

Advance Data Analysis

Customer_Table:

The screenshot shows a data schema editor interface for a table named 'customers'. At the top, there's a breadcrumb navigation: portfolio-project-478309 / Datasets / DataAnalyst_Project / Tables / customers. Below the navigation are several actions: Query, Open in, Share, Copy, Snapshot, Delete, Export, and Refresh. A filter bar allows entering a property name or value. The main area displays a table schema with columns: Field name, Type, Mode, Description, Key, Collation, Default value, Policy tags, and Data policies. The table contains ten rows corresponding to the fields: customer_key (INTEGER, NULLABLE), customer_id (INTEGER, NULLABLE), customer_number (STRING, NULLABLE), first_name (STRING, NULLABLE), last_name (STRING, NULLABLE), country (STRING, NULLABLE), marital_status (STRING, NULLABLE), gender (STRING, NULLABLE), birthdate (DATE, NULLABLE), and create_date (DATE, NULLABLE). At the bottom, there are buttons for Edit schema, View row access policies, and Describe data.

Field name	Type	Mode	Description	Key	Collation	Default value	Policy tags	Data policies
customer_key	INTEGER	NULLABLE	-	-	-	-	-	-
customer_id	INTEGER	NULLABLE	-	-	-	-	-	-
customer_number	STRING	NULLABLE	-	-	-	-	-	-
first_name	STRING	NULLABLE	-	-	-	-	-	-
last_name	STRING	NULLABLE	-	-	-	-	-	-
country	STRING	NULLABLE	-	-	-	-	-	-
marital_status	STRING	NULLABLE	-	-	-	-	-	-
gender	STRING	NULLABLE	-	-	-	-	-	-
birthdate	DATE	NULLABLE	-	-	-	-	-	-
create_date	DATE	NULLABLE	-	-	-	-	-	-

Datasets:

Customer Table: [gold.dim_customers.xlsx](#)

Advance Data Analysis

Product_Table:

The screenshot shows a data schema editor interface for a table named 'products'. The top navigation bar includes links for 'portfolioproject-478309 / Datasets / DataAnalyst_Project / Tables / products', 'Query', 'Open in', 'Share', 'Copy', 'Snapshot', 'Delete', 'Export', and 'Refresh'. Below the navigation is a tab bar with 'Schema' (selected), 'Details', 'Preview', 'Table explorer', 'Preview' (disabled), 'Insights', 'Lineage', 'Data profile', and 'Data Quality'. A search bar labeled 'Filter' with the placeholder 'Enter property name or value' is present. The main area displays a table schema with 11 columns: 'Field name', 'Type', 'Mode', 'Description', 'Key', 'Collation', 'Default value', 'Policy tags', and 'Data policies'. The columns for 'product_key', 'product_id', 'product_number', 'product_name', 'category_id', 'category', 'subcategory', 'maintenance', 'cost', 'product_line', and 'start_date' are listed. Each row has a checkbox in the first column. At the bottom are buttons for 'Edit schema' (highlighted in blue), 'View row access policies', and 'Describe data'.

	Field name	Type	Mode	Description	Key	Collation	Default value	Policy tags	Data policies
<input type="checkbox"/>	product_key	INTEGER	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	product_id	INTEGER	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	product_number	STRING	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	product_name	STRING	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	category_id	STRING	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	category	STRING	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	subcategory	STRING	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	maintenance	BOOLEAN	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	cost	INTEGER	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	product_line	STRING	NULLABLE	-	-	-	-	-	-
<input type="checkbox"/>	start_date	DATE	NULLABLE	-	-	-	-	-	-

[Edit schema](#) [View row access policies](#) [Describe data](#)

Datasets:

Product Table: [gold.dim_products.xlsx](#)

Advance Data Analysis

Sales_Table:

The screenshot shows the schema editor for a table named 'sales' in a dataset called 'DataAnalyst_Project'. The top navigation bar includes options like 'Query', 'Open in', 'Share', 'Copy', 'Snapshot', 'Delete', 'Export', and 'Refresh'. Below the navigation is a tab bar with 'Schema' selected, followed by 'Details', 'Preview', 'Table explorer', 'Preview' (which is currently active), 'Insights', 'Lineage', 'Data profile', and 'Data Quality'. A search bar labeled 'Filter' with the placeholder 'Enter property name or value' is positioned above the table. The main area displays a table with 10 columns: 'Field name', 'Type', 'Mode', 'Description', 'Key', 'Collation', 'Default value', 'Policy tags', and 'Data policies'. Each row represents a column in the table, with the first column containing a checkbox. The data for each column is as follows:

Field name	Type	Mode	Description	Key	Collation	Default value	Policy tags	Data policies
order_number	STRING	NULLABLE	-	-	-	-	-	-
product_key	INTEGER	NULLABLE	-	-	-	-	-	-
customer_key	INTEGER	NULLABLE	-	-	-	-	-	-
order_date	DATE	NULLABLE	-	-	-	-	-	-
shipping_date	DATE	NULLABLE	-	-	-	-	-	-
due_date	DATE	NULLABLE	-	-	-	-	-	-
sales_amount	INTEGER	NULLABLE	-	-	-	-	-	-
quantity	INTEGER	NULLABLE	-	-	-	-	-	-
price	INTEGER	NULLABLE	-	-	-	-	-	-

At the bottom, there are three buttons: 'Edit schema' (blue background), 'View row access policies', and 'Describe data'.

Datasets:

Sales Table: [gold.fact_sales.xlsx](#)

Advance Data Analysis

/* Changes over time Analysis

* Analyze how a measure evolves over time helps track trends and identify seasonality in your data.

* Changes over years: A high-level overview insights that helps with strategic decision-making.

* Changes over months: Detailed insight to discover seasonality in your data.

*/

select

```
extract(year from order_date)as year,  
extract(month from order_date)as month,  
sum(sales_amount)as total_sales,  
count(customer_key)as total_customer,  
sum(quantity)as total_quantity  
from `DataAnalyst_Project.sales`  
where order_date is not null  
group by extract(year from order_date),extract(month from order_date)  
order by year;
```

/*

* Aggregate the data progressively over time helps to understand whether our business is growing or declining.

* Calculate the total sales per month and the running total of sales over time.

*/

--changes OVER BY year OR month

SELECT

```
EXTRACT(month FROM order_date)AS month,  
SUM(sales_amount)AS total_sales,  
COUNT(DISTINCT customer_key)AS total_customer,  
SUM(quantity)AS total_quantity  
FROM `DataAnalyst_Project.sales`  
WHERE order_date IS NOT NULL  
GROUP BY EXTRACT(month FROM order_date)  
ORDER BY month;
```

Advance Data Analysis

--calculate the total sales per month AND

--the running total OF sales OVER time

select

```
months, total_sales,  
sum(total_sales) over (order by months)as running_total  
from (  
SELECT date_trunc(order_date, month)as months,  
sum(sales_amount)as total_sales  
from `DataAnatlyst_Project.sales`  
where order_date is not null  
group by date_trunc(order_date, month)  
order by months )sub;
```

/*

* Performance Analysis

* Comparing the current value to a target value.

* Helps measure success and compare performance.

* Analyze the yearly performance of products by comparing each products sales

to both it's average sales performance and the previous year's sales.

*/

with yearly_product_sales as(

select

```
extract(year from s.order_date)as order_year,  
p.product_name,  
sum(sales_amount)as current_sales  
from `DataAnatlyst_Project.sales` s  
left join `DataAnatlyst_Project.products` p  
on s.product_key = p.product_key  
where order_date is not null  
group by extract(year from s.order_date),
```

Advance Data Analysis

```
p.product_name  
)  
  
select order_year, product_name, current_sales,  
avg(current_sales) over (partition by product_name ) as avg_sales,  
(current_sales - avg(current_sales) over (partition by product_name )) as diff_avg,  
case  
when (current_sales - avg(current_sales) over (partition by product_name )) > 0 then 'Above the avg'  
when (current_sales - avg(current_sales) over (partition by product_name )) < 0 then 'Below the avg'  
else "avg"  
end as level_of_Avg,  
lag(current_sales) over(partition by product_name order by order_year) as previous_year_sales,  
(current_sales - lag(current_sales) over(partition by product_name order by order_year))as  
diff_previous_year_sales,  
case  
when (current_sales - lag(current_sales) over(partition by product_name order by order_year)) > 0  
then 'INCREASES'  
when (current_sales - lag(current_sales) over(partition by product_name order by order_year)) < 0  
then 'DECREASES'  
else "NO CHANGES"  
end as Level_of_previous_years_sales  
from yearly_product_sales  
order by product_name, order_year;
```

Advance Data Analysis

```
/*
```

* Part-to-whole Analysis

```
* Analyze how an individual part is performing compared to the overall,  
allowing us to understand which category has the greatest impact on the business.
```

```
* which categories contribute the most to overall sales
```

```
*/
```

```
with categories_sales as(
```

```
select
```

```
p.category,
```

```
sum(s.sales_amount)as total_sales
```

```
from `DataAnatlyst_Project.sales` s
```

```
left join `DataAnatlyst_Project.products` p
```

```
on s.product_key = p.product_key
```

```
group by p.category
```

```
)
```

```
select category, total_sales,
```

```
sum(total_sales) over() as overall_sales,
```

```
concat(round((total_sales /sum(total_sales) over()*100),2),'%') as percentage_of_total
```

```
from categories_sales
```

```
group by category, total_sales;
```

Advance Data Analysis

```
/*
 * Data Segmentation
 * Group the data based on a specific range
 * Helps understand the correlation between two measures.
 * Segment product into cost range and count how many products fall into each segment.
 */

with product_segment as (
    select
        cost,product_name, product_key,
        case
            when cost < 100 then "Below 100"
            when cost between 100 and 500 then "100 - 500"
            when cost between 500 and 1000 then "500 - 1000"
            else "above 1000"
        end as cost_range
    from `DataAnalyst_Project.products`
)
select cost_range, count(product_key)as total_product
from product_segment
group by cost_range
order by total_product desc;
```

Advance Data Analysis

```
/*
 * Group customers into three segments based on their spending behavior.
 * Vip: at least 12 months of history and spending more than 5000.
 * Regular: at least 12 months of history but spending 5000 or less
 * New: lifespan less than 12 months
 * And find the total number of customers by each group.
 */

with customer_spending as (
    select
        c.customer_key,
        sum(s.sales_amount) as total_sales,
        min(s.order_date) as first_date,
        max(s.order_date) as last_date,
        date_diff(max(s.order_date), min(s.order_date), month) as lifespan
    from `DataAnatlyst_Project.sales` s
    left join `DataAnatlyst_Project.customers` c
        on s.customer_key = c.customer_key
    group by c.customer_key
)
select count(customer_key) as total_customer, customer_Segment
from(
    select customer_key, total_sales, lifespan,
    case
        when lifespan >= 12 and total_sales > 5000 then " VIP "
        when lifespan >= 12 and total_sales <= 5000 then "Regular"
        else "New"
    end as customer_Segment
    from customer_spending
)sub
group by customer_segment
order by total_customer desc;
```

Advance Data Analysis

/*

* Build Customer Report

Purpose:

-This report consolidates key customer metrics and behaviors

Highlights:

* Gathers essential fields such as names, age, and transaction details.

* Segments customer into categories (Vip, Regular, New) and age groups.

* Aggregate customer-Level metrics:

-Total orders

-Total sales

-Total quantity purchased

-Total products

-Lifespan (in months)

* calculates valuable KPI's:

-Recency (months since last orders)

-Average order value (Avg order value = Total_sales / Total_orders)

-average monthly spend (Avg monthly _spend = Total_Sales / Total_months)

*/

```
create view `DataAnalyst_Project.report_customer` as
```

```
with Base_Query as (
```

```
--Base_Query: Retrieves core columns for Tables
```

```
select
```

```
s.order_number,
```

```
s.order_date,
```

```
s.sales_amount,
```

```
s.product_key,
```

```
s.quantity,
```

```
c.customer_key,
```

```
c.customer_number,
```

```
c.birthdate,
```

Advance Data Analysis

```
concat(c.first_name,'',c.last_name)as Full_name,  
date_diff(current_date(),c.birthdate,year)as age  
from `DataAnatlyst_Project.sales` s  
left join `DataAnatlyst_Project.customers` c  
on s.customer_key = c.customer_key  
where s.order_date is not null  
)  
, customer_aggregation as(  
-- Customer_aggregation: Summarizes key metrics at the customer level.  
select  
customer_key,  
customer_number,  
Full_name,  
age,  
count(distinct order_number)as Total_orders,  
sum(sales_amount)as Total_sales,  
sum(quantity)as Total_quantity,  
count(distinct product_key)as Total_products,  
max(order_date)as last_order_date,  
date_diff(max(order_date),min(order_date),month)as lifespan  
from Base_Query  
group by customer_key,  
customer_number,  
Full_name,  
age  
)
```

Advance Data Analysis

```
Select customer_key, customer_number, Full_name, age,  
case  
    when age <20 then "Under 20"  
    when age between 20 and 29 then "20 - 29"  
    when age between 30 and 39 then "30 - 39"  
    when age between 40 and 49 then "40 - 49"  
    else "above 50"  
end as age_group,  
case  
    when lifespan >= 12 and total_sales >5000 then " VIP "  
    when lifespan >=12 and total_sales <=5000 then "Regular"  
    else "New"  
end as customer_Segment ,  
date_diff(current_date(), last_order_date, month)as Recency,  
Total_orders,  
Total_sales,  
Total_quantity,  
Total_products,  
lifespan,  
--compute avg order values (av0)  
case  
    when total_sales = 0 then 0  
    else total_sales/total_orders  
end as avg_order_values,  
--compute avg monthly spends  
case  
    when lifespan = 0 then total_sales  
    else lifespan / total_sales  
end as avg_monthly_spents  
from customer_aggregation;
```

Advance Data Analysis

```
/*
```

* Build product report

Purpose:

- This report consolidates key product metrics and behaviors.

Highlight:

- * Gather essential fields such as product Name, category, subcategory, and cost.

- * Segementas products by revenue to identify high-performers, mid-range, or Low-Performers.

- * Aggregates Product-Level metrics:

 - Total orders

 - Total sales

 - Total quantity sold

 - Total customers (Unique)

 - Lifespan(in months)

- * Calculate valuable KPI's:

 - Recency(month since last sales)

 - Average order(Avg order)

 - Average monthly revenue.

```
*/
```

```
create view `DataAnalyst_Project.report_product` as
```

```
with Base_Query as (
```

```
--Base_Query: Retrieves core columns for Tables
```

```
select
```

```
s.order_number,
```

```
s.order_date,
```

```
s.customer_key,
```

```
s.sales_amount,
```

```
s.quantity,
```

```
p.product_key,
```

```
p.product_name,
```

```
p.category,
```

```
p.subcategory,
```

Advance Data Analysis

```
p.cost  
from `DataAnatlyst_Project.sales` s  
left join `DataAnatlyst_Project.products` p  
on s.product_key = p.product_key  
where s.order_date is not null  
)  
--Product Aggregations: Summarizes key metrics at the product level  
,Product_aggregation as (  
select  
product_key,  
product_name,  
category,  
subcategory,  
cost,  
count(distinct order_number)as Total_orders,  
count(distinct customer_key)as Total_customers,  
sum(quantity)as Total_quantity,  
sum(sales_amount)as Total_sales,  
max(order_date)as last_sales_date,  
round(avg(sales_amount/nullif(quantity,0)),2)as avg_selling_price,  
date_diff(max(order_date),min(order_date),month)as lifespan  
from Base_Query  
group by product_key,  
product_name,  
category,  
subcategory,  
cost)  
--The final Query: combines all product results into one output.  
select  
product_key,  
product_name,
```

Advance Data Analysis

```
category,  
subcategory,  
cost,  
last_sales_date,  
date_diff(current_date(),last_sales_date,month)as Recency_in_month,  
case  
when Total_sales >50000 then " High_Performance"  
when Total_sales >=10000 then " mid-Range"  
else "low-performance"  
end as Product_segment,  
Total_orders,  
Total_customers,  
Total_sales,  
Total_quantity,  
lifespan,  
avg_selling_price,  
--avg order revenue  
case  
when Total_orders =0 then 0  
else round((Total_sales / Total_orders),2)  
end as avg_order_revenue,  
--avg monthly revenue  
case  
when lifespan=0 then Total_sales  
else round((Total_sales/lifespan),2)  
end as avg_monthly_revenue  
from Product_aggregation;
```

Advance Data Analysis

Build Customer Report:

● portfolio-project-478309 / Datasets / DataAnalyst_Project / Tables / report_customer

report_customer ★ Query Open in ▾ Share ▾ Copy Delete ↻ Refresh

Schema	Details	Table explorer	Preview	Insights	Lineage	Data profile	Data Quality	
<input type="checkbox"/>	Field name	Type	Mode	Description	Key	Collation	Default value	Policy tags ⓘ
<input type="checkbox"/>	customer_key	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	customer_number	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Full_name	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	age	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	age_group	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	customer_Segment	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Recency	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_orders	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_sales	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_quantity	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_products	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	lifespan	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	avg_order_values	FLOAT	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	avg_monthly_spents	FLOAT	NULLABLE	-	-	-	-	-

[Edit schema](#)

Build Product Report:

● portfolio-project-478309 / Datasets / DataAnalyst_Project / Tables / report_product

report_product ★ Query Open in ▾ Share ▾ Copy Delete ↻ Refresh

Schema	Details	Table explorer	Preview	Insights	Lineage	Data profile	Data Quality	
<input type="checkbox"/>	product_key	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	product_name	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	category	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	subcategory	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	cost	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	last_sales_date	DATE	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Recency_in_month	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Product_segment	STRING	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_orders	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_customers	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_sales	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	Total_quantity	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	lifespan	INTEGER	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	avg_selling_price	FLOAT	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	avg_order_revenue	FLOAT	NULLABLE	-	-	-	-	-
<input type="checkbox"/>	avg_monthly_revenue	FLOAT	NULLABLE	-	-	-	-	-

[Edit schema](#) [Describe data](#)

Results:

Build Customer Report: [Build_Customer_report.xlsx](#)

Build Product Report: [Build_Product_Report.xlsx](#)