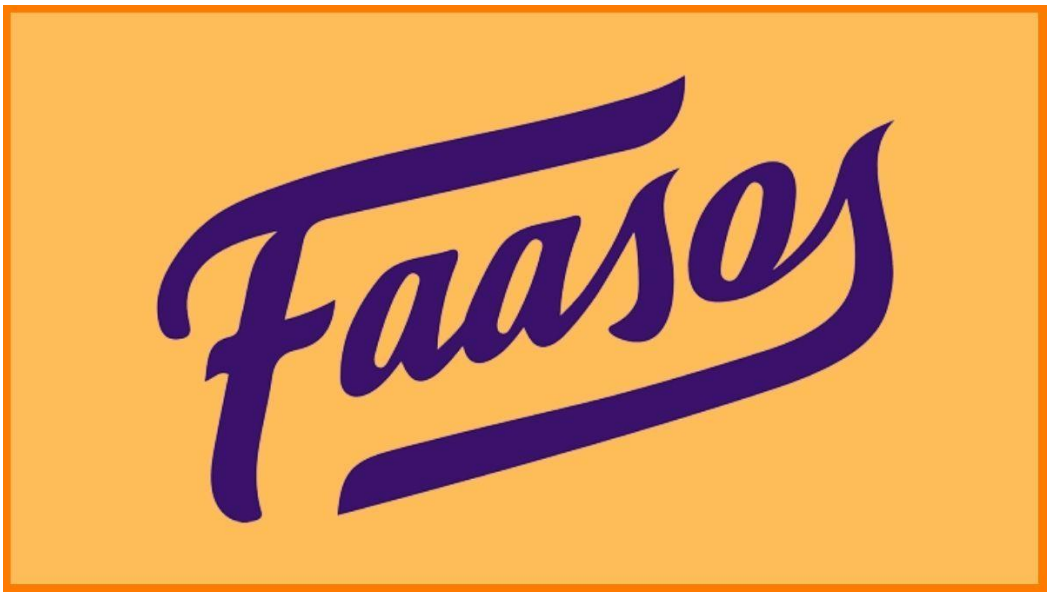
Faasos Data Analysis using MSSQL



Faasos is an Indian "food on demand" service that was incorporated in 2004. It is one of the brands owned by the online restaurant company, Rebel Foods.

The project is based on to perform the data analysis over the given data and generate insights to better understanding of the function of the company.

Tables present in the database

- **Rolls**

	roll_id	roll_name
1	1	Non Veg Roll
2	2	Veg Roll

- **Driver Order**

	order_id	driver_id	pickup_time	distance	duration	cancellation
1	1	1	2021-01-01 18:15:34.000	20km	32 minutes	
2	2	1	2021-01-01 19:10:54.000	20km	27 minutes	
3	3	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN
4	4	2	2021-01-04 13:53:03.000	23.4	40	NaN
5	5	3	2021-01-08 21:10:57.000	10	15	NaN
6	6	3	NULL	NULL	NULL	Cancellation
7	7	2	2021-01-08 21:30:45.000	25km	25mins	NULL
8	8	2	2021-01-10 00:15:02.000	23.4 km	15 minute	NULL
9	9	2	NULL	NULL	NULL	Customer Cancellation
10	10	1	2021-01-11 18:50:20.000	10km	10minutes	NULL

- **Customer Orders**

	order_id	customer_id	roll_id	not_include_items	extra_items_included
1	1	101	1		
2	2	101	1		
3	3	102	1		
4	3	102	2		NaN
5	4	103	1	4	
6	4	103	1	4	
7	4	103	2	4	
8	5	104	1	NULL	1
9	6	101	2	NULL	NULL
10	7	105	2	NULL	1
11	8	102	1	NULL	NULL
12	9	103	1	4	1,5
13	10	104	1	NULL	NULL
14	10	104	1	2,6	1,4

Questions:

Q1- How many rolls were ordered?

Ans.

```
select count(roll_id) as Total_orders
from customer_orders;
```

	Total_orders
1	14

Q2- How many unique customers are there?

Ans.

```
select count(distinct(customer_id)) as Customers
from customer_orders;
```

	Customers
1	5

Q3- How many successful orders were delivered by each of the driver?

	order_id	driver_id	pickup_time	distance	duration	cancellation
1	1	1	2021-01-01 18:15:34.000	20km	32 minutes	
2	2	1	2021-01-01 19:10:54.000	20km	27 minutes	
3	3	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN
4	4	2	2021-01-04 13:53:03.000	23.4	40	NaN
5	5	3	2021-01-08 21:10:57.000	10	15	NaN
6	6	3	NULL	NULL	NULL	Cancellation
7	7	2	2021-01-08 21:30:45.000	25km	25mins	NULL
8	8	2	2021-01-10 00:15:02.000	23.4 km	15 minute	NULL
9	9	2	NULL	NULL	NULL	Customer Cancellation
10	10	1	2021-01-11 18:50:20.000	10km	10minutes	NULL

Driver order table shows the cancellation column values where the order picked up by the driver is either cancelled or successfully delivered.

If the value shows “Cancellation” or “Customer Cancellation” this means the order is cancelled, otherwise the remaining values indicates the order have been delivered although some values may be NaN, NULL or empty.

```
select driver_id, count(order_id) as Orders_Delivered
from(
select *, case when cancellation in ('Cancellation', 'Customer Cancellation') then
'cancel' else 'not_cancel' end as cancel_details
from driver_order) as t
where cancel_details = 'not_cancel'
group by driver_id;
```

	driver_id	Orders_Delivered
1	1	4
2	2	3
3	3	1

Q4- How many of each type of roll was delivered?

Ans.

```
select roll_id, count(roll_id) as orders
from customer_orders
where order_id in (
select order_id
from(
select *, case when cancellation in ('Cancellation', 'Customer Cancellation') then
                'cancel' else 'not_cancel' end as cancel_details
from driver_order) as temp
where cancel_details = 'not_cancel')
group by roll_id;
```

	order_id	driver_id	pickup_time	distance	duration	cancellation	cancel_details
1	1	1	2021-01-01 18:15:34.000	20km	32 minutes		not_cancel
2	2	1	2021-01-01 19:10:54.000	20km	27 minutes		not_cancel
3	3	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN	not_cancel
4	4	2	2021-01-04 13:53:03.000	23.4	40	NaN	not_cancel
5	5	3	2021-01-08 21:10:57.000	10	15	NaN	not_cancel
6	6	3	NULL	NULL	NULL	Cancellation	cancel
7	7	2	2021-01-08 21:30:45.000	25km	25mins	NULL	not_cancel
8	8	2	2021-01-10 00:15:02.000	23.4 km	15 minute	NULL	not_cancel
9	9	2	NULL	NULL	NULL	Customer Cancellation	cancel
10	10	1	2021-01-11 18:50:20.000	10km	10minutes	NULL	not_cancel

- The inner most query results are as follow, where the cancellation values have been replaced by either ‘not_cancel’ or ‘cancel’ in order to clean the data.

- The next inner query gives the order_id where cancel_details are not cancelled.

	order_id
1	1
2	2
3	3
4	4
5	5
6	7
7	8
8	10

- The last overall query gives the results for each type of roll that was ordered.

	roll_id	orders
1	1	9
2	2	3

Q5- How many Veg and Non-Veg rolls were ordered by each customer?

Ans.

```
select a.customer_id, a.count, b.roll_name from
(select customer_id, roll_id, count(roll_id) as count
from customer_orders
group by customer_id, roll_id)a inner join rolls b on a.roll_id = b.roll_id
order by a.customer_id;
```

The result shows the count of veg and non-veg rolls ordered by each customer_id.

	customer_id	count	roll_name
1	101	2	Non Veg Roll
2	101	1	Veg Roll
3	102	2	Non Veg Roll
4	102	1	Veg Roll
5	103	3	Non Veg Roll
6	103	1	Veg Roll
7	104	3	Non Veg Roll
8	105	1	Veg Roll

Q6- What was the maximum number of rolls delivered in a single order?

Ans.

```
select order_id, counts
```

```
from(
select *, DENSE_RANK() over (order by counts desc) rnk
from(
select order_id, count(roll_id) as counts
from customer_orders
where order_id in (
select order_id
from(select *, case when cancellation in ('Cancellation', 'Customer Cancellation')
then 'cancel' else 'not_cancel' end as cancel_details from driver_order) tmp)
group by order_id) temp)t
where rnk = 1;
```

Five sub-queries have been used to reach at the final result.

Interpreting each query result:

1. `select *, case when cancellation in ('Cancellation', 'Customer Cancellation') then 'cancel' else 'not_cancel' end as cancel_details from driver_order`

	order_id	driver_id	pickup_time	distance	duration	cancellation	cancel_details
1	1	1	2021-01-01 18:15:34.000	20km	32 minutes		not_cancel
2	2	1	2021-01-01 19:10:54.000	20km	27 minutes		not_cancel
3	3	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN	not_cancel
4	4	2	2021-01-04 13:53:03.000	23.4	40	NaN	not_cancel
5	5	3	2021-01-08 21:10:57.000	10	15	NaN	not_cancel
6	6	3	NULL	NULL	NULL	Cancellation	cancel
7	7	2	2021-01-08 21:30:45.000	25km	25mins	NULL	not_cancel
8	8	2	2021-01-10 00:15:02.000	23.4 km	15 minute	NULL	not_cancel
9	9	2	NULL	NULL	NULL	Customer Cancellation	cancel
10	10	1	2021-01-11 18:50:20.000	10km	10minutes	NULL	not_cancel

The query outputs the results where the cancellation details are presented as either ‘not_cancel’ or ‘cancel’.

2. `select order_id from(select *, case when cancellation in ('Cancellation', 'Customer Cancellation') then 'cancel' else 'not_cancel' end as cancel_details from driver_order) tmp`

The query results in showing the order_id from the above query result.

	order_id
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

3.
```sql
select order_id, count(roll_id) as counts
from customer_orders
where order_id in (
select order_id
from(select *, case when cancellation in
    ('Cancellation', 'Customer Cancellation') then
    'cancel' else 'not_cancel' end as cancel_details
    from driver_order) tmp)
group by order_id;
```

	order_id	counts
1	1	1
2	2	1
3	3	2
4	4	3
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	2

The query selects the order_id, count of order_id from customer_orders and maching the result with the order_id retrieved from the query 2.

4.
```sql
select *, DENSE_RANK() over (order by counts desc)
rnk
from(
select order_id, count(roll_id) as counts
from customer_orders
where order_id in (
select order_id
from(select *, case when cancellation in
('Cancellation', 'Customer Cancellation') then
'cancel' else 'not_cancel' end as cancel_details
from driver_order) tmp)
group by order_id) temp
```

	order_id	counts	rnk
1	4	3	1
2	3	2	2
3	10	2	2
4	1	1	3
5	2	1	3
6	5	1	3
7	6	1	3
8	7	1	3
9	8	1	3
10	9	1	3

The query gives rank to each order_id based on the counts in descending order.

5.
```sql
select order_id, counts
from(
select *, DENSE_RANK() over (order by counts desc) rnk
from(
select order_id, count(roll_id) as counts
from customer_orders
where order_id in (
select order_id
from(select *, case when cancellation in ('Cancellation', 'Customer
Cancellation') then 'cancel' else 'not_cancel' end as cancel_details from
driver_order) tmp)
group by order_id) temp)t
where rnk = 1;
```

	order_id	counts
1	4	3

The full query result shows the order_id and the counts of the rolls that were placed in a single order.

Q7- For each customer, how many delivered rolls had at least 1 change and how many had no change?

Ans.

```sql
with temp_customer_orders(order_id, customer_id, roll_id, not_include_items,
extra_items_included, order_date) as
(select order_id, customer_id, roll_id,
            case when not_include_items is null or not_include_items = ' ' then '0'
            else not_include_items end as not_include_items,
```

```
            case when extra_items_included is null or extra_items_included = 'NaN' or
            extra_items_included = ' ' then '0'
            else extra_items_included end as extra_items_included, order_date
from customer_orders
)
,

temp_driver_order(order_id, driver_id, pickup_time, distance, duration,
new_cancellation) as
(select order_id, driver_id, pickup_time, distance, duration,
      case when cancellation in ('Cancellation', 'Customer Cancellation') then 0
      else 1 end as new_cancellation
from driver_order
)
select customer_id, Chg_No_Chg, count(order_id) as Count_Changes
from
(select *, case when not_include_items = '0' and extra_items_included = '0' then 'No
change' else 'Change' end as Chg_No_Chg
from temp_customer_orders where order_id in(
select order_id from temp_driver_order where new_cancellation=1))a
group by customer_id, Chg_No_Chg
order by customer_id;
```

CTE Expressions are used to create two tables that are used for data cleaning before extracting the result.

1.

	order_id	customer_id	roll_id	not_include_items	extra_items_included	order_date
1	1	101	1	0	0	2021-01-01 18:05:02.000
2	2	101	1	0	0	2021-01-01 19:00:52.000
3	3	102	1	0	0	2021-01-02 23:51:23.000
4	3	102	2	0	0	2021-01-02 23:51:23.000
5	4	103	1	4	0	2021-01-04 13:23:46.000
6	4	103	1	4	0	2021-01-04 13:23:46.000
7	4	103	2	4	0	2021-01-04 13:23:46.000
8	5	104	1	0	1	2021-01-08 21:00:29.000
9	6	101	2	0	0	2021-01-08 21:03:13.000
10	7	105	2	0	1	2021-01-08 21:20:29.000
11	8	102	1	0	0	2021-01-09 23:54:33.000
12	9	103	1	4	1.5	2021-01-10 11:22:59.000
13	10	104	1	0	0	2021-01-11 18:34:49.000
14	10	104	1	2.6	1.4	2021-01-11 18:34:49.000

```
with temp_customer_orders(order_id, customer_id, roll_id, not_include_items,
extra_items_included, order_date) as
(select order_id, customer_id, roll_id,
        case when not_include_items is null or not_include_items = ' ' then '0'
        else not_include_items end as not_include_items,
        case when extra_items_included is null or extra_items_included = 'NaN' or
        extra_items_included = ' ' then '0'
        else extra_items_included end as extra_items_included, order_date
from customer_orders
)
select * from temp_customer_orders
```

The first table helps to clean the customer_orders data, where not_include_items is null or ‘ ‘ by replacing it with 0 and extra_items_included is null or NaN by replacing it with 0.

2.

	order_id	driver_id	pickup_time	distance	duration	new_cancellation
1	1	1	2021-01-01 18:15:34.000	20km	32 minutes	1
2	2	1	2021-01-01 19:10:54.000	20km	27 minutes	1
3	3	1	2021-01-03 00:12:37.000	13.4km	20 mins	1
4	4	2	2021-01-04 13:53:03.000	23.4	40	1
5	5	3	2021-01-08 21:10:57.000	10	15	1
6	6	3	NULL	NULL	NULL	0
7	7	2	2021-01-08 21:30:45.000	25km	25mins	1
8	8	2	2021-01-10 00:15:02.000	23.4 km	15 minute	1
9	9	2	NULL	NULL	NULL	0
10	10	1	2021-01-11 18:50:20.000	10km	10minutes	1

```
with temp_driver_order(order_id, driver_id, pickup_time, distance, duration,
new_cancellation) as
(select order_id, driver_id, pickup_time, distance, duration,
    case when cancellation in ('Cancellation', 'Customer Cancellation') then 0
    else 1 end as new_cancellation
from driver_order
)
select * from temp_driver_order
```

The second table helps to clean the driver_order data, where if cancellation details are cancelled then replace it with 0 else with 1.

Then combining both the tables to get desired results.

	customer_id	Chg_No_Chg	Count_orders
1	101	No change	2
2	102	No change	3
3	103	Change	3
4	104	Change	2
5	104	No change	1
6	105	Change	1

The result shows that the customer_id where it has not made any changes, count the orders that have been placed and where the changes have been initiated count the orders.

Q8- How many rolls were delivered that had both exclusions and extras?

Ans.

```
with temp_customer_orders(order_id, customer_id, roll_id, not_include_items,
extra_items_included, order_date) as
```

```
(select order_id, customer_id, roll_id,
            case when not_include_items is null or not_include_items = ' ' then '0'
            else not_include_items end as not_include_items,
            case when extra_items_included is null or extra_items_included = 'NaN' or
            extra_items_included = ' ' then '0'
            else extra_items_included end as extra_items_included, order_date
from customer_orders
),

temp_driver_order(order_id, driver_id, pickup_time, distance, duration,
new_cancellation) as
(select order_id, driver_id, pickup_time, distance, duration,
      case when cancellation in ('Cancellation', 'Customer Cancellation') then 0 else
      1 end as new_cancellation
from driver_order
)
select Inc_Exc, count(Inc_Exc) as counts
from(
select *, case when not_include_items != '0' and extra_items_included != '0' then
'both inc exc' else 'eitheir 1 inc or exc' end as Inc_Exc
from temp_customer_orders where order_id in(
select order_id from temp_driver_order where new_cancellation=1)) a
group by Inc_Exc;
```

CTE Expressions are used to create two tables that are used for data cleaning before extracting the result.

1.

	order_id	customer_id	roll_id	not_include_items	extra_items_included	order_date
1	1	101	1	0	0	2021-01-01 18:05:02.000
2	2	101	1	0	0	2021-01-01 19:00:52.000
3	3	102	1	0	0	2021-01-02 23:51:23.000
4	3	102	2	0	0	2021-01-02 23:51:23.000
5	4	103	1	4	0	2021-01-04 13:23:46.000
6	4	103	1	4	0	2021-01-04 13:23:46.000
7	4	103	2	4	0	2021-01-04 13:23:46.000
8	5	104	1	0	1	2021-01-08 21:00:29.000
9	6	101	2	0	0	2021-01-08 21:03:13.000
10	7	105	2	0	1	2021-01-08 21:20:29.000
11	8	102	1	0	0	2021-01-09 23:54:33.000
12	9	103	1	4	1.5	2021-01-10 11:22:59.000
13	10	104	1	0	0	2021-01-11 18:34:49.000
14	10	104	1	2.6	1.4	2021-01-11 18:34:49.000

```
with temp_customer_orders(order_id, customer_id, roll_id, not_include_items,
extra_items_included, order_date) as
(select order_id, customer_id, roll_id,
        case when not_include_items is null or not_include_items = ' ' then '0'
        else not_include_items end as not_include_items,
        case when extra_items_included is null or extra_items_included = 'NaN' or
        extra_items_included = ' ' then '0'
        else extra_items_included end as extra_items_included, order_date
from customer_orders
)
select * from temp_customer_orders
```

The first table helps to clean the customer_orders data, where not_include_items is null or ‘ ‘ by replacing it with 0 and extra_items_included is null or NaN by replacing it with 0.

2.

	order_id	driver_id	pickup_time	distance	duration	new_cancellation
1	1	1	2021-01-01 18:15:34.000	20km	32 minutes	1
2	2	1	2021-01-01 19:10:54.000	20km	27 minutes	1
3	3	1	2021-01-03 00:12:37.000	13.4km	20 mins	1
4	4	2	2021-01-04 13:53:03.000	23.4	40	1
5	5	3	2021-01-08 21:10:57.000	10	15	1
6	6	3	NULL	NULL	NULL	0
7	7	2	2021-01-08 21:30:45.000	25km	25mins	1
8	8	2	2021-01-10 00:15:02.000	23.4 km	15 minute	1
9	9	2	NULL	NULL	NULL	0
10	10	1	2021-01-11 18:50:20.000	10km	10minutes	1

```
with temp_driver_order(order_id, driver_id, pickup_time, distance, duration,
new_cancellation) as
(select order_id, driver_id, pickup_time, distance, duration,
    case when cancellation in ('Cancellation', 'Customer Cancellation') then 0
    else 1 end as new_cancellation
from driver_order
)
select * from temp_driver_order
```

The second table helps to clean the driver_order data, where if cancellation details are cancelled then replace it with 0 else with 1.

Then combining both the tables to get desired results.

	Inc_Exc	counts
1	both inc exc	1
2	eitheir 1 inc or exc	11

The results shows that the customers which have made the change in their order, where they have included extra items or excluded extra items.

Q9- What was the total number of rolls ordered for each hour of the day?

Ans.

```
select hours_bin, count(hours_bin) as Rolls_Ordered
from(
select *, concat(cast(DATEPART(hour,order_date)AS varchar), '-',
cast(DATEPART(hour,order_date)+1 as varchar)) as hours_bin from  customer_orders)a
group by hours_bin;
```

	hours_bin	Rolls_Ordered
1	11-12	1
2	13-14	3
3	18-19	3
4	19-20	1
5	21-22	3
6	23-24	3

Data Cleaning is the most important part in this question as the order_date column is datetime stamp out of which the hour value has to be extracted, then hours are converted in to bins to find how many rolls were ordered at the peak hours.

Q10- What was the number of orders for each day of the week?

Ans.

```
select dow, COUNT(distinct order_id) as num_orders
from(
select *, DATENAME(DW, order_date) as dow from customer_orders)a
group by dow
order by num_orders;
```

	dow	num_orders
1	Sunday	1
2	Monday	2
3	Saturday	2
4	Friday	5

The result shows the number of orders placed by the customers the most in which day of the week.

Q11- What was the average time in minutes it took for each driver to arrive at FASOOS HQ to pick up the order?

Ans.

```
select driver_id, AVG(diff) as Avg_time_taken from(
select * from(
select *, ROW_NUMBER() over (partition by order_id order by diff) as rnk
from(
select a.order_id, a.customer_id, a.roll_id, a.not_include_items,
a.extra_items_included, a.order_date, b.driver_id, b.pickup_time, b.distance,
b.duration, b.cancellation, DATEDIFF(minute, a.order_date, b.pickup_time) as diff
from customer_orders a
inner join driver_order b
on a.order_id = b.order_id
where b.pickup_time is not null)a)b
where rnk = 1)c
group by driver_id;
```

	driver_id	Avg_time_taken
1	1	14
2	2	20
3	3	10

The result consists of four sub-queries which needs to be understood step by step.

1.
```
select a.order_id, a.customer_id, a.roll_id, a.not_include_items,
a.extra_items_included, a.order_date, b.driver_id, b.pickup_time, b.distance,
b.duration, b.cancellation, DATEDIFF(minute, a.order_date, b.pickup_time) as
diff
from customer_orders a
inner join driver_order b
on a.order_id = b.order_id
where b.pickup_time is not null
```

	order_id	customer_id	roll_id	not_include_items	extra_items_included	order_date	driver_id	pickup_time	distance	duration	cancellation	diff
1	1	101	1			2021-01-01 18:05:02.000	1	2021-01-01 18:15:34.000	20km	32 minutes		10
2	2	101	1			2021-01-01 19:00:52.000	1	2021-01-01 19:10:54.000	20km	27 minutes		10
3	3	102	1			2021-01-02 23:51:23.000	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN	21
4	3	102	2		NaN	2021-01-02 23:51:23.000	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN	21
5	4	103	1	4		2021-01-04 13:23:46.000	2	2021-01-04 13:53:03.000	23.4	40	NaN	30
6	4	103	1	4		2021-01-04 13:23:46.000	2	2021-01-04 13:53:03.000	23.4	40	NaN	30
7	4	103	2	4		2021-01-04 13:23:46.000	2	2021-01-04 13:53:03.000	23.4	40	NaN	30
8	5	104	1	NULL	1	2021-01-08 21:00:29.000	3	2021-01-08 21:10:57.000	10	15	NaN	10
9	7	105	2	NULL	1	2021-01-08 21:20:29.000	2	2021-01-08 21:30:45.000	25km	25mins	NULL	10
10	8	102	1	NULL	NULL	2021-01-09 23:54:33.000	2	2021-01-10 00:15:02.000	23.4 km	15 minute	NULL	21
11	10	104	1	NULL	NULL	2021-01-11 18:34:49.000	1	2021-01-11 18:50:20.000	10km	10minutes	NULL	16
12	10	104	1	2,6	1,4	2021-01-11 18:34:49.000	1	2021-01-11 18:50:20.000	10km	10minutes	NULL	16

First query results in creating a join between customer_orders and driver_order. Also extracting minutes difference between order_date column and pickup_time column, which gives the difference between the time taken between order placed and order picked up by the driver.

2.
```
select *, ROW_NUMBER() over (partition by order_id order by diff) as rnk
from(
select a.order_id, a.customer_id, a.roll_id, a.not_include_items,
a.extra_items_included, a.order_date, b.driver_id, b.pickup_time, b.distance,
b.duration, b.cancellation, DATEDIFF(minute, a.order_date, b.pickup_time) as
diff
from customer_orders a
inner join driver_order b
on a.order_id = b.order_id
where b.pickup_time is not null)a
```

	order_id	customer_id	roll_id	not_include_items	extra_items_included	order_date	driver_id	pickup_time	distance	duration	cancellation	diff	rnk
1	1	101	1			2021-01-01 18:05:02.000	1	2021-01-01 18:15:34.000	20km	32 minutes		10	1
2	2	101	1			2021-01-01 19:00:52.000	1	2021-01-01 19:10:54.000	20km	27 minutes		10	1
3	3	102	1			2021-01-02 23:51:23.000	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN	21	1
4	3	102	2		NaN	2021-01-02 23:51:23.000	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN	21	2
5	4	103	1	4		2021-01-04 13:23:46.000	2	2021-01-04 13:53:03.000	23.4	40	NaN	30	1
6	4	103	1	4		2021-01-04 13:23:46.000	2	2021-01-04 13:53:03.000	23.4	40	NaN	30	2
7	4	103	2	4		2021-01-04 13:23:46.000	2	2021-01-04 13:53:03.000	23.4	40	NaN	30	3
8	5	104	1	NULL	1	2021-01-08 21:00:29.000	3	2021-01-08 21:10:57.000	10	15	NaN	10	1
9	7	105	2	NULL	1	2021-01-08 21:20:29.000	2	2021-01-08 21:30:45.000	25km	25mins	NULL	10	1
10	8	102	1	NULL	NULL	2021-01-09 23:54:33.000	2	2021-01-10 00:15:02.000	23.4 km	15 minute	NULL	21	1
11	10	104	1	NULL	NULL	2021-01-11 18:34:49.000	1	2021-01-11 18:50:20.000	10km	10minutes	NULL	16	1
12	10	104	1	2,6	1,4	2021-01-11 18:34:49.000	1	2021-01-11 18:50:20.000	10km	10minutes	NULL	16	2

Next query provides the rank to the difference which was calculated in first query. Since the driver take same time in picking up order.

3.
```
select * from(
select *, ROW_NUMBER() over (partition by order_id order by diff) as rnk
from(
```

```
select a.order_id, a.customer_id, a.roll_id, a.not_include_items,
a.extra_items_included, a.order_date, b.driver_id, b.pickup_time, b.distance,
b.duration, b.cancellation, DATEDIFF(minute, a.order_date, b.pickup_time) as
diff
from customer_orders a
inner join driver_order b
on a.order_id = b.order_id
where b.pickup_time is not null)a)b
where rnk = 1
```

	order_id	customer_id	roll_id	not_include_items	extra_items_included	order_date	driver_id	pickup_time	distance	duration	cancellation	diff	rnk
1	1	101	1			2021-01-01 18:05:02.000	1	2021-01-01 18:15:34.000	20km	32 minutes		10	1
2	2	101	1			2021-01-01 19:00:52.000	1	2021-01-01 19:10:54.000	20km	27 minutes		10	1
3	3	102	1			2021-01-02 23:51:23.000	1	2021-01-03 00:12:37.000	13.4km	20 mins	NaN	21	1
4	4	103	1	4		2021-01-04 13:23:46.000	2	2021-01-04 13:53:03.000	23.4	40	NaN	30	1
5	5	104	1	NULL	1	2021-01-08 21:00:29.000	3	2021-01-08 21:10:57.000	10	15	NaN	10	1
6	7	105	2	NULL	1	2021-01-08 21:20:29.000	2	2021-01-08 21:30:45.000	25km	25mins	NULL	10	1
7	8	102	1	NULL	NULL	2021-01-09 23:54:33.000	2	2021-01-10 00:15:02.000	23.4 km	15 minute	NULL	21	1
8	10	104	1	NULL	NULL	2021-01-11 18:34:49.000	1	2021-01-11 18:50:20.000	10km	10minutes	NULL	16	1

Only selecting the rows where rank is 1.

4.
```
select driver_id, AVG(diff) as Avg_time_taken from(
select * from(
select *, ROW_NUMBER() over (partition by order_id order by diff) as rnk
from(
select a.order_id, a.customer_id, a.roll_id, a.not_include_items,
a.extra_items_included, a.order_date, b.driver_id, b.pickup_time, b.distance,
b.duration, b.cancellation, DATEDIFF(minute, a.order_date, b.pickup_time) as
diff
from customer_orders a
inner join driver_order b
on a.order_id = b.order_id
where b.pickup_time is not null)a)b
where rnk = 1)c
group by driver_id;
```

	driver_id	Avg_time_taken
1	1	14
2	2	20
3	3	10

Q12- Is there any relationship between the number of rolls and how long the order takes to prepare?

Ans.

```
select order_id, count(roll_id) as count_, SUM(diff)/COUNT(roll_id) as time_taken from
(select a.order_id, a.customer_id, a.roll_id, a.not_include_items,
a.extra_items_included, a.order_date, b.driver_id, b.pickup_time, b.distance,
b.duration, b.cancellation, DATEDIFF(minute, a.order_date, b.pickup_time) as diff

from customer_orders a
inner join driver_order b
on a.order_id = b.order_id
where b.pickup_time is not null)a
group by order_id;
```

	order_id	count_	time_taken
1	1	1	10
2	2	1	10
3	3	2	21
4	4	3	30
5	5	1	10
6	7	1	10
7	8	1	21
8	10	2	16

The result shows the order_id, the rolls that have been ordered and time taken to prepare these rolls.

Q13- What was the average distance travelled for each customer?

Ans.

```
select customer_id, SUM(distance)/COUNT(order_id) as avg_distance_travel from
(select * from(
select *, ROW_NUMBER() over (partition by order_id order by diff) as rnk
from(
select a.order_id, a.customer_id, a.roll_id, a.not_include_items,
a.extra_items_included, a.order_date, b.driver_id, b.pickup_time,
cast(trim(replace(lower(b.distance), 'km', ' ')) as float) as distance, b.duration,
b.cancellation, DATEDIFF(minute, a.order_date, b.pickup_time) as diff
from customer_orders a
inner join driver_order b
on a.order_id = b.order_id
where b.pickup_time is not null)a)b
where rnk = 1)c
group by customer_id;
```

	customer_id	avg_distance_travel
1	101	20
2	102	18.4
3	103	23.4
4	104	10
5	105	25

Q14- Wat is the difference between the longest and the shortest delivery times for all orders?

Ans.

```
select MAX(duration_) - MIN(duration_) as diff_duration from
(select *, cast(case when duration like '%min%' then left(duration, CHARINDEX('m',
duration)-1) else duration end as int) as duration_
from driver_order
where duration is not null)a;
```

	diff_duration
1	30

Q15- What is the average speed for each driver for each delivery?

Ans.

```
select *, round((distance*60)/duration_,2) as avg_speed from
(select order_id, driver_id, sum(CAST(trim(REPLACE(lower(distance), 'km', ' ')) AS
float)) as distance, sum(cast(case when duration like '%min%' then left(duration,
CHARINDEX('m', duration)-1) else duration end as int)) as duration_
from driver_order where distance is not null
group by driver_id, order_id)a;
```

	order_id	driver_id	distance	duration_	avg_speed
1	1	1	20	32	37.5
2	2	1	20	27	44.44
3	3	1	13.4	20	40.2
4	4	2	23.4	40	35.1
5	5	3	10	15	40
6	7	2	25	25	60
7	8	2	23.4	15	93.6
8	10	1	10	10	60

Q16- What is the successful delivery percentage for each driver?

Ans.

```
select *, concat(round(cast(successful_delivery as float)/trips,2)*100, '%') as
success_rate from
(select driver_id, sum(can_perc) as successful_delivery, count(driver_id) as trips
from
(select *, case when lower(cancellation) like '%cancel%' then 0 else 1 end as can_perc
from driver_order)a
group by driver_id)b;
```

	driver_id	successful_delivery	trips	success_rate
1	1	4	4	100%
2	2	3	4	75%
3	3	1	2	50%