Data Scientist
Kristina Kaliagina

Producing Insights for Clients

1. Attached file “insights.ipynb” and a PowerPoint presentation “Presentation” with the data analysis

2. Additional Metrics/insights:

   1. Using *LocationId* and *TxnDateTime* we can estimate the share of transactions per month by location (online and in-store). It can give us information about how digital is our client. Also we can look at average *spendout* online and in-store.

   2. Also using *LocationId*, spendout and *TxnDateTime* we can look at average daily *spendout* online and in-store to understand where customers tend to spend more money.

   3. Using *UserPostalSector*, *LocationId* and *TxnDateTime* we can estimate in which locations customers tend to buy online or in-store. It can help us to estimate in which areas we should increase the number of deliverers and where we could open new stores.

   4. Using *SpendIn* and *spendout* we can estimate the share of *SpendIn* in *spendout* (share = *SpendIn/spendout*). It helps to understand the quality of the stores

   5. Using *MerchantPostCode* and *spendout* we can estimate the profit in each location. It helps to understand where we are successful compared to our rivals or our stores in other locations.

   6. Using *UserType, spendout* and *TxnDateTime* it is worth estimating our focus on consumer/business, average *spendout* by consumer/business

   7. Using *Age/Income, TxnDateTime, spendout* we can estimate how many transactions are accumulated by different groups of *Age/Income* and what is the average *spendout* by different age/group. It helps to understand our target group of customers to our rivals or ourselves.

3. Additional fields:

   Txntype - how client paid for purchase (cash or card) - it can help to estimate the need for staff and self-service cash desks

   UserMaritalStatus - single, married, married and have children - helps to customize and give special offers

SpendInReason - why we refunded our client (bad quality, system error and etc) - to understand which area of business we can develop. Despite the lower average spendout lidl has Lower amount of transactions per client. Reason for refunding might demonstrate the weaknesses of Lidl.

SQL query

This SQL query first declares three variables *@row, @year* and *@quarter*. *@row* is initialized to 1, while *@year* and *@quarter* are initialized to null.

The next block of code creates a temporary table *#yearQuarterTable*, which contains three columns: Row, Year, and Quarter. The Row column is populated using the *ROW_NUMBER()* function over the Year and Quarter columns, ordered by Year and Quarter. The Year and Quarter columns are derived from the date column in *the MainTable*, using the *YEAR()* and *DATEPART()* functions, respectively. The table is filtered so that only records with a date in or after 2019 are included. The table is then grouped by Year and Quarter.

The next block of code selects a specific row from the *#yearQuarterTable*, based on the value of *@row,* and assigns the Year and Quarter values to *@year* and *@quarter,* respectively.

A temporary table *#merch_comp* is then created, which has four columns: merchant_parent, Year, Quarter, Spend, and Txns. This table will be used to store aggregated data about merchant spending and transactions.

The next block of code begins a *WHILE* loop, which will continue until *@year* is set to null. Within the loop, a temporary table *#merch_users* is created, which has a single column for *customerid*. This table is populated with *distinct customerids* from *the MainTable*, filtered to include only records where the transaction date is in the same Year and Quarter as the current iteration of the loop, where the *merchant_parent* is *'b365'*, the spend is negative, and *the country* is *'UK'*.

The next block of code populates the *#merch_comp* table with aggregated data from *the MainTable*. The table is filtered to include only records where the transaction date is in the same Year and Quarter as the current iteration of the loop, where the spend is negative, the *merchant_parent* is one of the five listed values, *the country* is 'UK', and the *customerid* is in the *#merch_users* table. The table is then grouped by Year, Quarter, and merchant_parent, and the *ABS()* function is used to convert the spend values to positive numbers.

Finally, the *#merch_users* table is dropped, and *@year* and *@quarter* are updated to the next row in the *#yearQuarterTable*. The loop will continue until all rows in *#yearQuarterTable* has been processed.

At the first iteration of the loop, the *#merch_users* table might be populated with *customerids* for transactions in *Q1 2019* where the *merchant_parent* is *'b365'* and the *spend* is *negative*. The *#merch_comp* table might be populated with aggregated spend and transaction data for *Q1 2019* for each of the five listed merchants, where the customerid is in the *#merch_users*

table. The loop would then move on to *Q2 2019*, and so on, until all year-quarter pairs in *#yearQuarterTable* has been processed.

Here are some examples of how the tables evolve:

#merch_users (example for year=2019, quarter=1):

customerid
1
2
…

#merch_comp (example for year=2019, quarter=1):

Merchant_parent	Year	Quarter	Spend	Txns
b365	2019	1	1000	10
the national lottery	2019	1	2000	20
sky bet	2019	1	1500	15
paddypower	2019	1	1200	12
william hill	2019	1	1800	18

The result table of this query would be the *#merch_comp* table, which contains aggregated spend and transaction data for each merchant and quarter. The query is aiming to identify which of the five listed merchants have the highest spend and/or transaction volume in each quarter, for transactions made.