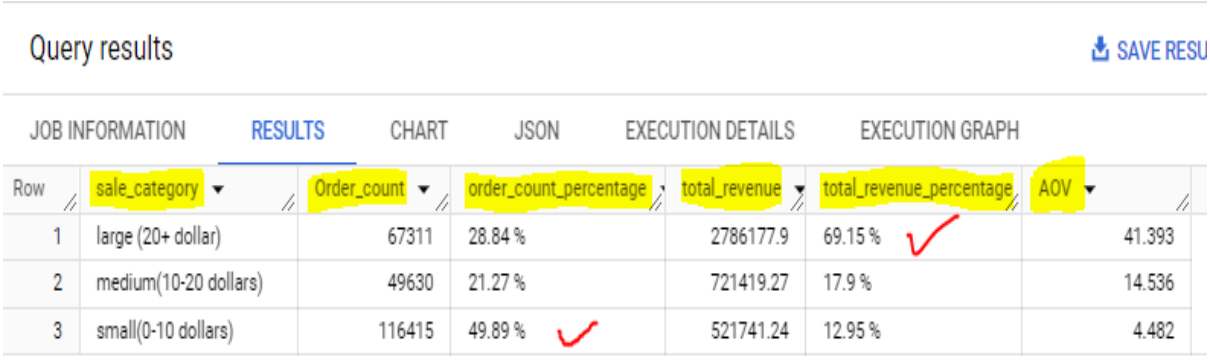
Q.1) Find the number of orders that have small, medium or large order value (small:0-10 dollars, medium:10-20 dollars, large:20+)

Query:

```
with order_sales as ( select BASKET_ID, sum(SALES_VALUE) as total_sales from
ecom.transaction group by BASKET_ID
),

order_category as ( select os.BASKET_ID , os.total_sales,
case
when os.total_sales>=0 and os.total_sales<=10 then "small (0-10 dollars)"
when os.total_sales>10 and os.total_sales<=20 then "medium (10-20 dollars)"
else "large (20+ dollars)"
end as sale_category

from order_sales as os )


select oc.sale_category, count(*) as Order_count,
concat(round( (count(*)/sum(count(*)) over () )*100,2)," %")  as order_count_percentage,

round(sum(oc.total_sales),3) as total_revenue,
concat(round( (sum(oc.total_sales)/sum(sum(oc.total_sales)) over () )*100,2)," %")  as
total_revenue_percentage,

round(sum(oc.total_sales)/count(*),3) as AOV,

from order_category oc group by oc.sale_category order by total_revenue desc ;
```

Result Screenshot:



Query results

SAVE RESU

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

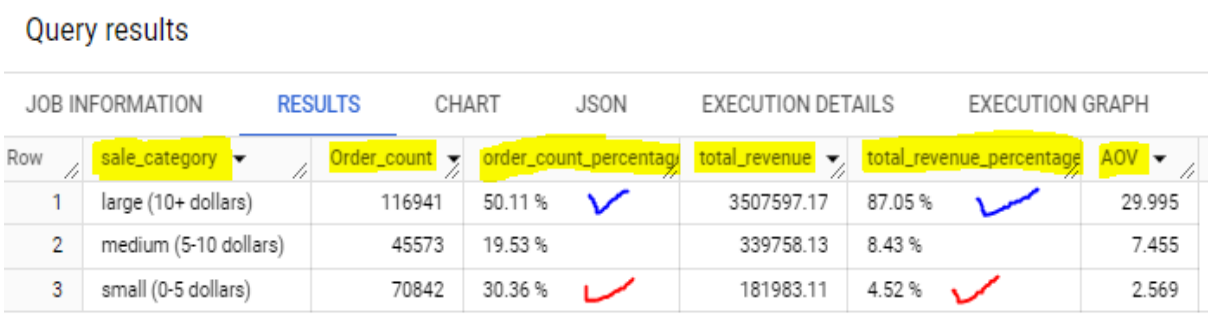
Row	sale_category	Order_count	order_count_percentage	total_revenue	total_revenue_percentage	AOV
1	large (20+ dollar)	67311	28.84 %	2786177.9	69.15 %	41.393
2	medium(10-20 dollars)	49630	21.27 %	721419.27	17.9 %	14.536
3	small(0-10 dollars)	116415	49.89 %	521741.24	12.95 %	4.482

Q.2) Find the number of orders that are small, medium or large order value (small:0-5 dollars, medium:5-10 dollars, large:10+)

Query:

```
with order_sales as ( select BASKET_ID, sum(SALES_VALUE) as total_sales from
ecom.transaction group by BASKET_ID ),


order_category as ( select os.BASKET_ID , os.total_sales,
case
when os.total_sales>=0 and os.total_sales<=5 then "small (0-5 dollars)"
when os.total_sales>5 and os.total_sales<=10 then "medium (5-10 dollars)"
else "large (10+ dollars)"
end as sale_category

from order_sales as os )


select oc.sale_category, count(*) as Order_count,
concat(round( (count(*)/sum(count(*)) over () )*100,2)," %")  as order_count_percentage,

round(sum(oc.total_sales),3) as total_revenue,
concat(round( (sum(oc.total_sales)/sum(sum(oc.total_sales)) over () )*100,2)," %")  as
total_revenue_percentage,

round(sum(oc.total_sales)/count(*),3) as AOV,

from order_category oc group by oc.sale_category order by total_revenue desc ;
```

Result Screenshot:



Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	sale_category	Order_count	order_count_percentage	total_revenue	total_revenue_percentage	AOV
1	large (10+ dollars)	116941	50.11 %	3507597.17	87.05 %	29.995
2	medium (5-10 dollars)	45573	19.53 %	339758.13	8.43 %	7.455
3	small (0-5 dollars)	70842	30.36 %	181983.11	4.52 %	2.569

Q.3) Find top 3 stores with highest foot traffic for each week (Foot traffic: number of customers transacting)

Query:

```
with t as ( select
WEEK_NO, STORE_ID , count(distinct household_key) as foot_traffic
from `ecom.transaction`
group by WEEK_NO, STORE_ID
order by  WEEK_NO, STORE_ID asc ) ,

t1 as ( select
t.*, dense_rank() over (partition by t.week_no order by foot_traffic desc) as store_ranking
from t order by  t.WEEK_NO asc )

select * from t1 where t1.store_ranking <=3
```

Result Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	WEEK_NO	STORE_ID	foot_traffic	store_ranking
1	1	32004	5	1
2	1	324	3	2
3	1	367	3	2
4	1	396	3	2
5	1	446	3	2
6	1	358	2	3
7	1	634	2	3
8	1	288	2	3
9	1	306	2	3
10	1	359	2	3

Q.3 (A) Unique overall Top 3 store list on weekly basis,

How many week they are Present in the Top 3 list?

If they are present then their overall “avg_ranking” in the top3 list.

Result Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETA

Row	store_id	total_sales	week_count_store_Present_top3_list	avg_ranking
1	367	134105.46	102	1.0
2	343	49179.25	62	2.0
3	381	36617.94	48	2.0
4	292	25024.57	32	3.0
5	356	24416.5	33	2.0
6	406	18295.82	13	3.0
7	32004	11110.57	20	2.0
8	361	9027.73	9	3.0
9	372	6251.05	9	3.0
10	321	4777.53	7	3.0

Q.3 (B) Stores' performance comparison - Overall Individual performance of each store on weekly sales basis, present in Transaction table.

Query:

```
with store as (

select  store_id, round(sum(SALES_VALUE),2) as store_total_sales, count(distinct WEEK_NO)as
week_count_of_store_sales,
count(distinct household_key) as no_of_customer_catered, count(distinct BASKET_ID) as
no_of_Order_catered,
round(sum(SALES_VALUE) / count(distinct BASKET_ID),2) as AOV_of_Store
from `ecom.transaction` group by STORE_ID order by store_total_sales desc

)


select  s.week_count_of_store_sales, count(*) as stores_count,
round(sum(s.store_total_sales),2)  as group_sales
from store s group by s.week_count_of_store_sales order by group_sales desc;
```

Result Screenshot 1 & 2:

Store table:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

low	store_id	store_total_sales	week_count_of_store_sales	no_of_customer_catered	no_of_Order_catered	AOV_of_Store
1	367	134105.46	102	248	6516	20.58
2	406	108814.66	101	121	4359	24.96
3	361	72493.72	86	90	4028	18.0
4	429	70752.61	100	91	2967	23.85
5	343	70265.6	94	221	4102	17.13
6	356	69026.02	102	110	3842	17.97
7	375	65788.5	102	199	3176	20.71
8	381	65400.58	102	269	4306	15.19
9	292	65201.6	102	190	3586	18.18
10	31782	61012.23	102	131	3734	16.34
11	321	59675.48	102	136	3497	17.06

Results per page: 50 1 – 50 of 552

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECU

Row	week_count_of_store_sales	stores_count	group_sales
1	102	44	1926924.92
2	101	27	922027.32
3	100	17	547792.18
4	99	8	227086.57
5	98	3	65605.32
6	97	1	14288.23
7	94	3	84008.76
8	92	1	5158.64
9	86	1	72493.72
10	78	1	19363.93
11	74	1	3614.87

Q.3 (C) Overall Ecommerce Company performance on weekly sales basis, as per Transaction table.

Query:

```
select WEEK_NO, round(sum(SALES_VALUE),2) as weekly_sales, count(distinct household_key) as no_of_customer_catered,count(distinct BASKET_ID) as no_of_Order_catered,count(distinct store_id) as no_of_store_id  from `ecom.transaction` group by WEEK_NO order by WEEK_NO
```

Result screenshot:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	WEEK_NO	weekly_sales	no_of_customer_catered	no_of_Order_catered	no_of_store_id
1	1	2453.52	81	120	61
2	2	5854.47	165	286	89
3	3	6822.98	218	415	88
4	4	7915.42	262	487	107
5	5	10038.67	350	652	111
6	6	12566.74	415	735	118
7	7	14041.61	470	856	117
8	8	15679.87	509	939	123
9	9	15653.89	588	1041	129

Results per page: 50 1 – 50 of 102

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	WEEK_NO	weekly_sales	no_of_customer_catered	no_of_Order_catered	no_of_store_id
93	93	40455.49	1186	2179	128
94	94	49079.33	1300	2579	124
95	95	44113.04	1263	2433	129
96	96	43258.24	1252	2405	123
97	97	44593.66	1276	2474	127
98	98	49447.7	1331	2606	122
99	99	50361.02	1267	2445	129
100	100	45633.8	1259	2396	135
101	101	42201.11	1217	2295	130
102	102	36112.16	1146	2025	122

Results per page: 200 1 – 102 of 102

Q.4) Create a basic customer profiling with first, last visit, number of visits, average money spent per visit and total money spent order by highest avg money.

Query:

```
with customer_spend_data as (

select household_key , min(DAY) as first_visit_day , max(Day) as last_visit_day,
MAX(MAX(DAY)) OVER () + 1 AS reference_last_day,

(MAX(MAX(DAY)) OVER () + 1 - max(DAY) ) as Recency_in_Days,

count(distinct BASKET_ID) as frequency_number_of_visits , round(sum(SALES_VALUE),3) as
total_spend,

round(sum(SALES_VALUE)/ count(distinct BASKET_ID) , 3) as avg_spent_per_order_visit

from `ecom.transaction` group by household_key order by total_spend desc ),


RFM_table as (

SELECT a.*,

--All percentiles for Recency
b.percentiles[offset(20)] AS r20,
b.percentiles[offset(40)] AS r40,
b.percentiles[offset(60)] AS r60,
b.percentiles[offset(80)] AS r80,
b.percentiles[offset(100)] AS r100,

--All percentiles for FREQUENCY
c.percentiles[offset(20)] AS f20,
c.percentiles[offset(40)] AS f40,
c.percentiles[offset(60)] AS f60,
c.percentiles[offset(80)] AS f80,
c.percentiles[offset(100)] AS f100,

--All percentiles for Monetary,
d.percentiles[offset(20)] AS m20,
d.percentiles[offset(40)] AS m40,
d.percentiles[offset(60)] AS m60,
d.percentiles[offset(80)] AS m80,
d.percentiles[offset(100)] AS m100,

FROM customer_spend_data a,
(SELECT APPROX_QUANTILES(Recency_in_Days, 100) percentiles FROM
customer_spend_data ) b,
(SELECT APPROX_QUANTILES(frequency_number_of_visits, 100) percentiles FROM
customer_spend_data ) c,
(SELECT APPROX_QUANTILES(avg_spent_per_order_visit, 100) percentiles FROM
customer_spend_data) d ),
```

```
RFM_score_table as (

select
rfm.household_key,
rfm.Recency_in_Days, rfm.frequency_number_of_visits , rfm.avg_spent_per_order_visit,


-- Recency --> High value ==> Low score

CASE
when rfm.Recency_in_Days <=rfm.r20 then 5
when rfm.Recency_in_Days <= rfm.r40 and rfm.Recency_in_Days > rfm.r20 then 4
when rfm.Recency_in_Days <= rfm.r60 and rfm.Recency_in_Days > rfm.r40 then 3
when rfm.Recency_in_Days <= rfm.r80 and rfm.Recency_in_Days > rfm.r60 then 2
else 1
End as r_score ,


-- frequency and monetary --> high value ==> high score

CASE
when rfm.frequency_number_of_visits <=rfm.f20 then 1
when rfm.frequency_number_of_visits <= rfm.f40 and rfm.frequency_number_of_visits > rfm.f20
then 2
when rfm.frequency_number_of_visits <= rfm.f60 and rfm.frequency_number_of_visits > rfm.f40
then 3
when rfm.frequency_number_of_visits <= rfm.f80 and rfm.frequency_number_of_visits > rfm.f60
then 4
else 5
End as f_score,


CASE
when rfm.avg_spent_per_order_visit <=rfm.m20 then 1
when rfm.avg_spent_per_order_visit <= rfm.m40 and rfm.avg_spent_per_order_visit > rfm.m20
then 2
when rfm.avg_spent_per_order_visit <= rfm.m60 and rfm.avg_spent_per_order_visit > rfm.m40
then 3
when rfm.avg_spent_per_order_visit <= rfm.m80 and rfm.avg_spent_per_order_visit > rfm.m60
then 4
else 5
End as m_score


from RFM_table as rfm  )




final as (select

rst.*, CAST(ROUND((rst.f_score + rst.m_score) / 2, 0) AS INT64) AS fm_score

from RFM_score_table rst),
```

```sql
-- Segmentation

rfm_final as (SELECT
f.household_key,
f.Recency_in_Days, f.frequency_number_of_visits , f.avg_spent_per_order_visit,

f.r_score, f.f_score, f.m_score,

f.fm_score,

CASE WHEN (r_score = 5 AND fm_score = 5)
OR (r_score = 5 AND fm_score = 4)
OR (r_score = 4 AND fm_score = 5)
THEN 'Champions'
WHEN (r_score = 5 AND fm_score =3)
OR (r_score = 4 AND fm_score = 4)
OR (r_score = 3 AND fm_score = 5)
OR (r_score = 3 AND fm_score = 4)
THEN 'Loyal Customers'
WHEN (r_score = 5 AND fm_score = 2)
OR (r_score = 4 AND fm_score = 2)
OR (r_score = 3 AND fm_score = 3)
OR (r_score = 4 AND fm_score = 3)
THEN 'Potential Loyalists'
WHEN r_score = 5 AND fm_score = 1 THEN 'Recent Customers'
WHEN (r_score = 4 AND fm_score = 1)
OR (r_score = 3 AND fm_score = 1)
THEN 'Promising'
WHEN (r_score = 3 AND fm_score = 2)
OR (r_score = 2 AND fm_score = 3)
OR (r_score = 2 AND fm_score = 2)
THEN 'Customers Needing Attention'
WHEN r_score = 2 AND fm_score = 1 THEN 'About to Sleep'
WHEN (r_score = 2 AND fm_score = 5)
OR (r_score = 2 AND fm_score = 4)
OR (r_score = 1 AND fm_score = 3)
THEN 'At Risk'
WHEN (r_score = 1 AND fm_score = 5)
OR (r_score = 1 AND fm_score = 4)
THEN 'Cant Lose Them'
WHEN r_score = 1 AND fm_score = 2 THEN 'Hibernating'
WHEN r_score = 1 AND fm_score = 1 THEN 'Lost'
END AS rfm_segment

FROM final f
ORDER BY f.household_key)


select rf.rfm_segment, count(*) as customer_count_per_rfm_segment,
concat (round((count(*) / sum(count(*)) over ())*100, 2),"%") as Percentage_of_total
from rfm_final rf group by rf.rfm_segment;
```

Result Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

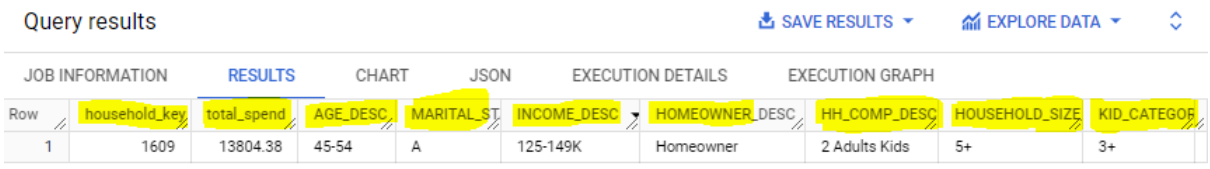
Row	rfm_segment	customer_count_per_rfm_segment	Percentage_of_total
1	Loyal Customers	589	23.56%
2	Potential Loyalists	423	16.92%
3	Champions	397	15.88%
4	Customers Needing Attention	392	15.68%
5	At Risk	338	13.52%
6	Hibernating	153	6.12%
7	Lost	89	3.56%
8	Cant Lose Them	68	2.72%
9	About to Sleep	23	0.92%
10	Promising	22	0.88%
11	Recent Customers	6	0.24%

Q.5) Do a single customer analysis selecting most spending customer for whom we have demographic information (because not all customers in transaction data are present in demographic table)(show the demographic as well as total spent)

Query:

```
with t1 as ( select * from `ecom.demographic`

where  MARITAL_STATUS_CODE != 'U'

and HOMEOWNER_DESC != 'Unknown' and HH_COMP_DESC != 'Unknown' and KID_CATEGORY_DESC != 'None/Unknown' ) ,


t2 as ( select

household_key , round(sum(SALES_VALUE),3) as total_spend

from `ecom.transaction`

group by household_key )


select t1.household_key , t2.total_spend, t1.* except( household_key ) from t1 left join t2 on t1.household_key = t2.household_key

order by t2.total_spend desc limit 1
```

Result Screenshot:



Q.6) Find products (product table : SUB_COMMODITY_DESC) which are most frequently bought together and the count of each combination bought together. do not print a combination twice (A-B / B-A)

Query:

```
with t as (select BASKET_ID,household_key,PRODUCT_ID  from `ecom.transaction`),

t1 as (select t.*, p.SUB_COMMODITY_DESC from t join `ecom.product` p on t.product_id =
p.PRODUCT_ID),

output_table as (select a.SUB_COMMODITY_DESC , b.SUB_COMMODITY_DESC, count(*) as
purchase_frequency

from t1 a inner join t1 b on a.basket_id=b.basket_id and a.household_key = b.household_key

where a.SUB_COMMODITY_DESC < b.SUB_COMMODITY_DESC

group by a.SUB_COMMODITY_DESC , b.SUB_COMMODITY_DESC

order by purchase_frequency desc ),


final_table as (select  ot.*,

case

when ot.purchase_frequency > 5000 then "Products_combination_present_in_5000+ orders"
when ot.purchase_frequency > 3000 and ot.purchase_frequency <= 5000 then
"Products_combination_present_in_3001 to 5000 orders"
when ot.purchase_frequency > 1000 and ot.purchase_frequency <= 3000 then
"Products_combination_present_in_1001 to 3000 orders"
when ot.purchase_frequency > 500 and ot.purchase_frequency <= 1000 then
"Products_combination_present_in_501 to 1000 orders"
when ot.purchase_frequency > 0 and ot.purchase_frequency <= 500 then
 "Products_combination_present_in_1 to 500 orders"
else "Products_not_present_in_Zero_Orders"

end as product_combination_purchase_frequency_category

from output_table ot )


select ft.product_combination_purchase_frequency_category,
count(*) as products_combination_count
from final_table ft group by ft.product_combination_purchase_frequency_category
order by products_combination_count
```

Result Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	SUB_COMMODITY_DESC	SUB_COMMODITY_DESC_1	purchase_frequency
1	FLUID MILK WHITE ONLY	YOGURT NOT MULTI-PACKS	5953
2	BANANAS	FLUID MILK WHITE ONLY	4365
3	FLUID MILK WHITE ONLY	SOFT DRINKS 12/18&15PK CA...	4326
4	FLUID MILK WHITE ONLY	MAINSTREAM WHITE BREAD	3934
5	BANANAS	YOGURT NOT MULTI-PACKS	3847
6	FLUID MILK WHITE ONLY	SHREDDED CHEESE	3840
7	FLUID MILK WHITE ONLY	SFT DRNK 2 LITER BTL CARB I...	3494
8	FRZN SS PREMIUM ENTREES/...	YOGURT NOT MULTI-PACKS	3344
9	BABY FOOD - BEGINNER	BABY FOOD JUNIOR ALL BRAN...	3290
10	SHREDDED CHEESE	YOGURT NOT MULTI-PACKS	3189

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	product_combination_purchase_frequency_category	products_combination_count
1	Products_combination_present_in_5000+ orders	1
2	Products_combination_present_in_3001 to 5000 orders	10
3	Products_combination_present_in_1001 to 3000 orders	196
4	Products_combination_present_in_501 to 1000 orders	850
5	Products_combination_present_in_1 to 500 orders	579033

Q.7) Find the weekly change in Revenue Per Account (RPA) (difference in spending by each customer compared to last week)(use lag function)

Query:

```
with all_customer_week_combination as (

-- dummy table with all combinations
select   a.* , b.*, 0 AS weekly_revenue  -- Add rows for missing weeks with zero revenue
from
(select distinct household_key from `ecom.transaction`)a,
(SELECT distinct week_no FROM `ecom.transaction`)b

UNION Distinct

-- original table

SELECT household_key,week_no,round(SUM(SALES_VALUE),2) AS weekly_revenue
FROM `ecom.transaction`
GROUP BY household_key,week_no

order by household_key,week_no ),



weekly_revenue_table as (
select acw.household_key,acw.week_no, round(sum(weekly_revenue),2) as weekly_revenue_1 from
all_customer_week_combination acw group by household_key,week_no
),




weekly_RPA_table as (select
wrt.* , lag(wrt.weekly_revenue_1,1,0) over( partition by wrt.household_key order by
wrt.week_no asc ) as last_week_total_revenue,

round(wrt.weekly_revenue_1 - lag(wrt.weekly_revenue_1,1,0) over( partition by
wrt.household_key order by wrt.week_no asc ) , 2) as weekly_change_RPA

from weekly_revenue_table wrt order by wrt.household_key,wrt.week_no )



select wrt.*,

case

when wrt.last_week_total_revenue != 0 and wrt.weekly_change_RPA < 0 then "Negative RPA and
also, shopped last week means decreased_week_spend"
when wrt.last_week_total_revenue != 0 and wrt.weekly_change_RPA > 0 then "Positive RPA and
also, shopped last week means increased_week_spend"
when wrt.last_week_total_revenue != 0 and wrt.weekly_change_RPA = 0 then "0 RPA and shopped
last week means consistent_Revenue"
```

```
when wrt.last_week_total_revenue = 0 and wrt.weekly_change_RPA < 0 then "Negative RPA but
Not shopped last week means Not_shopping_this_week"
when wrt.last_week_total_revenue = 0 and wrt.weekly_change_RPA > 0 then "Positive RPA but
Not shopped last week means Resume_shopping_this_week"
when wrt.last_week_total_revenue = 0 and wrt.weekly_change_RPA = 0 then "Not shopped last
week and 0 RPA means missing_shopping"


end as weekly_RPA_category

from weekly_RPA_table as wrt order by wrt.household_key,wrt.week_no
```

Result Screenshot:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	household_key	week_no	weekly_revenue_1	last_week_total_revenue	weekly_change_RPA	weekly_RPA_category
1	1	1	0.0	0.0	0.0	Not shopped last week and 0 RPA means missing_shopping
2	1	2	0.0	0.0	0.0	Not shopped last week and 0 RPA means missing_shopping
3	1	3	0.0	0.0	0.0	Not shopped last week and 0 RPA means missing_shopping
4	1	4	0.0	0.0	0.0	Not shopped last week and 0 RPA means missing_shopping
5	1	5	0.0	0.0	0.0	Not shopped last week and 0 RPA means missing_shopping
6	1	6	0.0	0.0	0.0	Not shopped last week and 0 RPA means missing_shopping
7	1	7	0.0	0.0	0.0	Not shopped last week and 0 RPA means missing_shopping
8	1	8	42.58	0.0	42.58	Positive RPA but Not shopped last week means Resume_shopping_this_week
9	1	9	0.0	42.58	-42.58	Negative RPA and also, shopped last week means decreased_week_spend

oad more

Understanding - Target_Customer_persona:

In Transaction table, **we have 2500 unique household_key** , but only 801 unique household_key , present in given Customer Demographic table. (1699 customers data not present in Demographic Table).

So, all the below insights are based on Only 801 unique household_key, present in given Customer Demographic table.

1. Based on Age wise

Query:

```
with td as (select t.household_key, t.* except (household_key),d.* except
(household_key) from `ecom.transaction` t inner join `ecom.demographic` d on
t.household_key=d.household_key )

select  AGE_DESC,count(distinct td.household_key) as customer_count, count(distinct
BASKET_ID) as order_count,round(sum(SALES_VALUE),3) as total_spend,
round(sum(SALES_VALUE)/ count(distinct BASKET_ID) , 3) as AOV_Age_wise,
round(sum(SALES_VALUE)/ count(distinct td.household_key) , 3) as Average_Customer_spend
from td group by td.AGE_DESC
order by total_spend desc
```

Result Screenshot:

Query results

SAVE RESULTS EX

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	AGE_DESC	customer_count	order_count	total_spend	AOV_Age_wise	Average_Customer_spend
1	45-54	288	43495	827984.9	19.036	2874.948
2	35-44	194	30491	622164.35	20.405	3207.033
3	25-34	142	19365	389545.17	20.116	2743.276
4	65+	72	10051	151606.81	15.084	2105.65
5	55-64	59	8642	150371.27	17.4	2548.666
6	19-24	46	6920	108404.35	15.665	2356.616

2. Based on Marital_Status

Query:

```
with td as (select t.household_key, t.* except (household_key),d.* except
(household_key) from `ecom.transaction` t inner join `ecom.demographic` d on
t.household_key=d.household_key )

select  MARITAL_STATUS_CODE,count(distinct td.household_key) as customer_count,
count(distinct BASKET_ID) as order_count,round(sum(SALES_VALUE),3) as total_spend,
round(sum(SALES_VALUE)/ count(distinct BASKET_ID) , 3) as AOV_MARITAL_STATUS_wise,
round(sum(SALES_VALUE)/ count(distinct td.household_key) , 3) as Average_Customer_spend
from td group by td.MARITAL_STATUS_CODE
order by total_spend desc
```

Result Screenshot:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	MARITAL_STATUS_	customer_count	order_count	total_spend	AOV_MARITAL_STATUS_wise	Average_Customer_spend
1	A	340	49269	1045561.93	21.221	3075.182
2	U	344	53040	904703.93	17.057	2629.953
3	B	117	16655	299810.99	18.001	2562.487

3. Based on Income_DESC

Query:

```
with td as (select t.household_key, t.* except (household_key),d.* except
(household_key) from `ecom.transaction` t inner join `ecom.demographic` d on
t.household_key=d.household_key )

select  Income_DESC,count(distinct td.household_key) as customer_count, count(distinct
BASKET_ID) as order_count,round(sum(SALES_VALUE),3) as total_spend,
round(sum(SALES_VALUE)/ count(distinct BASKET_ID) , 3) as AOV_Age_wise,
round(sum(SALES_VALUE)/ count(distinct td.household_key) , 3) as Average_Customer_spend
from td group by td.Income_DESC
order by total_spend desc
```

Result Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	Income_DESC	customer_count	order_count	total_spend	AOV_Age_wise	Average_Customer_spend
1	50-74K	192	27467	547139.05	19.92	2849.683
2	35-49K	172	24943	414471.79	16.617	2409.72
3	75-99K	96	12414	279738.22	22.534	2913.94
4	25-34K	77	12519	189846.93	15.165	2465.545
5	Under 15K	61	10714	169160.19	15.789	2773.118
6	15-24K	74	10336	151340.28	14.642	2045.139
7	125-149K	38	6298	150464.7	23.891	3959.597
8	150-174K	30	4274	126501.69	29.598	4216.723
9	100-124K	34	5402	100931.47	18.684	2968.573
10	250K+	11	2194	58935.02	26.862	5357.729
11	175-199K	11	1883	47268.08	25.103	4297.098
12	200-249K	5	520	14279.43	27.46	2855.886

4. Based on Homeowner_Desc

Query:

```
with td as (select t.household_key, t.* except (household_key),d.* except
(household_key) from `ecom.transaction` t inner join `ecom.demographic` d on
t.household_key=d.household_key )

select  homeowner_desc,count(distinct td.household_key) as customer_count,
count(distinct BASKET_ID) as order_count,round(sum(SALES_VALUE),3) as total_spend,
round(sum(SALES_VALUE)/ count(distinct BASKET_ID) , 3) as AOV_Age_wise,
round(sum(SALES_VALUE)/ count(distinct td.household_key) , 3) as Average_Customer_spend
from td group by td.homeowner_desc
order by total_spend desc
```

Result Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	homeowner_desc	customer_count	order_count	total_spend	AOV_Age_wise	Average_Customer_spend
1	Homeowner	504	72090	1519166.59	21.073	3014.219
2	Unknown	233	35325	561266.78	15.889	2408.87
3	Renter	42	7587	118735.77	15.65	2827.042
4	Probable Owner	11	1879	27236.49	14.495	2476.045
5	Probable Renter	11	2083	23671.22	11.364	2151.929

Insights:

- From Query -1 and Query-2 output, we can understand the distribution of orders across small, medium, and large categories.
  1. Almost **50%** of Total Orders are in “small (0-10 dollars)” Category, which further breakdown as **30%** of Total orders are in small (0-5 dollars) and rest **20%** of Total orders in small (5-10 dollars) category.

     ➔ But this small (0-10 dollars) category‘s contribution to total revenue is Only **13%** of Total Revenue.

  2. Almost **29%** of Total Orders are in “large (20+ dollars)” Category. But this large (20+ dollars) category‘s contribution to total revenue is **Highest one** i.e. **69%** of Total Revenue.

  3. Almost **21%** of Total Orders are in “medium (10-20 dollars)” Category. But this medium (10-20 dollars) category‘s contribution to total revenue is Only **18%** of Total Revenue.

Recommendations:

Based on the above insights from this query, we analysed that a significant portion (50%) of orders fall into the "small (0-10 dollars)" category, since the e-commerce company can implement various strategies to improve profitability and revenue:

- **Targeted Promotions:** Offer discounts or special deals for customers who typically make small orders to incentivize them to spend more.
- **Free Shipping Thresholds:** Implement a free shipping threshold that encourages customers to add more products to their cart and reach a higher order value to qualify for free shipping.
- **Product Pricing and Assortment:** Understanding the dominant order value segment (small, medium, or large) can influence product pricing strategy. As a significant portion of customers tend towards small orders, the company might consider offering more competitively priced products or the company might consider offering smaller package of product options to cater to that segment.
- **Product Bundles:** As a significant portion of customers tend towards small orders; the company might consider offering to create curated product bundles that cater to specific customer needs or interests, potentially combining high-margin items with lower-priced ones to increase overall order value.

- **Upselling and Cross-selling:** Implement upselling and cross-selling strategies to recommend complementary products during the checkout process, potentially nudging customers towards larger orders.

- From Query -3, 3A, 3B and 3C output, we identify the top 3 stores (likely referring to vendors or brands on the platform) with the highest number of customers making transactions each week. Also, all stores performance comparison across platform.

  This query can reveal valuable insights into customer behavior and store performance, potentially leading to strategies for increasing profitability and revenue for the e-commerce company.

  Potential Insights:

  By analyzing the results, the e-commerce company can gain valuable insights into:

  - **Brand Awareness/ recognition:** "Store id = 367"consistently in the top 3 list for All 102 weeks, might have stronger brand recognition or a loyal customer base. This can inform targeted marketing efforts for other stores.
  - **Customer Preferences:** The data reveals only approx. Top 110 stores out of 552 stores based on overall sales, are attracting the most customers each week. This can indicate popular brands or product categories that resonate well with the customer base.
  - **Store Performance Comparison:** Comparing the performance of different stores across weeks , we identify top 110 stores performance based on revenue and rest 442 stores are underperforming stores.

Recommendations:

Based on these insights, the e-commerce company can implement strategies to increase profitability and revenue:

- **Promote Top Stores:** Highlight the top-performing stores on the platform through targeted promotions, banner placements, or dedicated sections. This can attract more customers to these stores and potentially increase sales. Analyze product selection, pricing, promotions, and marketing efforts of these stores to replicate their success in other stores.

- **Optimize Underperforming Stores:** Analyze the reasons behind underperformance for certain stores. It could be due to factors like limited product offerings, lack of promotions, or negative customer reviews. This might involve working with the vendors to improve product offerings, pricing, or marketing strategies.

- **Platform Improvements:** Enhance the platform's search and recommendation features to ensure customers can easily discover stores and products that align with their interests.

- **Seasonal Trends:** Analyzing the data over a longer period can reveal seasonal trends. Certain stores might see a surge in customer activity during specific times of the year.
- **Improve Product Assortment:** Analyze which product categories are driving customer activity for top stores. This can inform decisions about expanding the product assortment for other stores or introducing similar successful products from different vendors.

- **Data-Driven Inventory Management:** Knowing which stores are performing well can inform inventory management decisions. Amazon can prioritize stocking popular products in high-traffic stores, reducing the risk of stockouts and lost sales.

- From Query -4 RFM (Recency, Frequency, Monetary) analysis output, we are able to do and understand the behaviour-based customer segmentation for Marketing purpose. It groups customers based on history – how recently and how often they bought, and how much they spent.

  It helps the marketing team to divide customers into various categories or clusters, who are more likely to respond to promotions or future personalisation service.

  Potential Insights:

  By analyzing the results, the e-commerce company can gain valuable insights:

  - Top 5 customer segments are “Loyal Customers (23.56%), Potential Loyalists (16.92%) , Champions (15.88%) , Customers Needing Attention(15.68%) and At Risk(13.52%)”.

Recommendations:

Based on these insights, the e-commerce company can implement strategies to increase profitability and revenue:

Customer Segment	Activity	Actionable Tip
Champions	Bought recently, buy often and spend the most!	Reward them. Can be early adopters for new products. Will promote your brand.
Loyal Customers	Spend good money with us often. Responsive to promotions.	Upsell higher value products. Ask for reviews. Engage them.
Potential Loyalist	Recent customers, but spent a good amount and bought more than once.	Offer membership / loyalty program, recommend other products.
Recent Customers	Bought most recently, but not often.	Provide on-boarding support, give them early success, start building relationship.
Promising	Recent shoppers, but haven't spent much.	Create brand awareness, offer free trials
Customers Needing Attention	Above average recency, frequency and monetary values. May not have bought very recently though.	Make limited time offers. Recommend based on past purchases. Reactivate them.
About To Sleep	Below average recency, frequency and monetary values. Will lose them if not reactivated.	Share valuable resources, recommend popular products / renewals at discount, reconnect with them.
At Risk	Spent big money and purchased often. But long time ago. Need to bring them back!	Send personalized emails to reconnect, offer renewals, provide helpful resources.
Can't Lose Them	Made biggest purchases, and often. But haven't returned for a long time.	Win them back via renewals or newer products, don't lose them to competition, talk to them.
Hibernating	Last purchase was long back, low spenders and low number of orders.	Offer other relevant products and special discounts. Recreate brand value.
Lost	Lowest recency, frequency and monetary scores.	Revive interest with reach out campaign, ignore otherwise.

- From Query -6 output, we can identify the buying behaviour of customers in terms of products combinations. This helps to know that how often two products are purchased together within the same basket. Higher co-occurrence suggests a stronger association between products.

  Potential Insights:

  There are 233356 orders, 2500 customers and 580090 Product combinations in the given data.

  Out of 580090 Product combinations, only few (around top150 based on purchase_frequency value) combinations are present in 1 to 3% of total orders i.e. 233356 orders.

  Highest selling Product combination is “FLUID MILK WHITE ONLY” and “YOGURT NOT MULTI-PACKS”, which is present in 5953 orders out of total orders i.e. 233356 orders.

Recommendations:

Based on these insights, the e-commerce company can implement strategies to increase profitability and revenue:

- **Product Recommendations:** Recommend products frequently bought together on product pages and during checkout to increase average order value.
- **Personalized Offers:** Use customer purchase history to suggest product bundles or targeted promotions based on frequently bought together products.
- **Store Layout Optimization:** Analyze co-occurrence data to inform store layout decisions, placing complementary products in close proximity to encourage impulse purchases.
- **Inventory Management:** Identify products that are often bought together to optimize inventory levels and minimize stockouts.

- From Query -7 output, Weekly RPA Change provides valuable insights into customer behavior and spending trends. It essentially tracks how a customer's spending habits are evolving over time.

Recommendations:

Based on these insights, the e-commerce company can implement strategies to increase profitability and revenue:

- **Targeted Campaigns:** Reach out to customers with missing weeks through targeted email campaigns or personalized offers to re-engage them.
- **Customer Segmentation:** Segment customers based on their weekly RPA patterns (consistent, erratic, seasonal) to tailor marketing and promotional strategies.
- **Retention Strategies:** Develop targeted retention strategies for customers with declining weekly RPA or frequent missing weeks to prevent churn.