**UNDERSTANDING THE PROBLEM**

The aim of our project lies in comprehending the problem and defining our objectives. We aim to construct a database for prospective customers who are considering subscribing to fund houses. These fund houses manage various funds, each investing in a multitude of stocks. Our goal is to provide our users with a wealth of data and insights, assisting them in making informed decisions about which funds to invest in and the amount to allocate to each.

To achieve this, we identify various entities and their attributes, and establish relationships among these entities using an Entity-Relationship (ER) Diagram. This diagram serves as a blueprint for our database creation, ensuring a structured and efficient design. By doing so, we strive to empower our users with the necessary tools to navigate the complex world of investment, thereby aligning with our mission of facilitating informed financial decision-making.

* **Identifying The Entities:**

**Fund Houses**: [Fund houses, also known as Asset Management Companies (AMCs), are organizations that pool money from investors and invest it into various financial instruments like equities, mutual funds, and securities1](https://groww.in/p/mutual-fund-house). [These companies make investment based on various market conditions and risks.](https://groww.in/p/mutual-fund-house)

**Funds**: [In the context of fund houses, funds refer to the pooled money from multiple investors that is set aside for investment](https://www.etmoney.com/mutual-funds/fund-houses). [These funds are managed by the fund houses or Asset Management Companies (AMCs) and are invested in various financial instruments like equities, mutual funds, securities, etc](https://www.etmoney.com/mutual-funds/fund-houses).

**Stocks**: [In the context of fund houses and funds, stocks represent the individual shares of a company that are available for investment](https://groww.in/blog/top-stocks-bought-and-sold-by-mutual-funds). [Fund houses, through their various funds, invest in these stocks based on extensive research and analysis](https://groww.in/blog/top-stocks-bought-and-sold-by-mutual-funds). [The fund managers, employed by the fund houses, select stocks to invest in based on market trends, macroeconomic and microeconomic factors, and other relevant information](https://groww.in/blog/top-stocks-bought-and-sold-by-mutual-funds).

* **Identifying the relationships:**

We have 3 different relationships between the entities. Namely:

1. Users SUBSCRIBING to fund houses.
2. Fund houses MANAGING various funds.
3. Funds in turn have various stocks to their name to which they INVEST.
4. Fund house consists of portfolio
5. Investors hold shares in fund house
6. Fund manager supervise funds
7. Funds have risk information
8. Funds have grading in the market
9. Funds have strategies to invest in the market
10. Stock has its details

* **Identifying the Attributes:**

1.Fund Houses:

1. Fund House Name
2. Fund House ID
3. Profit/Loss Ratio
4. Market Rating
5. Category

2.Funds:

1. Fund ID
2. Name Of Fund
3. Fund Size
4. Risk
5. Rating
6. Returns

3.Stocks

1. Name Of Stock
2. Stock ID
3. SName
4. Value
5. Quantity

4.Sector

1. Stock id

2. SName

5.Company

1.Company id

2.Value

3.Name

4.Sname

6.Stock price

1.Stock ID

2.1Yhigh

3.1Ylow

7.Riskmetric

1.Risk metric id

2.fund id

3.metric type

4.value

8.Fundmanager

1.Manager id

2.Name

3.Experience

9.Benchmark

1.Benchmark id

2.Index value

3.Index name

4.date

10.Market

1.Market id

2.Index name

3.fund id

4.Value

11.Fund strategy

1.Strategy id

2.fund id

3.Strategy type

12.Performance metric

1.Performance id

2.fund id

3.Metric type

4.Value

13.Transactions

1.Transaction id

2.date

3.amount

14.Portfolio

1.Portfolio id

2.Invester id

3.fund id

4.Quantity

15.Investors

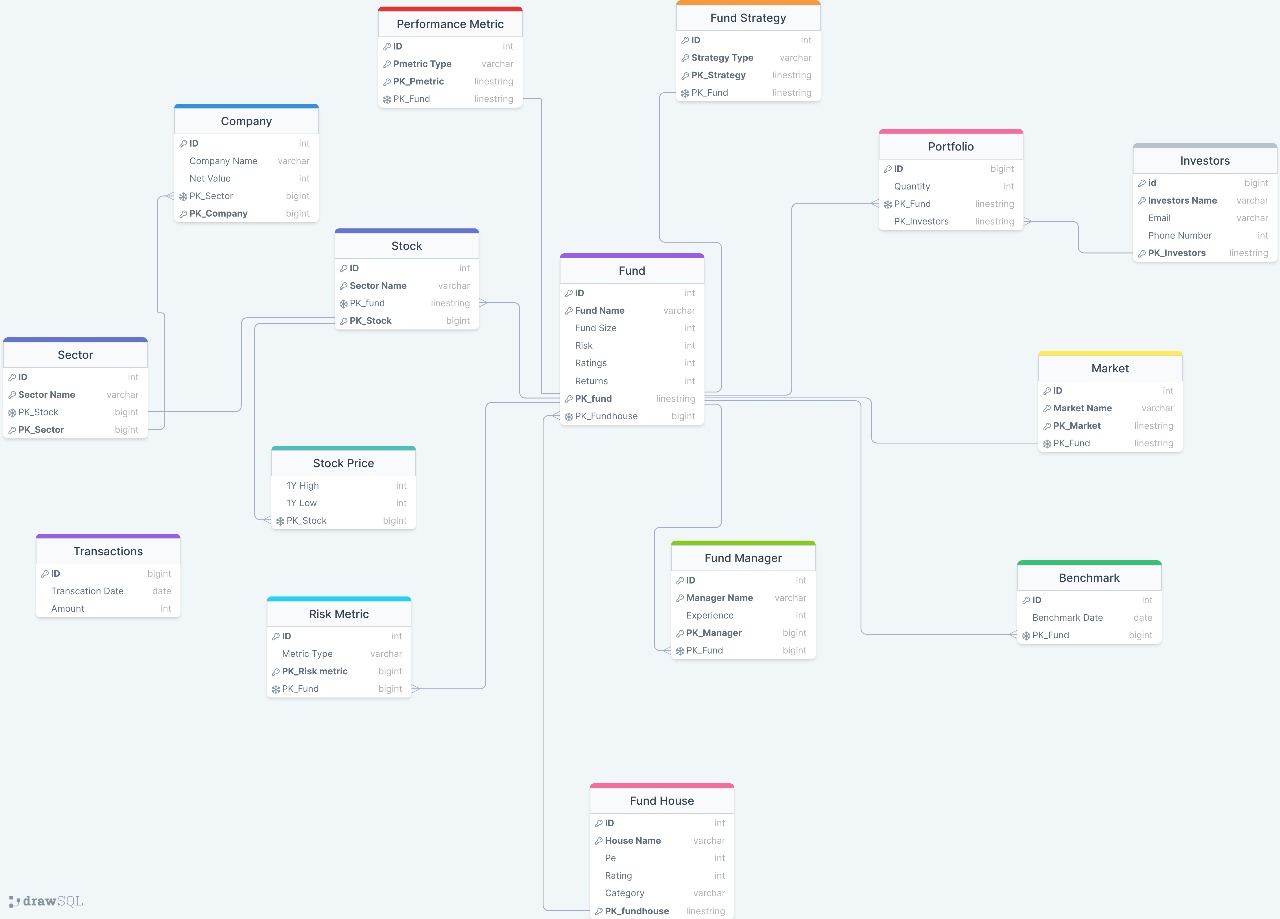
1.Investor id

2.Name

3.Email

4.Phone Number

* **ER DIAGRAM:**

A diagram of a flowchart

Description automatically generated

**CREATE QUERIES:**

CREATE TABLE FUNDHOUSE(

    house\_id INT NOT NULL,

    housename VARCHAR(256),

    pe INT,

    rating INT,

    category VARCHAR(256),

    PK\_fundhouse VARCHAR(100) UNIQUE

    CONSTRAINT FK\_fundhouse PRIMARY KEY (house\_id,housename)

);

-- FUNDS

CREATE TABLE FUND(

    fund\_id INT NOT NULL,

    fund\_name VARCHAR(256),

    fund\_size INT,

    risk INT,

    rating INT,

    returns INT,

    PK\_fund VARCHAR(100) UNIQUE,

    FK\_funhouse VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_fund PRIMARY KEY (fund\_id, fund\_name),

    FOREIGN KEY (FK\_fundhouse) REFERENCES FUNDHOUSE(PK\_fundhouse)

);

-- STOCKS

CREATE TABLE STOCKS(

    stock\_id INT NOT NULL,

    stock\_name VARCHAR(256),

    net\_value INT,

    quantity INT,

    PK\_stock VARCHAR(100) UNIQUE,

    FK\_fund VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_stock PRIMARY KEY (stock\_id, stock\_name),

    FOREIGN KEY (FK\_fund) REFERENCES FUND(PK\_fund)

);

-- SECTOR

CREATE TABLE SECTOR(

    sector\_id INT NOT NULL,

    sector\_name VARCHAR(256),

    PK\_sector VARCHAR(100) UNIQUE,

    FK\_stock VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_sector PRIMARY KEY (sector\_id, sector\_name),

    FOREIGN KEY (FK\_stock) REFERENCES STOCKS(PK\_stock)

);

-- COMPANY

CREATE TABLE COMPANY(

    company\_id INT NOT NULL,

    company\_name VARCHAR(256),

    net\_value INT NOT NULL,

    PK\_company VARCHAR(100) UNIQUE,

    FK\_stock VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_company PRIMARY KEY (company\_id, company\_name),

    FOREIGN KEY (FK\_stock) REFERENCES STOCKS(PK\_stock)

);

-- STOCK\_PRICE

CREATE TABLE STOCK\_PRICE(

    sprice\_high INT NOT NULL,

    low INT NOT NULL,

    FK\_stock VARCHAR(100) UNIQUE,

    FOREIGN KEY (PK\_stock) REFERENCES STOCKS(PK\_stock)

);

-- RISK\_METRIC

CREATE TABLE RISK\_METRIC(

    rmetric\_id INT NOT NULL,

    rmetric\_type VARCHAR(100),

    PK\_riskmetric VARCHAR(100) UNIQUE,

    FK\_fund VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_riskmetric PRIMARY KEY (rmetric\_id, metric\_type),

    FOREIGN KEY (PK\_fund) REFERENCES FUND(PK\_fund)

);

-- FUND\_MANAGER

CREATE TABLE FUND\_MANAGER(

    manager\_id INT NOT NULL,

    manager\_name VARCHAR(256),

    experience INT,

    PK\_manager VARCHAR(100) UNIQUE,

    FK\_fund VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_manager PRIMARY KEY (manager\_id, manager\_name),

    FOREIGN KEY (PK\_fund) REFERENCES FUND(PK\_fund)

);

-- BENCH\_MARK

CREATE TABLE BENCHMARK(

    benchmark\_id INT NOT NULL PRIMARY KEY,

    benchmark\_date DATE,

    FK\_fund VARCHAR(100) UNIQUE,

    FOREIGN KEY (FK\_fund) REFERENCES FUND(PK\_fund)

);

-- MARKET

CREATE TABLE MARKET(

    market\_id INT NOT NULL,

    market\_name VARCHAR(256),

    PK\_market VARCHAR(100) UNIQUE,

    FK\_fund VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_market PRIMARY KEY (market\_id, market\_name),

    FOREIGN KEY (FK\_fund) REFERENCES FUND(PK\_fund)

);

-- FUND\_STRATEGY

CREATE TABLE FUND\_STRATEGY(

    strategy\_id INT NOT NULL,

    strategy\_type VARCHAR(256),

    PK\_strategy VARCHAR(100) UNIQUE,

    FK\_fund VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_strategy PRIMARY KEY (strategy\_id, strategy\_type),

    FOREIGN KEY (FK\_fund) REFERENCES FUND(PK\_fund)

);

-- PERFORMANCE\_METRIC

CREATE TABLE PERFORMANCE\_METRIC(

    pmetric\_id INT UNIQUE NOT NULL,

    pmetric\_type VARCHAR(256),

    PK\_pmetric VARCHAR(100) UNIQUE,

    FK\_fund VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_pmetric PRIMARY KEY (pmetric\_id, pmetric\_type),

    FOREIGN KEY (FK\_fund) REFERENCES FUND(PK\_fund)

);

-- INVESTORS

CREATE TABLE INVESTORS(

    investor\_id INT UNIQUE NOT NULL,

    investor\_name VARCHAR(256),

    email VARCHAR(100),

    phone\_number INT,

    PK\_investor VARCHAR(100) UNIQUE,

    CONSTRAINT PK\_investor PRIMARY KEY (investor\_id, investor\_name)

);

-- PORTFOLIO

CREATE TABLE PORTFOLIO(

    portfolio\_id INT UNIQUE NOT NULL PRIMARY KEY,

    quantity INT,

    FK\_fund VARCHAR(100) UNIQUE,

    FK\_investor VARCHAR(100) UNIQUE,

    FOREIGN KEY (FK\_fund) REFERENCES FUND(PK\_fund),

    FOREIGN KEY (FK\_investor) REFERENCES FUND(PK\_investor)

);

-- TRANSACTIONS

CREATE TABLE TRANSACTIONS(

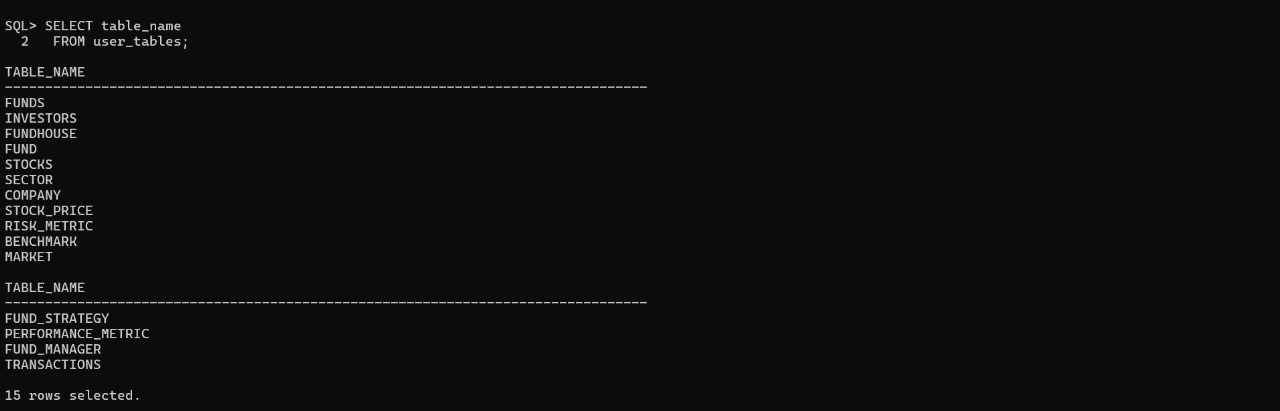
    transaction\_id INT UNIQUE NOT NULL PRIMARY KEY,

    transaction\_date DATE,

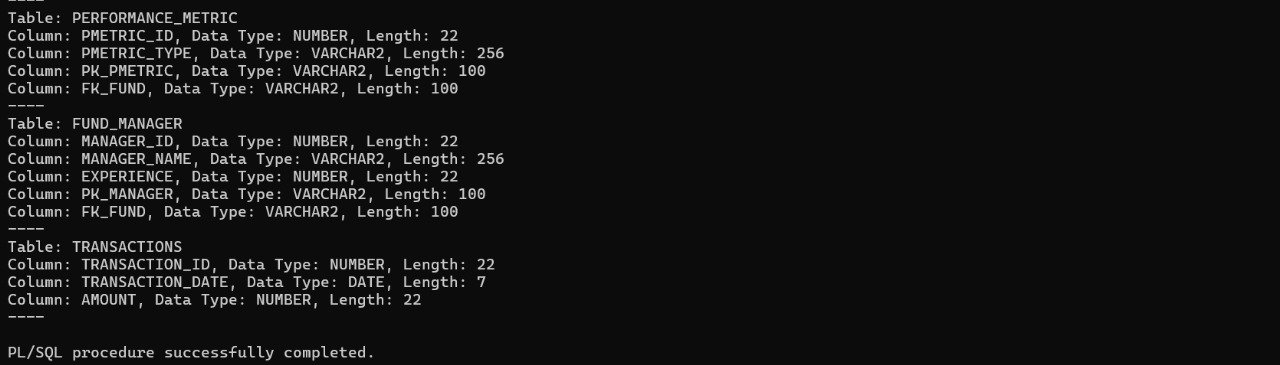
    amount INT

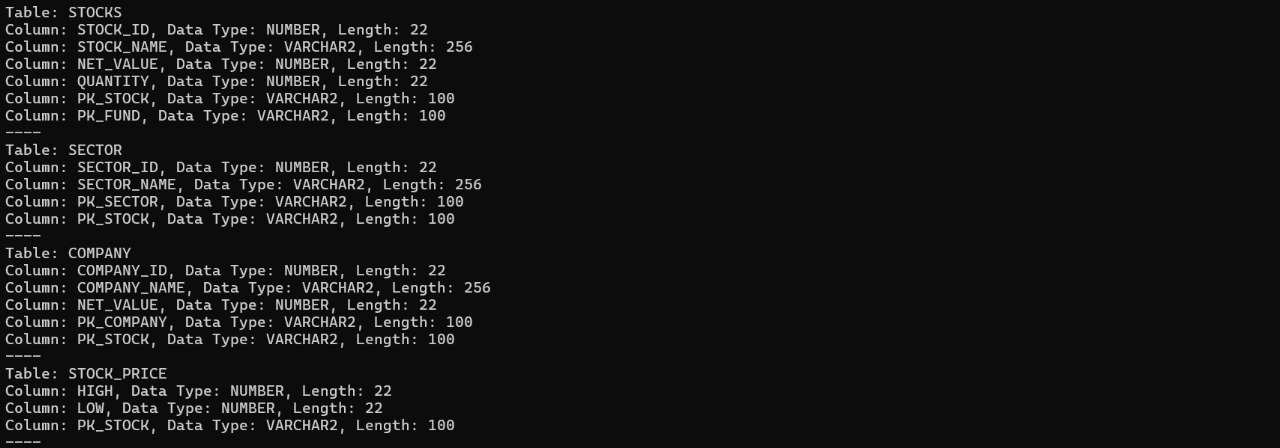
);

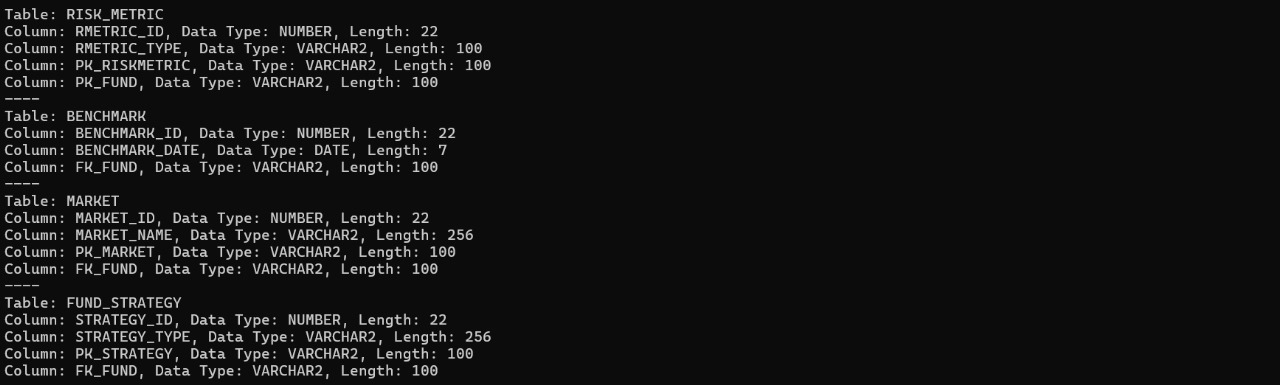
**TABLES:**











**INSERT QUERIES:**

FUNDHOUSE:

INSERT INTO FUNDHOUSE VALUES (1, 'Reliance MF', 20, 4, 'Large Cap', 'R\_MF\_001');

INSERT INTO FUNDHOUSE VALUES (2, 'HDFC AMC', 25, 5, 'Mid Cap', 'HDFC\_AMC\_001');

INSERT INTO FUNDHOUSE VALUES (3, 'ICICI Prudential', 18, 3, 'Small Cap', 'ICICI\_AMC\_001');

FUNDS:

INSERT INTO FUND VALUES (101, 'Reliance LC Fund', 10000000, 3, 4, 12, 'R\_LCF\_001', 'R\_MF\_001');

INSERT INTO FUND VALUES (102, 'HDFC Mid-Cap Fund', 5000000, 4, 5, 15, 'HDFC\_MCOF\_001', 'HDFC\_AMC\_001');

INSERT INTO FUND VALUES (103, 'ICICI Small Cap Fund', 2000000, 5, 3, 18, 'ICICI\_SCF\_001', 'ICICI\_AMC\_001');

STOCKS:

INSERT INTO STOCKS VALUES (201, 'Reliance Ind', 2000, 1000, 'RIL\_001', 'R\_LCF\_001');

INSERT INTO STOCKS VALUES (202, 'Infosys', 3000, 500, 'INFY\_001', 'HDFC\_MCOF\_001');

INSERT INTO STOCKS VALUES (203, 'Tata Motors', 500, 200, 'TML\_001', 'ICICI\_SCF\_001');

SECTOR:

INSERT INTO SECTOR VALUES (301, 'Oil & Gas', 'OilGas\_001', 'RIL\_001');

INSERT INTO SECTOR VALUES (302, 'IT', 'IT\_001', 'INFY\_001');

COMPANY:

INSERT INTO COMPANY VALUES (401, 'Reliance Ind', 2000, 'RIL\_001', 'RIL\_001');

INSERT INTO COMPANY VALUES (402, 'Infosys', 3000, 'INFY\_001', 'INFY\_001');

INSERT INTO COMPANY VALUES (403, 'Tata Motors', 500, 'TML\_001', 'TML\_001');

RISK\_METRIC:

INSERT INTO RISK\_METRIC VALUES (501, 'Std Deviation', 'SD\_001', 'R\_LCF\_001');

INSERT INTO RISK\_METRIC VALUES (502, 'Beta', 'Beta\_001', 'HDFC\_MCOF\_001');

INSERT INTO RISK\_METRIC VALUES (503, 'VaR', 'VaR\_001', 'ICICI\_SCF\_001');

FUND\_MANAGER:

INSERT INTO FUND\_MANAGER VALUES (601, 'John D', 10, 'John\_Doe\_001', 'R\_LCF\_001');

INSERT INTO FUND\_MANAGER VALUES (602, 'Jane S', 15, 'Jane\_Smith\_001', 'HDFC\_MCOF\_001');

INSERT INTO FUND\_MANAGER VALUES (603, 'David S', 8, 'David\_Singh\_001', 'ICICI\_SCF\_001');

BENCHMARK:

INSERT INTO BENCHMARK VALUES (701, '2024-04-01', 'R\_LCF\_001');

INSERT INTO BENCHMARK VALUES (702, '2024-04-01', 'HDFC\_MCOF\_001');

INSERT INTO BENCHMARK VALUES (703, '2024-04-01', 'ICICI\_SCF\_001');

MARKET:

INSERT INTO MARKET VALUES (801, 'BSE', 'BSE\_001', 'R\_LCF\_001');

INSERT INTO MARKET VALUES (802, 'NSE', 'NSE\_001', 'HDFC\_MCOF\_001');

INSERT INTO MARKET VALUES (803, 'NSE', 'NSE\_001', 'ICICI\_SCF\_001');

FUND\_STRATEGY:

INSERT INTO FUND\_STRATEGY VALUES (901, 'Growth', 'Growth\_001', 'R\_LCF\_001');

INSERT INTO FUND\_STRATEGY VALUES (902, 'Value', 'Value\_001', 'HDFC\_MCOF\_001');

INSERT INTO FUND\_STRATEGY VALUES (903, 'Blend', 'Blend\_001', 'ICICI\_SCF\_001');

PERFORMANCE\_METRIC:

INSERT INTO PERFORMANCE\_METRIC VALUES (1001, 'Sharpe Ratio', 'Sharpe\_001', 'R\_LCF\_001');

INSERT INTO PERFORMANCE\_METRIC VALUES (1002, 'Treynor Ratio', 'Treynor\_001', 'HDFC\_MCOF\_001');

INSERT INTO PERFORMANCE\_METRIC VALUES (1003, 'Alpha', 'Alpha\_001', 'ICICI\_SCF\_001');

INVESTORS:

INSERT INTO INVESTORS VALUES (1101, 'Amit K', 'amit@example.com', 9876543210, 'Amit\_001');

INSERT INTO INVESTORS VALUES (1102, 'Rajesh S', 'rajesh@example.com', 9876543211, 'Rajesh\_001');

INSERT INTO INVESTORS VALUES (1103, 'Sunita J', 'sunita@example.com', 9876543212, 'Sunita\_001');

PORTFOLIO:

INSERT INTO PORTFOLIO VALUES (1201, 100, 'R\_LCF\_001', 'Amit\_001');

INSERT INTO PORTFOLIO VALUES (1202, 50, 'HDFC\_MCOF\_001', 'Rajesh\_001');

INSERT INTO PORTFOLIO VALUES (1203, 200, 'ICICI\_SCF\_001', 'Sunita\_001');

TRANSACTIONS:

INSERT INTO TRANSACTIONS VALUES (2001, '2024-04-01', 5000);

INSERT INTO TRANSACTIONS VALUES (2002, '2024-04-02', 7000);

INSERT INTO TRANSACTIONS VALUES (2003, '2024-04-03', 10000);

STOCK\_PRICE:

INSERT INTO STOCK\_PRICE VALUES (1501, 1000, 'RIL\_001');

INSERT INTO STOCK\_PRICE VALUES (1502, 1500, 'INFY\_001');

INSERT INTO STOCK\_PRICE VALUES (1503, 200, 'TML\_001');

**SCREENSHOTS:**

**ALTER QUERIES:**

ALTER TABLE FUNDHOUSE MODIFY (housename VARCHAR(10));

ALTER TABLE FUNDHOUSE MODIFY (category VARCHAR(10)),

ALTER TABLE FUNDHOUSE MODIFY (PK\_fundhouse VARCHAR(10));

ALTER TABLE FUND MODIFY (FK\_fundhouse VARCHAR(10));

ALTER TABLE FUND MODIFY (PK\_fund VARCHAR(10));

ALTER TABLE FUND MODIFY (fund\_name VARCHAR(10));

ALTER TABLE STOCKS MODIFY (stock\_name VARCHAR(10));

ALTER TABLE STOCKS MODIFY (PK\_stock VARCHAR(10));

ALTER TABLE STOCKS MODIFY (FK\_fund VARCHAR(10));

ALTER TABLE SECTOR MODIFY( sector\_name VARCHAR(10);

ALTER TABLE SECTOR MODIFY (PK\_sector VARCHAR(10));

ALTER TABLE SECTOR MODIFY (FK\_stock VARCHAR(10));

ALTER TABLE COMPANY MODIFY(company\_name VARCHAR(10));

ALTER TABLE COMPANY MODIFY (PK\_company VARCHAR(10));

**SCREENSHOTS:**

**CONTRAINT QUERIES:**

Check Constraint for Non-Empty Company Names:

ALTER TABLE COMPANY

ADD CONSTRAINT chk\_non\_empty\_company\_name

CHECK (LENGTH(company\_name) > 0);

Check Constraint for Positive Net Value (assuming net\_value should be positive):

ALTER TABLE COMPANY

ADD CONSTRAINT chk\_positive\_net\_value

CHECK (net\_value > 0);

**SCREENSHOTS:**

**SQL FUNCTIONS QUERIES:**

Scaler function:

SELECT TO\_CHAR(transaction\_date, 'YYYY-MM-DD') FROM

TRANSACTIONS;

SELECT length(fund\_name) from FUND;

GROUP FUNCTION:

SELECT COUNT(\*) FROM FUNDHOUSE;

**SCREENSHOTS:**

**SUBQUERY:**

To find amount of money investor has invested:

SELECT SUM(amount) AS total\_investment

FROM TRANSACTIONS

WHERE transaction\_id IN (

SELECT transaction\_id

FROM PORTFOLIO

WHERE FK\_investor = (SELECT PK\_investor FROM INVESTORS WHERE investor\_name = 'InvestorName')

);

To find sum of portfolios of an investor:

SELECT SUM(quantity) AS total\_portfolio

FROM PORTFOLIO

WHERE FK\_investor = (SELECT PK\_investor FROM INVESTORS WHERE investor\_name = 'InvestorName');

**SCREENSHOTS:**

**JOINS:**

Intersect to find common investors in 2 funds:

SELECT SUM(p.quantity \* s.net\_value) AS portfolio\_value

FROM PORTFOLIO p

JOIN STOCKS s ON p.FK\_fund = s.PK\_stock

WHERE p.FK\_investor = (SELECT PK\_investor FROM INVESTORS WHERE investor\_name = 'InvestorName');

Intersect to find companies in same sector:

SELECT company\_id, company\_name

FROM COMPANY

WHERE FK\_sector = (SELECT PK\_sector FROM SECTOR WHERE sector\_name = 'Sector X')

INTERSECT

SELECT company\_id, company\_name

FROM COMPANY

WHERE FK\_sector = (SELECT PK\_sector FROM SECTOR WHERE sector\_name = 'Sector Y');

Minus to find exclusive investors in fund a not fund b:

SELECT investor\_id, investor\_name

FROM PORTFOLIO

WHERE FK\_fund = (SELECT PK\_fund FROM FUND WHERE fund\_name = 'Fund A')

MINUS

SELECT investor\_id, investor\_name

FROM PORTFOLIO

WHERE FK\_fund = (SELECT PK\_fund FROM FUND WHERE fund\_name = 'Fund B');

Join To find the sum of investors portfolio:

SELECT SUM(p.quantity \* s.net\_value) AS portfolio\_value

FROM PORTFOLIO p

JOIN STOCKS s ON p.FK\_fund = s.PK\_stock

WHERE p.FK\_investor = (SELECT PK\_investor FROM INVESTORS WHERE investor\_name = 'InvestorName');

Join to find investor transaction history:

SELECT t.transaction\_id, t.transaction\_date, t.amount

FROM TRANSACTIONS t

JOIN PORTFOLIO p ON t.transaction\_id = p.transaction\_id

WHERE p.FK\_investor = (SELECT PK\_investor FROM INVESTORS WHERE investor\_name = 'InvestorName');

Join to find fund vs benchmark performance:

SELECT f.fund\_name, b.benchmark\_date, f.returns, b.benchmark\_value

FROM FUND f

JOIN BENCHMARK b ON f.PK\_fund = b.FK\_fund

WHERE f.rating >= 4

ORDER BY f.fund\_name, b.benchmark\_date;

Join to find top performing funds based on rating/returns:

SELECT f.fund\_name, fh.housename, f.rating, f.returns

FROM FUND f

JOIN FUNDHOUSE fh ON f.FK\_fundhouse = fh.PK\_fundhouse

WHERE f.rating >= 4

ORDER BY f.returns DESC;

**SCREENSHOTS:**

**VIEWS QUERIES:**

View for top performing funds:

CREATE VIEW TopPerformingFunds AS

SELECT fund\_name, returns, rating

FROM FUND

WHERE rating >= 4

ORDER BY returns DESC;

View for Investor portfolio:

CREATE VIEW InvestorPortfolioView AS

SELECT i.investor\_id, i.investor\_name, p.quantity, f.fund\_name

FROM INVESTORS i

JOIN PORTFOLIO p ON i.PK\_investor = p.FK\_investor

JOIN FUND f ON p.FK\_fund = f.PK\_fund;

View for fundmanagers and their associated funds:

CREATE VIEW FundManagerView AS

SELECT fm.manager\_id, fm.manager\_name, fm.experience, f.fund\_name, fh.housename

FROM FUND\_MANAGER fm

JOIN FUND f ON fm.FK\_fund = f.PK\_fund

JOIN FUNDHOUSE fh ON f.FK\_fundhouse = fh.PK\_fundhouse;

**SCREENSHOTS:**

**TRIGGERS QUERIES:**

Trigger to update portfolio:

CREATE OR REPLACE TRIGGER UpdatePortfolioQuantity

AFTER INSERT ON TRANSACTIONS

FOR EACH ROW

BEGIN

UPDATE PORTFOLIO

SET quantity = quantity + :NEW.amount

WHERE portfolio\_id = :NEW.portfolio\_id;

END;

/

Trigger to log investor activities:

CREATE OR REPLACE TRIGGER LogInvestorActivity

AFTER INSERT ON INVESTORS

FOR EACH ROW

BEGIN

INSERT INTO INVESTOR\_ACTIVITY\_LOG (activity\_type, activity\_date)

VALUES ('New Investor Added', SYSDATE);

END;

/

Trigger to fundsize:

CREATE OR REPLACE TRIGGER UpdateFundSize

AFTER INSERT ON TRANSACTIONS

FOR EACH ROW

BEGIN

UPDATE FUND

SET fund\_size = fund\_size + :NEW.amount

WHERE fund\_id = :NEW.fund\_id;

END;

/CREATE OR REPLACE TRIGGER UpdateFundSize

AFTER INSERT ON TRANSACTIONS

FOR EACH ROW

BEGIN

UPDATE FUND

SET fund\_size = fund\_size + :NEW.amount

WHERE fund\_id = :NEW.fund\_id;

END;

/

**SCREENSHOTS:**

**CURSORS:**

Cursor to retrieve portfolio details:

DECLARE

v\_portfolio\_id PORTFOLIO.portfolio\_id%TYPE;

v\_quantity PORTFOLIO.quantity%TYPE;

CURSOR portfolio\_cursor IS

SELECT portfolio\_id, quantity

FROM PORTFOLIO;

BEGIN

OPEN portfolio\_cursor;

LOOP

FETCH portfolio\_cursor INTO v\_portfolio\_id, v\_quantity;

EXIT WHEN portfolio\_cursor%NOTFOUND

DBMS\_OUTPUT.PUT\_LINE('Portfolio ID: ' || v\_portfolio\_id || ', Quantity: ' || v\_quantity);

END LOOP;

CLOSE portfolio\_cursor;

END;

/

Cursor to calc total net val of stocks:

DECLARE

v\_total\_net\_value NUMBER := 0;

CURSOR stock\_net\_value\_cursor IS

SELECT SUM(net\_value) AS total\_net\_value

FROM STOCKS;

BEGIN

OPEN stock\_net\_value\_cursor;

FETCH stock\_net\_value\_cursor INTO v\_total\_net\_value;

DBMS\_OUTPUT.PUT\_LINE('Total Net Value of Stocks: ' || v\_total\_net\_value);

CLOSE stock\_net\_value\_cursor;

END;

/

Cursor to retriev investor names and email:

DECLARE

v\_investor\_name INVESTORS.investor\_name%TYPE;

v\_email INVESTORS.email%TYPE;

CURSOR investor\_cursor IS

SELECT investor\_name, email

FROM INVESTORS;

BEGIN

OPEN investor\_cursor;

LOOP

FETCH investor\_cursor INTO v\_investor\_name, v\_email;

EXIT WHEN investor\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Investor: ' || v\_investor\_name || ', Email: ' || v\_email);

END LOOP;

CLOSE investor\_cursor;

END;

/

FUNDHOUSE:

A screenshot of a computer

Description automatically generated

* Normalization Forms:
* 1NF (Yes): Single value per cell, unique column names. Meets 1NF requirements.
* 2NF (Yes): No non-key attributes depend on a part of the primary key. Meets 2NF.
* 3NF (Potential Violation): The rating attribute might depend on both the house\_id and other house-specific factors. This could violate 3NF. Option 1: Introduce a separate FUND\_RATING table.
* BCNF (Same as 3NF): BCNF is a stricter form of 3NF, so considerations are the same.

A screenshot of a black and white screen

Description automatically generated

* Functional Dependencies:
* house\_id -> {housename, pe, rating, category} (Full dependency)
* PK\_fundhouse -> {house\_id, housename} (Primary key definition)

STOCKS:

A screenshot of a black and white screen

Description automatically generated

* Normalization Forms:
* 1NF (Yes): Single value per cell, unique column names. Meets 1NF requirements.
* 2NF (Yes): The FK\_fund now references the single-column fund\_id in the FUND table, addressing the composite key issue and achieving 2NF.
* 3NF (Yes): No further dependencies to consider. Meets 3NF.
* BCNF (Yes): BCNF also met as there are no determinant dependencies.
* Functional Dependencies:
* stock\_id -> {stock\_name, net\_value, quantity, FK\_fund} (Full dependency)
* PK\_stock -> {stock\_id, stock\_name} (Primary key definition)

SECTOR:

A black and white screen with white text

Description automatically generated

* Normalization Forms:
* 1NF (Yes): Single value per cell, unique column names. Meets 1NF requirements.
* 2NF (Yes): No non-key attributes depend on a part of the primary key. Meets 2NF.
* 3NF (Yes): No further dependencies to consider. Meets 3NF.
* BCNF (Yes): BCNF also met as there are no determinant dependencies.
* Functional Dependencies:
* sector\_id -> {sector\_name, FK\_stock} (Full dependency)
* PK\_sector -> {sector\_id, sector\_name} (Primary key definition)

COMPANY:

A screenshot of a computer

Description automatically generated

* Normalization Forms:
* 1NF (Yes): Single value per cell, unique column names. Meets 1NF requirements.
* 2NF (Yes): No non-key attributes depend on a part of the primary key. Meets 2NF.
* 3NF (Yes): No further dependencies to consider. Meets 3NF.
* BCNF (Yes): BCNF also met as there are no determinant dependencies.
* Functional Dependencies:
* company\_id -> {company\_name, net\_value, FK\_stock} (Full dependency)
* PK\_company -> {company\_id, company\_name} (Primary key definition)

RISK METRIC:

A screenshot of a computer

Description automatically generated

* Normalization Forms:
* 1NF (Yes): The table adheres to 1NF as it has single values per cell and unique column names.
* 2NF (Yes): No non-key attribute depends on a part of the primary key. Both rmetric\_type and FK\_fund are fully dependent on the combined primary key (rmetric\_id, rmetric\_type).
* 3NF (Yes): There are no further dependencies to consider. All non-key attributes are fully dependent on the primary key, satisfying 3NF requirements.
* BCNF (Yes): BCNF is a stricter form of 3NF, and since there are no determinant dependencies beyond the primary key, the table is also in BCNF.
* Functional Dependencies:
* rmetric\_type (identifies the type of risk metric)
* FK\_fund (establishes the relationship with a specific fund)
* PK\_riskmetric -> (rmetric\_id, rmetric\_type) (uniquely identifies a risk metric record)