1. When new order is placed it makes an entry in purchase history table along with invoice ID and it stores the price of the books and quantity at time of order placed and when the revenue is generated it is calculated by the product of quantity and price of book.
2. When the new order is placed it schedule an email to the authors using bull queue and redis. It sends an email to 100 authors at same time . It can be check on Bull dashboard to check the status of job schedule to send an email which can be seen on route **localhost:3000/api/queues** according to the code written.
3. Data Modeling: Designing a suitable data model is the foundation of a well-performing database. This involves identifying entities, their attributes, and the relationships between them. Choices may include using entity-relationship diagrams (ERDs) and normalization techniques to ensure data integrity and reduce redundancy.

Choosing Storage Engine: MySQL offers different storage engines, each with its own strengths and weaknesses. The two most common are InnoDB and MyISAM. InnoDB is ACID-compliant, supports transactions, and is more suitable for applications with high write loads and transactional requirements. MyISAM offers full-text searching and better performance for read-heavy workloads but lacks transaction support.

Indexing Strategy: Proper indexing is crucial for query performance. Choices involve identifying the right columns to index based on query patterns and balancing the trade-offs between read and write performance. Common types of indexes include B-tree indexes, full-text indexes, and spatial indexes.

Partitioning: Partitioning involves splitting large tables into smaller, more manageable chunks based on certain criteria (e.g., range, hash, or key). This can improve query performance and manageability, especially for very large datasets. Choices include deciding on the partitioning key and method based on access patterns and scalability requirements.

Normalization vs. Denormalization: Normalization reduces data redundancy by organizing data into separate tables and linking them through relationships. Denormalization involves reintroducing redundancy to optimize query performance. The choice between normalization and denormalization depends on factors such as read vs. write performance requirements and the complexity of queries.

Data Types: Choosing appropriate data types for columns impacts storage space and query performance. MySQL offers various data types, including numeric types, string types, date and time types, and JSON support. Choices involve selecting the most suitable data type for each column based on the nature of the data and the operations performed on it.

Query Optimization: Writing efficient queries is crucial for database performance. Choices include optimizing SQL queries by using appropriate join techniques, avoiding unnecessary subqueries, and utilizing indexing effectively. Additionally, using MySQL's query optimizer hints and profiling tools can help fine-tune query performance.

Backup and Recovery: Implementing a robust backup and recovery strategy is essential for data integrity and business continuity. Choices involve selecting appropriate backup methods (e.g., full backups, incremental backups) and scheduling backups based on recovery point objectives (RPOs) and recovery time objectives (RTOs).

Security: Securing the database against unauthorized access and data breaches is critical. Choices include implementing access controls, encrypting sensitive data, and regularly updating MySQL to patch security vulnerabilities. Additionally, using techniques like parameterized queries and prepared statements helps prevent SQL injection attacks.

Scalability and High Availability: Planning for scalability and high availability involves making choices such as implementing replication, clustering, or sharding to distribute workload and ensure continuous availability. MySQL offers features like MySQL Cluster and MySQL InnoDB Cluster for high availability solutions.