University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

Essay / Assignment Title:Designing A Database System for A stock Exchange Market

Programme title:Enterprise Data Warehouses and Database Management Systems

Name:Eniola Bashirah Uthman

Year:03/06/2023

TABLE OF CONTENTS

Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people texts I clearly declare it through the good use of references following academic ethics.

(In the case that is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

....ENIOLA UTHMAN...

Date: ..JUNE......................./.....3RD...../....2023.....

CHAPTER 1

1.0 INTRODUCTION

Almost all firms use data warehouses for decision support systems as well as strategic business intelligence tools. It aids in the systematic organization, storage, and analysis of massive amounts of daily transactional data, which differs greatly from operational database systems.

In this paper, a database management system for a stock exchange market is created to display all entities and attributes as well as their relationships.

The stock market is a secondary market in which shares or stocks of various listed firms are traded. The stock market allows corporations to raise capital by selling shares to the public, and investors to profit from the increase in value of those shares over time. Stock prices fluctuate during the day and over time due to a variety of variables.

A DBMS is used in a stock market because it provides the necessary infrastructure and features to handle complex and critical data management requirements of the market, ensuring data integrity, security, and efficient operations. By utilizing a DBMS, stock markets can efficiently manage and process large volumes of data, this helps in facilitating smooth and reliable trading activities, decision-making, and regulatory compliance.

When creating a database system for a stock exchange market, several elements are considered to make up the structure of the system. These are Entities, Relationships, Database tables, indexes, as well as attributes such stock price, quantity, ClientID, timestamp etc. These elements are explained below:

1.1. ENTITIES

These are the main objects that are stored in the database. These include the stock, Stock History, Client and Transactions.

- **Stocks:** This entity will store information about all the stocks available in markets. Its attributes includes; StockID, StockName, CurrentPrice ,ISIN.StockID is the unique identifier for each stock, Name is the name of the company associated with the stock, ISIN which stands for international securities identifier Number, is a unique 12 character alphanumeric code used to identify specific securities,such as stocks,bonds, and other financial instruments, in the global marketplace.

- **Client:** This entity stores the information about all the clients who registered with the stock exchange. Its attributes include ClientID, Name, and AccountBalance. ClientID is the unique identifier for each client, while Name stores their personal information. AccountBalance stores the current balance available in a client's account.
- **Stock History:** This entity stores each buy and sell order placed by clients. It attributes includes StockID,Pice and Date_Time. StockID is the foreign key referencing the respective stock entities. Price stores the quantity of the stock and price per share respectively.
- **Transactions:** This entity keeps track of every transactions made in the stock exchange.It has the following attributes: TransID, ClientID, StockID, Price, Quantity, and DateTime.ClientID and StockID are foreign keys that reference the respective client and stock entities, whereas TransID is the unique identifier for each deal. Quantity and price store the quantity of stock and price per share for the transaction, respectively, while Date_Time stores the date and time the trade is executed.

1.2 RELATIONSHIPS

The relationships here in this work describe the associations between the entities, which are typically represented by foreign keys. These relationships include many -to-many, one -to-many, and many-to-one relationship.

1.3 ATTRIBUTES

This database refers to the properties of the entities stored in the database, and these include stock price, Client ID, Transaction Type, Transaction ID, quantity, stock name, ISIN, stock ID, Account balance, Stock History ID, and time stamp.

1.4 TYPE OF DATABASE MANAGEMENT SYSTEM USED

In this work, the Relational Database Management System (RDMS) is utilized to create a database system for the stock exchange market. Because the system is structured and well-defined, and there are clearer relationships between the various units. To facilitate effective querying and analysis, the data must also be kept in a consistent and orderly manner. Using SQL queries, an RDMS enables simple data management, maintenance, and retrieval. It also supports ACID transactions, which ensure data integrity and consistency in a financial system such as a stock exchange market. This is crucial in a stock exchange market where accurate and consistent data is necessary for transactions and historical analysis.

CHAPTER 2(USING ENTITY RELATIONSHIP DIAGRAM TO ILLUSTRATE THE RELATIONSHIP BETWEEN ENTITIES)

This section will provide an ER diagram to show the relationship between the entities of the stock exchange market.

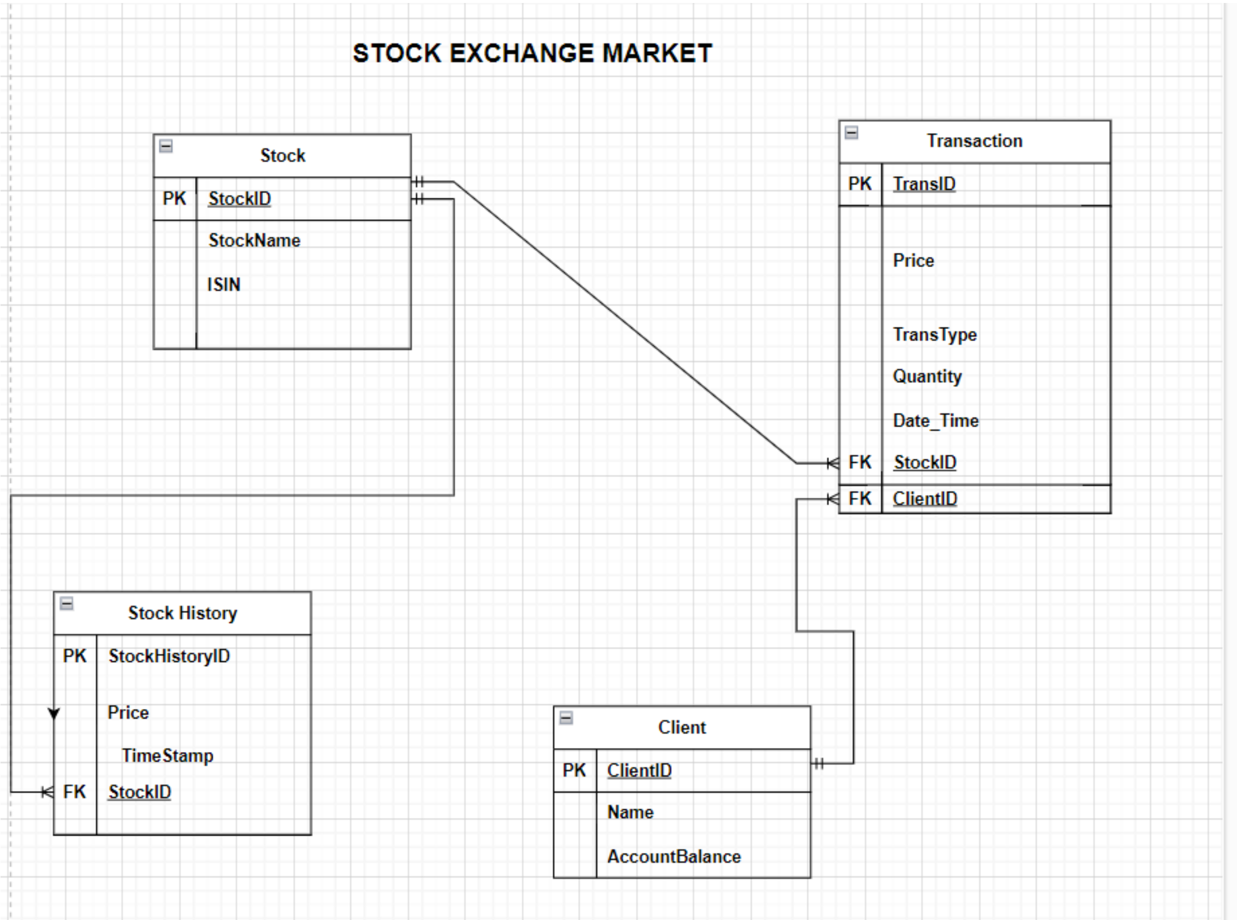
2.1 ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram (ERD) is a graphical representation of a relationship between people, objects, locations, concepts, or events and information technology (IT). An ERD employs data modeling approaches to aid in the definition of business processes and to serve as the foundation for a rational database.

Since 176, P.P.Chen founded the ERD during his researches , the ERD was considered a very good way of to do a conceptual model for databases. ERD is a graphical way to give information about databases in the system.Using ERD, database designers can convert this information to build database tables. Information that has been obtained about database helps to reach relational database schema.

For this paper, the ER diagram for the stock exchange marker is shown below.

Figure 2.1



2.2 THE ENTITIES OF THE ER DIAGRAM

Stock

This entity represents a stock available in the stock exchange market. It has the following attributes:

• Stock ID: A unique identification assigned to each stock.

• Stock Name: The name or symbol that relates to the stock.

• ISIN:The ISIN (International securities identification Numbering system).This a 12 digit alphanumeric code that uniquely identifies a specific security.

Stock History

This stores the historical prices of each stores.It has the following attributes:

- StockHistoryID: A unique identifier for each customer and it is the primary key of the stock history table.
- Stock ID: A reference to the stock entity's Stock ID, establishing a relationship between the two entities.
- Price: the price of the stock at a specific point in time.
- Date_Time: The date and time when the price was recorded.

Client

This entity represents a client who participates in the stock exchange market. It has the following attributes:

• ClientID: A unique identifier for each customer

• Name: The client's name; and

• Account Balance: The amount of money available in the client's account for stock purchases and sales.

Transactions

This entity keeps track of the purchases and sales made by clients. It has the following characteristics:

- Transaction ID: An identifier that is unique to each transaction.
- ClientID: A reference to the Client ID of the client entity, which links the transaction to a specific client.
- Price: The price at which the stock was purchased or sold.
- Date_Time: The date and time the transaction took place.
- StockID: A reference to the stock entity's stock ID, indicating the stock in question.
- Quantity: The number of stocks purchased or sold in the transaction.

- Transaction Type: This indicates whether the transaction is a buy or sale transaction.

2.3 THE RELATIONSHIP OF ENTITIES IN ER DIAGRAM

The relationship of the ER diagram is summarized below:

- Clients have a one-to-many relationship with Transactions.One client can have multiple transactions,but each transaction belongs to a specific client.
- Stocks have a one-to may-relationship with stock History.One stock can have multiple historical price records,but each historical price record to a specific stock.
- Stocks History have a many-to-one relationship with stocks.Multiple historical price records belong to a specific stock.
- Transactions have a many -to-one-relationship with both clients and stocks.Multiple transactions can be associated with specific stock and client.

CHAPTER 3

In this chapter, the ER Diagram will be converted to collection of possible SCHEMAS and tables will be created based on SCHEMAS.

3.1 CONVERTING THE ER DIAGRAM TO SCHEMA

Based on the figure 2.1, the following can be created to form a SCHEMA for the stock market database:

- Stock- StockID(PK),StockName, ISIN
- Transaction- TransID(PK),Price,TransType,Quantity,Date_Time ,StockID(FK),ClientID(FK).
- Stock History-StockHistoryID(PK),StockID(FK),price,Date_Stime
- Client-ClientID (PK),Name,AccountBalance.

3.2 REASONS FOR THE FOREIGN KEYS AND PRIMARY KEYS IN THE SCHEMAS CREATED

The primary keys (PK) in the schemas offered uniquely identify each entry in a table or collection.They serve as the primary identification for entities in the database. The primary key for the Stocks Table is Stock ID, the primary key for the Clients Table is Client ID, the primary key for the stock History is the StockHistoryID and the primary key for the Transactions Table is Transaction ID. The primary key enables easy access to specific records and the establishment of associations with foreign keys.

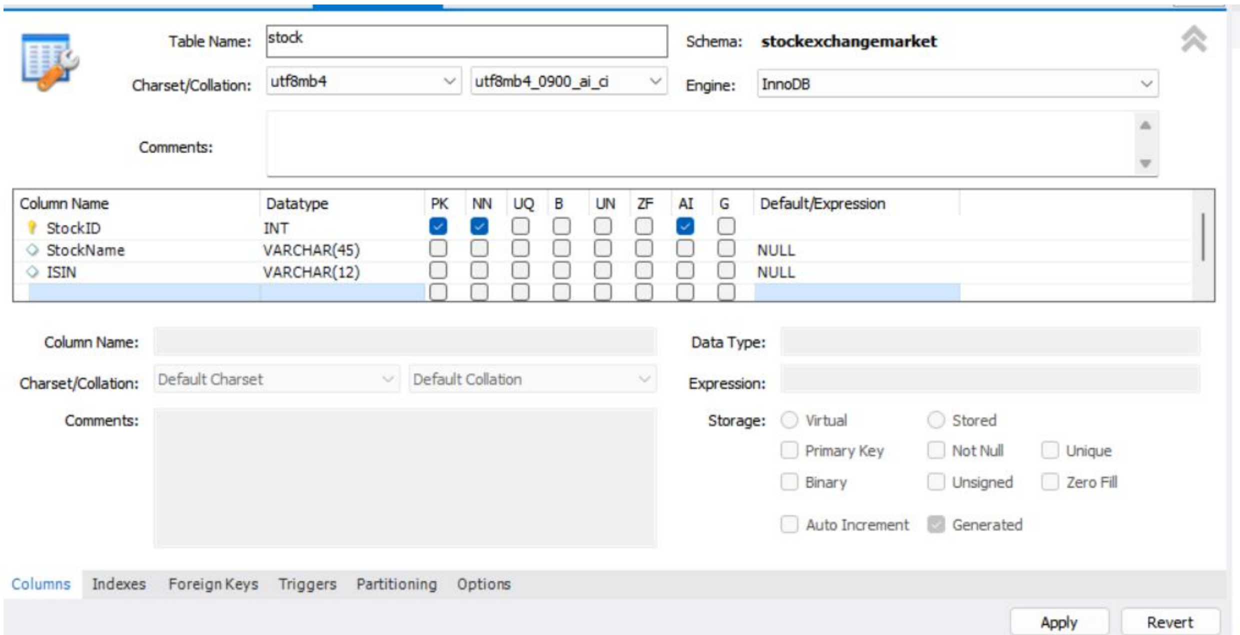
For the Foreign keys(FK) in the schemas above, they are used to create relationships between entities.They establish referential integrity,enforcing that values in a foreign key in one table must correspond to the values in another table’s primary key .The Stock ID is a foreign key in the stock history table because, the stock history table is referencing it (Stock ID) from the stocks table to associate the historical prices.

3.3 CREATING TABLES USING MYSQL DBMS

The tables created using MYSQL workbench are shown below:

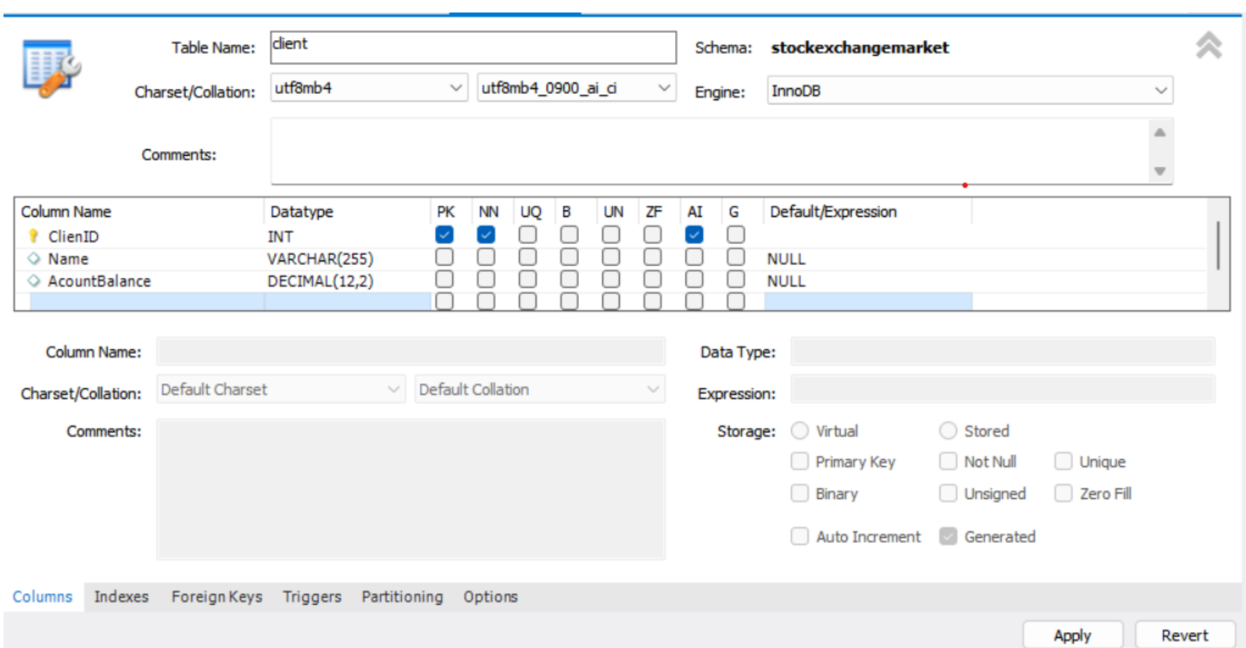
- The stock table shown below stores information about different stocks available in the market

STOCK TABLE



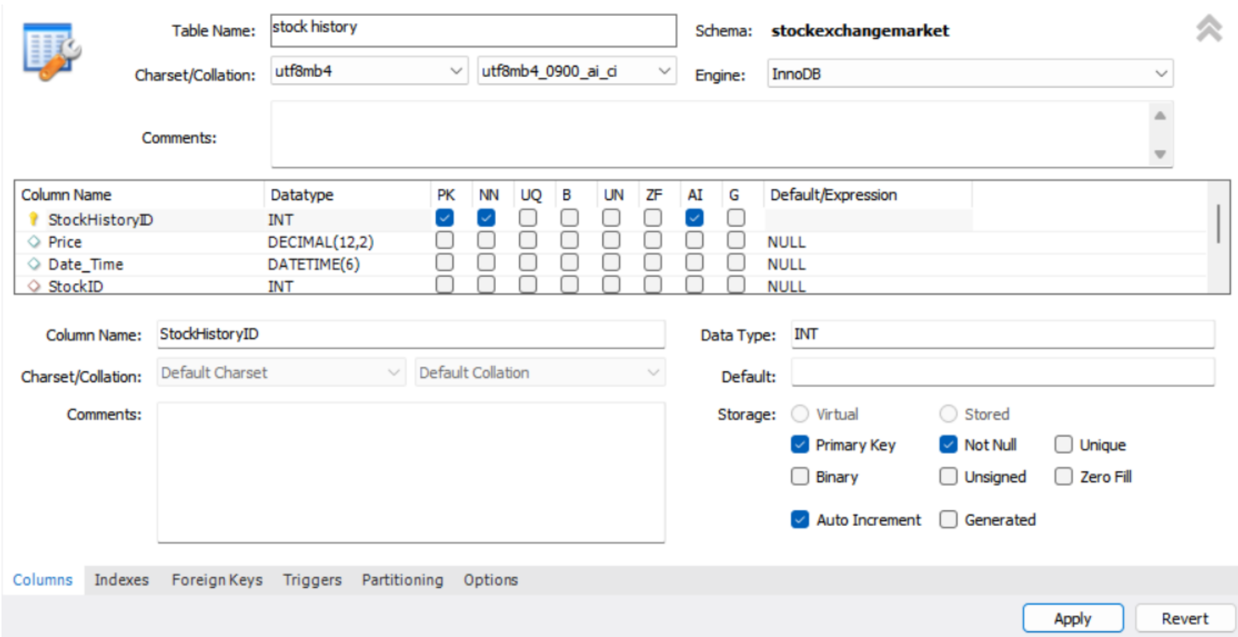
- The client table shown below stores information about clients or customers of the stock exchange .

CLIENT TABLE



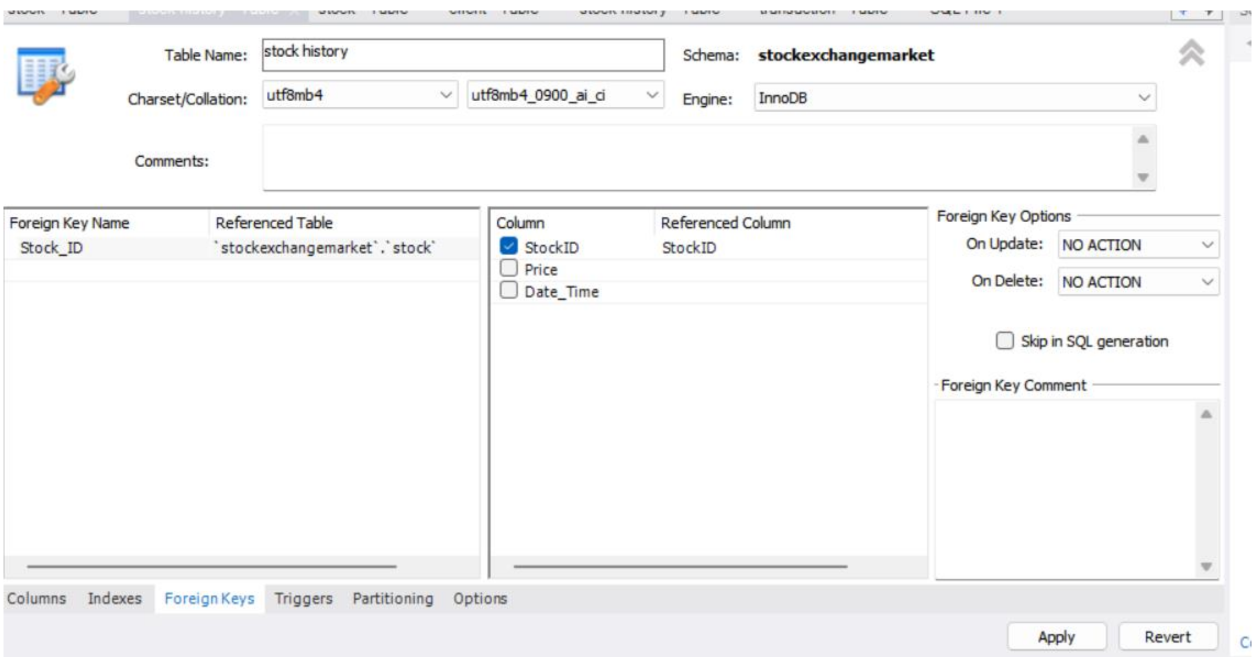
- The stock history table shown below stores the historical prices of each stock.

STOCK HISTORY TABLE



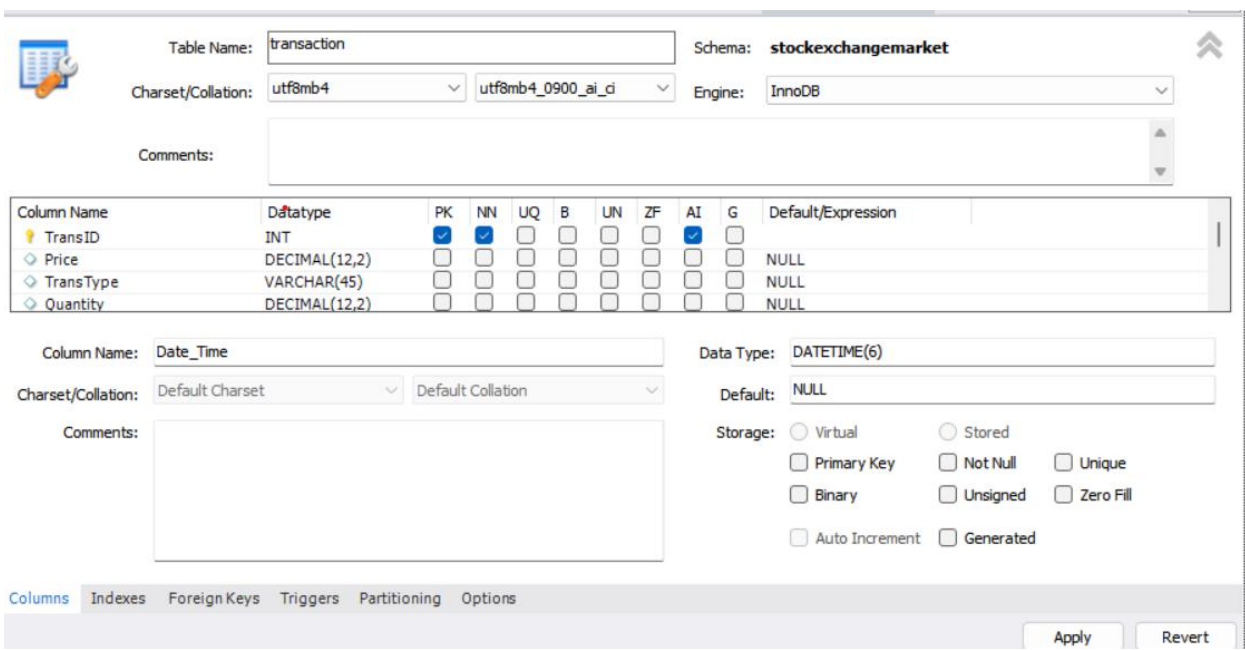
STOCK HISTORY FOREIGN KEYS

The stock history table has a foreign which is stock_ID entity referencing the Stocks entity's stockID column. Below is a picture showing the foreign key created in the DBMS.



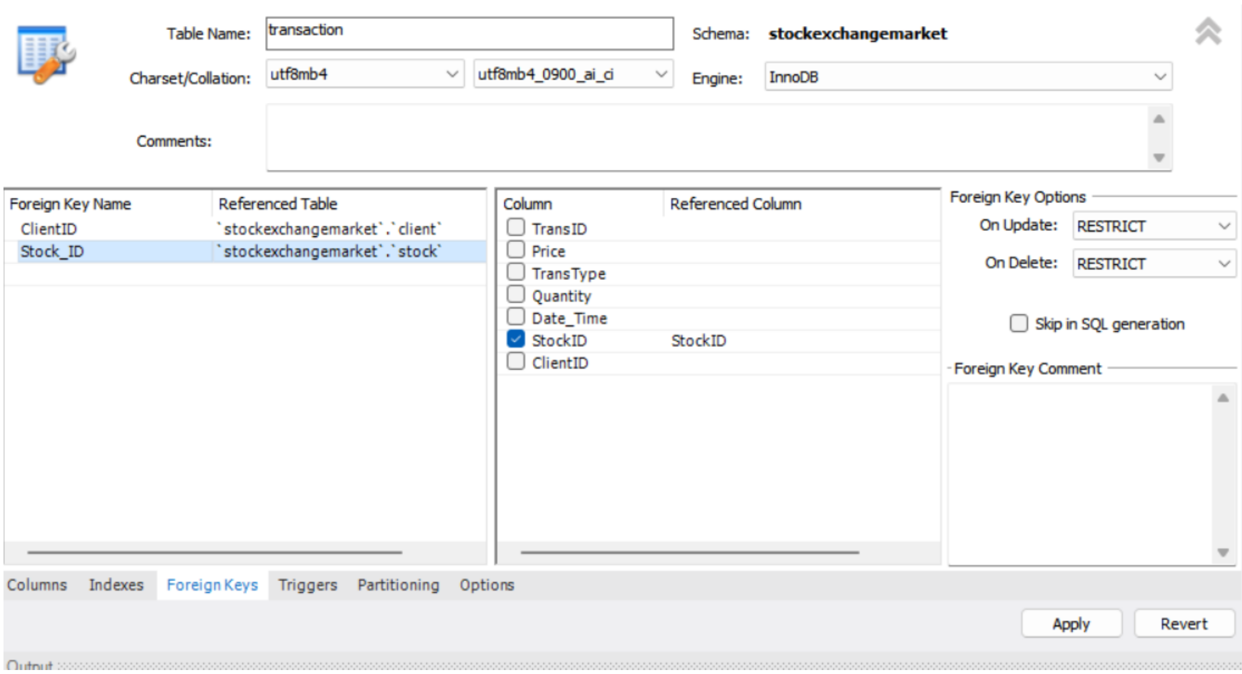
- The transaction table shown below stores the records of transactions made by clients for buying stocks.Below diagram shows the transaction table created in the DBMS.

TRANSACTION TABLE



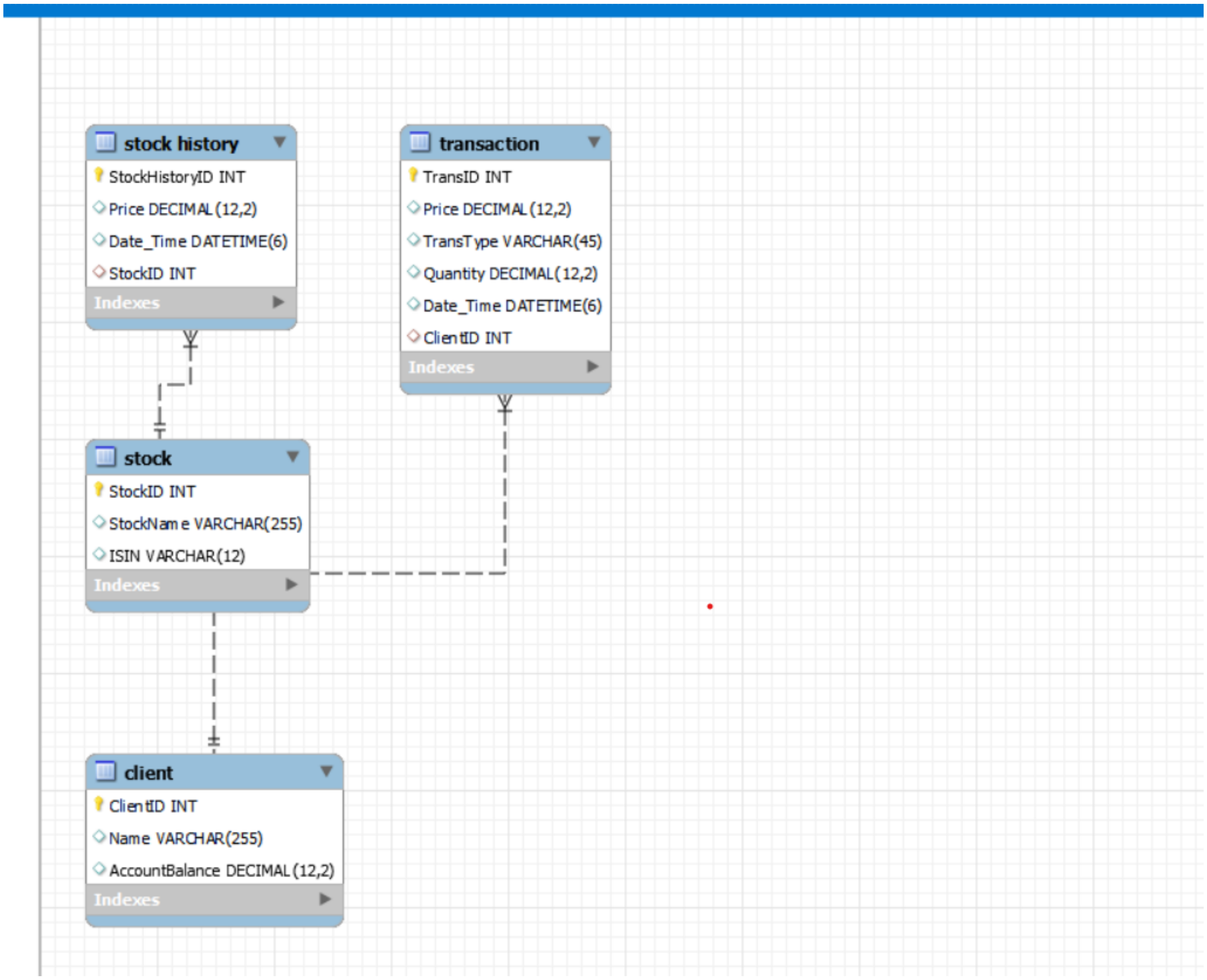
TRANSACTION TABLE FOREIGN KEYS

The transaction table has two foreign entities: stock_ID entity, which refers to the Stocks entity's stockID field, and ClientID entity, referencing the Client's entity ClientID.The image below depicts the foreign key created in the DBMS.



Following the creation of the tables containing the information for each entity, the DBMS built its own ER diagram illustrating the relationship between the entities based on the data established.The ER diagram is depicted below.

FIG 3A



THE ER DIAGRAM GENERATED MY THE DBMS(MYSQL)

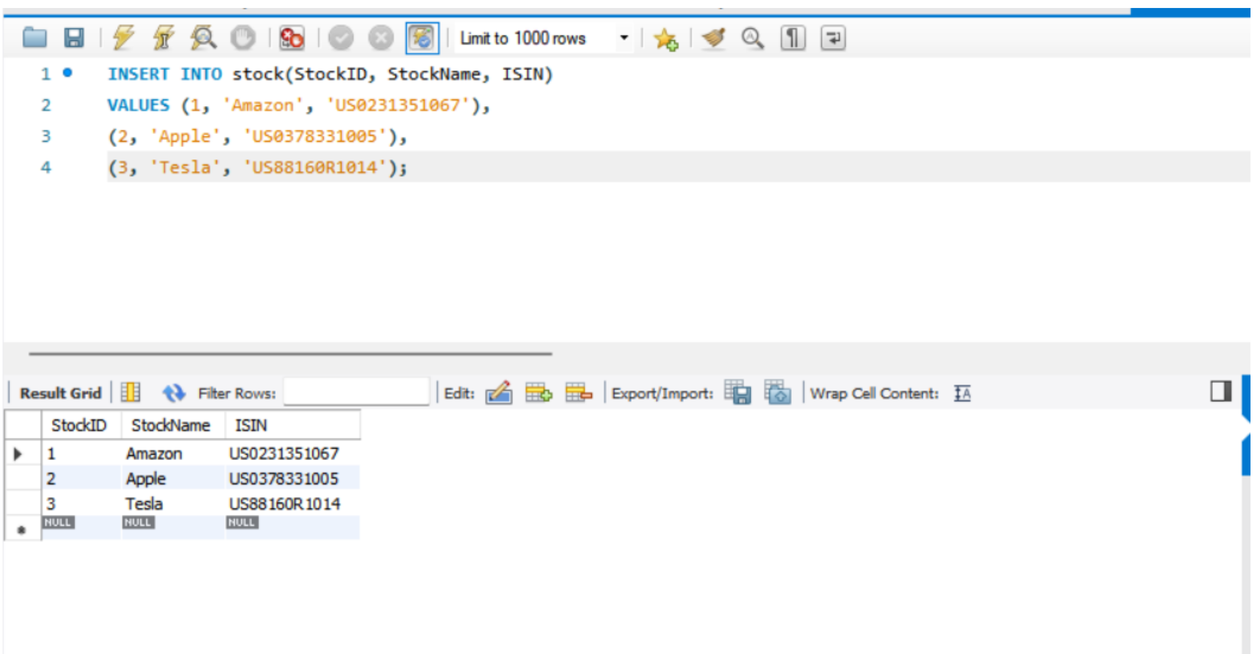
CHAPTER 4

In this chapter, values are filed into tables using SQL commands to enable retrieving and manipulating data from the database easier.

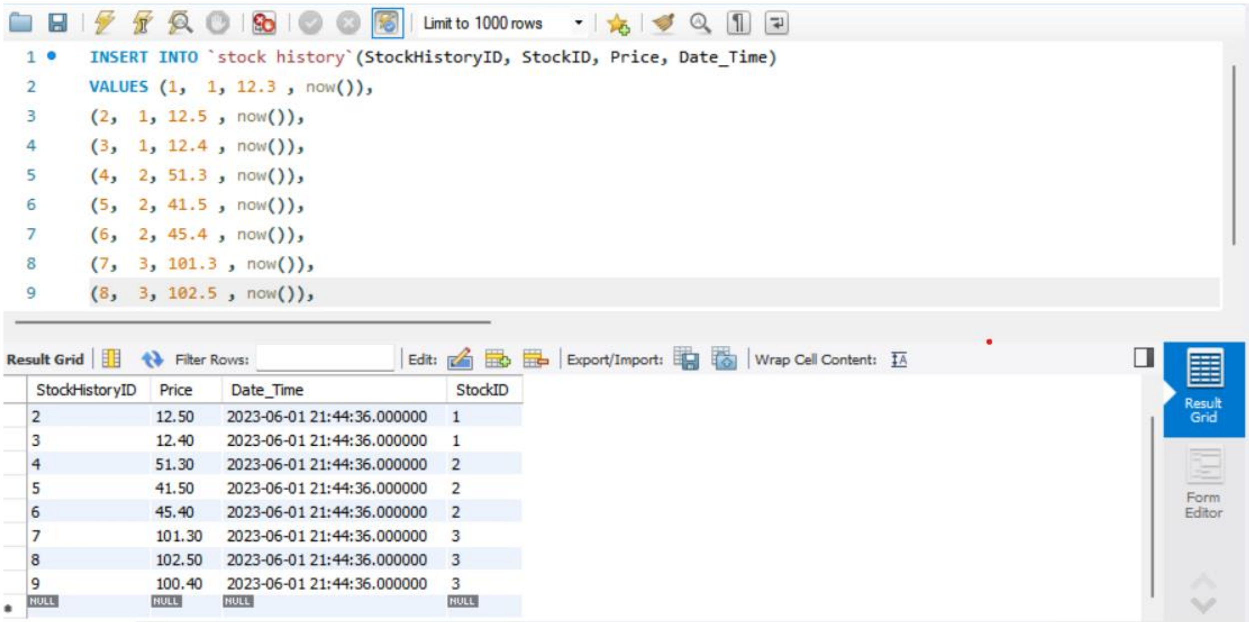
4.1 FIILING VALUES INTO TABLES USING SQL COMMAND

Table screenshots populated with values using SQL commands are shown below, using the table name as the header.:

STOCK TABLE WITH FILLED WITH VALUES



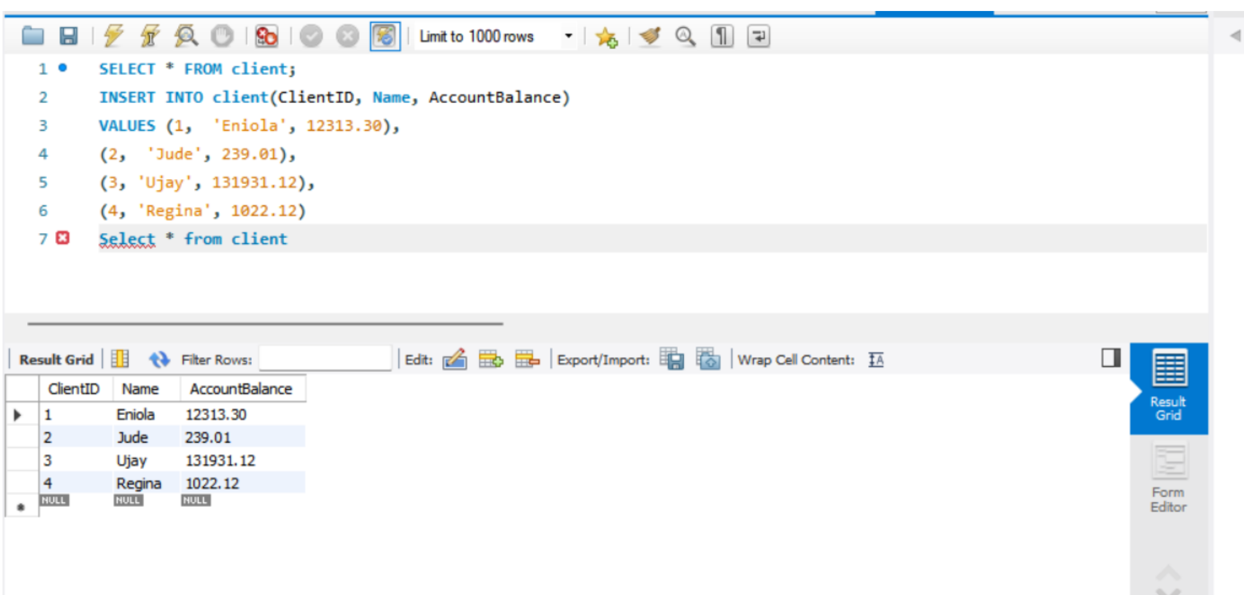
STOCK HISTORY TABLE WITH FILLED WITH VALUES



```
INSERT INTO `stock history`(StockHistoryID, StockID, Price, Date_Time)
VALUES (1,  1, 12.3 , now()),
(2,  1, 12.5 , now()),
(3,  1, 12.4 , now()),
(4,  2, 51.3 , now()),
(5,  2, 41.5 , now()),
(6,  2, 45.4 , now()),
(7,  3, 101.3 , now()),
(8,  3, 102.5 , now()),
```

StockHistoryID	Price	Date_Time	StockID
2	12.50	2023-06-01 21:44:36.000000	1
3	12.40	2023-06-01 21:44:36.000000	1
4	51.30	2023-06-01 21:44:36.000000	2
5	41.50	2023-06-01 21:44:36.000000	2
6	45.40	2023-06-01 21:44:36.000000	2
7	101.30	2023-06-01 21:44:36.000000	3
8	102.50	2023-06-01 21:44:36.000000	3
9	100.40	2023-06-01 21:44:36.000000	3
NULL	NULL	NULL	NULL

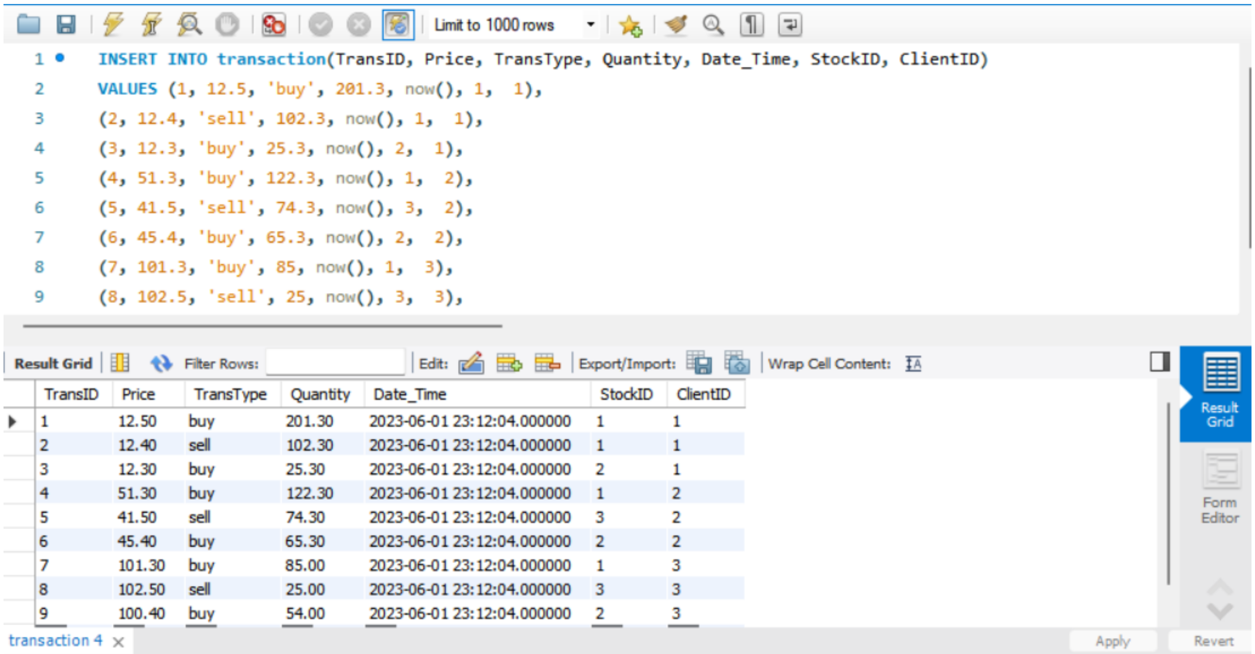
CLIENT TABLE WITH FILLED WITH VALUES



```
SELECT * FROM client;
INSERT INTO client(ClientID, Name, AccountBalance)
VALUES (1,  'Eniola', 12313.30),
(2,  'Jude', 239.01),
(3, 'Ujay', 131931.12),
(4, 'Regina', 1022.12)
Select * from client
```

ClientID	Name	AccountBalance
1	Eniola	12313.30
2	Jude	239.01
3	Ujay	131931.12
4	Regina	1022.12
NULL	NULL	NULL

TRANSACTION TABLE WITH FILLED WITH VALUES



```sql
INSERT INTO transaction(TransID, Price, TransType, Quantity, Date_Time, StockID, ClientID)
VALUES (1, 12.5, 'buy', 201.3, now(), 1,  1),
(2, 12.4, 'sell', 102.3, now(), 1,  1),
(3, 12.3, 'buy', 25.3, now(), 2,  1),
(4, 51.3, 'buy', 122.3, now(), 1,  2),
(5, 41.5, 'sell', 74.3, now(), 3,  2),
(6, 45.4, 'buy', 65.3, now(), 2,  2),
(7, 101.3, 'buy', 85, now(), 1,  3),
(8, 102.5, 'sell', 25, now(), 3,  3),
```

TransID	Price	TransType	Quantity	Date_Time	StockID	ClientID
1	12.50	buy	201.30	2023-06-01 23:12:04.000000	1	1
2	12.40	sell	102.30	2023-06-01 23:12:04.000000	1	1
3	12.30	buy	25.30	2023-06-01 23:12:04.000000	2	1
4	51.30	buy	122.30	2023-06-01 23:12:04.000000	1	2
5	41.50	sell	74.30	2023-06-01 23:12:04.000000	3	2
6	45.40	buy	65.30	2023-06-01 23:12:04.000000	2	2
7	101.30	buy	85.00	2023-06-01 23:12:04.000000	1	3
8	102.50	sell	25.00	2023-06-01 23:12:04.000000	3	3
9	100.40	buy	54.00	2023-06-01 23:12:04.000000	2	3

4.2 RETRIEVING AND MANIPULATING OF DATA FROM THE DATABASE CREATED

Data is retrieved from the database system using structured query language (SQL) once values are inserted into the table. Each query type written is executed, and a screenshot of the SQL instruction and result is shown below:

1. The first Query is written to determine the top 3 stocks with the highest trading volume.The sql command displayed in the screenshot below

```
-- determine the top 3 stocks with the highest trading volume
-- Select the isin and stock name, then sum their respective revenue
SELECT s.ISIN, s.StockName, SUM(t.Quantity * t.Price) AS revenue FROM stock AS s
JOIN transaction AS t
ON s.StockID = t.StockID --
GROUP BY 1, 2
ORDER BY revenue DESC -- sort in descending order
LIMIT 3 -- limit by 3 because we want the top 3


-- 2 NESTED
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ISIN	StockName	revenue
US0231351067	Amazon	28573.2200
US0378331005	Apple	8697.4100
US88160R1014	Tesla	5645.9500

From the above screenshots, the query retrieves data from ‘stock’ and ‘transaction ‘ tables and performs aggregation and sorting using Join and group query.

The result displayed shows that , the query retrieved the ISIN , stockName and revenue information of top 3 stocks based on their total revenue .

2. The second query seeks the client with the largest Account balance.The SQL command and result are shown below:

```
-- 2 NESTED
-- Client with highest account balance
SELECT * FROM client AS c
WHERE c.AccountBalance = (
        SELECT Max(AccountBalance) FROM client
    )
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ClientID	Name	AccountBalance
3	Ujay	131931.12
NULL	NULL	NULL

The query written above retrieves all rows form the ‘client’ table where Accountbalance value is equal to the maximum AccountBalance value found in the entire ‘Client’ table.

3. The third query is written to Retrieve the highest and the lowest prices for Amazon stock.The SQL command and result is shown in below screenshot:

```
JOIN `stock history` AS sh
ON s.StockID = sh.StockID
WHERE s.StockName = 'Amazon'
ORDER BY sh.Price
LIMIT 1)
    UNION ALL
(SELECT 'Highest' AS Title, Price FROM `stock` AS s
JOIN `stock history` AS sh
ON s.StockID = sh.StockID
```

```sql
WHERE s.StockName = 'Amazon'
ORDER BY sh.Price DESC
LIMIT 1)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title	Price
Lowest	12.30
Highest	12.50

The query written in uses the ‘UNION ALL’ operator to combine result of the two queries into a single result set.

4. The fourth query used a trigger to handle market orders.The query command and result is displayed in the below screenshot:

```sql
-- 4 TRIGGER
DELIMITER //
DROP TRIGGER IF EXISTS fetch_last_price //

CREATE TRIGGER fetch_last_price BEFORE INSERT ON transaction
FOR EACH ROW
BEGIN
    IF (NEW.Price IS null) THEN
        SET NEW.Price =  (SELECT Price FROM `stock history` sh WHERE StockID = NEW.StockID ORDER BY Date_time DESC LIMIT 1);

        END IF;
Save the script to a file.

DELIMITER ;

INSERT INTO transaction(TransID, Price, TransType, Quantity, Date_Time, StockID, ClientID)
VALUES (11, null, 'buy', 201.3, now(), 1,  1);
SELECT * FROM transaction;
```

TransID	Price	TransType	Quantity	Date_Time	StockID	ClientID
1	12.50	buy	201.30	2023-06-01 23:12:04.000000	1	1
2	12.40	sell	102.30	2023-06-01 23:12:04.000000	1	1
3	12.30	buy	25.30	2023-06-01 23:12:04.000000	2	1
4	51.30	buy	122.30	2023-06-01 23:12:04.000000	1	2
5	23.40	buy	74.30	2023-06-01 23:12:04.000000	3	2
7	101.30	buy	85.00	2023-06-01 23:12:04.000000	1	3
8	102.50	sell	25.00	2023-06-01 23:12:04.000000	3	3
9	100.40	buy	54.00	2023-06-01 23:12:04.000000	2	3
10	12.30	buy	201.30	2023-06-03 00:18:08.000000	1	1
11	12.30	buy	201.30	2023-06-04 02:34:55.000000	1	1
12	12.30	buy	201.30	2023-06-03 00:11:12.000000	1	1
13	12.30	buy	201.30	2023-06-03 00:11:55.000000	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

transaction 27 transaction 47 transaction 70

The SQL command results reveal that the query created a trigger that automatically sets the 'Price' column of a new transaction to the most recent price from the 'stock history' table if the 'Price' value is null

5. The 5th query is used to create a Store procedure for the database. Below is the sql command and result;

```sql
-- 5 Stored Procedure - create stored procedure the total daily revenue
DROP PROCEDURE IF EXISTS SelectDailyRevenue;

DELIMITER //
CREATE PROCEDURE SelectDailyRevenue()
        BEGIN
        SELECT SUM(t.Quantity * t.Price)
        FROM transaction t
        WHERE Date_time >= NOW() - INTERVAL 24 HOUR;
        END //
DELIMITER ;
CALL SelectDailyRevenue()
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SUM(t.Quantity * t.Price)
9903.9600

The query above creates a stored procedure that calculates and returns the total revenue generated in the last 24hours based on the quantity and price of transactions in the 'transaction' table.

6. The state query is created to insert new records into the 'stock table'.

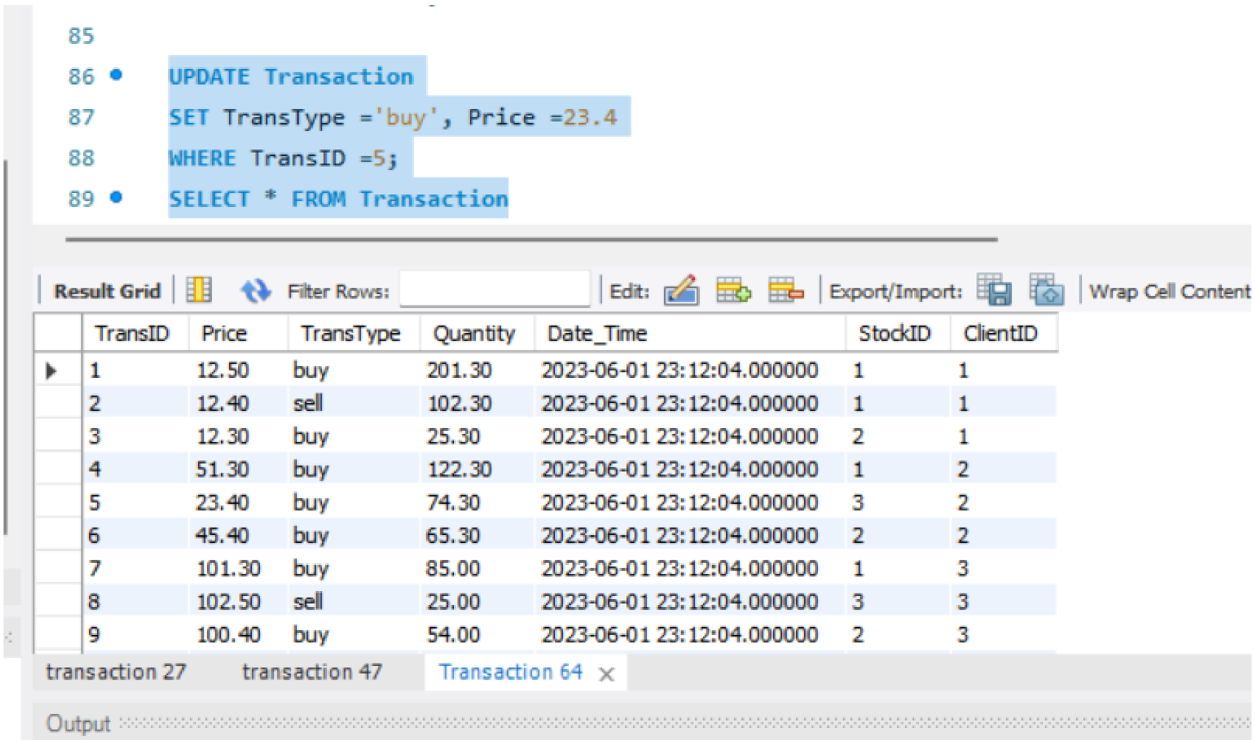
```sql
INSERT INTO stock(StockName,ISIN)
VALUES ('Facebook','US0393833101');
SELECT * FROM stock
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

StockID	StockName	ISIN
1	Amazon	US0231351067
2	Apple	US0378331005
3	Tesla	US88160R1014
4	Facebook	US0393833101
NULL	NULL	NULL

The result shows that,the query inserted new record with name ‘Facebook’ and ‘ISIN’ 'US0393833101' to the ‘stock table’ .

7. The 7th query is an update query which is executed to change the ‘TransType’ and ‘Price’ in the ‘Transaction Table’ where the ‘TransactionID’ =5. Screenshot for the result is shown below:



```
UPDATE Transaction
SET TransType ='buy', Price =23.4
WHERE TransID =5;
SELECT * FROM Transaction
```


TransID	Price	TransType	Quantity	Date_Time	StockID	ClientID
1	12.50	buy	201.30	2023-06-01 23:12:04.000000	1	1
2	12.40	sell	102.30	2023-06-01 23:12:04.000000	1	1
3	12.30	buy	25.30	2023-06-01 23:12:04.000000	2	1
4	51.30	buy	122.30	2023-06-01 23:12:04.000000	1	2
5	23.40	buy	74.30	2023-06-01 23:12:04.000000	3	2
6	45.40	buy	65.30	2023-06-01 23:12:04.000000	2	2
7	101.30	buy	85.00	2023-06-01 23:12:04.000000	1	3
8	102.50	sell	25.00	2023-06-01 23:12:04.000000	3	3
9	100.40	buy	54.00	2023-06-01 23:12:04.000000	2	3

From the result displayed above, the ‘TransType’ and ‘Price’ for ‘TransactionID 5’ is changed from a ‘sell’ and ‘41.5’ to ‘buy’ and ‘23.4’ respectively.

8. The Delete query is the eighth and last query written. The sql statement deletes data from the 'Transaction table' Where the 'TransID' is 6.The data in row '6' was erased as a result.

```sql
DELETE FROM Transaction
WHERE TransID =6;
SELECT * FROM TRANSACTION
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

TransID	Price	TransType	Quantity	Date_Time	StockID	ClientID
1	12.50	buy	201.30	2023-06-01 23:12:04.000000	1	1
2	12.40	sell	102.30	2023-06-01 23:12:04.000000	1	1
3	12.30	buy	25.30	2023-06-01 23:12:04.000000	2	1
4	51.30	buy	122.30	2023-06-01 23:12:04.000000	1	2
5	23.40	buy	74.30	2023-06-01 23:12:04.000000	3	2
7	101.30	buy	85.00	2023-06-01 23:12:04.000000	1	3
8	102.50	sell	25.00	2023-06-01 23:12:04.000000	3	3
9	100.40	buy	54.00	2023-06-01 23:12:04.000000	2	3
10	12.30	buy	201.30	2023-06-03 00:18:08.000000	1	1

transaction 27 | transaction 47 | TRANSACTION 68

CHAPTER 5

This chapter discusses the Efficiency of the DBMS created by explaining the consistency, speed or availability, scalability and Indexes of the DBMS.

5.1. CONSISTENCY OF THE DBMS

For a DBMS, consistency refers to the property of ensuring that the a database remains in a valid state before and after any transaction. It guarantees that the database's data meets all defined integrity constraints, rules, and relationship.In order to achieve the consistency of the DBMS created in this work, the Database also has to meet the ACID properties because this is a stock exchange market. The ACID properties are set of properties that ensure reliability and consistency in a database transaction.

From the DBMS created in this paper, it is evident that the DBMS maintains its consistency throughout every transaction and operation carried out by the SQL command. A valid example of this is when the "SELECT * FROM transaction" was written, the query retrieves all rows form the 'transaction' table and the DBMS ensures consistency by returning the data as it is stored without any modification.

The Atomicity property which stands for 'A' in ACID, guarantees that the transaction is treated as a single, indivisible unit of work. If any transaction fails, the entire transaction is rolled back, and changes made within the transaction are undone. The INSERT, UPDATE and DELETE statements in chapter 4 are executed individually, ensuring that each statement is atomic.

The 'I' which stands for isolation in ACID, guarantees that concurrent transactions are executed in a way that they appear to be executed sequentially, without interference from other transactions.

Durability of the ACID property ensures that once a transaction is committed, its effects are permanent and will survive any subsequent system failures, such as power outages or crashes.The changes made by the SQL commands ,will be durable and persist in database even in the presence of failures.

5.2 SCALABILITY OF THE DBMS

Scalability refers to the ability of a DBMS to handle increasing amount of data,user requests and system workload without sacrificing performance .The INSERT INTO transaction query in chapter 4 shows that the DBMS ability to handle increasing data volumes and the query to determine the top 3 stocks with the highest trading volume demonstrates the DBMS's ability to process complex queries involving JOIN and GROUP BY operations.

5.3 AVAILABILITY OF THE DBMS

The availability of a DBMS refers to the ability to remain operational and accessible to users, even in the presence of failures or disruptions. The creation of a TRIGGER and subsequent insert statement with a NULL value for the price column demonstrate the DBMS's availability to enforce business rules and perform automatic actions. The TRIGGER ensures that if a transaction's price is not provided, it will be fetched from a stock history. If the TRIGGER is correctly defined and the required data is available, the DBMS can execute the trigger and maintain the integrity of the data.

5.4 CAP THEOREM

The CAP theorem also known as the Brewer's theorem, is a fundamental principle in a distributed systems that states that it is impossible for a distributed data system to simultaneously provide all of the three guarantees: Consistency, Availability and partition Tolerance similar to scalability. It's important to note that CAP theorem states that it is impossible for a distributed system to simultaneously achieve consistency, availability, and partition tolerance. While scalability is important in a stock market to support high trading volumes and accommodate market growth, it may not be as critical as consistency and availability. Ensuring consistency and availability takes precedence over sheer scalability. Consistency ensures the integrity of trades information, while availability ensures uninterrupted access to market.

Scalability is not the primary concern in a stock market because the market focuses on consistency and availability due to the need for accurate, reliable, and uninterrupted trading activities. Also, Stock market being a small market, its scalability requirement might not be as extreme as other industries with massive data volumes or high transaction rates.

CONCLUSION

In conclusion, this paper focuses on the creation of a database management system for a stock exchange market. The database system utilized a relational database management system (RDMS) to store and manage the entities involved, including stocks, clients, transactions, and stock history. The ER diagram provided a visual representation of the relationships between these entities, while the schema and table were created based on the ER Diagram. Using SQL commands, data population, retrieval, and manipulation were efficiently performed, ensuring the integrity and reliability of the data. The system demonstrated consistency by adhering to ACID properties, while also showcasing efficiency through speed, availability, scalability, and the use of indexes.

Overall, the implemented DBMS provides an effective solution for managing stock exchange market data, offering a well-structured and reliable platform for handling transactions, stock history, and client information.

REFERENCES

Assad,C.,et al (2020) '*NOSQL Databases:Yearning For Disambiguation'* Available at : https://arxiv.org/pdf/2003.04074.pdf (Accessed:3rd June 2023)

Debomita, M.,et al (July 2018) ' *A data warehouse Based Modelling Technique for stock Market Analysis'.* International Journal of Engineering & Technology.

Hala,K.(2015) '*Extracting Entity Relationship Diagram (ERD) from Relational Database Schema'*. International Journal of Database Theory and Application 8(1)15-26.

Jacqueline,B.(2019) '*What is Entity Relationship Management ERD)?'* Available at : https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD#:~:text=An%20entity%20relationship%20diagram%20(ERD,information%20technology%20(IT)%20system (Accessed : 25th May 2023).

Jingyi,S.and M.omar,S (2020)'*Short term stock market price trend prediction using comprehensive deep learning system'*.Journal of Big Data 7(66)

Omkar,P.,et al (2016) '*Inadequacies of the CAP Theorem'* International Journal of the Computer Application 151(10).