# PART 1 – Business & Tableau Question

Develop an interactive dashboard that support strategic decision making with Tableau.

Please add your answer/analysis together with the dashboard.

*Answer:* The question is unclear. Perhaps the segments we should prioritize are “Segment Size”, “Spend per Customer”, “Population Growth” and “Overall Spend”?

A screenshot of a computer screen

Description automatically generated

# PART 2 - SQL Question & PowerBI Question

Load these files into a SQL database. Double click the icons below to open them.

File Structure

Graphical user interface, table

Description automatically generated

Provide the SQL Statement for each of the questions below.

## Question 1: Percentage Distribution

Total Sales by Product Category and the contribution against total sales.

Expected Results:

A screenshot of a computer

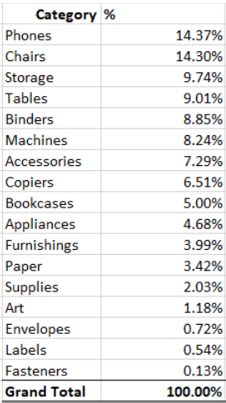
Description automatically generated

*Answer:*

|  |
| --- |
| SELECT ProductsCSV.CATEGORY, ROUND(SUM(OrdersCSV.SALES),3) as SUM\_SALES, ROUND(SUM(OrdersCSV.SALES) /(SELECT SUM(OrdersCSV.SALES)  FROM OrdersCSV INNER JOIN ProductsCSV ON OrdersCSV.PRODUCT\_ID = ProductsCSV.ID) \*100,2) AS SALES\_PERCENTAGE  FROM OrdersCSV INNER JOIN ProductsCSV ON OrdersCSV.PRODUCT\_ID = ProductsCSV.ID  GROUP BY ProductsCSV.CATEGORY |

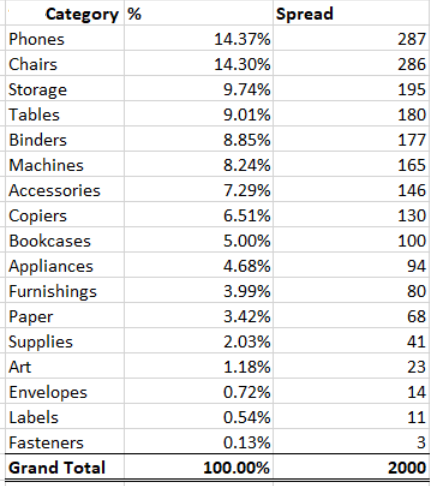
## Question 2: Whole number allocation

You have a product distribution as below:



You are required to spread a fixed value of 2,000 across all products based on this distribution percentage. However, the spread of these number needs to be in whole number and not decimal point. The sum of all this product must add up to exactly **2,000**.

Expected Results:



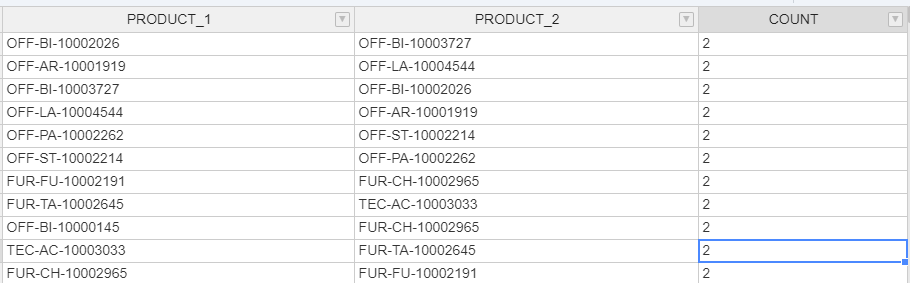
*Answer:*

|  |
| --- |
| SELECT ProductsCSV.CATEGORY, ROUND(SUM(OrdersCSV.SALES) /(SELECT SUM(OrdersCSV.SALES)  FROM OrdersCSV INNER JOIN ProductsCSV ON OrdersCSV.PRODUCT\_ID = ProductsCSV.ID) \*2000,0) AS SPREAD\_FIX\_2000  FROM OrdersCSV INNER JOIN ProductsCSV ON OrdersCSV.PRODUCT\_ID = ProductsCSV.ID  GROUP BY ProductsCSV.CATEGORY  SELECT SUM(OrdersCSV.SALES) /SUM(OrdersCSV.SALES) \*2000 AS SPREAD\_FIX\_2000\_TOTAL  FROM OrdersCSV INNER JOIN ProductsCSV ON OrdersCSV.PRODUCT\_ID = ProductsCSV.ID |

## Question 3: Basket Data

What are the 2 most common products purchased together in an order?

Expected output:



*Answer:*

|  |
| --- |
| WITH PRODUCTS AS (      SELECT table1.PRODUCT\_ID AS ID\_PRODUCT1,             table2.PRODUCT\_ID AS ID\_PRODUCT2,             COUNT(\*) AS COUNT\_OCCURENCE      FROM OrdersCSV table1 JOIN OrdersCSV table2 ON table1.ORDER\_ID = table2.ORDER\_ID      WHERE table1.PRODUCT\_ID < table2.PRODUCT\_ID      GROUP BY table1.PRODUCT\_ID, table2.PRODUCT\_ID)    SELECT ID\_PRODUCT1,         ID\_PRODUCT2,         COUNT\_OCCURENCE  FROM (SELECT ID\_PRODUCT1,               ID\_PRODUCT2,               COUNT\_OCCURENCE,               RANK() OVER (ORDER BY COUNT\_OCCURENCE DESC) AS RANKED        FROM PRODUCTS) AS RANKED\_PRODUCTS  WHERE RANKED <= 2 |

## Question 4: Basket Data

With the dataset, develop a PowerBI dashboard that support strategic decision making.



# PART 3 - Business & SQL Question

Please find attached 2 csv files, a business analysis problem, and a SQL challenge below which we'd like you to tackle.

   
  
**Data overview:**  
  
Payments: Each line represents a transaction payment for a contract  
• TransactionID -> primary key / unique identifier  
• ContractID -> a contract will have multiple repayments  
• ClientID -> the client of the contract  
• TransactionDate -> This is date in a EPOCH format - you will have to figure out how to convert it  
• Amount -> Repayment amount  
• Payment Code -> DEFAULT means a payment was not made

Clients: Each line represents a client and their information  
• ClientID  
• Entity Type -> Business type  
• Entity Year Established -> First year the business was open  
  
**1. Business question:**  
Payment defaults are detrimental to the business and are a significant cost factor.  
Are there any key trends in the data which can help me avoid default-prone customers in the future?

**Feel free to process and explore the data in any tool. Please do the analysis in a dashboarding tool of your choice.**

*Answer:* I chose SQL for this analysis

Step 1.1: I check PRIMARY KEY (transaction\_id) column by using this query, which gives the result of 25559 (meaning all rows in this column is not null)

|  |
| --- |
| SELECT COUNT(transaction\_id)  FROM PaymentsCSV  WHERE transaction\_id IS NOT NULL |

Step 1.2: I set PRIMARY KEY for this column (transaction\_id) by this query

|  |
| --- |
| ALTER TABLE PaymentsCSV  ADD CONSTRAINT PK\_transaction\_id PRIMARY KEY (transaction\_id) |

Step 2.1: I use this query to understand that each client might have more than 1 business entity in dataset.

|  |
| --- |
| SELECT transaction\_id,COUNT(transaction\_id) AS ROWS\_REPEATED  FROM     ClientsCSV RIGHT JOIN                    PaymentsCSV ON ClientsCSV.client\_id = PaymentsCSV.client\_id  GROUP BY transaction\_id  HAVING COUNT(transaction\_id) >1 |

Step 2.2: I use this query to understand the dataset

|  |
| --- |
| SELECT PaymentsCSV.transaction\_id, PaymentsCSV.contract\_id, PaymentsCSV.client\_id,ClientsCSV.client\_id ,PaymentsCSV.transaction\_date, PaymentsCSV.payment\_amt, PaymentsCSV.payment\_code, ClientsCSV.entity\_type,                    ClientsCSV.entity\_year\_established  FROM     ClientsCSV RIGHT JOIN                    PaymentsCSV ON ClientsCSV.client\_id = PaymentsCSV.client\_id |

Step 3.1: I created a view called PAYMENT\_REPEATED\_RESEARCH

|  |
| --- |
| CREATE VIEW PAYMENT\_REPEATED\_RESEARCH AS  SELECT PaymentsCSV.client\_id,COUNT(transaction\_id) AS PAYMENT\_REPEATED,SUM(PaymentsCSV.payment\_amt) AS TOTAL\_PAYMENT\_AMT, PaymentsCSV.payment\_code, ClientsCSV.entity\_type, ClientsCSV.entity\_year\_established  FROM     ClientsCSV RIGHT JOIN                    PaymentsCSV ON ClientsCSV.client\_id = PaymentsCSV.client\_id  GROUP BY PaymentsCSV.client\_id,PaymentsCSV.payment\_code, ClientsCSV.entity\_type, ClientsCSV.entity\_year\_established |

Step 3.2: I executed these 3 queries, respectively. I conclude that there are:

* 1287 clients who proceed “PAYMENT” -> first query
* Among 1287 clients, 376 clients who proceed both “PAYMENT” and “DEFAULT” -> second query
* Among 1287 clients, 368 clients who proceed “DEFAULT” -> third query

|  |
| --- |
| SELECT PaymentsCSV.client\_id,COUNT(transaction\_id) AS PAYMENT\_REPEATED,SUM(PaymentsCSV.payment\_amt) AS TOTAL\_PAYMENT\_AMT, PaymentsCSV.payment\_code, ClientsCSV.entity\_type, ClientsCSV.entity\_year\_established  FROM     ClientsCSV RIGHT JOIN                    PaymentsCSV ON ClientsCSV.client\_id = PaymentsCSV.client\_id  WHERE PaymentsCSV.payment\_code = 'PAYMENT'  GROUP BY PaymentsCSV.client\_id,PaymentsCSV.payment\_code, ClientsCSV.entity\_type, ClientsCSV.entity\_year\_established    SELECT A.client\_id,A.PAYMENT\_REPEATED as PAYMENT\_REPEATED\_1,A.TOTAL\_PAYMENT\_AMT AS TOTAL\_PAYMENT\_AMT\_1, A.payment\_code AS payment\_code\_1,B.payment\_code AS payment\_code\_2,B.PAYMENT\_REPEATED as PAYMENT\_REPEATED\_2,B.TOTAL\_PAYMENT\_AMT AS TOTAL\_PAYMENT\_AMT\_2, A.entity\_type, A.entity\_year\_established  FROM PAYMENT\_REPEATED\_RESEARCH A INNER JOIN PAYMENT\_REPEATED\_RESEARCH B ON A.client\_id = B.client\_id  WHERE A.payment\_code = 'PAYMENT' AND B.payment\_code = 'DEFAULT'  ORDER BY A.client\_id    SELECT PaymentsCSV.client\_id,COUNT(transaction\_id) AS PAYMENT\_REPEATED,SUM(PaymentsCSV.payment\_amt) AS TOTAL\_PAYMENT\_AMT, PaymentsCSV.payment\_code, ClientsCSV.entity\_type, ClientsCSV.entity\_year\_established  FROM     ClientsCSV RIGHT JOIN                    PaymentsCSV ON ClientsCSV.client\_id = PaymentsCSV.client\_id  WHERE PaymentsCSV.payment\_code = 'DEFAULT'  GROUP BY PaymentsCSV.client\_id,PaymentsCSV.payment\_code, ClientsCSV.entity\_type, ClientsCSV.entity\_year\_established |

In addition, the dataset seems that each row of column client\_id is only created when the client proceeds “PAYMENT” although some clients proceeds both “PAYMENT” and “DEFAULT”.

Step 3.3: I created two queries, in order, that ensure there is no row value of client\_id column if the initial payment\_code is “DEFAULT”:

|  |
| --- |
| CREATE VIEW PAYMENT\_REPEATED\_RESEARCH\_2 AS  SELECT A.client\_id,A.PAYMENT\_REPEATED as PAYMENT\_REPEATED\_1,A.TOTAL\_PAYMENT\_AMT AS TOTAL\_PAYMENT\_AMT\_1, A.payment\_code AS payment\_code\_1,B.payment\_code AS payment\_code\_2,B.PAYMENT\_REPEATED as PAYMENT\_REPEATED\_2,B.TOTAL\_PAYMENT\_AMT AS TOTAL\_PAYMENT\_AMT\_2, A.entity\_type, A.entity\_year\_established  FROM PAYMENT\_REPEATED\_RESEARCH A INNER JOIN PAYMENT\_REPEATED\_RESEARCH B ON A.client\_id = B.client\_id  WHERE A.payment\_code = 'DEFAULT' AND B.payment\_code = 'DEFAULT'  SELECT \*  FROM PAYMENT\_REPEATED\_RESEARCH\_2  WHERE PAYMENT\_REPEATED\_1 != PAYMENT\_REPEATED\_2 |

In conclusion, there is one value factor in the data which can help the business avoid default-prone customers in the future that **clients, who never proceed “DEFAULT”, are more likely to only proceed value “PAYMENT” in payment\_code column.**

**2. SQL Challenge (Use SQL for this section)**

The business would like to understand what the overlap is between 2018 payment totals for their biggest clients and the rank of the overall payment totals within each entity. Note defaults do not count as payments.

**Please provide a csv containing the client IDs for 2018's top 20 clients when sorted on payment amount, as well as the entity type, 2018's total payment amount, and their overall (across all years) payment amount rank within the entity they belong to. Also include your SQL code.**

*Answer:*

**2018's top 20 clients when sorted on payment amount, as well as the entity type, 2018's total payment amount.**

|  |
| --- |
| SELECT TOP 20 A.client\_id, A.payment\_code,B.entity\_type, SUM(payment\_amt) AS TOTAL\_PAYMENT  FROM PaymentsCSV A LEFT JOIN ClientsCSV B ON A.client\_id= B.client\_id  WHERE FORMAT(DATEADD(SECOND, transaction\_date, '19700101'),'yyyy-MM-dd') > '2017-12-31' AND '2019-01-01'> FORMAT(DATEADD(SECOND, transaction\_date, '19700101'),'yyyy-MM-dd') AND payment\_code = 'PAYMENT'  GROUP BY A.client\_id,A.payment\_code,B.entity\_type  ORDER BY TOTAL\_PAYMENT DESC |



**Overall (across all years) payment amount rank within the entity they belong to.**

|  |
| --- |
| SELECT RANK() OVER (ORDER BY SUM(payment\_amt) DESC) AS ENTITY\_RANK,B.entity\_type, SUM(payment\_amt) AS TOTAL\_PAYMENT  FROM PaymentsCSV A LEFT JOIN ClientsCSV B ON A.client\_id= B.client\_id  WHERE  A.payment\_code = 'PAYMENT'  GROUP BY B.entity\_type |



# PART 4 - Python

