NYSync

CSE 460 - Data Model Query Language

Shailesh Mahto - 50540379
Shivam Bhalla -
Abhiroop Ghosh - 50541142

Problem Statement -

Navigating the diverse and bustling event scene in New York City can be overwhelming for residents and visitors alike. With countless events happening across the city, ranging from concerts and art exhibitions to food festivals and theater performances, individuals often struggle to find relevant information about upcoming events. The lack of a centralized platform that provides comprehensive event details, including event genre, artist information, and venue details, hinders people's ability to discover and attend events that align with their interests and preferences.

Reasons for Choosing a Database over an Excel File -

1. **Scalability:** A database can efficiently handle large volumes of event data, including event details, artist information, and venue details, without the limitations of Excel files, which may become slow and cumbersome as data volume increases.

2. **Data Integrity:** Databases enforce data integrity constraints, reducing the risk of errors and inconsistencies in event listings and related information compared to Excel files, where data validation and integrity checks are limited.

3. **Real-time Updates:** A database enables real-time updates to event listings, ensuring that users have access to the latest information about upcoming events, whereas Excel files require manual updates and may not reflect changes promptly.

4. **Advanced Search Functionality:** Databases offer advanced search and filtering options, allowing users to narrow down their search based on criteria such as event genre, date, location, and artist, which is challenging to achieve efficiently in Excel files.

5. **Collaborative Access:** A database allows multiple users to simultaneously access and update event data, facilitating collaboration among event organizers, promoters, and users. Changes made to the database are immediately visible to all users, facilitating real-time collaboration and decision-making.

Target User -

- **Event Attendees:** Individuals living in or visiting New York City who are interested in attending events across various genres.
- **Event Organizers:** Individuals or organizations hosting events in New York City who want to promote their events to a wider audience.
- **Venue Owners/Managers:** Owners or managers of venues hosting events in New York City who seek to attract event organizers and attendees to their establishments.
- **Promoters/Marketers:** Individuals or companies responsible for promoting events and driving ticket sales in New York City.

(Ignore add any points if y'all have)Intended Users -

- **Event Attendees:** Individuals living in or visiting New York City who are interested in attending events across various genres.
- **Event Organizers:** Individuals or organizations hosting events in New York City who want to promote their events to a wider audience.
- **Venue Owners/Managers:** Owners or managers of venues hosting events in New York City who seek to attract event organizers and attendees to their establishments.
- **Promoters/Marketers:** Individuals or companies responsible for promoting events and driving ticket sales in New York City.

Database Administration:

- The database will be administered by a dedicated team within the organization responsible for managing the NYC Events Platform (NEP).
This team will include:

- Database Administrators (DBAs): Responsible for database setup, configuration, security management, performance optimization, data maintenance, and backup and recovery.

- System Administrators: Responsible for managing the infrastructure and servers hosting the database platform, ensuring uptime, scalability, and overall system reliability.

Event Table:

Attributes:

- event_id (Primary Key): Unique identifier for each event.
- name: Name of the event.
- url: URL link to the event.
- start_date: Date when the event starts (datatype: DATE).
- start_time: Time when the event starts (datatype: TIME).
- price_min: Minimum price for tickets (datatype: DECIMAL).
- price_max: Maximum price for tickets (datatype: DECIMAL).
- seatmap_url: URL link to the seatmap of the venue.
- age_restrictions: Age restrictions for the event (datatype: INTEGER).
- venue_id (Foreign Key): Reference to the Venue table indicating where the event takes place.

(Action: ON DELETE CASCADE - If a venue is deleted, all events associated with that venue are also deleted.)

Genre Table:

Attributes:

- genre_id (Primary Key): Unique identifier for each genre.
- name: Name of the genre.

Event_Genre_Table:

Attributes:

- event_id (Foreign Key): Reference to the Event table indicating which event is associated with a genre.
- genre_id (Foreign Key): Reference to the Genre table indicating the genre of the event.

Venue Table:

Attributes:

- venue_id (Primary Key): Unique identifier for each venue.
- name: Name of the venue.
- postal_code: Postal code of the venue's location.
- city: City where the venue is located.
- state_name: Name of the state where the venue is located.
- state_code: Code of the state where the venue is located.
- country: Country where the venue is located.
- country_code: Code of the country where the venue is located.
- address: Address of the venue.
- latitude: Latitude coordinate of the venue's location.
- longitude: Longitude coordinate of the venue's location.

Artist Table:

- Attributes:
- artist_id (Primary Key): Unique identifier for each artist.
- name: Name of the artist.

Event_Artist Table:

Attributes -

- event_id (Foreign Key): Reference to the Event table indicating which event is associated with an artist. (Action: ON DELETE CASCADE - If an event is deleted, all artist associations related to that event are also deleted.)
- artist_id (Foreign Key): Reference to the Artist table indicating the artist performing at the event.

Summary of the above design:

- Primary keys are indicated for each table to uniquely identify records within the table.
- Foreign keys are specified where there are relationships between tables to maintain referential integrity.
- Detailed descriptions are provided for each attribute, including purpose and datatype.
- Default values are not specified unless explicitly required.
- Attributes that can be set to 'null' are indicated as such in the descriptions.

Implementation Steps:

Database Design:

- Design the database schema to accommodate event-related data, including tables for events, attractions, venues, artists, and genre associations.
- Define relationships between entities to maintain data integrity and facilitate efficient querying.

API Integration:

- Obtain an API key from Ticketmaster to access their API services.
- Utilize Ticketmaster API endpoints to fetch event data, including event details, attractions, venues, and artist information.
- Develop scripts to make HTTP requests to the Ticketmaster API, retrieve event data in JSON format, and parse the data for database insertion.

Data Import:

- Python scripts were employed to extract data from the Ticketmaster API responses via REST API calls.
- The requests library facilitated the fetching of API responses, storing the data in Python dictionaries.
- Data was structured into a DataFrame format using the pandas library, enabling easy manipulation and organization.
- The JSON data obtained from the Ticketmaster API responses was parsed using built-in Python functions and methods.
- The parsed data was transformed into a structured DataFrame, encapsulating event details such as name, start date, venue information, and artist details.
- Utilizing the to_sql() function provided by pandas, the event data was seamlessly transferred to a PostgreSQL database.
- The data import process ensured consistency and integrity by inserting the parsed event data into corresponding tables within the database.
- The combination of Python, pandas, and SQL Alchemy streamlined the data import process, enabling efficient integration of event data from the Ticketmaster API into the database system.

Designing Query:

- Design SQL queries to perform various operations on the database, such as retrieving event details, searching for events by location, date, genre, etc.
- Optimize queries for performance and efficiency to handle large volumes of event data effectively.