STOCK MARKET MANAGEMENT SYSTEM

Prepared by
21CSB0A06 (Anurag Nayak)
21CSB0A28 (KKS Kapardheeswar)

INDEX

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.

ENTITIES

1)USERS: Represents the users of the system and includes attributes such as name, date of birth, user ID, password, user type, and email.

2)USER_PHONE_NO: Stores the phone numbers associated with users. It has attributes like phone number and user ID.

3)ADMINS: Represents the administrators in the system and includes the admin ID and user ID.

4)BROKER: Represents the brokers in the system. It includes attributes like broker ID, user ID, and admin ID.

5)CUSTOMER: Represents the customers in the system. It includes attributes like customer ID, user ID, and broker ID.

6)PORTFOLIO: Stores information about the portfolios associated with customers. It includes attributes like portfolio ID, initial investment, and customer ID.

7)ORDERS: Represents the orders placed by customers. It includes attributes like order ID, quantity, price, order type, and timestamp.

8)STOCKS: Represents the stocks available in the system. It includes attributes like stock symbol, price, timestamp, quantity, and order ID.

9)STOCKS_LISTED_IN: Stores the stock symbols and the stock exchanges they are listed on.

10)STOCKS_INDICES: Represents the indices that include specific stocks. It includes attributes like index name and stock symbol.

11)TECHNICALS: Stores technical information about stocks such as opening price, closing price, high price, low price, and stock symbol.

12)CONTAINS: Associates portfolios with the stocks they contain. It includes attributes like portfolio ID and stock symbol.

13)FUNDAMENTALS: Stores fundamental information about stocks, including attributes like P/E ratio, EPS, dividend yield, market capitalization, and stock symbol.

14)COMPANY: Represents the companies associated with stocks. It includes attributes like company name, location, stock symbol, and sector ID.

15)HOLDERS: Stores information about the stockholders. It includes attributes like holder ID and percentages of different types of holdings.

16)SECTOR: Represents the sectors to which companies belong. It includes attributes like sector name and performance indicators.

17)COMPANY_HELPLINE: Stores the contact information for companies' helplines. It includes attributes like helpline number and company ID.

RELATIONSHIPS

1. One-to-One Relationships:
   - USERS and ADMINS: Each user can have at most one corresponding admin.
   - USERS and BROKER: Each user can have at most one corresponding broker.
   - USERS and CUSTOMER: Each user can have at most one corresponding customer.

2. One-to-Many Relationships:
   - USERS and USER_PHONE_NO: Each user can have multiple phone numbers associated with them.
   - ADMINS and USERS: Each admin can be associated with multiple users.

3. Many-to-One Relationships:
   - BROKER and USERS: Each broker can be associated with multiple users.
   - CUSTOMER and USERS: Each customer can be associated with multiple users.
   - PORTFOLIO and CUSTOMER: Each portfolio belongs to a single customer.
   - ORDERS and CUSTOMER: Each order is placed by a single customer.
   - STOCKS and ORDERS: Each stock can be associated with multiple orders.
   - STOCKS_LISTED_IN and STOCKS: Each stock can be listed on multiple stock exchanges.

- STOCKS_INDICES and STOCKS: Each stock can be part of multiple stock indices.
- TECHNICALS and STOCKS: Each technical data entry corresponds to a single stock.
- CONTAINS and PORTFOLIO: Each portfolio can contain multiple stocks.
- FUNDAMENTALS and STOCKS: Each fundamental data entry corresponds to a single stock.
- COMPANY and STOCKS: Each stock corresponds to a single company.
- HOLDERS and STOCKS: Each stock can have multiple holders.
- SECTOR and COMPANY: Each company belongs to a single sector.
- COMPANY_HELPLINE and COMPANY: Each company can have multiple helpline numbers.

4. Many-to-Many Relationships:
   - PORTFOLIO and STOCKS: Each portfolio has many stocks and each stock can be present in many portfolios.

RELATIONAL SCHEMA

USERS
(user_id NUMBER PRIMARY KEY,
 name VARCHAR(255),
date_of_birth DATE,
phone_number VARCHAR(20),
password VARCHAR(255),
role VARCHAR(255),
email VARCHAR(255))

USER_PHONE_NO (
phone_number VARCHAR(20),
user_id NUMBER REFERENCES USERS(user_id))

ADMINS (
admin_id NUMBER PRIMARY KEY,
 user_id NUMBER REFERENCES USERS(user_id))

BROKER (
broker_id NUMBER PRIMARY KEY,
user_id NUMBER REFERENCES USERS(user_id),
 broker_code NUMBER)

CUSTOMER (
customer_id NUMBER PRIMARY KEY,
user_id NUMBER REFERENCES USERS(user_id),
customer_code NUMBER)

PORTFOLIO (
portfolio_id NUMBER PRIMARY KEY,
total_value NUMBER(10,2),
customer_id NUMBER REFERENCES CUSTOMER(customer_id))

ORDERS (
order_id NUMBER PRIMARY KEY,
amount NUMBER(10,2),
quantity NUMBER,
type VARCHAR(10),
timestamp TIMESTAMP,
customer_id NUMBER REFERENCES CUSTOMER(customer_id))

STOCKS (
stock_symbol VARCHAR(10) PRIMARY KEY,
price NUMBER(10,2),
timestamp TIMESTAMP,
volume NUMBER,
order_id NUMBER REFERENCES ORDERS(order_id))

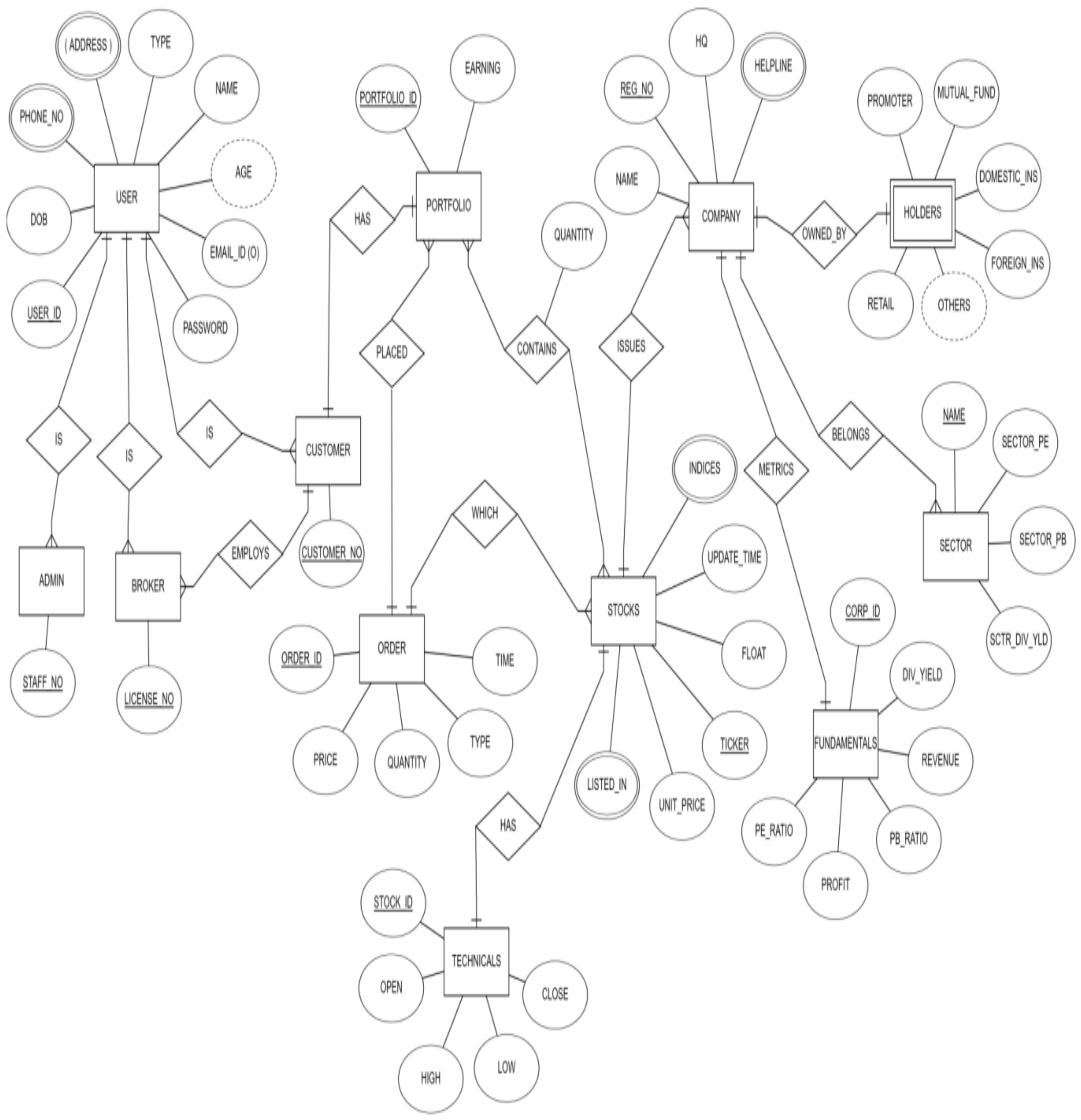
STOCKS_LISTED_IN (
stock_exchange VARCHAR(20),
stock_symbol VARCHAR(10) REFERENCES
STOCKS(stock_symbol))

STOCKS_INDICES (
index_name VARCHAR(20),
stock_symbol VARCHAR(10) REFERENCES
STOCKS(stock_symbol))

```
TECHNICALS (
technical_id NUMBER PRIMARY KEY,
high NUMBER(10,2),
low NUMBER(10,2),
open NUMBER(10,2),
close NUMBER(10,2),
stock_symbol VARCHAR(10) REFERENCES
STOCKS(stock_symbol))

CONTAINS (
portfolio_id NUMBER REFERENCES PORTFOLIO(portfolio_id),
stock_symbol VARCHAR(10) REFERENCES
STOCKS(stock_symbol), quantity NUMBER)

FUNDAMENTALS (
fundamental_id NUMBER PRIMARY KEY,
earnings_per_share NUMBER(10,2),
price_to_earnings_ratio NUMBER(10,2),
debt_to_equity_ratio NUMBER(10,2),
market_capitalization NUMBER,
dividend_yield NUMBER(10,2),
stock_symbol VARCHAR(10) REFERENCES
STOCKS(stock_symbol))

COMPANY (
company_id NUMBER PRIMARY KEY,
name VARCHAR(255),
location VARCHAR(255),
stock_symbol VARCHAR(10) REFERENCES
STOCKS(stock_symbol))
```

```
HOLDERS (
holder_id NUMBER PRIMARY KEY,
holder1 NUMBER(5,2),
holder2 NUMBER(5,2),
holder3 NUMBER(5,2),
holder4 NUMBER(5,2),
holder5 NUMBER(5,2),
stock_symbol VARCHAR(10) REFERENCES
STOCKS(stock_symbol))

SECTOR (
sector_name VARCHAR(255),
average_high NUMBER(10,2),
average_low NUMBER(10,2),
average_open NUMBER(10,2),
sector_id NUMBER PRIMARY KEY)

COMPANY_HELPLINE (
helpline_number NUMBER PRIMARY KEY,
company_id NUMBER REFERENCES COMPANY(company_id))
```

ER DIAGRAM


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

RELATIONAL MODEL

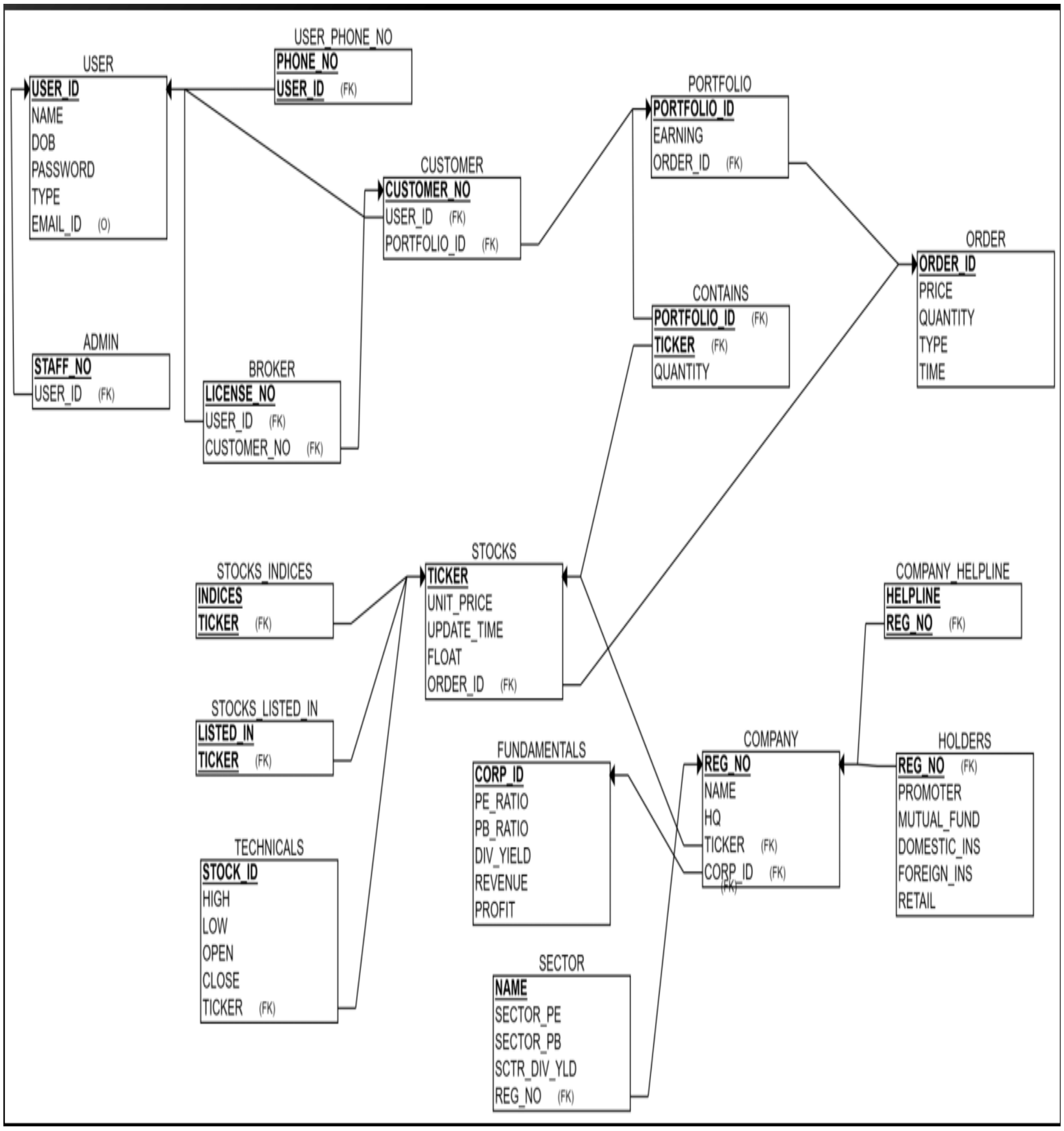


TABLE CREATION CODE

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

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

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

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)
);

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
);

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)
);
```

```
CREATE TABLE STOCKS_LISTED_IN(
LISTED_IN VARCHAR2(50),
TICKER VARCHAR2(50),
PRIMARY KEY(LISTED_IN,TICKER),
FOREIGN KEY (TICKER) REFERENCES STOCKS(TICKER)
);

CREATE TABLE STOCKS_INDICES(
STOCK_INDEX VARCHAR2(50),
TICKER VARCHAR2(50),
PRIMARY KEY(STOCK_INDEX,TICKER),
FOREIGN KEY (TICKER) REFERENCES STOCKS(TICKER)
);

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)
);

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 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)
);

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 )
);

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 )
);

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 )
);

CREATE TABLE company_helpline (
    helpline NUMBER(20),
    reg_no   NUMBER(20),
    PRIMARY KEY ( helpline,
                  reg_no ),
    FOREIGN KEY ( reg_no )
        REFERENCES company ( reg_no )
);
```

INSERTING VALUES

```
--USERS:
INSERT INTO USERS
VALUES('John',TO_DATE('1990-01-01','YYYY-MM-DD'),001,'1234','
Customer','john@example.com');
INSERT INTO USERS
VALUES('Steve',TO_DATE('1970-04-05','YYYY-MM-DD'),003,'efgh','
Admin','steve@example.com');
INSERT INTO USERS
VALUES('Mary',TO_DATE('1980-02-03','YYYY-MM-DD'),002,'abcd','
Broker','mary@example.com');
INSERT INTO USERS
VALUES('Richard',TO_DATE('1960-07-09','YYYY-MM-DD'),004,'ijkl',
'Customer','richard@example.com');
INSERT INTO USERS
VALUES('Mark',TO_DATE('1950-08-11','YYYY-MM-DD'),005,'mnop',
'Customer','mark@example.com');
INSERT INTO USERS VALUES('Jane Smith',
TO_DATE('1995-05-05','YYYY-MM-DD'), 013, 'pksf923', 'Broker',
'jane.smith@example.com');
INSERT INTO USERS VALUES('Robert Johnson',
TO_DATE('1985-12-31', 'YYYY-MM-DD'), 024, 'pvrm163', 'Broker',
'robert.johnson@example.com');
INSERT INTO USERS VALUES('Emily Davis',
TO_DATE('1992-03-15', 'YYYY-MM-DD'), 035, 'pasl1e33', 'Broker',
'emily.davis@example.com');
INSERT INTO USERS VALUES('William Brown',
TO_DATE('1998-07-20', 'YYYY-MM-DD'), 046, 'pasf23', 'Broker',
'william.brown@example.com');
```

```
INSERT INTO USERS VALUES('Anurag Nayak',
TO_DATE('1997-07-10', 'YYYY-MM-DD'), 069, 'paef23', 'Customer',
'anurag@example.com');
INSERT INTO USERS VALUES('Kapardhi', TO_DATE('1996-08-24',
'YYYY-MM-DD'), 073, 'pasft3', 'Customer', 'kappa@example.com');

SELECT * FROM USERS;

--USER_PHONE_NO:
INSERT INTO USER_PHONE_NO VALUES(1234567890,001);
INSERT INTO USER_PHONE_NO VALUES(2345678901,002);
INSERT INTO USER_PHONE_NO VALUES(3456789012,003);
INSERT INTO USER_PHONE_NO VALUES(4567890123,004);
INSERT INTO USER_PHONE_NO VALUES(5678901234,005);
INSERT INTO USER_PHONE_NO VALUES(6789012345,013);
INSERT INTO USER_PHONE_NO VALUES(7890123456,013);
INSERT INTO USER_PHONE_NO VALUES(8901234567,024);
INSERT INTO USER_PHONE_NO VALUES(9012345678,024);
INSERT INTO USER_PHONE_NO VALUES(1234567890,035);
INSERT INTO USER_PHONE_NO VALUES(2345678901,035);
INSERT INTO USER_PHONE_NO VALUES(3456789012,046);
INSERT INTO USER_PHONE_NO VALUES(4567890123,046);
INSERT INTO USER_PHONE_NO VALUES(5678901234,046);


SELECT * FROM USER_PHONE_NO;

--ADMINS:
INSERT INTO ADMINS VALUES(01, 003);
```

```
SELECT * FROM ADMINS;

--BROKER:
INSERT INTO BROKER VALUES(10001,002,20001);
INSERT INTO BROKER VALUES(10002,013,20002);
INSERT INTO BROKER VALUES(10003,024,20003);
INSERT INTO BROKER VALUES(10004,035,20004);
INSERT INTO BROKER VALUES(10005,046,20005);

SELECT * FROM BROKER;

--CUSTOMER:
INSERT INTO CUSTOMER VALUES(20001,001,30001);
INSERT INTO CUSTOMER VALUES(20002,004,30002);
INSERT INTO CUSTOMER VALUES(20003,005,30003);
INSERT INTO CUSTOMER VALUES(20004,069,30004);
INSERT INTO CUSTOMER VALUES(20005,073,30005);

select * from customer;

--PORTFOLIO:
INSERT INTO PORTFOLIO VALUES(30001,100000,40001);
INSERT INTO PORTFOLIO VALUES(30002,200000,40002);
INSERT INTO PORTFOLIO VALUES(30003,300000,40003);
INSERT INTO PORTFOLIO VALUES(30004,200000,40004);
INSERT INTO PORTFOLIO VALUES(30005,500000,40005);

select * from portfolio;

--ORDERS:
```

```
INSERT INTO ORDERS
VALUES(40001,1000,10,'BUY',TO_TIMESTAMP('2020-01-01
09:30:00','YYYY-MM-DD HH24:MI:SS'));
INSERT INTO ORDERS
VALUES(40002,1500,20,'SELL',TO_TIMESTAMP('2020-02-02
10:30:00','YYYY-MM-DD HH24:MI:SS'));
INSERT INTO ORDERS
VALUES(40003,2000,30,'BUY',TO_TIMESTAMP('2020-03-03
11:30:00','YYYY-MM-DD HH24:MI:SS'));
INSERT INTO ORDERS
VALUES(40004,2500,40,'SELL',TO_TIMESTAMP('2020-04-04
12:30:00','YYYY-MM-DD HH24:MI:SS'));
INSERT INTO ORDERS
VALUES(40005,3000,50,'BUY',TO_TIMESTAMP('2020-05-05
13:30:00','YYYY-MM-DD HH24:MI:SS'));

select * from orders;

--STOCKS:
INSERT INTO STOCKS
VALUES('AAPL',125.50,TO_TIMESTAMP('2020-01-01
09:30:00','YYYY-MM-DD HH24:MI:SS'),1000,40001);
INSERT INTO STOCKS
VALUES('MSFT',110.20,TO_TIMESTAMP('2020-02-02
10:30:00','YYYY-MM-DD HH24:MI:SS'),1500,40002);
INSERT INTO STOCKS
VALUES('AMZN',1790.23,TO_TIMESTAMP('2020-03-03
11:30:00','YYYY-MM-DD HH24:MI:SS'),2000,40003);
```

```
INSERT INTO STOCKS
VALUES('FB',195.32,TO_TIMESTAMP('2020-04-04
12:30:00','YYYY-MM-DD HH24:MI:SS'),2500,40004);
INSERT INTO STOCKS
VALUES('GOOGL',1280.39,TO_TIMESTAMP('2020-05-05
13:30:00','YYYY-MM-DD HH24:MI:SS'),3000,40005);

select * from stocks;

--STOCKS_LISTED_IN:
INSERT INTO STOCKS_LISTED_IN VALUES('NASDAQ','AAPL');
INSERT INTO STOCKS_LISTED_IN VALUES('NYSE','MSFT');
INSERT INTO STOCKS_LISTED_IN VALUES('NASDAQ','AMZN');
INSERT INTO STOCKS_LISTED_IN VALUES('NYSE','FB');
INSERT INTO STOCKS_LISTED_IN
VALUES('NASDAQ','GOOGL');

select * from stocks_listed_in;

--STOCKS_INDICES:
INSERT INTO STOCKS_INDICES VALUES('NASDAQ 100','AAPL');
INSERT INTO STOCKS_INDICES VALUES('NYSE 100','MSFT');
INSERT INTO STOCKS_INDICES VALUES('NASDAQ
100','AMZN');
INSERT INTO STOCKS_INDICES VALUES('NYSE 100','FB');
INSERT INTO STOCKS_INDICES VALUES('NASDAQ
100','GOOGL');

select * from stocks_indices;
```

```
--TECHNICALS:
INSERT INTO TECHNICALS
VALUES(1,130.65,120.23,125.53,126.12,'AAPL');
INSERT INTO TECHNICALS
VALUES(2,115.34,105.62,110.43,109.34,'MSFT');
INSERT INTO TECHNICALS
VALUES(3,1800.31,1730.23,1770.23,1771.24,'AMZN');
INSERT INTO TECHNICALS
VALUES(4,200.43,190.65,199.12,199.54,'FB');
INSERT INTO TECHNICALS
VALUES(5,1290.65,1250.65,1280.72,1282.39,'GOOGL');

select * from technicals;

--CONTAINS:
INSERT INTO CONTAINS VALUES(30001,'AAPL',100);
INSERT INTO CONTAINS VALUES(30002,'MSFT',100);
INSERT INTO CONTAINS VALUES(30003,'AMZN',50);
INSERT INTO CONTAINS VALUES(30004,'FB',200);
INSERT INTO CONTAINS VALUES(30005,'GOOGL',150);

select * from contains;

--FUNDAMENTALS:
INSERT INTO FUNDAMENTALS
VALUES(1,3.45,8.23,2.43,10000,2000);
INSERT INTO FUNDAMENTALS
VALUES(2,2.43,6.32,1.5,20000,5000);
INSERT INTO FUNDAMENTALS
VALUES(3,5.54,1.23,0.23,30000,10000);
```

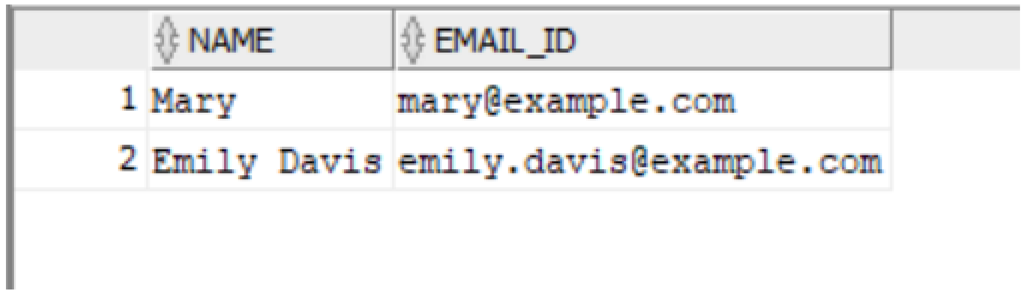
```
INSERT INTO FUNDAMENTALS
VALUES(4,4.23,1.63,1.45,40000,15000);
INSERT INTO FUNDAMENTALS
VALUES(5,3.45,5.32,2.5,50000,20000);

select * from fundamentals;

--COMPANY:
INSERT INTO COMPANY VALUES(1,'Apple
Inc','Cupertino','AAPL',1);
INSERT INTO COMPANY VALUES(2,'Microsoft
Corp','Redmond','MSFT',2);
INSERT INTO COMPANY VALUES(3,'Amazon
Inc','Seattle','AMZN',3);
INSERT INTO COMPANY VALUES(4,'Facebook Inc','Menlo
Park','FB',4);
INSERT INTO COMPANY VALUES(5,'Alphabet Inc','Mountain
View','GOOGL',5);

select * from company;

--HOLDERS:
INSERT INTO HOLDERS VALUES (1, 50.00, 25.00, 10.00, 10.00,
5.00);
INSERT INTO HOLDERS VALUES (2, 60.00, 20.00, 5.00, 10.00,
5.00);
INSERT INTO HOLDERS VALUES (3, 40.00, 15.00, 25.00, 10.00,
10.00);
INSERT INTO HOLDERS VALUES (4, 45.00, 10.00, 20.00, 15.00,
10.00);
```

```
INSERT INTO HOLDERS VALUES (5, 55.00, 5.00, 20.00, 15.00,
5.00);

select * from holders;

--SECTOR:
INSERT INTO SECTOR VALUES('Technology',5.34,1.43,1.5,1);
INSERT INTO SECTOR VALUES('Technology',4.53,8.75,1.23,2);
INSERT INTO SECTOR VALUES('Delivery',6.63,1.23,0.43,3);
INSERT INTO SECTOR VALUES('Social media',4.32,1.64,1.25,4);
INSERT INTO SECTOR VALUES('Technology',5.67,6.3,2.3,5);

select * from sector;

--COMPANY_HELPLINE:
INSERT INTO COMPANY_HELPLINE VALUES(444111222,1);
INSERT INTO COMPANY_HELPLINE VALUES(333111222,2);
INSERT INTO COMPANY_HELPLINE VALUES(222111222,3);
INSERT INTO COMPANY_HELPLINE VALUES(111331122,4);
INSERT INTO COMPANY_HELPLINE VALUES(123321123,5);

select * from company_helpline;
```

QUERIES

1)Retrieve the names and email addresses of all brokers who have a phone number starting with '234'.

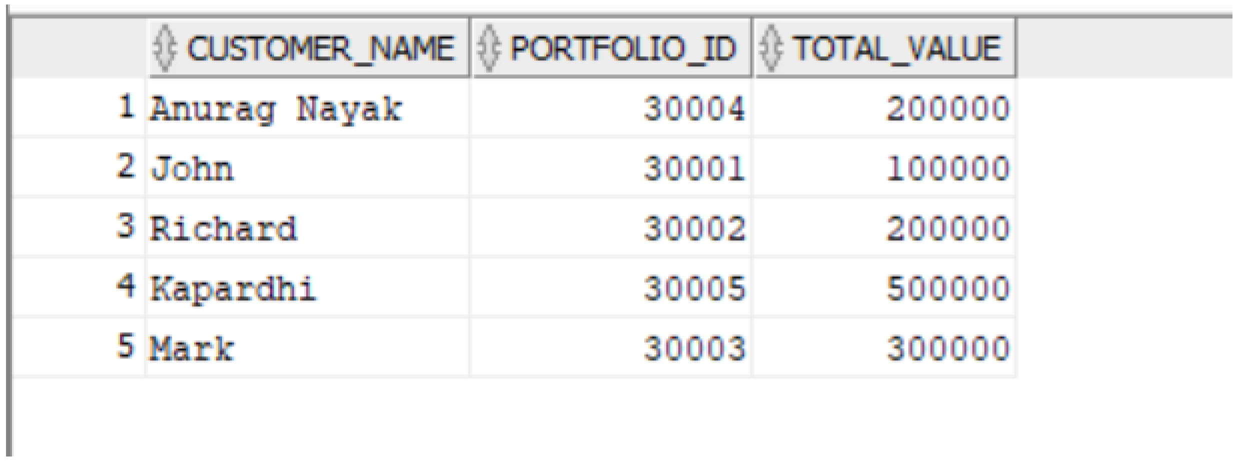
```
SELECT u.NAME, u.EMAIL_ID
FROM USERS u
JOIN BROKER b ON u.USER_ID = b.USER_ID
JOIN USER_PHONE_NO up ON up.USER_ID =
b.USER_ID
WHERE up.PHONE_NO LIKE '234%';
```



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

2)Retrieve the total value of each portfolio along with the corresponding customer name

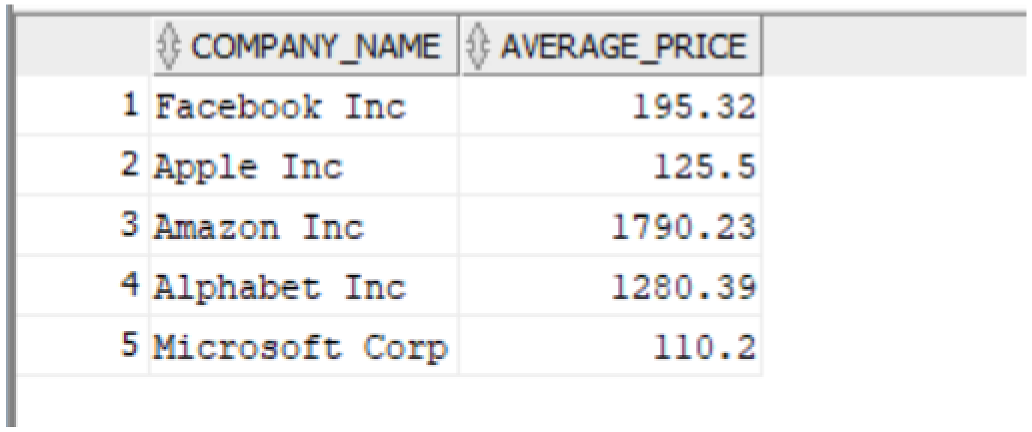
```
Select U.Name As Customer_Name, P.Portfolio_Id, Sum(P.Earning) As
Total_Value
From Customer C
Join Users U On C.User_Id = U.User_Id
Join Portfolio P On C.Portfolio_Id =
P.Portfolio_Id Group By U.Name, P.Portfolio_Id;
```



	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

3)Retrieve the average price of stocks for each company.

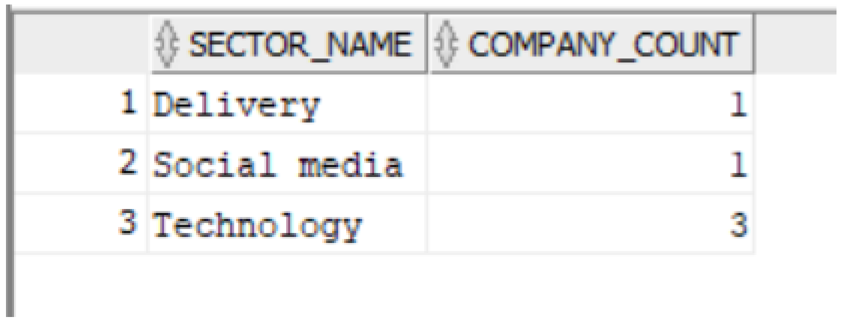
```
SELECT c.company_name, AVG(s.unit_price) AS average_price
FROM company c
JOIN stocks s ON c.ticker = s.ticker
GROUP BY c.company_name;
```



	COMPANY_NAME	AVERAGE_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

4)Retrieve the sector along with the count of companies in that sector.

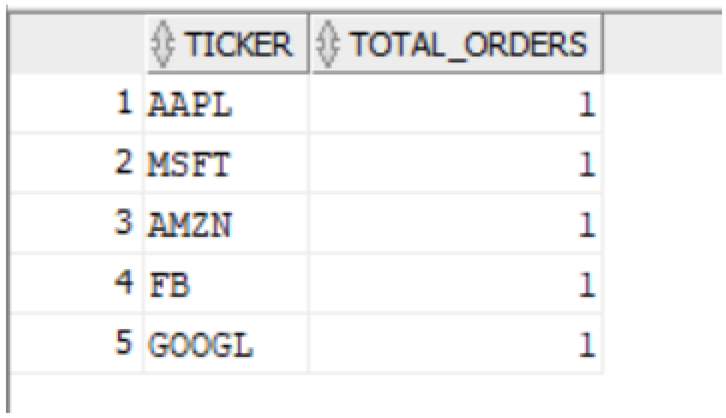
```
SELECT sector_name, COUNT(*) AS company_count
FROM sector
GROUP BY sector_name;
```



	SECTOR_NAME	COMPANY_COUNT
1	Delivery	1
2	Social media	1
3	Technology	3

5)Retrieve the total number of orders placed for each stock.

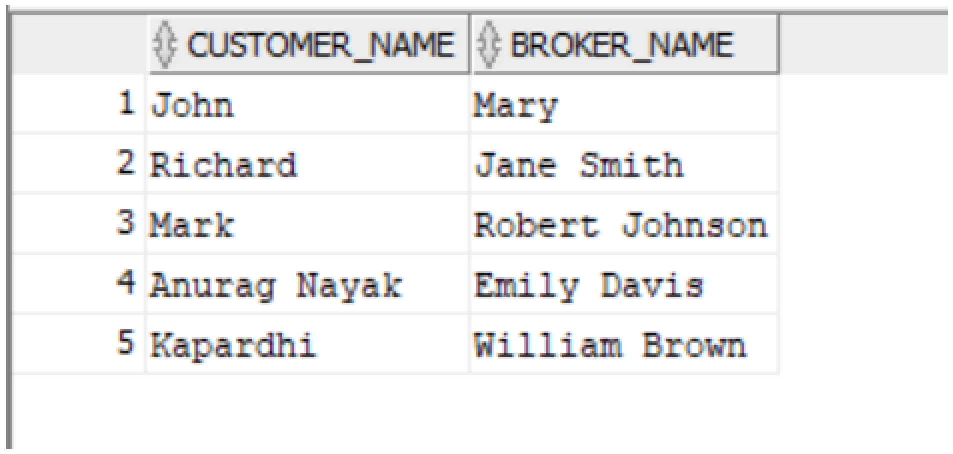
```
SELECT s.ticker, COUNT(*) AS total_orders
FROM orders o
JOIN stocks s ON o.order_id = s.order_id
GROUP BY s.ticker;
```



	TICKER	TOTAL_ORDERS
1	AAPL	1
2	MSFT	1
3	AMZN	1
4	FB	1
5	GOOGL	1

6)Retrieve the names of the customers along with their brokers' names.

```
SELECT c.name AS customer_name, b.name AS broker_name
FROM customer cu
JOIN users c ON cu.user_id = c.user_id
JOIN broker br ON cu.customer_no = br.customer_no
JOIN users b ON br.user_id = b.user_id;
```



	CUSTOMER_NAME	BROKER_NAME
1	John	Mary
2	Richard	Jane Smith
3	Mark	Robert Johnson
4	Anurag Nayak	Emily Davis
5	Kapardhi	William Brown

LEARNINGS AND OUTCOMES

1. Data organization: The database design demonstrates how to organize and store data related to a financial trading system. It highlights the importance of properly structuring tables and establishing relationships between them to ensure efficient data management.

2. Normalization: The use of normalized tables (e.g., USERS, CUSTOMER, BROKER) helps eliminate data redundancy and maintain data consistency. It promotes data integrity and reduces the risk of inconsistencies or anomalies.

3. Entity-relationship modeling: The database design showcases the relationships between entities, such as USERS, CUSTOMER, BROKER, and their associated attributes. It provides insights into how different entities are connected and how data flows between them.

4. Role-based access control: The inclusion of roles (e.g., Admin, Broker, Customer) in the USERS table allows for role-based access control. It enables different levels of permissions and restrictions based on the user's role, ensuring data security and privacy.

5. Data integrity and constraints: The use of primary keys, foreign keys, and constraints (e.g., REFERENCES, PRIMARY KEY) enforces data integrity rules. It helps maintain the consistency and accuracy of data by preventing invalid or inconsistent entries.

6. Data analysis and reporting: The database design includes tables like ORDERS, STOCKS, PORTFOLIO, and TECHNICALS, which facilitate data analysis and reporting. It enables tracking and analysis of

stock trades, portfolio performance, technical indicators, and other relevant information.

7. Scalability and extensibility: The modular design of the database allows for scalability and extensibility. New entities or features can be added easily by creating additional tables and establishing appropriate relationships with existing entities.

8. Data consistency and accuracy: The use of referential integrity through foreign key constraints ensures data consistency and accuracy. It helps maintain the relationships between entities, preventing orphaned or inconsistent data.

9. Efficient data retrieval: The database design supports efficient data retrieval through appropriate indexing and query optimization. It enables fast and reliable access to relevant information, improving system performance.

10. Integration with external systems: The database design can be integrated with external systems, such as stock market data feeds or reporting tools, to enhance functionality and provide real-time updates.

Overall, the database design provides a foundation for building a robust and scalable financial trading system. It emphasizes the importance of data organization, normalization, relationship management, and data integrity to ensure accurate, secure, and efficient data management and analysis.

CONCLUSION

In conclusion, the database project for the financial trading system demonstrates the effective organization, management, and analysis of data related to stock trading, user information, portfolios, orders, and other relevant entities. The project showcases the application of normalization principles, entity-relationship modeling, and data integrity constraints to ensure data consistency, accuracy, and security.

By implementing a relational database schema and establishing relationships between tables, the project enables efficient data retrieval and analysis. It allows for the tracking and management of user information, stock trades, portfolio performance, technical indicators, and other crucial data points.

The project highlights the importance of proper database design in supporting scalable and extensible systems. It demonstrates the modular structure that allows for the addition of new entities and features without disrupting existing functionality. The use of primary and foreign keys, along with constraints, enforces data integrity and prevents inconsistencies or invalid entries.

Overall, the database project serves as a valuable foundation for building a robust and efficient financial trading system. It provides a structured approach to store, retrieve, and analyze data, facilitating informed decision-making and enhancing the overall trading experience. The project's design choices and implementation demonstrate best practices in database management and offer insights into the complexities of handling financial data in a real-world setting.