Customer Subscription Management System

DS5110 - Project Report

Ashish Magadum, Varsha Ranganathan, Vignesh Sankar

Introduction

The Customer Subscription Management System Database stores important information about the customers who hold a subscription from a company such as their subscription details, revenue generated, feedback, activities with the product, and the marketing campaigns that were run for those customers by the company.
The main objective of this Project is to help Marketing Stakeholders of the companies run simple queries on the Database, get the trends of customers' journey with the Product, and come up with strategic business decisions that can increase user engagement and thereby the total annual revenue of the company. Some examples of companies that operate primarily on a subscription-based business are Adobe, Microsoft, etc.

Database Design

- A user can hold multiple subscriptions and each subscription would generate a revenue in USD.
- We keep a track of the Marketing emails that are sent to users to keep them engaged with the subscriptions of the company.
- Depending on the user's interaction with the product, we also track the engagement index for each user.
- A user is meant to be less engaged if his engagement index is less than 0.25 and vice versa.
- A user could have performed activities with the product like Install, Download, and Update on different dates, but every activity is unique to a user.
- Users could receive one or many updates on the products, and every update is unique to a user.
- We also store the feedback and the preferences that each of the users have in separate tables.

From the ERD, by converting the entities to Relations, we get the below relations

1. USER (User_id, user_name, region, Email_id, age, category, gender)
2. SUBSCRIPTION (User_id, Subscription_id, Product_id, Product_name, Channel_id, Channel_name, Start_date, End_date)
3. REVENUE (Revenue_id, Subscription_id, revenue_type, Gross_arr_in_usd)
4. ENGAGEMENT (user_id, user_type, engagement_index) – Weak Entity
5. ACTIVITIES (Activity_id, user_id, activity_type, activity_date)
6. PRODUCT_UPDATES (Update_id, User_id, Update_name)
7. CAMPAIGN_TOUCH (Campaign_id, User_id, Campaign_name)
8. FEEDBACK (feedback_id, user_id, feedback_date, Feedback)
9. PREFERENCES (preference_id, user_id, preference_name)
10. TOUCHED_BY (User_id, Campaign_id, Touch_date)
11. PREFERS (User_id, Preference_id)

The above relations comes directly from the ERD by transforming each of the entities to a relation. Since there exists a Many to Many relation between **PREFERENCES - USER** and **CAMPAIGN_TOUCH - USER**, we take the relationship between the entities and add the primary keys of both the tables to the identifying relationship.

We can observe that the above relations are not in 3NF. It is necessary to decompose the relations into 3NF to improve readability and reduce redundancy in the database.

3NF Decompositions:

The above relations are already in 2NF because there is no partial functional dependency between any of the non-prime attributes on any proper subset of the primary key. However, they are not in 3NF because, in the **Subscription Table**, there exists a transitive dependency between `Subscription_id`, `Product_id`, and `Product_name` such that `Subscription_id` $\rightarrow$ `Product_id` $\rightarrow$ `Product_name`. Similarly, there exists a transitive dependency between `Subscription_id`, `Channel_id`, and `Channel_name` such that `Subscription_id` $\rightarrow$ `Channel_id` $\rightarrow$ `Channel_name`.

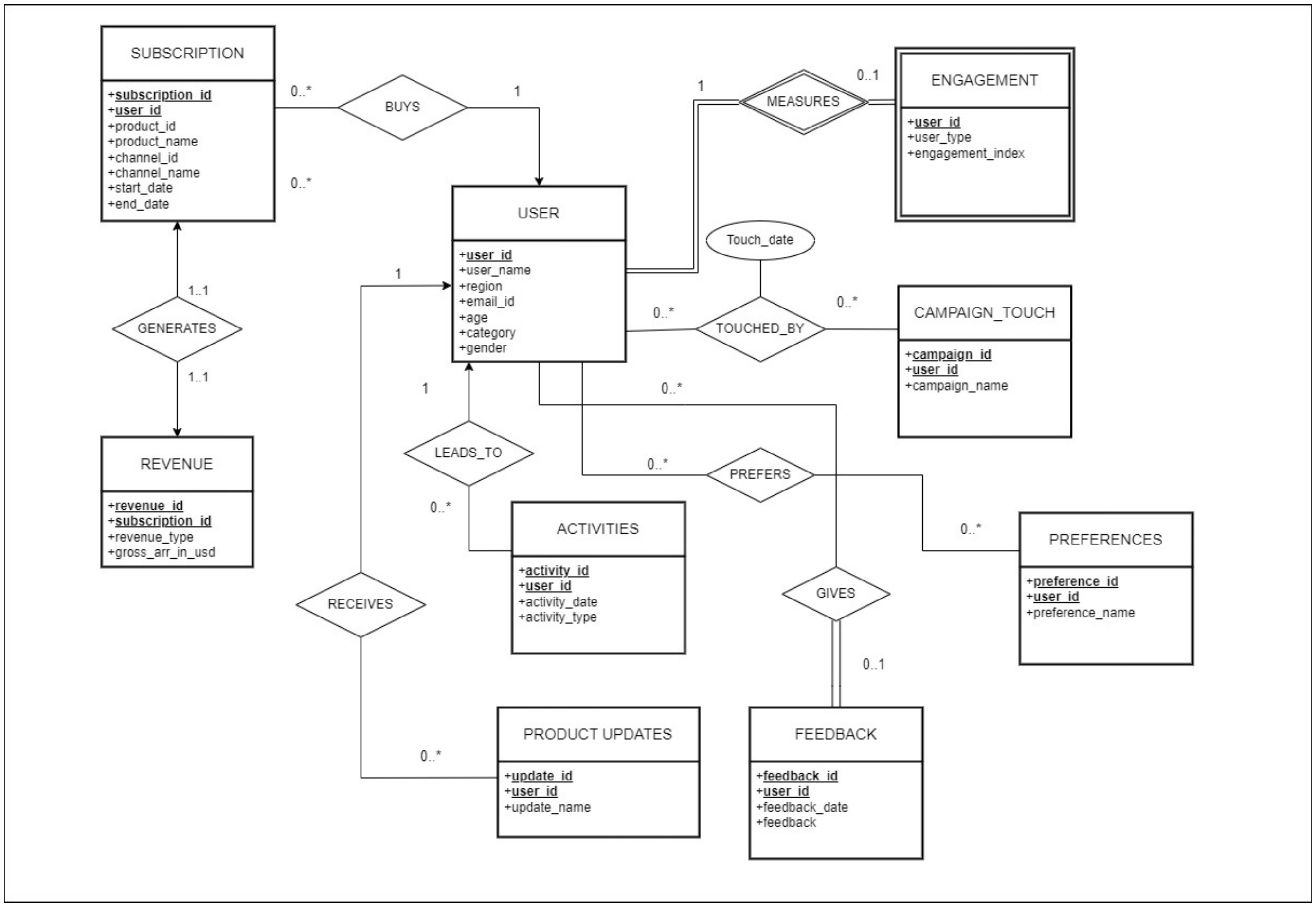


Figure 1: ER Diagram

After 3NF Decomposition, the relations would look like the below.

1. USER (User_id, user_name, region, Email_id, age, category, gender)
2. SUBSCRIPTION (User_id, Subscription_id, Product_id, Channel_id,Start_date, End_date)
3. PRODUCT(Product_id,Product_Name)
4. CHANNEL(Channel_id,Channel_Name)
5. REVENUE (Revenue_id, Subscription_id, revenue_type, Gross_arr_in_usd)
6. ENGAGEMENT (user_id, user_type, engagement_index) – Weak Entity
7. ACTIVITIES (Activity_id, user_id, activity_type, activity_date)
8. PRODUCT_UPDATES (Update_id, User_id, Update_name)
9. CAMPAIGN_TOUCH (Campaign_id, User_id, Campaign_name)
10. FEEDBACK (feedback_id, user_id, feedback_date, Feedback)
11. PREFERENCES (preference_id, user_id, preference_name)
12. TOUCHED_BY (User_id, Campaign_id, Touch_date)
13. PREFERS (User_id, Preference_id)

Data Collection

1. **Collection of Customer Information from Kaggle Dataset:**
We began by sourcing a dataset titled "Customer Subscription Data" from Kaggle, a well-known platform for data science and machine learning projects. This dataset, accessible via the provided URL, contains a variety of customer-related attributes and serves as the foundational element of our data collection process. The dataset can be accessed at the following URL: `https://www.kaggle.com/datasets/gsagar12/dspp1?select=customer_info.csv`
2. **Use of the Faker Python Library to Augment Data:**
To enrich our dataset, we utilized the 'Faker' library in Python. Faker is a valuable tool for generating synthetic data, which is especially useful when we need to expand datasets with diverse and realistic data entries. We used Faker to generate additional data, thereby introducing new variables and attributes to the existing dataset. This approach allowed us to create new columns and tables, enhancing the dataset's complexity and utility for our analysis and modeling purposes. The synthetic data created was seamlessly integrated with our original data, ensuring a richer and more comprehensive dataset for our project.
3. **Organization of Synthetic Data into Pandas Data Frames:**
After generating data with Faker, we organized this data into separate Pandas data frames. Pandas, a powerful Python library for data manipulation and analysis, is particularly effective for managing tabular data. Each data frame was structured to mirror the specific tables of our intended database schema, facilitating efficient data management.
4. **Establishing Connection to the Database Backend from Python:**
We established a connection between our Python environment and the database backend. This involved configuring necessary parameters such as the host, database name, user credentials, and more. This connection is pivotal for enabling direct execution of database operations from within Python.
5. **Transferring Data to the Database Using DML Statements:**
Using Data Manipulation Language (DML) statements, we transferred the data from the Pandas data frames to the corresponding tables in our database. This process involved executing INSERT statements to populate the database tables with both the Kaggle dataset and the Faker-generated data.
6. **Attention to Data Integrity and Consistency:**
Throughout the data loading process, we paid close attention to maintaining data integrity and consistency. This step was crucial for ensuring the data types, constraints, and relationships were correctly upheld, and the data was accurately mapped to the appropriate tables and columns in the database. Ensuring data integrity is fundamental to the reliability and functionality of the database for any application.

Application Description

The application features a user-friendly interface that facilitates the execution of SQL queries on a database. Additionally, the application provides data visualization capabilities, including the generation of bar graphs and pie charts, which aids in the comprehension and analysis of data patterns.

The system is designed to respond to user queries, retrieving data from the backend as per the availability and relevance to the user's request. This demonstrates the application's adaptability and responsiveness to various data retrieval needs.

Backend Functionality and Data Handling:

- **Establishing Database Connection:** The user can initiate a query by inputting it in the UI and when it is run, it triggers the creation of a SQL connection between the MySQL client and the server via the PyMySQL module.
- **Data Organization:** Once the data is retrieved, it is structured into a DataFrame using the Pandas library, a powerful tool in Python for data manipulation.
- **Graphical Data Representation:** For the effective presentation of data, the application employs visualization tools such as Matplotlib and Seaborn, which transforms the data contained within the DataFrame into graphical formats for enhanced analytical clarity.

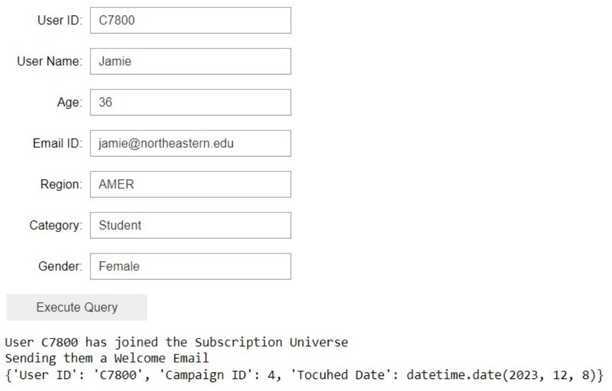
This UI is engineered to simplify the process of database querying and data analysis, offering a streamlined, code-minimal approach for users to engage with and visualise data.

Some important features of the application include:

View_channel_revenue_summary: This view compiles data to offer insights into the generation of revenue through diverse channels. It aids in analyzing revenue, assists in making strategic business decisions, and is useful for identifying emerging trends.

View_low_engaged_users: This specialized view curates a concise and easily navigable set of data targeting users who demonstrate low engagement levels. It serves as an effective tool for analyzing patterns of low engagement, enabling the identification of potential areas for improvement in user interaction and engagement strategies. This view is instrumental in pinpointing specific user segments that may require additional attention or targeted interventions to enhance their engagement with the application or service.

Trig_send_welcome_email: This trigger is designed to automatically send a welcome email to new users as soon as they join the subscription universe. It acts as an automated response mechanism, ensuring that every new subscriber receives a prompt and personalized welcome, enhancing user engagement and experience from the moment they sign up.



User ID:	C7800
User Name:	Jamie
Age:	36
Email ID:	jamie@northeastern.edu
Region:	AMER
Category:	Student
Gender:	Female

```
User C7800 has joined the Subscription Universe
Sending them a Welcome Email
{'User ID': 'C7800', 'Campaign ID': 4, 'Tocuhed Date': datetime.date(2023, 12, 8)}
```

Figure 2: A Trigger to send welcome email when a new user subscribes

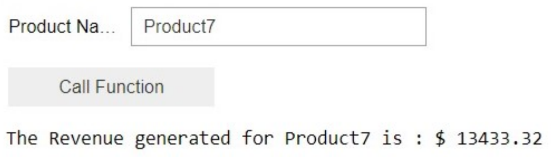
SP_non_touched_users: Identifies users who have not been reached by any advertising campaign. This stored procedure is crucial for targeting untapped segments of the user base, ensuring that marketing efforts are comprehensive and inclusive.

SP_send_exp_reminder: Sends out reminders to users about their subscription renewal one day before the expiration date. This procedure helps in reducing churn by reminding users to renew, thereby maintaining continuity in subscription services.

SP_find_peak_usage: Determines the day of the year that experienced the highest number of app launches. This stored procedure is valuable for understanding user behaviour patterns and planning for high-traffic periods.

Func_user_engagement: Calculates the distinct count of users whose engagement index is below 0.5. This function is instrumental in identifying less engaged users, providing insights for strategies to enhance user engagement.

Func_product_revenue: Computes the revenue generated by a specific product. This function aids in analyzing the financial performance of products, contributing to effective product management and marketing strategies.



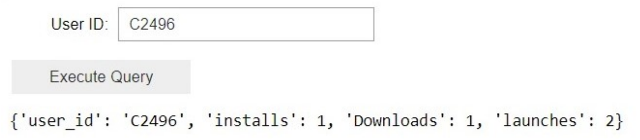
Product Na... Product7

```
The Revenue generated for Product7 is : $ 13433.32
```

Figure 3: Query to retrieve a product's revenue

Func_most_popular_product: Identifies the product with the highest number of subscriptions. This function is key in understanding consumer preferences and trends, and in guiding product development and promotional activities.

Func_get_user_revenue: Retrieves the total revenue generated by a specific user, given their user ID. This function offers valuable insights into the spending patterns and value contribution of individual users, aiding in personalized marketing and customer relationship management.



User ID: C2496

```
{'user_id': 'C2496', 'installs': 1, 'Downloads': 1, 'launches': 2}
```

Figure 4: A query that retrieves user's activity counts

Reports and Visualizations:

Figure 5 offers a graphical representation of revenue by geographic region and subscription duration. This information can enable the company to discern emerging trends and adjust business strategies accordingly. By leveraging these insights, the company can enhance customer engagement with the most profitable subscription plans, thus driving revenue growth. Moreover, by tracking the performance of different subscription models, the company can allocate resources more efficiently and develop targeted promotional campaigns to boost underperforming segments.

Figure 6 presents the number of active users participating in monthly activities throughout a year. It shows a trend of high user engagement at the beginning and end of the year, with some variability in the middle months. The company can utilize this data to identify when to ramp up promotional

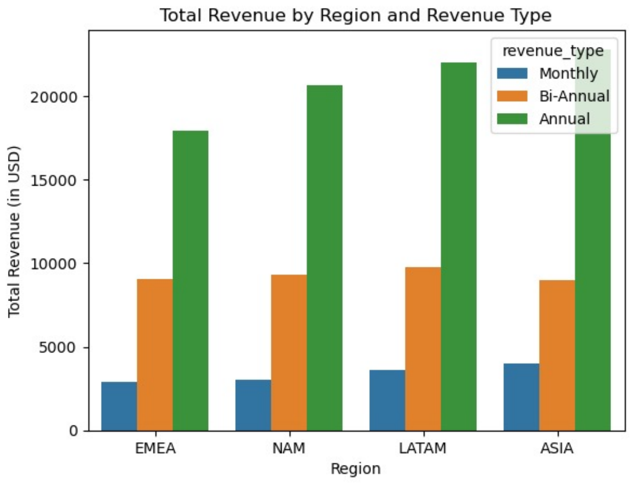


Figure 5: Revenue generated by each region

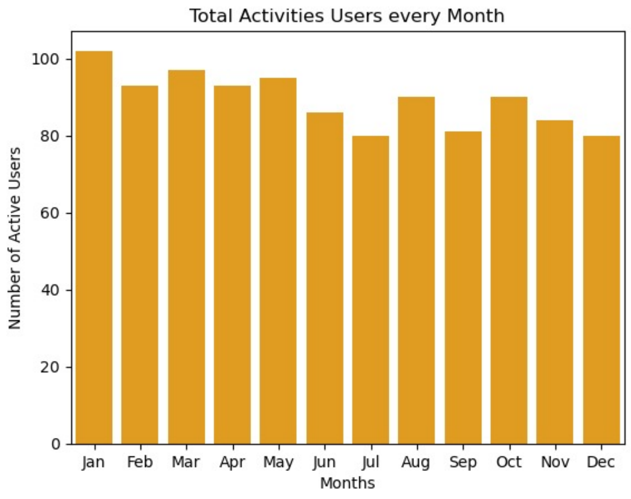


Figure 6: Active users every month

efforts or launch new features to leverage periods of high activity. During months with lower engagement, targeted initiatives can be implemented to maintain user interest. This cyclical insight is invaluable for efficient resource allocation, planning strategic operations, and enhancing overall user retention and acquisition strategies.

Figure 7 shows how many emails were received by different users, with the X-axis categorizing the users by the number of emails received and the Y-axis quantifying the users. The majority received a single email, with a notable decrease as the number of emails increases. This pattern indicates a potential strategy for the company to focus on sending fewer, more impactful emails to avoid overwhelming users. By analyzing this trend, the company can tailor its communication strategy to enhance engagement and improve the effectiveness of its email campaigns. Understanding user thresholds for email communication can lead to more personalized marketing efforts and better customer retention.

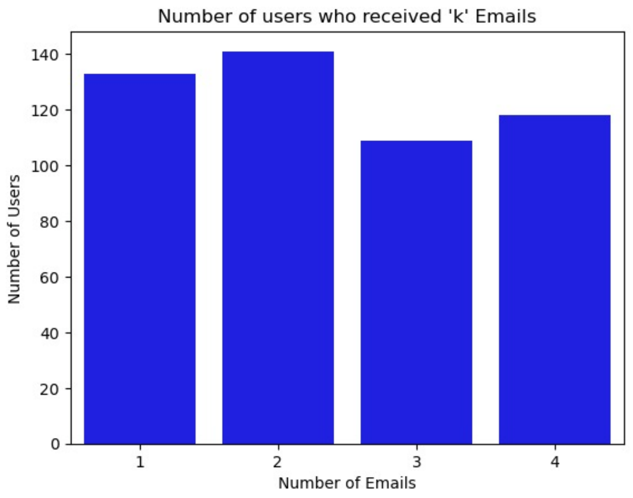


Figure 7: Number of users who received ’k’ emails

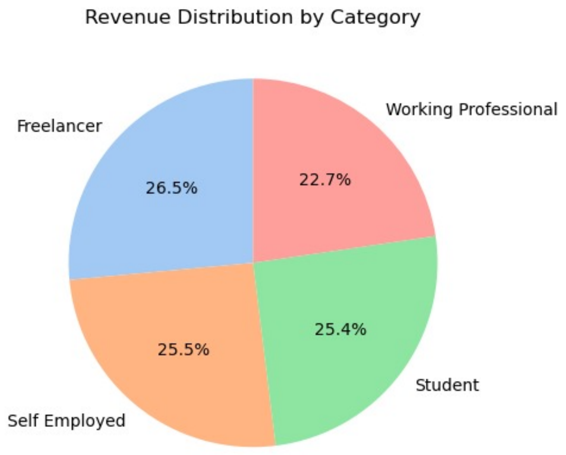


Figure 8: Revenue distribution by category

The pie chart in Figure 8 depicts the division of revenue across four categories of earners: Freelancers, Working Professionals, Students, and Self-Employed individuals. With genuine data on revenue distribution by professional category, a company can refine its market segmentation and create targeted marketing initiatives. This data would enable the development of tailored products and services, optimized resource allocation, and dynamic pricing strategies that resonate with each customer group. Additionally, it would provide insights for enhancing customer retention programs and allow the company to adapt swiftly to market trends, thereby driving sustained revenue growth and competitive advantage.

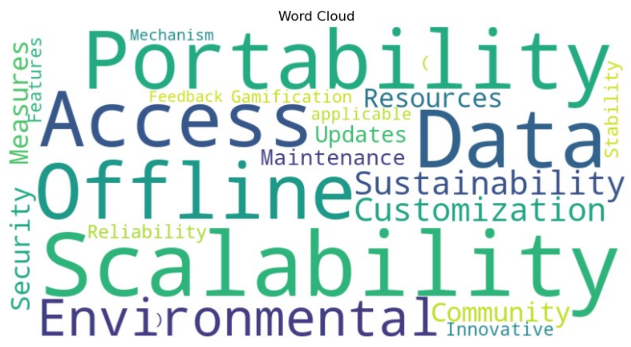


Figure 9: Word Cloud

The word cloud in Figure 9 showcases a range of terms that could represent features or aspects of a product or service valued by customers. Larger words indicate higher frequency or preference, pointing to 'Scalability', 'Offline', 'Access', and 'Portability' as potential top priorities for customers. Marketers can analyze such visual data to discern customer priorities and adjust their strategies accordingly. By leveraging tools like the NLTK library to analyze customer feedback, companies can create word clouds to quickly visualize customer sentiment and preferences, enabling data-driven decisions to enhance product development and marketing campaigns.

Conclusion

We have gained a deep understanding of how to design, develop and test database system. Through this project, we have learned how to adhere to the standard conventions of our technology stack and writing code that can be easily understood by others. By optimizing the design in an iterative fashion, we learned how important data de-duplication is for large applications. By structuring the database to eliminate redundancy and dependency issues, we have learned to ensure data integrity and maintainability. This involved breaking down large tables into smaller, related tables and establishing appropriate relationships between them. Designing queries which mimic real-world retrieval techniques in a business setting has enabled us to up skill ourselves and be prepared to build real world applications. If we had more time to work on this project, we would improve user's experience by building a more interactive user interface. We would also build REST APIs to connect to the backend, which would then prevent direct access to the database.

To the future students of DS 5110, we would advice you to spend a good amount of time on your assignments because that will help you with your project.