Question 1:

You have three tables:

Customers table:
https://drive.google.com/file/d/17oRPW3jA2mCsTOXKKt6Xx7wSgD78twq8/view?usp=sharing

Orders table:
https://drive.google.com/file/d/1TWBB9rqGRnWOJsDcwkUE2HkJNEafawQm/view?usp=sharing

Payments table:
https://drive.google.com/file/d/1vakf_7F9cb7ZKOeCofyI370Paiswg3JX/view?usp=sharing

A.Which payment_type has the fastest average delivery day where delivery day can be found by using order_delivered_customer_date and order_purchase_timestamp columns? You should be careful about choosing these dates that shouldn't be null.
(Hint: The result should show only one payment_type with the average_delivery_day info)

```
SELECT
 payment_type,
 AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)) AS
AVG_DELIVERY_DAY
FROM bootcamp2022-372013.final.ORDERS
FULL JOIN bootcamp2022-372013.final.PAYMENTS AS PAY
USING(order_id)
WHERE order_delivered_customer_date IS NOT NULL AND order_purchase_timestamp IS NOT NULL
GROUP BY payment_type
```

B.What is the maximum payment_value of each state where order status is neither unavailable nor canceled? Additionally, the max_payment_revenue column should have 1 decimal point.
(Hint: The result table will only have state and max_payment_revenue columns)

```
SELECT
  customer_state,
  ROUND(MAX(payment_value),1) AS max_payment_revenue
FROM bootcamp2022-372013.final.CUSTOMERS AS cust
FULL JOIN bootcamp2022-372013.final.ORDERS AS orders
```

```
USING(customer_id)
FULL JOIN bootcamp2022-372013.final.PAYMENTS AS payments
USING(order_id)
WHERE order_status != 'unavailable' AND order_status != 'canceled'
GROUP BY customer_state;
```

C.In the payments table, how many distinct order_ids that exist more than once?

```
WITH NUM_CITY AS(
SELECT DISTINCT
  order_id,
  COUNT(order_id) OVER(PARTITION BY order_id ORDER BY order_id) AS NUM_OF_ORDER
FROM bootcamp2022-372013.final.PAYMENTS
)
SELECT
  order_id,
  NUM_OF_ORDER
FROM NUM_CITY
WHERE NUM_OF_ORDER>1
```

D.For each row, find the payment_value of previous row and leading row in the partition of order_id and order of payment_sequential ascendingly.(Hint: The result table should have all columns in the payments table and two new columns:previous_payment_value and leading_payment_value.Also, we don’t want null values for these two new fields)

```
WITH VALUES AS (
SELECT
 *,
 LAG(payment_value) OVER(PARTITION BY order_id ORDER BY payment_sequential ASC) AS
PREVIOUS_PAYMENT_VALUE,
 LEAD(payment_value) OVER(PARTITION BY order_id ORDER BY payment_sequential ASC) AS
LEADING_PAYMENT_VALUE
FROM bootcamp2022-372013.final.PAYMENTS
)
SELECT
```

```
 *
FROM VALUES
WHERE PREVIOUS_PAYMENT_VALUE IS NOT NULL AND LEADING_PAYMENT_VALUE IS NOT NULL
```

E.Get all the columns in the customers table and add three new columns:

* order_id

* order_purchase_date (Hint: convert the timestamp to the date value)

* payment_value

* payment_value_int (Hint: payment_value but in the integer format. You should use a function to convert it type)

```
SELECT
 C.*,
 O.order_id,
 DATE(O.order_purchase_timestamp) AS order_purchase_date,
 P.payment_value,
 CAST(payment_value AS INTEGER) AS payment_value_int
FROM bootcamp2022-372013.final.CUSTOMERS AS C
LEFT JOIN bootcamp2022-372013.final.ORDERS O
USING (customer_id)
LEFT JOIN bootcamp2022-372013.final.PAYMENTS  AS P
USING (order_id)
```

F.Get all columns from the payments table excluding the payment_type column, and create payment_method column instead of that. This new column should have this format:
If payment_type is “not_defined”, then payment_method will be 0
If payment_type is “credit_card”, then payment_method will be 1
If payment_type is “voucher”, then payment_method will be 2
If payment_type is “debit_card”, then payment_method will be 3
If payment_type is “boleto”, then payment_method will be 4

```
 SELECT
 * EXCEPT(payment_type),
```

```
 CASE
 WHEN payment_type = 'not_defined' THEN 0
 WHEN payment_type = 'credit_card' THEN 1
 WHEN payment_type = 'voucher' THEN 2
 WHEN payment_type = 'debit_card' THEN 3
 WHEN payment_type = 'boleto' THEN 4
 ELSE NULL
 END AS payment_method
FROM bootcamp2022-372013.final.PAYMENTS
```

Question 2:

You have one table:

Netflix_movies: https://drive.google.com/file/d/1VRdGqa3KmgDj_BEX4_OBEtDa-JRmE1LH/view? usp=sharing

A.How many times each word is used in the table, that you get when you split the movie_name by each space?
(Hint: The result table only have 2 columns: words_used_in_movie_names, number_of_occurence. Ex. You can imagine that asks for how many times you see Rome word in this table)

```
WITH UNNEST_WORDS AS (
SELECT
  movie_name,
  WORDS
FROM bootcamp2022-372013.final.netflix,UNNEST(REGEXP_EXTRACT_ALL(movie_name,r"(\w+)"))
WORDS
)
SELECT
  WORDS,
  COUNT(*) AS count
FROM UNNEST_WORDS
GROUP BY WORDS
ORDER BY count DESC
```

B.If a movie_name has “**Vol. 1**”, then change it to “Part 1”; if has “**Vol. 2**”, then change it to “Part. 2”. Else movie_name. After these changes, name show this column as “movie_name” instead of showing the original movie_name column.

```
SELECT
  REGEXP_REPLACE(REGEXP_REPLACE(movie_name, 'Vol. 1', 'Part 1'), 'Vol. 2',   'Part 2') AS
movie_name
FROM bootcamp2022-372013.final.netflix
```

C.In this table, we have some movie_names that occur multiple times. Create a new column in the result table, call this column as “**previous_year**” and show the previous year of each movie if they exist multiple times in that table. In the result table, there shouldn’t be any null previous_year values.

```
WITH year_table AS(
SELECT
  movie_name,
  lag(year) over(partition by movie_name order by year) as previous_year
FROM bootcamp2022-372013.final.netflix
)
SELECT
  movie_name,
  previous_year
FROM year_table
WHERE previous_year IS NOT NULL
ORDER BY movie_name
```