count
1	AC	2017	1	null
2	AC	2017	1	0
3	AC	2017	2	0
4	AC	2017	2	0
5	AC	2017	2	0

2. Distribution of customers across the states in Brazil

```
SELECT customer_state, COUNT(DISTINCT customer_unique_id) as customer_count
FROM Project.customers
GROUP BY customer_state
ORDER BY customer_state
```

The screenshot shows the Google Cloud BigQuery console. The query editor on the right contains the following SQL:

```
1 SELECT customer_state, COUNT(DISTINCT customer_unique_id) as customer_count
2 FROM Project.customers
3 GROUP BY customer_state
4 ORDER BY customer_state
5
```

The query results are displayed in a table with the following data:

Row	customer_state	customer_count
1	AC	77
2	AL	401
3	AM	143
4	AP	67
5	BA	3277
6	CE	1313
7	DF	2075
8	ES	1964
9	GO	1952

4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.
 1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table

```
SELECT
  ROUND((((SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2018
    AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
    THEN payment_value ELSE 0 END) - SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017
    AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
    THEN payment_value ELSE 0 END)) / SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017
    AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
    THEN payment_value ELSE 0 END)) * 100, 2) AS cost_increase_percentage
FROM `Project.orders` o
JOIN `Project.payments` p ON o.order_id = p.order_id
WHERE o.order_status = 'delivered'
```

Business Case: Target SQL - Probi x Query results - BigQuery - Scaler x

console.cloud.google.com/bigquery?project=scaler-dsml-sql-377817&ws=1m40!1m4!1m3!1sscml-sql-377817!2sbqjob_6c523480_186...

InfyTQ

Start your Free Trial with \$300 in credit. Don't worry--you won't be charged if you run out of credits. [Learn more](#)

Google Cloud Scaler-DSML-SQL Search (/) for resources, docs, products, and more Search

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

Explorer + ADD DATA

Viewing all resources. [Show starred resources](#)

scaler-dsml-sql-377817

Query Editor: "Unsaved query 8"

```

1 SELECT
2   ROUND(((SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2018
3     AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
4     THEN payment_value ELSE 0 END) - SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017
5     AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
6     THEN payment_value ELSE 0 END)) / SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017
7     AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
8     THEN payment_value ELSE 0 END)) * 100, 2) AS cost_increase_percentage
9 FROM `Project.orders` o
10 JOIN `Project.payments` p ON o.order_id = p.order_id
11 WHERE o.order_status = 'delivered'
12

```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	cost_increase_percentage
1	143.33

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

```

select c.customer_state,
       sum(oi.price) as total_price,
       avg(oi.price) as average_price,
       sum(oi.freight_value) as total_freight,
       avg(oi.freight_value) as average_freight
from `Project.orders` as o
join `Project.customers` as c on o.customer_id=c.customer_id
join `Project.order_items` as oi on o.order_id=oi.order_id
group by c.customer_state
order by total_price desc

```

Query Editor: "Unsaved query 8"

```

1 select c.customer_state,
2       sum(oi.price) as total_price,
3       avg(oi.price) as average_price,
4       sum(oi.freight_value) as total_freight,
5       avg(oi.freight_value) as average_freight
6 from `Project.orders` as o
7 join `Project.customers` as c on o.customer_id=c.customer_id
8 join `Project.order_items` as oi on o.order_id=oi.order_id
9 group by c.customer_state
10 order by total_price desc
11

```

Query results

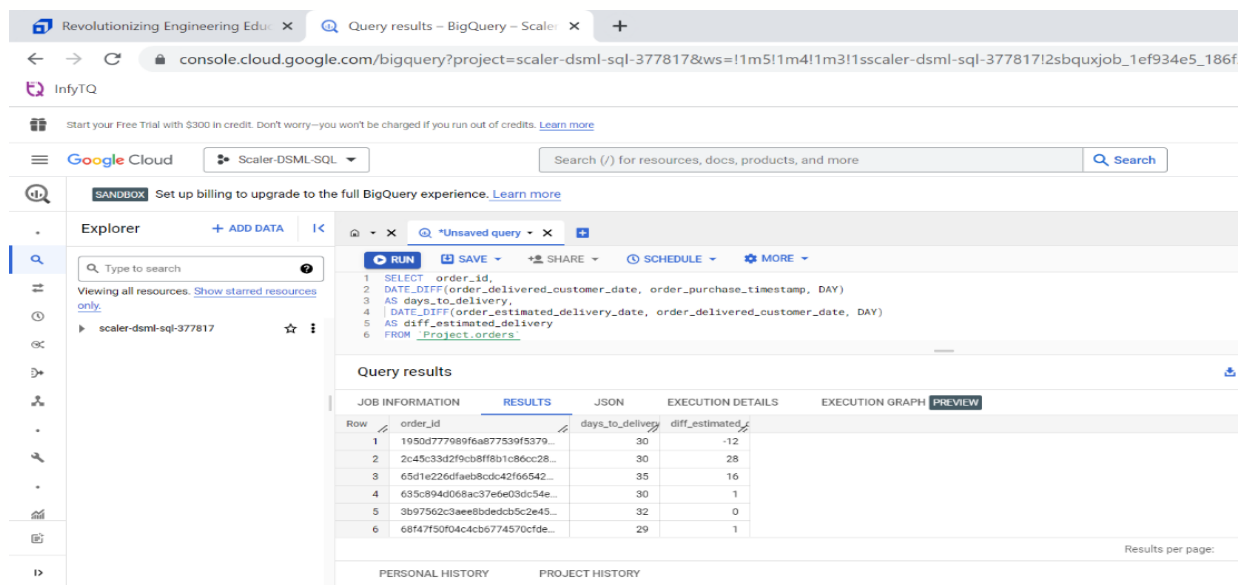
JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	customer_state	total_price	average_price	total_freight	average_freight
1	SP	5202955.05...	109.653629...	718723.069...	15.1472753...
2	RJ	1824092.66...	125.117818...	305589.310...	20.9609239...
3	MG	1585308.02...	120.748574...	270853.460...	20.6301668...
4	RS	750304.020...	120.337453...	135522.740...	21.7358043...
5	PR	683083.760...	119.004139...	117851.680...	20.5316515...
6	SC	520553.340...	124.653577...	89660.2600...	21.4703667...

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
SELECT order_id,  
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)  
AS days_to_delivery,  
DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)  
AS diff_estimated_delivery  
FROM `Project.orders`
```



The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with the Google Cloud logo and a search bar. Below that, the 'Explorer' panel on the left shows the project 'scaler-dsml-sql-377817'. The main area displays a SQL query and its results. The query is the same as the one in the previous block. The results are shown in a table with columns: Row, order_id, days_to_delivery, and diff_estimated_delivery. There are 6 rows of data.

Row	order_id	days_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f4750f04c4cb6774570cfd...	29	1

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- $\text{time_to_delivery} = \text{order_purchase_timestamp} - \text{order_delivered_customer_date}$
- $\text{diff_estimated_delivery} = \text{order_estimated_delivery_date} - \text{order_delivered_customer_date}$

```
SELECT order_id,  
TIMESTAMP_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY)  
AS time_to_delivery,  
TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)  
AS diff_estimated_delivery  
FROM `Project.orders`
```

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud Scaler-DSML-SQL Search (/) for resources, docs, products, and more

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

Explorer + ADD DATA

Q Type to search

Viewing all resources. [Show starred resources only.](#)

scaler-dsml-sql-377817

Q *Unsaved query 2

RUN SAVE SHARE SCHEDULE MORE

```

1 SELECT order_id,
2    TIMESTAMP_DIFF(order_purchase_timestamp, order_delivered_customer_date, DAY)
3    AS time_to_delivery,
4    TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)
5    AS diff_estimated_delivery
6 FROM Project.orders

```

Query results

Row	order_id	time_to_delivery	diff_estimated
1	1950d777989f6a877539f5379...	-30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	-30	28
3	65d1e226dfaeb8cdc42f66542...	-35	16
4	635c994d068ac37e6e03dc54e...	-30	1
5	3b97562c3aee8bdedcb5c2e45...	-32	0
6	68f47f50f04c4cb6774570cfe...	-29	1

PERSONAL HISTORY PROJECT HISTORY

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

SELECT

c.customer_state,
 AVG(o.freight_value) AS avg_freight_value,
 AVG(EXTRACT(DAY FROM o.order_delivered_customer_date) - EXTRACT(DAY FROM o.order_purchase_timestamp)) AS avg_time_to_delivery,
 AVG(EXTRACT(DAY FROM o.order_estimated_delivery_date) - EXTRACT(DAY FROM o.order_delivered_customer_date)) AS avg_diff_estimated_delivery

FROM

Project.customers c
 JOIN Project.orders o ON c.customer_id = o.customer_id
 JOIN Project.order_items oi ON o.order_id = oi.order_id

GROUP BY

c.customer_state

Revolutionizing Engineering Edu... Query results - BigQuery - Scaler-DSML-SQL

console.cloud.google.com/bigquery?project=scaler-dsml-sql-377817&ws=1m151m41m311sscaler-dsml-sql-3778172sbquxjob1ef934e5_186...

InfyTQ

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud Scaler-DSML-SQL Search (/) for resources, docs, products, and more

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

Explorer + ADD DATA

Q Type to search

Viewing all resources. [Show starred resources only.](#)

scaler-dsml-sql-377817

Q *Unsaved query 3

RUN SAVE SHARE SCHEDULE MORE

```

1 SELECT
2   c.customer_state,
3   AVG(o.freight_value) AS avg_freight_value,
4   AVG(EXTRACT(DAY FROM o.order_delivered_customer_date) - EXTRACT(DAY FROM o.order_purchase_timestamp)) AS avg_time_to_delivery

```

Query results

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated
1	MT	28.1662843...	0.21311475...	-0.0356798...
2	MA	38.2570024...	0.12124999...	-0.1012500...
3	AL	35.8436711...	0.03512980...	-0.1615925...
4	SP	15.1472753...	0.47630428...	-0.2987102...
5	MG	20.6301668...	0.25741271...	-0.1535186...
6	PE	32.9178626...	1.32331500...	-0.8430698...
7	RJ	20.9609229...	0.55061501...	-0.4597766...

Results per page: 50 1 - 27 of 27

PERSONAL HISTORY PROJECT HISTORY

4. Sort the data to get the following:

5. 1.Top 5 states with highest average freight value in desc

```
SELECT c.customer_state,  
       AVG(oi.freight_value) as avg_freight_value  
FROM Project.customers c  
JOIN Project.orders o on c.customer_id = o.customer_id  
JOIN Project.order_items oi on o.order_id = oi.order_id  
GROUP BY c.customer_state  
ORDER BY avg_freight_value desc  
LIMIT 5;
```

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the SQL query for finding the top 5 states with the highest average freight value. The query results are displayed in a table with columns 'customer_state' and 'avg_freight_value'.

Row	customer_state	avg_freight_value
1	RR	42.9844230...
2	PB	42.7238039...
3	RO	41.0697122...
4	AC	40.0733695...
5	PI	39.1479704...

2.Top 5 states with lowest average freight value in asc

```
SELECT c.customer_state,  
       AVG(oi.freight_value) AS avg_freight_value  
FROM Project.customers c  
JOIN Project.orders o ON c.customer_id = o.customer_id  
JOIN Project.order_items oi ON o.order_id = oi.order_id  
GROUP BY c.customer_state  
ORDER BY avg_freight_value ASC  
LIMIT 5;
```


Google Cloud

Scaler-DSML-SQL

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

Search

SANDBOX

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

Explorer

+ ADD DATA

<

Viewing all resources. [Show starred resources only.](#)

scaler-dsml-sql-377817

☆

⋮

1

SELECT c.customer_state,

2

AVG(o1.freight_value) AS avg_freight_value

3

FROM Project.customers c

4

JOIN Project.orders o ON c.customer_id = o.customer_id

5

JOIN Project.order_items o1 ON o.order_id = o1.order_id

6

GROUP BY c.customer_state

7

ORDER BY avg_freight_value ASC

8

LIMIT 5;

^

Query results

SAVE RESULTS

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	customer_state	avg_freight_valu
1	SP	15.1472753...
2	PR	20.5316515...
3	MG	20.6301668...
4	RJ	20.9609239...
5	DF	21.0413549...

PERSONAL HISTORY

PROJECT HISTORY

6. Top 5 states with highest/lowest average time to delivery

6.1 Top 5 states with highest average time to delivery

```

SELECT c.customer_state,
AVG(DATE_DIFF(o.order_purchase_timestamp, o.order_delivered_customer_date, DAY)) AS avg_time_to_delivery
FROM Project.customers c
JOIN Project.orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY avg_time_to_delivery DESC
LIMIT 5;

```

InfyTQ

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud

Scaler-DSML-SQL

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

Search

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

Explorer

ADD DATA

Viewing all resources. [Show starred resources only.](#)

scaler-dsml-sql-377817

1 SELECT c.customer_state,
2 AVG(DATE_DIFF(o.order_purchase_timestamp, o.order_delivered_customer_date, DAY)) AS avg_time_to_delivery
3 FROM Project.customers c
4 JOIN Project.orders o ON c.customer_id = o.customer_id
5 GROUP BY c.customer_state
6 ORDER BY avg_time_to_delivery DESC
7 LIMIT 5;
8

Query results

JOB INFORMATIONRESULTSJSONEXECUTION DETAILSEXECUTION GRAPHPREVIEW

Row	customer_state	avg_time_to_dgl
1	SP	-8.2980614...
2	PR	-11.526711...
3	MG	-11.543813...
4	DF	-12.509134...
5	SC	-14.479560...

PERSONAL HISTORYPROJECT HISTORY

6.2 Top 5 states with lowest average time to delivery

```

SELECT c.customer_state,
AVG(DATE_DIFF(o.order_purchase_timestamp, o.order_delivered_customer_date, DAY)) AS avg_time_to_delivery
FROM Project.customers c
JOIN Project.orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY avg_time_to_delivery ASC
LIMIT 5;

```

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the following SQL query:

```

1 SELECT c.customer_state,
2 AVG(DATE_DIFF(o.order_purchase_timestamp, o.order_delivered_customer_date, DAY)) AS avg_time_to_delivery
3 FROM Project.customers c
4 JOIN Project.orders o ON c.customer_id = o.customer_id
5 GROUP BY c.customer_state
6 ORDER BY avg_time_to_delivery ASC
7 LIMIT 5;

```

The query results are displayed in a table with the following data:

Row	customer_state	avg_time_to_delivery
1	RR	-28.975609...
2	AP	-26.731343...
3	AM	-25.986206...
4	AL	-24.040302...
5	PA	-23.316067...

7. Top 5 states where delivery is really fast/ not so fast compared to estimated date

7.1 Top 5 states where delivery is really fast as compared to estimated date

```

select c.customer_state,
avg(date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) as avg_delivery_diff
FROM Project.customers c
join Project.orders o ON c.customer_id = o.customer_id
group by c.customer_state
having avg_delivery_diff < 0
order by avg_delivery_diff ASC
limit 5;

```

Revolutionizing Engineering Edu... Query results - BigQuery - Scale...

console.cloud.google.com/bigquery?project=scaler-dsml-sql-377817&ws=!1m40!1m4!1m3!1ssc... InfyTQ

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud Scaler-DSML-SQL Search (/) for resources, docs, products, and more Search

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

Explorer + ADD DATA

Type to search

Viewing all resources. [Show starred resources only.](#)

scaler-dsml-sql-377817

nsaved query 2 - x *Unsaved query 3 - x *Unsaved query 4 - x *Unsaved query 5 - x *Unsaved query 6 - x *Unsaved query 7 - x

RUN SAVE SHARE SCHEDULE MORE

```

1 select c.customer_state,
2 avg(date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) as avg_delivery_diff
3 FROM Project.customers c
4 join Project.orders o ON c.customer_id = o.customer_id
5 group by c.customer_state
6 having avg_delivery_diff < 0
7 order by avg_delivery_diff ASC
8 limit 5;

```

Query results

Row	customer_state	avg_delivery_diff
1	AC	-19.762500...
2	RO	-19.131687...
3	AP	-18.731343...
4	AM	-18.606896...
5	RR	-16.414634...

PERSONAL HISTORY PROJECT HISTORY

7.2 Top 5 states where delivery is really not so fast as compared to estimated date

```

select c.customer_state,
avg(date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS avg_delivery_diff
FROM Project.customers c
join Project.orders o ON c.customer_id = o.customer_id
group by c.customer_state
having avg_delivery_diff > 0
order by avg_delivery_diff DESC
limit 5;

```

Revolutionizing Engineering Edu... Query results - BigQuery - Scale...

console.cloud.google.com/bigquery?project=scaler-dsml-sql-377817&ws=!1m45!1m4!1m3!1ssc... InfyTQ

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud Scaler-DSML-SQL Search (/) for resources, docs, products, and more Search

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

Explorer + ADD DATA

Type to search

Viewing all resources. [Show starred resources only.](#)

scaler-dsml-sql-377817

nsaved query 3 - x *Unsaved query 4 - x *Unsaved query 5 - x *Unsaved query 6 - x *Unsaved query 7 - x *Unsaved query 8 - x

RUN SAVE SHARE SCHEDULE MORE

```

1 select c.customer_state,
2 avg(date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS avg_delivery_diff
3 FROM Project.customers c
4 join Project.orders o ON c.customer_id = o.customer_id
5 group by c.customer_state
6 having avg_delivery_diff > 0
7 order by avg_delivery_diff desc
8 limit 5;

```

Query results

Row	customer_state	avg_delivery_diff
1	AL	-7.9471032...
2	MA	-8.7684797...
3	SE	-9.1731343...
4	ES	-9.6185463...
5	BA	-9.9348894...

PERSONAL HISTORY PROJECT HISTORY

6. Payment type analysis:

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

```
WITH monthly_order_counts AS
(
  SELECT p.payment_type,
         EXTRACT(YEAR FROM o.order_purchase_timestamp) as year,
         EXTRACT(MONTH FROM o.order_purchase_timestamp) as month,
         o.order_purchase_timestamp,
         COUNT(o.order_id) as count
FROM Project.orders as o join Project.payments as p ON o.order_id = p.order_id
GROUP BY
  p.payment_type,
  EXTRACT(YEAR FROM o.order_purchase_timestamp),
  EXTRACT(MONTH FROM o.order_purchase_timestamp),
  o.order_purchase_timestamp
ORDER BY p.payment_type, year, month
)
SELECT m.payment_type, m.year, m.month, m.order_purchase_timestamp,
       m.count - LAG(m.count, 1) OVER (PARTITION BY m.payment_type ORDER BY m.year asc, m.month
       ASC) as count_change
FROM monthly_order_counts as m
ORDER BY m.payment_type, m.year, m.month
```

The screenshot shows the Google Cloud BigQuery interface. The query editor at the top contains the SQL code for the payment type analysis. Below the editor, the 'Query results' section displays a table with the following data:

Row	payment_type	year	month	order_purchase_timestamp	count_change
1	UPI	2016	10	2016-10-05 01:47:40 UTC	null
2	UPI	2016	10	2016-10-04 15:02:37 UTC	0
3	UPI	2016	10	2016-10-10 09:51:13 UTC	0
4	UPI	2016	10	2016-10-04 19:41:32 UTC	0
5	UPI	2016	10	2016-10-05 11:23:13 UTC	0
6	UPI	2016	10	2016-10-07 08:49:03 UTC	0
7	UPI	2016	10	2016-10-04 13:02:10 UTC	0
8	UPI	2016	10	2016-10-06 12:23:07 UTC	0

The interface also shows a sidebar with the Explorer view, a search bar, and various toolbars for running, saving, and scheduling queries. The bottom of the interface displays 'Results per page: 50' and '1 - 50 of 101214'.

2. Count of orders based on the no. of payment installments

```
SELECT
  payment_installments,
  COUNT(*) AS order_counts
FROM `Project.payments`
GROUP BY
  payment_installments
ORDER BY
  payment_installments
```

The screenshot shows the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, the project name 'Scaler-DSML-SQL', and a search bar. Below the navigation bar, the 'Explorer' panel on the left shows the project structure. The main panel displays a SQL query and its results.

Query:

```
1 SELECT
2   payment_installments,
3   COUNT(*) AS order_count
```

Query results:

Row	payment_installments	order_count
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626