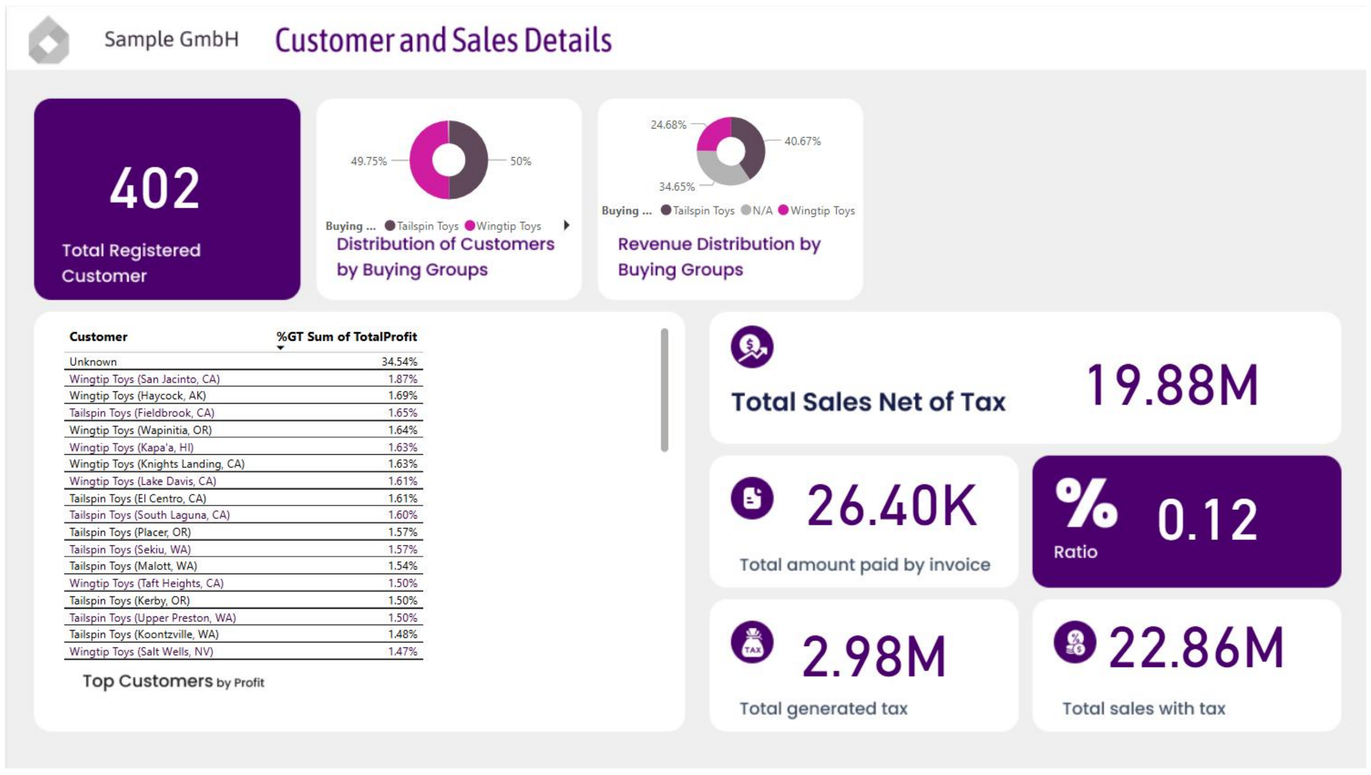
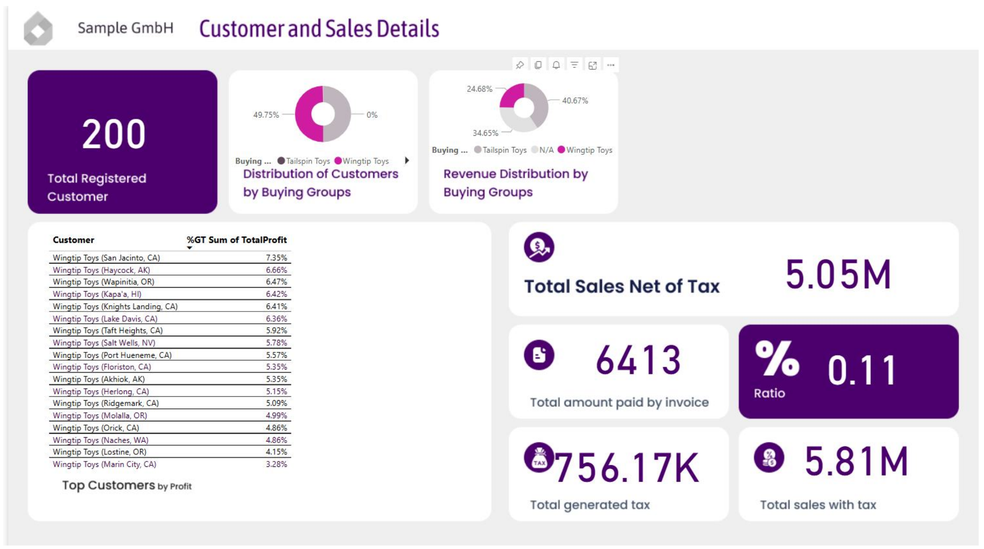
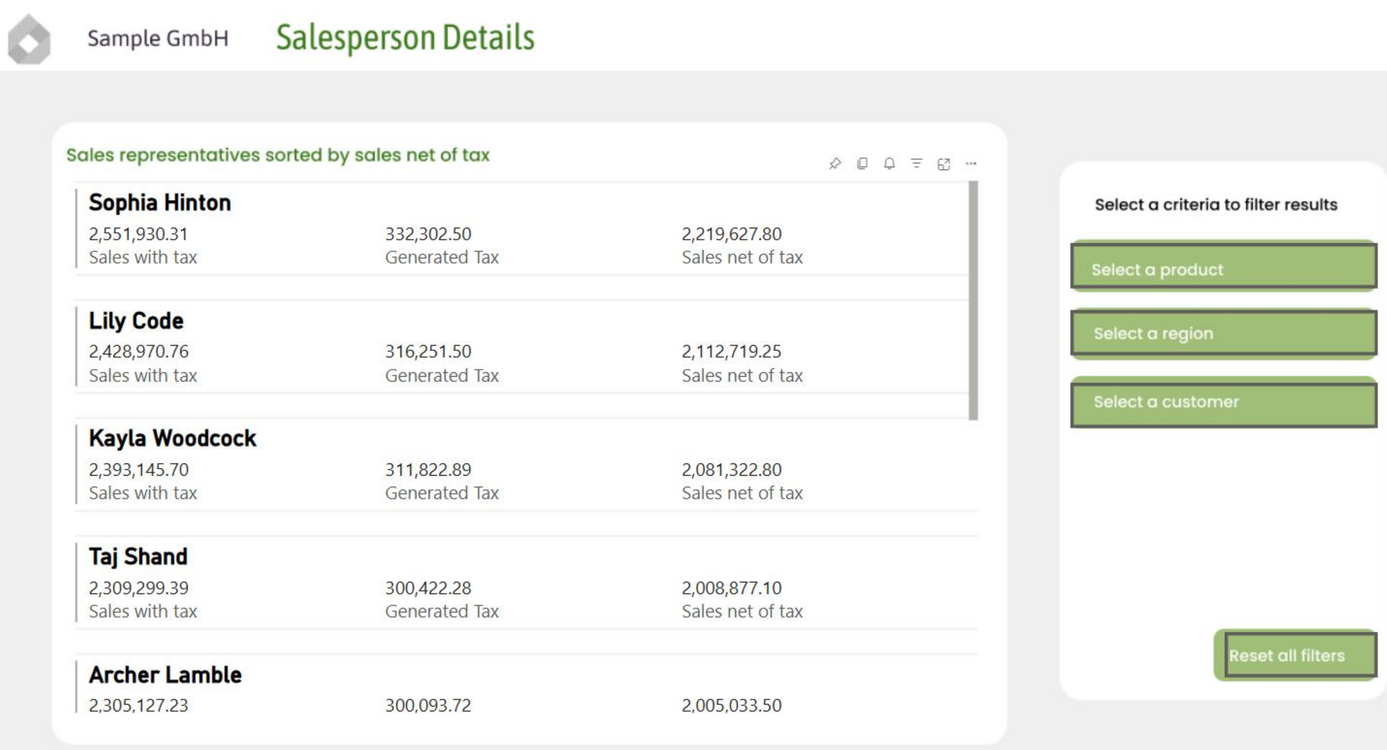
Portfolio:
SQL and PowerBI Learning Project

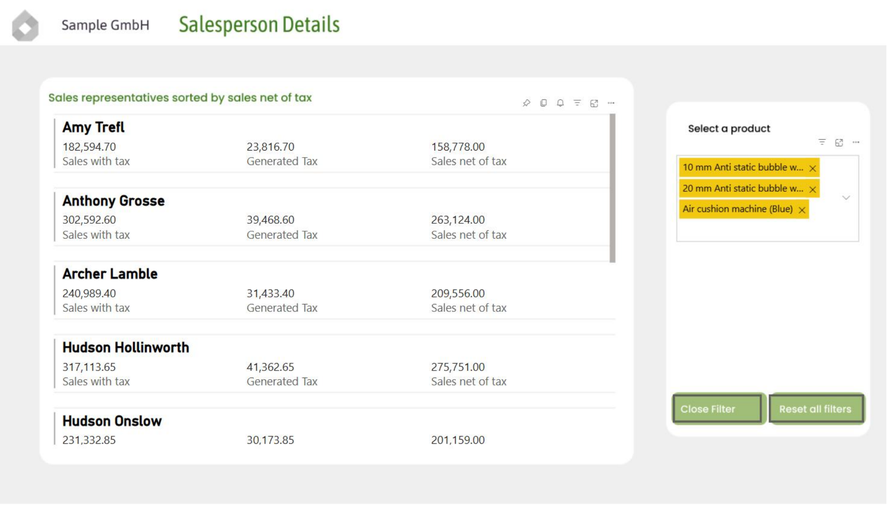
Kaan Bora Karapınar

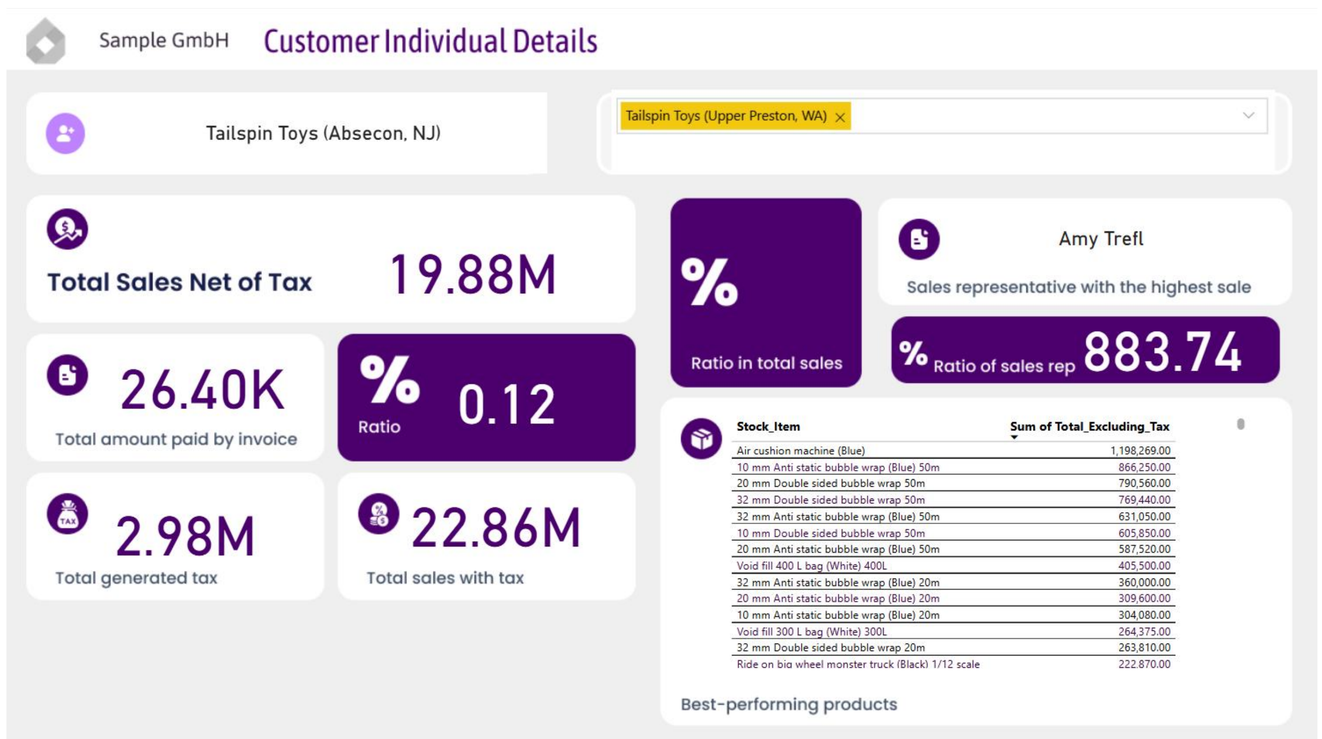
A look at the completed project

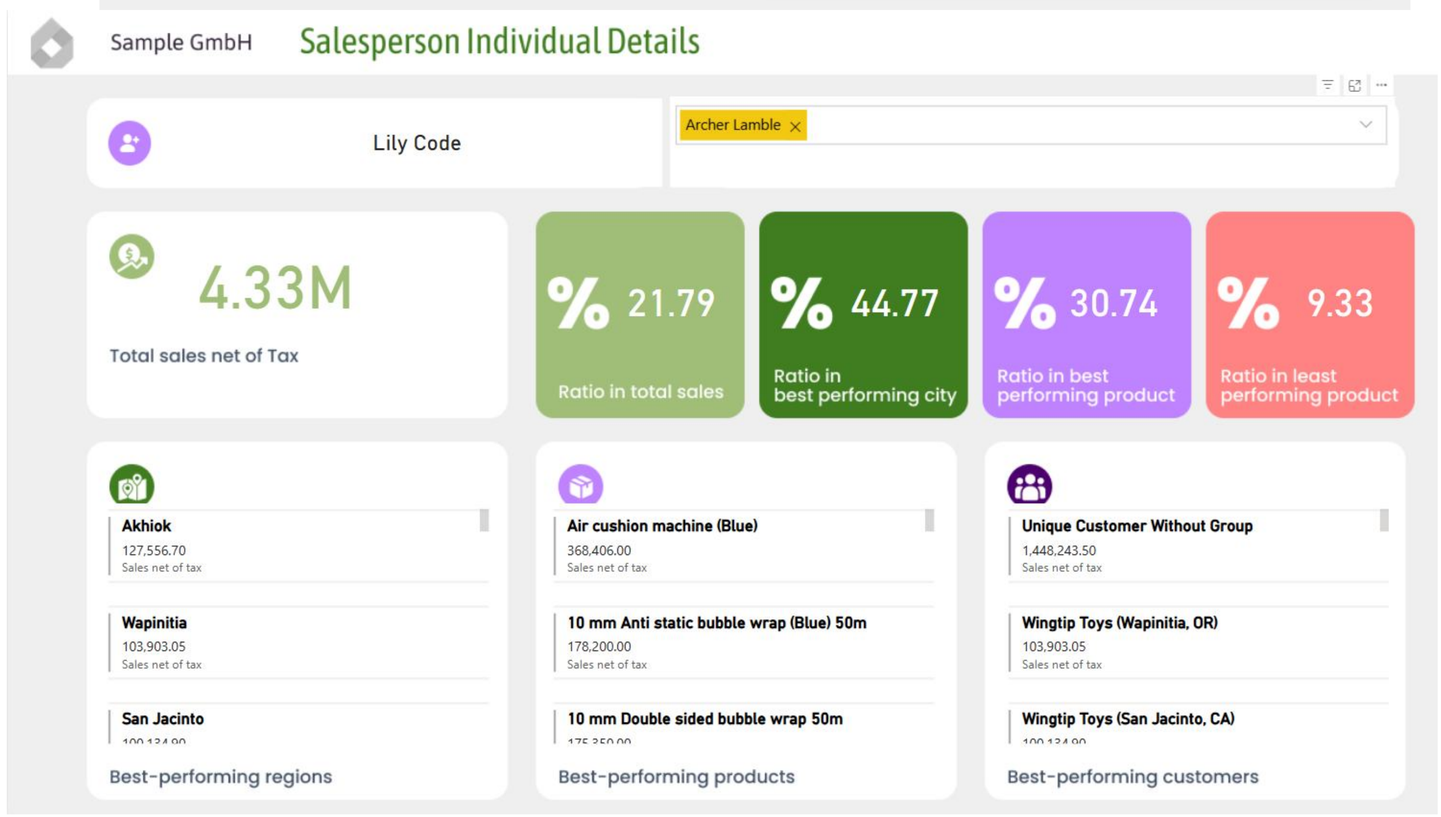








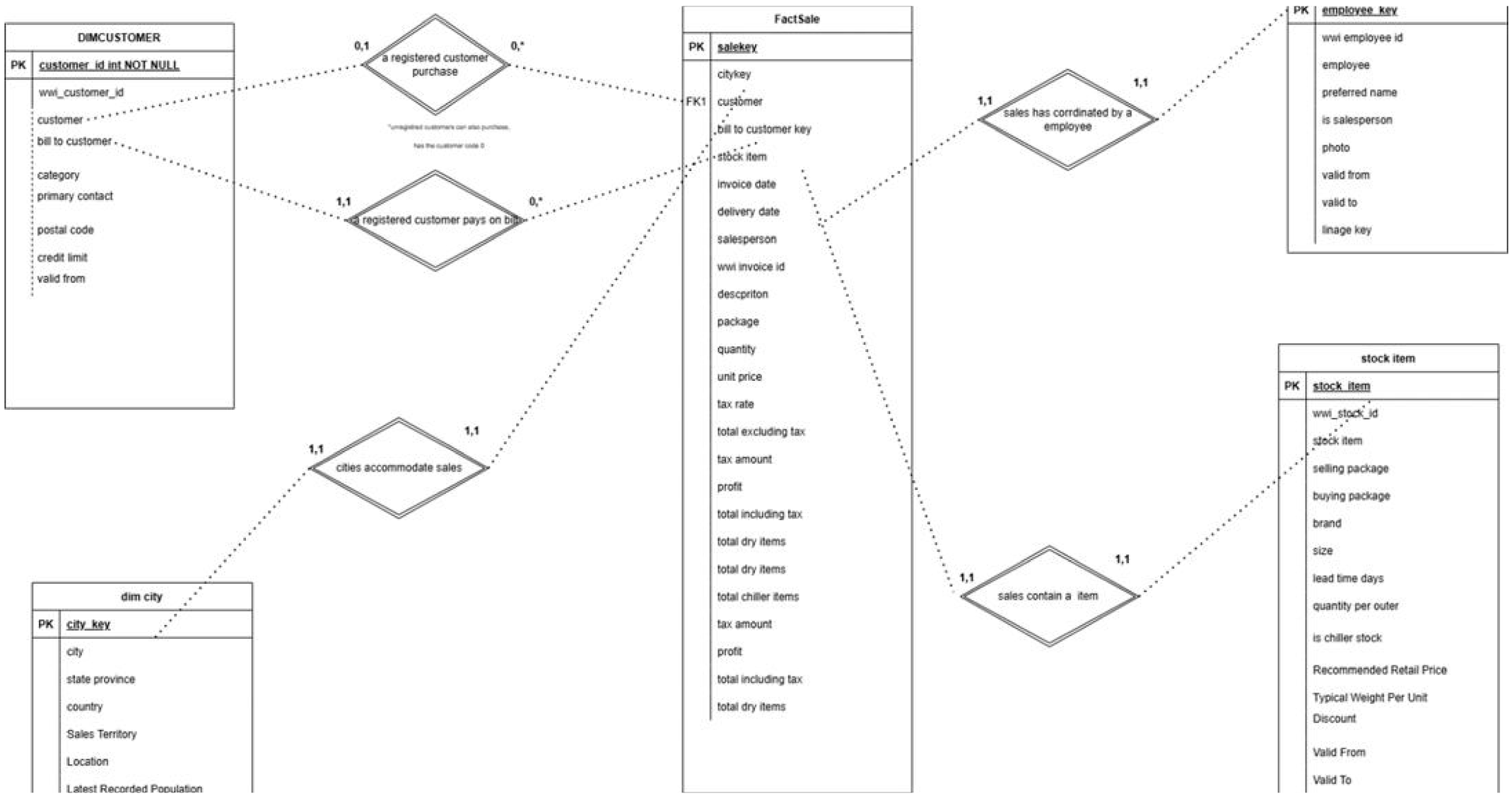


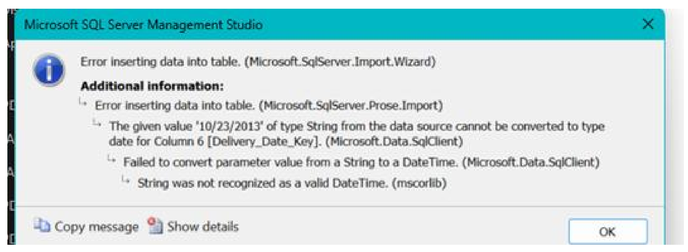


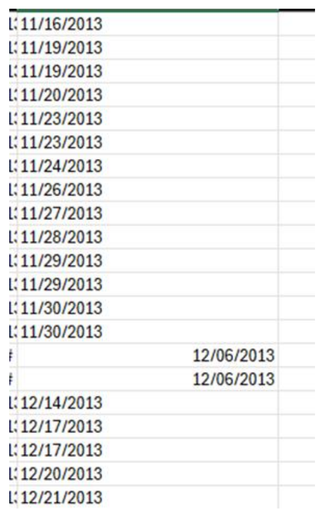
Getting Started and Gathering Insights

- Since this is a learning project, I took some steps that could have been considered unnecessary (PowerBI does most of the work automatically)

- I acquired a database containing a CSV and Excell files from a Datacamp course and **created an ERM with draw.io to get an understanding of the dataset before importing it into my local MSSQL database**. I used Excel to look through the data and define relationships accordingly.







- Troubleshooting: While checking the integrity of the data, I noticed that FactSale data was in American MM/DD/YYYY format, while MSSQL expected European DD/MM/YYYY format.

- I used a Python script to control whether American and European date formats are mixed.

File - C:\Users\borakarapinar\PycharmProjects\SQL demo project\datecheck.py

```
for line in reader:

        print("currently iterating Delivery Date Key")

        currentlist = line[6].split("/")

        #skipping first line with if
        if (currentlist[0] == "Delivery Date Key"):
            continue
        if (currentlist[0] == "NULL"):
            continue
        if (currentlist[1] == "NULL"):
            continue

        intvalfirstelement = int(currentlist[0])
        intvalsecondelement = int(currentlist[1])

        ismonth = intvalfirstelement <= 12 and intvalfirstelement > 0
        isday= intvalsecondelement <= 31 and intvalsecondelement > 0

        result_at_current_iteration = ismonth and isday
        if(not result_at_current_iteration):
            print("problem at: "+ line[5])
            delivery_date_key_pass_tests = False

        print("currently iterating Invoice Date Key")

        currentlist = line[5].split("/")

        # skipping first line with if
        if (currentlist[0] == "Delivery Date Key"):
            continue
        if (currentlist[0] == "NULL"):
            continue
        if (currentlist[1] == "NULL"):
            continue

        intvalfirstelement = int(currentlist[0])
        intvalsecondelement = int(currentlist[1])

        ismonth = intvalfirstelement <= 12 and intvalfirstelement > 0
        isday = intvalsecondelement <= 31 and intvalsecondelement > 0

        result_at_current_iteration = ismonth and isday
        if (not result_at_current_iteration):
            print("problem at: " + line[5])
            invoice_date_key_pass_tests = False

    if(delivery_date_key_pass_tests):
        print("success at delivery date")
    if (invoice_date_key_pass_tests):
        print("success at invoice date")
```



```
for line in reader:

    invoicenull = line[5] == "NULL"
    invoiceheader = line[5] == "Invoice Date Key"

    deliverynull = line[6] == "NULL"
    deliveryheader = line[6] == "Delivery Date Key"

    if(not invoicenull and not invoiceheader):
        currentlistforinvoice = line[5].split("/")
        temp1 = currentlistforinvoice[1]
        currentlistforinvoice[1] = currentlistforinvoice[0]
        currentlistforinvoice[0] = temp1

        line[5] = "/".join(currentlistforinvoice)

    if (not deliverynull and not deliveryheader):
        currentlistfordelivery = line[6].split("/")
        temp2 = currentlistfordelivery[1]
        currentlistfordelivery[1] = currentlistfordelivery[0]
        currentlistfordelivery[0] = temp2

        line[6] = "/".join(currentlistforinvoice)



    writer.writerow(line)
```

Defining the scope of the project: Questions

- Questions I considered after designing the ERM diagram:

Customer Analysis:

- Who are our top 5 customers based on total revenue?
  - For each top customer, which 10 salespeople contributed the most to their sales?

Salespeople:

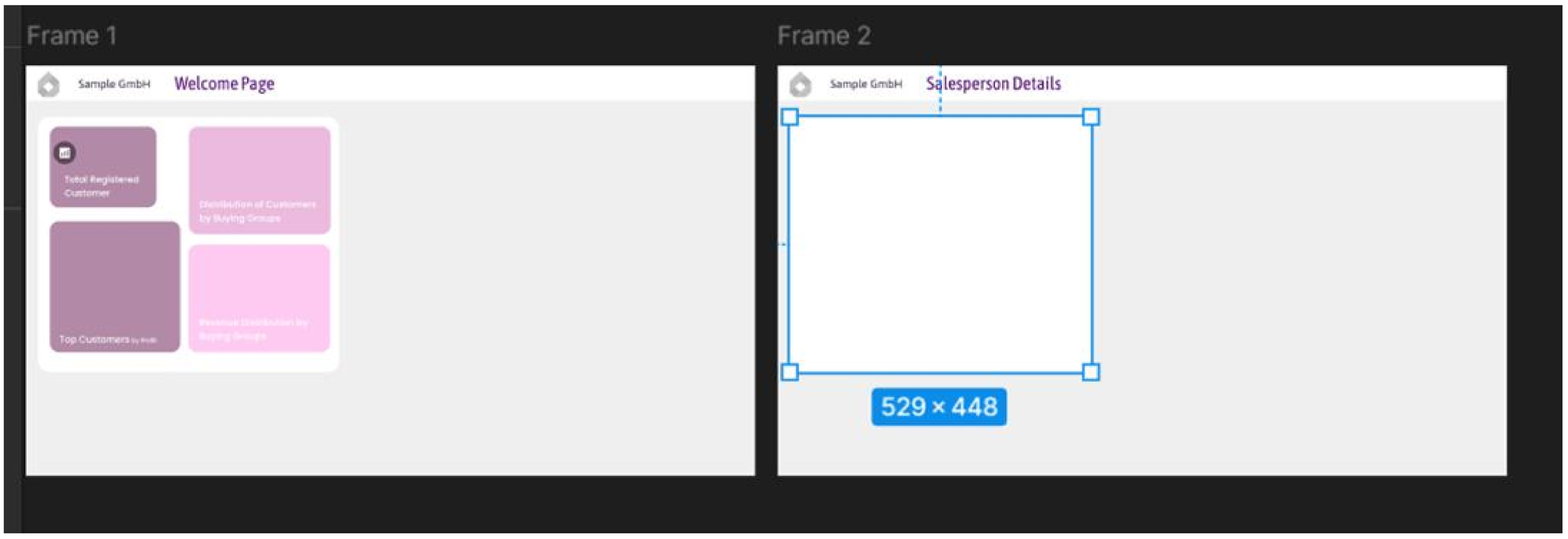
- Who are our top 5 salespeople by total revenue?
- For each top salesperson, what are their highest and lowest revenue-generating products?

Revenue:

- What percentage of total sales is paid via customer invoices compared to other payment methods?

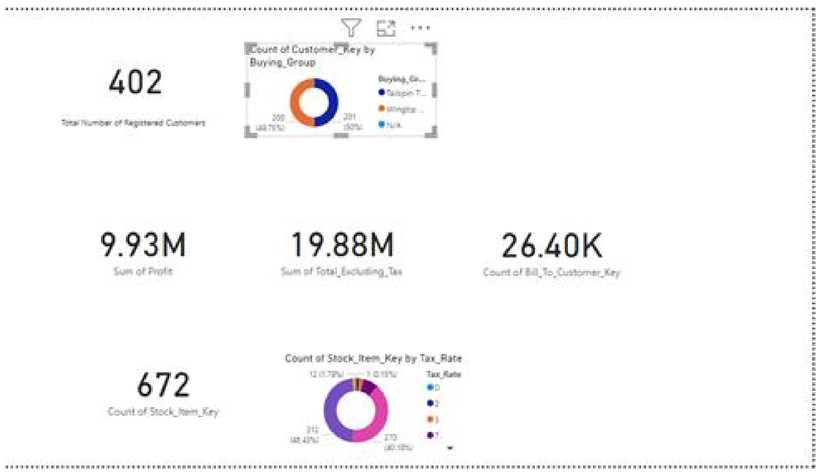
Creating a brand identity

- I have created a brand identity for my pseudo company, as there is none given, to use while creating report templates



Deepening insights with SQL

- I've started to create tables, visualisations and slicers in PowerBI, which I'll customise to my corporate design. Meanwhile, I've started writing SQL views and functions to understand my data and find out what's there to project.




```sql
CREATE FUNCTION dbo.getSalesOfCurrentKey
(
    @Current_Customer_Key INT
)
RETURNS TABLE
AS
RETURN
(
    select
        Salesperson_Key,
        SUM(Total_Excluding_Tax) AS TotalWithoutTax,
        SUM(Profit) AS TotalProfit
    from PowerBITestDatabase.dbo.FactSale as FC
    join DimEmployee as EE on FC.Salesperson_Key = EE.Employee_Key
    where FC.Customer_Key = @Current_Customer_Key
    group by Salesperson_Key

)
```

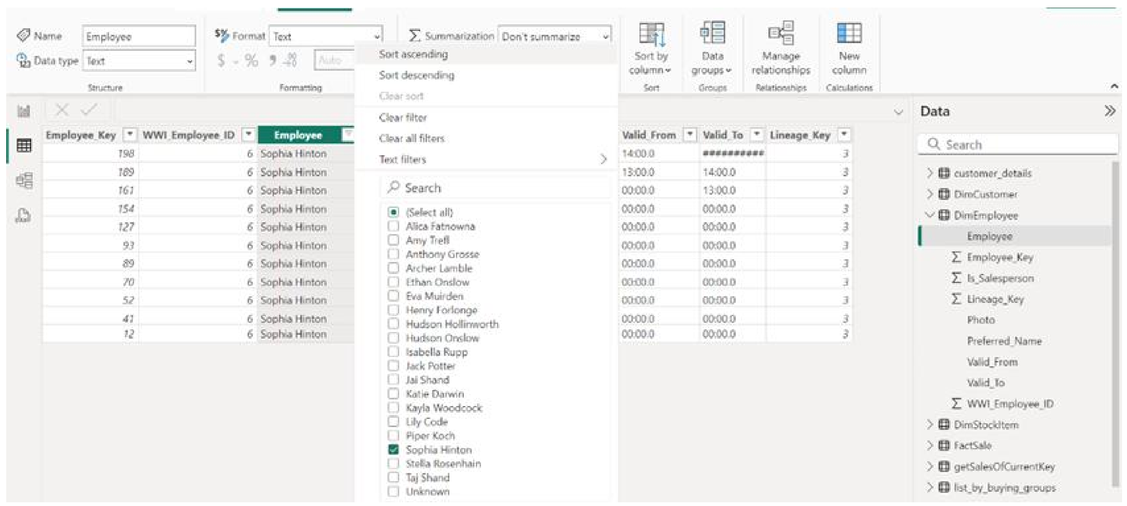
- My iterative approach (using MSSQL cursors) to determine the top 6 customers and their top 10 sales representatives along with the percentage of each customer's sales volume achieved by these representatives:

```sql
DECLARE @CurrentTotalWithoutTax INT
DECLARE @TotalPercentage DECIMAL(10, 2)

Declare Customer_Cursor CURSOR FOR
SELECT TOP 6 c.Customer_Key, C.TotalWithoutTax
from customer_details as c
order by c.TotalWithoutTax desc

OPEN Customer_Cursor
FETCH NEXT FROM Customer_Cursor INTO @CurrentSalesCustomerKey,@CurrentTotalWithoutTax

WHILE @@FETCH_STATUS = 0
BEGIN
    select TOP 10
        @CurrentSalesCustomerKey as Customer_Key,
        *,
        sales.TotalWithoutTax / @CurrentTotalWithoutTax *100 as percentage_in_sales

     INTO #TempSales
    from getSalesOfCurrentKey( @Current_Customer_Key @CurrentSalesCustomerKey) as sales
    order by TotalWithoutTax desc

    SELECT @TotalPercentage = SUM(percentage_in_sales)
    FROM #TempSales

    PRINT 'For Customer Key: ' + CAST(@CurrentSalesCustomerKey AS VARCHAR(20)) +
          ', first 10 salespersons make %: ' + CAST(@TotalPercentage AS VARCHAR(20))

    SELECT *
    FROM #TempSales

    DROP TABLE #TempSales

    FETCH NEXT FROM Customer_Cursor INTO @CurrentSalesCustomerKey,@CurrentTotalWithoutTax

END

CLOSE Customer_Cursor
```

```
For Customer Key: 0, first 10 salespersons make %: 40.33
For Customer Key: 273, first 10 salespersons make %: 51.74
[2024-07-27 21:03:16] [S0001][8153] Warning: Null value is eliminated by an aggregate or other SET operation.
For Customer Key: 102, first 10 salespersons make %: 64.05
[2024-07-27 21:03:16] [S0001][8153] Warning: Null value is eliminated by an aggregate or other SET operation.
For Customer Key: 281, first 10 salespersons make %: 56.79
For Customer Key: 398, first 10 salespersons make %: 61.12
[2024-07-27 21:03:16] [S0001][8153] Warning: Null value is eliminated by an aggregate or other SET operation.
For Customer Key: 384, first 10 salespersons make %: 58.77
[2024-07-27 21:03:16] 60 rows affected in 215 ms
```

	Customer_Key	Salesperson_Key	TotalWithoutTax	TotalProfit	percentage_in_sales
1	273	70	24981	10214.5	7.031098276633651
2	273	93	23390	11424.5	6.583298854748053
3	273	81	22852.800003051758	9482.199996948242	6.432099704483836
4	273	48	19643	10161	5.5286763319288585
5	273	97	18162	11373	5.111837272335791
6	273	134	16437	8985	4.626322500021109
7	273	19	15664.900001525879	8396.20000876294	4.4090089029409185
8	273	102	14528.5	7704	4.089160214245707
9	273	105	14520.400024414062	7295.299987792969	4.08688041262115
10	273	108	13656.799998792969	6373.649993896484	3.84381341253359

- Troubleshooting: During the design of the report page for sales representatives, **PowerBI displayed multiple keys for each representative.** Initially, I attempted to redefine the relationship between the sales and employee tables, but the employee table did, in fact, contain multiple keys for the same sales representative.



- PowerBI would still generate an accurate report for aggregated sales and the share of total sales volume for each sales representative. However, this functional dependency may eventually result in inconsistencies, beginning with the previous calculations concerning the top 10 representatives.

- I have updated the Employe and Sales tables using Python and Pandas Dataframes, which I have used as a temporary relational database. It would appear that there are only 19 unique employees. **(Please refer to the code on the following page.)**

```
select Employee, Employee_Key
from DimEmployee
where Employee IN (
    select distinct Employee
    from DimEmployee
    )
order by Employee asc
```

	Employee	Employee_Key
1	Alica Fatnowna	17
2	Alica Fatnowna	92
3	Alica Fatnowna	112
4	Alica Fatnowna	121
5	Alica Fatnowna	128
6	Alica Fatnowna	135
7	Alica Fatnowna	139
8	Alica Fatnowna	169
9	Alica Fatnowna	186
10	Alica Fatnowna	201
11	Amy Trefl	199
12	Amy Trefl	188
13	Amy Trefl	170
14	Amy Trefl	155
15	Amy Trefl	108
16	Amy Trefl	103
17	Amy Trefl	65
18	Amy Trefl	78
19	Amy Trefl	83
20	Amy Trefl	23
21	Amy Trefl	35

	Customer_Key	Salesperson_Key	Employee	percentage_in_sales	TotalProfit	TotalWithoutTax
1	273	108	Amy Trefl	3.84381341253336	6373.64999389648	13656.7999877929
2	273	105	Hudson Onslow	4.08688041262115	7295.29998779297	14520.4000244141
3	273	102	Jack Potter	4.0891602142457	7784	14528.5
4	273	19	Archer Lamble	4.40900892948918	8396.20000076294	15664.9800015259
5	273	134	Hudson Hollinworth	4.62632580021109	8985	16437
6	273	97	Taj Shand	5.11183727233579	11373	18162
7	273	48	Lily Code	5.52867633192885	10101	19445
8	273	81	Kayla Woodcock	6.43209970468383	9482.19999694824	22852.8000030518
9	273	93	Sophia Hinton	6.58329885474805	11424.5	23398
10	273	70	Sophia Hinton	7.03109827663651	10214.5	24981

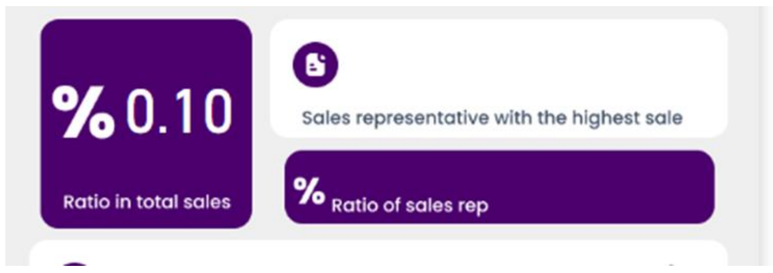
	Customer_Key	Salesperson_Key	Employee	percentage_in_sales	TotalProfit	TotalWithoutTax
1	273	30017	Sophia Hinton	18.15394616588896	30337.100006103516	64499.699951171875
2	273	30014	Kayla Woodcock	14.708564489173042	24608.199989318848	52258.50003051758
3	273	30009	Hudson Onslow	10.220691098449466	17831.29998779297	36313.40002441406
4	273	30015	Lily Code	10.029806384293595	18858.699996948242	35635.19999694824
5	273	30008	Hudson Hollinworth	9.74039736212345	21315.449999809265	34606.949999809265
6	273	30002	Amy Trefl	9.182252392784683	15723.049987792969	32623.899993896484
7	273	30019	Taj Shand	8.548535992326144	17814.04998779297	30372.349983215332
8	273	30003	Anthony Grosse	7.840050321870058	16418.10001373291	27855.149990081787
9	273	30011	Jack Potter	7.1668453932533875	13856.5	25463.300003051758
10	273	30004	Archer Lamble	4.4090089029409185	8396.20000076294	15664.900001525879

```python
import pyodbc
import pandas as pd
import datamanagement

#first distinct employees.
firstsql = "select distinct Employee from DimEmployee"
employees = datamanagement.getdata(firstsql)
print(type(employees))

## employee key relation
secondsql = 'select Employee, Employee_Key from DimEmployee where Employee IN ( select distinct
Employee from DimEmployee ) order by Employee asc'
employee_key_table = datamanagement.getdata(secondsql)
print(employee_key_table)

## assign distinct employees new keys, assign # to "unknown"
    ##unknown is currently located at index 10

#new indexes ata
indexes= list(range(3001,3020))
indexes.append(0)
employees['key'] = indexes

#check whether we visited before
employees['visited'] = 0
employees['visited_sales'] = 0


## change the key of first entry and remove the remaining

for index in range(len(employee_key_table)):
    name = employee_key_table['Employee'].iloc[index]
    print(name)

    #index pointer for EMPLOYEES dataframe, is not for EMPLOYEEKEYTABLE!
    row_index = employees.loc[employees['Employee'] == name].index[0]

    #sql2 = f"DELETE FROM DimEmployee where Employee_Key ={key1} "
    #sql3 = f"UPDATE DimEmployee SET Employee_Key = {key2} WHERE Employee_Key = {key1}"


    if employees.at[row_index, 'visited'] == 0:

        key_to_assign = employees.at[row_index, 'key']
        key_to_delete = employee_key_table.at[index, 'Employee_Key']
        employees.at[row_index, 'visited'] = 1

        sql3 = f"UPDATE DimEmployee SET Employee_Key = {key_to_assign} WHERE Employee_Key =
{key_to_delete}"
        datamanagement.setdata(sql3)
        print(sql3)
        print()
        print("key changed")


        # assign new key
    else:
        key_to_delete = employee_key_table.at[index, 'Employee_Key']
        sql2 = f"DELETE FROM DimEmployee where Employee_Key ={key_to_delete) "

        datamanagement.setdata(sql2)
        print(sql2)
        print()
        print("key deleted")


##CHANGING SALES RECORDS


for index_oldkey in range(len(employee_key_table)):
    #get employee name
    name = employee_key_table['Employee'].iloc[index_oldkey]

    #current key, should be deleted, from second table
    key_to_delete = employee_key_table.at[index_oldkey, 'Employee_Key']

    #get row of the employee from first table
    row_index_in_employee_Table = employees.loc[employees['Employee'] == name].index[0]


    #get new key to assign
    key_to_assign = employees.at[row_index_in_employee_table, 'key']

    #sql to get sales with old key
    sql_getsalesofsalesperson= f'select Sale_Key from FactSale where Salesperson_Key = {key_to_delete}'
    print(sql_getsalesofsalesperson)
    print()
    #get these sales to a dataframe
    df_currentsales_witholdkey = 0
    df_currentsales_witholdkey = datamanagement.getdata(sql_getsalesofsalesperson)


    #for entries in these sales
    for sales_key_index in range(len(df_currentsales_witholdkey)):

        #get sales key, turn into int
        current_sales_key = int(df_currentsales_witholdkey.at[sales_key_index, 'Sale_Key'])
        sql_updatesalesperson= f'UPDATE FactSale SET Salesperson_Key = {key_to_assign} WHERE Sale_Key =
{current_sales_key}'

        datamanagement.setdata(sql_updatesalesperson)
        print(sql_updatesalesperson)
        print()
```

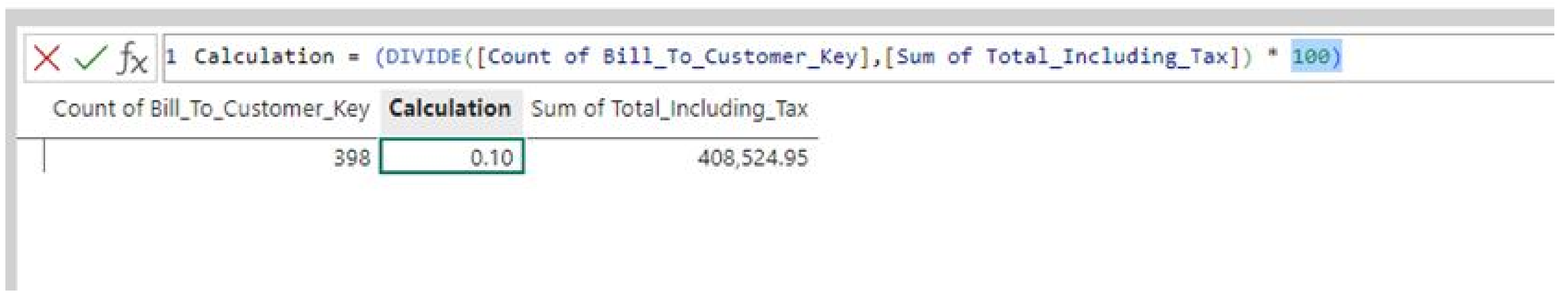
Customer data was already clean, there were no duplicates:

```sql
select count(distinct Customer)
from DimCustomer

select count(Customer)
from DimCustomer
```

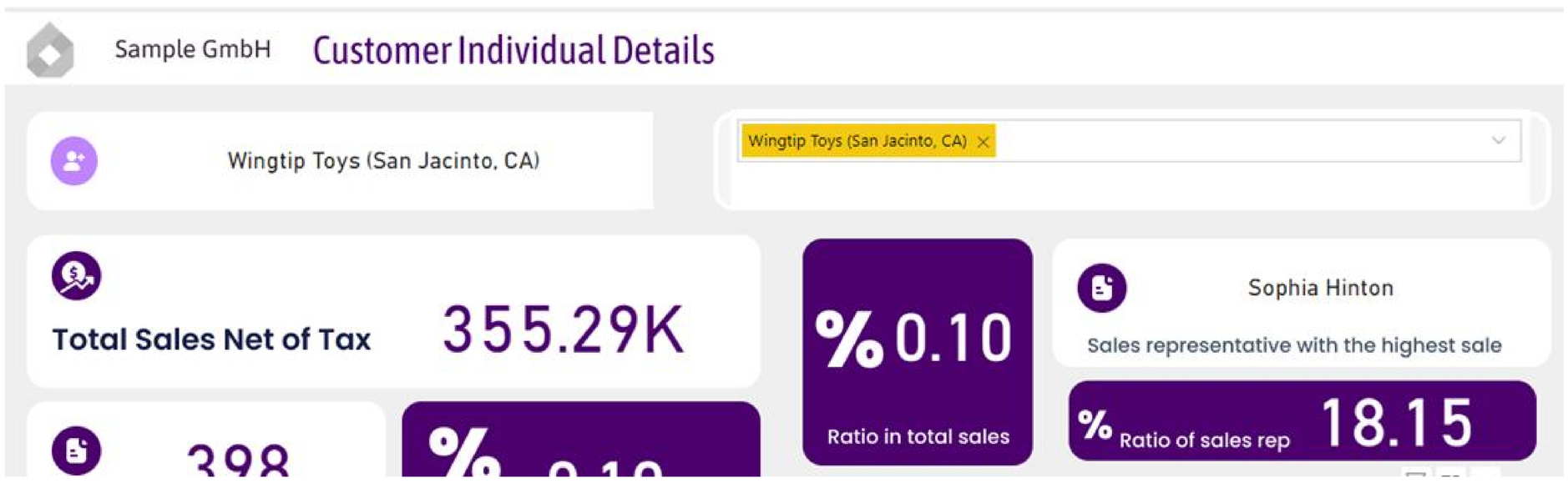
The integration of SQL and PowerBI capabilities



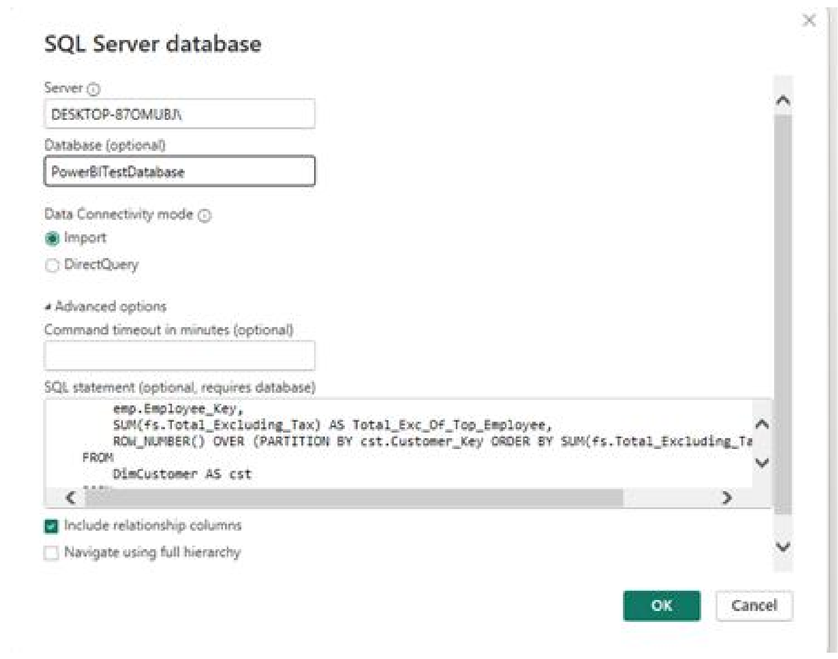
- I have created a custom calculation for the ratio in the total sales field via PowerBI.

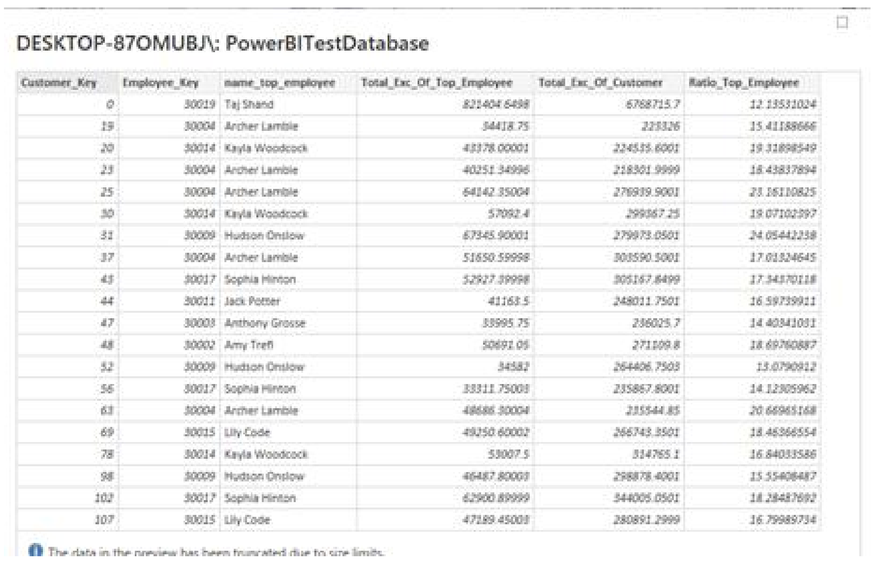


- In the handling of other complex comparative fields, I have found SQL queries to be the most effective way:




```sql
WITH CTE_1 AS (
    SELECT
        cst.Customer_Key,
        emp.Employee_Key,
        SUM(fs.Total_Excluding_Tax) AS Total_Exc_Of_Top_Employee,
        ROW_NUMBER() OVER (PARTITION BY cst.Customer_Key ORDER BY SUM(fs.Total_Excluding_Tax) DESC) AS rn
    FROM
        DimCustomer AS cst
    JOIN
        dbo.FactSale FS ON cst.Customer_Key = FS.Customer_Key
    JOIN
        DimEmployee AS emp ON emp.Employee_Key = fs.Salesperson_Key
    GROUP BY
        cst.Customer_Key, emp.Employee_Key
),

CTE_2 AS(

    select cst.Customer_Key, sum(fs.Total_Excluding_Tax) Total_Exc_Of_Customer
from DimCustomer as cst join dbo.FactSale FS on cst.Customer_Key = FS.Customer_Key
group by cst.Customer_Key

)

SELECT
    CTE_1.Customer_Key,
    CTE_1.Employee_Key,
    Employee as name_top_employee,
    Total_Exc_Of_Top_Employee,
    Total_Exc_Of_Customer,
    (Total_Exc_Of_Top_Employee / Total_Exc_Of_Customer) * 100 AS Ratio_Top_Employee
FROM
    CTE_1 join CTE_2 on CTE_1.Customer_Key = CTE_2.Customer_Key
join DimEmployee as emp on CTE_1.Employee_Key = emp.Employee_Key
WHERE
    rn = 1
ORDER BY
    Customer_Key;
```

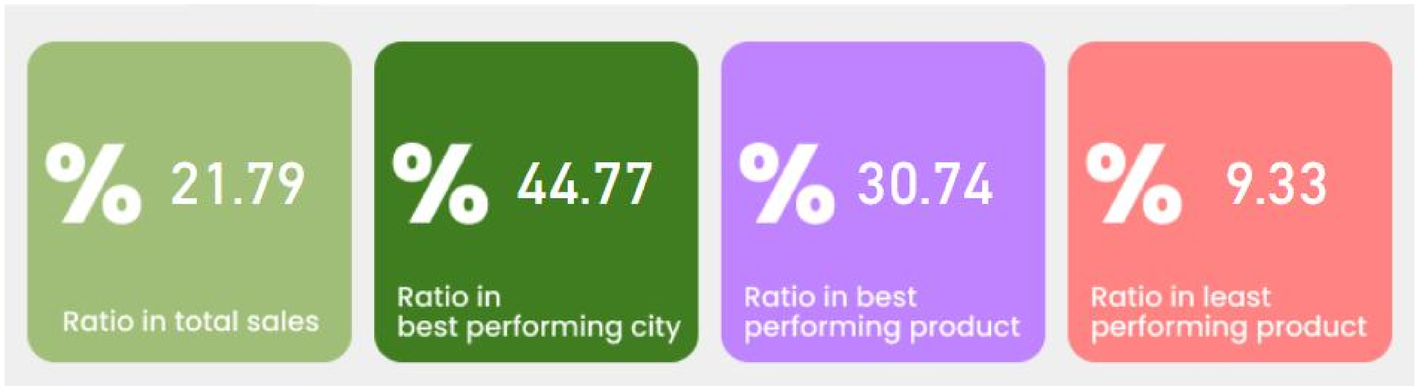




DESKTOP-87OMUBJ\: PowerBITestDatabase

Customer_Key	Employee_Key	name_top_employee	Total_Exc_Of_Top_Employee	Total_Exc_Of_Customer	Ratio_Top_Employee
0	30019	Taj Shand	821404.6488	6788715.7	12.13531024
19	30004	Archer Lamble	54418.75	213326	15.41188666
20	30014	Kayla Woodcock	43378.00001	224535.6001	19.31898549
23	30004	Archer Lamble	40251.34996	218301.9999	18.43837894
25	30004	Archer Lamble	64142.35004	276939.9001	23.16110825
30	30014	Kayla Woodcock	57092.4	299367.25	19.07102397
31	30009	Hudson Onslow	67345.90001	279973.0501	24.05442238
37	30004	Archer Lamble	51650.59998	303590.5001	17.01524645
43	30017	Sophia Hinton	52927.39998	305167.8499	17.34370118
44	30011	Jack Potter	41163.5	248011.7501	16.59739911
47	30003	Anthony Grosse	33995.75	236025.7	14.40341031
48	30002	Amy Trefl	50691.05	271109.8	18.69760887
52	30009	Hudson Onslow	34582	264406.7503	13.0790912
56	30017	Sophia Hinton	33311.75003	235867.8001	14.12305962
63	30004	Archer Lamble	48686.30004	235544.85	20.66965168
69	30015	Lily Code	49250.60002	266743.3501	18.46366554
78	30014	Kayla Woodcock	53007.5	314765.1	16.84033586
98	30009	Hudson Onslow	46487.80003	298878.4001	15.55408487
102	30017	Sophia Hinton	62900.89999	344005.0501	18.28487692
107	30015	Lily Code	47189.45003	280891.2999	16.79989734

The data in the preview has been truncated due to size limits.




```sql
DROP FUNCTION getAggregatedCitySales

CREATE FUNCTION getAggregatedCitySales
(
    @Current_City_Key INT
)
RETURNS TABLE
AS
RETURN(select @Current_City_Key as City_Key, sum(Total_Excluding_Tax) as City_Total_Excluding_Tax_Total
       from DimCity join dbo.FactSale FS on DimCity.City_Key = FS.City_Key
    where DimCity.City_Key = @Current_City_Key

    )

CREATE FUNCTION getScalarValuedAggregatedCitySales
(
    @Current_City_Key INT
)
RETURNS DECIMAL(18, 2)
AS
BEGIN
    DECLARE @City_Total_Excluding_Tax DECIMAL(18, 2);

    SELECT @City_Total_Excluding_Tax = SUM(Total_Excluding_Tax)
    FROM FactSale
    WHERE City_Key = @Current_City_Key;

    RETURN @City_Total_Excluding_Tax;
END

CREATE FUNCTION getScalarValuedAggregatedProductSales
(
    @Current_Product_Key INT
)
RETURNS DECIMAL(18, 2)
AS
BEGIN
    DECLARE @Product_Total_Excluding_Tax DECIMAL(18, 2);

    SELECT @Product_Total_Excluding_Tax = SUM(Total_Excluding_Tax)
    FROM FactSale
    WHERE Stock_Item_Key = @Current_Product_Key;

    RETURN @Product_Total_Excluding_Tax;
END


CREATE FUNCTION getAggregatedProductSales
(
    @Current_Product_Key INT
)
RETURNS TABLE
AS
RETURN(select @Current_Product_Key as Product_Key, sum(Total_Excluding_Tax) as
Product_Total_Excluding_Tax_Total
       from FactSale
        where FactSale.Stock_Item_Key = @Current_Product_Key

    )


WITH

    totalSales as (

    select sum(Total_Excluding_Tax) total_exluding_tax
    from FactSale
    ),

    empSales as (
        select Employee_Key, sum(Total_Excluding_Tax) emp_total_excluding_tax
        from DimEmployee emp join FactSale fs on emp.Employee_Key = fs.Salesperson_Key
        group by emp.Employee_Key

    ),

    empRegionBest as (

        select *
        from (select emp.Employee_Key,
                     fs.City_Key,
                     SUM(fs.Total_Excluding_Tax)
        AS Total_Exc_Of_Top_City,
                     ROW_NUMBER() OVER (PARTITION BY emp.Employee_Key ORDER BY
SUM(fs.Total_Excluding_Tax) DESC) AS rn
              from DimEmployee emp
                       join FactSale fs on emp.Employee_Key = fs.Salesperson_Key
              group by emp.Employee_Key, fs.City_Key) as f
        where rn = 1

    ),

    empItemBest as (

        select *
        from (
        select
        emp.Employee_Key,
        fs.Stock_Item_Key,
        SUM(fs.Total_Excluding_Tax) AS Total_Exc_Of_Top_Item,
            ROW_NUMBER() OVER (PARTITION BY emp.Employee_Key ORDER BY SUM(fs.Total_Excluding_Tax) DESC)
AS rn
        from DimEmployee emp join FactSale fs on emp.Employee_Key = fs.Salesperson_Key
        group by emp.Employee_Key, fs.Stock_Item_Key ) as e
        where rn = 1

    ),

    empItemWorst as (

        select *
        from (
        select
        emp.Employee_Key,
        fs.Stock_Item_Key,
        SUM(fs.Total_Excluding_Tax) AS Total_Exc_Of_Worst_Item,
            ROW_NUMBER() OVER (PARTITION BY emp.Employee_Key ORDER BY SUM(fs.Total_Excluding_Tax) ASC) AS
rn
        from DimEmployee emp join FactSale fs on emp.Employee_Key = fs.Salesperson_Key
        group by emp.Employee_Key, fs.Stock_Item_Key ) as ef
        where rn = 1

    )


SELECT
    empSales.Employee_Key,
    (empSales.emp_total_excluding_tax / totalSales.total_exluding_tax) * 100 AS empRatio,
    (empRegionBest.Total_Exc_Of_Top_City /
dbo.getScalarValuedAggregatedCitySales(empRegionBest.City_Key)) * 100 AS topCityRatio,
    (empItemBest.Total_Exc_Of_Top_Item /
dbo.getScalarValuedAggregatedProductSales(empItemBest.Stock_Item_Key)) * 100 AS topItemRatio,
    (empItemWorst.Total_Exc_Of_Worst_Item /
dbo.getScalarValuedAggregatedProductSales(empItemWorst.Stock_Item_Key)) * 100 AS worstItemRatio
FROM
    totalSales,
    empSales
JOIN empRegionBest ON empSales.Employee_Key = empRegionBest.Employee_Key
JOIN empItemBest ON empRegionBest.Employee_Key = empItemBest.Employee_Key
JOIN empItemWorst ON empItemBest.Employee_Key = empItemWorst.Employee_Key;
```

Thank you for your attention.

Vielen Dank für Ihre Aufmerksamkeit.

I look forward to your feedback.
Ich freue mich auf ihre Rückmeldung.

Kaan Bora Karapınar