Cohort Analysis- Monthly Customer Retention for a specific year

This query produces a cohort table used to track the monthly customer retention for each cohort in a month. It can help companies understand the customer base because old customers are valuable since they can give referrals, provide feedback to improve products and promote products on social media. By knowing the retention rate, the company can know whether it is required to do something on improving the customer retention or it is already doing good. This query requires one parameter, which is year. The rate in this query is measured using percentage.

```
set linesize 500
set pagesize 20

accept v_year char prompt 'Enter the year (yyyy) :'

column  month           format a9        heading 'Month'
column  new_customers   format 99999     heading 'New Customers'

ttitle col 60 'Monthly customer retention rate on year &v_year' skip 1-
       col 60  ========================================== skip 2

-- function to convert number into month
create or replace function no_to_month_name(no in number)
return char is
    v_month_str char(9);
begin
    case no
    when 1 then
        v_month_str := 'JANUARY';
    when 2 then
        v_month_str := 'FEBRUARY';
    when 3 then
        v_month_str := 'MARCH';
    when 4 then
        v_month_str := 'APRIL';
    when 5 then
        v_month_str := 'MAY';
    when 6 then
        v_month_str := 'JUNE';
    when 7 then
        v_month_str := 'JULY';
    when 8 then
        v_month_str := 'AUGUST';
    when 9 then
        v_month_str := 'SEPTEMBER';
    when 10 then
        v_month_str := 'OCTOBER';
    when 11 then
        v_month_str := 'NOVEMBER';
    when 12 then
        v_month_str := 'DECEMBER';
    end case;
    return v_month_str;
```

```
end;
/

SET LINESIZE 250
SET PAGESIZE 200

-- get first appearance of each customer id vs month
create or replace view first_purchase as
    select
        customerNumber,
        min(cal_month_no_in_year) as first_purchase_month
    from
        sales_fact
    join
        dim_date using (date_key)
    join
        dim_customers using (customer_key)
    where
        cal_year = &v_year
    group by
        customerNumber;


-- check user activity in each month
create or replace view customer_subsequent_purchase as
    select
        distinct customerNumber,
        (cal_month_no_in_year - first_purchase_month) as subsequent_month
    from
        sales_fact
    join
        dim_date using (date_key)
    join
        dim_customers using (customer_key)
    join
        first_purchase fa using (customerNumber)
    where
        cal_year = &v_year
    order by
        customerNumber,
        subsequent_month;

-- count number of first purchase customers in each month
create or replace view cohort_size as
    select
        first_purchase_month,
        count(first_purchase_month) as first_customers_qty
    from
        first_purchase
    group by
        first_purchase_month
    order by first_purchase_month;

-- retention table first_purchase X subsequent_month
create or replace view retention_table as
```

```
    select
        first_purchase_month,
        subsequent_month,
        count(first_purchase_month) as retained_qty
    from
        customer_subsequent_purchase
    join
        first_purchase using (customerNumber)
    group by
        first_purchase_month,
        subsequent_month
    order by
        first_purchase_month,
        subsequent_month;


create or replace view cohort_analysis as
    select
      no_to_month_name(first_purchase_month) as month,
      first_customers_qty as new_customers,
      subsequent_month,
      retained_qty * 100 / first_customers_qty as retention_rate
    from
        retention_table
    left join
        cohort_size using (first_purchase_month)
    where
        first_purchase_month IS NOT NULL
    order by
        first_purchase_month,
        subsequent_month;


select * from cohort_analysis
pivot(
    sum(trunc(retention_rate,2))
    for subsequent_month
    in (
        0,1,2,3,4,5,6,7,8,9,10,11
    )
);


clear column
ttitle off
clear breaks
```

Monthly customer retention rate on year 2020

Month	New Customers	0	1	2	3	4	5	6	7	8	9	10	11
JANUARY	8732	100	25.21	25.88	25.72	27	25.36	26.53	29.09	24.12	25.6	29.24	27.95
FEBRUARY	5723	100	25.44	23.23	26.8	26.43	26.07	28.13	24.96	25.44	29.79	29.21	
MARCH	4256	100	23.56	26.5	26.08	26.38	30.12	25.93	24.74	30	27.74		
APRIL	3017	100	25.65	26.05	25.02	27.94	23.66	24.79	28.8	28.27			
MAY	2499	100	26.69	26.69	27.53	23.6	25.05	30.45	26.53				
JUNE	1798	100	27.3	29.42	24.24	23.74	28.69	28.8					
JULY	1425	100	29.26	22.59	24.84	28.21	28.56						
AUGUST	1037	100	23.91	23.81	29.12	29.99							
SEPTEMBER	677	100	25.4	32.49	26.44								
OCTOBER	507	100	26.82	32.14									
NOVEMBER	462	100	23.59										
DECEMBER	312	100											

The new customers column is the quantity of new customers of that month, calculated based on the month that the customer made the first purchase. The 0 to 11 is the subsequent month, for example, if the month is February and col is 1, which means it is customer retention month in March which in this case is 25.44%. Meaning that from new customers obtained in february 5723, only 1455 purchased again our product in March. The 0 is always 100% retention because it is the new user collection month.

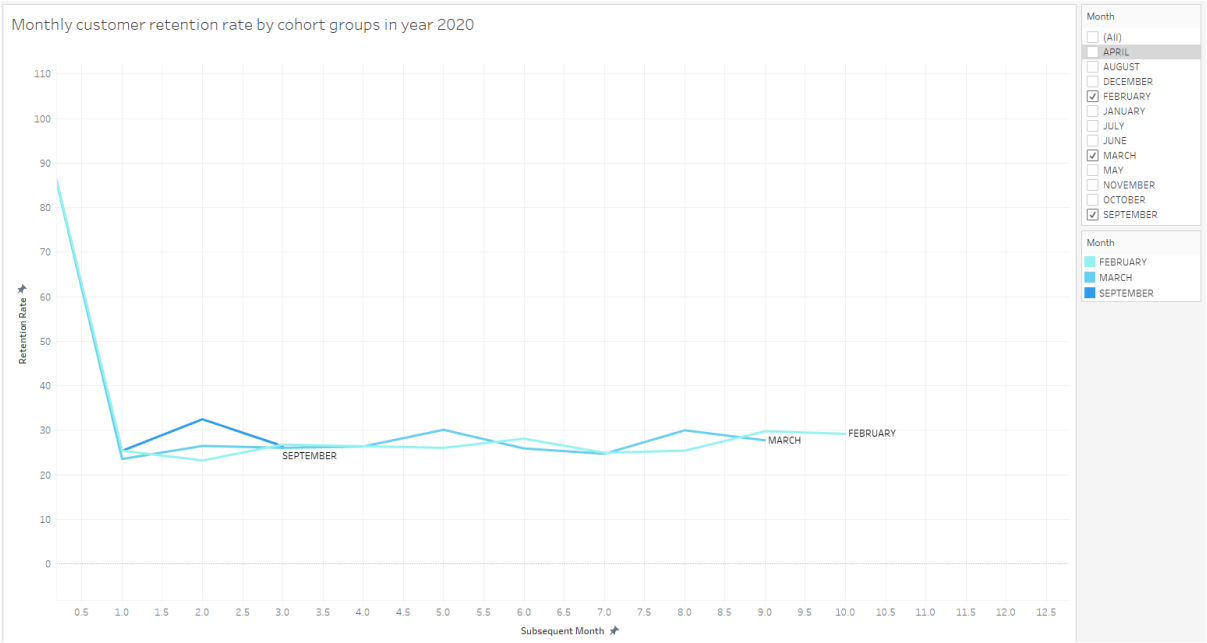


Tableau can be used to visualize the monthly customer retention by using the line chart. The right side panel is used to filter out the month we need to compare, in this case, customer retention of February, March and September is being visualized.

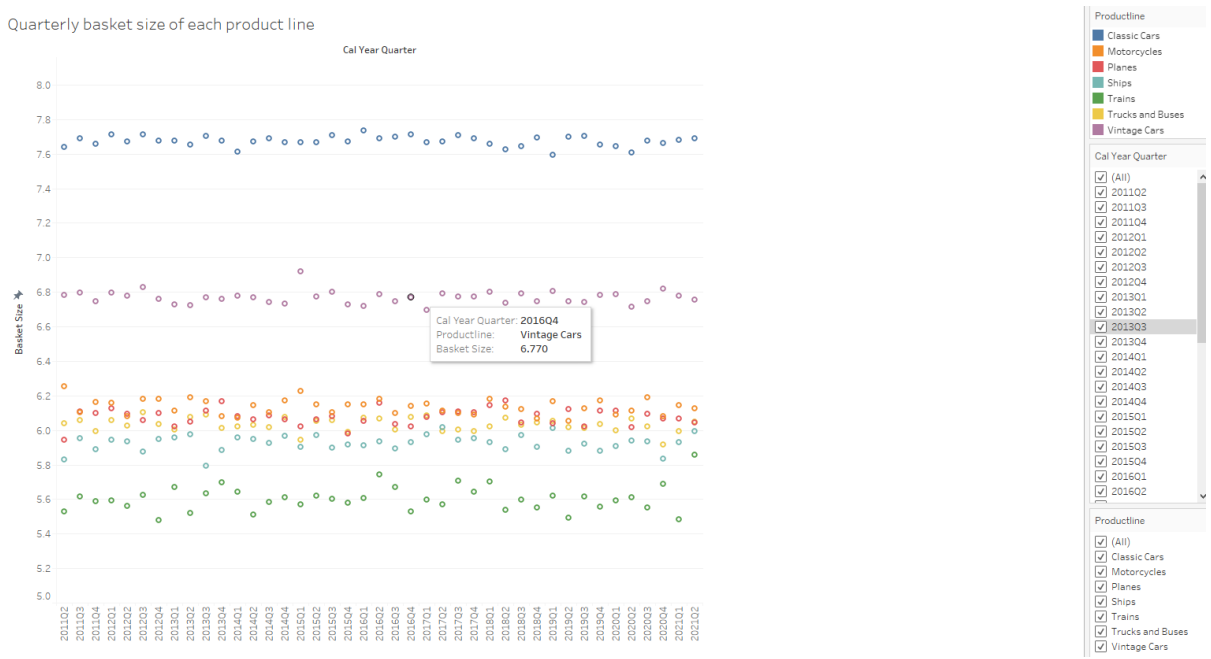
Basket Size - Quarterly Average Basket Size of each product line

The purpose of this query is to know the average basket size for each product line quarterly. It can help the company assess the marketing and sales efforts, and helps understand customers better.It can help companies to know whether it is the right product always selling to the consumers. The product line is also ranked and displayed horizontally based on basket size, which is shown on Best product, 2nd, 3nd and worst product for user to easily spot the best and worst product at a glance in a year quarter.

Quarterly average basket size of each product line from 2019 to 2021

CAL_YE	Motorcycles	Classic Cars	Trucks and Buses	Vintage Cars	Planes	Ships	Trains	Best Product	2nd	3nd	Worst Product
2019Q1	6.16	7.59	6.05	6.80	6.03	6.01	5.62	Classic Cars	Vintage Cars	Motorcycles	Trains
2019Q2	6.05	7.69	6.01	6.74	6.12	5.88	5.49	Classic Cars	Vintage Cars	Planes	Trains
2019Q3	6.12	7.70	6.01	6.74	6.02	5.92	5.61	Classic Cars	Vintage Cars	Motorcycles	Trains
2019Q4	6.17	7.65	6.03	6.78	6.11	5.88	5.55	Classic Cars	Vintage Cars	Motorcycles	Trains
2020Q1	6.08	7.64	6.00	6.78	6.11	5.90	5.59	Classic Cars	Vintage Cars	Planes	Trains
2020Q2	6.11	7.60	6.06	6.71	6.01	5.93	5.61	Classic Cars	Vintage Cars	Motorcycles	Trains
2020Q3	6.19	7.67	6.01	6.74	6.09	5.93	5.55	Classic Cars	Vintage Cars	Motorcycles	Trains
2020Q4	6.08	7.66	5.91	6.81	6.06	5.83	5.69	Classic Cars	Vintage Cars	Motorcycles	Trains
2021Q1	6.14	7.68	5.99	6.77	6.06	5.93	5.48	Classic Cars	Vintage Cars	Motorcycles	Trains
2021Q2	6.12	7.69	6.04	6.75	6.04	5.99	5.85	Classic Cars	Vintage Cars	Motorcycles	Trains

To interpret the report, let say in 2019 Q4, the Motorcycles have average basket size of 6.17 which means averagely, each transaction made in this product line consist of 6 products. And in 2019, q4, Motorcycles is the third product line which has the third highest average basket size compared to its peers and the worst product line in terms of basket size is Trains.



The circular graph can be used to visualize the changing of basket size for each product line quarter over quarter. From the graph, it is shown that the blue color one, which is classic cars, always have the top basket size, a super performer compared to others and also the vintage cars. The worst product line, the green one which is always at the bottom. The yellow ,red and orange product line keeps competing with each other for the third best performer.

YOY- Year over year revenue

This query produces a report to show the year over year growth on revenue of each product line on a yearly basis where the range of years is specified by user. It is helpful for business management teams to know which business is growing and which business encounters stunted growth.The YoY is very efficient to review because each column actually involves information that comes from two year, for example, in column 2017, it consists of revenue from 2016 and 2017.It can facilitate the cross comparison of the revenue for each product line quickly. The rate used in this report is percentage. The dynamic query is applied so that it can show the result of the user input year range.

```
set linesize 200
set pagesize 20

accept v_startY char prompt 'Enter the starting year (yyyy) :'
accept v_endY char prompt 'Enter the ending year (yyyy) :'

-- get revenue by product line && year
create or replace view yearlyProductLineRevenue as
    with LineRevenue as(
        select
            productLine,
            cal_year,
            productCode,
            priceEach,
            sum(quantityOrdered) as TotalOrder,
            sum(quantityOrdered) * priceEach as Revenue
        from
            dim_products
        join
            sales_fact using(product_key)
        join
            dim_date using(date_key)
        group by
            productLine,
            cal_year,
            productCode,
            priceEach
        order by
            productCode
    ),
    LookupRevenue as(
        select
            productLine,
            cal_year,
            sum(Revenue) as TotalRevenue,
            lag(sum(Revenue))
            over (
                partition by productLine
                order by productLine
            ) as PrevRevenue
```

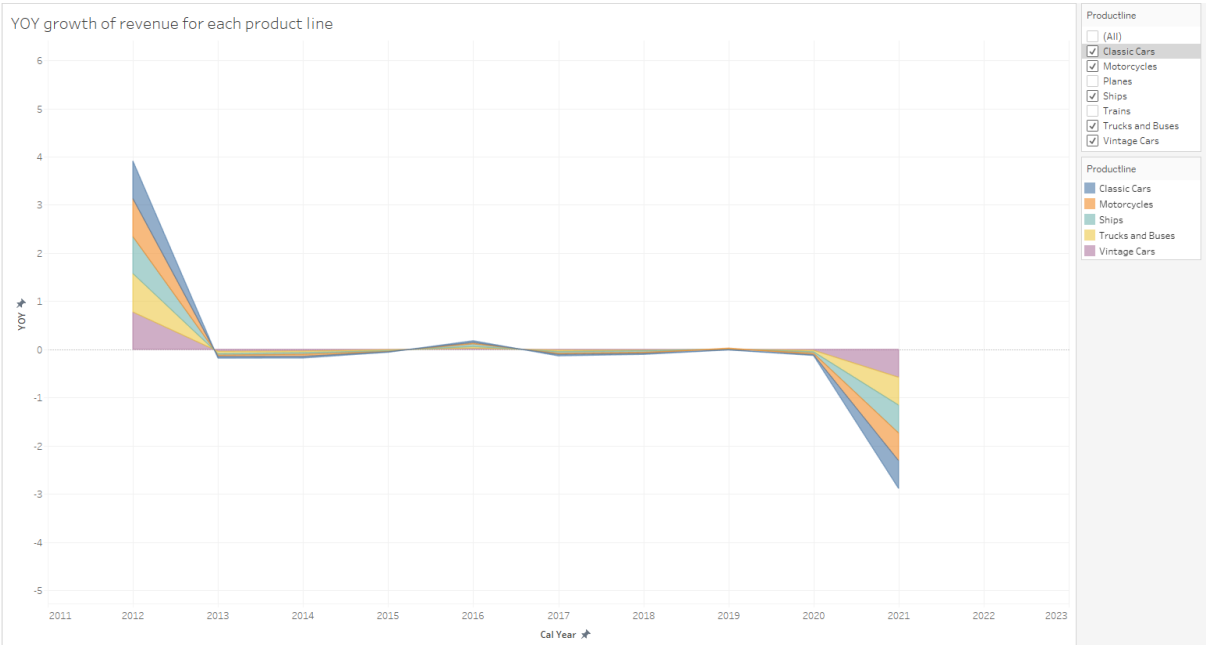
```
        from
            LineRevenue
        group by
            productLine,
            cal_year
        order by
            productLine,
            cal_year
    )
    select
        productLine,
        cal_year,
        TotalRevenue,
        PrevRevenue,
        (TotalRevenue - PrevRevenue) / PrevRevenue as YOY
    from
        LookupRevenue
    where
        PrevRevenue is not null;

-- select * from yearlyProductLineRevenue;

-- since cant specify the year, dynamic query is required
create or replace function generate_yoy_query_str(v_start number,
v_end number)
return varchar2
is
    sql_stmt1    varchar2(1000);
    sql_stmt2    varchar2(1000);
    sql_stmt3    varchar2(1000);
    final_sql   varchar2(1000);
    year_str    varchar2(1000);
    startY      number;
    endY         number;
begin
    startY := v_start;
    endY   := v_end;

    sql_stmt1 := '
    create or replace view yoy_view as
    select *
    from
    (
        select
            productLine,
            cal_year,
            yoy
        from
            yearlyProductLineRevenue
    )';

    sql_stmt2 := '
    )
    order by
        productLine';
```

```
    while startY <= endY
    loop
        year_str := year_str || startY || ',';
        startY := startY+1;
    end loop;

    year_str  := substr(year_str,1, length(year_str)-1);
    sql_stmt3 := '
    pivot
    (
        sum(trunc(yoy * 100,2)) for cal_year in (' || year_str ||
')';

    final_sql := sql_stmt1 || sql_stmt3 || sql_stmt2;

    return final_sql;
end;
/



declare
begin
    -- dbms_output.put_line(generate_yoy_query_str(&v_startY,
&v_endY));
    execute immediate generate_yoy_query_str(&v_startY, &v_endY);
end;
/

column productLine format a20 heading 'Product Line';

ttitle col 10 'YOY revenue growth of each product line from
&v_startY to &v_endY' skip 1-
       col 10
========================================================== skip 2

select * from yoy_view;
```

YOY revenue growth of each product line from 2017 to 2021

Product Line	2017	2018	2019	2020	2021
Classic Cars	-3.13	-1.96	-.26	-1.69	-57.77
Motorcycles	-2.39	-2.45	1.25	-3.11	-57.45
Planes	-2.19	-2.59	-.58	-.38	-58.21
Ships	-3.32	-3.14	.41	-2.95	-57.86
Trains	-3.65	-3.88	2.09	-1.52	-58.31
Trucks and Buses	-2.98	-1.08	.97	-3.11	-57.79
Vintage Cars	-1.66	-1.76	-.85	-1.85	-57.91

To interpret the table, let's say the Classic Cars in 2017, the YoY revenue growth is -3.31%, which means the revenue generated in 2016 is more than 2017, the Classic Cars have negative growth in revenue in 2017. In 2019, the Motorcycles revenue have growth about 1.25% in revenue.



The area chart can be used to visualize the revenue growth for each product line. Above the 0 line beans the growth is positive while below the line 0 means the product line has a negative growth in that year.