Digital Marketing - Real Time Analytics

UNIVERSITY OF CALIFORNIA, LOS ANGELES

DATA MANAGEMENT – FINAL PROJECT 2023

GROUP 11

Calin Banciu (106091767)
Shangran Gao (906082537)
Zhiyi Liu (005870079)
Nikitha Sethumadhavan (106083102)
Anyao Wang (506082412)
Adia Yao (205949406)
Rateeb Yehya (805920023)
Nihong Yi (806080652)

Table of Contents

List of Figures

List of Tables

Acknowledgements

We would like to express our gratitude to Professor Franck Leveneur and TAs Janak Raj Chadha and Anirudh Chakravorty for their valuable guidance and support throughout the project. Their insights and feedback helped us refine our approach and overcome various challenges. We appreciate their time and effort in providing us with the necessary resources to succeed in the data management class. Thank you all for your guidance and encouragement. We accumulate a lot of first-hand practical experience through this project and understand the potential challenges we may face in the future, which will better prepare us for future careers. Kudos!

Executive Summary

In this project, our team explored the potential of real-time marketing (RTM) by simulating an RTM application on the SingleStore platform. Our aim was to ingest data generated from the simulation, conduct in-depth analytics, and create insightful visualizations using Tableau. Our team meticulously go through the technical and business details and creatively use the knowledge we have learned so far to tackle the data engineering issues and business analysis problems. Our main contributions can be summarized as follows：

First of all, we do our best to make sure that all the data transformations and calculations align with real business logic. For example, not only do we exclude purchase records happening earlier than notification time, but also make sure the purchases records are not duplicated. To achieve this, we creatively use ROW_NUMBER and RANK functions. This shows our understanding of both business scenarios and data engineering skills.

Secondly, we conduct a thorough analysis by designing KPIs that cover different angles, with both overviews and details. The visualizations are suitable for both CEOs making strategic decisions and business analysts doing daily work. The dashboards we created reveal a lot of critical information about not only the performance of the platform and of the vendors, but also about the way in which the subscribers are responding to the offers. Some of the more notable insights gained from these visualizations are the impressive percentage of subscribers that make a purchase after receiving a notification at nearly 15%, and the near 100% values for the subscriber retention rate and the vendor success rate, all of which leads us to believe that the platform is successful.

Lastly, we provide detailed feedback about the challenges we faced in the project and further research we can do based on our experience. This could be of great benefit not only for our own study purpose but also for the design of databases and simulation applications.

Project Statement

Real-time marketing (RTM) is a new approach that companies are adopting today to interact with their customers in real-time, based on data about how they engage with the company's product or service through multiple channels. It contrasts with traditional marketing approaches that require planning out a marketing strategy for months before implementation. Companies engage customers with short, personalized messages about product information, customer benefits, promotion details, etc., on their mobile phones, at the right time and location. This methodology often involves a sense of urgency, as customers realize that the advertisements they receive are tailored to current events or trends, and hence the offerings provided could be the most helpful in the present moment. As a result, real-time marketing is often a creative approach to attract more customers and boost engagement, leading to increased conversion rates at each stage of the sales funnel. To conduct effective real-time marketing, it is essential to develop a holistic and well-organized system of collection and processing of customer behavior data across multiple channels and try to respond to them over the channel they prefer.

In this project, we used the SingleStore platform to simulate an RTM application and conduct data analytics based on the real-time data we collect from the business process. For the business process, advertisements from multiple businesses (or vendors) are sent to customers via notifications on their cell phone, based on information about their behaviors and real-time locations. When more than one company's offers match a specific customer, there is a bidding process to decide on which company's offer to send as the final offer. The overview of the business process is shown below (Figure 1: An Overview of the Business Process Simulated on SingleStore Platform).

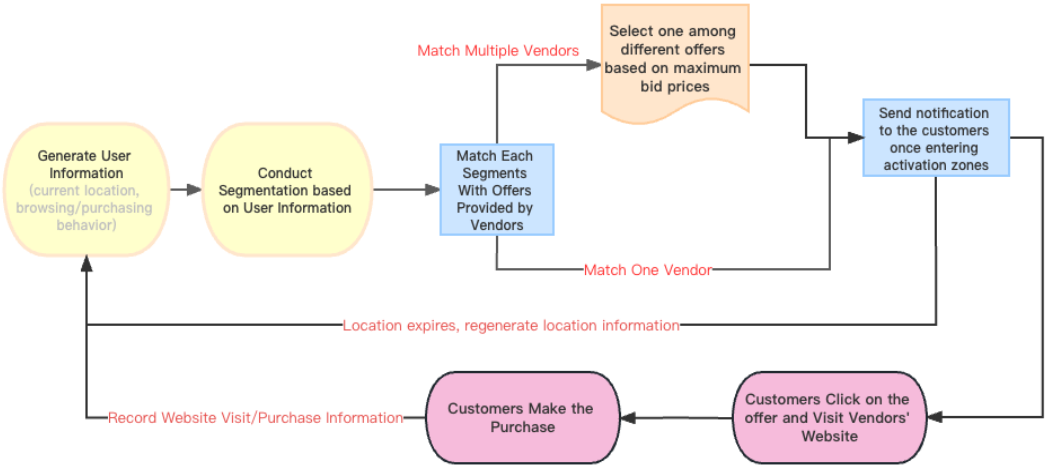

Figure 1: An Overview of the Business Process Simulated on SingleStore Platform

For each stage of the business process, we collected real-time data generated by using the MySQL client, restructured the database, and created KPIs accordingly to measure the performance of real-time digital marketing for vendors and subscribers. In the end, we created three dynamic dashboards using these KPI matrices for the purpose of tracking and analyzing business performance in the future for a larger scope of stakeholders.

Data Dictionary and Schema

In this project, we have developed a data dictionary to document the structure and content of the database used for The RTM simulation. The data dictionary provides a comprehensive description of the data attributes, its relevant entity, as well as the data type, maximum length, data types, constraints, and detailed description of the data's business understanding. For the ease of relating the database to the business process that the data was from, a series of renaming of the columns were conducted. The columns of the same data that was put in multiple relations sometimes were found to have quite distinctive naming (such as 'customer' from `offers` table and `vendor` column in the `purchase` table), these names were modified with such methodology that the names are consistent across the database. More information about the business domain and the originated table was added into the column name to make it more understandable. `Is_needed` column indicate it the data elements were eventually utilized for fact and dimensional table as well as KPI building. Overall, the data dictionary serves as a reference tool for developers, analysts, and other stakeholders to understand the data and its usage in the RTM applications. The data dictionary will also facilitate future maintenance and enhancement of the database by

providing a detailed description of its content and structure. You can find the data dictionary in Appendix: Table 3.

Moreover, we have created the relational schema of Martech database from SingleStore simulation (Figure 2: Relational Schema of Martech Database From SingleStore Simulation). These tables can be classified into different business domains: **customer information** and **segmentation & offer information** tell us about the business subjects; **real-time updates** and **business process** show us the dynamic process from identifying potential subscribers, matching subscribers with offers, ranking offers based on bids, pushing notifications to the subscribers, to finalizing purchases.

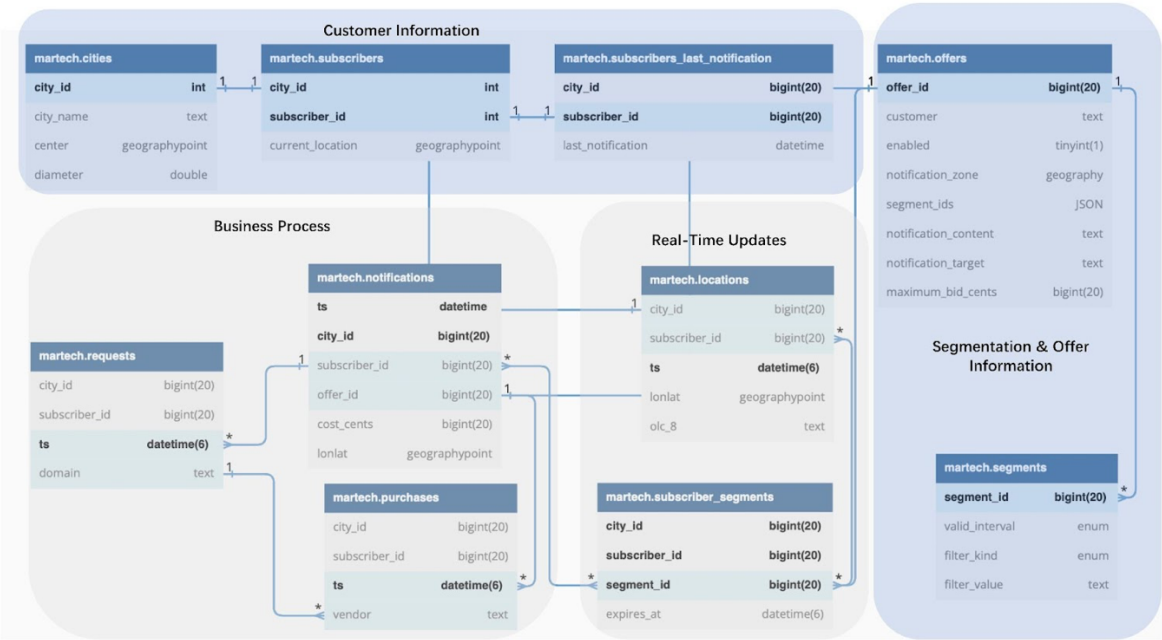

Figure 2: Relational Schema of Martech Database From SingleStore Simulation

Data Literacy

The data generated during the RTM simulation was stored in the Martech database in the SingleStore system. In total, there are 12 base tables (not including the tables for functions), as listed in Table 1, that were involved in the creation of the dim and fact tables, and the subsequent KPI building. The "usage" column provides information about the dim/fact tables that these tables were built upon. Additionally, the table shape information was provided to understand the data dimensions and volume obtained from the Martech database. SingleStore stores most tables in a distributed manner, where the rows in one table are stored across multiple distributions. This significantly decreases the time required for data querying and increases the flexibility of data storage. The "storage_type" column reflects the storage type to which a specific table belongs in the SingleStore database. There are three different categories: firstly, the in-memory rowstore storage for day-to-day data extraction;

secondly, the columnstore category for columnar data; and thirdly, the ObjectStore, which was not involved in this project.

For the use of data sources, we only chose a subset of the tables. Some of them don't have business meanings, such as the `session` table; some of them, such as `worldcities`, are not needed in our case since we only look at data from New York city. Hence we mainly make use of 6 tables to extract information and to create dimension and fact tables for later analysis. In the USAGE column, only underlined tables are the ones we use for our KPI analysis and visualization. We get dimension information of subscribers and offers, and fact measures of notifications and purchases from original tables. However, more could be done if we could tackle data integrity issues. The rest of the tables are for ideal use cases, including analyzing the dynamic bidding process of vendors and tracking subscribers' online journey and space movement.

Table 1: Main Data Source, Volume, and Usage

Table Name	Database	TABLE_TYPE	Distributed Table	STORAGE_TYPE	TABLE_ROWS	TABLE_COLUMNS	USAGE
segments	MARTECH	BASE TABLE	NO	INMEMORY_ROWSTORE	277	4	segment_dim
offers	MARTECH	BASE TABLE	NO	INMEMORY_ROWSTORE	200	8	offer_dim
subscribers_last_notification	MARTECH	BASE TABLE	YES	INMEMORY_ROWSTORE	45138	3	subscriber_dim
subscribers	MARTECH	BASE TABLE	YES	INMEMORY_ROWSTORE	50000	3	
notifications	MARTECH	BASE TABLE	YES	COLUMNSTORE	477897	6	notification_purchase_fact
purchases	MARTECH	BASE TABLE	YES	COLUMNSTORE	3301752	4	
subscriber_segments	MARTECH	BASE TABLE	YES	INMEMORY_ROWSTORE	1189052	4	bid_win_fact
requests	MARTECH	BASE TABLE	YES	COLUMNSTORE	11506396	4	click_through_fact
cities	MARTECH	BASE TABLE	NO	INMEMORY_ROWSTORE	1	4	geography analysis
locations	MARTECH	BASE TABLE	YES	COLUMNSTORE	8800000	5	
worldcities	MARTECH	BASE TABLE	YES	INMEMORY_ROWSTORE	128769	3	
sessions	MARTECH	BASE TABLE	YES	INMEMORY_ROWSTORE	1	3	

Data Processing and Architecture

We did data modeling to create dimension and fact tables, so that we can penetrate the business process from pushing the notification to finalizing a purchase. The modeling also gives us the maximum flexibility to analyze data from different granularities, including but not limited to subscribers, offers, and vendors. If we use aggregate sql queries, the dimensions we can use will be limited; nor can we dig deeper in Tableau. In best practice, deta engineering should always try to give business analysts the maximum flexibility especially under the condition of ELT.

To design reasonable dimensions and facts, we first do reverse engineering by observing and playing with data based on simple logic. With the understanding of business logic and key assumptions, we are able to identify the essential business dimensions and measures, which answer the following key questions: who is making the offers? Who is receiving the offers? Who is turning the offers received into purchases? When does every action mentioned above happen? Where does it happen? How much is the cost to make it happen?

With that in mind, we isolate the business entities and contain everything related to them, creating the dimension and fact tables as below (Figure 3)

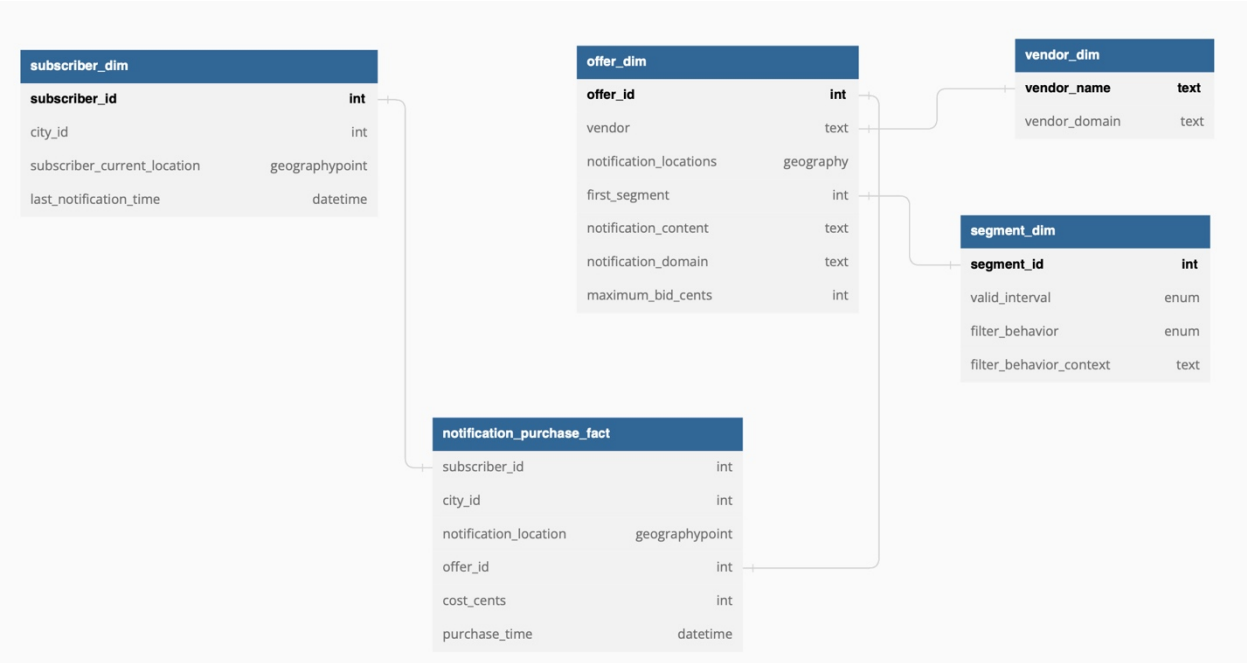

Figure 3: Fact and Dimension Tables

The new dimension and fact tables can be described as below:

- **Subscriber:**

Subscriber_dim contains all the information about the subscriber, including their id, location-related information, and last_notification time.

- **Offers:**

  Offers contains information about each offer's owner (vendor), offer's content, and the rule used to match with subscribers (segment_id). Most of the columns are directly from offers table, however, the segment_ids are wrapped in JSON, making it hard to match offers with segments. So we first used regular expressions to parse segment ids into two columns, then transformem the wide table into a long table.

- **Segments:**

  This table contains information about the rule used to match against subscribers, including the time interval, filter type, and specific filter content.

- **Vendors:**

  It contains information about the vendor and its domains. For some vendors, they can have more than one domain.

- **Fact table - Notification_purchase table:**

  This is the most important table for our analysis, which contains information about each notification and its follow-up purchase behavior. When a purchase happens for that notification, the value of purchase time is the actual time; otherwise, it's null. To create this table, we made some assumptions to build the connection between notifications and purchases table, since there is no unique identifier that tracks the subscriber behavior from a notification to a purchase. The assumptions are as follows：

  1. The subscriber_id, and vendor_name should be the same in both notifications and purchases.
  2. The notification_time should be earlier than the purchase time.
  3. Last but not least: each purchase time should only appear once. If we don't add this constraint, there could be a case where one purchase record is matched with two or more different notifications, which is counterfactual. For example, in the table below (Table 2), row 1 and row 2 are two purchase records from one single notification, while row 3 and row 4 are two notifications lead to the same purchase, which can't happen in real world and the second purchase time should be removed.

     We creatively used **rank** and **row_number** functions to eliminate the duplicate purchase records. That said, under this assumption it is possible that more than one purchase record is matched with the earlier notification record, leaving the later feasible

notification record with nothing. But on an aggregating level, it will not be a serious problem.

Table 2: purchase time processing

Row_number	Subscriber_id	Notification_time	Purchase_time
1	1	15:06	15:11
2	1	15:06	15:12
3	1	15:10	15:15
4	1	15:12	15:15

In best practice, we should keep notification and purchase tables as two fact tables; however, due to the difficulties we mention above, we manually linked them together to better conduct analysis.

The reasons we don't use some tables are as follows:

Locations: by checking data sanity, we find that for each subscriber at the same time, they can be at two different locations. The data sanity issue here prevents us from using it.

Subscriber_segments: this table is used to match the offers with available subscribers, which should be the key to tracking the bidding-winning process. However, because it doesn't have the timestamp for each snapshot, we can't really know who wins the bidding in each round (for the same offer, they can lose this round but win the next one).

Requests: requests table is helpful in tracking subscribers' behavior. They first send a request to let the app know that they can receive notifications, then send a request when they click the notification. However, because of lacking unique identifiers, we can't say which request is used to trigger the notification, and which request is sent from the notification (there could be multiple requests before and after receiving a notification). Also, it is hard to know if subscribers are clicking notifications or proactively sending requests.

Finally, the data modeling we did here is purely based on the KPIs chosen and the availability of data. Ideally, we should stick to STAR schema to isolate dimensions as clear as possible, to link fact tables with dimension tables via keys instead of natural keys, and to contain as much information as possible. We attached an ideal design in further research for reference.

KPIs, Visualizations & Insights

In this project, we have developed KPIs and visualizations to measure and analyze the performance of real-time digital marketing. The KPIs were developed based on the specific goals and objectives of the application. They reflect the effectiveness of the marketing strategies and tactics being used by the vendors and the engagement and response of the subscribers.

Knowing the importance of delivering relevant insights in a structured manner, one that resembles the narrative of a story, our team decided to split the visualizations in multiple dashboards, each with its own point of view on the performance of the company.

Figure 4: Visualization Overview

First, the user of our final product would be able to gauge the overall performance of the company by looking at the size of the subscriber base, and their reach aggregated by vendor and notification. Second, the user will have a chance to look at specific characteristics of the offers, such as notification frequency and purchase duration, which is shown at the vendor, segment, offer and subscriber levels.

Our third dashboard considers the effectiveness of the offers, as it looks into the average subscriber retention rate of the platform, as well as the subscriber conversion rate at the vendor, segment and offer levels. Finally, out last dashboard aims at allowing the user to appreciate the performance of the vendors the company is collaborating with, as well as a financial perspective of the performance of the company's offering. This is achieved by displaying the

vendor success rate and the offer acceptance rate for each vendor, as well as the average cost per purchase each vendor incurred.

Dashboard 1

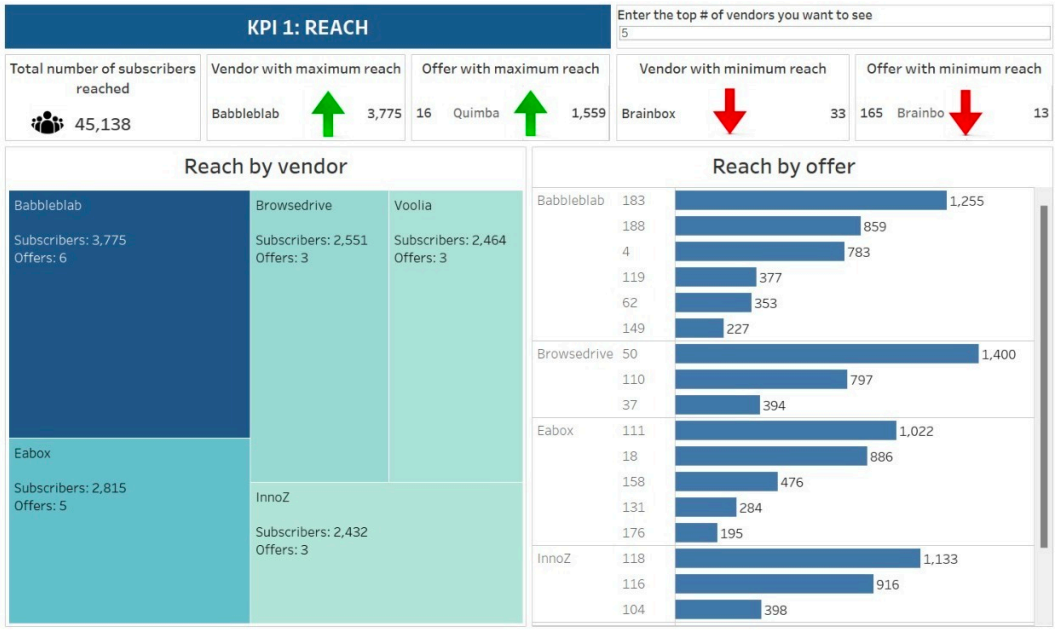

Figure 5: Dashboard 1

This is arguably the most important dashboard (Figure 5: Dashboard 1), as it shows metrics related to the reach of the platform. This is particularly relevant as it allows for the understanding of the overall size and popularity of the service, as well as the best and worst performing vendors and offers.

In the top row, we observe the total number of distinct subscribers that are active on the platform, the vendor and offer with the biggest audience, and the vendor and offer with the smallest one. An interesting thing to note is that the most successful vendor in terms of reach accounts for almost 10% of the total reach of the platform, while the least successful vendor accounts for less than 1% of the top vendor's reach. On the bottom left, the top n (n is specified by the user) vendors in terms of reach are displayed, showing the actual number of subscribers their audience consists of, as well as the number of offers each provide. Adjacent to that, we can see the distribution of reach for those top vendors by each individual offer.

Dashboard 2

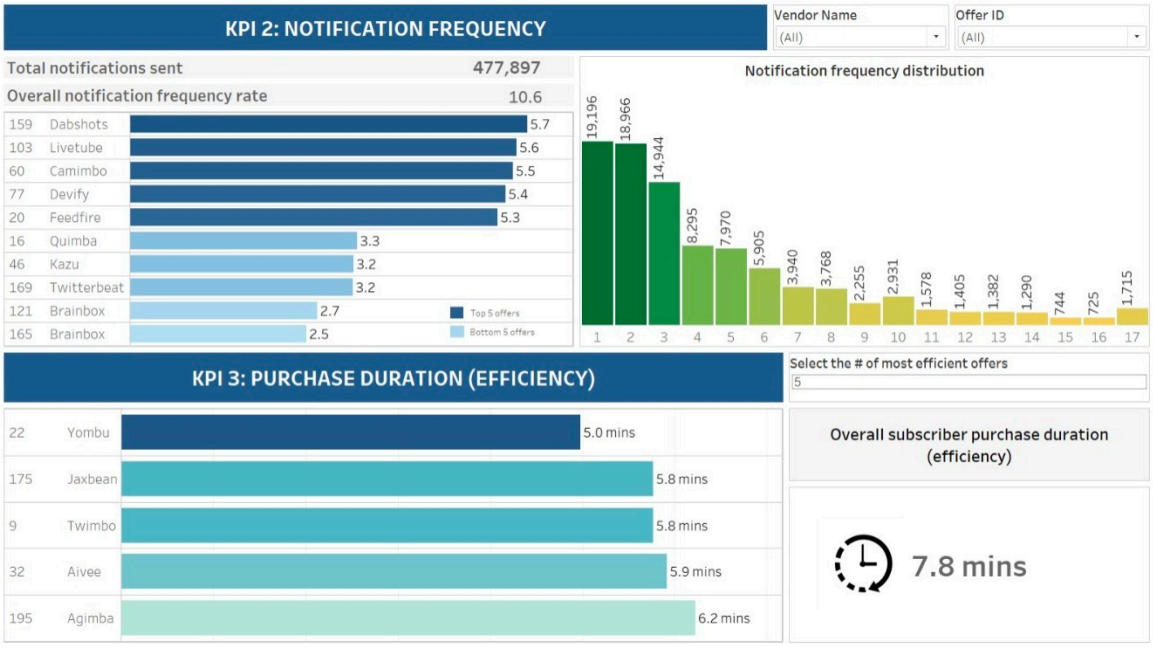

Figure 6: Dashboard 2

This dashboard (Figure 6: Dashboard 2) reveals key insights about the frequency and efficiency of the notifications on the platform. The notification frequency measures the average number of times a certain notification has been sent to a particular subscriber. The significance of this metric should not be understated, because ad fatigue, a phenomenon that occurs for customers that are shown a particular ad excessively, is significantly detrimental towards the effectiveness of the ad in question, and perhaps of the image or reputation of both the related vendor and the platform itself.

For this metric, we decided to not only show the 5 highest and lowest frequency offers, but also a histogram depicting the distribution of this variable. As we can see, most notifications only get shown 1 to 3 times. However, the prevalence of notifications that have been shown 17 times can be an alarming sign. In the bottom half of this dashboard, we displayed purchase efficiency, which measures the average time span between the moment the notification is received and the moment the purchase is made, if one happens. We assume this is relevant in gauging the effectiveness or popularity of any ad, as the lower the purchase duration is, the more enthusiastic we can presume the subscriber was when he made the purchase.

Dashboard 3

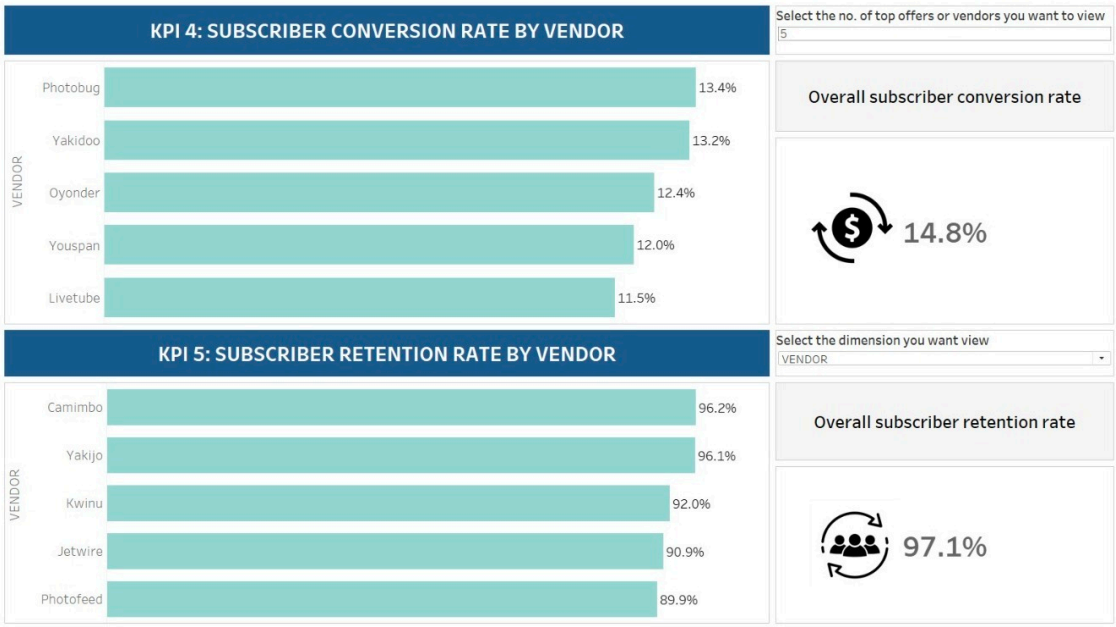

Figure 7: Dashboard 3

For our third dashboard (Figure 7: Dashboard 3) we decided to take a more in depth look at the metrics identifying the successful initiation and maintenance of a business relationship with a subscriber.

At first, we considered the subscriber conversion rate, which is the percentage of subscribers that make a purchase after receiving a notification. This can be analyzed at the offer, segment or vendor level, and we believe it is one of the most relevant measures in terms of the effectiveness of the ad. We also computed the overall rate for the platform, which at roughly 15% we believe is a good indicator of the efficacy of the platform's structure. The second relevant KPI we identified is the subscriber retention rate, which quantifies the percentage of subscribers who remain active, or keep accepting notifications, after they have received their first notification. An underlying assumption is the fact that subscribers receive or accept notifications based on their requests, and as such, the existence of at least two notifications denotes that they are in a sense "returning customers". The overall platform subscriber retention rate displayed in the bottom right, of 97.1%, is a very encouraging sign and could be indicative of the subscribers' appreciation towards the service.

Dashboard 4

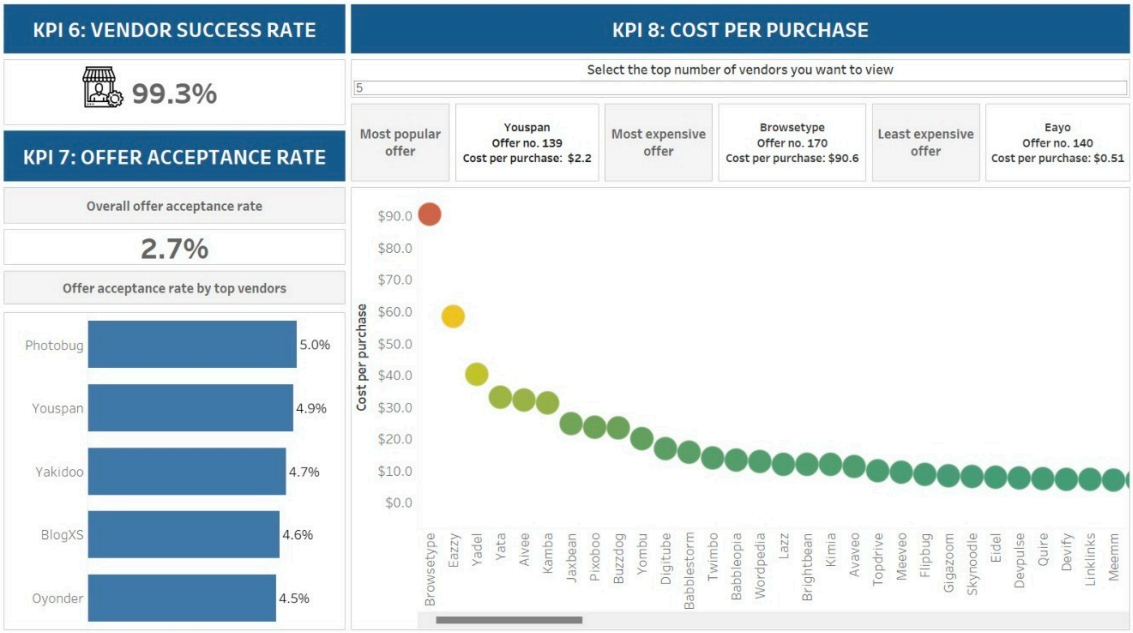

Figure 8: Dashboard 4

Last but not least, our fourth dashboard (Figure 8: Dashboard 4) is centered around the vendors and their performance on the platform.

In the top left, we see the vendor success rate, which quantifies the percentage of vendors that have had offers accepted, or were successful in achieving a sale by using the platform. This is extremely significant, as the near 100% value means that for almost all of the vendors, the use of the platform resulted in sales. Our next point of interest was understanding how efficient the vendors were in getting these sales. As such, we derived the offer acceptance rate, which is the percentage of offers that led to purchases for each vendor. Although the overall rate of 2.7% may seem small, we notice that for the more efficient vendors, 1 in every 20 offers results in a purchase. Our second metric aimed at understanding this efficiency is the cost per purchase, which measures the dollar amount each vendor incurred per purchase. This is achieved by computing how much each vendor spent for each of their offers and the amount of purchases each offer resulted in. A simple division of the two results in the metric of interest. This metric is displayed in such a way as to understand which of the vendors are the outliers in this regard. These values would be best used when compared to the net profits per unit of the vendors in an attempt to understand the value they are gaining from using the platform.

Suggestions to optimize reporting in Tableau

Despite the in-depth and wide-range analysis we have done in Tableau, we do believe that the reports can be improved if we have access to better data and more business feedbacks.

First of all, the dimensions of KPIs can be broadened. Currently we mainly look at data from the perspectives of subjects (vendors, subscribers), time duration (notification frequency, purchase duration), and monetization (cost per purchase), but we can also look at space dimension (Figure 9) and bidding process behind the scenes.

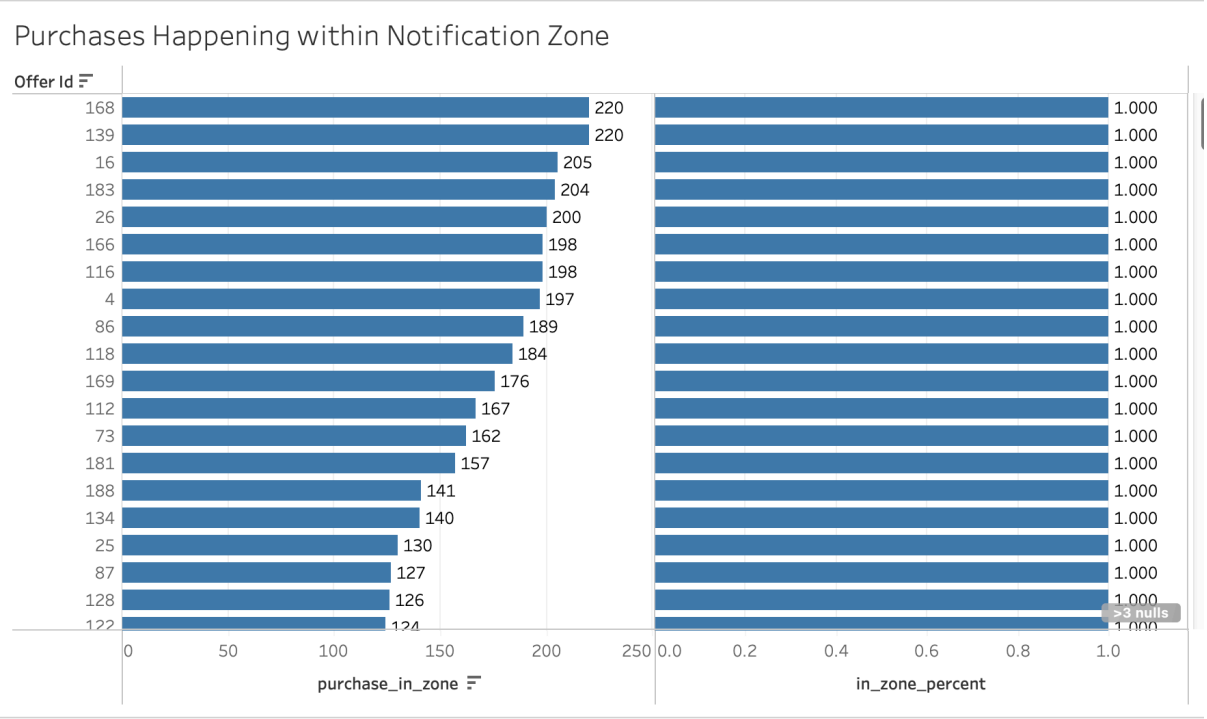

Figure 9: Location Analysis Example

Secondly, we can explore more types of plots to better fit each KPI. For instance, we can consider using candle stick charts to check the distribution of vendors' costs, which helps us quickly identify outliers (Figure 10).

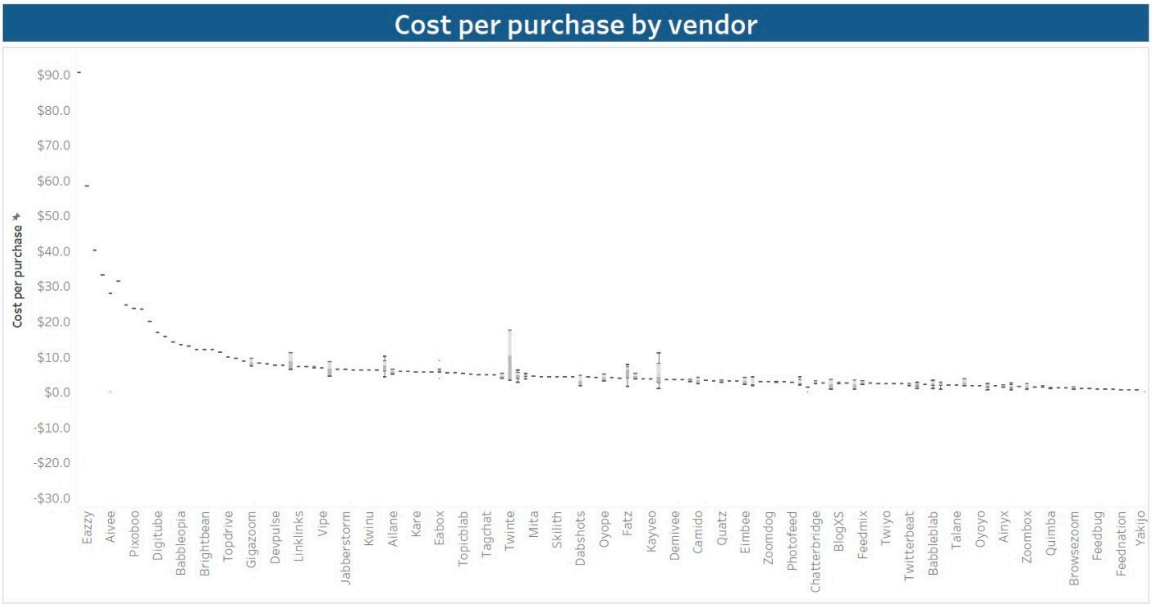

Figure 10: Candle Stick Chart Example

Project Challenges

We had a lot of challenges which can be categorized into three parts: Data Sanity, Data Integrity, and Technical Challenges.

Data Sanity

There is a lot of data in the purchase table that is not usable because it happens before the notification is pushed. Therefore, we filtered those purchases out before proceeding with the analysis. Also, in the `subscriber_segments` table, a lot of the `segment_id`s are not existent in the `offers` table, making it hard to count the total bids placed by vendors.

Another challenge we had in data sanity is that for a certain subscriber at a certain timestamp, we had two locations (Figure 11). This does not make sense unless we live in a quantum world. As a result, we decided to drop the whole table.

city_id	subscriber_id	ts	lonlat	olc_8
120658	1	2023-03-15 06:03:05.051292	POINT(-74.00348203 40.71286492)	87G7PX7W
120658	1	2023-03-15 06:03:05.051292	POINT(-74.00346974 40.71284983)	87G7PX7W

Figure 11: Different Locations for the Same Subscriber at One Time Point

Data integrity

The first thing we notice is that there is not always a clear foreign key in each table. If we want to join some tables together, we must preprocess the data or create some assumptions. As we mentioned earlier, we had to parse the `segment_ids` in the `offers` table to join it with `segments` table.

offer_id	customer	enabled	notification_zone	segment_ids	notification_content	notification_target
1	Skynoodle	1	POLYGON((-73.99500000 40.73500...	[1626570934,3566845242]	35% off at Skynoodle	skynoodle.gov
2	Yakidoo	1	POLYGON((-73.97500000 40.73000...	[2889025648]	38% off at Yakidoo	yakidoo.gov
3	Npath	1	POLYGON((-74.01500000 40.73500...	[3867057678,2675790737]	49% off at Npath	npath.name
4	Babbleblab	1	POLYGON((-74.00250000 40.71000...	[3814858283,2485076460]	41% off at Babbleblab	babbleblab.net
5	Ainyx	1	POLYGON((-73.98000000 40.73500...	[1865401059,4090435187]	20% off at Ainyx	ainyx.info
6	Kwideo	1	POLYGON((-73.99000000 40.71000...	[3027604494,3251695081]	42% off at Kwideo	kwideo.biz

Figure 12: original offer table - segment_ids in JSON

offer_id	vendor	notification_locations	notification_content	notification_domain	maximum_bid_cents	segment_id
1	Skynoodle	POLYGON((-73.99500000 40.73500...	35% off at Skynoodle	skynoodle.gov	10	1626570934
2	Yakidoo	POLYGON((-73.97500000 40.73000...	38% off at Yakidoo	yakidoo.gov	7	2889025648
3	Npath	POLYGON((-74.01500000 40.73500...	49% off at Npath	npath.name	8	3867057678
4	Babbleblab	POLYGON((-74.00250000 40.71000...	41% off at Babbleblab	babbleblab.net	12	3814858283
5	Ainyx	POLYGON((-73.98000000 40.73500...	20% off at Ainyx	ainyx.info	3	1865401059
6	Kwideo	POLYGON((-73.99000000 40.71000...	42% off at Kwideo	kwideo.biz	5	3027604494

Figure 13: offer_dim table - segment_id in long table format

Also, the schema misses some key tables or key columns in the existing tables (Figure 14), making it almost impossible to extract useful information from certain dimensions. Especially for the missing table issue, we lack a table to store the data from the bidding process. When a subscriber is matched with an offer from a vendor, our table only shows the winning offer (offer id), without information about the bids that were placed. Therefore, we cannot reliably calculate the bid winning rate without making assumptions. Also, we cannot dig deep into analysis at the bidding level.

city_id	subscriber_id	segment_id	expires_at	Missing: snapshot time
120658	1	256872119	2023-03-16 06:19:30.041554	
120658	1	617497744	2023-03-22 06:12:00.150642	
120658	1	843098579	2023-03-15 07:20:59.451324	
120658	1	1203318488	2023-03-22 06:17:23.862183	
120658	1	1232694629	2023-03-22 06:11:36.650259	
120658	1	1375950156	2023-04-15 06:19:01.161567	
120658	1	1583228190	2023-03-22 06:15:19.360488	
120658	1	2602662569	2023-03-15 07:21:57.542527	
120658	1	2889025648	2023-03-15 07:11:05.147922	
120658	1	2890117580	2023-03-22 06:11:05.147922	

Figure 14: Segments Table Missing Snapshot Time

Last but not least, the schema lacks unique identifiers that tracks subscribers' online journey. To calculate the Click-Through rate, we need to know which request prompts the notification, and which request leads to a purchase. In digital marketing world, fully understanding subscribers behaviors is the key to maximize the effect of campaigns.

Technical Challenges

Firstly, Tableau does not correctly capture the dates in the notification table. When we calculate the time difference between purchase and notification, some values turn out to be negative. To solve this challenge, we had to first format the cells on excel before connecting to Tableau.

Secondly, the time it takes Tableau to calculate the time difference between notification and purchase time is pretty long. It would be unrealistic if we want to streamline the analysis pipeline and conduct analysis on other big-data scenarios.

Further Research

Long Text Data Processing and Analysis

"Notification_content" is a great source of information about the specific promotion strategy provided by each vendor. Such strategies might include the discount they offer, the wording they use, the specific products they mentioned etc. These tactics are presumably impactful to the website-visiting as well "as purchase" decisions made by the users. All these

data points could be collected with natural language processing algorithms like Word2Vec for further analysis.

Streaming ETL

In this project, we ingested the data using SingleStore and then stopped the ingestion after we reached 2 GB of Disk Space. We then did all the necessary transformations and built out our fact and dimension tables, then extracted the static data to Tableau for analysis and visualization. However, from a business perspective, real time analytics is more useful to executives so that they can make decisions in a dynamic manner. Streamling ETL is process of moving real time data from one place to another, analyzing and visualizing data in real-time. This is an area we would love to explore for further project development.

Ideal data modelling schema

Based on our analysis, we have created an ideal data modelling schema that would suit our business objectives well (Figure 15: Ideal Data Modelling Schema).

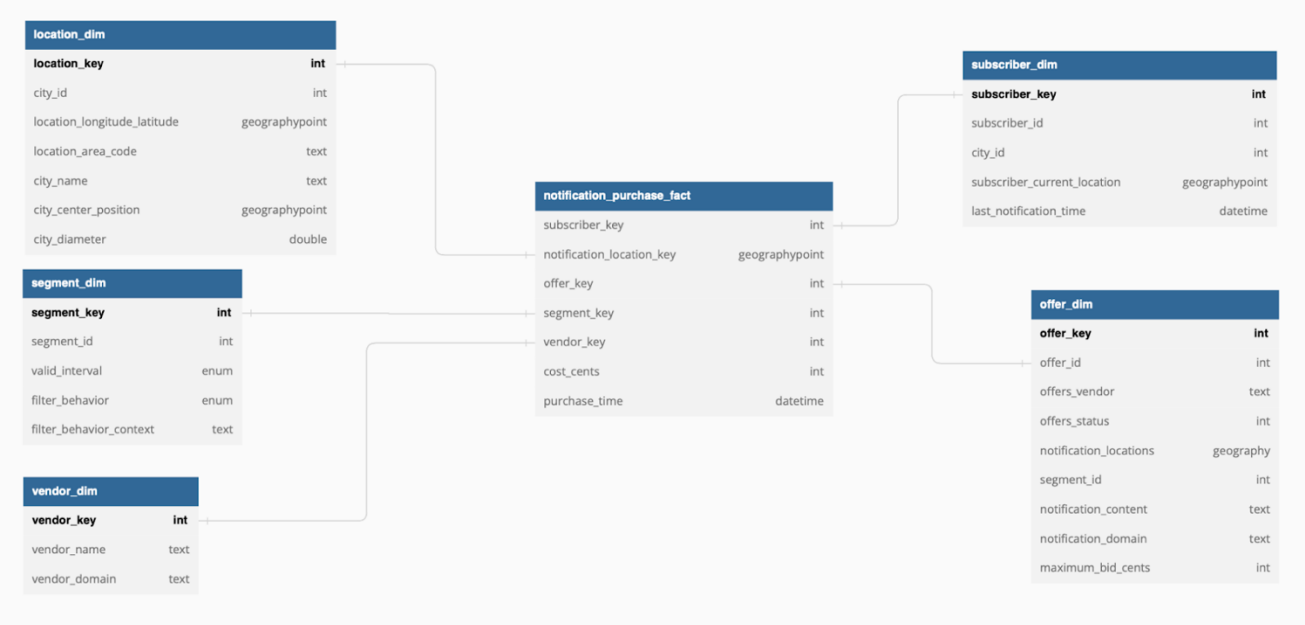

Figure 15: Ideal Data Modelling Schema

Appendix

Data Dictionary

Table 3: Data Dictionary

TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	DATA_TYPE	Rename To	ORDINAL_POSITION	DOMAIN_LENGTH	MAXIMUM_LENGTH	NUMERIC_SCALE	IS_NULLABLE	Is_Needed	Description
martech	cities	city_id	BIGINT		1	6	20	0	NO	1	unique city identifier
martech	cities	city_name	TEXT		2	8	21845	null	NO	1	the name of the city
martech	cities	center	geographypoint	city_center_position	3	null	nul	null	NO	1	the longitude and latitude of the center of the city
martech	cities	diameter	DOUBLE	city_diameter	4	3	22	null	YES	1	
martech	locations	city_id	BIGINT	location_city_id	1	6	20	0	NO	1	unique city identifier
martech	locations	subscriber_id	BIGINT	location_subscriber_id	2	5	20	0	NO	1	unique subscriber identifier
martech	locations	ts	DATETIME	location_time	3	null	null	null	NO	1	timestamp
martech	locations	lonlat	geographypoint	location_longitude_latitude	4	null	null	null	NO	1	the longitude and latitude of the location

martech	locations	olc_8	TEXT	location_area_code	5	8	21845	null	NO	1	area code
martech	offers	offer_id	BIGINT		1	5	20	0	NO	1	unique offer identifier
martech	offers	customer	TEXT	vendor	2	20	21845	null	NO	1	vendor who sent out the offer
martech	offers	enabled	TINYINT	offers_status	3	1	1	0	NO	1	if the offers are activated or not
martech	offers	notification_zone	geography	notification_locations	4	null	null	null	NO	1	a collection of the location details where the offers would be sent to the subscribers via notifications
martech	offers	segment_ids	JSON		5	null	null	null	NO	1	customer segments targeted by this specific notification
martech	offers	notification_content	TEXT		6	100	21845	null	NO	1	the written content contained

											within the notification
martech	offers	notification_target	TEXT	notification_domain	7	50	21845	null	NO	1	the name of the domain which gives notifications
martech	offers	maximum_bid_cents	BIGINT		8	2	20	0	NO	1	the maximum bid price in cents of a notification
martech	segments	segment_id	BIGINT		1	5	20	0	NO	1	unique customer segment identifier
martech	segments	valid_interval	ENUM('minute','hour','day','week','month')		2	6	6	null	NO	1	the time since the customer behavior occurred (which we used to identify customer segmentation) occurred
martech	segments	filter_kind	ENUM('olc_8','request','purchase')	filter_behavior	3	8	8	null	NO	1	the kind of behavior the

											customer made (within 'olc_8' <location>,'request' <visit the website> or 'purchase' <make the purchase>)
martech	segments	filter_value	TEXT	filter_behavior_context	4	50	21845	null	NO	1	the specific information of the customer behavior (<what location they went>, <which domain/website they visited>, <which product they purchased>)
martech	subscribers	city_id	BIGINT		1	6	20	0	NO	1	unique city identifier

martech	subscribers	subscriber_id	BIGINT		2	5	20	0	NO	1	unique subscriber identifier
martech	subscribers	current_location	geographypoint	subscriber_current_location	3	null	null	null	NO	1	the longitude and latitude of the current location
martech	subscriber_segments	city_id	BIGINT	subscriber_segmentation_city_id	1	6	20	0	NO	1	unique city identifier
martech	subscriber_segments	subscriber_id	BIGINT		2	5	20	0	NO	1	unique subscriber identifier
martech	subscriber_segments	segment_id	BIGINT		3	5	20	0	NO	1	unique segment identifier
martech	subscriber_segments	expires_at	DATETIME	subscriber_segmentation_expire_time	null	null	null	0	YES	1	the time when the customer is no longer inside one specific segment
martech	subscribers_last_notification	city_id	BIGINT		1	6	20	0	NO	1	unique city identifier
martech	subscribers_last_notification	subscriber_id	BIGINT		2	5	20	0	NO	1	unique subscriber identifier

martech	subscribers_last_notification	last_notification	DATETIME	last_notification_time	3	null	null	null	YES	1	the time when the subscriber received the last notification
martech	notifications	ts	DATETIME	notification_time	1	null	null	null	NO	1	the time when the subscriber received a notification
martech	notifications	city_id	BIGINT		1	6	20	0	NO	1	unique city identifier
martech	notifications	subscriber_id	BIGINT		1	5	20	0	NO	1	unique subscriber identifier
martech	notifications	offer_id	BIGINT		1	5	20	0	NO	1	unique offer identifier
martech	notifications	cost_cents	BIGINT		1	2	20	0	NO	1	the amount paid by a domain for this specific notification
martech	notifications	lonlat	geographypoint	notification_location	1	null	null	null	NO	1	the geographical position of the subscriber when

											receiving this notification
martech	requests	city_id	BIGINT	request_city_id	1	6	20	0	NO	1	unique city identifier
martech	requests	subscriber_id	BIGINT	request_subscriber_id	1	5	20	0	NO	1	unique subscriber identifier
martech	requests	ts	DATETIME	request_time	1	32	20	0	NO	1	the time when the subscriber visited the domain
martech	requests	domain	TEXT	request_domain	1	50	21845	null	NO	1	the name of the domain
martech	purchases	city_id	BIGINT		1	6	20	0	NO	1	unique city identifier
martech	purchases	subscriber_id	BIGINT		1	5	20	0	NO	1	unique subscriber identifier
martech	purchases	ts	DATETIME	purchase_time	1	null	null	null	NO	1	the time when the subscriber made the purchase
martech	purchases	vendor	TEXT	vendor_name	1	50	21845	null	NO	1	the vendor which the purchase was made from

martech	dynamic_subscriber_segments	city_id	bigint		1		null	0	NO	0	
martech	dynamic_subscriber_segments	subscriber_id	bigint		2		null	0	NO	0	
martech	dynamic_subscriber_segments	segment_id	bigint		3		null	0	NO	0	
martech	dynamic_subscriber_segments	expires_at	datetime		4		null	NULL	YES	0	
martech	dynamic_subscriber_segments_locations	city_id	bigint		1		null	0	NO	0	
martech	dynamic_subscriber_segments_locations	subscriber_id	bigint		2		null	0	NO	0	
martech	dynamic_subscriber_segments_locations	segment_id	bigint		3		null	0	NO	0	
martech	dynamic_subscriber_segments_locations	expires_at	datetime		4		null	NULL	YES	0	
martech	dynamic_subscriber_segments_purchases	city_id	bigint		1		null	0	NO	0	
martech	dynamic_subscriber_segments_purchases	subscriber_id	bigint		2		null	0	NO	0	
martech	dynamic_subscriber_segments_purchases	segment_id	bigint		3		null	0	NO	0	
martech	dynamic_subscriber_segments_purchases	expires_at	datetime		4		null	NULL	YES	0	
martech	dynamic_subscriber_segments_requests	city_id	bigint		1		null	0	NO	0	
martech	dynamic_subscriber_segments_requests	subscriber_id	bigint		2		null	0	NO	0	
martech	dynamic_subscriber_segments_requests	segment_id	bigint		3		null	0	NO	0	

martech	dynamic_subscriber_segments_requests	expires_at	datetime		4		null	NULL	YES	0	
martech	match_offers_to_subscribers	city_id	bigint		1		null	0	NO	0	
martech	match_offers_to_subscribers	subscriber_id	bigint		2		null	0	NO	0	
martech	match_offers_to_subscribers	best_offer_id	bigint		3		null	0	YES	0	
martech	match_offers_to_subscribers	cost_cents	bigint		4		null	0	YES	0	
martech	match_offers_to_subscribers	current_location	geographypoint		5		null	NULL	NO	0	
martech	sessions	session_id	text		1		21845	NULL	NO	0	
martech	sessions	is_controller	tinyint		2		NULL	0	NO	0	
martech	sessions	expires_at	datetime		3		NULL	NULL	NO	0	

Connection from MySQL client to SingleStore

Object Info	Session
Connection Details	
Name:	real_time
Host:	svc-904220e8-5a51-49dd-b8e6-798dfc4104cb-dml.aws-virginia-5.svc.singlestore.com
Port:	3306
Login User:	admin
Current User:	admin@%
SSL cipher:	AES256-SHA

Figure 16: Connection from MySQL to Singlestore

SQL code

```
-- subscriber_dim
SELECT
s.subscriber_id,
s.city_id,
s.current_location as subscriber_current_location,
sln.last_notification as last_notification_time
FROM
subscribers s
JOIN subscribers_last_notification sln
ON s.city_id = sln.city_id and s.subscriber_id = sln.subscriber_id
order by s.subscriber_id;


-- OFFER_DIM
-- first create a view in singlestore, then select * in tableau

create view offer_dim as
select
	offer_id, customer as vendor, notification_zone as notification_locations, notification_content, notification_target as
notification_domain, maximum_bid_cents,
	cast(REPLACE(REPLACE(SUBSTRING_INDEX(segment_ids,',',1),'[',''),']','') as unsigned) as first_segment
from offers
UNION ALL
select
```

```
    offer_id, customer as vendor, notification_zone as notification_locations, notification_content, notification_target as
notification_domain, maximum_bid_cents,
    cast(case when REPLACE(REPLACE(SUBSTRING_INDEX(segment_ids,',',-1),'[',''),']','') =
REPLACE(REPLACE(SUBSTRING_INDEX(segment_ids,',',1),'[',''),']','')
    then null else REPLACE(REPLACE(SUBSTRING_INDEX(segment_ids,',',-1),'[',''),']','') end as unsigned) as
first_segment
from offers
where segment_id is not null;

select * from offer_dim;


-- SEGMENT_DIM
select
segment_id,
valid_interval,
filter_kind as filter_behavior,
filter_value as filter_behavior_context
from segments;


-- vendor_dim

select distinct
customer as vendor_name,
notification_target as vendor_domain
from offers;

-- notification_purchase fact

select distinct
a.subscriber_id,
a.city_id,
a.lonlat as notification_location,
a.nts as notification_time,
a.offer_id,
a.customer as vendor_name,
a.cost_cents,
a.real_time as purchase_time
from
(select
n.subscriber_id,
n.city_id,
n.lonlat,
```

```
n.nts,
n.offer_id,
n.customer,
n.cost_cents,
n.pts,
n.rk,
n.rn,
case when pts is null then pts when pts is not null and n.rk = n.rn then pts else null end as real_time
from (
select
n.subscriber_id,
n.city_id,
n.lonlat,
n.ts as nts,
o.offer_id as offer_id,
o.customer,
n.cost_cents as cost_cents,
p.ts as pts,
rank() over (partition by n.subscriber_id order by p.ts) as rk,
row_number() over (partition by n.subscriber_id order by p.ts) as rn
from notifications n
join offers o
on n.offer_id = o.offer_id
left join purchases p
on n.subscriber_id = p.subscriber_id
and n.city_id = p.city_id
and o.customer = p.vendor
and n.ts < p.ts) n) a
```

KPI Calculation

Reach KPI Calculation

COUNTD(Subscriber ID)

Subscriber Conversion Rate KPI Calculation

- Create a calculated field called is_notified
  - If not isnull(notification_time) then subscriber_id
- Create a calculated field called is_purchased
  - If not isnull(purchase_time) then subscriber_id
- Create a calculated field called subscriber_conversiuon_rate

    - countd(is_notified) / countd(is_purchased)
- Then we can use this to aggregate by vendor, segment, and subscriber

Frequency KPI Calculation

- Assume each offer can have multiple notifications sent to multiple subscribers
- Create a calculated field called notification_per_sub_offer
    - {FIXED [Subscriber Id], [Offer Id]:COUNTD([Notification Time])}
- Create a calculated field called Frequency
    - SUM([notification_per_sub_offer])/COUNTD([Subscriber Id])
- Create a calculated field called purchase_conversion_rate
    - SUM([no_purchase])/SUM([notification_per_sub_offer])
- We interpret the frequency as 10.59 which means the average times of a subscriber is shown a notification is 10.59
- Then we can aggregate by offer id with dynamic setting, which allows to filter top offer ids with highest frequency
- We presented two Frequency histograms with one showing the overall frequency histogram, and the other showing customized frequency histogram filtered by vendor and offer id.
- Create a calculated field called FreqBins
    - IF [notification_per_sub_offer] == 1 THEN "1"
      ELSEIF [notification_per_sub_offer] == 2 THEN "2"
      ELSEIF [notification_per_sub_offer] == 3 THEN "3"
      ELSEIF [notification_per_sub_offer] == 4 THEN "4"
      ELSEIF [notification_per_sub_offer] == 5 THEN "5"
      ELSEIF [notification_per_sub_offer] == 6 THEN "6"
      ELSEIF [notification_per_sub_offer] == 7 THEN "7"
      ELSEIF [notification_per_sub_offer] == 8 THEN "8"
      ELSEIF [notification_per_sub_offer] == 9 THEN "9"
      ELSEIF [notification_per_sub_offer] == 10 THEN "10"
      ELSEIF [notification_per_sub_offer] == 11 THEN "11"
      ELSEIF [notification_per_sub_offer] == 12 THEN "12"
      ELSEIF [notification_per_sub_offer] == 13 THEN "13"
      ELSEIF [notification_per_sub_offer] == 14 THEN "14"
      ELSEIF [notification_per_sub_offer] == 15 THEN "15"
      ELSEIF [notification_per_sub_offer] == 16 THEN "16"
      ELSE "17"
      END

Subscriber Retention Rate KPI Calculation

- Assumption: assume subscribers receiving two or more notifications are considered to be retained

- This is a conservative way of calculating subscriber retention rate because the number of requests is always larger than the number of notifications.
- Create a calculated field called retained_subscribers
    - IF [notification_per_sub_offer]>=2

THEN [Subscriber Id]
ELSE 0
END

- Create a calculated field called subscriber retention rate
    - COUNTD([retained_subscribers])/COUNTD([Subscriber Id])
- The value is around 97.13%
- Then we can use this to aggregate by vendors with dynamic setting, which allows to filter top N vendors with highest subscribe retention rates

Efficiency

- Assumption: assume requests are all made after the offer/notification
- Create a calculated field called no_notification_per_sub_offer
    - {FIXED [Subscriber Id], [Offer Id]:COUNTD([Notification Time])}
    - Holding subscriber_id and offer_id, aggregate the distinct notification_time
    - This gives the number of notifications sent
- Display cost cents, no_notification_per_sub_offer, no_purchase by vendor name and offer id, so we can see how much was spent for each offer by the vendor (this is done by costs cents times number of notifications sent)

Cost Per Click

- Assumption: assume requests are all made after the offer/notification
- Create a calculated field called no_notification_per_sub_offer
    - {FIXED [Subscriber Id], [Offer Id]:COUNTD([Notification Time])}
    - Holding subscriber_id and offer_id, aggregate the distinct notification_time
    - This gives the number of notifications sent
- Display cost cents, no_(notification_(per_(sub_offer))), no_purchase by vendor name and offer id, so we can see how much was spent for each offer by the vendor (this is done by costs cents times number of notifications sent)
- Then we can also create a calculated field called costs_per_purchase
    - (SUM([Cost Cents])/SUM([no_purchase]))/100
    - This is done by getting the sum of cost cents (total money spent per offer to send notifications) divided by no_purchase
    - This gives us how much is spent to have one purchase

Purchase Conversion Rate

- Assume each offer can have multiple notifications sent to multiple subscribers
- Create a calculated field called no_purchase
    - IF [Purchase Time] == "NULL" THEN 0 ELSE 1 END

- Create a calculated field called notification_per_sub_offer
    - {FIXED [Subscriber Id], [Offer Id]:COUNTD([Notification Time])}
- Create a calculated field called purchase_conversion_rate
    - SUM([no_purchase])/SUM([notification_per_sub_offer])
- Then we can use this to aggregate by vendor and offer id
- We interpret the first row as the purchase rate of offer id 81 sent from vendor Abata is 0.02074 with 39 number of purchases and 1880 number of notifications per subscriber per offer

Purchase duration/ efficiency: Time between the notification and the purchase

- Applied filter to only consider those observations which contain purchase time
- Calculated field "Time difference" in minutes as the purchase time - notification time
- Presenting purchase efficiency by offer / vendor