**OLD&SOLD: Data Integration and Analysis Proposal**

**Le Lam Trung Hau - 17229041**

**Nguyen Dac Tri Tai - 19229097**

**Nguyen Thi Bach Hop - 19229091**

**Hieu Nguyen Minh - 19229089**

**Database and Big Data**

**Executive Summary**

OLD&SOLD has been facing issues in operations due to the physical filing system and spreadsheets to keep track of customers, sales, payments as well as other business services. This, in turn, has heavily affected the performance against other competitors in the market. Because of the obsolete physical filing system, expansion has been proven to be very difficult. It is extremely hard to maintain, not to mention scale and expand, the normal day to day business. In order to be able to maintain and scale business operations effectively with the possibility of rapid expansion, it is crucial to adapt a database management system. The objective and purpose of this data integration and analysis proposal is to eliminate as much as possible human errors in data inputs as well as the possibilities of losing track of data or the data itself which will heavily, if not already, affect the business operations. Additionally, to improve the efficiency of business operations and the opportunities to expand by discovering key features for driving in more revenues. This will be achieved through the implementation of a carefully planned and executed database management system which enables better storage, access, modification, and analysis of existing data as well as future data.

**Motivation**

As the company is an emerging business, the company has welcomed many customers and gradually established trust and long-term relationships with customers. However, after each transaction (buying and selling products), the management work takes a lot of time, effort, and resources. The process requires employees to manually log information into different log books which by now has become rather convoluted. Moreover, since the data migration to spreadsheets has not been completely finished, staff still have to manually input sales data, both purchasing and buying to the spreadsheets. Consequently, it is inevitable that the current data on management of employees, customers, and sales contains errors and perhaps some ineligible inputs. This way of storing is time-consuming and labor intensive, and oftentimes causes confusion among employees, loss of data entries. Moreover, the need to expand the market, promote products, and improve quality, as well as increase revenue is a major concern to the company. Therefore, it is at the utmost importance to plan, develop and integrate a database management system to maintain day to day business operations as well as the company competitive edge on the market. Furthermore, the database system will also provide our employees with tools to easily add, delete, update, and search the data. In addition, it will enable the company to collect and store more information that would be useful, if not important, for further analysis in order to improve and expand business operations.

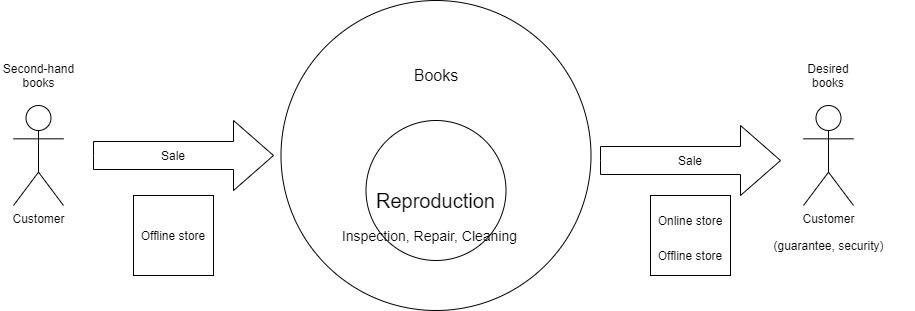
**Business Overview**

1. **Introduction**

OLD&SOLD is a book company that develops second-hand books sales business. The company has specialized in the reuse business and has been working on the development of new business formats. The company purchases customers' used books, recycles (inspect, repair, clean) the products and sells them to customers who need them. The store will continue to explore the possibilities of expanding the reuse business, develop new business formats, enhance handling corners, and expand the genres of products it handles. In addition, by actively working on complex offline store openings and larger stores in these business categories, the company will respond to the diversifying needs and needs of the customers and set a new standard as a "reuse relay base" and "reuse information transmission base."

1. **Business Model**

The company has established a "low-cost, high-return" business model by streamlining purchase, regeneration, and sales operations and operating a small number of elite stores. The company also achieves high management efficiency with low cost and high return that is unrivaled by other companies, achieves low prices with a small number of elite low-cost management.



The graph shows the company’s flow of operating the business with customers. Since our company has been operating the business through offline stores, we are implementing an online store for our business expansion. For the buying process, we provide the selling service for customers in offline stores exclusively. In order to sell books to our customers with the highest quality, we are responsible for evaluating the product carefully before buying and we do not expect to purchase books with extremely bad conditions, such as heavily damaged products. After completing the book’s purchase from customers, we will inspect, repair and clean our products to resell them to the desired customer. For the selling process, we will provide our service in both online and offline stores so that customers can buy our products via our retail website or our offline stores. However, web customers are expected to pick up and checkout their orders in the offline store since we still have not provided the delivery and online payment services.

1. **Business requirement**
2. Objectives

Currently, there are many companies wishing to introduce and sell books through their websites. Therefore, the development of an online book retailing system is essential. Moreover, building an online book website not only meets the needs of introducing and selling used books but also meets the needs of searching online books for readers.

1. Business Activities Description

In the offline store, customers can sell their old books (only offline stores accept collecting old books) as well as buying books. If they want to get more benefits from a company such as discounts and new product information, they can register a membership at the store and receive a physical member card. The employees communicate directly to customers at stores and introduce new campaigns, discounts, and other benefits to customers. They also prepare products following online orders which are managed by managers and then give goods to customers at stores. Besides, administrators and staff work together to manage suppliers, products as well as updating product information such as in stock or out of stock products, new collections, etc.

2.1. The process of buying books at offline store:

When customers come to buy books, they will look for the book to buy or provide information about the book (title, author, publisher) to staff to find the book. Generally, they will find the book themselves with the guidance of the in-store staff (Fig.1)

2.2. The process of buying books on online store:

Step 1: The customer browses the company’s retailing website on a web browser.

The browser will display the homepage of the online store, which shows the products that the business offers. Customers can browse products by pre-listed categories or use the search books function to search by product’s title.

Step 2: After looking for books they need to buy, customers can decide to buy the company’s products. In order to do this, the customer needs to put the product in the cart. Shopping cart is a place to store information about the products that customers intend to order. Customers can also manage their shopping carts. One customer can have any books in their shopping cart and each shopping cart belongs to exactly one customer. Furthermore, the customer can add or delete books in their shopping carts. The shopping cart will have the following information, including cart’s ID, created cart date, last updated cart date, books contained in the shopping cart, etc. Then, customers can continue to view other products or place an order.

Step 3: In order to purchase books on our company’s online retailing store, customers need to log into the store’s system. If the customer has not had the member account, the store’s website will direct he/she to the member account registration page to create a new one. The customer is required to provide the following information to the store, including customer’s name, phone number, email address, home address, and the passwords. After the registration, the customer will receive the member account ID and he/she can login into the store’s website with the account’s name and password. Note that one customer can only register one account and each account must be assigned to exact one customer.

Step 4: After providing sufficient information, customers can review, edit orders and send their orders. The system will notify customers that the order has been sent. Customers can place orders as many as they desire. For each order, the administrator is responsible for recording who placed this order, when the order was made, the ordered books, the order status and total calculated price. The store will contact the customer about the payment. Concerning the payment method, unfortunately, at the early version, the online payment service with credit cards is not available to the customer. In-store payment is available only. Thus, if customers accept paying in store, they need to come to our offline stores to pay and pick up their order at any time of their convenience. Moreover, since the company does not provide the delivery and shipping service, customers need to visit the offline store to pick up their purchased products. Although our registration system requires customers to input the customer’s shipping address and billing address and our database system also stores them, these data will not be used. Finally, the purchase is completed, and customers will come to the offline store to pick up their orders at any time.

\* The bookstore keeps a large number of used books in various genres. Each book is identified by its ISBN. The book in our online bookstore will contain the following information, including its authors’ names, title, edition, publication year, category, publisher, quantity in stock, selling price and illustrating images.

1. Actor’s requirements

* Primary functions of Offline store:

1. Collect used books from customers.
2. Sell renovated books to customers.
3. Allow customers to pick up purchased books.
4. Provide certain extra benefits to customers, such as: membership card, discounts, sales off, etc.

* Primary functions of Online store:

1. Sell renovated books to customers
2. Provide information about extra benefits to customers, such as: membership account, discounts, sales off, etc.

\*Note: The online store does not provide an online service of purchasing old books from customers. The online store also does not provide delivery service, which means that customers have to pick up purchased books at the offline store.

Customer: People who buy books at bookstores. They could be customers buying books directly at offline stores, or those buying through bookstore websites.

* Buy (online, offline) and sell (offline) old books.
* View the information of book products ((by author name, title, category, year or combinations)
* Register membership

Member: customer who have registered membership card at offline store or obtained membership account via online website.

* Log in, log out and change the passwords with the member account.
* View, delete, update his/her personal information.
* View the information of book products ((by author name, title, category, year or combinations))
* Receive updated information (release of new products, discounts, sales, etc.).
* Create a shopping cart
* Add/delete books from the shopping cart
* Place the order and trace the order
* Give feedback on the products.

In-store staff: employees standing at bookstores to guide and find books for customers.

* Provide available services (i.e., help customers find books they want,), as well as information about extra benefits and physical membership cards to customers.
* Prepare products for the online orders and give them to customers at offline stores

Administrator: the person who manages the operations of the bookstore, which include staff management and book management. The administrator will calculate book sales statistics and the number of customers who bought books from bookstores.

* Manage, view, modify customers’ personal information.
* Manage, view, modify employees’ personal information.
* Send update information to customers
* View and manage supplier and product information
* Receive and reply feedback and comments
* Manage orders

4. Use case description

To understand how different actors interact with our engineered online bookstore’s system, we first create a high-level use case diagram, which is shown in Figure 2 in the appendix. Then, we decompose it into a lower-level use case diagram for our company’s selling process. Figure 3 in the appendix is the use case diagram that describes all the activities that the web customer can do to purchase from our online store. The diagram includes 3 different actors, which are customer, member and administrator. Meanwhile, figure 4 in the appendix is the high-level use case diagram that describes all the activities that the administrator interacts with the online system during the process of buying used books from customers. Since the administrator is the only one who has the authority to manage this procedure, the diagram has only one actor. Additionally, we also create a priority matrix table of major use cases for the selling use case diagram to know which component in the system should be prioritized and implemented first (Table 1).

1. Customer

|  |  |  |
| --- | --- | --- |
| Use case’s name | Description | Priority |
| Search products | Web customers look for the product by inputting the information related (title, author, ISBN, etc.) | Medium |
| Browse catalogs | Web customers view the book list based on the different genres. | Medium |
| View recommended books | If customers do not have specific selection, they can view books that are recommended on the bookstore’s webpage. | Low |
| Add to cart | Customers can put their products into the shopping cart if they intend to purchase them. | Medium |

The “Register an account” has 1 minor use case, which is verifying. It describes the event that customers will receive a notification of the account verification from the bookstore’s system to reconfirm that the account has been successfully created. Note that the “Verify” use case is also mandatory and linked to the “checkout” use case because it will notify the members that their orders have been recorded into the system. Hence, its priority is high.

1. Member

The “Log in” use case includes 6 minor use cases, which are checking orders, canceling orders, checking out, changing personal information, passwords, logging out and sending feedback. In order to have a deeper understanding of how the log-in process works, we create a sequence diagram to illustrate the operation (Figure 5).

|  |  |  |
| --- | --- | --- |
| Use case’s name | Description | Priority |
| Check orders | Members modify their orders if they want to update or delete books in the cart of their orders. | Medium |
| Cancel orders | In case members do not want to purchase the book anymore, they cancel orders before checking out. | Medium |
| Check out | Members complete their orders on the online bookstore and checkout to get the invoice. | High |
| Change personal information, passwords | Members edit their personal information and update the new passwords in their member accounts if they wish to change. | Medium |
| Log out | Members log out the log in session on the store’s website. | Medium |
| Send feedbacks | Members can post their comments or questions about a product on the product’s page and wait for the reply from the administrator. | Low |

1. Administrator

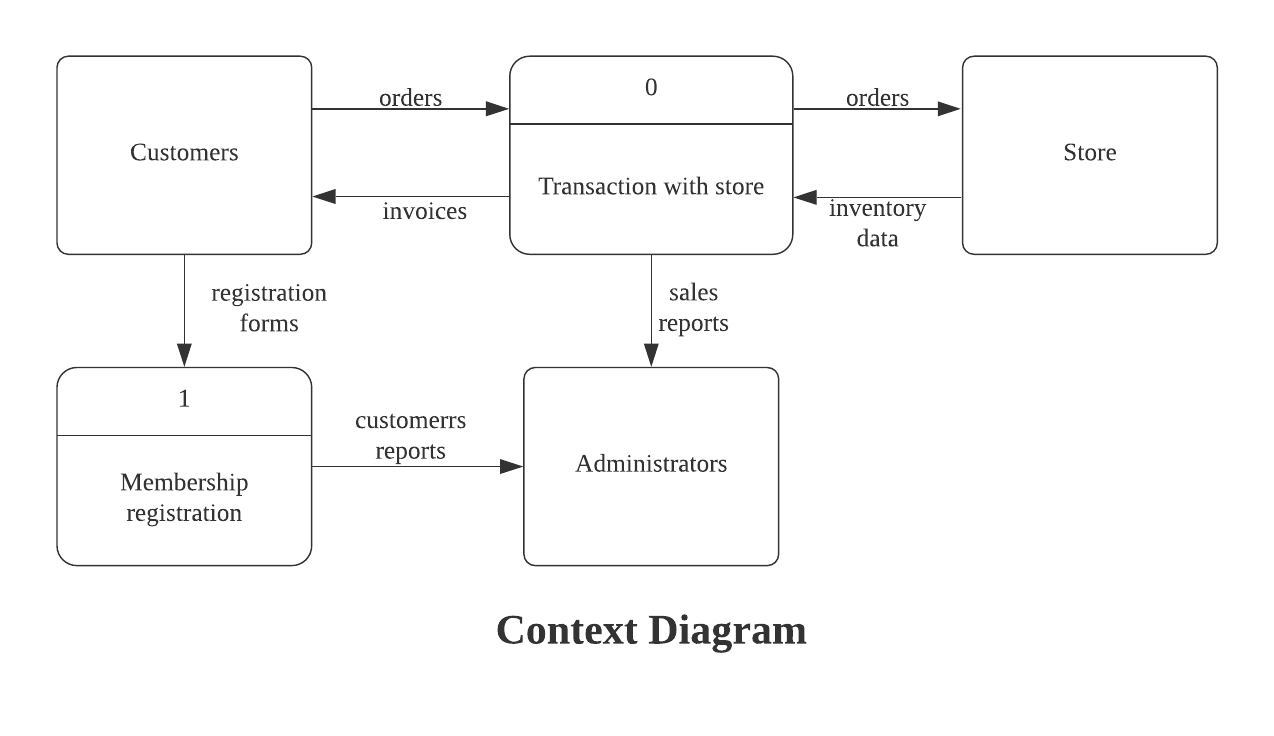
The “Manage books” use case includes 4 minor use cases, which are viewing books, adding books to the database, deleting books, and updating book’s information.

|  |  |  |
| --- | --- | --- |
| Use case’s name | Description | Priority |
| View books | The administrator will view the current stock levels and manually edit the stock levels to store and track stock of books for the online bookstore. | Medium |
| Add books | The administrator will be required to receive and add books to the stock lists. | Medium |
| Delete books | The administrator will delete books from the stock lists in case what books have been subtracted from the stock. | Medium |
| Update book’s information | The administrator will update the information of the book if there are any changes to it. | Low |

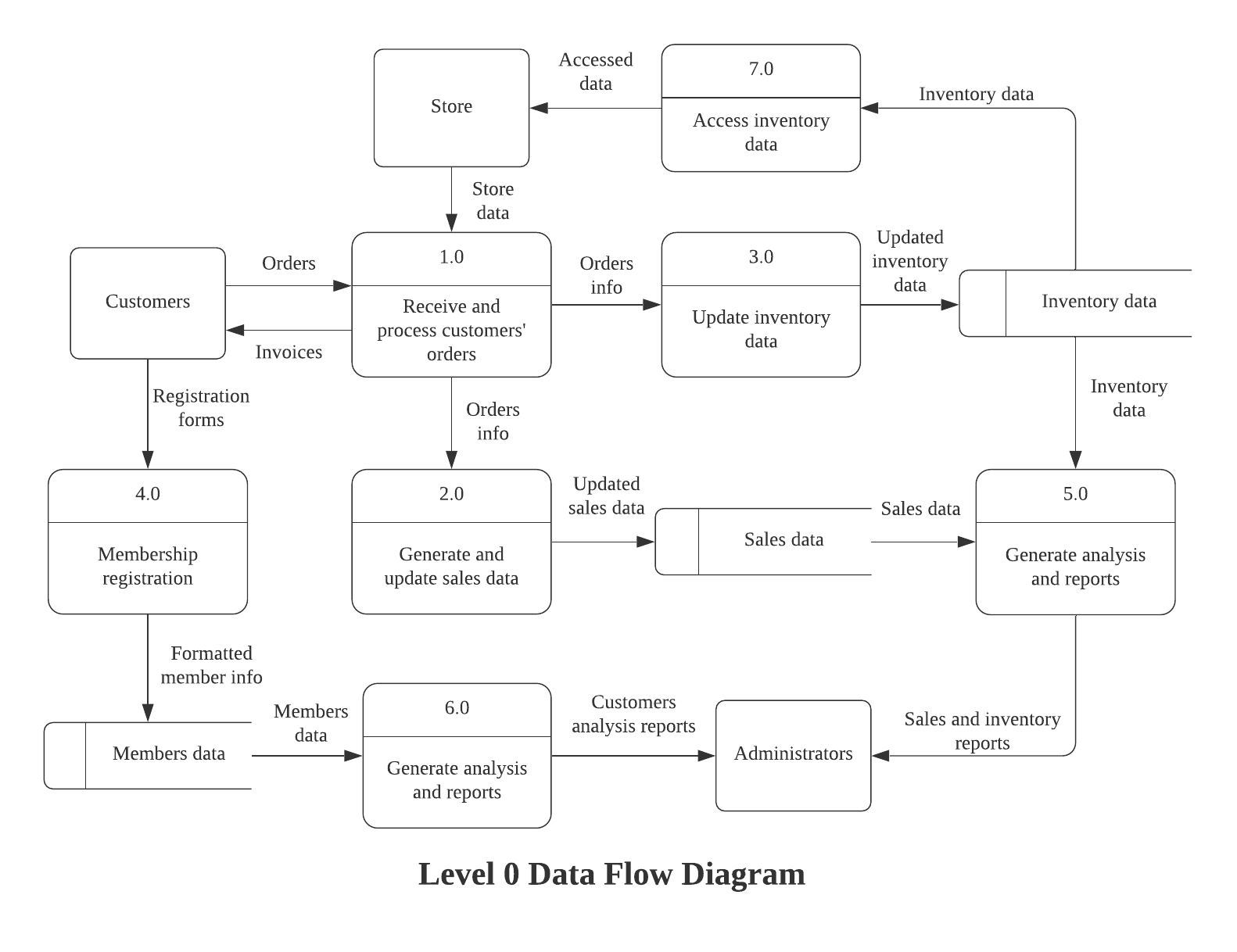
The “Manage member” use case includes 3 minor use cases, which are adding new member’s accounts, deleting member’s accounts and responding to member’s feedback on the product.

|  |  |  |
| --- | --- | --- |
| Use case’s name | Description | Priority |
| Add member’s accounts | When receiving information of the new registered account, the administrator will add it to the client repository. | Medium |
| Delete member’s accounts | The administrator will delete the member's account within the repository if required. | Medium |
| Respond to feedback | The administrator will answer any questions that the member posts on the product’s page. | Medium |

5. Data Flow

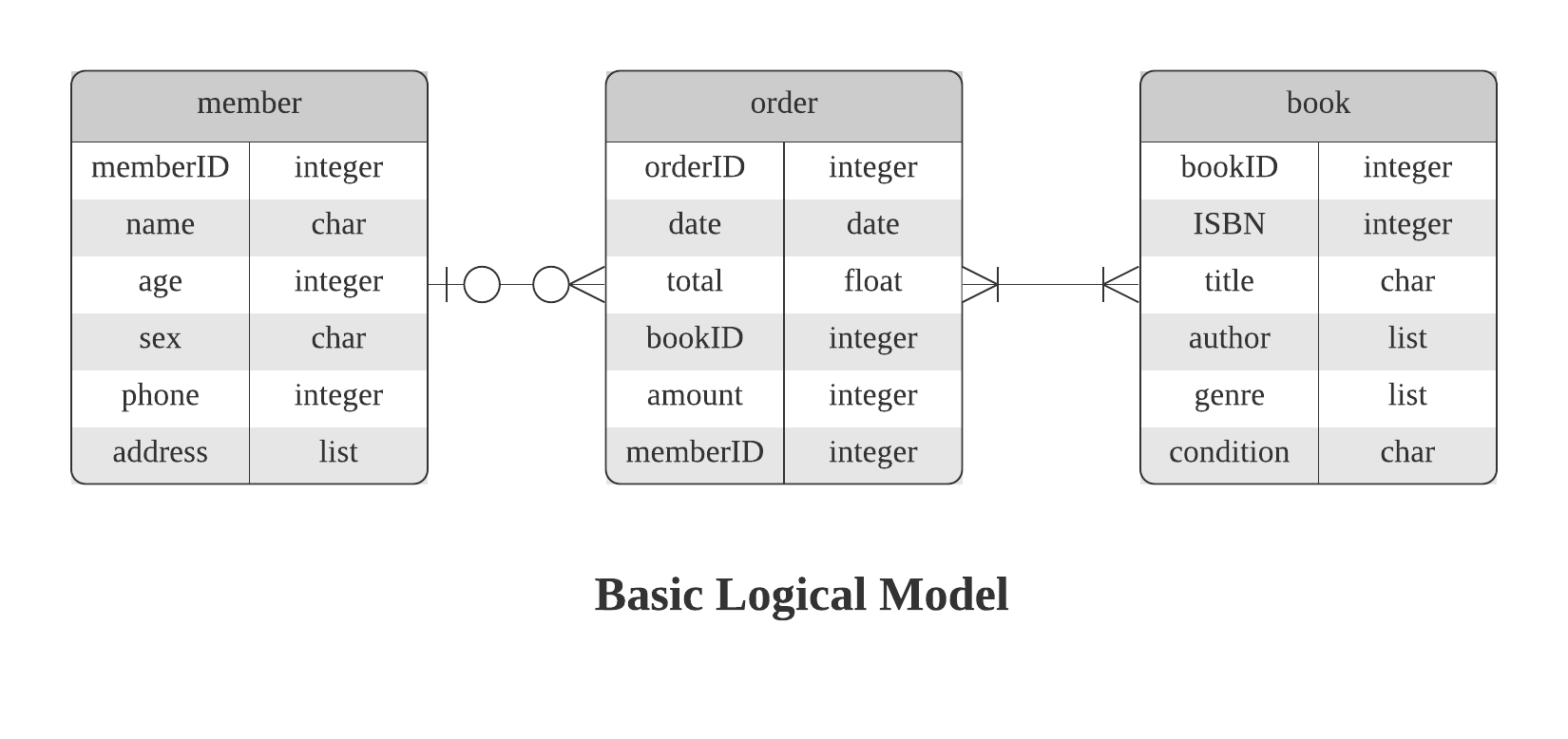


The *context diagram* explains the main flow of data which our business will mainly be collecting and dealing with. Since we are working directly with customers, most of our information will be coming directly from customers, more specifically their selling and purchasing orders. Once we get the orders, both selling and purchasing, we will begin to process the order in accordance with our database. In the case of online orders, we can forward those orders to our stores so we can preserve the items in that order for the customers to come pick up at a time of their convenience. After the process of purchasing or selling is completed, we would generate the receipts for the customers, this also means that information on the transactions will be collected and processed to be compatible for the format of our database. Afterwards, these sales data can then be easily query, search, or filter for other purposes. One of which is to use the data for more in-depth analysis of the current business performance as well as creating predictive models to help improve the company operations and make better business decisions. These sales reports and analysis will be available and accessible to the system administrators and the business managers as insights into the performance of the current business model. Apart from that, as explained from above, customers do have the option to sign up and become one of our members to enjoy more benefits when purchasing from or selling books to us. Through this process, we would need to collect customers information in order to create a membership for them. This information is crucial to our business operations as it will give us more understanding of our customers through analyses about segmentation and purchasing trends which will also be available to the administrators and business managers.



The level 0 data flow diagram can be decomposed into six smaller and more specific processes in the *decomposition figure* (Figure 6). As seen in the first process, we are dealing with our customers' orders, regardless of the type of order being for purchase or sell. Once we have acquired the order information, we would have to process it and access the inventory data to properly update it according to the order. The outcome of that process is the update inventory data that will then be used to generate the invoice for the customers as well as the sales data for our internal usage. And of course, the inventory data will be passed to the inventory so it would be updated properly. For the process 2 and process 3, they are quite similar as we are taking steps to validate our data before processing it and updating the sales as well as the inventory data accordingly. As for the fourth process, since we are dealing with customers’ information with regards to their membership registration, there are a few steps that have been listed to process and validate data before finalizing the registration process. Similarly, for process 5 and 6, here we are mainly querying our data for the purpose of analyzing and generating managerial reports about the performance of our business which will be presented to the business owners, business managers as well as the administrators.

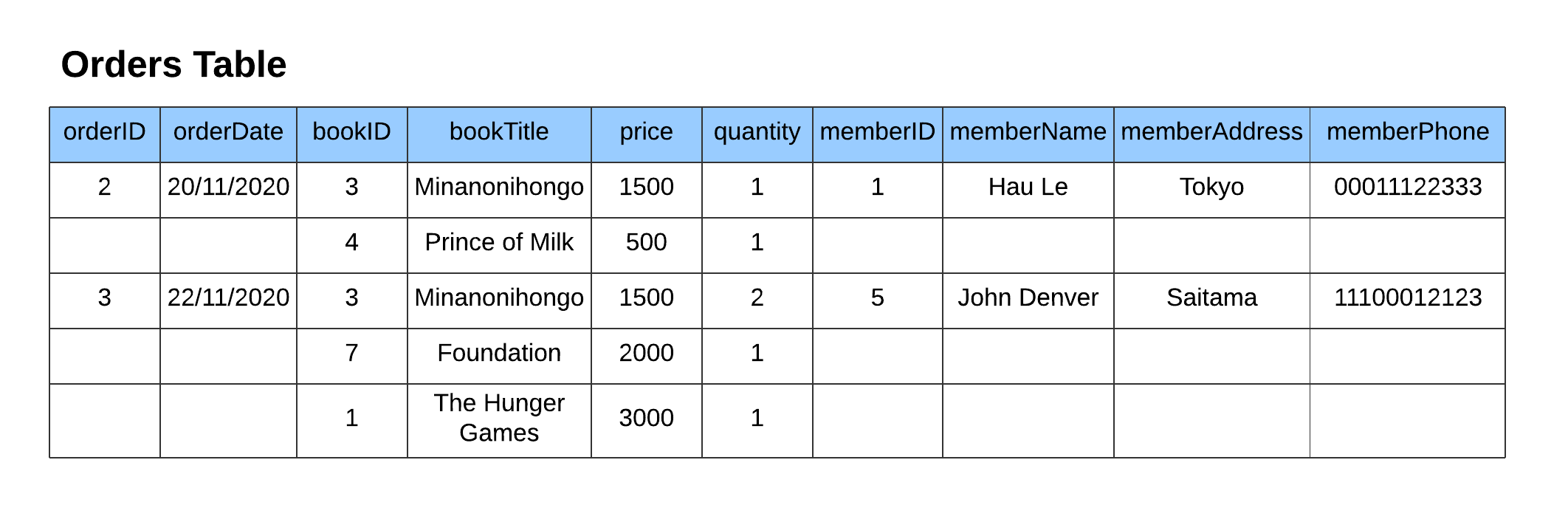
**Database Design**

****

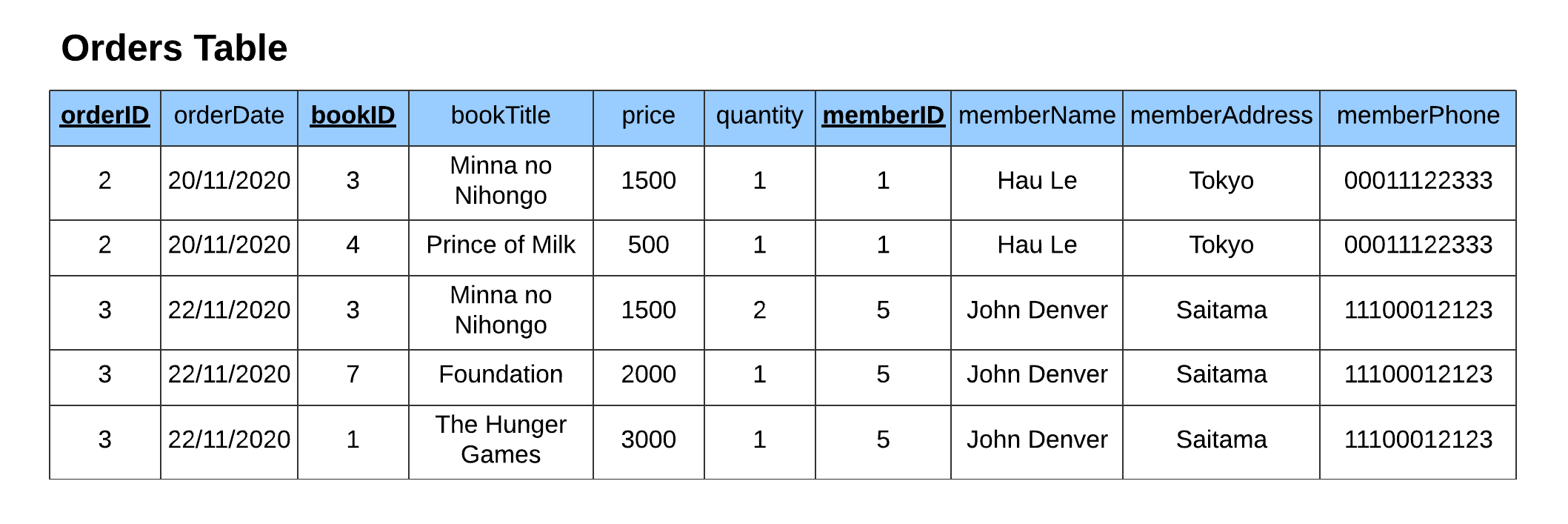
In the logical model, the relationships of our entities are as follows:

* A **member** *can have* 0 or many **orders**.
* An **order** *can have* 0 or 1 **member** included in it, since customers do not have to be members to order from us.
* An **order** *must have* at least 1 or many **books** included in it.
* A **book***must be included* in at least 1 or many **orders**.

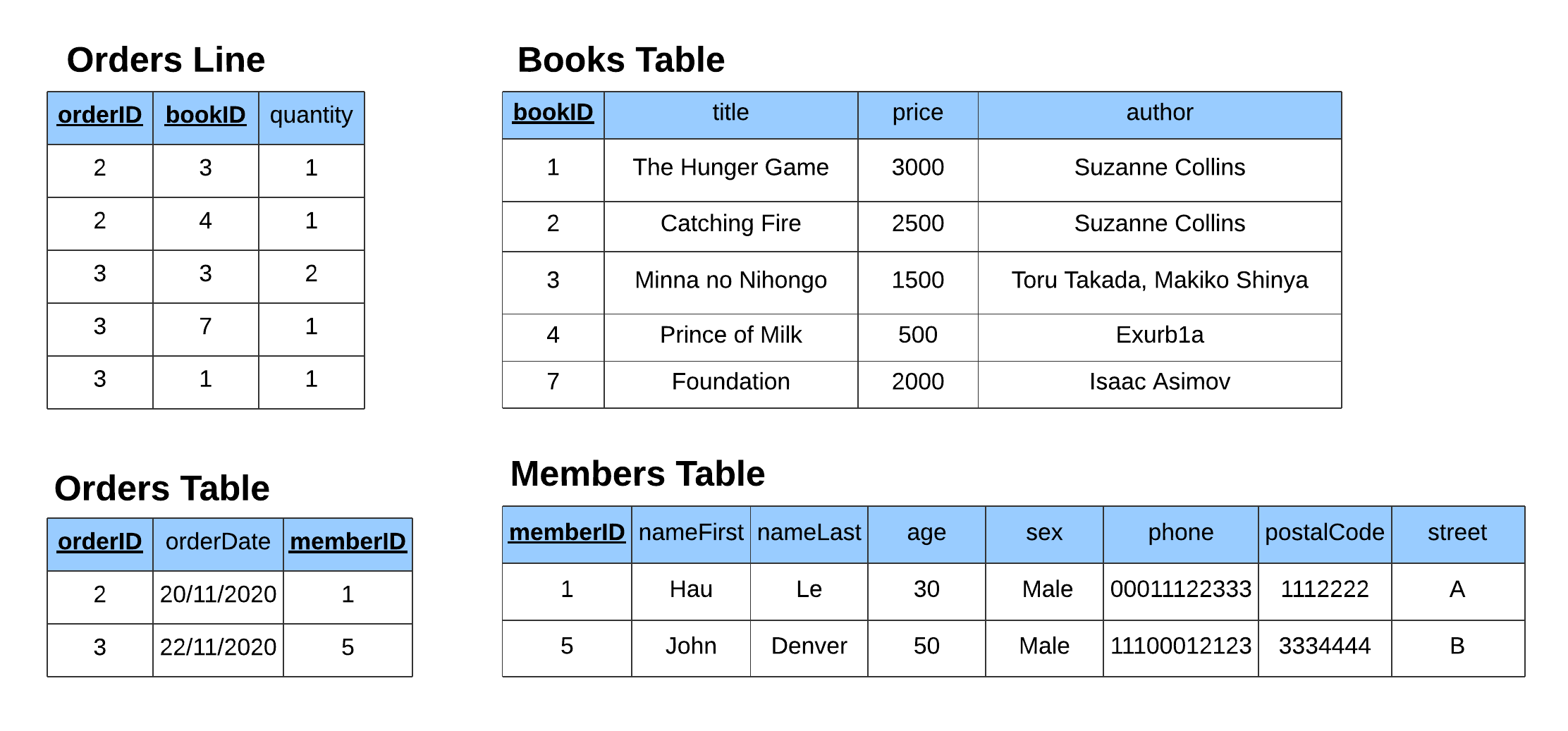
Moreover, the specific types of each attribute for each entry has also been included so it is more comprehensible when viewing the high-level model. On the surface, the model strictly follows our business requirements and operations hence it seems very logical. However, there are multiple problems with this model since it is violating multiple constraints on the implementation of it into a physical model. In order to address the problems, we will go through the process of normalization to turn our Entity-Relationship Diagram into a well-suited physical model that satisfies the system we are going to use, which is SQL. As an example, we will take a shortened version of the order table to carefully examine:

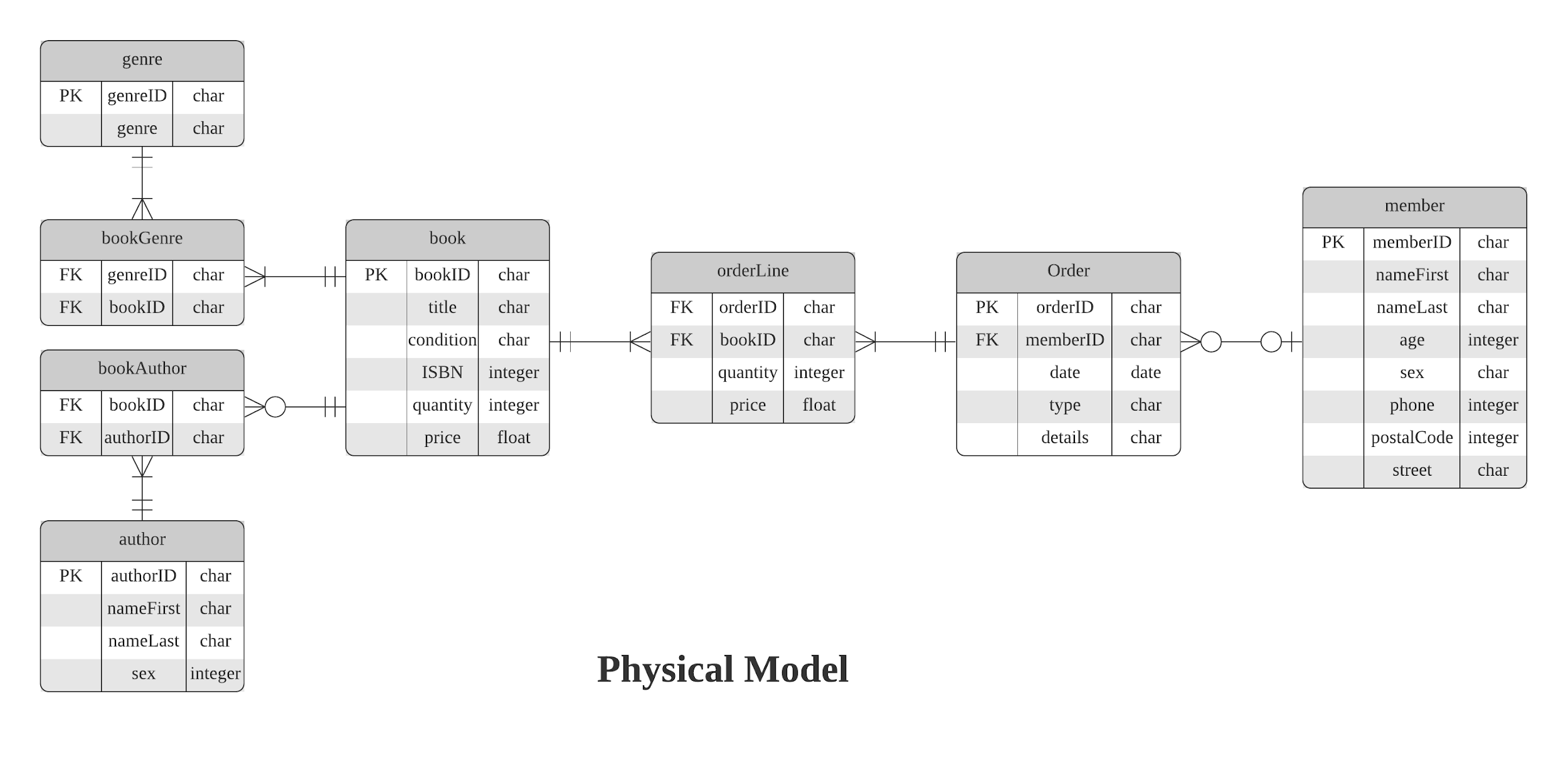


As explained above, an order must have at least one book in it but at the same time, it can also hold many books. Since we cannot limit our customers to only one title of a particular book for every transaction, an order in this example will have multiple books. It is very difficult to find out which books belong to which order since there are NULLs in our table. We can tackle this issue by filling out the rest of the missing information, turning the table into its first normal form (1NF) as shown below.



Now that the table is in 1NF, we can see that it is somewhat easier to keep track of orders and books as well as our members data. But there are still many problems with this kind of table. A slight modification in the name of a book entry would require the same modification to be updated across the whole table where that item exists. In this case, changing the name or the price of the book Minna no Nihongo would need an update in *orderID 2* as well as *orderID 3*, which will take a lot of time once our tables have grown larger or if we need to update multiple items. Another problem is that if we were to delete orderID 2, we would lose all information regarding *memberID 1* and *bookID 3, 4*. Or if we deleted a book entry, we would lose information regarding the orders where that book was included in. Additionally, if we need to add another book to an order then we would need to fill out all the information or there would be empty columns again. With all of these anomalies existing in this table form, we need to take another step of normalizing it as shown below.



We took two steps of normalizing our table from its 1NF to 2NF and 3NF by breaking them into smaller tables. In this form, if we need to update information on our members, we can adjust it directly in the *Members Table* and it does not require us to fill out information multiple times or at any other entries. The same principles can be applied to modification, insertion or deletion in any given tables. Once again, since this is just a subset of the fields available in our database, it does not fully represent our database design. We are facing similar anomalies as each book can have 0 or many authors and at the same time, an author must have at least 1 book or multiple books. This concept is also relevant in the book genres since a book can belong in multiple genres, and a genre can have multiple books included in it. With this in mind, the physical design of the model can be found below, tackling the aforementioned problems. 

The physical model here includes the relationship between each entity to make sure it is still logical and that it strictly follows the business requirements. The physical model can be briefly explained as follow:

* *member entity* has attributes such as *memberID* as the identifier followed by the name attributes being splitted into *nameFirst* and *nameLast* to avoid confusion in order of entry. Then it also contains *age*, *sex*, *phone*, *postalCode* and *street*. Each member can have 0 or multiple orders.
* *order entity* includes the unique identifier *orderID* and *date* of the order as well as the *type* of order, purchasing from or sellings to customers. Any additional optional details also can be included in each order entry. Additionally, an optional foreign key, *memberID*, can also be included to link to a particular member in the database. This is the case since the number of members in each order can be at minimum 0 and at maximum 1.
* *orderLine* entity tackles the problems of including multiple books into a single order. It contains compulsory foreign keys *orderID* and *bookID* as well as the *quantity* associated with the book and its *price*. *orderLine* entry should only contain a single *orderID* but a single *orderID* can be included in multiple *orderLine* entries. Likewise, each entry of *orderLine* should only include one and only one *bookID* while a *bookID* can appear multiple times or not at all in *orderLine*.
* *book* entity contains *bookID* as the unique identifier instead of the *ISBN* since the same book can have different *condition*. Condition here is referring to the level of damage to the books because we are dealing with secondhand books. It also includes the *quantity* that we have in stocks and the listing *price*. A *book* should appear in at least one or multiple *orderLine*, since a book is always associated with an *order*, regardless of the type of said order. At the same time, a book should always have at least one or many *genres*. In addition, a book should have at minimum 0 author and at maximum multiple authors.

|  |  |
| --- | --- |
| CREATE TABLE member (  memberID VARCHAR(10) NOT NULL,  nameFirst VARCHAR(20),  nameLast VARCHAR(20),  age INTEGER,  sex VARCHAR(2),  phone INTEGER,  email VARCHAR(30),  postalCode VARCHAR(7),  street VARCHAR(50),  CONSTRAINT pk\_member PRIMARY KEY (memberID)); | CREATE TABLE book (  bookID VARCHAR(10) NOT NULL,  title VARCHAR(20),  condition VARCHAR(1) NOT NULL,  ISBN VARCHAR(13),  quantity INTEGER,  price FLOAT(5),  CONSTRAINT pk\_book PRIMARY KEY (bookID)); |
| CREATE TABLE orders (  orderID VARCHAR(10) NOT NULL,  memberID VARCHAR(10) NOT NULL,  dates DATE,  type VARCHAR(3) NOT NULL,  details VARCHAR,  CONSTRAINT pk\_order PRIMARY KEY (orderID)); | CREATE TABLE author (  authorID VARCHAR(10) NOT NULL,  nameFirst VARCHAR(30),  nameLast VARCHAR(30),  sex VARCHAR(2),  CONSTRAINT pk\_author PRIMARY KEY (authorID)); |
| CREATE TABLE orderLine (  orderID VARCHAR(10) NOT NULL,  bookID VARCHAR(10) NOT NULL,  quantity INTEGER NOT NULL,  price INTEGER NOT NULL,  CONSTRAINT pk\_orderLine PRIMARY KEY (orderID, bookID)); | CREATE TABLE bookAuthor (  bookID VARCHAR(10) NOT NULL,  authorID VARCHAR(10) NOT NULL,  CONSTRAINT pk\_bookAuthor PRIMARY KEY (bookID, authorID)); |
| CREATE TABLE genre (  genreID VARCHAR(3) NOT NULL,  genre VARCHAR(30) NOT NULL,  CONSTRAINT pk\_genre PRIMARY KEY (genreID)); | CREATE TABLE bookGenre (  genreID VARCHAR(10) NOT NULL,  bookID VARCHAR(10) NOT NULL,  CONSTRAINT pk\_bookGenre PRIMARY KEY (genreID, bookID)); |

The table above shows the commands to create the database as well as the corresponding tables. Each table would specify the accepted data type for each column as well as their limited length.

|  |  |
| --- | --- |
| ALTER TABLE orders  ADD CONSTRAINT fk\_order\_member  FOREIGN KEY (memberID)  REFERENCES member (memberID); | ALTER TABLE bookGenre  ADD (  CONSTRAINT fk\_book\_genre  FOREIGN KEY (bookID)  REFERENCES book (bookID),  CONSTRAINT fk\_genre\_book  FOREIGN KEY (genreID)  REFERENCES genre (genreID)); |
| ALTER TABLE orderLine  ADD  CONSTRAINT fk\_order\_book  FOREIGN KEY (orderID)  REFERENCES orders (orderID),  CONSTRAINT fk\_book\_order  FOREIGN KEY (bookID)  REFERENCES book (bookID); | ALTER TABLE bookAuthor  ADD (  CONSTRAINT fk\_author\_book  FOREIGN KEY (authorID)  REFERENCES author (authorID),  CONSTRAINT fk\_book\_author  FOREIGN KEY (bookID)  REFERENCES book (bookID)); |

The table above shows commands which create the relation between tables in the database in order to maintain the structure of our database. Once we have successfully created the database, it is advised that we generate some dummy data to add into our database, which can be found in Table 2.

|  |  |
| --- | --- |
| **Query sales on each Genre** | SELECT genre.genre as Genres,  COUNT(CASE WHEN orders.type = 'SEL' THEN 1 END) as NumberofSales,  SUM(CASE WHEN orders.type = 'SEL' THEN orderLine.price END) as TotalSales,  COUNT(CASE WHEN orders.type = 'PUR' THEN 1 END) as NumberofPurchases,  SUM(CASE WHEN orders.type = 'PUR' THEN orderLine.price END) as TotalPurchases  FROM genre, bookGenre, orderLine, orders  WHERE genre.genreID = bookGenre.genreID  AND orderLine.bookID = bookGenre.bookID  GROUP BY genre.genre  ORDER BY genre.genre |
| **Query sales per member** | SELECT CONCAT(m.nameFirst, ' ', m.nameLast) as MemberName,  COUNT(CASE WHEN orders.type = 'SEL' THEN 1 END) as NumberofPurchases,  SUM(CASE WHEN orders.type = 'SEL' THEN orderLine.price END) as TotalSpent,  COUNT(CASE WHEN orders.type = 'PUR' THEN 1 END) as NumberofSales,  SUM(CASE WHEN orders.type = 'PUR' THEN orderLine.price END) as TotalGained  FROM orders, orderLine, member as m  WHERE m.memberID = orders.memberID  AND orderLine.orderID = orders.orderID  GROUP BY MemberName  ORDER BY MemberName |
| **Query order over a period of time** | SELECT \*  FROM orders as o  WHERE o.dates BETWEEN '2020-11-11' AND '2020-11-16'  ORDER BY o.dates |

The table above shows the commands to access the database as well as querying or selecting specific parts of the tables for further inspection. These are fairly basic commands, but they can immediately provide overall understanding of the data we are collecting. The result of these queries can be found in Figure 7, Figure 8, and Figure 9 respectively.

**Data Analysis**

Once we have established easy access to our database as well as any specific portion of it, we can perform multiple analyses on our queried data. But it is essential to build a robust Extract-Load-Transform ETL pipeline for the purpose of analysis, be it simple exploratory data analysis or predictive modelling. The main reason for this ETL process is because even though our database enforces more rigorous and strict inputs to maintain the structure of data, human mistakes and system errors are unavoidable. Within the data which is directly extracted from our database, also known as the raw data, there lies a few problems that we need to address before performing any forms of analysis. The raw data can and very likely will contain some missing or incomplete data, such as important fields on some records or even a portion of records themselves. This piece of information may be crucial for the kind of analysis that we want to move forwards with, such as datetime data for a time series analysis. Additionally, these missing data can also affect other models that we want to fit the data in to generate predictions. Similarly, there might be a lot of duplicated data which will change the significance of some features or fields in the model we are trying to improve or analyze. It can also create misleading information in visualization and analyses, and ultimately the more important managerial reports. Other problems with raw data could be the incorrect and inconsistent data, which will once again, create a lot of problems when we must deal with them. So, it is an important requirement to create a robust ETL pipeline to parse, validate, clean and finally split our data into training, testing, and validating sets for analyses. The cleaning process can be treated as two types, statistical cleaning and domain specific cleaning. Both can be performed manually if the data points which require correct are small or we can also automate the process as part of the ETL pipeline.

One of the analyses that was made available to us through the implementation of the database is members cluster analysis. For this analysis, we will require some additional attributes to ensure accuracy of the final result, which are the recency of purchase, the frequency of purchase and the total amount of purchasing transactions. The total amount of purchasing transactions of every member can be calculated by multiplying the quantity of book to its price for each purchase of the related member. The frequency of purchase is simply a count of all the purchases made by said member. And the recency attribute can be computed by taking the minimum of the difference between each member's last purchase date and the most recent date in the database. Then we would move forward with K-Means clustering as our method of choice because of its simplicity and ease of use as well as accuracy. To which, we will apply the elbow method to acquire the most optimal number of clusters. Once we have found the most optimal number of clusters, we will go ahead and fit our cleaned data into our K-Means model which will give us the corresponding cluster labels to each member ID. The next step is to cross-boxploting each cluster label with the respective attributes of recency, frequency and transactions to find out which group has the most frequent members, which group has the highest paying members and which group is most recent.

The next analysis that we can apply is regression analysis on our sales data, to find out which feature, or features have more significant impact on the overall sales of our business. There are two models here that we can build and test to see which one is more accurate or more informative for our business decisions. Both models use sales as the dependent variable. The first model would be based on our overall sales, so its independent variables are book genres, the conditions of the books, the authors information, book price range, the dates of orders. The second model would utilize our member data to generate a more refined model, which include the previous independent data with the addition of member data such as age group and genders. Before we move on with fitting our data into two respective models, it is wise to calculate variance inflation factor in order to detect any multicollinearity in our list of independent variables. This would ensure the better performance and accuracy of our model since we will effectively drop any independent variables that are highly correlated to one another. Once we have finished training our model, we will find the coefficients for each variable which will give us a better understanding of their impacts on our sales. We can also utilize this model to predict the near future sales of our business, but there is a better method for this particular task which is time series analysis.

For time series analysis, we can utilize the ARIMA (**A**utoreg**R**essive **I**ntegrated **M**oving **A**verage) model to perform time series forecasting with our sales data, the data that is time dependent. First, we need to set our date as the index of the data frame or the table that we are working directly with. But before we begin with any model, it is a prerequisite that we perform unit root and structural tests to ensure that our data is not stationary nor containing any structural break. Once we finish with that step, we can then extract the trend, seasonality and noise from our data with some visualization tool or statistically decomposing the seasonality. We also need to apply a quick grid search to find the best parameters for our ARIMA model then it would be fitted with the data to generate coefficients for each component of the model. Finally, if the model performed well and there were no unusual behaviors, we can use the model to validate our forecasting as well as apply the model to real data for some real-world future sales predictions.

**Implications**

There are 3 major beneficial part when the database is applied:

- Enforcing the standard characteristic and robustness

- Up-to-date technology, information and security

- In-depth analysis and reports, leading to improved business intelligence

Based on the data pipelines and scope of our company, we believe that having a database of inventory, member and sales data will optimize the process to be more efficient by ensuring the security and fluency and security of data input/output. For security, the database ensures the direct contact between administrators and customers in order to clear the inventory information and feedback. For fluency, the database prioritizes the order of use-cases from high to low, based on the order, the administrator and the system can address properly. The main function of the store is unchanged meanwhile customers now can easily search for inventory by accessing the main website and ordering the items. Transforming from manual records and storage to digital system by databases provide access swiftly and reduce consuming time to record data of company. For further updates for data or the website, the change is easier because the system is now digital, and all the change is updated through the computer. Based on the analytical part above, the cluster analyses allow us to separate and classify the group of sales. For example, with cluster analyses, we can realize which book’s genre has the most frequency and what not and how to improve them. We also use regression to calculate and predict the trend, seasonality of items. In this case, the multiple is used with the variables of genre, condition, date’s release, price, … This will give us a better understanding of their impacts on our sales, if the outcome has any problems or wanting to improve the outcome, we will have the knowledge as well as the tools to make appropriate changes. All in all, we wholeheartedly believe that the integration of this database would be the right next step for OLD&SOLD, enabling better business operations and opening up more opportunities to grow and expand.

**Appendix**

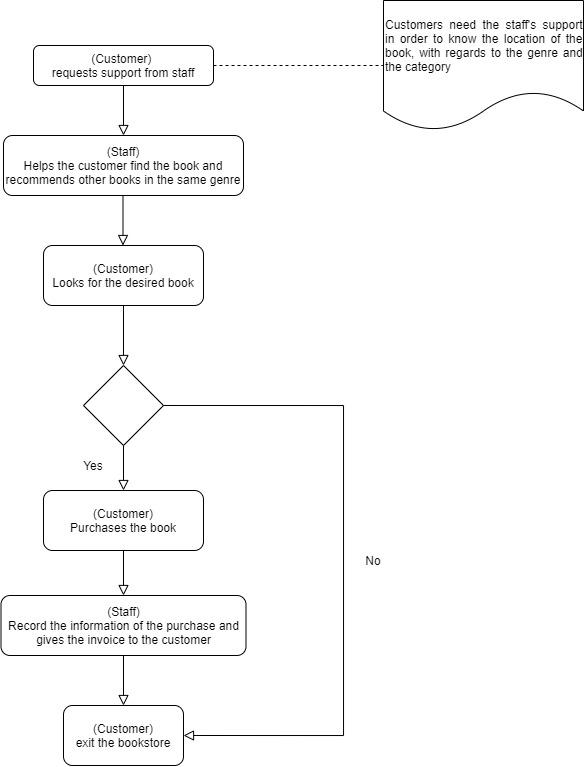
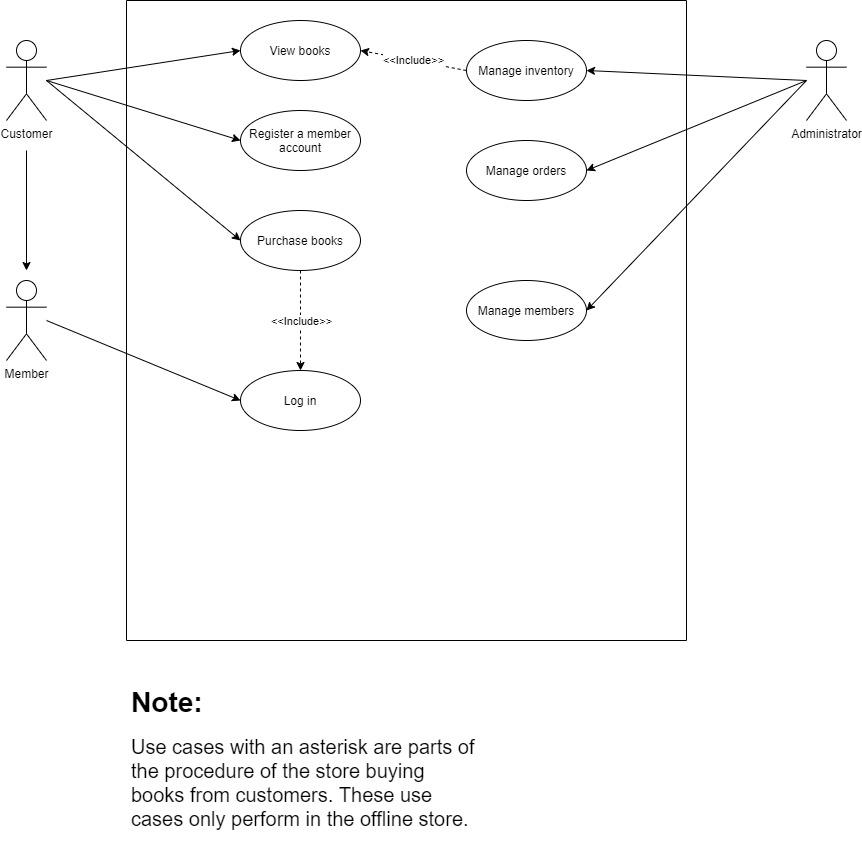


Figure 1: The process of customer purchasing books in the offline store.

Figure 2: High-level use case diagram 

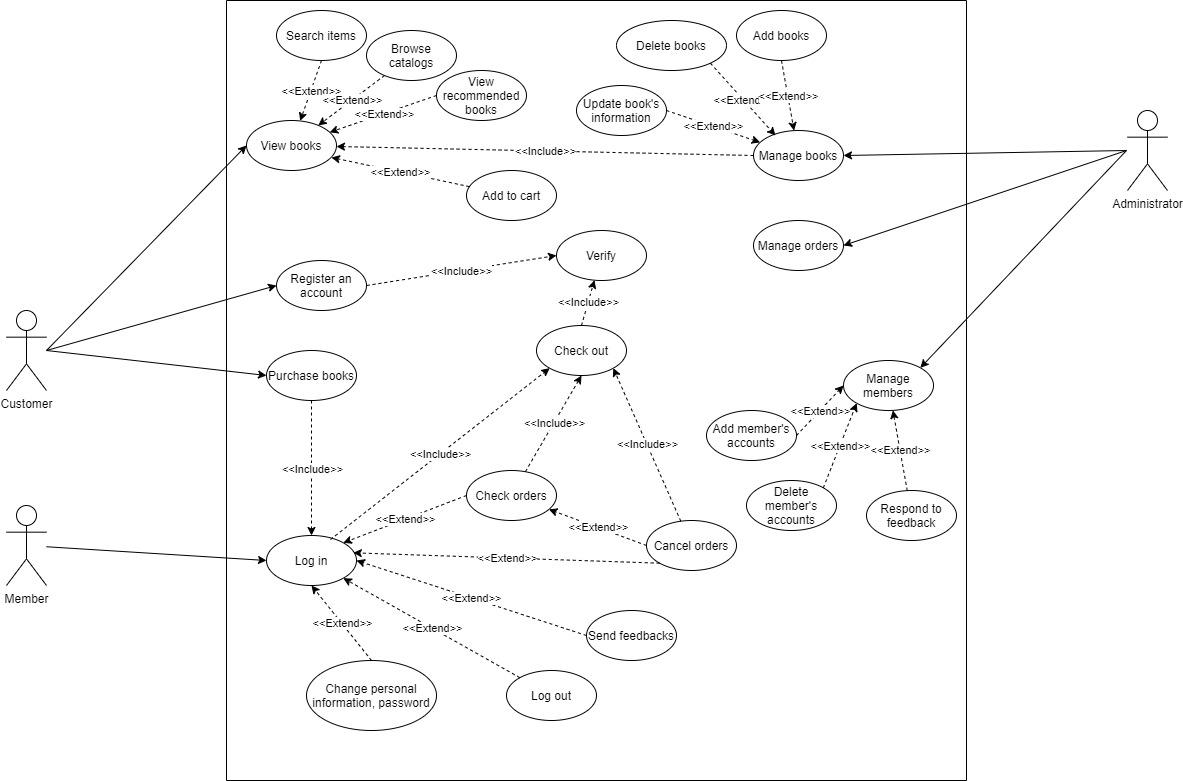


Figure 3: Customer, member and administrator interact with the online bookstore’s system during the process of customer and member purchasing books online.

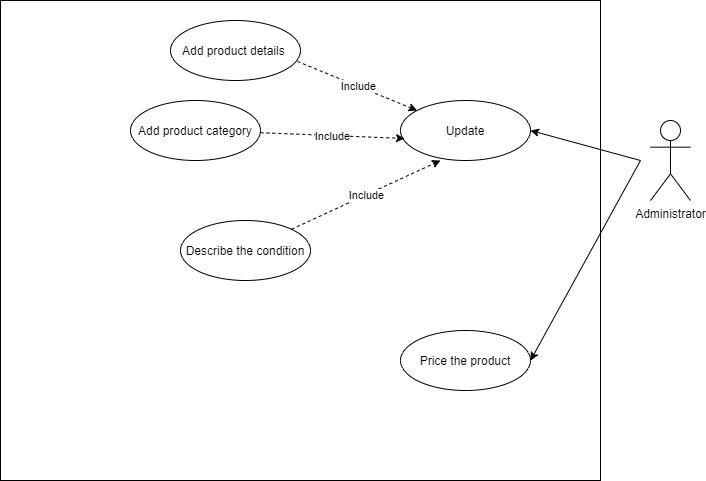


Figure 4: The administrator interacts with the system during the process of the store buying used books from customers.

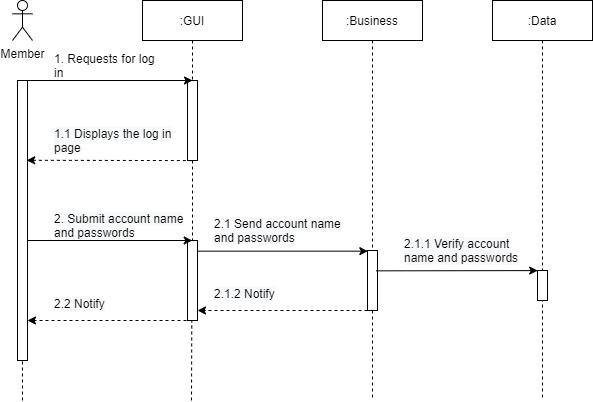


Figure 5: The process of member logging in the online bookstore’s system

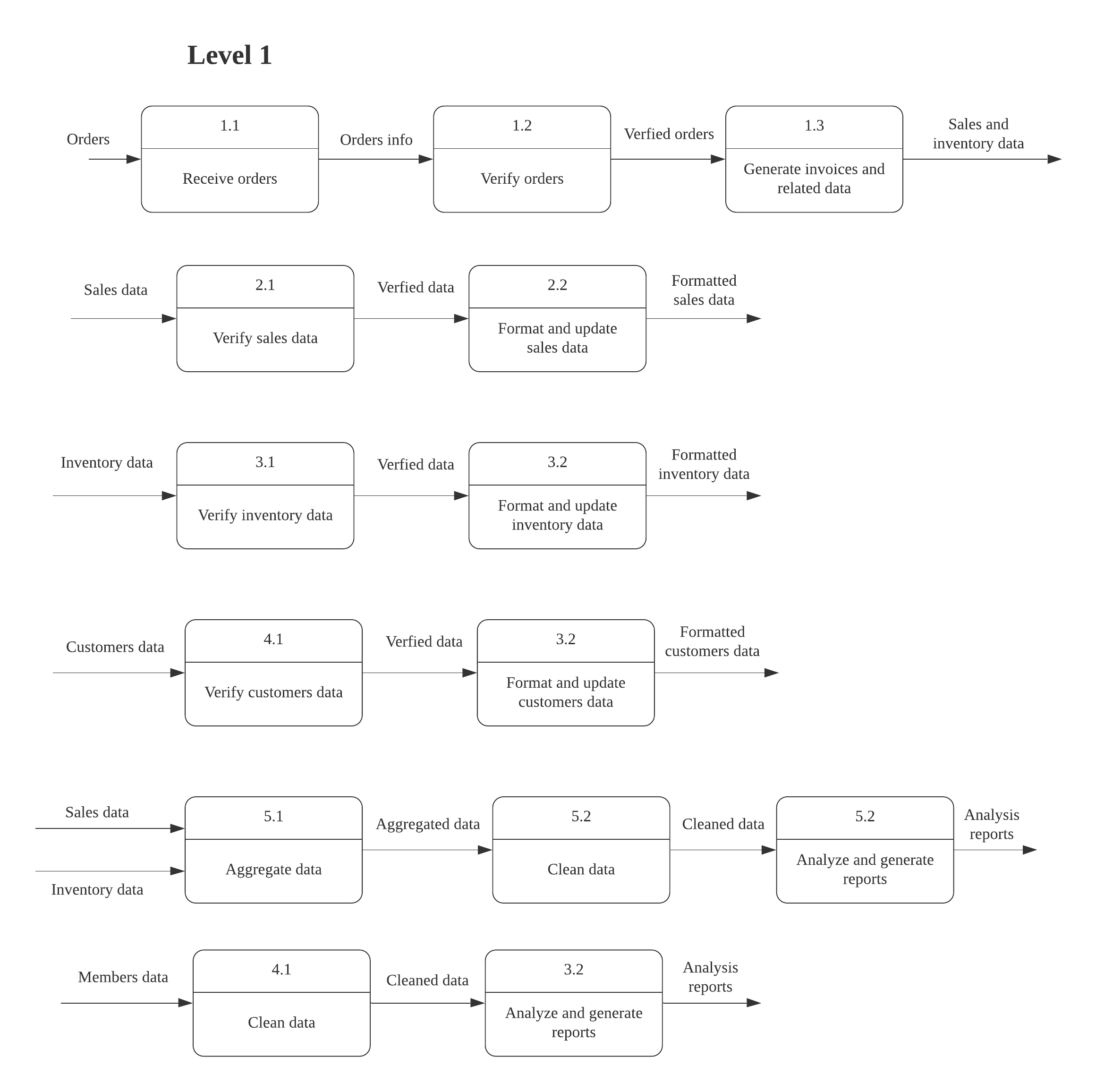


Figure 6. Decomposed processes in the data flow diagram

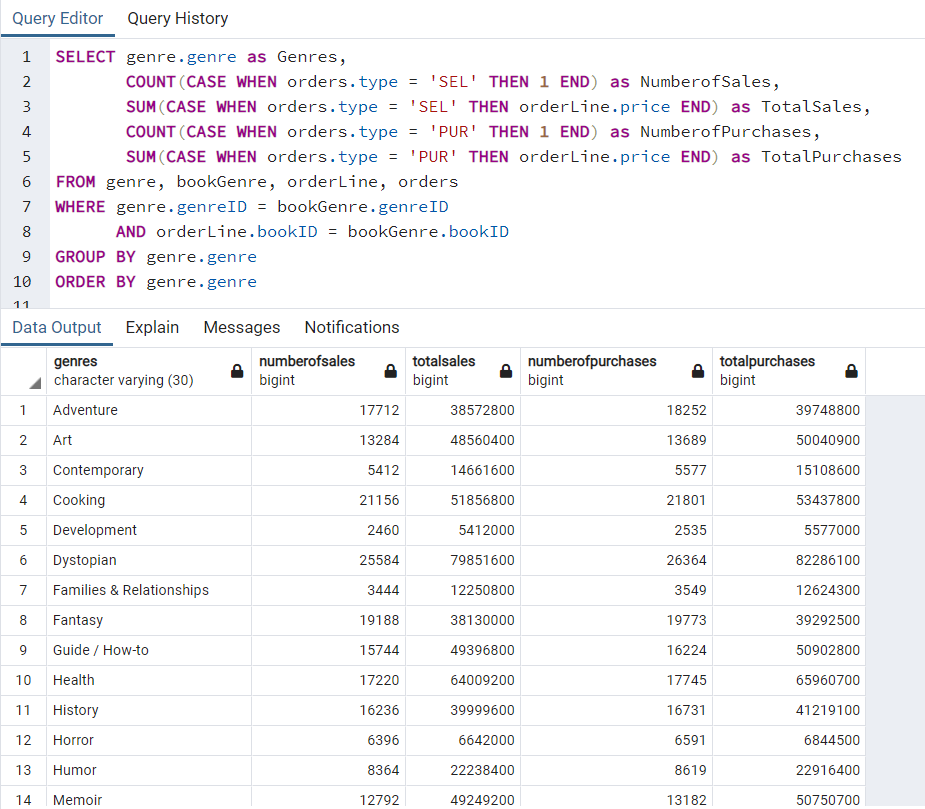


Figure 7. Sales and purchases queried by genres

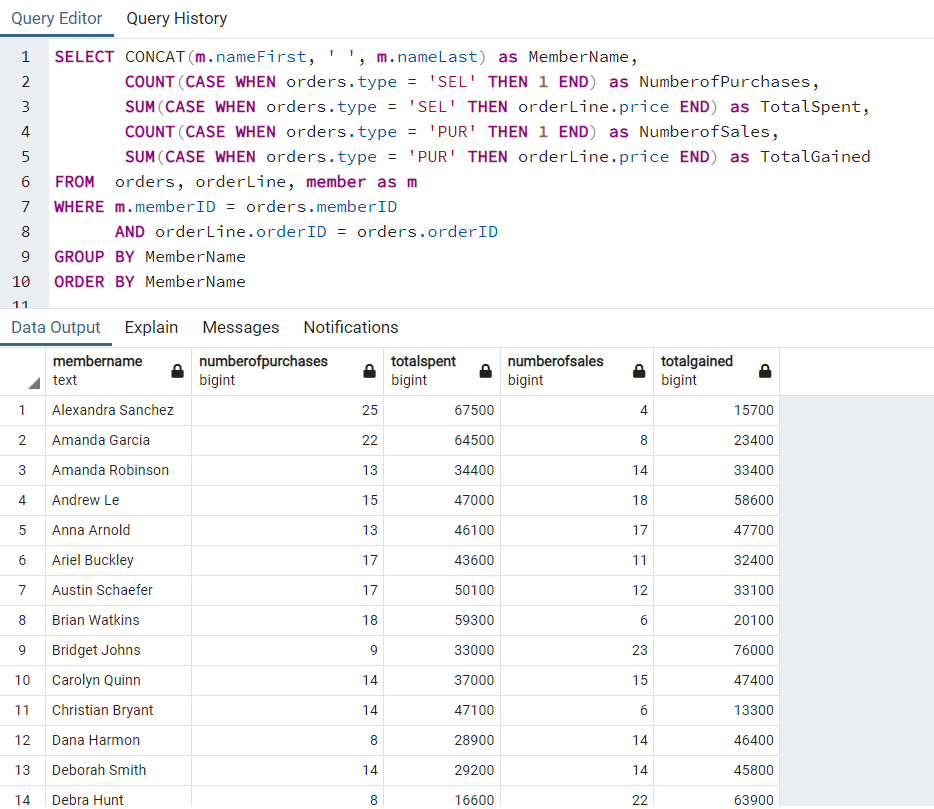


Figure 8. Sales and purchases queried by members.

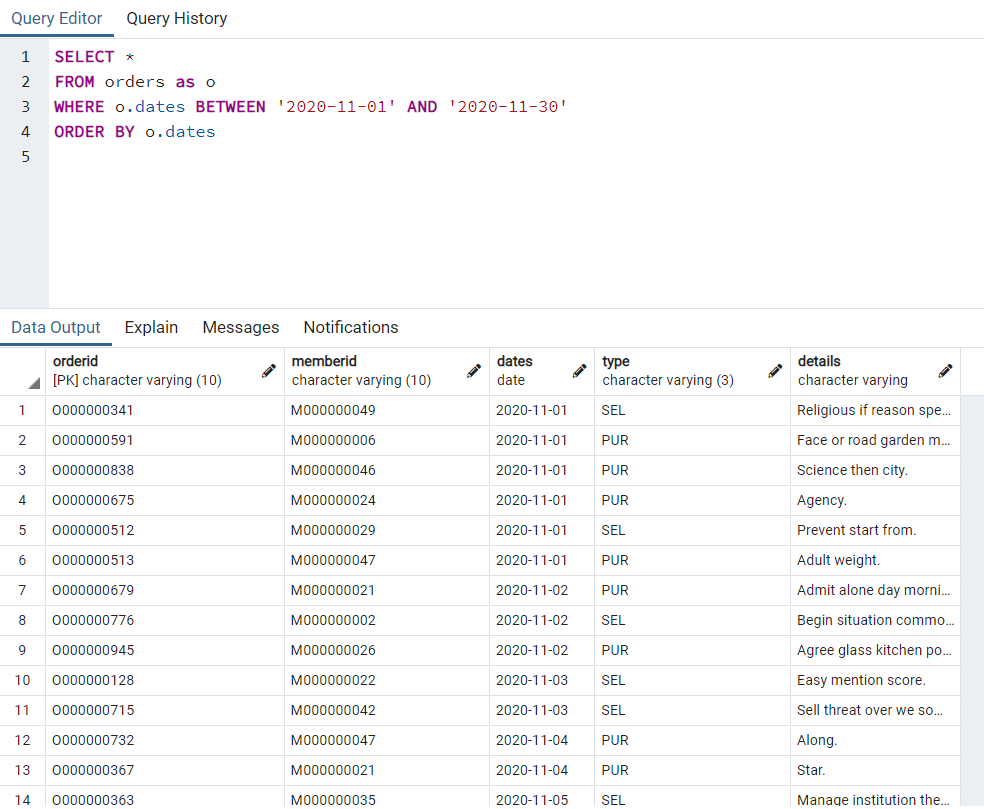


Figure 9. Transactions within this month

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Use case name | Ranking criteria | | | | | | Total score | Priority |
|  | 1 | 2 | 3 | 4 | 5 | 6 |  |  |
| View books | 5 | 5 | 5 | 4 | 5 | 5 | 29 | High |
| Register an account | 3 | 4 | 3 | 3 | 3 | 3 | 19 | Medium |
| Purchase books | 5 | 5 | 5 | 5 | 5 | 5 | 30 | High |
| Log in | 2 | 4 | 2 | 2 | 3 | 2 | 15 | Medium |
| Manage books | 4 | 4 | 5 | 5 | 5 | 5 | 28 | High |
| Manage orders | 5 | 4 | 5 | 5 | 5 | 5 | 29 | High |
| Manage members | 3 | 4 | 3 | 3 | 4 | 3 | 20 | Medium |

Table 1: Priority Matrix table for the “selling books to customer” use case diagram (Figure 3)

|  |
| --- |
| INSERT INTO book VALUES ('B000000001', 'The Hunger Game', 'A', '9780439023481', 1, 3000);  INSERT INTO book VALUES ('B000000002', 'Minna no Nihongo', 'A', '9780439023483', 3, 1500);  INSERT INTO book VALUES ('B000000003', 'Catching Fire', 'A', '9780439023481', 2, 3000);  INSERT INTO book VALUES ('B000000004', 'Foundation', 'B', '9780439023481', 1, 2000);  INSERT INTO book VALUES ('B000000005', 'Prince of Milk', 'C', '9780439023481', 1, 5); |
| INSERT INTO member VALUES ('M000000001', 'Hau', 'Le', 20, NULL, NULL, NULL, NULL);  INSERT INTO member VALUES ('M000000002', 'John', 'Denver', 50, NULL, NULL, NULL, NULL); |
| INSERT INTO orders VALUES ('S000000001', 'M000000001', NULL, 'SEL', NULL);  INSERT INTO orders VALUES ('S000000002', 'M000000002', NULL, 'SEL', NULL);  INSERT INTO orders VALUES ('S000000003', 'M000000001', NULL, 'SEL', NULL);  INSERT INTO orders VALUES ('S000000004', 'M000000002', NULL, 'SEL', NULL);  INSERT INTO orders VALUES ('S000000005', 'M000000002', NULL, 'PUR', NULL);  INSERT INTO orders VALUES ('S000000006', 'M000000002', NULL, 'PUR', NULL); |
| INSERT INTO orderLine VALUES ('S000000001', 'B000000001', 1, 3000);  INSERT INTO orderLine VALUES ('S000000001', 'B000000002', 1, 1500);  INSERT INTO orderLine VALUES ('S000000002', 'B000000003', 1, 3000);  INSERT INTO orderLine VALUES ('S000000003', 'B000000005', 1, 500); |
| INSERT INTO genre VALUES ('ACT', 'Action');  INSERT INTO genre VALUES ('EDU', 'Education');  INSERT INTO genre VALUES ('SCF', 'Sci-fi'); |
| INSERT INTO bookGenre VALUES ('ACT', 'B000000001');  INSERT INTO bookGenre VALUES ('ACT', 'B000000003');  INSERT INTO bookGenre VALUES ('EDU', 'B000000002');  INSERT INTO bookGenre VALUES ('SCF', 'B000000005'); |

Table 2: Pseudo-codes demonstrating insertion of dummy values into the database