#Question 1:

#A

```
SELECT
  payments.payment_type,
  AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)) AS ave
rage_delivery_day
FROM `sql-final.final_project.payments`
INNER JOIN `final_project.orders` AS orders ON orders.order_id = payments.order_id
WHERE  order_delivered_customer_date IS NOT NULL AND order_purchase_timestamp IS NOT
NULL
GROUP BY payment_type
ORDER BY average_delivery_day
LIMIT 1;
```

#B

```
SELECT
  customers.customer_state,
  ROUND(MAX(payment_value),1) AS max_payment_revenue
FROM `sql-final.final_project.payments`AS customers
INNER JOIN `sql-final.final_project.orders`
AS orders ON orders.customer_id = customers.customer_id
INNER JOIN `sql-final.final_project.payments`
AS payments ON payments.order_id = orders.order_id
WHERE order_status NOT IN("unavailable", "canceled")
GROUP BY customer_state;
```

#C

1st version of answer with Window functions:

```
WITH unique_orders AS(
  SELECT
    order_id,
    ROW_NUMBER() OVER(PARTITION BY order_id) AS numbered_orders
  FROM `sql-final.final_project.payments`
),
counted_orders AS(
  SELECT
    COUNT(order_id) AS counted_orders
  FROM unique_orders
  WHERE numbered_orders <> 1
  GROUP BY order_id
)
SELECT
  COUNT(counted_orders)
FROM counted_orders
```

OR

2nd version of answer:

```
WITH distinct_orders AS(
  SELECT DISTINCT
    COUNT(order_id) AS counted_order,
    order_id
  FROM `sql-final.final_project.payments`
  GROUP BY order_id
  HAVING COUNT(order_id)>1
)
SELECT
  COUNT(counted_order)
FROM distinct_orders
```

#D

1)

```
SELECT
  *,
LAG(payment_value,1,0) OVER(PARTITION BY order_id ORDER BY payment_sequential)
AS previous_payment_value,
LEAD(payment_value,1,0) OVER(PARTITION BY order_id ORDER BY payment_sequential)
AS leading_payment_value
FROM `sql-final.final_project.payments`
ORDER BY order_id, payment_sequential
```

OR

2)

```
WITH previous_payment_value AS(
  SELECT
    order_id,
    payment_sequential,
    payment_value,
    LAG(payment_value) OVER(PARTITION BY order_id ORDER BY payment_sequential) AS pre
vious_payment_value
  FROM `sql-final.final_project.payments`
),
leading_payment_value AS(
  SELECT
    order_id,
    payment_sequential,
    payment_value,
    LEAD(payment_value) OVER(PARTITION BY order_id ORDER BY payment_sequential) AS le
ading_payment_value
  FROM `sql-final.final_project.payments`
)
SELECT
  payments.*,
  previous_payment_value,
  leading_payment_value
```

```
FROM `sql-final.final_project.payments`
AS payments,  previous_payment_value, leading_payment_value
WHERE  previous_payment_value IS NOT NULL AND leading_payment_value IS NOT NULL
ORDER BY order_id, payment_sequential
```

#E

1)

```
SELECT
  customers.*,
  orders.order_id,
  DATE(order_purchase_timestamp) AS order_purchase_date,
  payments.payment_value,
  CAST(payment_value AS INT64) AS payment_value_int
FROM `sql-final.final_project.customers` AS customers
LEFT JOIN `sql-final.final_project.orders`
AS orders ON orders.customer_id = customers.customer_id
LEFT JOIN `sql-final.final_project.payments`
AS payments ON payments.order_id = orders.order_id
```

OR

2)

```
SELECT
  customers.*,
  orders.order_id,
  CAST(order_purchase_timestamp AS DATE) AS order_purchase_date,
  payments.payment_value,
  CAST(payment_value AS INT64) AS payment_value_int
FROM `sql-final.final_project.customers` AS customers
LEFT JOIN `sql-final.final_project.orders`
AS orders ON orders.customer_id = customers.customer_id
LEFT JOIN `sql-final.final_project.payments`
AS payments ON payments.order_id = orders.order_id
```

#F

```
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
  END AS payment_method
FROM `sql-final.final_project.payments`
```

#Question 2:

#A

```
WITH extract_all_words AS(
  SELECT
   words_used_in_movie_names
  FROM `sql-final.final_project.netflix_movies`,
  UNNEST(REGEXP_EXTRACT_ALL(movie_name, r"(\w+)")) AS words_used_in_movie_names
)
SELECT
  extract_all_words.words_used_in_movie_names,
  COUNT(words_used_in_movie_names) AS number_of_occurence
FROM extract_all_words
GROUP BY(words_used_in_movie_names)
```

#B

```
SELECT
  * EXCEPT(movie_name),
  REGEXP_REPLACE(REPLACE(movie_name, r'Vol. 1', 'Part 1'), r'Vol. 2', 'Part 2')
  AS movie_name
FROM `sql-final.final_project.netflix_movies`
```

#C

```
WITH previous_year_info AS (
   SELECT
       *,
       LAG(year) OVER (PARTITION BY movie_name ORDER BY year) AS previous_year
   FROM `sql-final.final_project.netflix_movies`
)
SELECT
    *
FROM previous_year_info
WHERE previous_year IS NOT NULL;
```