TABLE OF CONTENTS

LIST OF FIGURES

TEAM INTRODUCTION

The global financial system depends heavily on stocks, which have an impact on the monetary movement globally. Managing an investment portfolio has become extremely important in today's financial landscape to reach long-term financial objectives. This leads us to our project topic **Stock Investment Portfolio Management** as the database management systems play a crucial role in organizing, storing, and analysing large volumes of financial data.

Our team, **'The Bulls of Wall Street'** is composed of professionals with diverse expertise, each contributing to the development and implementation of our project. The team members of our project are as follows:

Adheesh Ghotikar

Adheesh graduated with Bachelor of Engineering degree in Computer Science Engineering from Osmania University in the year 2022. He has worked as a software developer and an automation engineer with ADP, streamlining integrity verification process of multiple ADP vendors thereby, eliminating manual health checks, saving 3+ hours per day. Adheesh has more than 1 year of industry experience.

Aishwarya Ashok Medhe

Aishwarya Ashok has worked as a software and network consulting engineer with Cisco for over 3 years wherein she crafted proactive and insightful solutions for the clients while creating a secure IT strategy for the future. She defined and implemented technical roadmaps spanning across different industry verticals. Aishwarya Ashok graduated with Bachelor of Engineering degree with major in Information Technology from University of Mumbai in the year 2020.

Nagajyothi MS

Nagajyothi has worked as a full-time data engineer with KPMG and has worked on various ERP systems. Prior to this, she has worked as a cybersecurity and ethical hacking intern with Tequed Labs and has an overall experience of 2 years. Nagajyothi graduated in the year 2022 with Bachelor of Engineering degree with Computer Science major.

Pranjal Mestry

Pranjal has worked as a data engineer with TCS for 3 years. During his tenure at TCS, he developed ETL pipelines for processing gis data for consumption by downstream applications. Pranjal graduated with Bachelor of Engineering degree with Electronics and Telecommunication major in the year 2021.

Vishak Viswanathan

Vishak has worked as a dot net developer with TCS. He comes with 3 years of experience in this field. Vishak graduated with a Bachelor of Engineering degree with a major in Computer Science and Engineering. Vishak has worked in designing, implementing, testing, and maintaining business-critical applications using dot net technology.

Description

The stock market portfolio management system provides a detailed platform for users to manage, retrieve, and track their stock investments. The system is equipped with capabilities such as user management, detailed portfolio tracking information, stock order transactions, archived order transactions, archived market data, and additional features like stock watchlist management. The stock investment portfolio management system is a highly consistent system that allows users to view and get the most recent information on stock prices, trading volume, ask prices, bid prices, order execution, and other market data. The system must be extremely available and cannot tolerate downtime, enabling users to execute stock orders and get portfolio information in real time.

The project's scope

The scope of this project is limited to the equity investments of the stocks and does not tap into the commodity, F&O, or derivatives sections.

Key features of the platform are as follows:

User Management

- Users can create, log in, and manage their platform account securely.
- Users can update their profile information, such as their email, passwords, phone number, and bank account.
- Users can view or download the P&L statements and tax P&L for a given time frame.
- Users can add or withdraw funds to and from the platform.

Portfolio Management

- Users can create multiple portfolios and organize them based on the sector of investment, the purpose of the investment, or the investment strategy.
- Users could track the stocks held by them in the portfolios, which include data such as name, quantity (units of stocks), value of purchase, running P&L, invested amount, etc., in real time.
- Users can track logged buy and sell orders for stocks that were executed in the past.
- Users can transfer stocks from one portfolio to another.

Order Transaction Management.

- Users can place buy or sell orders for the stocks in real time based on the following parameters:

a. Order type (buy or sell)
b. Price
c. Stop Loss
d. Product (intraday [Mis] or long-term [CNC])
e. Order Market Type (Market, Limit, SL, or M)

f. Quantity
g. Price trigger.

- Users can update the parameters of the orders after they are placed and before they are executed.
- Once the stock is squared off, the resultant P&L will be added to or deducted from the funds.
- Users will not be able to place a sell order if that stock is not held by them in their portfolio.
- The brokerage fee and taxes will be calculated and populated when the order is placed.

Market Data Retrieval

- Provides information about stocks such as name, sector, industry, real-time market price, market capitalization, and other fundamental parameters of the company.
- Access historical data on stocks, such as the open value, close value, high, low, and volume of that trading session.
- Integrates real-time market data updates for precise portfolio valuation.

Watch list creation.

- Users can add multiple stocks of their interest to multiple watch lists based on sector, investment type, or strategy for monitoring them.

Scalability and access.

- Implementation of partition and indexing strategies for reading and writing large volumes of data.
- Database sharding and replication strategies will be implemented to tackle the exponential growth of the data.
- Database triggers and constraints will be implemented when performing crud operations on certain tables as per the stock market rules.
- Implementation of encryption protocols to secure user-sensitive and market-sensitive information.

ENTITY RELATIONSHIP DIAGRAM

The ER diagram depicts the fundamental entities and relationships in the Stock Portfolio Management Database. In the ER diagram, the parameters that are bold and underlined are the primary keys, and the ones that are highlighted in bold are the foreign keys.

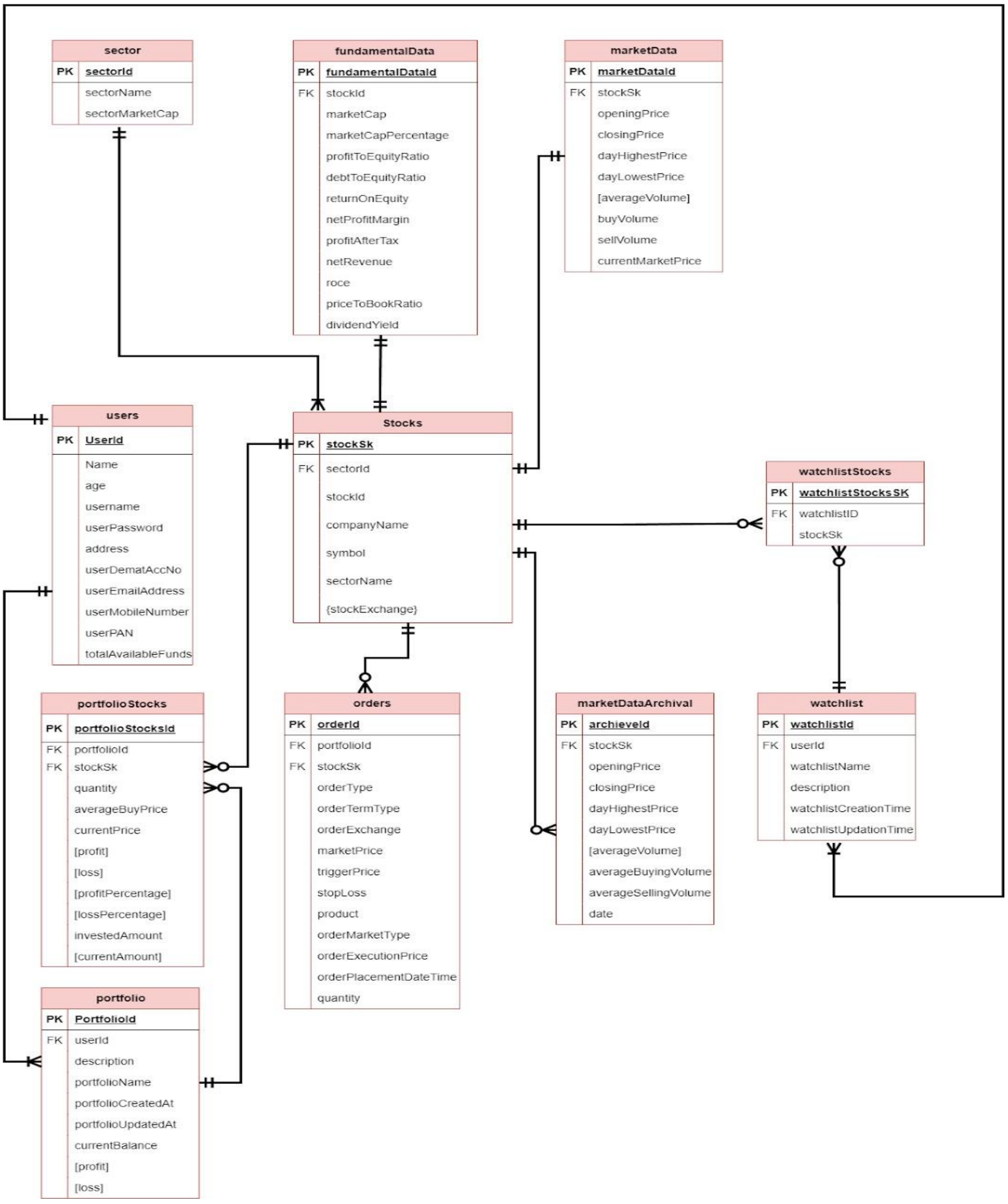


Fig 1.1: Entity Relationship Diagram for Stock Market Portfolio Management System

Entity Relationship diagram summary

- **Portfolio- Stock:** A portfolio can have either zero or multiple stocks. This is an optional many-to-many relationship. To build this relationship, an associative entity named **PortfolioStock** is used. The cardinality between '**stock**' and '**portfolioStock**' is mandatory one-to-optional-many.
- **Stock – marketData**: The cardinality between '**stock**' and '**marketData**' is mandatory one-to-one; that is, every Stock should have a single record in the '**marketData**'. '**marketData**' contains real time data of the stocks, which fluctuates every second, making this table read and write heavy. Therefore, it is separated from the '**stock**' table to minimize the load on read and write intensive operations.
- **Stock – marketDataArchival:** The cardinality between '**stock**' and '**marketData**' is mandatory one-to-optional-many. For every stock that is already listed, it can have a '**marketDataArchival**' record, but if the stock is newly listed someday, then it will not have a record in '**marketDataArchival'.**
- **Sector – Stock**: The cardinality between '**sector**' and '**stock**' is mandatory one-to-many. In a sector, there can be multiple stocks listed, but a stock should be listed under only one sector.
- **Stock-fundamentalData:** The cardinality between '**stock**' and '**fundamentalData**' is mandatory one-to-one. Each stock has a record present in the '**fundamentalData**' table, which defines all the fundamental parameters of the company's stock.
- **User – Portfolio:** The cardinality between '**user**' and '**portfolio**' is mandatory one-to-many. That means a single user should have at least one portfolio or can have multiple portfolios.
- **Portfolio – Orders:** The cardinality between '**portfolio**' and '**orders**' is one-to-optional many. That is, a portfolio may contain zero or multiple orders.
- **Stock – Order:** The cardinality between '**stock**' to '**Order**' is mandatory one-to-optional many. A stock can have multiple orders but an order is associated with only one stock.
- **User – Watchlist:** The cardinality between '**user**' and '**watchlist**' is mandatory one-to-many. A user can have one or more watchlists. If the user does not create a watchlist a default watchlist is assigned by the platform to the user.
- **Stock – Watchlist:** The cardinality between '**Stock**' to '**watchlist**' is optional many-to-many. A watchlist can have either zero or multiple stocks. Similarly, a stock may or may not be mapped to a watchlist. An associative entity called '**watchlistStock**' is used to establish a relationship between stock and Watchlist. The cardinality between stock and '**watchlistStock**' is mandatory one-to-optional-many. The cardinality between Watchlist to '**watchlistStock**' is mandatory one-to-optional-many.

Business rules

- All users must have a unique username and email id.
- Passwords and other sensitive information about the user must be encrypted.
- A portfolio must be associated with a user.
- A user must have different portfolio names under their account.
- Each stock has a unique symbol.
- Every order transaction must be associated with a particular stock.
- Order transactions must be either of buy or sell type.
- Order transactions can be either in pending or in executed state.
- All users must have an adequate amount of funds in their account to execute a buy order.
- All users must have an adequate quantity of stocks to complete a sell transaction.
- Every order can have only one order market type.
- There are two 'orderTermType', MIS (intraday) and CNC (Long term) and four 'orderMarketType' named limit order, SL, SL-M and GTT.
- Users will have a default watchlist which will be assigned by the platform.

Physical Diagram for Stock Portfolio Management System

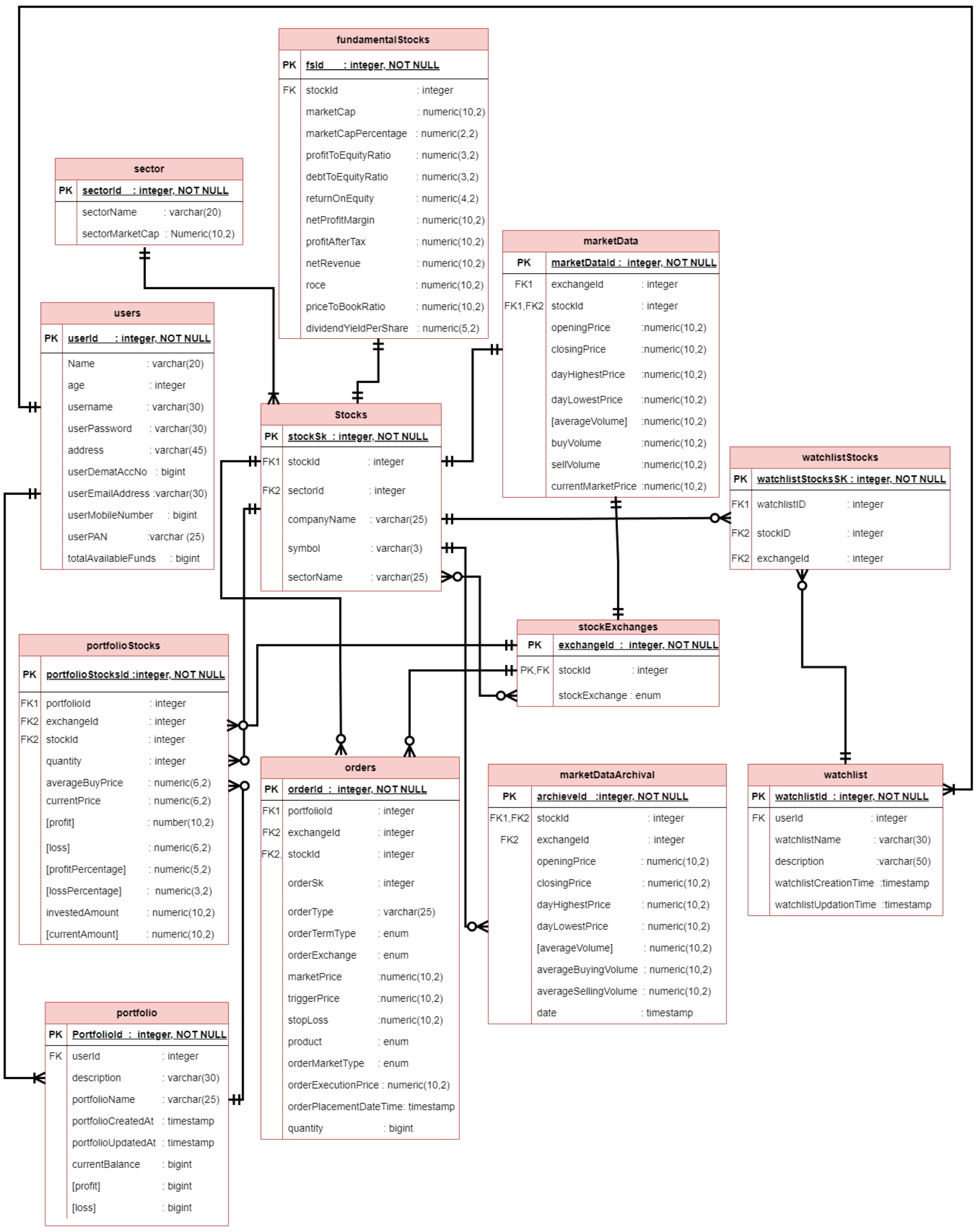

fundamentalStocks
PK fsId : integer, NOT NULL
FK stockId : integer
marketCap : numeric(10,2)
marketCapPercentage : numeric(2,2)
profitToEquityRatio : numeric(3,2)
debtToEquityRatio : numeric(3,2)
returnOnEquity : numeric(4,2)
netProfitMargin : numeric(10,2)
profitAfterTax : numeric(10,2)
netRevenue : numeric(10,2)
roce : numeric(10,2)
priceToBookRatio : numeric(10,2)
dividendYieldPerShare : numeric(5,2)
sector
PK sectorId : integer, NOT NULL
sectorName : varchar(20)
sectorMarketCap : Numeric(10,2)
marketData
PK marketDataId : integer, NOT NULL
FK1 exchangeId : integer
FK1,FK2 stockId : integer
openingPrice :numeric(10,2)
closingPrice :numeric(10,2)
dayHighestPrice :numeric(10,2)
dayLowestPrice :numeric(10,2)
[averageVolume] :numeric(10,2)
buyVolume :numeric(10,2)
sellVolume :numeric(10,2)
currentMarketPrice :numeric(10,2)
users
PK userId : integer, NOT NULL
Name : varchar(20)
age : integer
username : varchar(30)
userPassword : varchar(30)
address : varchar(45)
userDematAccNo : bigint
userEmailAddress :varchar(30)
userMobileNumber : bigint
userPAN :varchar (25)
totalAvailableFunds : bigint
Stocks
PK stockSk : integer, NOT NULL
FK1 stockId : integer
FK2 sectorId : integer
companyName : varchar(25)
symbol : varchar(3)
sectorName : varchar(25)
watchlistStocks
PK watchlistStocksSK : integer, NOT NULL
FK1 watchlistID : integer
FK2 stockID : integer
FK2 exchangeId : integer
stockExchanges
PK exchangeId : integer, NOT NULL
PK,FK stockId : integer
stockExchange : enum
portfolioStocks
PK portfolioStocksId :integer, NOT NULL
FK1 portfolioId : integer
FK2 exchangeId : integer
FK2 stockId : integer
quantity : integer
averageBuyPrice : numeric(6,2)
currentPrice : numeric(6,2)
[profit] : number(10,2)
[loss] : numeric(6,2)
[profitPercentage] : numeric(5,2)
[lossPercentage] : numeric(3,2)
investedAmount : numeric(10,2)
[currentAmount] : numeric(10,2)
orders
PK orderId : integer, NOT NULL
FK1 portfolioId : integer
FK2 exchangeId : integer
FK2, stockId : integer
orderSk : integer
orderType : varchar(25)
orderTermType : enum
orderExchange : enum
marketPrice :numeric(10,2)
triggerPrice :numeric(10,2)
stopLoss :numeric(10,2)
product : enum
orderMarketType : enum
orderExecutionPrice : numeric(10,2)
orderPlacementDateTime: timestamp
quantity : bigint
marketDataArchival
PK archieveId :integer, NOT NULL
FK1,FK2 stockId : integer
FK2 exchangeId : integer
openingPrice : numeric(10,2)
closingPrice : numeric(10,2)
dayHighestPrice : numeric(10,2)
dayLowestPrice : numeric(10,2)
[averageVolume] : numeric(10,2)
averageBuyingVolume : numeric(10,2)
averageSellingVolume : numeric(10,2)
date : timestamp
watchlist
PK watchlistId : integer, NOT NULL
FK userId : integer
watchlistName : varchar(30)
description :varchar(50)
watchlistCreationTime :timestamp
watchlistUpdationTime :timestamp
portfolio
PK PortfolioId : integer, NOT NULL
FK userId : integer
description : varchar(30)
portfolioName : varchar(25)
portfolioCreatedAt : timestamp
portfolioUpdatedAt : timestamp
currentBalance : bigint
[profit] : bigint
[loss] : bigint

Fig 1.2: Physical Diagram for Stock Market Portfolio Management System

DDL commands:

```
create schema stockPortfolioMGM;
```

Defining the ENUM for stockExchange and orders table

```
CREATE TYPE stockPortfolioMGM.orderTypeEnum AS ENUM ('BUY','SELL');
CREATE TYPE stockPortfolioMGM.orderTermTypeEnum AS ENUM ('MIS', 'CNC');
CREATE TYPE stockPortfolioMGM.orderExchangeEnum AS ENUM ('NSE', 'BSE');
CREATE TYPE stockPortfolioMGM.productEnum AS ENUM ('REGULAR', 'AMO', 'ACEBERG');
CREATE TYPE stockPortfolioMGM.orderMarketTypeEnum AS ENUM ('LIMIT', 'MARKET','SL','SL-M');
create type stockPortfolioMGM.stockExchangeType as enum ('NSE','BSE');
create type stockPortfolioMGM.orderstatusenum as enum ('OPEN','CLOSE');
```

DDL command for stockPortfolioMGM.sector

```
CREATE TABLE stockPortfolioMGM.sector (
   sectorId  SERIAL PRIMARY KEY NOT NULL,
   sectorName VARCHAR(30)  NOT NULL,
   sectorMarketCap NUMERIC(10,2) NOT NULL
);
```

Creation stockPortfolio

```
CREATE TABLE stockPortfolioMGM.stocks (
   stockSk SERIAL  PRIMARY KEY NOT NULL,
   stockId INT  UNIQUE NOT NULL,
   sectorId INT NOT NULL,
   companyName VARCHAR(50) NOT NULL,
   symbol VARCHAR(30) NOT NULL,
   FOREIGN KEY (sectorId) REFERENCES stockPortfolioMGM.sector(sectorId) ON DELETE CASCADE
);
CREATE INDEX idx_stockportfoliomgm_stocks_stocksk ON stockPortfolioMGM.stocks USING BTREE
(stocksk);
CREATE INDEX idx_stockportfoliomgm_stocks_stockid ON stockPortfolioMGM.stocks USING BTREE
(stockid);
CREATE SEQUENCE stockportfoliomgm.stockPortfoliomgm_stocks_stockId_seq;
ALTER TABLE stockportfoliomgm.stocks
ALTER COLUMN stockId SET DEFAULT
nextval('stockportfoliomgm.stockPortfoliomgm_stocks_stockId_seq');
```

DDL command for stockPortfolioMGM.fundamentalStocks

```
CREATE TABLE stockPortfolioMGM.fundamentalStocks (
   fsId SERIAL PRIMARY KEY NOT NULL,
   stockId INT NOT NULL,
   marketCap NUMERIC(10,2) NOT NULL,
   marketCapPercentage NUMERIC(2,2) NOT NULL,
   profitToEquityRatio NUMERIC(3,2) NOT NULL,
   debtToEquityRatio NUMERIC(3,2) NOT NULL,
   returnOnEquity NUMERIC(3,2) NOT NULL,
   netProfitMargin NUMERIC(10,2) NOT NULL,
   profitAfterTax NUMERIC(10,2) NOT NULL,
   netRevenue NUMERIC(10,2) NOT NULL,
   roce NUMERIC(3,2) NOT NULL,
   priceToBookRatio NUMERIC(3,2) NOT NULL,
   dividendYield NUMERIC(3,2),
   FOREIGN KEY (stockId) REFERENCES stockPortfolioMGM.stocks(stockId) ON DELETE CASCADE
);
CREATE INDEX idx_stockportfoliomgm_fundamental_stocks_stockId ON
stockPortfolioMGM.fundamentalStocks USING BTREE (fsId);
```

DDL command for stockPortfolioMGM.users

```
userId SERIAL  PRIMARY KEY NOT NULL,
   Name VARCHAR(50) NOT NULL,
   age INT NOT NULL,
   username VARCHAR(30) NOT NULL,
   userPassword VARCHAR(30) NOT NULL,
   address VARCHAR(100) NOT NULL,
   userDematAccNo VARCHAR(20) NOT NULL,
   userEmailAddress VARCHAR(30) NOT NULL,
   userMobileNumber NUMERIC(10,0),
   userPAN VARCHAR(20) NOT NULL,
   totalAvailableFunds NUMERIC(10,2)
);
CREATE INDEX idx_stockportfoliomgm_users_userId ON stockPortfolioMGM.users USING BTREE (userId);
```

DDL command for stockPortfolioMGM.portfolio

```
portfolioId SERIAL  PRIMARY KEY NOT NULL,
   userId INT NOT NULL,
   description VARCHAR(100),
   portfolioName VARCHAR(50) NOT NULL,
   portfolioCreatedAt TIMESTAMP NOT NULL,
   portfolioUpdatedAt TIMESTAMP NOT NULL,
   currentBalance NUMERIC(10,2) NOT NULL,
   profit NUMERIC(10,2) NOT NULL,
   loss NUMERIC(10,2) NOT NULL,
   FOREIGN KEY (userId) REFERENCES stockPortfolioMGM.users(userId) ON DELETE CASCADE
);
CREATE INDEX idx_stockportfoliomgm_portfolio_portfolioId ON stockPortfolioMGM.portfolio USING
BTREE (portfolioId);
```

DDL command for stockPortfolioMGM.stockExchanges

```
CREATE TABLE stockPortfolioMGM.stockExchanges (
   exchangeId SERIAL unique  NOT NULL,
   stockId INTEGER NOT NULL,
   stockExchange stockPortfolioMGM.stockExchangeType NOT NULL,
   PRIMARY KEY (exchangeId, stockId),
   FOREIGN KEY (stockId) REFERENCES stockPortfolioMGM.stocks(stockId) ON DELETE CASCADE
);

CREATE INDEX idx_stockExchanges_exchangeId_stockId ON stockPortfolioMGM.stockExchanges USING
BTREE (exchangeId, stockId);
```

DDL command for stockPortfolioMGM.portfolioStocks

```
CREATE TABLE stockPortfolioMGM.portfolioStocks (
   portfolioStocksId SERIAL  PRIMARY KEY NOT NULL,
   portfolioId INT NOT NULL,
   exchangeId INT NOT NULL,
   stockId INT NOT NULL,
   quantity INT NOT NULL,
   averageBuyPrice NUMERIC(10,2) NOT NULL,
   currentPrice NUMERIC(10,2) NOT NULL,
   profit NUMERIC(10,2) NOT NULL,
   loss NUMERIC(10,2) NOT NULL,
   profitPercentage NUMERIC(10,2) NOT NULL,
   lossPercentage NUMERIC(10,2) NOT NULL,
   investedAmount NUMERIC(10,2) NOT NULL,
   currentAmount NUMERIC(10,2) NOT NULL,
   FOREIGN KEY (portfolioId) REFERENCES stockPortfolioMGM.portfolio(portfolioId) ON DELETE
CASCADE,
   FOREIGN KEY (stockId) REFERENCES stockPortfolioMGM.stocks(stockId) ON DELETE cascade,
   FOREIGN KEY (exchangeId, stockId) REFERENCES stockPortfolioMGM.stockExchanges(exchangeId,
stockId) ON DELETE CASCADE
);
CREATE INDEX idx_portfolioStocks_portfoliostockId ON stockPortfolioMGM.portfolioStocks USING BTREE
(portfolioStocksId);
```

DDL command for stockPortfolioMGM.watchlist

```
CREATE TABLE stockPortfolioMGM.watchlist (
   watchlistId SERIAL  PRIMARY KEY,
   userId INTEGER NOT NULL ,
   watchlistName VARCHAR(255) NOT NULL,
   description TEXT NOT NULL,
   watchlistCreationTime TIMESTAMP NOT NULL,
   watchlistUpdationTime TIMESTAMP NOT NULL,
   FOREIGN KEY (userId) REFERENCES stockPortfolioMGM.users(userId) ON DELETE CASCADE
 );

CREATE INDEX idx_watchlist_watchlistId ON stockPortfolioMGM.watchlist USING BTREE (watchlistId);
```

DDL command for stockPortfolioMGM.marketDataArchival

```
CREATE TABLE stockPortfolioMGM.marketDataArchival (
   marketArchiveId SERIAL PRIMARY KEY,
   exchangeId INTEGER NOT NULL,
   stockId INTEGER NOT NULL,
   openingPrice NUMERIC(10,2) NOT NULL,
   closingPrice NUMERIC(10,2) NOT NULL,
   dayHighestPrice NUMERIC(10,2) NOT NULL,
   dayLowestPrice NUMERIC(10,2) NOT NULL,
   averageVolume NUMERIC(20,2),
   averageBuyingVolume NUMERIC(20,2) NOT NULL,
   averageSellingVolume NUMERIC(20,2) NOT NULL,
   date DATE NOT NULL,
   FOREIGN KEY (exchangeId, stockId) REFERENCES stockPortfolioMGM.stockExchanges(exchangeId,
stockId) ON DELETE CASCADE,
   FOREIGN KEY (stockId) REFERENCES stockPortfolioMGM.stocks(stockId) ON DELETE CASCADE
);

CREATE INDEX idx_marketDataArchival_marketArchiveId ON stockPortfolioMGM.marketDataArchival
USING BTREE (marketArchiveId);
CREATE INDEX idx_marketDataArchival_stockId ON stockPortfolioMGM.marketDataArchival USING
BTREE (stockId);
```

DDL command for stockPortfolioMGM.orders

```
CREATE TABLE stockportfoliomgm.orders (
   ordersk serial4 not null,
   orderid  INT NOT NULL,
   portfolioid int4 NOT NULL,
   exchangeid int4 NOT NULL,
   stockid int4 NOT NULL,
   ordertype stockportfoliomgm."ordertypeenum" NOT NULL,
   ordertermtype stockportfoliomgm."ordertermtypeenum" NOT NULL,
   orderexchange stockportfoliomgm."orderexchangeenum" NOT NULL,
   marketprice numeric(10, 2) NOT NULL,
   triggerprice numeric(10, 2) NULL,
   stoploss numeric(10, 2) NULL,
   product stockportfoliomgm."productenum" NOT NULL,
   ordermarkettype stockportfoliomgm."ordermarkettypeenum" NOT NULL,
   orderexecutionprice numeric(10, 2) NULL,
   orderplacementdatetime timestamp NOT NULL,
   quantity int4 NOT NULL,
   orderstatus stockportfoliomgm."orderstatusenum" DEFAULT 'OPEN'::stockportfoliomgm.orderstatusenum
NOT NULL,
   CONSTRAINT orders_pkey PRIMARY KEY (ordersk,orderid),
   CONSTRAINT orders_exchangeid_stockid_fkey FOREIGN KEY (exchangeid,stockid) REFERENCES
stockportfoliomgm.stockexchanges(exchangeid,stockid) ON DELETE CASCADE,
   CONSTRAINT orders_portfolioid_fkey FOREIGN KEY (portfolioid) REFERENCES
stockportfoliomgm.portfolio(portfolioid) ON DELETE CASCADE,
   CONSTRAINT orders_stockid_fkey FOREIGN KEY (stockid) REFERENCES
stockportfoliomgm.stocks(stockid) ON DELETE CASCADE
);
CREATE INDEX idx_orders_orderid ON stockportfoliomgm.orders USING btree (orderid);
```

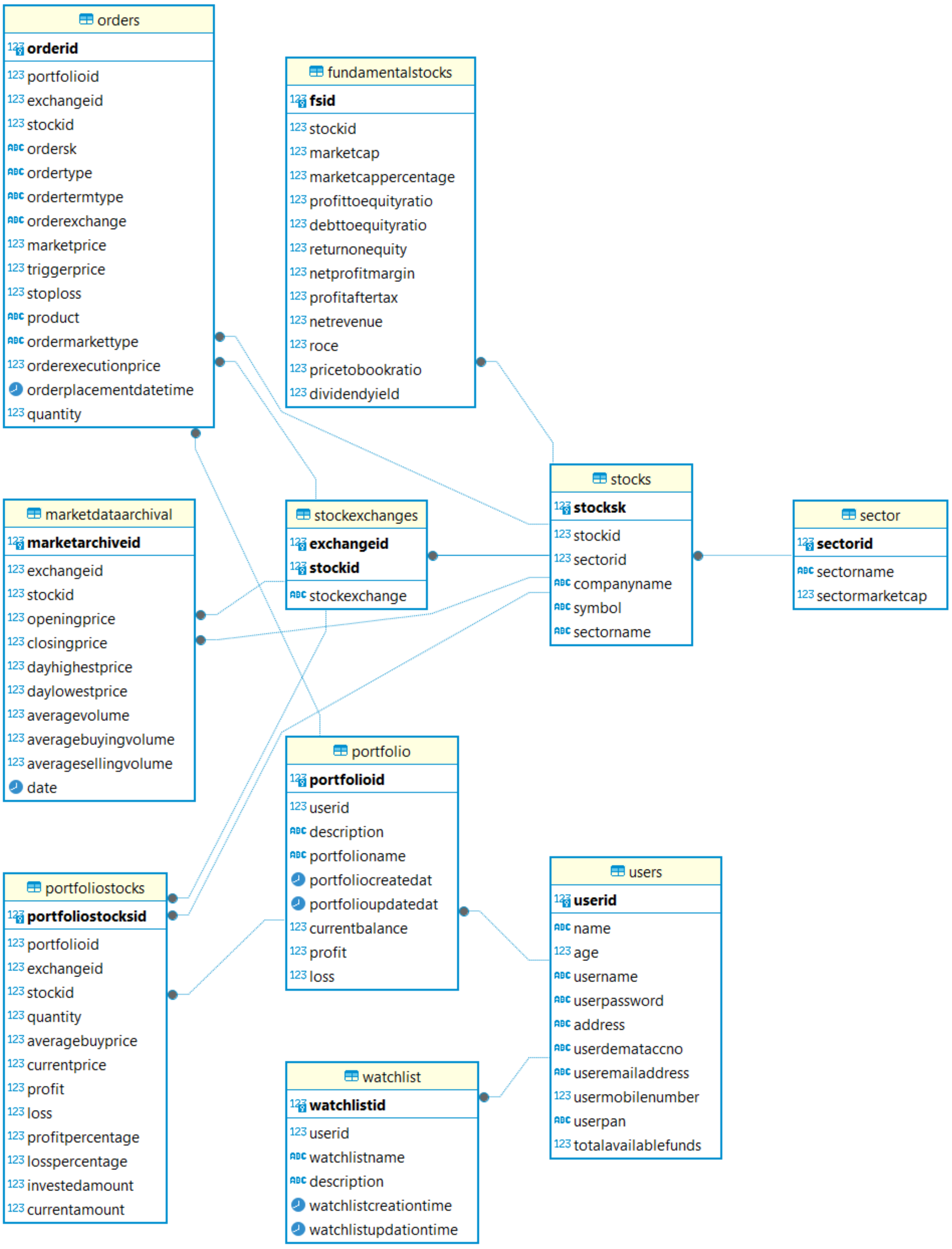
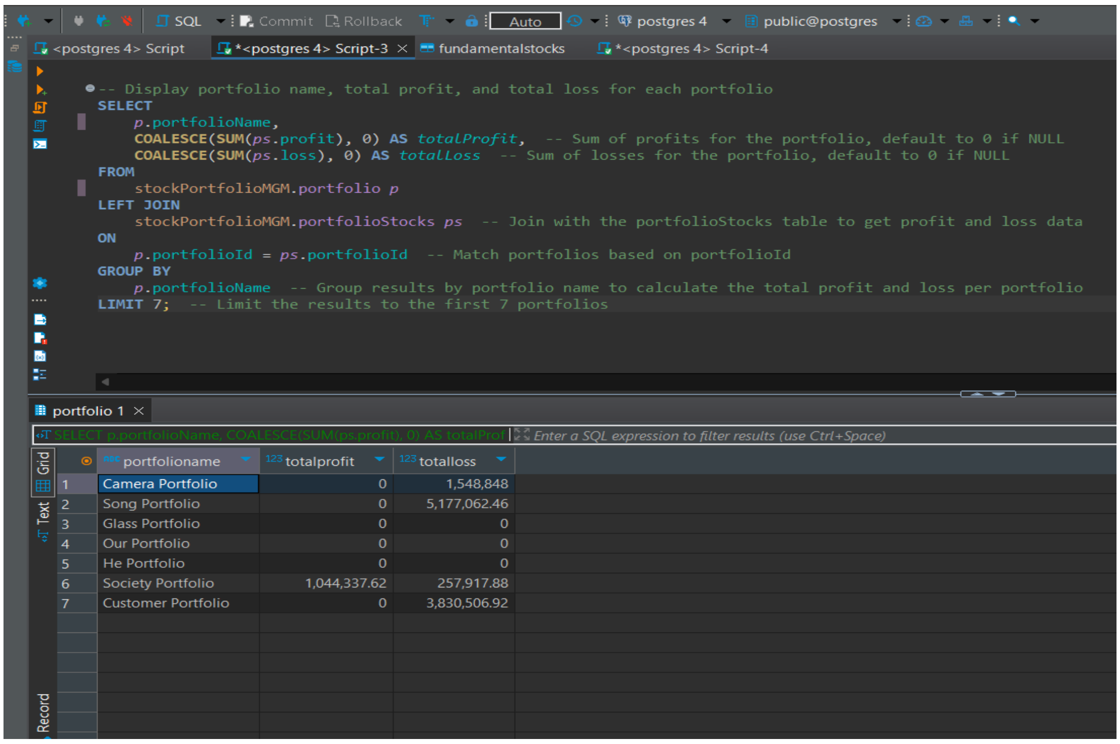


Fig 1.3: Physical Diagram generated by dbeaver (database IDE) after executing DDLstatements

Queries

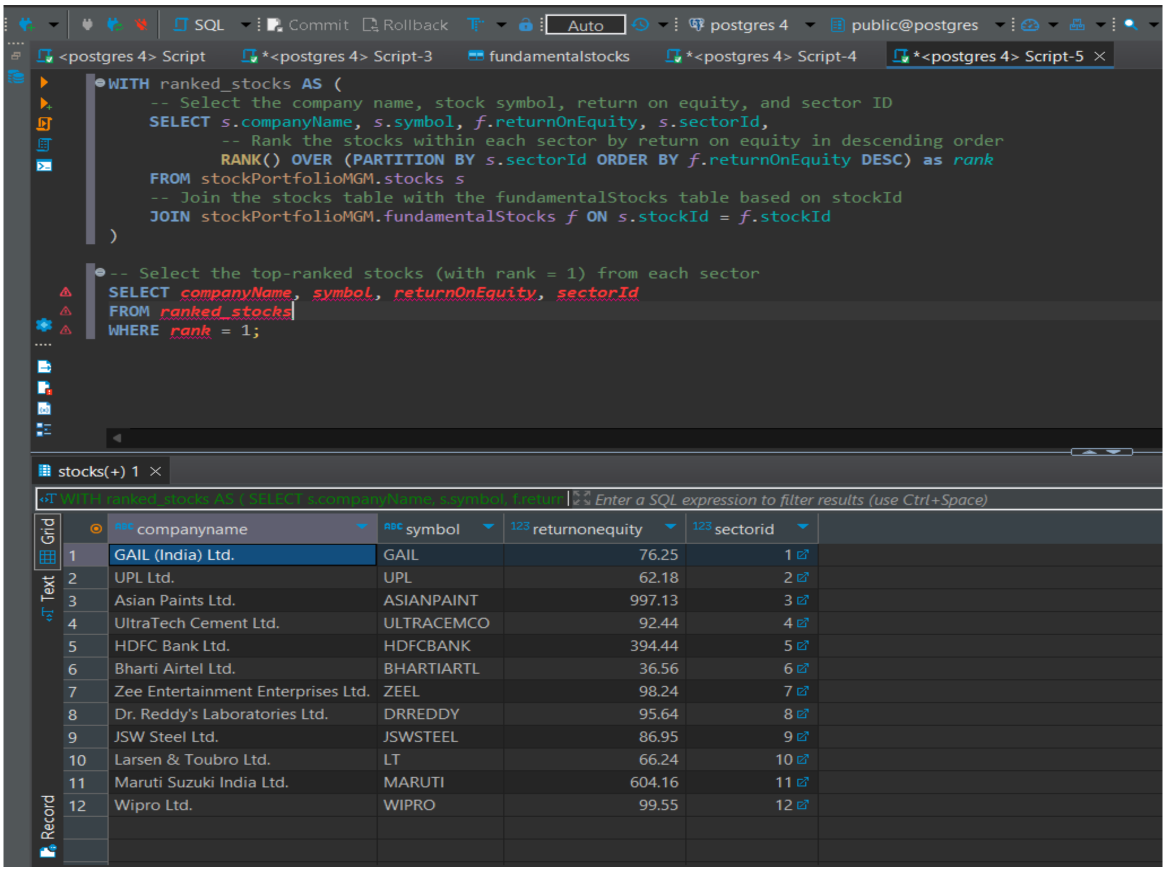
#1. Query to display the portfolio name, total profit, and total loss for each portfolio (top 7 results):



```
-- Display portfolio name, total profit, and total loss for each portfolio
SELECT
    p.portfolioName,
    COALESCE(SUM(ps.profit), 0) AS totalProfit,  -- Sum of profits for the portfolio, default to 0 if NULL
    COALESCE(SUM(ps.loss), 0) AS totalLoss  -- Sum of losses for the portfolio, default to 0 if NULL
FROM
    stockPortfolioMGM.portfolio p
LEFT JOIN
    stockPortfolioMGM.portfolioStocks ps  -- Join with the portfolioStocks table to get profit and loss data
ON
    p.portfolioId = ps.portfolioId  -- Match portfolios based on portfolioId
GROUP BY
    p.portfolioName  -- Group results by portfolio name to calculate the total profit and loss per portfolio
LIMIT 7;  -- Limit the results to the first 7 portfolios
```

	portfolioname	totalprofit	totalloss
1	Camera Portfolio	0	1,548,848
2	Song Portfolio	0	5,177,062.46
3	Glass Portfolio	0	0
4	Our Portfolio	0	0
5	He Portfolio	0	0
6	Society Portfolio	1,044,337.62	257,917.88
7	Customer Portfolio	0	3,830,506.92

#2 Query to find the stocks in each sector with the highest return on equity (ROE)

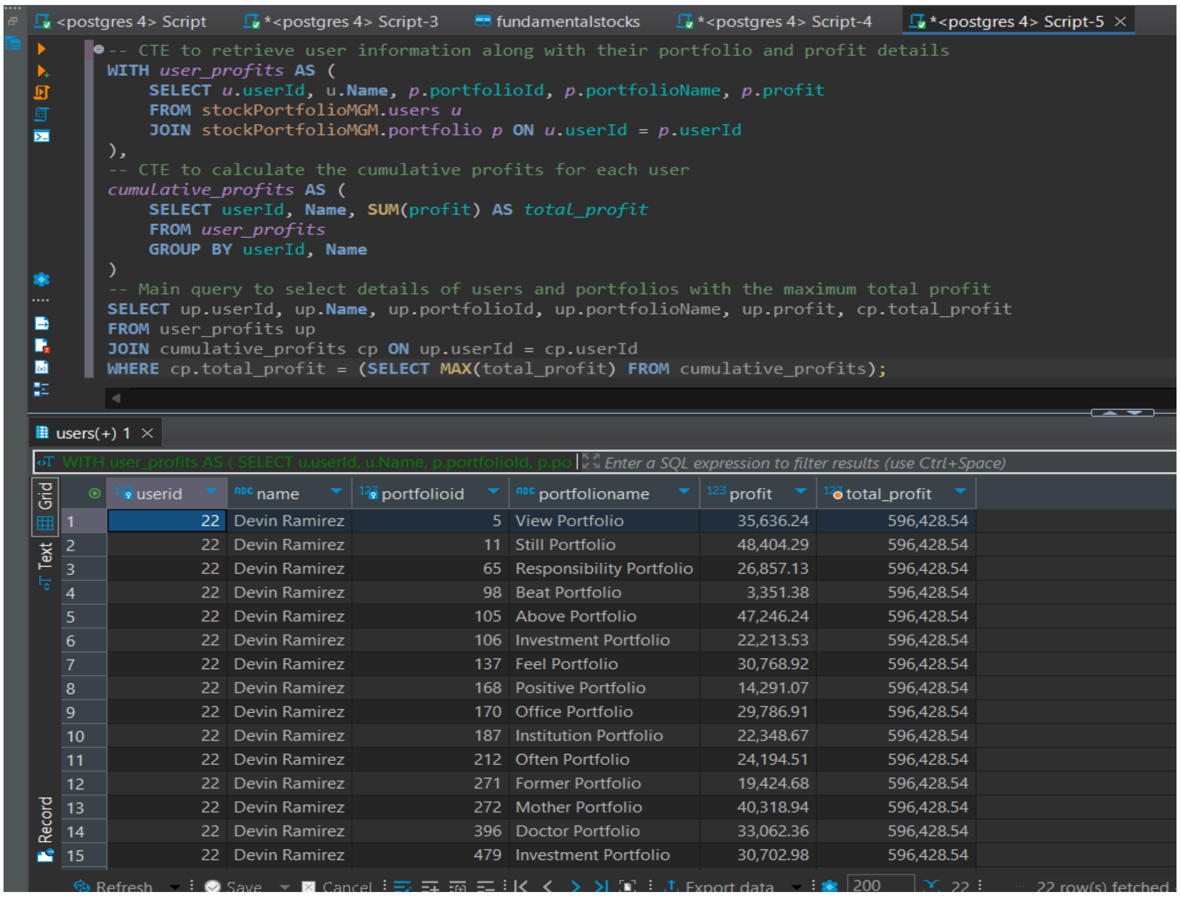


```
WITH ranked_stocks AS (
    -- Select the company name, stock symbol, return on equity, and sector ID
    SELECT s.companyName, s.symbol, f.returnOnEquity, s.sectorId,
           -- Rank the stocks within each sector by return on equity in descending order
           RANK() OVER (PARTITION BY s.sectorId ORDER BY f.returnOnEquity DESC) as rank
    FROM stockPortfolioMGM.stocks s
    -- Join the stocks table with the fundamentalStocks table based on stockId
    JOIN stockPortfolioMGM.fundamentalStocks f ON s.stockId = f.stockId
)

-- Select the top-ranked stocks (with rank = 1) from each sector
SELECT companyName, symbol, returnOnEquity, sectorId
FROM ranked_stocks
WHERE rank = 1;
```

	companyname	symbol	returnonequity	sectorid
1	GAIL (India) Ltd.	GAIL	76.25	1
2	UPL Ltd.	UPL	62.18	2
3	Asian Paints Ltd.	ASIANPAINT	997.13	3
4	UltraTech Cement Ltd.	ULTRACEMCO	92.44	4
5	HDFC Bank Ltd.	HDFCBANK	394.44	5
6	Bharti Airtel Ltd.	BHARTIARTL	36.56	6
7	Zee Entertainment Enterprises Ltd.	ZEEL	98.24	7
8	Dr. Reddy's Laboratories Ltd.	DRREDDY	95.64	8
9	JSW Steel Ltd.	JSWSTEEL	86.95	9
10	Larsen & Toubro Ltd.	LT	66.24	10
11	Maruti Suzuki India Ltd.	MARUTI	604.16	11
12	Wipro Ltd.	WIPRO	99.55	12

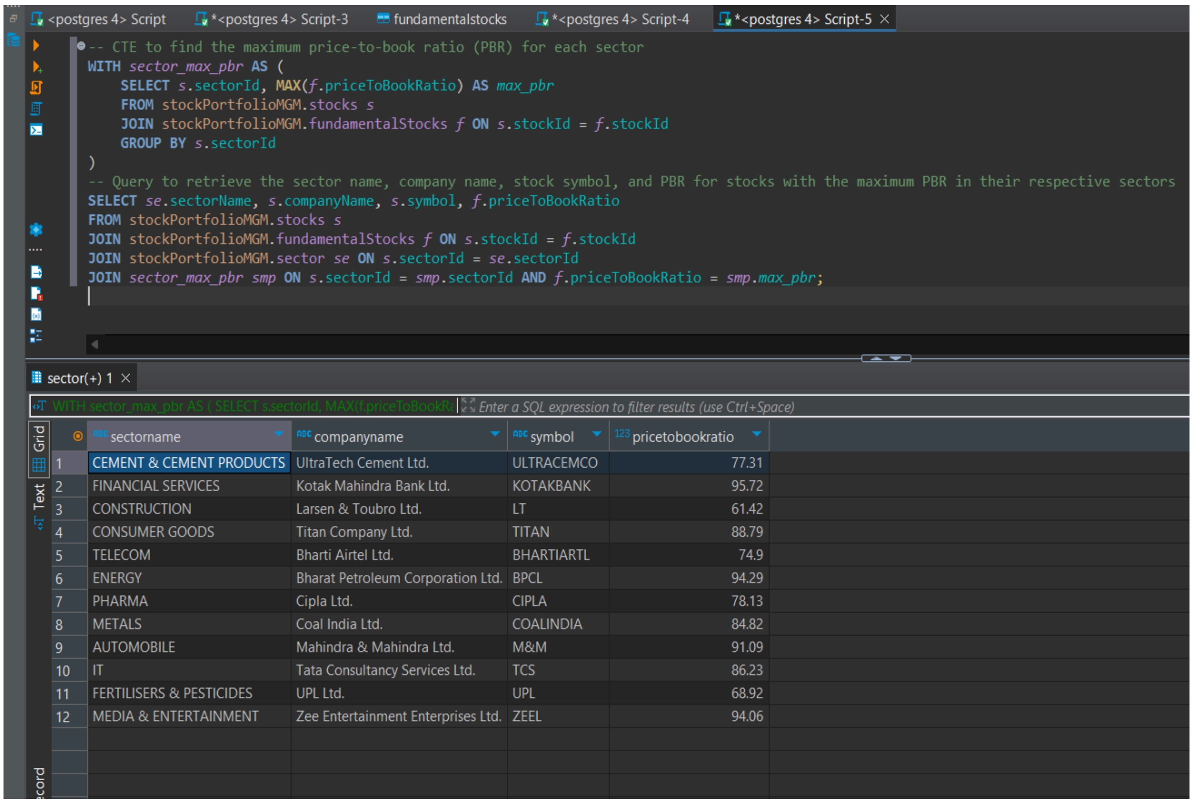
#3. Query to find the user(s) who have the highest cumulative profit across all their portfolios and list their portfolios along with their profit details.



```sql
-- CTE to retrieve user information along with their portfolio and profit details
WITH user_profits AS (
    SELECT u.userId, u.Name, p.portfolioId, p.portfolioName, p.profit
    FROM stockPortfolioMGM.users u
    JOIN stockPortfolioMGM.portfolio p ON u.userId = p.userId
),
-- CTE to calculate the cumulative profits for each user
cumulative_profits AS (
    SELECT userId, Name, SUM(profit) AS total_profit
    FROM user_profits
    GROUP BY userId, Name
)
-- Main query to select details of users and portfolios with the maximum total profit
SELECT up.userId, up.Name, up.portfolioId, up.portfolioName, up.profit, cp.total_profit
FROM user_profits up
JOIN cumulative_profits cp ON up.userId = cp.userId
WHERE cp.total_profit = (SELECT MAX(total_profit) FROM cumulative_profits);
```

	userid	name	portfolioid	portfolioname	profit	total_profit
1	22	Devin Ramirez	5	View Portfolio	35,636.24	596,428.54
2	22	Devin Ramirez	11	Still Portfolio	48,404.29	596,428.54
3	22	Devin Ramirez	65	Responsibility Portfolio	26,857.13	596,428.54
4	22	Devin Ramirez	98	Beat Portfolio	3,351.38	596,428.54
5	22	Devin Ramirez	105	Above Portfolio	47,246.24	596,428.54
6	22	Devin Ramirez	106	Investment Portfolio	22,213.53	596,428.54
7	22	Devin Ramirez	137	Feel Portfolio	30,768.92	596,428.54
8	22	Devin Ramirez	168	Positive Portfolio	14,291.07	596,428.54
9	22	Devin Ramirez	170	Office Portfolio	29,786.91	596,428.54
10	22	Devin Ramirez	187	Institution Portfolio	22,348.67	596,428.54
11	22	Devin Ramirez	212	Often Portfolio	24,194.51	596,428.54
12	22	Devin Ramirez	271	Former Portfolio	19,424.68	596,428.54
13	22	Devin Ramirez	272	Mother Portfolio	40,318.94	596,428.54
14	22	Devin Ramirez	396	Doctor Portfolio	33,062.36	596,428.54
15	22	Devin Ramirez	479	Investment Portfolio	30,702.98	596,428.54

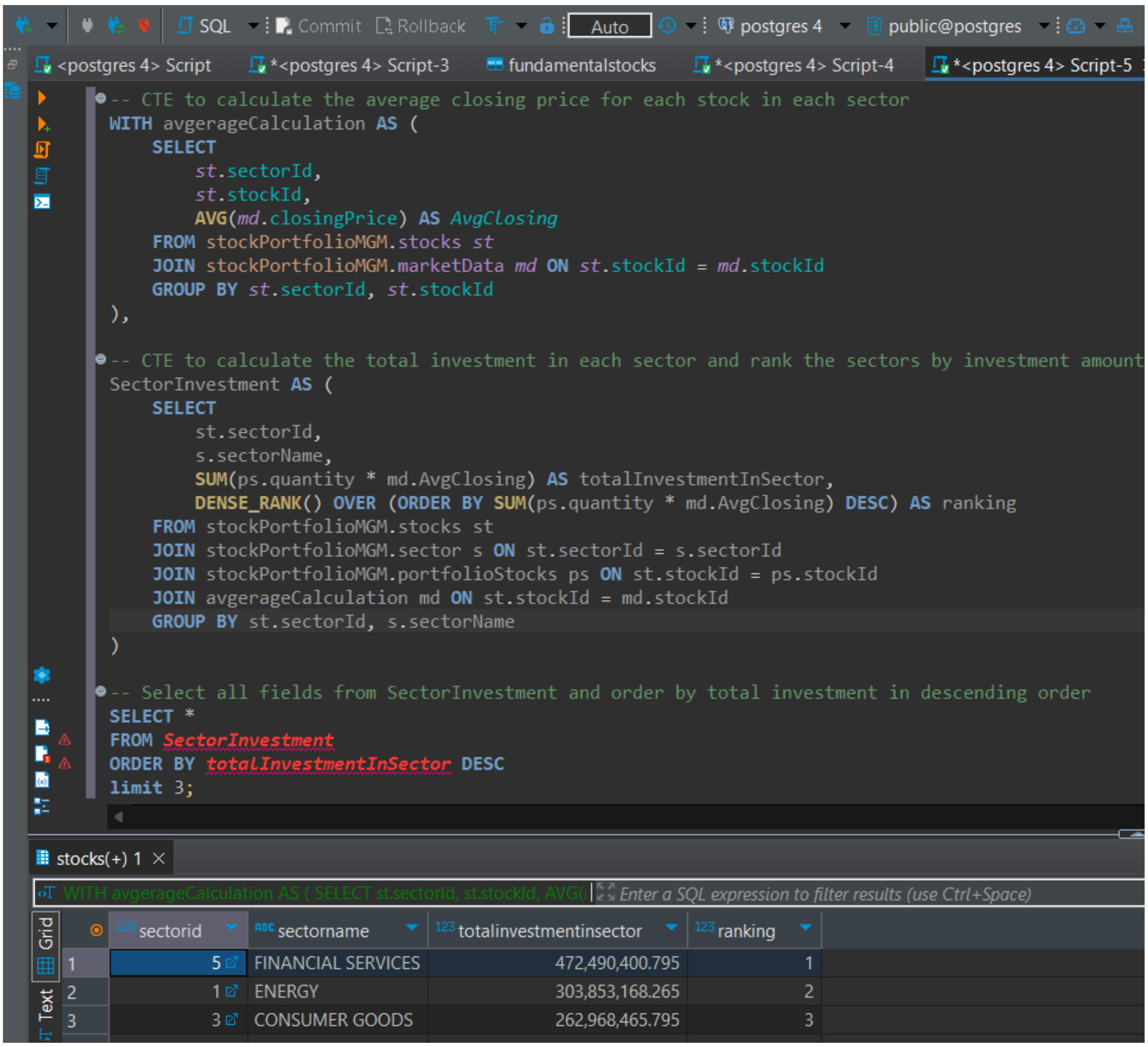
#4. Query to retrieve the stock details (name, symbol) that have the highest price-to-book ratio in each sector.



```sql
-- CTE to find the maximum price-to-book ratio (PBR) for each sector
WITH sector_max_pbr AS (
    SELECT s.sectorId, MAX(f.priceToBookRatio) AS max_pbr
    FROM stockPortfolioMGM.stocks s
    JOIN stockPortfolioMGM.fundamentalStocks f ON s.stockId = f.stockId
    GROUP BY s.sectorId
)
-- Query to retrieve the sector name, company name, stock symbol, and PBR for stocks with the maximum PBR in their respective sectors
SELECT se.sectorName, s.companyName, s.symbol, f.priceToBookRatio
FROM stockPortfolioMGM.stocks s
JOIN stockPortfolioMGM.fundamentalStocks f ON s.stockId = f.stockId
JOIN stockPortfolioMGM.sector se ON s.sectorId = se.sectorId
JOIN sector_max_pbr smp ON s.sectorId = smp.sectorId AND f.priceToBookRatio = smp.max_pbr;
```

	sectorname	companyname	symbol	pricetobookratio
1	CEMENT & CEMENT PRODUCTS	UltraTech Cement Ltd.	ULTRACEMCO	77.31
2	FINANCIAL SERVICES	Kotak Mahindra Bank Ltd.	KOTAKBANK	95.72
3	CONSTRUCTION	Larsen & Toubro Ltd.	LT	61.42
4	CONSUMER GOODS	Titan Company Ltd.	TITAN	88.79
5	TELECOM	Bharti Airtel Ltd.	BHARTIARTL	74.9
6	ENERGY	Bharat Petroleum Corporation Ltd.	BPCL	94.29
7	PHARMA	Cipla Ltd.	CIPLA	78.13
8	METALS	Coal India Ltd.	COALINDIA	84.82
9	AUTOMOBILE	Mahindra & Mahindra Ltd.	M&M	91.09
10	IT	Tata Consultancy Services Ltd.	TCS	86.23
11	FERTILISERS & PESTICIDES	UPL Ltd.	UPL	68.92
12	MEDIA & ENTERTAINMENT	Zee Entertainment Enterprises Ltd.	ZEEL	94.06

#5. Query to determine the top 3 sectors by total investment value, offering insights into investment patterns across sectors.



```sql
-- CTE to calculate the average closing price for each stock in each sector
WITH avgerageCalculation AS (
    SELECT
        st.sectorId,
        st.stockId,
        AVG(md.closingPrice) AS AvgClosing
    FROM stockPortfolioMGM.stocks st
    JOIN stockPortfolioMGM.marketData md ON st.stockId = md.stockId
    GROUP BY st.sectorId, st.stockId
),

-- CTE to calculate the total investment in each sector and rank the sectors by investment amount
SectorInvestment AS (
    SELECT
        st.sectorId,
        s.sectorName,
        SUM(ps.quantity * md.AvgClosing) AS totalInvestmentInSector,
        DENSE_RANK() OVER (ORDER BY SUM(ps.quantity * md.AvgClosing) DESC) AS ranking
    FROM stockPortfolioMGM.stocks st
    JOIN stockPortfolioMGM.sector s ON st.sectorId = s.sectorId
    JOIN stockPortfolioMGM.portfolioStocks ps ON st.stockId = ps.stockId
    JOIN avgerageCalculation md ON st.stockId = md.stockId
    GROUP BY st.sectorId, s.sectorName
)

-- Select all fields from SectorInvestment and order by total investment in descending order
SELECT *
FROM SectorInvestment
ORDER BY totalInvestmentInSector DESC
limit 3;
```

	sectorid	sectorname	totalinvestmentinsector	ranking
1	5	FINANCIAL SERVICES	472,490,400.795	1
2	1	ENERGY	303,853,168.265	2
3	3	CONSUMER GOODS	262,968,465.795	3

.