Stock Market Management System: A Comprehensive Solution for Analyzing Data

BY

KKS KAPARDHEESWAR-21CSB0A28

ANURAG NAYAK-21CSB0A06

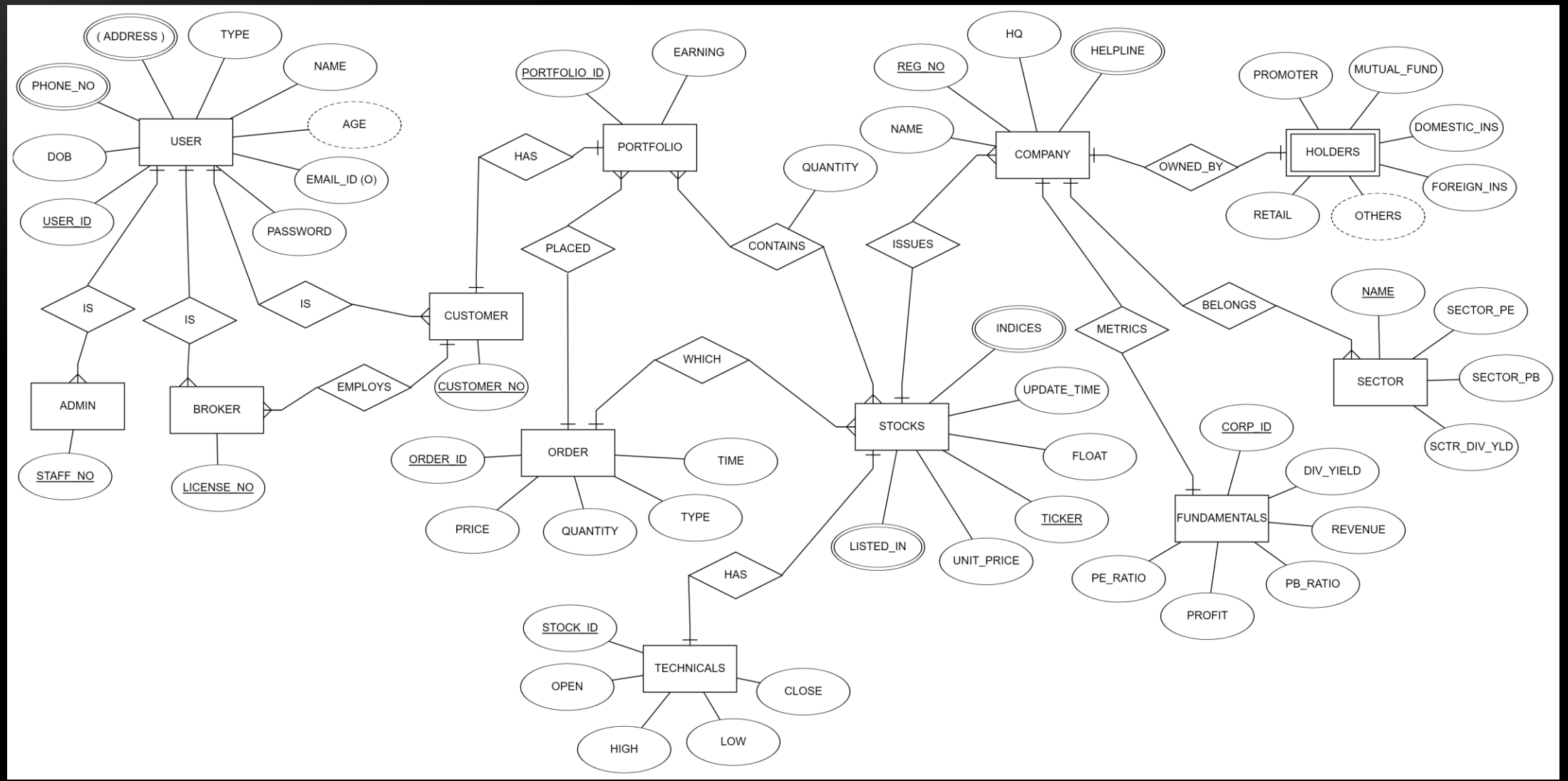
Introduction
ER Diagram
Tables and Relationships
Normalization
Queries
Challenges and Solutions
Conclusion

Introduction

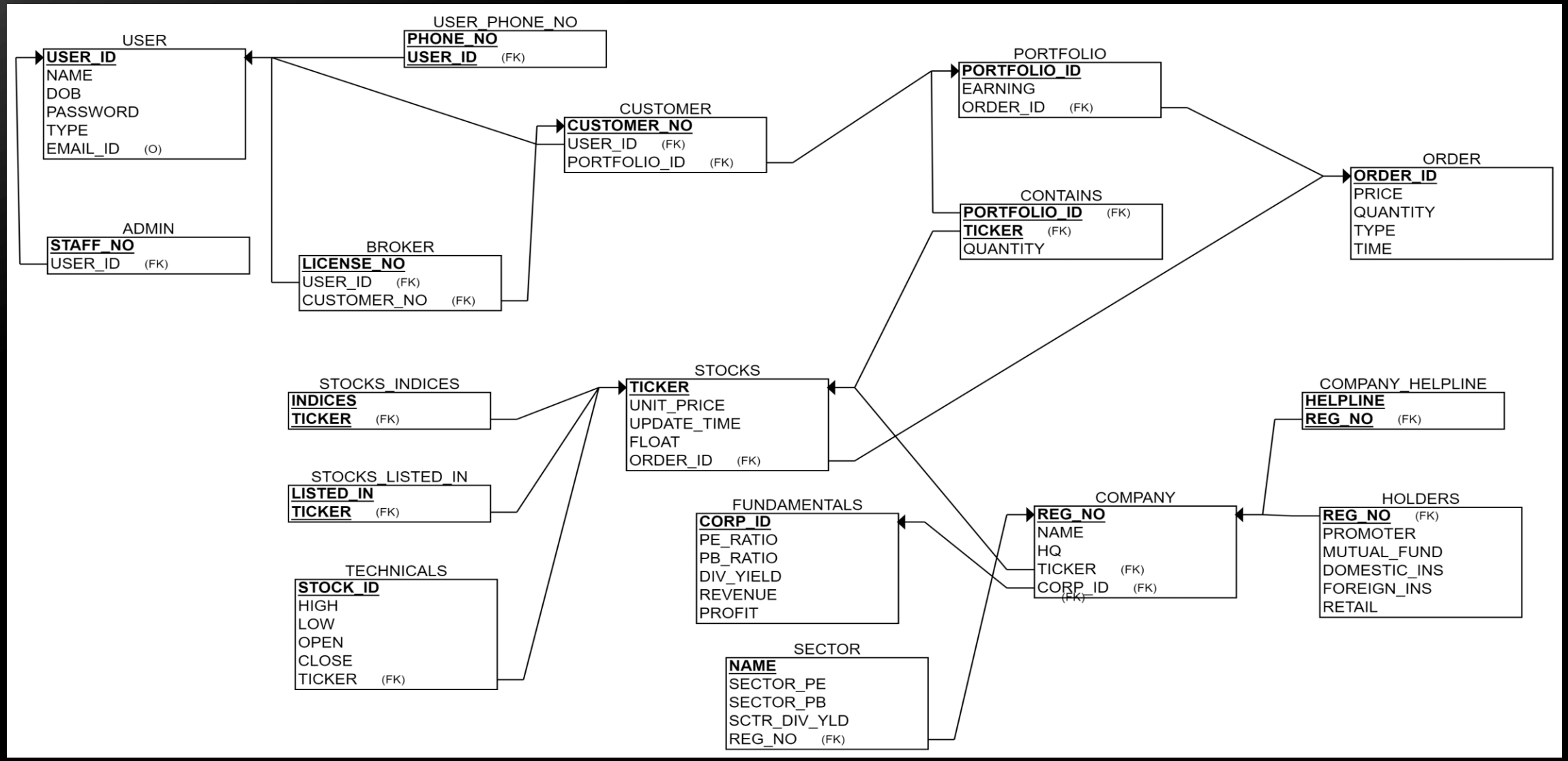
The Stock Market Management System is a powerful database that manages all the information related to stock market management. It includes tables for users, administrators, brokers, customers, portfolios, orders, stocks, technical, fundamentals, company information, holders, sectors, and company helplines.

This database provides a comprehensive solution for managing and analyzing stock market data. With its user-friendly interface and advanced features, it is an indispensable tool for anyone involved in stock market trading.

ER Diagram


(ADDRESS)
TYPE
NAME
PHONE_NO
AGE
USER
DOB
EMAIL_ID (O)
USER_ID
PASSWORD
IS
IS
IS
ADMIN
BROKER
CUSTOMER
EMPLOYS
CUSTOMER_NO
STAFF_NO
LICENSE_NO
PORTFOLIO_ID
EARNING
HAS
PORTFOLIO
PLACED
CONTAINS
QUANTITY
WHICH
ORDER_ID
ORDER
TIME
PRICE
QUANTITY
TYPE
HAS
STOCK_ID
TECHNICALS
OPEN
CLOSE
HIGH
LOW
HQ
REG_NO
HELPLINE
NAME
COMPANY
ISSUES
OWNED_BY
PROMOTER
MUTUAL_FUND
HOLDERS
DOMESTIC_INS
FOREIGN_INS
RETAIL
OTHERS
INDICES
METRICS
BELONGS
NAME
SECTOR_PE
SECTOR
SECTOR_PB
SCTR_DIV_YLD
STOCKS
UPDATE_TIME
FLOAT
TICKER
LISTED_IN
UNIT_PRICE
CORP_ID
DIV_YIELD
FUNDAMENTALS
REVENUE
PE_RATIO
PB_RATIO
PROFIT

Relational Model



A Visual Journey Through Tables and Relationships

The USERS Table

The USERS table stores information about the users of the system, including their name, date of birth, user ID, password, type, and email ID. The user ID column is the primary key of this table, which means that it uniquely identifies each row of data in the table.

Other tables in the schema reference the user ID column in the USERS table to establish relationships between the data stored in those tables and the users who created or interacted with that data.

```
CREATE TABLE USERS(
    NAME VARCHAR2(50),
    DOB DATE,
    USER_ID NUMBER(20) PRIMARY KEY,
    PASSWORD VARCHAR2(50),
    TYPE VARCHAR2(50),
    EMAIL_ID VARCHAR2(50)
);
```

The ADMINS Table

The ADMINS table stores information about the administrators of the system. It has two columns: staff number and user ID. The staff number column is the primary key of this table, and the user ID column references the user ID column in the USERS table.

This table establishes a relationship between the administrators and the users who have administrative privileges, allowing the system to control access to sensitive information and functionality within the system.

```
CREATE TABLE ADMINS(
    STAFF_NO NUMBER(20) PRIMARY KEY,
    USER_ID NUMBER(20)  REFERENCES
USERS(USER_ID)
);
```

The BROKER and CUSTOMER Tables

The BROKER table stores information about the brokers in the system, including their license number, user ID, and customer number. The license number column is the primary key of this table, and the user ID and customer number columns reference the user ID and customer number columns in the USERS and CUSTOMER tables, respectively.

The CUSTOMER table stores information about the customers in the system, including their customer number, user ID, and portfolio ID. The customer number column is the primary key of this table, and the user ID and portfolio ID columns reference the user ID and portfolio ID columns in the USERS and PORTFOLIO tables, respectively.

```
CREATE TABLE BROKER(
    LISCENSE_NO NUMBER(20) PRIMARY KEY,
    USER_ID NUMBER(20) REFERENCES
USERS(USER_ID),
    CUSTOMER_NO NUMBER(20)
REFERENCES CUSTOMER(CUSTOMER_NO)
);

CREATE TABLE CUSTOMER(
    CUSTOMER_NO NUMBER(20) PRIMARY
KEY,
    USER_ID NUMBER(20) REFERENCES
USERS(USER_ID),
    PORTFOLIO_ID NUMBER(20) REFERENCES
PORTFOLIO(PORTFOLIO_ID)
);
```

The PORTFOLIO and ORDERS Tables

The PORTFOLIO table stores information about the portfolios in the system, including their portfolio ID and earnings. The portfolio ID column is the primary key of this table, and the order ID column references the order ID column in the ORDERS table.

The ORDERS table stores information about the orders placed by the customers in the system, including their order ID, price, quantity, and order type. The order ID column is the primary key of this table, and it is referenced by the order ID column in the PORTFOLIO table.

```
CREATE TABLE PORTFOLIO(
    PORTFOLIO_ID NUMBER(20) PRIMARY KEY,
    EARNING NUMBER(20),
    ORDER_ID NUMBER(20) REFERENCES
ORDERS(ORDER_ID)
);
```

```
CREATE TABLE ORDERS(
        ORDER_ID NUMBER(20) PRIMARY KEY,
        PRICE NUMBER(20),
        QUANTITY NUMBER(20),
        ORDER_TYPE VARCHAR2(50),
        ORDER_TIME TIMESTAMP
);
```

Understanding the FUNDAMENTALS Table

The FUNDAMENTALS table contains financial data for each corporation, including PE ratio, PB ratio, dividend yield, revenue, and profit.

The CORP_ID column serves as the primary key for this table, allowing it to be linked to other tables in the database.


```
CREATE TABLE FUNDAMENTALS (
    CORP_ID    NUMBER(20) PRIMARY
KEY,
    PE_RATIO   NUMBER(20, 5),
    PB_RATIO   NUMBER(20, 5),
    DIV_YIELD NUMBER(20, 5),
    REVENUE    NUMBER(20, 5),
    PROFIT     NUMBER(20, 5)
);
```

Exploring the COMPANY Table

The COMPANY table stores information about each corporation, such as its registration number, name, headquarters, and ticker symbol.

The foreign keys in this table link it to the FUNDAMENTALS and STOCKS tables, allowing financial data and stock prices to be associated with each company.

```
CREATE TABLE company (
   reg_no       NUMBER(20) PRIMARY KEY,
   company_name VARCHAR2(50),
   headquarters VARCHAR2(50),
   ticker       VARCHAR2(50),
   corp_id      NUMBER(20),
   FOREIGN KEY ( ticker )
      REFERENCES stocks ( ticker ),
   FOREIGN KEY ( corp_id )
      REFERENCES fundamentals ( corp_id )
);
```

Examining the SECTOR Table

The SECTOR table stores information about different sectors of the economy, including their names, PE ratios, PB ratios, and dividend yields.

The REG_NO column serves as the primary key for this table and links it to the COMPANY table, allowing sector data to be associated with each company.

```
CREATE TABLE sector (
   sector_name    VARCHAR2(50),
   sector_pe      NUMBER(20, 5),
   sector_pb      NUMBER(20, 5),
   sector_div_yld NUMBER(20, 5),
   reg_no         NUMBER(20),
   primary key (sector_name,reg_no),
   FOREIGN KEY ( reg_no )
      REFERENCES company ( reg_no )
);
```

The USER_PHONE_NO Table

The USER_PHONE_NO table stores the phone numbers of the users in the system. It has two columns: phone number and user ID. The phone number column is the primary key of this table, and the user ID column references the user ID column in the USERS table.

This table allows multiple phone numbers to be associated with a single user, and it also enables the system to retrieve all phone numbers associated with a particular user by querying the USER_PHONE_NO table using the user ID column.

```
CREATE TABLE USER_PHONE_NO(
   PHONE_NO NUMBER(20),
   USER_ID NUMBER(20)
REFERENCES USERS(USER_ID),
   PRIMARY KEY
(PHONE_NO,USER_ID)
);
```

Analyzing the HOLDERS Table

The HOLDERS table contains data on the ownership of each corporation's stock, including the percentage owned by promoters, mutual funds, domestic and foreign institutional investors, and retail investors.

Like the COMPANY table, this table has a foreign key linking it to the COMPANY table, allowing ownership data to be associated with each company.

```
CREATE TABLE holders (
   reg_no       NUMBER(20) PRIMARY KEY,
   promoter     NUMBER(20, 5),
   mutual_fund  NUMBER(20, 5),
   domestic_ins NUMBER(20, 5),
   foreign_ins  NUMBER(20, 5),
   retail       NUMBER(20, 5),
   FOREIGN KEY ( reg_no )
      REFERENCES company ( reg_no )
);
```

TECHNICALS table

```
CREATE TABLE technicals (
   stock_id    NUMBER(20) PRIMARY KEY,
   price_high  NUMBER(20, 5),
   price_low   NUMBER(20, 5),
   price_open  NUMBER(20, 5),
   price_close NUMBER(20, 5),
   ticker      VARCHAR2(50),
   FOREIGN KEY ( ticker )
      REFERENCES stocks ( ticker )
);
```

STOCKS table

```
CREATE TABLE stocks (
   ticker      VARCHAR2(50) PRIMARY KEY,
   unit_price   NUMBER(20, 5),
   update_time  TIMESTAMP,
   stocks_float NUMBER(20),
   order_id     NUMBER(20),
   FOREIGN KEY ( order_id )
      REFERENCES orders ( order_id )
);
```

STOCK details

```
CREATE TABLE stocks_listed_in (
   listed_in VARCHAR2(50),
   ticker    VARCHAR2(50),
   PRIMARY KEY ( listed_in,
                 ticker ),
   FOREIGN KEY ( ticker )
      REFERENCES stocks ( ticker )
);
```

CONTAINS TABLE

```
CREATE TABLE contains (
   portfolio_id NUMBER(20)
      REFERENCES portfolio ( portfolio_id ),
   ticker      VARCHAR2(50)
      REFERENCES stocks ( ticker ),
   quantity    NUMBER(20),
   PRIMARY KEY ( portfolio_id,
                 ticker )
);
```

```
CREATE TABLE stocks_indices (
   stock_index VARCHAR2(50),
   ticker      VARCHAR2(50),
   PRIMARY KEY ( stock_index,
                 ticker ),
   FOREIGN KEY ( ticker )
      REFERENCES stocks ( ticker )
);
```

Functional Dependencies for Each Table

Users

{USER_ID} -> {NAME, DOB, PASSWORD, TYPE, EMAIL_ID}

User Phone No

{PHONE_NO, USER_ID} -> {}

Admins

{STAFF_NO} -> {USER_ID}

Broker

{LISCENSE_NO} -> {USER_ID, CUSTOMER_NO}

Stocks

{TICKER} -> {UNIT_PRICE, UPDATE_TIME, STOCKS_FLOAT, ORDER_ID}

Portfolio

{PORTFOLIO_ID} -> {EARNING, ORDER_ID}

Orders

{ORDER_ID} -> {PRICE, QUANTITY, ORDER_TYPE, ORDER_TIME}

Customer

{CUSTOMER_NO} -> {USER_ID, PORTFOLIO_ID}

Stocks Listed In

{LISTED_IN, TICKER} -> {}

Stocks Indices

{STOCK_INDEX, TICKER} -> {}

Technicals

{STOCK_ID} -> {PRICE_HIGH, PRICE_LOW, PRICE_OPEN, PRICE_CLOSE, TICKER}

Contains

{PORTFOLIO_ID, TICKER} -> {QUANTITY}

Fundamentals

{CORP_ID} -> {PE_RATIO, PB_RATIO, DIV_YIELD, REVENUE, PROFIT}

Company

{reg_no} -> {company_name, headquarters, ticker, corp_id}

Holders

{reg_no} -> {promoter, mutual_fund, domestic_ins, foreign_ins, retail}

Sector

{sector_name, reg_no} -> {sector_pe, sector_pb, sector_div_yld}

Company Helpline

{helpline, reg_no} -> {}

Tables meet the BCNF standard if they possess a solitary candidate key and absence of any significant functional dependencies.

Since all tables meets the BCNF standard we can say that our DB model is Normalized.

Content in TABLES

USERS table

	NAME	DOB	USER_ID	PASSWORD	TYPE	EMAIL_ID
1	John	01-01-90	1	1234	Customer	john@example.com
2	Steve	05-04-70	3	efgh	Admin	steve@example.com
3	Mary	03-02-80	2	abcd	Broker	mary@example.com
4	Richard	09-07-60	4	ijkl	Customer	richard@example.com
5	Mark	11-08-50	5	mnop	Customer	mark@example.com
6	Jane Smith	05-05-95	13	pksf923	Broker	jane.smith@example.com
7	Robert Johnson	31-12-85	24	pvrm163	Broker	robert.johnson@example.com
8	Emily Davis	15-03-92	35	pasl1e33	Broker	emily.davis@example.com
9	William Brown	20-07-98	46	pasf23	Broker	william.brown@example.com
10	Anurag Nayak	10-07-97	69	paef23	Customer	anurag@example.com
11	Kapardhi	24-08-96	73	pasft3	Customer	kappa@example.com

USER's Phone No table

	PHONE_NO	USER_ID
1	1234567890	1
2	1234567890	35
3	2345678901	2
4	2345678901	35
5	3456789012	3
6	3456789012	46
7	4567890123	4
8	4567890123	46
9	5678901234	5
10	5678901234	46
11	6789012345	13
12	7890123456	13
13	8901234567	24
14	9012345678	24

BROKERS table

	LISCENSE_NO	USER_ID	CUSTOMER_NO
1	10001	2	20001
2	10002	13	20002
3	10003	24	20003
4	10004	35	20004
5	10005	46	20005

CUSTOMERS table

	CUSTOMER_NO	USER_ID	PORTFOLIO_ID
1	20001	1	30001
2	20002	4	30002
3	20003	5	30003
4	20004	69	30004
5	20005	73	30005

PORTFOLIO's table

	PORTFOLIO_ID	EARNING	ORDER_ID
1	30001	100000	40001
2	30002	200000	40002
3	30003	300000	40003
4	30004	200000	40004
5	30005	500000	40005

ORDERS table

	ORDER_ID	PRICE	QUANTITY	ORDER_TYPE	ORDER_TIME
1	40001	1000	10	BUY	01-01-20 9:30:00.000000000 AM
2	40002	1500	20	SELL	02-02-20 10:30:00.000000000 AM
3	40003	2000	30	BUY	03-03-20 11:30:00.000000000 AM
4	40004	2500	40	SELL	04-04-20 12:30:00.000000000 PM
5	40005	3000	50	BUY	05-05-20 1:30:00.000000000 PM

STOCKS table

	TICKER	UNIT_PRICE	UPDATE_TIME	STOCKS_FLOAT	ORDER_ID
1	AAPL	125.5	01-01-20 9:30:00.000000000 AM	1000	40001
2	MSFT	110.2	02-02-20 10:30:00.000000000 AM	1500	40002
3	AMZN	1790.23	03-03-20 11:30:00.000000000 AM	2000	40003
4	FB	195.32	04-04-20 12:30:00.000000000 PM	2500	40004
5	GOOGL	1280.39	05-05-20 1:30:00.000000000 PM	3000	40005

STOCKS listed IN :

	LISTED_IN	TICKER
1	NASDAQ	AAPL
2	NASDAQ	AMZN
3	NASDAQ	GOOGL
4	NYSE	FB
5	NYSE	MSFT

TECHNICAL table

	STOCK_ID	PRICE_HIGH	PRICE_LOW	PRICE_OPEN	PRICE_CLOSE	TICKER
1	1	130.65	120.23	125.53	126.12	AAPL
2	2	115.34	105.62	110.43	109.34	MSFT
3	3	1800.31	1730.23	1770.23	1771.24	AMZN
4	4	200.43	190.65	199.12	199.54	FB
5	5	1290.65	1250.65	1280.72	1282.39	GOOGL

CONTAINS table

	PORTFOLIO_ID	TICKER	QUANTITY
1	30001	AAPL	100
2	30002	MSFT	100
3	30003	AMZN	50
4	30004	FB	200
5	30005	GOOGL	150

FUNDAMENTALS table

	CORP_ID	PE_RATIO	PB_RATIO	DIV_YIELD	REVENUE	PROFIT
1	1	3.45	8.23	2.43	10000	2000
2	2	2.43	6.32	1.5	20000	5000
3	3	5.54	1.23	0.23	30000	10000
4	4	4.23	1.63	1.45	40000	15000
5	5	3.45	5.32	2.5	50000	20000

COMPANY table

	REG_NO	COMPANY_NAME	HEADQUARTERS	TICKER	CORP_ID
1	1	Apple Inc	Cupertino	AAPL	1
2	2	Microsoft Corp	Redmond	MSFT	2
3	3	Amazon Inc	Seattle	AMZN	3
4	4	Facebook Inc	Menlo Park	FB	4
5	5	Alphabet Inc	Mountain View	GOOGL	5

HOLDERS table

	REG_NO	PROMOTER	MUTUAL_FUND	DOMESTIC_INS	FOREIGN_INS	RETAIL
1	1	50	25	10	10	5
2	2	60	20	5	10	5
3	3	40	15	25	10	10
4	4	45	10	20	15	10
5	5	55	5	20	15	5

SECTORS table

	SECTOR_NAME	SECTOR_PE	SECTOR_PB	SECTOR_DIV_YLD	REG_NO
1	Technology	5.34	1.43	1.5	1
2	Technology	4.53	8.75	1.23	2
3	Delivery	6.63	1.23	0.43	3
4	Social media	4.32	1.64	1.25	4
5	Technology	5.67	6.3	2.3	5

COMPANY Helplines table

	HELPLINE	REG_NO
1	111331122	4
2	123321123	5
3	222111222	3
4	333111222	2
5	444111222	1

Insights into SQL Queries

Query 1: Retrieving Broker Names and Email Addresses who have a phone number starting with '234'.

The first query is aimed at retrieving the names and email addresses of all brokers who have a phone number starting with '234'. The SQL code for this query involves joining three tables - USERS, BROKER, and USER_PHONE_NO - to extract the required information. The WHERE clause filters out only those records where the phone number starts with '234' and the user role is 'Broker'. The resulting output provides us with the names and email addresses of all such brokers.

This query can be useful in situations where we need to contact brokers with a specific phone number prefix. By filtering out the results based on the phone number prefix, we can easily obtain the relevant contact information without having to manually search through a large database.

```
SELECT u.NAME, u.EMAIL_ID
FROM USERS u
JOIN BROKER b ON u.USER_ID = b.USER_ID
JOIN USER_PHONE_NO up ON up.USER_ID =
b.USER_ID
WHERE up.PHONE_NO LIKE '234%';
```

	NAME	EMAIL_ID
1	Mary	mary@example.com
2	Emily Davis	emily.davis@example.com

Query 2: Retrieving Portfolio Values and Customer Names

The second query is aimed at retrieving the total value of each portfolio along with the corresponding customer name. The SQL code for this query involves joining two tables - CUSTOMER and PORTFOLIO - to extract the required information. The resulting output provides us with a list of all portfolios along with their total value and the name of the customer associated with each portfolio.

This query can be useful in situations where we need to analyze the performance of different portfolios or compare the values of different customers' portfolios. By obtaining the total value of each portfolio and the corresponding customer name, we can easily perform such analyses without having to manually calculate the values or match them with the respective customers.

```
Select
   U.Name        As Customer_Name,
   P.Portfolio_Id,
   Sum(P.Earning) As Total_Value
From
        Customer C
   Join Users     U On C.User_Id = U.User_Id
   Join Portfolio P On C.Portfolio_Id =
P.Portfolio_Id
Group By
   U.Name,
   P.Portfolio_Id;
```

	CUSTOMER_NAME	PORTFOLIO_ID	TOTAL_VALUE
1	Anurag Nayak	30004	200000
2	John	30001	100000
3	Richard	30002	200000
4	Kapardhi	30005	500000
5	Mark	30003	300000

Query 3: Average Stock Prices of Top-Tech Companies

Let's take a closer look at the query:

SELECT C.company_name, AVG(S.unit_price) AS avg_stock_price
FROM company C
JOIN stocks S ON C.ticker = S.ticker
GROUP BY C.company_name;

The first line selects the company name and the average price of its stocks. The second line renames the average price column as AVERAGE_PRICE. The third line joins the COMPANY and STOCKS tables using the SYMBOL column. The fourth line groups the results by company name.

This query is efficient because it combines data from two tables and calculates the average price for each company in one step. It also eliminates duplicates by grouping the results by company name.

```
SELECT
   c.company_name,
   AVG(s.unit_price) AS avg_stock_price
FROM
        company c
   JOIN stocks s ON c.ticker = s.ticker
GROUP BY
   c.company_name;
```

	COMPANY_...	AVG_STOCK_PRICE
1	Facebook Inc	195.32
2	Apple Inc	125.5
3	Amazon Inc	1790.23
4	Alphabet Inc	1280.39
5	Microsoft Corp	110.2

Microsoft Corp

Microsoft Corp is another tech giant that has been around for decades. Its products include Windows, Office, and Xbox. According to our query, the average price of Microsoft's stocks is $110.20.

Amazon Inc

Amazon Inc is an e-commerce giant that has disrupted the retail industry. It has also expanded into other areas such as cloud computing and entertainment. According to our query, the average price of Amazon's stocks is $1790.23.

Apple Inc

Apple Inc is one of the most valuable companies in the world. Its products are known for their innovation and design. According to our query, the average price of Apple's stocks is $125.50.

Use Cases

The Stock Market Management System can be used by a wide range of individuals and organizations involved in stock market trading. For example, individual investors can use the system to manage their own portfolios and make informed investment decisions. Brokers can use the system to manage their clients' portfolios and execute trades on their behalf.

Institutional investors, such as hedge funds and mutual funds, can use the system to manage large portfolios and make strategic investment decisions. Finally, financial analysts and researchers can use the system to conduct in-depth analysis of stock market trends and identify potential investment opportunities.

Challenges and Solutions

One of the biggest challenges facing the Stock Market Management System is the sheer volume of data that it needs to manage. With millions of trades taking place every day, the system needs to be able to handle a massive amount of data without slowing down or crashing. To address this challenge, the system uses advanced algorithms and data compression techniques to optimize performance and ensure that all data is processed quickly and accurately.

Another challenge facing the system is the need to ensure data privacy and security. With so much sensitive financial information being stored in the database, it is critical that the system is able to protect against unauthorized access or theft. To address this challenge, the system uses advanced encryption and authentication protocols to ensure that all user data is kept secure at all times.

END