SQL Queries

Note:

- We have used BigQuery for executing the SQL queries
- We have given a brief problem statement before writing the queries
- We have also shown the first few rows that our query returned as result

1. **Question:** *Get the names of unique state-city pairs from customers and sellers table*

```
SELECT
  DISTINCT customer_state, customer_city
FROM `target.customers`
UNION DISTINCT (
  SELECT
    DISTINCT seller_state, seller_city
  FROM `target.sellers`
);
```

Result:

	customer_state	customer_city
1	AC	rio branco
2	AM	manaus
3	BA	bahia
4	BA	ipira
5	BA	irece
6	BA	ilheus
7	BA	guanambi
8	BA	salvador
9	BA	eunapolis
10	BA	barro alto

2. **Question:** *Find the the total number of orders placed month by month over the given timeframe of the data.*

```
WITH base1 AS (
  SELECT
    *, EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month
  FROM `target.orders`
)
SELECT year, month, COUNT(*) AS monthly_sales
FROM base1
GROUP BY 1, 2 ORDER BY 1, 2;
```

Result:

Row	year	month	monthly_sales
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
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631

3. **Question:** *How many and what % of orders were placed during dawn [12AM-6AM), morning [6AM-12PM), afternoon [12PM-6PM) and night [6PM-12AM)?*

```
WITH base1 AS (
  SELECT
    *, EXTRACT(HOUR FROM order_purchase_timestamp) AS hour
  FROM `target.orders`
),
base2 AS (
  SELECT
    *,
    CASE
      WHEN hour BETWEEN 0 AND 5 THEN 1
      WHEN hour BETWEEN 6 AND 11 THEN 2
      WHEN hour BETWEEN 12 AND 17 THEN 3
      WHEN hour BETWEEN 18 AND 23 THEN 4
      ELSE 100
    END AS slot
  FROM base1
)
SELECT
  slot, COUNT(*) AS order_count_per_slot,
  ROUND(100 * (COUNT(*) / 99441), 2) AS perc_orders_per_slot
FROM base2
GROUP BY 1 ORDER BY 1;
```

Result:

JOB INFORMATION | RESULTS | JSON

Row	slot	order_count...	perc_orders...
1	1	4740	4.77
2	2	22240	22.37
3	3	38361	38.58
4	4	34100	34.29

4. **Question:** *Find out the total number of orders placed over time (month by month) across different states in Brazil.*

```
WITH base1 AS (
  SELECT o.order_id, o.customer_id, o.order_purchase_timestamp,
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    c.customer_unique_id, c.customer_state, c.customer_city
  FROM `target.orders` o JOIN `target.customers`c ON o.customer_id = c.customer_id
)
SELECT
  customer_state, year, month, CONCAT(year, '-', month) AS year_month,
  COUNT(*) AS order_count
FROM base1
WHERE
  (year = 2017 AND month BETWEEN 1 AND 12) OR
  (year = 2018 AND month BETWEEN 1 AND 8)
GROUP BY 1, 2, 3 ORDER BY customer_state, year, month
```

Results:

ow	customer_state	year	month	year_month	order_count
1	AC	2017	1	2017-1	2
2	AC	2017	2	2017-2	3
3	AC	2017	3	2017-3	2
4	AC	2017	4	2017-4	5
5	AC	2017	5	2017-5	8
6	AC	2017	6	2017-6	4
7	AC	2017	7	2017-7	5
8	AC	2017	8	2017-8	4
9	AC	2017	9	2017-9	5
10	AC	2017	10	2017-10	6
11	AC	2017	11	2017-11	5
12	AC	2017	12	2017-12	5

5. **Question:** *Show the count of customers in every unique pair of states and cities in Brazil and order the data in descending order of customer count.*

```
SELECT
  customer_state, customer_city,COUNT(*) AS customer_count
FROM `target.customers`
GROUP BY 1, 2 ORDER BY COUNT(*) DESC
```

Result:

Row	customer_state	customer_city	customer_count
1	SP	sao paulo	15540
2	RJ	rio de janeiro	6882
3	MG	belo horizonte	2773
4	DF	brasilia	2131
5	PR	curitiba	1521
6	SP	campinas	1444
7	RS	porto alegre	1379
8	BA	salvador	1245
9	SP	guarulhos	1189
10	SP	sao bernardo do campo	938
11	RJ	niteroi	849

6. **Question:** *Find yearly revenue for the years 2017 and 2018 (and only choose the months Jan-Aug)*

```
WITH base1 AS (
  SELECT order_id, ROUND(SUM(payment_value), 2) AS total_order_price
  FROM `target.payments`
  WHERE payment_value > 0 GROUP BY 1
),
base2 AS (
  SELECT * FROM `target.orders`
  WHERE order_status IN ('invoiced', 'shipped', 'delivered')
),
base3 AS (
  SELECT b1.*, b2.order_purchase_timestamp,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month
  FROM base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
)
SELECT year, ROUND(SUM(total_order_price), 0) AS yearly_revenue
FROM base3
WHERE
    (year = 2017 AND month BETWEEN 1 AND 8) OR
    (year = 2018 AND month BETWEEN 1 AND 8)
GROUP BY 1 ORDER BY year
```

Result:

Row	year	yearly_revenue
1	2017	3540334.0
2	2018	8583918.0

7. **Question:** *Find monthly revenue (Jan-Aug) for the years 2017 and 2018 and calculate the % increment from 2017 to 2018 at the month level*

```
WITH base1 AS (
  SELECT order_id, ROUND(SUM(payment_value), 2) AS total_order_price
  FROM `target.payments`
  WHERE payment_value > 0 GROUP BY 1
),
base2 AS (
  SELECT * FROM `target.orders`
  WHERE order_status IN ('invoiced', 'shipped', 'delivered')
),
base3 AS (
  SELECT b1.*, b2.order_purchase_timestamp,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month
  FROM base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
),
base4 AS (
  SELECT month, year, SUM(total_order_price) AS mon_revenue
  FROM base3
  WHERE
    (year = 2017 AND month BETWEEN 1 AND 8) OR
    (year = 2018 AND month BETWEEN 1 AND 8)
  GROUP BY 1, 2 ORDER BY month, year
),
base5 AS (
  SELECT *,
    LAG(mon_revenue) OVER(PARTITION BY month ORDER BY year) AS prev_revenue
  FROM base4 ORDER BY month
)
SELECT month, mon_revenue AS revenue_2018, prev_revenue AS revenue_2017,
  ROUND(100 * ((mon_revenue - prev_revenue) / prev_revenue), 2) AS
  perc_cost_increase
FROM base5 WHERE year = 2018
```

Result:

Row	month	revenue_2018	revenue_2017	perc_cost_increase
1	1	1097990.9600000002	134491.66999999987	716.4
2	2	978835.47000000207	276888.91999999952	253.51
3	3	1150469.08999999	419780.45000000088	174.06
4	4	1154606.6399999952	399642.97000000061	188.91
5	5	1144824.2799999982	578655.4000000027	97.84
6	6	1020494.2899999963	497162.32000000309	105.26
7	7	1039801.0899999999	577096.29000000376	80.18
8	8	996896.14999999932	656616.08000000182	51.82

8. **Question:** Find the mean & sum of order price and freight value by customer state

```
WITH base1 AS (
  SELECT
    order_id, SUM(price) AS order_price,
    SUM(freight_value) AS order_freight_value
  FROM `target.order_items` GROUP BY 1
),
base2 AS (
  SELECT * FROM `target.orders` WHERE order_status = 'delivered'
)
SELECT
  c.customer_state,
  ROUND(SUM(order_price), 2) AS sum_order_price,
  ROUND(AVG(order_price), 2) AS mean_order_price,
  ROUND(SUM(order_freight_value), 2) AS sum_order_freight,
  ROUND(AVG(order_freight_value), 2) AS mean_order_freight,
FROM
  base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
  JOIN `target.customers`c ON b2.customer_id = c.customer_id
GROUP BY 1
```

Result:

v	customer_state	sum_order_price	mean_order_price	sum_order_freight	mean_order_freight
1	GO	282836.7	144.53	51375.65	26.25
2	SP	5067633.16	125.12	702069.99	17.33
3	RS	728897.47	136.37	132575.32	24.8
4	BA	493584.14	151.59	97553.67	29.96
5	MG	1552481.83	136.73	266409.84	23.46
6	MT	152191.62	171.77	29032.8	32.77
7	RJ	1759651.13	142.48	295750.44	23.95
8	SC	507012.13	142.98	88115.65	24.85
9	SE	56574.19	168.88	13714.94	40.94
10	PE	251889.49	158.12	57082.56	35.83
11	TO	48402.51	176.65	11604.86	42.35
12	CE	219757.38	171.82	46679.39	36.5

9. **Question:** *Calculate:*

- *the difference in days b/w order delivery date and order purchase date(time_to_delivery)*
- *the difference in days b/w order delivery date and expected delivery date(diff_estimated_delivery)*

```
WITH base1 AS
  (SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL)
SELECT order_id, ord_date, delv_date, est_delv_date,
  DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
  DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
FROM base1
```

Result:

v	order_id	ord_date	delv_date	est_delv_date	time_to_delivery	diff_estimated_delivery
1	635c894d...	2017-04-15 ...	2017-05-16 ...	2017-05-18 ...	31	-2
2	3b97562c...	2017-04-14 ...	2017-05-17 ...	2017-05-18 ...	33	-1
3	68f47f50f...	2017-04-16 ...	2017-05-16 ...	2017-05-18 ...	30	-2
4	276e9ec3...	2017-04-08 ...	2017-05-22 ...	2017-05-18 ...	44	4
5	54e1a3c2...	2017-04-11 ...	2017-05-22 ...	2017-05-18 ...	41	4
6	fd04fa410...	2017-04-12 ...	2017-05-19 ...	2017-05-18 ...	37	1
7	302bb810...	2017-04-19 ...	2017-05-23 ...	2017-05-18 ...	34	5
8	66057d37...	2017-04-15 ...	2017-05-24 ...	2017-05-18 ...	39	6
9	19135c94...	2017-07-11 ...	2017-08-16 ...	2017-08-14 ...	36	2
10	4493e45e...	2017-07-11 ...	2017-08-14 ...	2017-08-14 ...	34	0

10. **Question:** *Group data by state and take mean of freight_value, time_to_delivery, diff_estimated_delivery*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, AVG(order_freight_val) AS mean_order_freight_val,
  AVG(time_to_delivery) AS mean_time_to_delivery,
  AVG(diff_estimated_delivery) AS mean_diff_estimated_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1
```

Result:

w	customer_state	mean_order_freight_val	mean_time_to_delivery	mean_diff_estimated_delivery
1	GO	26.252248339294837	15.536024527337734	-12.185487991824232
2	SP	17.334796513063754	8.7005729243838132	-11.075542055613214
3	RS	24.805449101796405	15.248502994011989	-13.910366766467066
4	BA	29.961200859950871	19.278562653562592	-10.794533169533183
5	MG	23.463963360930077	11.944953320415706	-13.242998062356881
6	MT	32.768397291196386	18.003386004514677	-14.363431151241524
7	RJ	23.947404048582975	15.23700404858303	-11.761214574898739
8	SC	24.84930908065428	14.902989283699952	-11.503384094754646
9	SE	40.940119402985111	21.462686567164194	-10.020895522388059
10	PE	35.8333709981167	18.395480225988727	-13.29378531073446

11. Question: *Show the Top 5 states with highest mean freight_value*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(order_freight_val), 2) AS mean_order_freight_val
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_order_freight_val DESC LIMIT 5
```

Result:

v	customer_state	mean_order_freight_val
1	PB	48.84
2	RR	48.34
3	RO	46.43
4	AC	45.55
5	PI	42.98

12. Question: *Show the Top 5 states with lowest mean freight_value*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(order_freight_val), 2) AS mean_order_freight_val
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_order_freight_val LIMIT 5
```

Result:

	customer_state	mean_order_freight_val
1	SP	17.33
2	MG	23.46
3	PR	23.49
4	DF	23.86
5	RJ	23.95

13. Question: *Show the Top 5 states with highest mean time_to_delivery (Longest delivery time)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(time_to_delivery), 2) AS mean_time_to_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_time_to_delivery DESC LIMIT 5
```

Result:

v	customer_state	mean_time_to_delivery
1	RR	29.34
2	AP	27.18
3	AM	26.36
4	AL	24.5
5	PA	23.73

14. Question: *Show the Top 5 states with lowest mean time_to_delivery (Shortest delivery time)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state, ROUND(AVG(time_to_delivery), 2) AS mean_time_to_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_time_to_delivery LIMIT 5
```

Result:

v	customer_state	mean_time_to_delivery
1	SP	8.7
2	PR	11.94
3	MG	11.94
4	DF	12.9
5	SC	14.9

15. Question: *Show the Top 5 states with highest mean diff_estimated_delivery (Late delivery)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state,
  ROUND(AVG(diff_estimated_delivery), 2) AS mean_diff_estimated_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_diff_estimated_delivery DESC LIMIT 5
```

Result:

w	customer_state	mean_diff_estimated_delivery
1	AL	-8.71
2	MA	-9.57
3	SE	-10.02
4	ES	-10.5
5	BA	-10.79

16. Question: *Show the Top 5 states with lowest mean diff_estimated_delivery (Early delivery)*

```
WITH base1 AS (
  SELECT order_id, customer_id, order_purchase_timestamp AS ord_date,
    order_delivered_customer_date AS delv_date,
    order_estimated_delivery_date AS est_delv_date
  FROM `target.orders`
  WHERE order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
),
base2 AS (
  SELECT *, DATE_DIFF(DATE(delv_date), DATE(ord_date), DAY) AS time_to_delivery,
    DATE_DIFF(DATE(delv_date), DATE(est_delv_date), DAY) AS diff_estimated_delivery
  FROM base1
),
base3 AS (
  SELECT order_id, ROUND(SUM(freight_value), 2) AS order_freight_val
  FROM `target.order_items` GROUP BY 1
)
SELECT customer_state,
  ROUND(AVG(diff_estimated_delivery), 2) AS mean_diff_estimated_delivery
FROM
  base2 b2 JOIN base3 b3 ON b2.order_id = b3.order_id
  JOIN `target.customers` c ON b2.customer_id = c.customer_id
GROUP BY 1 ORDER BY mean_diff_estimated_delivery LIMIT 5
```

Result:

	customer_state	mean_diff_estimated_delivery
1	AC	-20.72
2	RO	-20.1
3	AP	-19.69
4	AM	-19.57
5	RR	-17.29

17. **Question:** *Show the Month over Month count of orders for different payment types*

```
WITH base1 AS(
SELECT DISTINCT order_id, payment_type
FROM ( SELECT * FROM `target.payments`
  WHERE payment_type NOT IN ('not_defined') AND payment_value > 0)
  ),
base2 AS (
  SELECT * FROM `target.orders`
  WHERE order_status IN ('shipped','invoiced','delivered')
),
base3 AS (
  SELECT payment_type, EXTRACT(YEAR FROM b2.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM b2.order_purchase_timestamp) AS month, COUNT(*) AS
order_count
  FROM base1 b1 JOIN base2 b2 ON b1.order_id = b2.order_id
  GROUP BY 1, 2, 3 ORDER BY 1, 2, 3
)
SELECT *, CONCAT(year, '-', month) AS year_month FROM base3
WHERE (year = 2017 AND month BETWEEN 1 AND 12) OR
  (year = 2018 AND month BETWEEN 1 AND 8)
```

Result:

w	payment_type	year	month	order_count	year_month
1	UPI	2017	1	190	2017-1
2	UPI	2017	2	380	2017-2
3	UPI	2017	3	577	2017-3
4	UPI	2017	4	483	2017-4
5	UPI	2017	5	754	2017-5
6	UPI	2017	6	700	2017-6
7	UPI	2017	7	826	2017-7
8	UPI	2017	8	915	2017-8
9	UPI	2017	9	879	2017-9
10	UPI	2017	10	966	2017-10

18. **Question:** *Show the distribution of payment installments*

```
WITH base1 AS (
  SELECT order_id, SUM(payment_installments) AS total_installments
  FROM `target.payments`
  WHERE payment_type NOT IN ('not_defined') AND payment_installments NOT IN (0)
  GROUP BY 1
)
SELECT total_installments, COUNT(*) AS freq
FROM base1
GROUP BY 1 ORDER BY total_installments
```

Result:

w	total_installments	freq
1	1	46261
2	2	13605
3	3	10709
4	4	7223
5	5	5295
6	6	3967
7	7	1689
8	8	4239
9	9	693
10	10	5224
11	11	129