**Stockton Symphony Database ERD Report**

**Introduction**

This report outlines the process of designing the Entity-Relationship Diagram (ERD) for the Stockton Symphony Database. The primary goal of the project is to unify concert data and performance data into a relational database, enabling the symphony to perform predictive and analytical tasks such as revenue prediction and trend analysis.

**Thought Process**

Understanding the Requirements:

* The project requirements were analyzed to define the key data entities and their relationships. The data included:
  + **Concert Data** including series type (Pops or Classical), date, time, concert name, single ticket revenue and sales, subscription revenue and sales, total revenue, and total ticket sales.
  + **Music Performed Data** includes details about the pieces performed, their composers and date performed etc.

Key Design Considerations:

* To handle both datasets effectively:
  + A Concert table was designed as the central entity.
  + Supporting entities like Performance, Work, Ensemble, Ticket Sale, and Ticket Type were added to normalize the data and establish relationships.
* The focus was on creating a flexible and scalable design to support analytical queries.

Mapping Relationships:

* Relationships were drawn based on the logical connections:
  + Each concert can have multiple performances.
  + Each performance is associated with a specific work.
  + Concerts involve ensembles and generate ticket sales, which are categorized by ticket types.

**Entity-Relationship Diagram**

The ERD was developed with the following entities:

1. Concert:
   * Represents concert-level details like name, series type, date, and total revenue.
   * Primary Key: ConcertID.
2. Performance:
   * Links concerts with individual works performed.
   * Primary Key: PerformanceID.
   * Foreign Keys: ConcertID, WorkID.
3. Work:
   * Stores details about pieces performed, including the composer and title.
   * Primary Key: WorkID.
4. Ensemble:
   * Captures information about the ensemble performing the works.
   * Primary Key: EnsembleID.
5. TicketSale:
   * Tracks ticket sales and revenue details for each concert.
   * Primary Key: TicketSaleID.
   * Foreign Key: ConcertID, TicketTypeID.
6. TicketType:
   * Defines types of tickets (e.g., subscription, single ticket).
   * Primary Key: TicketTypeID.

**Insights from the ERD**

1. Centralized Concert Data:
   * The Concert table acts as the anchor for all ticketing, revenue, and performance data.
   * Relationships allow easy retrieval of all data related to a specific concert.
2. Normalization:
   * The schema eliminates redundancy by storing related data in separate tables (e.g., Work stores information about pieces, while Performance links pieces to concerts).
3. Scalability:
   * The schema design supports additional data integration, such as adding new ticket types or ensembles without restructuring the database.
4. Support for Analysis:
   * The schema supports queries for revenue trends, popular works, and ticket sales analysis.

**Follow-Up Questions from the Client**

1. Data Structure and Scope:
   * Should the database track venue-specific details (e.g., seating capacity, location)?
   * Is there a need to track individual audience members or buyers?
2. Revenue Breakdown:
   * Do we need to separate online vs. offline ticket sales?
   * Should donations or sponsorships be incorporated into revenue tracking?
3. Performance Details:
   * Should the database capture encore pieces or unplanned performances?
   * Is tracking conductor or soloist details important?
4. Future Use Cases:
   * Are there additional use cases for the database, such as email campaigns or ticket pricing optimization?

**Conclusion**

This ERD provides a well-structured and normalized schema that integrates concert and performance data, supporting Stockton Symphony's needs for data-driven decision-making. The design ensures flexibility, scalability, and ease of analysis. Further refinements can be made based on client feedback and additional requirements.