# E-commerce SQL Analysis

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

**Description:**

- Here we are calculating the revenue by creating bins of 0-10,10-20 and 20+ called as order value and querying the number of products within each group.
- The sales_value and quantity multiplied here refers to the revenue generated by the products.
- We ordered the output such that a group with a larger number of products comes on top.

- Small – revenue(0-10 dollars)
- Medium - revenue(10-20 dollars)
- Large - revenue(20+ dollars)
- Product_count - number of products in each group.

**SQL Query:**

```
SELECT order_value,count(PRODUCT_ID) product_count
from (
select PRODUCT_ID,
case
  when quantity*sales_value > 0 and quantity*sales_value < 10 then 'small'
  when quantity*sales_value > 10 and quantity*sales_value < 20 then 'medium'
  when quantity*sales_value > 20 then 'large'
  end as order_value
FROM `plated-analyzer-386814.ecommerce.transaction`
 )
 where order_value is not null
 group by order_value
 order by order_value desc
```

Output:

| order_value | product_count |
|---|---|
| small | 1177069 |
| medium | 59279 |
| large | 39958 |

# Find the top 3 stores with highest foot traffic for each week.

Description:

- We are finding the top3 stores with the highest foot traffic(most number of visitors) for each week.
- We here used Cte to avoid confusion within the subqueries.
- We counted the trans_time in the table w.r.t Week_no and store_id and named as foot_traffic.
- Then we applied the dense_rank() function on the foot traffic column and filtered out the top three ranks from the table, which resulted in the required output.

SQL Query:

```
with cte as
(select  WEEK_NO,store_id,
count(TRANS_TIME) foot_traffic
 from `ecommerce.transaction`
 group by 1,2
 order by foot_traffic desc
),
cte2 as (
  select *, dense_rank()over(partition by week_no order by foot_traffic desc) as rnk
from cte)

select week_no,store_id,foot_traffic from cte2
where rnk<4
order by week_no
```

**OUTPUT:**

| Row | week_no | store_id | foot_traffic |
|---|---|---|---|
| 1 | 1 | 324 | 80 |
| 2 | 1 | 321 | 68 |
| 3 | 1 | 32004 | 67 |
| 4 | 2 | 375 | 99 |
| 5 | 2 | 292 | 86 |
| 6 | 2 | 315 | 74 |
| 7 | 3 | 367 | 169 |
| 8 | 3 | 375 | 158 |
| 9 | 3 | 356 | 96 |
| 10 | 4 | 367 | 249 |

**# Create a basic customer profiling with first visit,last visit,number of visits,average money spent per visit and total money spent, ordered by highest avg money.**

**Description:**

- The problem statement clearly asked about customer profiling(analysis) for
- their information over the metrics provided.
- We joined the required tables of demographics and transactions to get the sale_value,number of visits and other required columns.
- We used window functions over the day for first,alst visits and number of total visits for each customer.
- We calculated the average spent and total spent by columns sales and quantity and ordered the output by highest average spent.
- We can see that the customer with id 755 has the highest average spent on each visit making him top on the list.
- Customer Analysis plays a vital role for performance of the wen=bsite and calculating the metrics required for the increase or decrease in the supply or demand.

SQL Query:

```
Select distinct d.household_key,min(day)over(partition by
d.household_key)first_visit,max(day)over(partition by d.household_key) last_visit,
count(day)over(partition by d.household_key)
no_of_visits,round(avg(sales_value)over(partition by d.household_key),2) avg_spent,
round(sum(sales_value*QUANTITY)over(partition by d.household_key)) total_spent
 from `ecommerce.transaction` t join `ecommerce.demographic` d
 on d.household_key = t.household_key
order by avg_spent desc
```

OUTPUT:

| Row | household_key | first_visit | last_visit | no_of_visits | avg_spent | total_spent |
|---|---|---|---|---|---|---|
| 1 | 755 | 36 | 709 | 576 | 9.48 | 72283100.0 |
| 2 | 1357 | 59 | 710 | 255 | 6.62 | 2905595.0 |
| 3 | 2162 | 98 | 710 | 1165 | 6.23 | 9045084.0 |
| 4 | 2097 | 26 | 645 | 710 | 6.23 | 17284560.0 |
| 5 | 101 | 116 | 708 | 583 | 6.23 | 3327739.0 |
| 6 | 853 | 65 | 707 | 1232 | 5.59 | 12713742.0 |
| 7 | 513 | 21 | 705 | 1084 | 5.57 | 14920.0 |
| 8 | 2203 | 42 | 711 | 501 | 5.49 | 1794892.0 |
| 9 | 13 | 101 | 709 | 1174 | 5.47 | 39279351.0 |
| 10 | 671 | 77 | 694 | 474 | 5.35 | 9664692.0 |

#.Do a single customer profiling with most spending customer from whom we have demographic info(because not all customers in transaction table are present demographic table)(show  demographic as well as total spent)

Description:

- We are doing customer profiling for the one person who is the highest spending customer.
- We joined the transactions and demographic tables to get the customer information.
- We query out the first visit,last visit,total number of visits, total amount spent, income range,marital status.
- Here the household_id 755 is the customer who spent the highest.

SQL Query:

```
select distinct d.household_key,min(day)over(partition by d.household_key )first_visit,
  max(day)over(partition by d.household_key) last_visit,
  count(day)over(partition by d.household_key) no_of_visits,
  round(avg(sales_value)over(partition by d.household_key),2) avg_spent,
round(sum(Quantity* sales_value)over(partition by d.household_key)) as total_amt_spent,
 MARITAL_STATUS_CODE = 'U' marital_status,INCOME_DESC as income
  from `ecommerce.demographic` d left join `ecommerce.transaction` t
  on d.household_key = t.household_key
 order by total_amt_spent desc  limit 1
```

OUTPUT:

| Row | household_key | first_visit | last_visit | no_of_visits | avg_spent | total_amt_spent | marital_stat | income |
|---|---|---|---|---|---|---|---|---|
| 1 | 755 | 36 | 709 | 576 | 9.48 | 72283100.0 | true | 50-74K |

#. Find products(product table: sub_commodity_desc) which are most frequently bought together and count of each combination, do not print a combination twice.

Description:

- Here we are querying the top  commodities which are brought frequently in combination.
- We query the number of commodities that are bought in combination for better understanding.
- We can see that Seasoning & spics top's the list with count of 629 frequently bought.

SQL Query:

```
  select SUB_COMMODITY_DESC, count(SUB_COMMODITY_DESC) as no_of_commodity
  from `ecommerce.product`
  where SUB_COMMODITY_DESC like '%&%'  or  SUB_COMMODITY_DESC like '%/%'
  group by SUB_COMMODITY_DESC
  order by count(SUB_COMMODITY_DESC) desc
limit 10
```

**OUTPUT:**

| Row | SUB_COMMODITY_DESC | no_of_commodity |
|---|---|---|
| 1 | SPICES & SEASONINGS | 629 |
| 2 | SOFT DRINKS 12/18&15PK CAN CAR | 404 |
| 3 | GADGETS/TOOLS | 395 |
| 4 | FRZN SS PREMIUM ENTREES/DNRS/N | 392 |
| 5 | SS ECONOMY ENTREES/DINNERS ALL | 297 |
| 6 | RTS SOUP: CHUNKY/HOMESTYLE ET | 290 |
| 7 | FRZN SS PREMIUM ENTREES/DNRS/T | 286 |
| 8 | HARDBACK/TRADE EVERYDAY | 254 |
| 9 | SNACKS/APPETIZERS | 251 |
| 10 | FITNESS&DIET - BARS | 247 |

**# Find the weekly change in revenue per account(RPA)(difference in spending by each customer compared to last week).**

**Description:**

- We here query the revenue change per each customer across each week by comparing to previous week's revenue.
- We joined the tables transactions and demographics for required metrics for each customer.
- We got the week number,customer_id, revenue or total spent by the customer.
- Here we subtracted the current week revenue with previous week revenue for each customer and got the difference in revenue.
- The average difference in revenue per customer in comparison to previous week is also calculated by taking the average of the revenue difference w.r.t each customer with each week number.
- We ordered the output with respect to customer_id and submitted two snapshots with different customer_id.

**SQL Query:**

```
with cte as(
select distinct d.household_key,week_no,
round(sum(quantity*SALES_VALUE)over(partition by d.household_key,week_no)) as
cus_revenue
from `ecommerce.transaction` t right join `ecommerce.demographic` d
on d.household_key = t.household_key
order by 1,2
),
cte1 as (
select household_key,week_no,cus_revenue, cus_revenue - lag(cus_revenue,1)over(partition
by household_key order by week_no) diff_rev
from cte
order by 1,2
)
select *,avg(diff_rev)over(partition by household_key) from cte1
order by household_key
```

**OUTPUT:**

| Row | household_key | week_no | cus_revenue | diff_rev | f0_ |
|---|---|---|---|---|---|
| 1 | 1 | 8 | 49.0 | null | -0.06153846153... |
| 2 | 1 | 10 | 22.0 | -27.0 | -0.06153846153... |
| 3 | 1 | 13 | 14.0 | -8.0 | -0.06153846153... |
| 4 | 1 | 14 | 34.0 | 20.0 | -0.06153846153... |
| 5 | 1 | 15 | 11.0 | -23.0 | -0.06153846153... |
| 6 | 1 | 16 | 14.0 | 3.0 | -0.06153846153... |
| 7 | 1 | 17 | 20.0 | 6.0 | -0.06153846153... |
| 8 | 1 | 19 | 50.0 | 30.0 | -0.06153846153... |
| 9 | 1 | 20 | 52.0 | 2.0 | -0.06153846153... |
| 10 | 1 | 22 | 44.0 | -8.0 | -0.06153846153... |

| Row | household_key | week_no | cus_revenue | diff_rev | f0_ |
|---|---|---|---|---|---|
| 67 | 7 | 4 | 59.0 | null | -0.34782608695... |
| 68 | 7 | 5 | 62.0 | 3.0 | -0.34782608695... |
| 69 | 7 | 27 | 34.0 | -28.0 | -0.34782608695... |
| 70 | 7 | 28 | 31.0 | -3.0 | -0.34782608695... |
| 71 | 7 | 29 | 16.0 | -15.0 | -0.34782608695... |
| 72 | 7 | 30 | 6.0 | -10.0 | -0.34782608695... |
| 73 | 7 | 31 | 8.0 | 2.0 | -0.34782608695... |
| 74 | 7 | 32 | 61.0 | 53.0 | -0.34782608695... |
| 75 | 7 | 34 | 47.0 | -14.0 | -0.34782608695... |
| 76 | 7 | 40 | 27.0 | -20.0 | -0.34782608695... |

**# Department wise total discounts given to customers and the revenue made by the departments.**

Description:

- Here we query the department wise discounts provided to the customers and revenue made by each department.
- We joined the product and transactions table for the required metric to be calculated for each department.
- We added the different discount columns and grouped them according to the departments.
- We calculated the revenue for each  department.
- We ordered the output by revenue  desc.
- We observe that the discounts don't bring the most revenue, Grocery provided more discounts but KIOSK-GAS department leads the list with highest revenue.

SQL Query:

```
SELECT
DEPARTMENT,abs(round(sum(COUPON_DISC+RETAIL_DISC+COUPON_MATCH_DISC)))
total_discount,round(sum(SALES_VALUE*quantity)) as revenue
FROM `plated-analyzer-386814.ecommerce.transaction` t join `ecommerce.product` p
on t.product_id = p.PRODUCT_ID
group by DEPARTMENT
order by revenue desc limit 10
```

OUTPUT:

| Row | DEPARTMENT | total_discount | revenue |
|---|---|---|---|
| 1 | KIOSK-GAS | 8506.0 | 3359884678.0 |
| 2 | MISC SALES TRAN | 1531.0 | 700067265.0 |
| 3 | GROCERY | 423959.0 | 3227567.0 |
| 4 | DRUG GM | 56031.0 | 759344.0 |
| 5 | MEAT | 87842.0 | 437302.0 |
| 6 | PRODUCE | 34814.0 | 386766.0 |
| 7 | MEAT-PCKGD | 60524.0 | 311348.0 |
| 8 | DELI | 11354.0 | 150943.0 |
| 9 | PASTRY | 8816.0 | 84641.0 |
| 10 | NUTRITION | 6234.0 | 81959.0 |

# Top 10 revenue generated manufacturer and their respective departments

Description:

- In this question we query the manufacturer and their respective departments and the number of products produced by each manufacturer and the revenue generated.
- We joined the required tables transactions and products for the required analysis.
- We counted the number of products per manufacturer and then by department and calculated the revenue for each manufacturer.
- Here we took a sample of 10 manufacturers and in that id with 69 leads the list with more number of departments and products manufacturing.

SQL Query:

```
SELECT p.Manufacturer,department,count(t.PRODUCT_ID)
no_of_products,round(sum(sales_value*quantity)) as revenue
FROM `plated-analyzer-386814.ecommerce.transaction` t join `ecommerce.product` p
on t.product_id = p.PRODUCT_ID
group by 1,2
order by revenue desc
limit 10
```

OUTPUT:

| Row | Manufacturer | department | no_of_products | revenue |
|---|---|---|---|---|
| 1 | 69 | KIOSK-GAS | 10936 | 3359884678.0 |
| 2 | 69 | MISC SALES TRAN | 1709 | 700043838.0 |
| 3 | 69 | GROCERY | 292749 | 889370.0 |
| 4 | 2 | PRODUCE | 73721 | 195794.0 |
| 5 | 1208 | GROCERY | 17377 | 155095.0 |
| 6 | 103 | GROCERY | 18980 | 123124.0 |
| 7 | 317 | GROCERY | 19628 | 88347.0 |
| 8 | 1251 | GROCERY | 17712 | 82597.0 |
| 9 | 69 | MEAT-PCKGD | 12651 | 67193.0 |
| 10 | 544 | GROCERY | 23074 | 64165.0 |

# Top 5 weeks with highest purchases

Description:

- We query out the top 5 weeks with the highest purchases.
- We count the number of purchases(basket_id) group by each week and sort them in decreasing order of number of purchases such that the highest value hits the top of the list.

SQL Query:

```
SELECT week_no,count(basket_id) no_of_purchases
FROM `plated-analyzer-386814.ecommerce.transaction`
group by 1
order by no_of_purchases desc limit 5
```

**OUTPUT:**

| Row | week_no | no_of_purchases |
|---|---|---|
| 1 | 92 | 16519 |
| 2 | 99 | 16151 |
| 3 | 85 | 15725 |
| 4 | 68 | 15687 |
| 5 | 94 | 15680 |

# Marital status wise purchases and quantity

Description:

- We here query the quantity and purchase according to the marital status.
- We wrote a simple case statement to abbreviate the marital status column.
- Then we calculated the total quantity ,total purchase for each group and ordered the output by purchase decrement such that highest purchase group comes on top of the list.

SQL Query:

```
SELECT
 distinct case
  when MARITAL_STATUS_CODE = 'A' then 'Married'
  when marital_status_code = 'B' then 'Unmarried'
  when MARITAL_STATUS_CODE = 'U' then 'Unknown'
  end as marital_status,
sum(quantity) quantity,round(sum(sales_value),2) purchase
FROM
`plated-analyzer-386814.ecommerce.transaction` t  join `ecommerce.demographic` d on
t.household_key = d.household_key
group by 1
order by purchase desc
```

OUTPUT:

| Row | marital_status | quantity | purchase |
|---|---|---|---|
| 1 | Married | 41685435 | 1045561.93 |
| 2 | Unknown | 30958411 | 904703.93 |
| 3 | Unmarried | 9495157 | 299810.99 |

# most frequent time for transaction to occur

Description:

- Here we calculated the highest possible timings for a transaction to occur from the given data.
- We simply grouped the trans_time column towards each time and counted the total times it occurs.
- From the below output 1836 means (6:36 pm) ,1753 means (5:53 pm) are the timings where the transactions occur frequently.
-

SQL Query:

```
select trans_time, count(TRANS_TIME)
from `ecommerce.transaction`
group by 1
order by 2 desc
limit 10
```

OUTPUT:

| | | |
|---|---|---|
| 1 | 1836 | 2602 |
| 2 | 1753 | 2602 |
| 3 | 1718 | 2587 |
| 4 | 1744 | 2556 |
| 5 | 1842 | 2537 |
| 6 | 1724 | 2524 |
| 7 | 1732 | 2504 |
| 8 | 1719 | 2500 |
| 9 | 1755 | 2489 |
| 10 | 1815 | 2468 |

## Recommendations:

**The Data provided is of an ecommerce website where the customer demographics,their purchase products and the transactions are recorded.**

1. **The Analysis done on the data provides us with different kinds of information in which the spent by each customer and average rate of spending for a visit is the keys for the website growth and success.**
2. **The revenue generated by the departments,products and commodities tells us about the performance of each category which has to be improved accordingly.**
3. **The more time spent on the website makes the customer buy anything related or non-related to him.**
4. **Suggestions and recommendations are the key parts for the sales improvement in any of the commodity or product as it feels like best customer experience rather than searching again and again for the required products.**
5. **Maintaining the payments gateway to be very efficient is also a key factor for a smoother experience of shopping.**
6. **Providing the alternative discounts ,ads,and notifications also helps in retaining the customer and bring them back to the website or app and buy for something.**
7. **Choosing the right product for the customer is always confusion for customer, So when they click on ad or notification they should be drawn to the best product with review,quality and expert suggestions to help them in buying the trustworthy product on our website, which leads for loyalty of customers for user experience.**
8. **Making an attractive UI is also a factor for the customers' interest in our business.**
9. **Categorizing the products according to their performance helps us in easy understanding of the sales and profit for the department of products.**
10. **Making the Products available across different locations and price ranges gives a good customer base for any online ecommerce business website or app.**

**THANK YOU**