

# Portfolio: SQL and PowerBI Learning Project

Kaan Bora Karapınar

# A look at the completed project

**Customer and Sales Details**

**402**  
Total Registered Customer

**Revenue Distribution by Buying Groups**

Buying Group	Percentage
Tailspin Toys	34.65%
N/A	34.68%
Wingtip Toys	40.67%

**Total Sales Net of Tax** **19.88M**

**26.40K** **% 0.12**  
Total amount paid by invoice

**2.98M** **% 22.86M**  
Total generated tax Total sales with tax

**Customer and Sales Details**

**200**  
Total Registered Customer

**Revenue Distribution by Buying Groups**

Buying Group	Percentage
Tailspin Toys	34.65%
Wingtip Toys	40.67%
N/A	24.68%

**Total Sales Net of Tax** **5.05M**

**6413** **% 0.11**  
Total amount paid by invoice

**756.17K** **% 5.81M**  
Total generated tax Total sales with tax

**Salesperson Details**

**Sales representatives sorted by sales net of tax**

Salesperson	Generated Tax	Sales net of tax
Sophia Hinton	332,302.50	2,219,627.80
Lily Code	316,251.50	2,112,719.25
Kayla Woodcock	311,822.89	2,081,322.80
Taj Shand	300,422.28	2,008,877.10
Archer Lamble	300,093.72	2,005,033.50

Select a criteria to filter results

- Select a product**
- Select a region**
- Select a customer**

**Reset all filters**

**Salesperson Details**

**Sales representatives sorted by sales net of tax**

Salesperson	Generated Tax	Sales net of tax
Amy Trefl	23,816.70	158,778.00
Anthony Grossie	39,468.60	263,124.00
Archer Lamble	31,433.40	209,556.00
Hudson Hollinworth	41,362.65	275,751.00
Hudson Onslow	30,173.85	201,159.00

Select a product

- 10 mm Anti static bubble wrap**
- 20 mm Anti static bubble wrap**
- Air cushion machine (Blue)**

**Close Filter** **Reset all filters**

**Customer Individual Details**

**Tailspin Toys (Absecon, NJ)**

**Total Sales Net of Tax** **19.88M**

**26.40K** **% 0.12**  
Total amount paid by invoice

**2.98M** **% 22.86M**  
Total generated tax Total sales with tax

**Amy Trefl**  
Sales representative with the highest sale

**% Ratio of sales rep** **883.74**

Stock Item	Sum of Total Excluding Tax
Air cushion machine (Blue)	1,198,269.00
10 mm Anti static bubble wrap (Blue) 50m	866,250.00
20 mm Double sided bubble wrap 50m	790,560.00
32 mm Double sided bubble wrap 50m	769,440.00
32 mm Anti static bubble wrap (Blue) 50m	631,050.00
10 mm Double sided bubble wrap 50m	605,850.00
20 mm Anti static bubble wrap (Blue) 50m	587,520.00
Void fill 400 L bag (White) 400L	405,500.00
32 mm Anti static bubble wrap (Blue) 20m	360,000.00
20 mm Anti static bubble wrap (Blue) 20m	309,600.00
10 mm Anti static bubble wrap (Blue) 20m	304,080.00
Void fill 300 L bag (White) 300L	264,375.00
32 mm Double sided bubble wrap 20m	263,810.00
Ride on bia wheel monster truck (Black) 1/12 scale	222,870.00

**Best-performing products**

**Salesperson Individual Details**

**Lily Code**

**4.33M**  
Total sales net of Tax

**% 21.79** **% 44.77**  
Ratio in total sales Ratio in best performing city

**% 30.74** **% 9.33**  
Ratio in best performing product Ratio in least performing product

**Akhik**  
Sales net of tax

**Wapinitia**  
Sales net of tax

**San Jacinto**  
Sales net of tax

**Best-performing regions**

**Air cushion machine (Blue)**  
Sales net of tax

**10 mm Anti static bubble wrap (Blue) 50m**  
Sales net of tax

**10 mm Double sided bubble wrap 50m**  
Sales net of tax

**Unique Customer Without Group**  
Sales net of tax

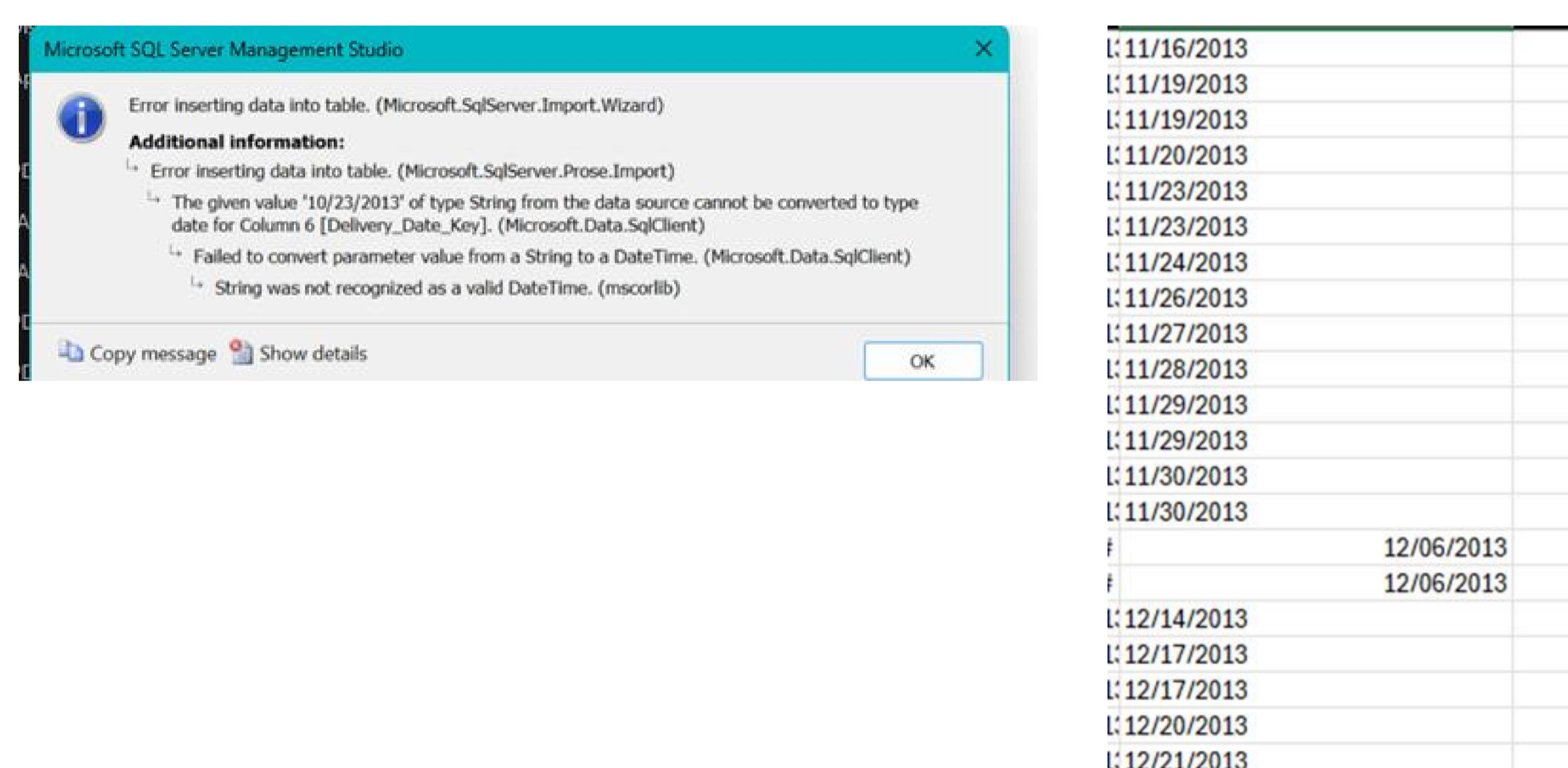
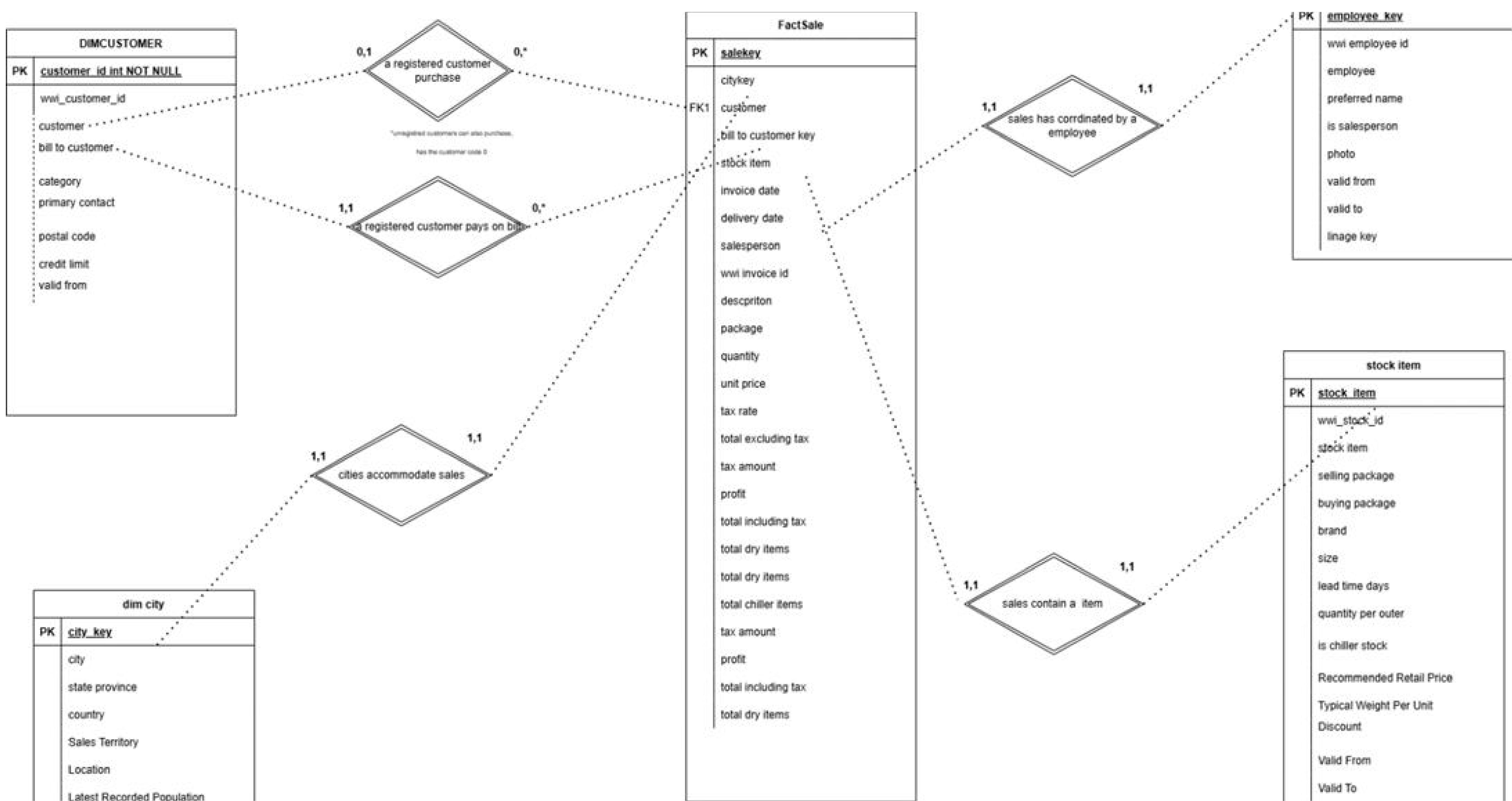
**Wingtip Toys (Wapinitia, OR)**  
Sales net of tax

**Wingtip Toys (San Jacinto, CA)**  
Sales net of tax

**Best-performing customers**

# 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 Excel 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\borakanapinar\PycharmProjects\SQL\_demo\project\datecheck.py

```
11  for line in reader:
12
13      print("currently iterating Delivery Date Key")
14
15      currentlist = line[6].split("/")
16
17      #skipping first line with if
18      if (currentlist[0] == "Delivery Date Key"):
19          continue
20      if (currentlist[0] == "NULL"):
21          continue
22      if (currentlist[1] == "NULL"):
23          continue
24
25      intvalfirstelement = int(currentlist[0])
26      intvalsecondelement = int(currentlist[1])
27
28      ismonth = intvalfirstelement <= 12 and intvalfirstelement > 0
29      isday= intvalsecondelement <= 31 and intvalsecondelement > 0
30
31      result_at_current_iteration = ismonth and isday
32      if(not result_at_current_iteration):
33          print("problem at: " + line[5])
34          delivery_date_key_pass_tests = False
35
36      print("currently iterating Invoice Date Key")
37
38      currentlist = line[5].split("/")
39
40      # skipping first line with if
41      if (currentlist[0] == "Delivery Date Key"):
42          continue
43      if (currentlist[0] == "NULL"):
44          continue
45      if (currentlist[1] == "NULL"):
46          continue
47
48      intvalfirstelement = int(currentlist[0])
49      intvalsecondelement = int(currentlist[1])
50
51      ismonth = intvalfirstelement <= 12 and intvalfirstelement > 0
52      isday = intvalsecondelement <= 31 and intvalsecondelement > 0
53
54      result_at_current_iteration = ismonth and isday
55      if (not result_at_current_iteration):
56          print("problem at: " + line[5])
57          invoice_date_key_pass_tests = False
58
59      if(delivery_date_key_pass_tests):
60          print("success at delivery date")
61      if (invoice_date_key_pass_tests):
62          print("success at invoice date")
63
64
65      for line in reader:
66
67          invoicenull = line[5] == "NULL"
68          invoiceheader = line[5] == "Invoice Date Key"
69
70          delivernull = line[6] == "NULL"
71          deliveryheader = line[6] == "Delivery Date Key"
72
73          if(not invoicenull and not invoiceheader):
74              currentlistforinvoice = line[5].split("/")
75              temp1 = currentlistforinvoice[1]
76              currentlistforinvoice[1] = currentlistforinvoice[0]
77              currentlistforinvoice[0] = temp1
78
79              line[5] = "/".join(currentlistforinvoice)
80
81              if (not delivernull and not deliveryheader):
82                  currentlistfordelivery = line[6].split("/")
83                  temp2 = currentlistfordelivery[1]
84                  currentlistfordelivery[1] = currentlistfordelivery[0]
85                  currentlistfordelivery[0] = temp2
86
87                  line[6] = "/".join(currentlistfordelivery)
88
89
90      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:

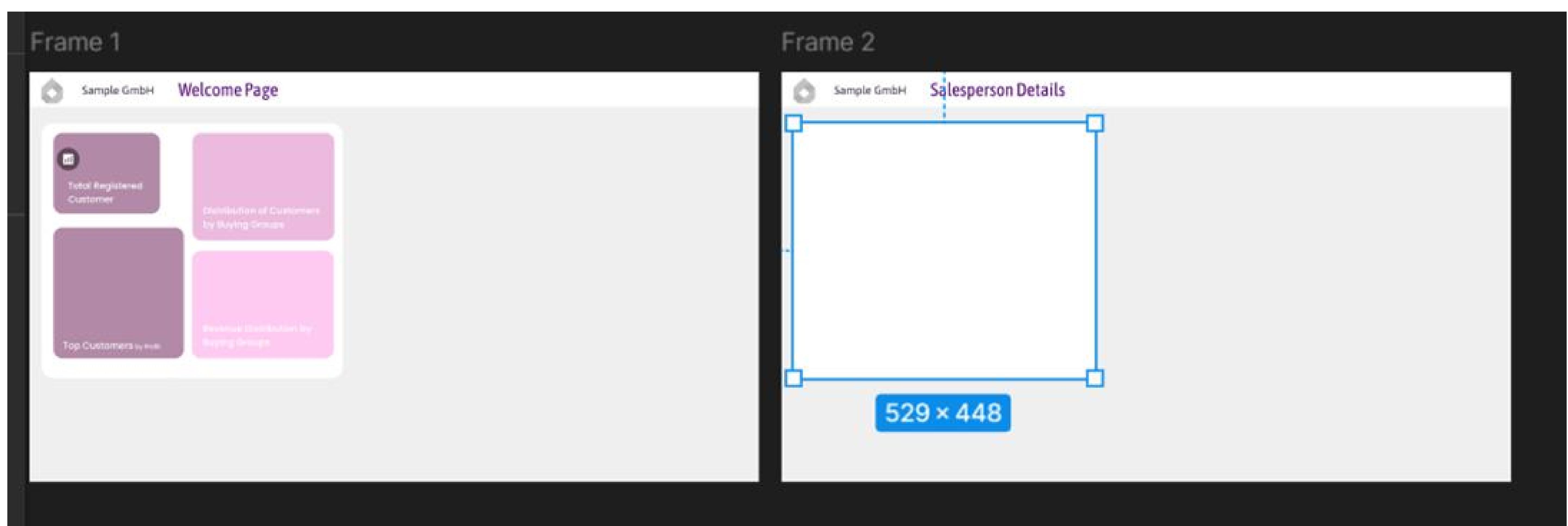
- Who are our top 5 salespeople by total revenue?
- For each top salesperson, what are their highest and lowest revenue-generating products?

Revenue:

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

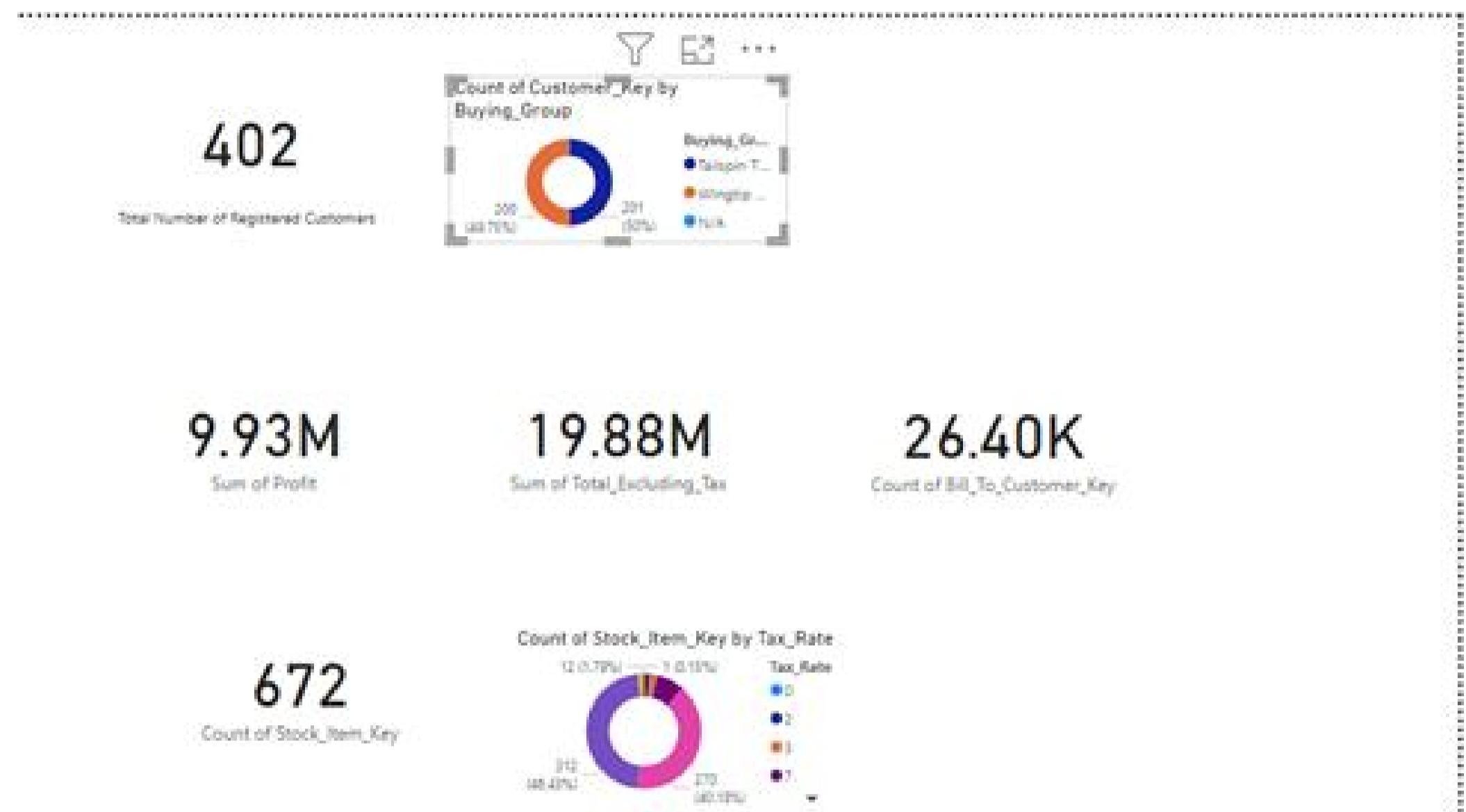
## Creating a brand identity

- 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.



```
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:

```
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( @CurrentCustomer_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
2	273	93	23390	11424.5
3	273	81	22852.800003051758	9482.19996948242
4	273	48	19643	10161
5	273	97	18162	11373
6	273	134	16437	8985
7	273	19	15664.900001525879	8396.20000076294
8	273	102	14528.5	7704
9	273	105	14520.400024414062	7295.29998792969
10	273	108	13656.79998792969	6375.649993896484

- **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 Employee 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.**)

Employee_Key	Employee
1	Alicia Fatnawa
2	Alicia Fatnawa
3	Alicia Fatnawa
4	Alicia Fatnawa
5	Alicia Fatnawa
6	Alicia Fatnawa
7	Alicia Fatnawa
8	Alicia Fatnawa
9	Alicia Fatnawa
10	Alicia Fatnawa
11	Amy Trefl
12	Amy Trefl
13	Amy Trefl
14	Amy Trefl
15	Amy Trefl
16	Amy Trefl
17	Amy Trefl
18	Amy Trefl
19	Amy Trefl
20	Amy Trefl
21	Amy Trefl
22	Amy Trefl
23	Amy Trefl
24	Amy Trefl
25	Amy Trefl
26	Amy Trefl
27	Amy Trefl
28	Amy Trefl
29	Amy Trefl
30	Amy Trefl
31	Amy Trefl
32	Amy Trefl
33	Amy Trefl
34	Amy Trefl
35	Amy Trefl
36	Amy Trefl
37	Amy Trefl
38	Amy Trefl
39	Amy Trefl
40	Amy Trefl
41	Amy Trefl
42	Amy Trefl
43	Amy Trefl
44	Amy Trefl
45	Amy Trefl
46	Amy Trefl
47	Amy Trefl
48	Amy Trefl
49	Amy Trefl
50	Amy Trefl
51	Amy Trefl
52	Amy Trefl
53	Amy Trefl
54	Amy Trefl
55	Amy Trefl
56	Amy Trefl
57	Amy Trefl
58	Amy Trefl
59	Amy Trefl
60	Amy Trefl
61	Amy Trefl
62	Amy Trefl
63	Amy Trefl
64	Amy Trefl
65	Amy Trefl
66	Amy Trefl
67	Amy Trefl
68	Amy Trefl
69	Amy Trefl
70	Amy Trefl
71	Amy Trefl
72	Amy Trefl
73	Amy Trefl
74	Amy Trefl
75	Amy Trefl
76	Amy Trefl
77	Amy Trefl
78	Amy Trefl
79	Amy Trefl
80	Amy Trefl
81	Amy Trefl
82	Amy Trefl
83	Amy Trefl
84	Amy Trefl
85	Amy Trefl
86	Amy Trefl
87	Amy Trefl
88	Amy Trefl
89	Amy Trefl
90	Amy Trefl
91	Amy Trefl
92	Amy Trefl
93	Amy Trefl
94	Amy Trefl
95	Amy Trefl
96	Amy Trefl
97	Amy Trefl
98	Amy Trefl
99	Amy Trefl
100	Amy Trefl
101	Amy Trefl
102	Amy Trefl
103	Amy Trefl
104	Amy Trefl
105	Amy Trefl
106	Amy Trefl
107	Amy Trefl
108	Amy Trefl
109	Amy Trefl
110	Amy Trefl
111	Amy Trefl
112	Amy Trefl
113	Amy Trefl
114	Amy Trefl
115	Amy Trefl
116	Amy Trefl
117	Amy Trefl
118	Amy Trefl
119	Amy Trefl
120	Amy Trefl
121	Amy Trefl
122	Amy Trefl
123	Amy Trefl
124	Amy Trefl
125	Amy Trefl
126	Amy Trefl
127	Amy Trefl
128	Amy Trefl
129	Amy Trefl
130	Amy Trefl
131	Amy Trefl
132	Amy Trefl
133	Amy Trefl
134	Amy Trefl
135	Amy Trefl
136	Amy Trefl
137	Amy Trefl
138	Amy Trefl
139	Amy Trefl
140	Amy Trefl
141	Amy Trefl
142	Amy Trefl
143	Amy Trefl
144	Amy Trefl
145	Amy Trefl
146	Amy Trefl
147	Amy Trefl
148	Amy Trefl
149	Amy Trefl
150	Amy Trefl
151	Amy Trefl
152	Amy Trefl
153	Amy Trefl
154	Amy Trefl
155	Amy Trefl
156	Amy Trefl
157	Amy Trefl
158	Amy Trefl
159	Amy Trefl
160	Amy Trefl
161	Amy Trefl
162	Amy Trefl
163	Amy Trefl
164	Amy Trefl
165	Amy Trefl
166	Amy Trefl
167	Amy Trefl
168	Amy Trefl
169	Amy Trefl
170	Amy Trefl
171	Amy Trefl
172	Amy Trefl
173	Amy Trefl
174	Amy Trefl
175	Amy Trefl
176	Amy Trefl
177	Amy Trefl
178	Amy Trefl
179	Amy Trefl
180	Amy Trefl
181	Amy Trefl
182	Amy Trefl
183	Amy Trefl
184	Amy Trefl
185	Amy Trefl
186	Amy Trefl
187	Amy Trefl
188	Amy Trefl
189	Amy Trefl
190	Amy Trefl
191	Amy Trefl
192	Amy Trefl
193	Amy Trefl
194	Amy Trefl
195	Amy Trefl
196	Amy Trefl
197	Amy Trefl
198	Amy Trefl
199	Amy Trefl
200	Amy Trefl
201	Amy Trefl
202	Amy Trefl
203	Amy Trefl
204	Amy Trefl
205	Amy Trefl
206	Amy Trefl
207	Amy Trefl
208	Amy Trefl
209	Amy Trefl
210	Amy Trefl
211	Amy Trefl
212	Amy Trefl
213	Amy Trefl
214	Amy Trefl
215	Amy Trefl
216	Amy Trefl
217	Amy Trefl
218	Amy Trefl
219	Amy Trefl
220	Amy Trefl
221	Amy Trefl
222	Amy Trefl
223	Amy Trefl
224	Amy Trefl
225	Amy Trefl
226	Amy Trefl
227	Amy Trefl
228	Amy Trefl
229	Amy Trefl
230	Amy Trefl
231	Amy Trefl
232	Amy Trefl
233	Amy Trefl
234	Amy Trefl
235	Amy Trefl
236	Amy Trefl
237	Amy Trefl
238	Amy Trefl
239	Amy Trefl
240	Amy Trefl
241	Amy Trefl
242	Amy Trefl
243	Amy Trefl
244	Amy Trefl
245	Amy Trefl
246	Amy Trefl
247	Amy Trefl
248	Amy Trefl
249	Amy Trefl
250	Amy Trefl
251	Amy Trefl
252	Amy Trefl
253	Amy Trefl
254	Amy Trefl
255	Amy Trefl
256	Amy Trefl
257	Amy Trefl
258	Amy Trefl
259	Amy Trefl
260	Amy Trefl
261	Amy Trefl
262	Amy Trefl
263	Amy Trefl
264	Amy Trefl
265	Amy Trefl
266	Amy Trefl
267	Amy Trefl
268	Amy Trefl
269	Amy Trefl
270	Amy Trefl
271	Amy Trefl
272	Amy Trefl
273	Amy Trefl
274	Amy Trefl
275	Amy Trefl
276	Amy Trefl
277	Amy Trefl
278	Amy Trefl
279	Amy Trefl
280	Amy Trefl
281	Amy Trefl
282	Amy Trefl
283	Amy Trefl
284	Amy Trefl
285	Amy Trefl
286	Amy Trefl
287	Amy Trefl
288	Amy Trefl
289	Amy Trefl
290	Amy Trefl
291	Amy Trefl
292	Amy Trefl
293	Amy Trefl
294	Amy Trefl
295	Amy Trefl
296	Amy Trefl
297	Amy Trefl
298	Amy Trefl
299	Amy Trefl
300	Amy Trefl
301	Amy Trefl
302	Amy Trefl
303	Amy Trefl
304	Amy Trefl
305	Amy Trefl
306	Amy Trefl
307	Amy Trefl
308	Amy Trefl
309	Amy Trefl
310	Amy Trefl
311	Amy Trefl
312	Amy Trefl
313	Amy Trefl
314	Amy Trefl
315	Amy Trefl
316	Amy Trefl
317	Amy Trefl
318	Amy Trefl
319	Amy Trefl
320	Amy Trefl
321	Amy Trefl
322	Amy Trefl
323	Amy Trefl
324	Amy Trefl
325	Amy Trefl
326	Amy Trefl
327	Amy Trefl
328	Amy Trefl
329	Amy Trefl
330	Amy Trefl
331	Amy Trefl
332	Amy Trefl
333	Amy Trefl
334	Amy Trefl
335	Amy Trefl
336	Amy Trefl
337	Amy Trefl
338	Amy Trefl
339	Amy Trefl
340	Amy Trefl
341	Amy Trefl
342	Amy Trefl
343	Amy Trefl
344	Amy Trefl
345	Amy Trefl
346	Amy Trefl
347	Amy Trefl
348	Amy Trefl
349	Amy Trefl
350	Amy Trefl
351</	

**Customer data was already clean, there were no duplicates:**

```
select count(distinct Customer)
from DimCustomer

select count(Customer)
from DimCustomer
```

```
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 relations
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 key, assign # to "unknown"
#unknown is currently located at index 19

#new indexes at
indexes= list(range(30000,30020))
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 reuse the remaining
for index in range(len(employee_key_table)):
    name = employee_key_table['Employee'].iloc[index]
    print(name)

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

sql1 = f"DELETE FROM DimEmployee where Employee_Key = {key}"
sql2 = f"UPDATE DimEmployee SET Employee_Key = {row_index} WHERE Employee_Key = {key}"

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")

#RECHANGING SALES RECORDS

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

    #get all sales with old key
    sql_getsalesofsalesperson = f"select Sale_Key from FactSale where Salesperson_Key = {key_to_delete}"
    print(sql_getsalesofsalesperson)
    print()
    #put these sales to a data frame
    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)):

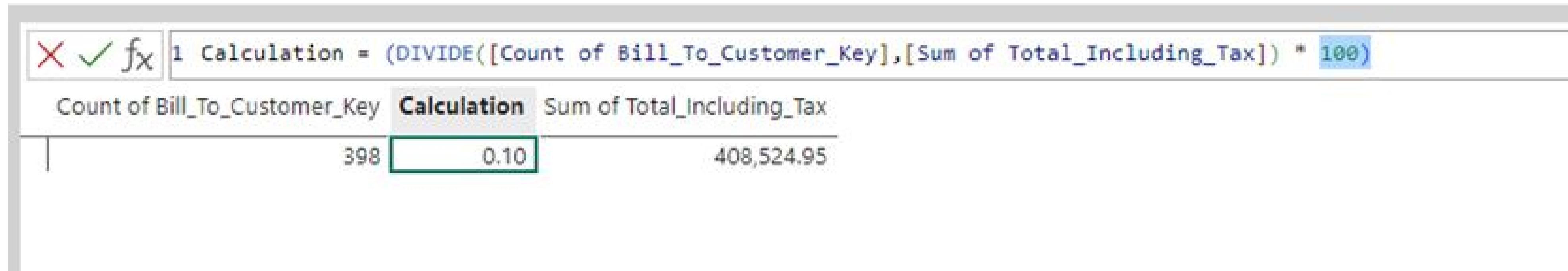
        #get sales key, turn into int
        current_sales_key = int(df_currentsales_witholdkey.at[sales_key_index, 'Sale_Key'])
        sql_updatesalesperson = f"UPDATE FactSale SET Salesperson_Key = {key_to_assign} WHERE Sale_Key = {current_sales_key}"

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

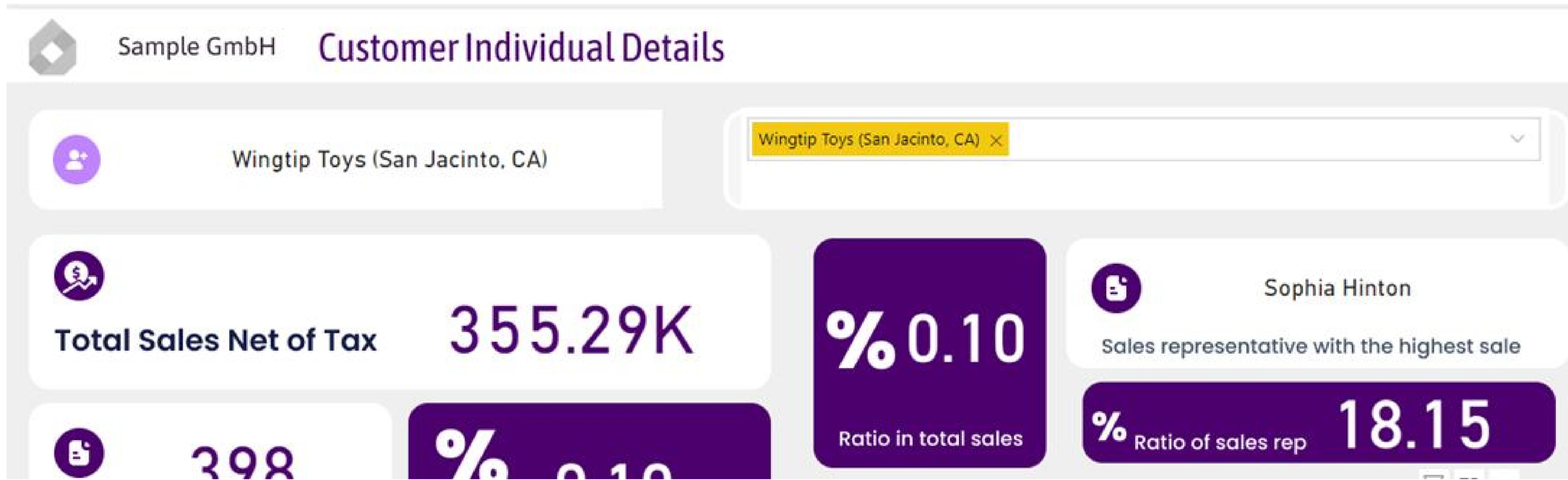
# The integration of SQL and PowerBI capabilities



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

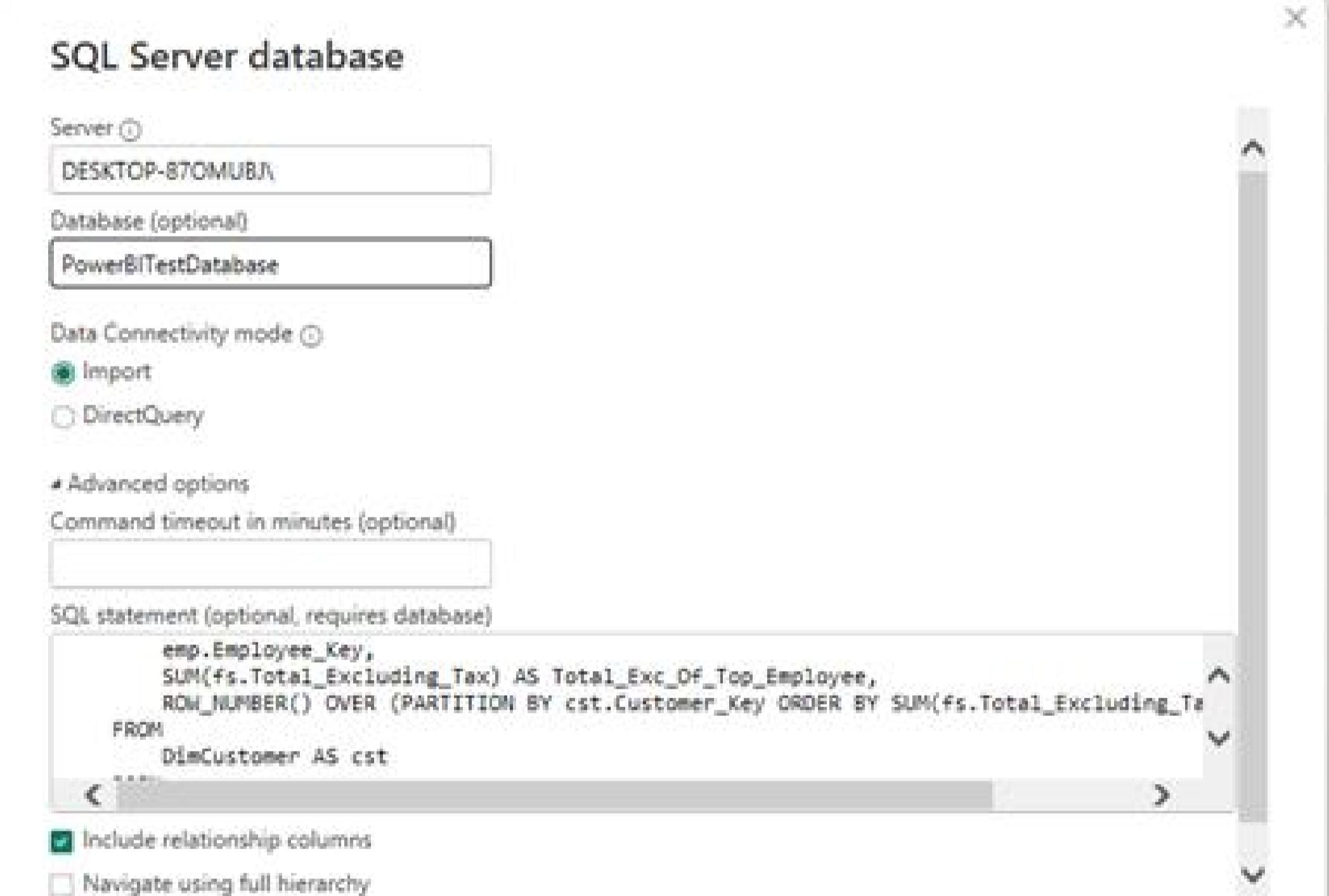


- In the handling of other complex comparative fields, I have found SQL queries to be the most effective way:



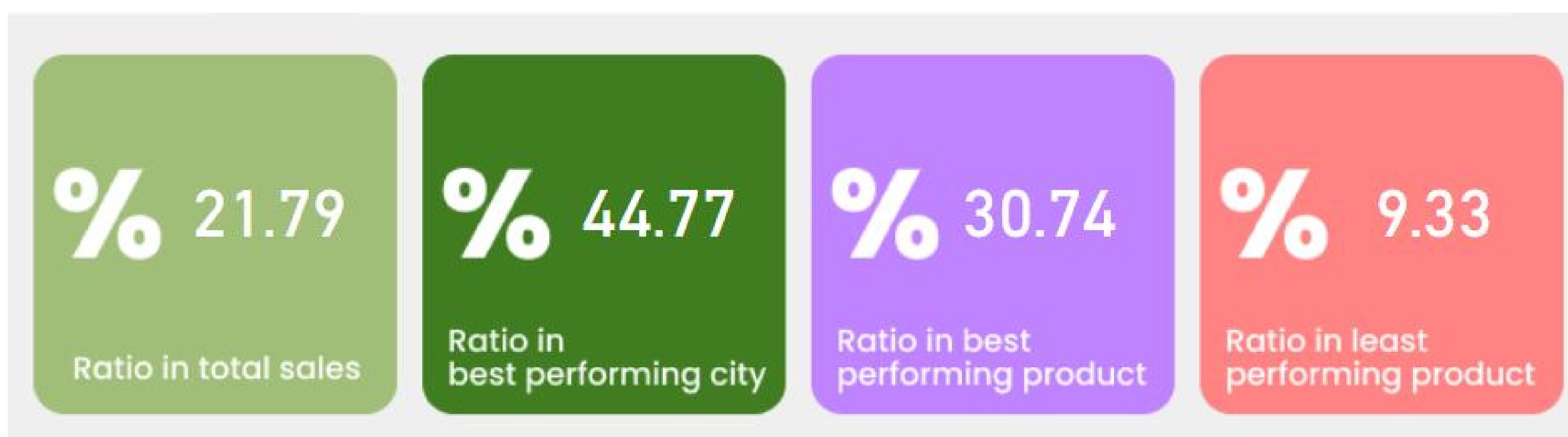
A screenshot of an SQL Server Management Studio window. It contains a complex Common Table Expression (CTE) query. The code includes multiple joins between DimCustomer, FactSale, and DimEmployee tables, along with various aggregate functions like SUM and ROW\_NUMBER, and calculations for ratios and percentages.

```
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;
```



A screenshot of a PowerBI preview pane titled "DESKTOP-870MUB\J: PowerBITestDatabase". It displays a table with columns: Customer\_Key, Employee\_Key, name\_top\_employee, Total\_Exc\_of\_top\_employee, Total\_Exc\_of\_Customer, and Ratio\_top\_employee. The table contains 107 rows of data. A note at the bottom of the preview pane states: "The data in the preview has been truncated due to size limits."

Customer_Key	Employee_Key	name_top_employee	Total_Exc_of_top_employee	Total_Exc_of_Customer	Ratio_top_employee
0	30019	Taj Stand	821404.6498	6768715.7	12.3531024
19	30004	Archer Lamble	34418.75	238326	15.41188666
20	30014	Kayla Woodcock	4378.00071	224535.6001	19.3189549
23	30004	Archer Lamble	40251.34998	218301.9999	18.43837894
25	30004	Archer Lamble	64142.35004	276939.9001	23.15110825
30	30014	Kayla Woodcock	57082.4	299867.25	19.07102297
31	30009	Hudson Osnlow	67345.90001	279973.0501	24.05442238
37	30004	Archer Lamble	51650.59998	305590.5001	17.0324645
43	30017	Sophia Hinton	52527.39998	305187.8499	17.34437018
44	30011	Jack Potter	41163.5	248011.7501	16.59739911
47	30003	Anthony Grosse	33995.75	236025.7	14.40341031
48	30002	Amy Treff	50691.05	271109.8	18.67960887
52	30009	Hudson Osnlow	34582	264406.7503	13.0790012
56	30017	Sophia Hinton	33311.75009	235857.8001	14.12305962
63	30004	Archer Lamble	48686.30004	235544.85	20.6998168
69	30015	Lily Code	49250.60002	266743.3501	18.48366554
78	30014	Kayla Woodcock	53007.5	314765.1	16.64033586
98	30009	Hudson Osnlow	46487.80003	298878.4001	15.55408487
102	30017	Sophia Hinton	62900.89999	344005.0501	18.28487692
107	30015	Lily Code	47189.45003	280891.2999	16.79988734



```

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