Group 8

Icon

Description automatically generated

Andrew Walton

McKay Flake

WingKi Yu

Alexis Navarro

Table of Contents

**Table of Contents1**

**Executive Summary 2**

**Organization/Application Introduction 3**

**Initial Requirements4**

**Database Design 5**

**Database Implementation 7**

**Database Demonstration 9**

Executive Summary

The following executive summary outlines the key points of our group project focused on creating a transactional database for Spotify.

The project aims to develop a transactional database that will help Spotify to efficiently manage its large volume of data related to customer demographics such as age, name, gender and so on. The database will also be designed to enable efficient transaction processing and support the rapid growth and scalability of Spotify's platform.

The project will involve creating logical and physical models to better understand the relationships within the data that will be captured. The project will also involve creating the database and testing its functionality and reliability. The database will be built in Microsoft Azure and be coded in SQL. .

Once complete it will showcase an effective understanding of the process of creating, implementing and testing a transactional database. The database will be able to create the necessary data to support transactions more quickly and efficiently, improve the accuracy and relevance of personalized recommendations, and enhance the overall user experience on the platform. It will also position Spotify for continued growth and innovation in the rapidly-evolving music streaming industry.

Organization/Application Introduction

**Vision and objectives for the organization**

We want to see spotify grow, add new products and expand to new markets. A solid grasp on one’s data will only support the organization in these goals and transnational database such as the one we created in a great step in achieving this objective.

**Products/services provided by the organization**

The main product that Spotify provides is access and custom recommendations to music. Spotify also provides access to exclusive podcasts and music videos.

**A description of how transactional databases are most likely used to support operations for the organization.**

Transactional databases are critical to supporting the day-to-day operations of Spotify. These databases are designed to manage large volumes of data related to user activities, such as new customers and customers demographics. Transactional databases allow Spotify to efficiently store and retrieve this data in real-time. Additionally, transactional databases can support complex transactions, such as processing payments and managing subscription renewals, which are essential to Spotify's business model.

Initial Requirements

Spotify wants to create a database to track new customers' key demographics in order to recommend the best music possible to remain a top music provider in the industry.

For all new customers (entity) they will be uniquely identified by their given customer ID number (unique identifier attribute). Each customer also has a name field where we capture first name, last name and middle initial (attributes). We will also capture their email address and date they signed up (2 more attributes).

All customers are required to enter key demographics (weak entity) about themselves to recommend music. This demographic information is identified by the customer ID. New Customers will be required to enter up to three music genres they like (multi-valued attribute), gender, age and ethnicity.

The second main objective is to collect payment information for the new customers that signed up for the premium subscription model. Not all new customers are premium customers (participation constraint). In the premium customers field we will collect credit card number, billing address, name on card (first , last, middle initial). The combination of card number and billing address are the attributes uses to uniquely identify each individual.

Database Design

# Conceptual Model

Diagram

Description automatically generated

Here we created the initial conceptual model for our database. We chose to make customer and entity with demographics as a weak entity, and we also created the premium subscription entity. The premium subscription entity has the most attributes as we want to have the most information about our premium subscribers. We also added cardinality constraints to restrict premium subscriptions to be used by one person, and each person can only have one premium account.

# Logical Model

Table

Description automatically generated

We got closer to our database design by creating a logical model based off what we modeled in our conceptual model. We can start to visualize what we have once we create our database. We also can see more clearly the relationships between the different entities here and we can see how they the tables are related and linked to each other.

Database Implementation

CREATE TABLE customer (

  customer\_id VARCHAR(11) NOT NULL PRIMARY KEY,

  first\_name VARCHAR(105) NOT NULL,

  middle\_initial CHAR(1),

  last\_name VARCHAR(105) NOT NULL,

  Sign\_up\_Date DATE,

  email VARCHAR(105)

);

Graphical user interface, text, application, email

Description automatically generated

CREATE TABLE demographics (

  customer\_id VARCHAR(11) FOREIGN KEY REFERENCES customer(customer\_id),

  age INT,

  ethnicity VARCHAR(50) CHECK (ethnicity IN ('American Indian or Alaska Native', 'Asian', 'Black or African American', 'Hispanic or Latino', 'Native Hawaiian or Other Pacific Islander', 'White')),

  gender VARCHAR(6) CHECK (gender IN ('male', 'female')),

    PRIMARY KEY (age, ethnicity, gender)

);

Graphical user interface, text, application, email

Description automatically generated

CREATE TABLE PremiumSubscription (

    customer\_id VARCHAR(11) FOREIGN KEY REFERENCES customer(customer\_id),

        subscription\_id VARCHAR(11) PRIMARY KEY,

    first\_name VARCHAR(105),

    middle\_initial CHAR(1),

    last\_name VARCHAR(105),

    Street VARCHAR(105),

    city VARCHAR(105),

    state CHAR(2),

    zip\_code CHAR(5),

    country VARCHAR(105)

);

Graphical user interface, text, application, email

Description automatically generated

Here we can see some of the most fundamental code that helped us create our database. We created tables for Premium Subscription, Demographics, and Customers. We had to be conscious to use the correct data types for each attribute (eg Varchar vs Char). We also had to ensure that we correctly implemented the Primary and Foreign Keys.

Database Demonstration

|  |  |  |
| --- | --- | --- |
| Category | Description | Status |
| Customer | Store customer information including name, email, and signup date | COMPLETE |
| Customer | Assign each customer a unique ID number | COMPLETE |
| Key demographics | Capture customer demographics including gender, age, and ethnicity | COMPLETE |
| Key demographics | Link customer demographics to customer ID | COMPLETE |
| Premium customer | Store payment information for premium customers including billing address and name on card | COMPLETE |
| Premium customer | Use subscription ID as the unique identifier for premium customers | COMPLETE |
| Premium customer / customer | Ensure that not all new customers are premium customers | COMPLETE |

# Customer Information

#### Store Customer Information

INSERT INTO customer (first\_name, last\_name) VALUES

('Mark', 'Huges')

Graphical user interface, text, application

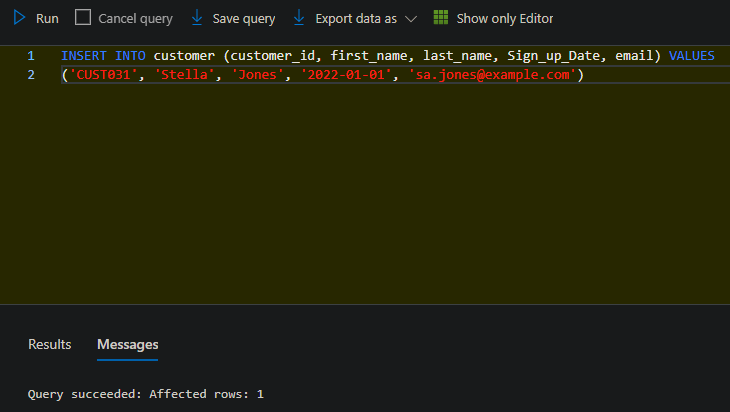
Description automatically generated

INSERT INTO customer (customer\_id, first\_name, last\_name, Sign\_up\_Date, email) VALUES

('CUST031', 'Stella', 'Jones', '2022-01-01', 'sa.jones@example.com')

SELECT \* FROM [dbo].[customer]

WHERE customer\_id = 'CUST031'



Graphical user interface, text, application

Description automatically generated

#### Update Customer Information

Objective: Update a customer’s last name based on their customer id

UPDATE [dbo].[customer]

SET last\_name = 'Wu'

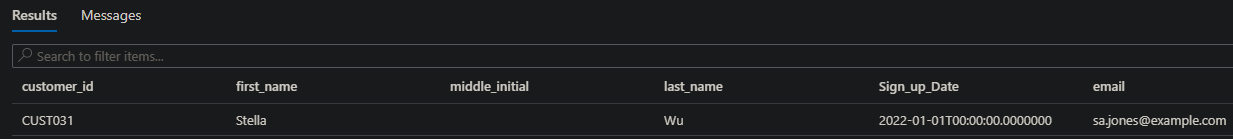
WHERE customer\_id = 'CUST031'

SELECT \* FROM customer

WHERE customer\_id = 'CUST031'

A screenshot of a computer

Description automatically generated with medium confidence



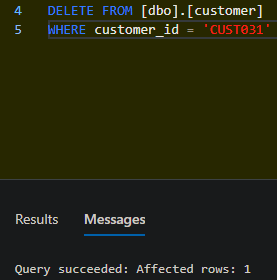
#### Delete a Customer

Objective: Delete a customer based on their customer id

DELETE FROM [dbo].[customer]

WHERE customer\_id = 'CUST031'

SELECT \* FROM [dbo].[customer]



A screenshot of a computer

Description automatically generated with medium confidence

# Demographics

#### Select All Customers by Ethnicity

Objective: Select all customers and return their ethnicity and email based on their ethnicity being “Asian”

SELECT d.ethnicity, c.email

FROM [dbo].[demographics]d, [dbo].[customer]c

WHERE d.customer\_id = c.customer\_id

AND d.ethnicity = 'Asian'

Text

Description automatically generated

#### Select Customers with a Premium Subscription

Objective: Show all customers with a Premium Subscription

SELECT \* FROM [dbo].[customer]c

LEFT OUTER JOIN [dbo].[PremiumSubscription]p

ON p.customer\_id = c.customer\_id

ORDER BY c.customer\_id ASC

Graphical user interface, application, table

Description automatically generated

#### Remove Columns in a Table

Objective: Remove columns from a table in our database

ALTER TABLE [dbo].[demographics]

DROP COLUMN priceGraphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

#### Add Demographic Information

Objective: Add demographic information for a customer

INSERT INTO [dbo].[demographics] (Genres, Age, Ethnicity, Gender)

VALUES

 ('Country', 52, 'White', 'male');

Graphical user interface, text, application, email

Description automatically generated

# Premium Subscription

#### Add Column to Table

Objective: Add a new column to a table in our database

ALTER TABLE [dbo].[PremiumSubscription] ADD price money NULL

Graphical user interface, text, application, email

Description automatically generated

#### Update Price

Objective: Update the price of the premium subscription

UPDATE dbo.PremiumSubscription

SET price = price --9.99

WHERE price = 9.99

GO

Graphical user interface, text, application, email

Description automatically generated