Competitor Benchmarking Database

Introduction

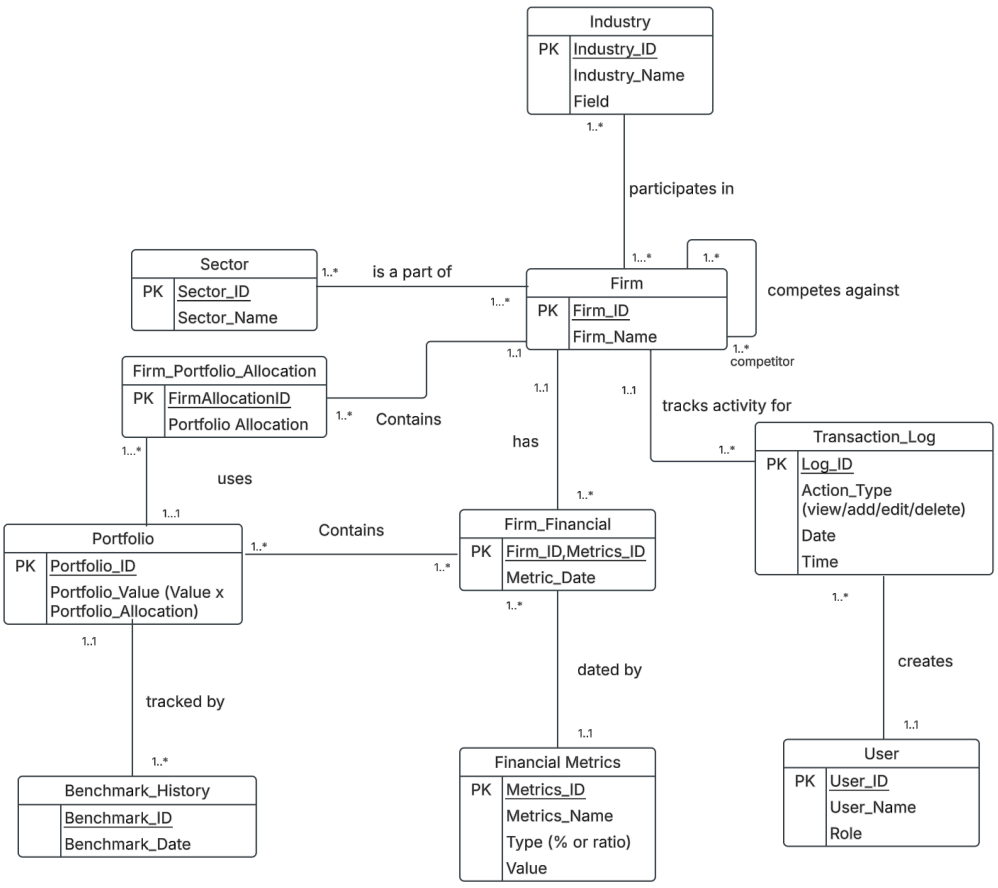
Firms want to know how well they track their competitors over time, but it requires sifting through years of financial data. Similar industry portfolios may include too many firms that don't quite fit a particular firm's market, while a firm may be a part of many industries that require separate calculations to arrive at a curated portfolio that suits their business model.

To solve this, our team built a SQL-based relational database that gives firms more control over defining their competitors. Instead of being locked into pre-defined industry groupings, firms can create custom competitor portfolios based on who they see as peers. From there, they can generate and track financial benchmarks relevant to their business.

Our schema includes tables for firms, industries, sectors, portfolios, financial metrics, and competitor relationships. Detailed financial information on economic data, like revenue, net income, and growth rates, is stored across different portfolios and periods. This application allows a firm to select its competitors to form a unique portfolio, and the database will generate financial metrics unique to its competitors' portfolios. Since a firm's financials can be tied to specific competitor groups in the database, performance comparisons are much faster. Additionally, a user account system and a transaction log were included to record who made what changes and when, helping with overall transparency and data integrity in a collaborative team setting.

Primary users of this database would include equity research analysts, corporate strategy teams, portfolio managers, and consultants. Overall, anyone needing dynamic, tailored benchmarking of financial performance across a custom-defined set of competitors, rather than relying on static industry classifications.

Entity Relationship Model Diagram



Relationship Sentences

1. One firm must be a part of one (or many) industries
2. One industry must contain one (or many) firms
3. One firm must be a part of one (or many) sectors
4. One sector must contain one (or many) firms
5. One firm must compete against one (or many) competitor firms
6. One competitor firm must be a competitor to one (or many) firms
7. One firm must have one (or many) firm financials
8. One firm financial must belong to one firm

9. One firm financial must belong to one (or many) portfolios
10. One portfolio must contain one (or many) firm financials
11. One portfolio must be tracked by one benchmark history
12. One benchmark history must track one portfolio
13. One firm financial must date one financial metric
14. One financial metric must be dated by one or many firm financial
15. One firm must have tracked activity by one (or many) transaction logs
16. One transaction log must track activity for one firm
17. One transaction log must be created by one user
18. One user must create one (or many) transaction logs
19. One firm must contain one or many portfolio allocations
20. One firm portfolio allocation must belong to one firm
21. One portfolio value must be calculated using one firm portfolio allocation
22. One firm portfolio allocation must be used to calculate one portfolio
23. One portfolio uses one to many portfolio allocations

RDM

Industry(Industry_ID, Industry_Name, Field)
Firm(Firm_ID, Firm_Name, Parent_Firm_ID(fk))
Competitor_Firm(Firm_ID(fk), Competitor_ID(fk))
Firm_Industry(Firm_ID, Industry_ID)
Sector(Sector_ID, Sector_Name)
Firm_Sector(Firm_ID, Sector_ID)
Portfolio(Portfolio_ID, Portfolio_Value)
Firm_Portfolio_Allocation(FirmAllocationID, Portfolio Allocation, Firm_ID(fk), Portfolio_ID(fk))
Benchmark_History(Benchmark_ID, Benchmark_Date, Portfolio_ID(fk))
Financial_Metrics(Metrics_ID, Metrics_Name, Type, Value)
Firm_Financial(Firm_ID(fk), Metric_ID(fk), Metric_Date)
Firm_Financial_Portfolio(Portfolio_ID, Firm_ID(fk), Metric_ID(fk))
User(User_ID, User_Name, Role)
Transaction_Log(Log_ID, Action_Type, Date, Time, Firm_ID(fk), User_ID(fk))

Normalization

The steps to follow for each relation are:

1. Write out the relation including all attribute names. Indicate keys and foreign keys.
2. State the Key for the relation and write down all Functional Dependencies.
3. Go through the definitions of each normal form starting with 1NF and going up to BCNF.
4. If a relation meets the definition of a normal form, move up to the next higher normal form.
5. If a relation fails to meet the definition of a normal form (e.g., it contains a partial-key dependency or it contains a transitive dependency), then split the relation into two new relations.
6. Begin the normalization process from the beginning with each of these two new relations.

Normalization is an essential part of designing a relational database because it ensures the structure of the data is logical, efficient, and free of common anomalies. More specifically, it eliminates data redundancies, prevents update anomalies, ensures data integrity, and simplifies queries and maintenance. Below are the steps used to achieve the normal forms:

Decomposition: We had two incidences where we needed to use decomposition. In the table: Financial_Metrics(Metrics_ID, Metrics_Name, Type, Value), we have transitive dependency Metrics_ID → Metrics_Name → Type, and this was corrected using decomposition into two relationships: R1(Metrics_ID, Metrics_Name, Value), R2(Metrics_Name, Type)

The other incidence involved the table: Industry(Industry_ID, Industry_Name, Field), which may be transitive dependency because Industry_ID → Industry_Name → Field. Industry Name may determine field. To correct this we used decomposed into two relationships: R1(Industry_ID, Industry_Name), R2(Industry_Name, Field)

Industry(<u>Industry_ID</u>, Industry_Name, Field)
Step 1: Primary Key? Yes Industry_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? Yes Industry_Name -> Field
Step 4. Does non-key determine part of Key? No BCNF
Relations:

R1(Industry_ID, Industry_Name)
R2(Industry_Name, Field)

Firm(Firm_ID, Firm_Name, Parent_Firm_ID(fk))
Step 1: Primary Key? Yes Firm_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations:R1(Firm_ID, Firm_Name, Parent_Firm_ID)

Competitor_Firm(Firm_ID(fk), Competitor_ID(fk))
Step 1: Primary Key? Yes Firm_ID, Competitor_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1(Firm_ID, Competitor_ID)

Firm_Industry(Firm_ID, Industry_ID)
Step 1: Primary Key? Yes Firm_ID, Industry_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1(Firm_ID, Industry_ID)

Sector(Sector_ID, Sector_Name)
Step 1: Primary Key? Sector_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1(Sector_ID, Sector_Name)

Portfolio(Portfolio_ID, Portfolio_Value)
Step 1: Primary Key? Yes Portfolio_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No

Step 4. Does non-key determine part of Key? No BCNF
Relations: R1 (Portfolio_ID, Portfolio_Value)

Firm_Portfolio_Allocation(FirmAllocationID, Portfolio Allocation, Firm_ID(fk), Portfolio_ID(fk))
Step 1: Primary Key? Yes FirmAllocationID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1(FirmAllocationID, Portfolio_Allocation, Firm_ID, Portfolio_ID)

Benchmark_History(Benchmark_ID, Benchmark_Date, Portfolio_ID(fk))
Step 1: Primary Key? Yes Benchmark_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1 (Benchmark_ID, Benchmark_Date, Portfolio_ID)

Financial_Metrics(Metrics_ID, Metrics_Name, Type, Value)
Step 1: Primary Key? Yes Metrics_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? Yes Metrics_Name -> Type
Step 4. Does non-key determine part of Key? No BCNF
Relations:
R1(Metrics_ID, Metrics_Name, Value)
R2(Metrics_Name, Type)

Firm_Financial(Firm_ID(fk), Metric_ID(fk), Metric_Date)
Step 1: Primary Key? Yes (Firm_ID, Metric_ID, Metric_Date)
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1(Firm_ID, Metric_ID, Metric_Date)

Firm_Financial_Portfolio(Portfolio_ID, Firm_ID(fk), Metric_ID(fk))
Step 1: Primary Key? (Portfolio_ID, Firm_ID, Metric_ID)
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No

Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1(Portfolio_ID, Firm_ID, Metric_ID)

User(<u>User_ID</u>, User_Name, Role)
Step 1: Primary Key? Yes User_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1(User_ID, User_Name, Role)

Transaction_Log(<u>Log_ID</u>, Action_Type, Date, Time, Firm_ID(fk), User_ID(fk))
Step 1: Primary Key? Yes Log_ID
Step 2: Partial key dependencies? Part of the key determines a non-key attribute: No
Step 3: Do we have transitive dependency problems? Non-key attributes determine-non key attributes? No assuming no User_ID -> Firm_ID
Step 4. Does non-key determine part of Key? No BCNF
Relations: R1 (Log_ID, Action_Type, Date, Time, Firm_ID, User_ID)

SQL DDL

```
-- Table: Industry (Normalized from R1(Industry_ID, Industry_Name))
CREATE TABLE Industry (
   Industry_ID INT NOT NULL,
   Industry_Name VARCHAR(255) NOT NULL
);

-- Table: Industry_Field (Normalized from R2(Industry_Name, Field))
-- This table stores the field associated with an industry name.
-- An industry can have multiple fields.
CREATE TABLE Industry_Field (
   Industry_Name VARCHAR(255) NOT NULL,
   Field VARCHAR(255) NOT NULL
);

-- Table: Firm (Normalized from R1(Firm_ID, Firm_Name, Parent_Firm_ID))
CREATE TABLE Firm (
```

```
    Firm_ID INT NOT NULL,
    Firm_Name VARCHAR(255) NOT NULL,
    Parent_Firm_ID INT NULL -- Nullable because not all firms have a parent
);

-- Table: Competitor_Firm (Normalized from R1(Firm_ID, Competitor_ID))
-- This is a junction table representing a many-to-many relationship between firms (as
competitors).
CREATE TABLE Competitor_Firm (
    Firm_ID INT NOT NULL,
    Competitor_ID INT NOT NULL -- Refers to another Firm_ID
);

-- Table: Firm_Industry (Normalized from R1(Firm_ID, Industry_ID))
-- Junction table for many-to-many relationship between Firm and Industry.
CREATE TABLE Firm_Industry (
    Firm_ID INT NOT NULL,
    Industry_ID INT NOT NULL
);

-- Table: Sector (Normalized from R1(Sector_ID, Sector_Name))
CREATE TABLE Sector (
    Sector_ID INT NOT NULL,
    Sector_Name VARCHAR(255) NOT NULL
);

-- Table: Firm_Sector (Inferred, common practice for Firm and Sector M2M relationship)
-- Junction table for many-to-many relationship between Firm and Sector.
CREATE TABLE Firm_Sector (
    Firm_ID INT NOT NULL,
    Sector_ID INT NOT NULL
);

-- Table: Portfolio (Normalized from R1(Portfolio_ID, Portfolio_Value))
CREATE TABLE Portfolio (
    Portfolio_ID INT NOT NULL,
    Portfolio_Value DECIMAL(18, 2) NOT NULL
);
```

```
-- Table: Firm_Portfolio_Allocation (Normalized from R1(FirmAllocationID,
Portfolio_Allocation, Firm_ID, Portfolio_ID))
CREATE TABLE Firm_Portfolio_Allocation (
    FirmAllocationID INT NOT NULL,
    Portfolio_Allocation DECIMAL(5, 2) NOT NULL, -- e.g., percentage like 0.75 for 75%
    Firm_ID INT NOT NULL,
    Portfolio_ID INT NOT NULL
);

-- Table: Benchmark_History (Normalized from R1(Benchmark_ID, Benchmark_Date,
Portfolio_ID))
CREATE TABLE Benchmark_History (
    Benchmark_ID INT NOT NULL,
    Benchmark_Date DATE NOT NULL,
    Portfolio_ID INT NOT NULL,
    Benchmark_Value DECIMAL(18, 4) NOT NULL -- Added a value for the benchmark
);

-- Table: Metric_Type (Normalized from R2(Metrics_Name, Type) of Financial_Metrics)
CREATE TABLE Metric_Type (
    Metrics_Name VARCHAR(255) NOT NULL,
    Type VARCHAR(100) NOT NULL -- e.g., 'Ratio', 'Currency', 'Growth Rate'
);

-- Table: Financial_Metric (Normalized from R1(Metrics_ID, Metrics_Name, Value) of
Financial_Metrics)
-- Note: The 'Value' from the original R1 is better suited for the Firm_Financial table,
-- as a metric definition (like 'Revenue') doesn't have a single universal value.
-- This table will store the definition of a metric.
CREATE TABLE Financial_Metric (
    Metrics_ID INT NOT NULL,
    Metrics_Name VARCHAR(255) NOT NULL -- This will be the FK to Metric_Type
);

-- Table: Firm_Financial (Normalized from R1(Firm_ID, Metric_ID, Metric_Date))
-- This table stores the actual financial metric values for a firm at a specific date.
CREATE TABLE Firm_Financial (
    Firm_ID INT NOT NULL,
    Metrics_ID INT NOT NULL,
```

```
    Metric_Date DATE NOT NULL,
    Metric_Value DECIMAL(18, 2) NOT NULL -- The actual value of the metric
);

-- Table: Firm_Financial_Portfolio (Normalized from R1(Portfolio_ID, Firm_ID,
Metric_ID))
-- This table links financial metrics of a firm to a specific portfolio.
-- It might represent the portion of a firm's financial metric attributable to a portfolio,
-- or a specific metric relevant to the firm's involvement in that portfolio.
CREATE TABLE Firm_Financial_Portfolio (
    Portfolio_ID INT NOT NULL,
    Firm_ID INT NOT NULL,
    Metrics_ID INT NOT NULL,
    Attributed_Value DECIMAL(18, 2) NULL -- Example: value of that metric for the firm
within this portfolio context
);

-- Table: User_Account (Normalized from R1(User_ID, User_Name, Role), renamed from
User to avoid keyword conflict)
CREATE TABLE User_Account (
    User_ID INT NOT NULL,
    User_Name VARCHAR(100) NOT NULL,
    Role VARCHAR(50) NOT NULL
);

-- Table: Transaction_Log (Normalized from R1(Log_ID, Action_Type, Date, Time,
Firm_ID, User_ID))
CREATE TABLE Transaction_Log (
    Log_ID INT NOT NULL,
    Action_Type VARCHAR(100) NOT NULL, -- e.g., 'INSERT', 'UPDATE',
'DELETE_FIRM'
    Transaction_Date DATE NOT NULL,
    Transaction_Time TIME NOT NULL,
    Firm_ID INT NULL,       -- Nullable if action is not firm-specific
    User_ID INT NOT NULL
);

--
################################################################################
```

```
-- ## ALTER TABLE Statements for PRIMARY KEY Constraints ##
--
##############################################################################

ALTER TABLE Industry ADD CONSTRAINT PK_Industry PRIMARY KEY
(Industry_ID);
ALTER TABLE Industry ADD CONSTRAINT UQ_Industry_Name UNIQUE
(Industry_Name); -- For FK reference from Industry_Field

-- Corrected PK for Industry_Field to be a composite key
ALTER TABLE Industry_Field ADD CONSTRAINT PK_Industry_Field PRIMARY KEY
(Industry_Name, Field);

ALTER TABLE Firm ADD CONSTRAINT PK_Firm PRIMARY KEY (Firm_ID);

ALTER TABLE Competitor_Firm ADD CONSTRAINT PK_Competitor_Firm PRIMARY
KEY (Firm_ID, Competitor_ID);

ALTER TABLE Firm_Industry ADD CONSTRAINT PK_Firm_Industry PRIMARY KEY
(Firm_ID, Industry_ID);

ALTER TABLE Sector ADD CONSTRAINT PK_Sector PRIMARY KEY (Sector_ID);
ALTER TABLE Sector ADD CONSTRAINT UQ_Sector_Name UNIQUE (Sector_Name);

ALTER TABLE Firm_Sector ADD CONSTRAINT PK_Firm_Sector PRIMARY KEY
(Firm_ID, Sector_ID);

ALTER TABLE Portfolio ADD CONSTRAINT PK_Portfolio PRIMARY KEY
(Portfolio_ID);

ALTER TABLE Firm_Portfolio_Allocation ADD CONSTRAINT
PK_FirmPortfolioAllocation PRIMARY KEY (FirmAllocationID);

ALTER TABLE Benchmark_History ADD CONSTRAINT PK_Benchmark_History
PRIMARY KEY (Benchmark_ID);

ALTER TABLE Metric_Type ADD CONSTRAINT PK_Metric_Type PRIMARY KEY
(Metrics_Name);
```

```
ALTER TABLE Financial_Metric ADD CONSTRAINT PK_Financial_Metric PRIMARY
KEY (Metrics_ID);
ALTER TABLE Financial_Metric ADD CONSTRAINT UQ_Financial_Metric_Name
UNIQUE (Metrics_Name); -- To ensure it links correctly
ALTER TABLE Firm_Financial ADD CONSTRAINT PK_Firm_Financial PRIMARY KEY
(Firm_ID, Metrics_ID, Metric_Date);

ALTER TABLE Firm_Financial_Portfolio ADD CONSTRAINT
PK_Firm_Financial_Portfolio PRIMARY KEY (Portfolio_ID, Firm_ID, Metrics_ID);

ALTER TABLE User_Account ADD CONSTRAINT PK_User_Account PRIMARY KEY
(User_ID);
ALTER TABLE User_Account ADD CONSTRAINT UQ_User_Name UNIQUE
(User_Name);

ALTER TABLE Transaction_Log ADD CONSTRAINT PK_Transaction_Log PRIMARY
KEY (Log_ID);

--
##############################################################################
-- ## ALTER TABLE Statements for FOREIGN KEY Constraints (Corrected for MSSQL)
##
--
##############################################################################

-- Industry_Field references Industry
ALTER TABLE Industry_Field
ADD CONSTRAINT FK_Industry_Field_Industry_Name FOREIGN KEY (Industry_Name)
REFERENCES Industry(Industry_Name);

-- Firm self-references for Parent_Firm_ID
ALTER TABLE Firm
ADD CONSTRAINT FK_Firm_Parent_Firm FOREIGN KEY (Parent_Firm_ID)
REFERENCES Firm(Firm_ID);

-- Competitor_Firm references Firm (twice)
ALTER TABLE Competitor_Firm
ADD CONSTRAINT FK_Competitor_Firm_Firm FOREIGN KEY (Firm_ID)
REFERENCES Firm(Firm_ID);
```

```
ALTER TABLE Competitor_Firm
ADD CONSTRAINT FK_Competitor_Firm_Competitor FOREIGN KEY (Competitor_ID)
REFERENCES Firm(Firm_ID);

-- Firm_Industry references Firm and Industry
ALTER TABLE Firm_Industry
ADD CONSTRAINT FK_Firm_Industry_Firm FOREIGN KEY (Firm_ID) REFERENCES
Firm(Firm_ID);

ALTER TABLE Firm_Industry
ADD CONSTRAINT FK_Firm_Industry_Industry FOREIGN KEY (Industry_ID)
REFERENCES Industry(Industry_ID);

-- Firm_Sector references Firm and Sector
ALTER TABLE Firm_Sector
ADD CONSTRAINT FK_Firm_Sector_Firm FOREIGN KEY (Firm_ID) REFERENCES
Firm(Firm_ID);

ALTER TABLE Firm_Sector
ADD CONSTRAINT FK_Firm_Sector_Sector FOREIGN KEY (Sector_ID) REFERENCES
Sector(Sector_ID);

-- Firm_Portfolio_Allocation references Firm and Portfolio
ALTER TABLE Firm_Portfolio_Allocation
ADD CONSTRAINT FK_FirmPortfolioAllocation_Firm FOREIGN KEY (Firm_ID)
REFERENCES Firm(Firm_ID);

ALTER TABLE Firm_Portfolio_Allocation
ADD CONSTRAINT FK_FirmPortfolioAllocation_Portfolio FOREIGN KEY (Portfolio_ID)
REFERENCES Portfolio(Portfolio_ID);

-- Benchmark_History references Portfolio
ALTER TABLE Benchmark_History
ADD CONSTRAINT FK_Benchmark_History_Portfolio FOREIGN KEY (Portfolio_ID)
REFERENCES Portfolio(Portfolio_ID);

-- Financial_Metric references Metric_Type
ALTER TABLE Financial_Metric
```

```
ADD CONSTRAINT FK_Financial_Metric_Metric_Type FOREIGN KEY (Metrics_Name)
REFERENCES Metric_Type(Metrics_Name);

-- Firm_Financial references Firm and Financial_Metric
ALTER TABLE Firm_Financial
ADD CONSTRAINT FK_Firm_Financial_Firm FOREIGN KEY (Firm_ID) REFERENCES
Firm(Firm_ID);

ALTER TABLE Firm_Financial
ADD CONSTRAINT FK_Firm_Financial_Metric FOREIGN KEY (Metrics_ID)
REFERENCES Financial_Metric(Metrics_ID);

-- Firm_Financial_Portfolio references Portfolio, Firm, and Financial_Metric
ALTER TABLE Firm_Financial_Portfolio
ADD CONSTRAINT FK_FFP_Portfolio FOREIGN KEY (Portfolio_ID) REFERENCES
Portfolio(Portfolio_ID);

ALTER TABLE Firm_Financial_Portfolio
ADD CONSTRAINT FK_FFP_Firm FOREIGN KEY (Firm_ID) REFERENCES
Firm(Firm_ID);

ALTER TABLE Firm_Financial_Portfolio
ADD CONSTRAINT FK_FFP_Financial_Metric FOREIGN KEY (Metrics_ID)
REFERENCES Financial_Metric(Metrics_ID);

-- Transaction_Log references Firm and User_Account
ALTER TABLE Transaction_Log
ADD CONSTRAINT FK_TransactionLog_Firm FOREIGN KEY (Firm_ID) REFERENCES
Firm(Firm_ID);

ALTER TABLE Transaction_Log
ADD CONSTRAINT FK_TransactionLog_User FOREIGN KEY (User_ID) REFERENCES
User_Account(User_ID);

--
##############################################################################
-- ## INSERT Statements for Sample Data   ##
--
##############################################################################
```

```
-- Data for: Industry
INSERT INTO Industry (Industry_ID, Industry_Name) VALUES
(1, 'Technology'),
(2, 'Healthcare'),
(3, 'Finance'),
(4, 'Manufacturing'),
(5, 'Retail'),
(6, 'Energy');

-- Data for: Industry_Field
-- With the composite PK (Industry_Name, Field), these inserts are now valid.
INSERT INTO Industry_Field (Industry_Name, Field) VALUES
('Technology', 'Software Development'),
('Technology', 'Hardware Manufacturing'),
('Healthcare', 'Pharmaceuticals'),
('Healthcare', 'Medical Devices'),
('Finance', 'Banking'),
('Finance', 'Investment Management'),
('Manufacturing', 'Automotive'),
('Retail', 'E-commerce'),
('Energy', 'Renewable Energy');

-- Data for: Firm
INSERT INTO Firm (Firm_ID, Firm_Name, Parent_Firm_ID) VALUES
(101, 'AlphaTech Solutions', NULL),
(102, 'BetaHealth Corp', NULL),
(103, 'GammaFinance Group', NULL),
(104, 'Delta Innovations', 101), -- Subsidiary of AlphaTech
(105, 'Epsilon Motors', NULL),
(106, 'Zeta Retail Inc.', NULL),
(107, 'Omega Energy Co.', NULL),
(108, 'Theta Pharma', 102), -- Subsidiary of BetaHealth
(109, 'Iota Investments', 103);

-- Data for: Competitor_Firm
-- AlphaTech competes with GammaFinance (e.g. in FinTech) and Epsilon Motors (e.g. in
automotive tech)
INSERT INTO Competitor_Firm (Firm_ID, Competitor_ID) VALUES
(101, 103),
```

```
(101, 105),
(102, 101), -- BetaHealth competes with AlphaTech (e.g. in health tech)
(105, 101); -- Epsilon Motors competes with AlphaTech

-- Data for: Firm_Industry
INSERT INTO Firm_Industry (Firm_ID, Industry_ID) VALUES
(101, 1), -- AlphaTech in Technology
(102, 2), -- BetaHealth in Healthcare
(103, 3), -- GammaFinance in Finance
(104, 1), -- Delta Innovations in Technology
(105, 4), -- Epsilon Motors in Manufacturing
(105, 1), -- Epsilon Motors also in Technology (e.g. self-driving)
(106, 5), -- Zeta Retail in Retail
(107, 6), -- Omega Energy in Energy
(108, 2); -- Theta Pharma in Healthcare

-- Data for: Sector
INSERT INTO Sector (Sector_ID, Sector_Name) VALUES
(10, 'Information Technology'),
(20, 'Life Sciences'),
(30, 'Financial Services'),
(40, 'Industrials'),
(50, 'Consumer Discretionary'),
(60, 'Utilities & Energy');

-- Data for: Firm_Sector
INSERT INTO Firm_Sector (Firm_ID, Sector_ID) VALUES
(101, 10), -- AlphaTech in Information Technology
(102, 20), -- BetaHealth in Life Sciences
(103, 30), -- GammaFinance in Financial Services
(104, 10), -- Delta Innovations in Information Technology
(105, 40), -- Epsilon Motors in Industrials
(106, 50), -- Zeta Retail in Consumer Discretionary
(107, 60), -- Omega Energy in Utilities & Energy
(101, 30); -- AlphaTech also in Financial Services (FinTech)

-- Data for: Portfolio
INSERT INTO Portfolio (Portfolio_ID, Portfolio_Value) VALUES
(1001, 5000000.00),
```

(1002, 12000000.00),
(1003, 750000.00),
(1004, 25000000.00);

-- Data for: Firm_Portfolio_Allocation
INSERT INTO Firm_Portfolio_Allocation (FirmAllocationID, Portfolio_Allocation, Firm_ID, Portfolio_ID) VALUES
(1, 0.25, 101, 1001), -- 25% of AlphaTech in Portfolio 1001
(2, 0.40, 102, 1001), -- 40% of BetaHealth in Portfolio 1001
(3, 0.35, 103, 1001), -- 35% of GammaFinance in Portfolio 1001
(4, 0.60, 101, 1002), -- 60% of AlphaTech in Portfolio 1002
(5, 0.40, 104, 1002), -- 40% of Delta Innovations in Portfolio 1002
(6, 1.00, 105, 1003), -- 100% of Epsilon Motors in Portfolio 1003
(7, 0.50, 106, 1004),
(8, 0.50, 107, 1004);

-- Data for: Benchmark_History
INSERT INTO Benchmark_History (Benchmark_ID, Benchmark_Date, Portfolio_ID, Benchmark_Value) VALUES
(1, '2023-01-01', 1001, 105.50),
(2, '2023-04-01', 1001, 107.20),
(3, '2023-07-01', 1001, 106.80),
(4, '2023-10-01', 1001, 108.90),
(5, '2023-01-01', 1002, 1200.00),
(6, '2023-06-01', 1002, 1250.75),
(7, '2024-01-01', 1002, 1300.50),
(8, '2024-01-15', 1003, 98.2);

-- Data for: Metric_Type
INSERT INTO Metric_Type (Metrics_Name, Type) VALUES
('Revenue', 'Currency'),
('Net Income', 'Currency'),
('P/E Ratio', 'Ratio'),
('Debt-to-Equity', 'Ratio'),
('Employee Count', 'Count'),
('YoY Growth', 'Percentage');

-- Data for: Financial_Metric
INSERT INTO Financial_Metric (Metrics_ID, Metrics_Name) VALUES

```
(1, 'Revenue'),
(2, 'Net Income'),
(3, 'P/E Ratio'),
(4, 'Debt-to-Equity'),
(5, 'Employee Count'),
(6, 'YoY Growth');

-- Data for: Firm_Financial
INSERT INTO Firm_Financial (Firm_ID, Metrics_ID, Metric_Date, Metric_Value)
VALUES
-- AlphaTech Financials
(101, 1, '2023-12-31', 15000000.00), -- Revenue
(101, 2, '2023-12-31', 2500000.00),  -- Net Income
(101, 3, '2023-12-31', 25.5),        -- P/E Ratio
(101, 5, '2023-12-31', 500),         -- Employee Count
-- BetaHealth Financials
(102, 1, '2023-12-31', 35000000.00), -- Revenue
(102, 2, '2023-12-31', 4500000.00),  -- Net Income
(102, 4, '2023-12-31', 0.8),         -- Debt-to-Equity
(102, 5, '2023-12-31', 1200),        -- Employee Count
-- GammaFinance Financials
(103, 1, '2023-06-30', 22000000.00),
(103, 6, '2023-06-30', 15.5),        -- YoY Growth
-- Epsilon Motors Financials
(105, 1, '2024-01-31', 75000000.00),
(105, 5, '2024-01-31', 2500);

-- Data for: Firm_Financial_Portfolio
INSERT INTO Firm_Financial_Portfolio (Portfolio_ID, Firm_ID, Metrics_ID,
Attributed_Value) VALUES
(1001, 101, 1, 3750000.00), -- 25% of AlphaTech's Revenue (15M) attributed to Portfolio
1001
(1002, 101, 1, 9000000.00), -- 60% of AlphaTech's Revenue (15M) attributed to Portfolio
1002
(1001, 102, 1, 14000000.00); -- 40% of BetaHealth's Revenue (35M) attributed to Portfolio
1001

-- Data for: User_Account
INSERT INTO User_Account (User_ID, User_Name, Role) VALUES
```

```
(10, 'Alice Wonderland', 'Administrator'),
(20, 'Bob TheBuilder', 'Analyst'),
(30, 'Charlie Brown', 'Data Entry'),
(40, 'Diana Prince', 'Analyst'),
(50, 'Edward Scissorhands', 'System Monitor');

-- Data for: Transaction_Log
INSERT INTO Transaction_Log (Log_ID, Action_Type, Transaction_Date,
Transaction_Time, Firm_ID, User_ID) VALUES
(10001, 'CREATE_FIRM', '2023-01-15', '09:30:00', 101, 10),
(10002, 'UPDATE_PORTFOLIO_VALUE', '2023-01-20', '10:00:00', NULL, 20),
(10003, 'ADD_FIRM_TO_INDUSTRY', '2023-02-01', '11:15:30', 101, 10),
(10004, 'CREATE_USER', '2023-02-05', '14:00:00', NULL, 10),
(10005, 'ASSIGN_FIRM_TO_PORTFOLIO', '2023-03-10', '16:45:00', 102, 20),
(10006, 'RECORD_FINANCIALS', '2023-04-01', '09:00:00', 101, 30),
(10007, 'DELETE_FIRM_COMPETITOR_LINK', '2024-02-10', '12:00:00', 101, 40),
(10008, 'UPDATE_FIRM_PARENT', '2024-03-15', '10:20:30', 104, 10);
```

Application Scenarios

1. A user want to see a list of firms in the “Technology” industry with the names of their direct competitors

```
SELECT
   t1.Firm_Name AS Technology_Firm,
   t3.Firm_Name AS Competitor_Firm
FROM
   (SELECT f.Firm_ID, f.Firm_Name
    FROM Firm f
    JOIN Firm_Industry fi ON f.Firm_ID = fi.Firm_ID
    JOIN Industry i ON fi.Industry_ID = i.Industry_ID
    WHERE i.Industry_Name = 'Technology') AS t1
JOIN
   Competitor_Firm t2 ON t1.Firm_ID = t2.Firm_ID
JOIN
   Firm t3 ON t2.Competitor_ID = t3.Firm_ID
ORDER BY
   t1.Firm_Name, t3.Firm_Name;
```

	Technology_Firm	Competitor_Firm
1	AlphaTech Solutions	Epsilon Motors
2	AlphaTech Solutions	GammaFinance Group
3	Epsilon Motors	AlphaTech Solutions

The query works by first identifying all firms in the 'Technology' industry. Then, it looks up each of these technology firms in the Competitor_Firm table to find their competitor IDs. Finally, it uses these competitor IDs to retrieve the names of the competitor firms from the Firm table. The output is a list where each row shows a technology firm and one of its competitors, ordered for readability.

2. A user wants to see each portfolio value and the count of firms that have an allocation within that portfolio

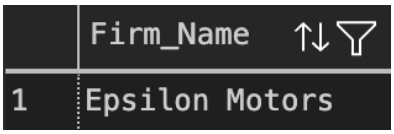
```
SELECT
   p.Portfolio_ID,
   p.Portfolio_Value,
   COUNT(fpa.Firm_ID) AS Number_of_Firms
FROM
   Portfolio p
LEFT JOIN
   Firm_Portfolio_Allocation fpa ON p.Portfolio_ID = fpa.Portfolio_ID
GROUP BY
   p.Portfolio_ID, p.Portfolio_Value
ORDER BY
   p.Portfolio_ID;
```

	Portfolio_ID	Portfolio_Value	Number_of_Firms
1	1001	5000000.00	3
2	1002	12000000.00	2
3	1003	750000.00	1
4	1004	25000000.00	2

This query retrieves each portfolio's ID and value, and then counts the number of firms linked to that portfolio through the Firm_Portfolio_Allocation table. The LEFT JOIN ensures that all portfolios are listed, even those with no firms allocated to them (where the count will be zero). Portfolio_ID then orders the results.

3. A user wants to see firms that are classified under both the “Technology” and “ Manufacturing” industries

```
SELECT
   f.Firm_Name
FROM
   Firm f
JOIN
   Firm_Industry fi1 ON f.Firm_ID = fi1.Firm_ID
JOIN
   Industry i1 ON fi1.Industry_ID = i1.Industry_ID AND i1.Industry_Name =
'Technology'
JOIN
   Firm_Industry fi2 ON f.Firm_ID = fi2.Firm_ID
JOIN
   Industry i2 ON fi2.Industry_ID = i2.Industry_ID AND i2.Industry_Name =
'Manufacturing';
```



	Firm_Name
1	Epsilon Motors

This query finds firms that are listed in both the 'Technology' and 'Manufacturing' industries. It does this by checking the firm's connections to each industry separately and then making sure the same firm has both connections.

4. An auditor wants to see which users have performed any actions (insert/update/delete) on a AlphaTech Solutions and order by most recent to oldest transaction

```
SELECT
   ua.User_Name,
   tl.Action_Type,
   tl.Transaction_Date,
   tl.Transaction_Time,
   f.Firm_Name
FROM
   User_Account ua
JOIN
   Transaction_Log tl ON ua.User_ID = tl.User_ID
```

```
JOIN
    Firm f ON tl.Firm_ID = f.Firm_ID
WHERE
    f.Firm_Name = 'AlphaTech Solutions'
ORDER BY
    tl.Transaction_Date DESC, tl.Transaction_Time DESC;
```

	User_Name	Action_Type	Transaction_Date	Transaction_Time	Firm_Name
1	Diana Prince	DELETE_FIRM_COMPETITOR_LINK	2024-02-10	12:00:00	AlphaTech Solutions
2	Charlie Brown	RECORD_FINANCIALS	2023-04-01	09:00:00	AlphaTech Solutions
3	Alice Wonderland	ADD_FIRM_TO_INDUSTRY	2023-02-01	11:15:30	AlphaTech Solutions
4	Alice Wonderland	CREATE_FIRM	2023-01-15	09:30:00	AlphaTech Solutions

This query looks into the history of actions recorded in the Transaction_Log. It finds all the entries that are associated with the firm named 'AlphaTech Solutions' and then shows you who performed those actions and when they happened, listing the most recent actions first.

5. An investor wants to identify firms that have a debt-to-equity ratio greater than 0.7 as of their latest financial reporting

```
SELECT
    f.Firm_Name,
    ff.Metric_Value AS DebtToEquity,
    ff.Metric_Date AS Reporting_Date
FROM
    Firm f
JOIN
    Firm_Financial ff ON f.Firm_ID = ff.Firm_ID
JOIN
    Financial_Metric fm ON ff.Metrics_ID = fm.Metrics_ID
WHERE
    fm.Metrics_Name = 'Debt-to-Equity'
    AND (ff.Firm_ID, ff.Metric_Date) IN (
        SELECT
            Firm_ID,
            MAX(Metric_Date)
        FROM
            Firm_Financial
        WHERE
```

```
        Metrics_ID = (SELECT Metrics_ID FROM Financial_Metric WHERE
Metrics_Name = 'Debt-to-Equity')
    GROUP BY
        Firm_ID
  )
  AND ff.Metric_Value > 0.7;
```

	Firm_Name	DebtToEquity	Reporting_Date
1	BetaHealth Corp	0.80	2023-12-31

This query efficiently retrieves the most recent Debt-to-Equity ratio for each firm. Then it displays the firm's name, that latest ratio, and the reporting date, but only for those firms where this latest ratio exceeds 0.7. It uses a subquery to determine the newest reporting date for each firm's Debt-to-Equity metric.

Conclusion

This database project was both rewarding and challenging, offering our team the opportunity to translate theoretical knowledge into a practical, business-oriented solution. One of the most demanding aspects was developing the initial proposal. It required extensive brainstorming to identify a unique business case that addressed a real need within an already saturated market. Our goal was to help firms benchmark themselves against selected competitors, an issue without a universal solution. Turning this concept into a functional database system required significant collaboration and time.

Once we had a clear business objective, designing the Entity Relationship Diagram (ERD) became more manageable. Visualizing relationships between entities such as firms, competitors, and industry averages helped us better understand how the data would interact. The normalization process, though complex, reinforced our grasp of relational integrity and allowed us to build a scalable and structured database.

Another challenge was developing realistic application scenarios. Crafting insightful, multi-table queries pushed us to think creatively about how businesses might use the system to extract meaningful insights. Despite the difficulty, it was gratifying to see our queries produce outputs aligned with practical use cases.

This project highlighted the importance of creativity in database design, particularly when building scenarios that reflect real-world applications. If given the chance to do this project again, we would focus more on refining our business case and ensuring our queries showcase the database's full capabilities.

Overall, the system we developed offers a relevant, actionable data dashboard. It enables firms to select a portfolio of competitors and receive tailored financial metrics, making the application both functional and strategically valuable. This project significantly deepened our understanding of how database systems can support real-world business decisions.