Designing and Implementing a Relational Database for Efficient Business Operations: A Case Study with Vanguard Industries

Prepared by:

Aditya Kochle, Antonio Marcos Margiotta, Marcus Nogueira and Rohan Benjamin

Instructor:

Guillaume Faddoul

Date:

December 2024

Introduction

Name of the Organization: Vanguard Industries

Industry: Artificial Intelligence/Fintech

Short Description: Vanguard Industries is a fictional AI startup focused on integrating artificial intelligence solutions into the modern finance industry. Its primary goal is to drive innovation and disruption in the fintech sector through advanced AI applications. The project specifically focuses on creating a database to streamline operations across product development, sales, marketing, client management, human resources, and operations.

Data Source: Since this is a fictional organization, all data used in the project is generated using Mockaroo, ensuring realistic variations and relationships for testing.

Reason for Choosing This Organization: We selected this organization because the AI/Fintech sector aligns with their career interests and provides an opportunity to explore database applications in a dynamic and innovative industry.

List of Entities and Descriptions

This database schema captures the core functions and relationships within the organization. It includes entities for managing AI product development, tracking sales transactions, monitoring marketing campaigns, maintaining client and employee information, and overseeing operational processes. Each entity is defined with specific attributes to ensure comprehensive data management and facilitate seamless organizational workflows.

1. **Product**
    - **Description:** Manages the details and development lifecycle of AI products.
    - **Attributes:**
        - Product_ID: Unique identifier for each product.
        - Product_Name: Name of the product.
        - Development_Start_Date: Start date of development.
        - Development_Status: Current status (e.g., ongoing, completed).
        - Release_Date: Official release date of the product.
        - Expected_Date_of_Completion: Estimated completion date.
        - Actual_Date_of_Completion: Actual completion date (if applicable).

2. **Sales (Associative Entity)**

    - **Description:** Tracks sales transactions and associated details.
    - **Attributes:**
        - Sales_ID: Unique identifier for each sale.
        - Sale_Rate: Rate per unit of the product.
        - Sales_Amount: Total amount for the sale.
        - Sale_Date: Date the sale occurred.
        - Sales_Quantity: Quantity of products sold.
3. **Marketing Campaign**

    - **Description:** Records and tracks details of marketing campaigns.
    - **Attributes:**
        - Campaign_ID: Unique identifier for each campaign.
        - Campaign_Name: Name of the campaign.
        - Target_Audience: Audience targeted by the campaign.
        - Budget: Budget allocated for the campaign.
        - Start_Date: Campaign start date.
        - End_Date: Campaign end date.
        - Channel: Marketing channel used (e.g., social media, TV).
        - Engagement_Produced: Metrics for campaign engagement (e.g., views, clicks).
4. **Client**

    - **Description:** Holds information about the organization's clients and their relationships.
    - **Attributes:**
        - Client_ID: Unique identifier for each client.
        - Client_Name: Name of the client.
        - Industry: Industry the client operates in.
        - Email: Client email address.
        - Phone: Client phone number.
5. **Employee**

    - **Description:** Maintains employee data within the organization.
    - **Attributes:**
        - Employee_ID: Unique identifier for each employee.
        - Name: Full name of the employee.
        - Role: Employee's role (e.g., Developer, Manager).
        - Hire_Date: Date the employee was hired.
        - Total_Compensation: Total salary and benefits package.
6. **Operations**

    - **Description:** Manages the operational processes within the organization.

- **Attributes:**
  - Operations_ID: Unique identifier for each operation.
  - Process_Name: Name of the process being tracked.
  - AI_Automation_Level: Level of AI automation in the process.
  - Status: Current status (e.g., completed, pending).
  - OperationType: Type of operation (e.g., data preprocessing, deployment).

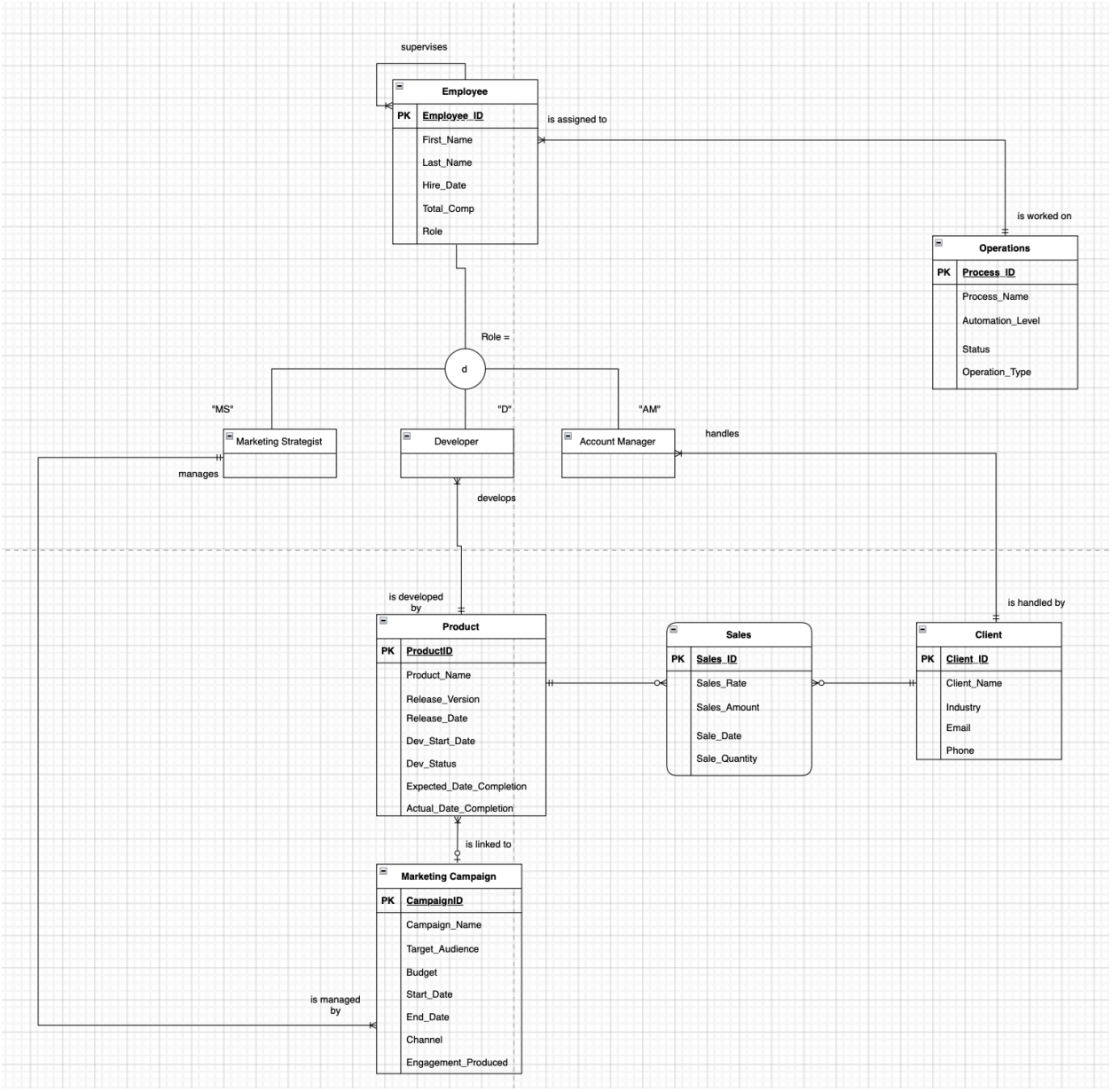
Business Rules

1. Each product may have a marketing campaign; each marketing campaign must be linked to one or more products.
2. Each client may purchase one or more products; each product can be sold to multiple clients.
3. Each marketing strategist must be assigned to one or more marketing campaigns; each marketing campaign must be managed by only one marketing strategist.
4. Each manager supervises many employees; all employees must report to one manager.
5. Each developer (employee) must work on one product; each product must have at least one developer.
6. Each employee must be assigned to one operation process type; each operation process type must involve multiple employees.
7. Each account manager (employee) must manage only one client account; each client account may be managed by multiple account managers.

The business rules outlined establish the structural and operational framework for managing relationships and workflows within the organization. These rules ensure clarity and efficiency by defining key associations between entities such as products, marketing campaigns, clients,

employees, and operations. For example, linking marketing campaigns to specific products ensures targeted promotions, while associating clients with multiple products facilitates a broader sales strategy. Assigning marketing strategists to specific campaigns promotes accountability, and the hierarchical structure, where employees report to managers, supports clear lines of supervision. Requiring developers to work on products guarantees dedicated focus on product development, and connecting employees to specific operation process types fosters specialization. Finally, the relationship between account managers and client accounts emphasizes personalized client service while allowing collaborative management. These rules ensure operational coherence, accountability, and alignment with the organization's strategic objectives.

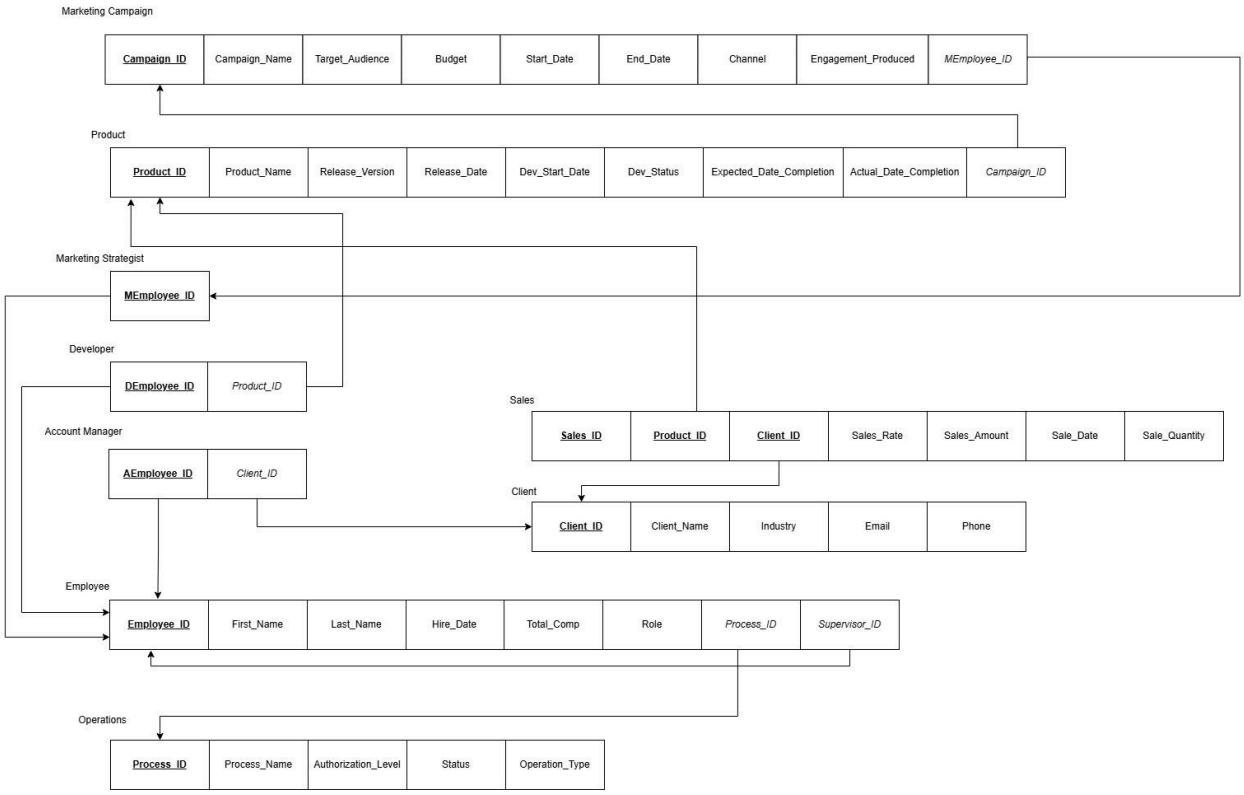
ER Diagram (or EER Diagram)



Streamlined Roles and Lifecycle Integration

Clearly defined employee roles, including Marketing Strategists, Developers, and Account Managers, ensure responsibilities are well understood while emphasizing collaboration across interconnected roles. Product lifecycle tracking seamlessly integrates development, marketing, and delivery processes, providing end-to-end visibility and efficiency.

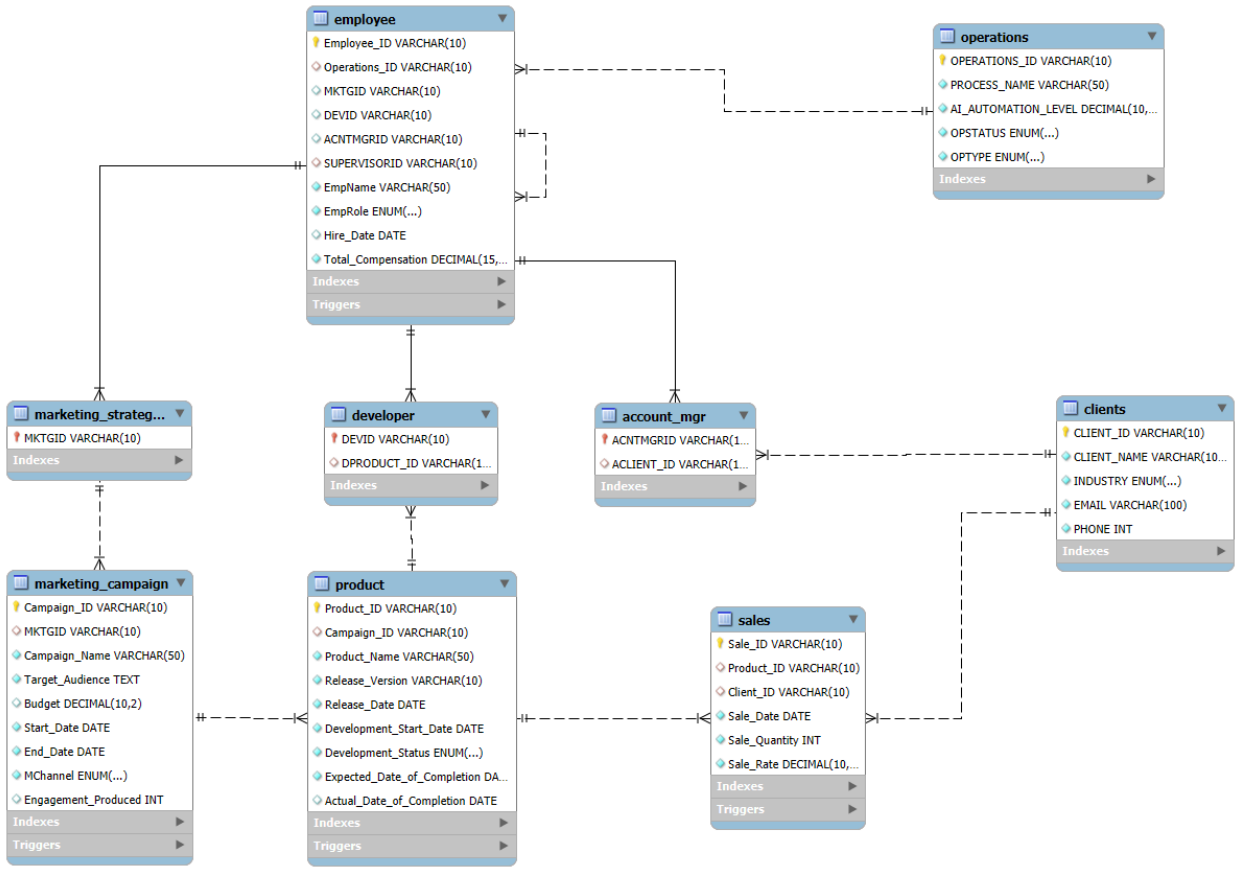
3NF Relation Model



Integrated Data Relationships and Workflow Alignment

The model emphasizes clear, normalized connections between entities, aligning the data structure with business strategy. By linking development, marketing, sales, and client relationships, it provides an end-to-end view of operations. Operational linkages tie employees to specific business processes, ensuring the model reflects real-world workflows seamlessly.

2nd EER Diagram



List of SQL Statements:

Create Tables:

The SQL schema below outlines a comprehensive database system for managing an AI company's business processes, including clients, employees, marketing campaigns, operations, products, developers, and sales. Here's a summary:

Schemas and Tables

1. Clients Table: Stores information about clients, including `CLIENT_ID`, `CLIENT_NAME`, `INDUSTRY`, `EMAIL`, and `PHONE`. Constraints ensure unique client entries and enforce data integrity.

2. Operations Table: Tracks operational processes, including `OPERATIONS_ID`, `PROCESS_NAME`, `AI_AUTOMATION_LEVEL`, `OPSTATUS`, and `OPTYPE`. Automation levels and status are validated through constraints.

3. Employee Table: Manages employee data with roles like developers, marketing strategists, and account managers. Includes hierarchical relationships (e.g., supervisor IDs) and constraints to validate hiring dates.

4. Marketing Strategist and Account Manager Tables: Sub-tables of employees link marketing campaigns and client accounts, ensuring specific roles are appropriately assigned.

5. Marketing Campaign Table: Tracks campaigns linked to products, including details like `Campaign_ID`, `Target_Audience`, `Budget`, `Start_Date`, and `End_Date`. Triggers validate campaign dates.

6. Product Table: Records product details, linking them to marketing campaigns, developers, and statuses like "Ongoing" or "Completed." Triggers enforce date relationships and auto-update completion dates.

7. Developer Table: Associates developers with specific products, ensuring accountability for product development.

8. Sales Table: Tracks sales transactions, including `Sale_ID`, `Product_ID`, `Client_ID`, `Sale_Date`, `Sale_Quantity`, and `Sale_Rate`. Triggers prevent future-dated sales.

Triggers
- Date Validation Triggers: Ensure data consistency by validating dates (e.g., hire dates, campaign dates, product release dates, and sales dates).
- Status-Dependent Triggers: Automatically update the actual completion date of products when their development status changes to "Completed."

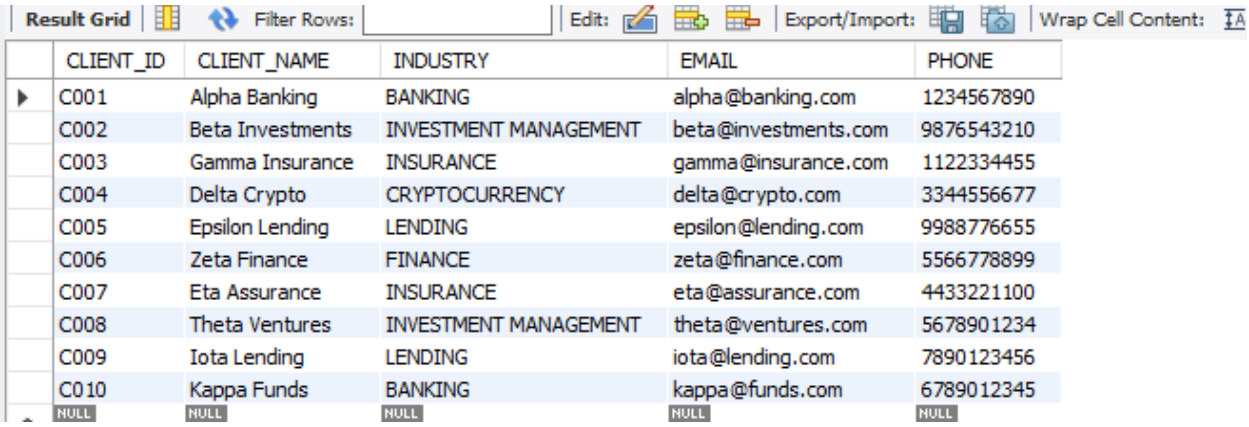
Data Insertions
- Example entries populate all tables, illustrating relationships:
  - Clients span industries like banking, insurance, and cryptocurrency.
  - Operations include processes like data cleaning and model training with varying automation levels.
  - Employees are linked to operations and products with diverse roles such as developers and marketing strategists.
  - Marketing Campaigns target different audiences with defined budgets and channels.
  - Products are in various stages of development, tied to campaigns or operating independently.
  - Sales record transactional details for specific products and clients.

This schema enforces strong relational integrity, ensures consistent and accurate data management, and supports business processes for our AI company. It highlights operational transparency, accountability, and workflow alignment among the various roles and entities.

-- CREATING SCHEMAS FOR USAGE

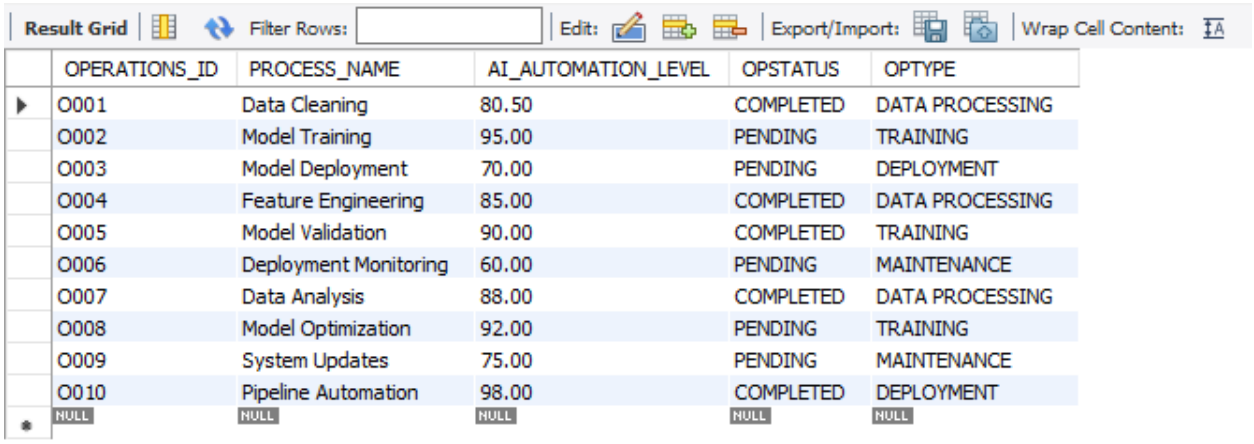
```
CREATE SCHEMA AICOMPANY_DB2;
USE AICOMPANY_DB2;

DROP TABLE IF EXISTS CLIENTS;
CREATE TABLE CLIENTS
(
CLIENT_ID               VARCHAR(10) NOT NULL UNIQUE,
CLIENT_NAME VARCHAR(100) NOT NULL UNIQUE,
INDUSTRY                ENUM('BANKING', 'INVESTMENT MANAGEMENT', 'INSURANCE',
'CRYPTOCURRENCY', 'LENDING', 'FINANCE' ) NOT NULL,
EMAIL                   VARCHAR(100) NOT NULL,
PHONE                   VARCHAR(15) NOT NULL,
CONSTRAINT CLIENT_ID PRIMARY KEY (CLIENT_ID)
);


INSERT INTO CLIENTS (CLIENT_ID, CLIENT_NAME, INDUSTRY, EMAIL, PHONE)VALUES
('C001', 'Alpha Banking', 'BANKING', 'alpha@banking.com', 1234567890),
('C002', 'Beta Investments', 'INVESTMENT MANAGEMENT', 'beta@investments.com', 9876543210),
('C003', 'Gamma Insurance', 'INSURANCE', 'gamma@insurance.com', 1122334455),
('C004', 'Delta Crypto', 'CRYPTOCURRENCY', 'delta@crypto.com', 3344556677),
```

('C005', 'Epsilon Lending', 'LENDING', 'epsilon@lending.com', 9988776655),
('C006', 'Zeta Finance', 'FINANCE', 'zeta@finance.com', 5566778899),
('C007', 'Eta Assurance', 'INSURANCE', 'eta@assurance.com', 4433221100),
('C008', 'Theta Ventures', 'INVESTMENT MANAGEMENT', 'theta@ventures.com', 5678901234),
('C009', 'Iota Lending', 'LENDING', 'iota@lending.com', 7890123456),
('C010', 'Kappa Funds', 'BANKING', 'kappa@funds.com', 6789012345);


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CLIENT_ID	CLIENT_NAME	INDUSTRY	EMAIL	PHONE
C001	Alpha Banking	BANKING	alpha@banking.com	1234567890
C002	Beta Investments	INVESTMENT MANAGEMENT	beta@investments.com	9876543210
C003	Gamma Insurance	INSURANCE	gamma@insurance.com	1122334455
C004	Delta Crypto	CRYPTOCURRENCY	delta@crypto.com	3344556677
C005	Epsilon Lending	LENDING	epsilon@lending.com	9988776655
C006	Zeta Finance	FINANCE	zeta@finance.com	5566778899
C007	Eta Assurance	INSURANCE	eta@assurance.com	4433221100
C008	Theta Ventures	INVESTMENT MANAGEMENT	theta@ventures.com	5678901234
C009	Iota Lending	LENDING	iota@lending.com	7890123456
C010	Kappa Funds	BANKING	kappa@funds.com	6789012345
NULL	NULL	NULL	NULL	NULL

```
DROP TABLE IF EXISTS OPERATIONS;
CREATE TABLE OPERATIONS
(
OPERATIONS_ID                VARCHAR(10) NOT NULL,
PROCESS_NAME                 VARCHAR(50) NOT NULL UNIQUE,
AI_AUTOMATION_LEVEL          DECIMAL (10,2) NOT NULL CHECK (AI_AUTOMATION_LEVEL <=
100),
OPSTATUS                     ENUM('PENDING', 'COMPLETED', 'DROPPED') NOT NULL,
OPTYPE                       ENUM('DATA PROCESSING', 'MODEL', 'TRAINING', 'DEPLOYMENT',
'MAINTENANCE') NOT NULL,
CONSTRAINT OPERATIONSID PRIMARY KEY (OPERATIONS_ID)
);

INSERT INTO OPERATIONS (OPERATIONS_ID, PROCESS_NAME, AI_AUTOMATION_LEVEL,
OPSTATUS, OPTYPE)VALUES
('O001', 'Data Cleaning', 80.50, 'COMPLETED', 'DATA PROCESSING'),
('O002', 'Model Training', 95.00, 'PENDING', 'TRAINING'),
('O003', 'Model Deployment', 70.00, 'PENDING', 'DEPLOYMENT'),
('O004', 'Feature Engineering', 85.00, 'COMPLETED', 'DATA PROCESSING'),
('O005', 'Model Validation', 90.00, 'COMPLETED', 'TRAINING'),
('O006', 'Deployment Monitoring', 60.00, 'PENDING', 'MAINTENANCE'),
('O007', 'Data Analysis', 88.00, 'COMPLETED', 'DATA PROCESSING'),
('O008', 'Model Optimization', 92.00, 'PENDING', 'TRAINING'),
('O009', 'System Updates', 75.00, 'PENDING', 'MAINTENANCE'),
('O010', 'Pipeline Automation', 98.00, 'COMPLETED', 'DEPLOYMENT');
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

OPERATIONS_ID	PROCESS_NAME	AI_AUTOMATION_LEVEL	OPSTATUS	OPTYPE
O001	Data Cleaning	80.50	COMPLETED	DATA PROCESSING
O002	Model Training	95.00	PENDING	TRAINING
O003	Model Deployment	70.00	PENDING	DEPLOYMENT
O004	Feature Engineering	85.00	COMPLETED	DATA PROCESSING
O005	Model Validation	90.00	COMPLETED	TRAINING
O006	Deployment Monitoring	60.00	PENDING	MAINTENANCE
O007	Data Analysis	88.00	COMPLETED	DATA PROCESSING
O008	Model Optimization	92.00	PENDING	TRAINING
O009	System Updates	75.00	PENDING	MAINTENANCE
O010	Pipeline Automation	98.00	COMPLETED	DEPLOYMENT
NULL	NULL	NULL	NULL	NULL

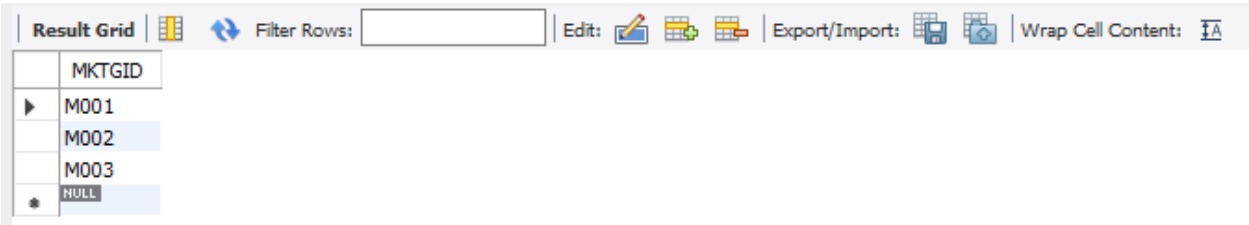
```
DROP TABLE IF EXISTS EMPLOYEE;
CREATE TABLE EMPLOYEE
(
Employee_ID       VARCHAR(10) PRIMARY KEY,
Operations_ID     VARCHAR(10),
MKTGID            VARCHAR(10) UNIQUE, -- Added UNIQUE constraint
DEVID             VARCHAR(10) UNIQUE, -- Added UNIQUE constraint
ACNTMGRID         VARCHAR(10) UNIQUE, -- Added UNIQUE constraint
SUPERVISORID  VARCHAR(10),
EmpName           VARCHAR(50) NOT NULL,
EmpRole           ENUM('AI ENGINEER', 'CYBERSECURITY ANALYST', 'DATA SCIENTIST',
'CRYPTOCURRENCY ANALYST', 'AI RESEARCH SCIENTIST', 'AI PRODUCT MANAGER', 'MACHINE
LEARNING ENGINEER', 'MARKETING STRATEGIST', 'DEVELOPER', 'ACCOUNT MANAGER') NOT
NULL,
Hire_Date             DATE,
Total_Compensation     DECIMAL(15, 2) NOT NULL CHECK (Total_Compensation > 0),
CONSTRAINT EMP_FK FOREIGN KEY (Operations_ID) REFERENCES OPERATIONS(Operations_ID),
CONSTRAINT EMP_FK2 FOREIGN KEY (SUPERVISORID) REFERENCES EMPLOYEE(Employee_ID)
);

-- Trigger for Hire_Date validation
DELIMITER //
CREATE TRIGGER hire_date_check
BEFORE INSERT ON EMPLOYEE
FOR EACH ROW
BEGIN
    IF NEW.Hire_Date > CURRENT_DATE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Hire_Date cannot be in the future';
    END IF;
END;
//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER hire_date_check2
BEFORE UPDATE ON EMPLOYEE
FOR EACH ROW
BEGIN
   IF NEW.Hire_Date > CURRENT_DATE THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Hire_Date cannot be in the future';
   END IF;
END;
//
DELIMITER ;

INSERT INTO EMPLOYEE (Employee_ID, Operations_ID, MKTGID, DEVID, ACNTMGRID,
SUPERVISORID, EmpName, EmpRole, Hire_Date, Total_Compensation)
VALUES
('E001', 'O001', NULL, 'D001', NULL, NULL, 'Alice Johnson', 'DEVELOPER', '2022-01-15', 85000.00),
('E002', 'O002', 'M001', NULL, NULL, 'E001', 'Bob Smith', 'MARKETING STRATEGIST', '2021-05-20',
75000.00),
('E003', 'O003', NULL, NULL, 'A001', 'E001', 'Charlie Brown', 'ACCOUNT MANAGER', '2023-03-10',
70000.00),
('E004', 'O004', NULL, 'D002', NULL, 'E001', 'David Lee', 'DEVELOPER', '2021-12-01', 82000.00),
('E005', 'O005', 'M002', NULL, NULL, 'E002', 'Emily Davis', 'MARKETING STRATEGIST', '2022-06-15',
76000.00),
('E006', 'O006', NULL, 'D003', NULL, 'E001', 'Frank Miller', 'DEVELOPER', '2023-02-10', 81000.00),
('E007', 'O007', NULL, NULL, 'A002', 'E003', 'Grace Hall', 'ACCOUNT MANAGER', '2021-08-05',
72000.00),
('E008', 'O008', 'M003', NULL, NULL, 'E002', 'Hannah Wilson', 'MARKETING STRATEGIST', '2023-05-01',
74000.00),
('E009', 'O009', NULL, 'D004', NULL, 'E001', 'Ian Brown', 'DEVELOPER', '2021-11-20', 83000.00),
('E010', 'O010', NULL, NULL, 'A003', 'E003', 'Jessica Green', 'ACCOUNT MANAGER', '2022-04-25',
71000.00);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

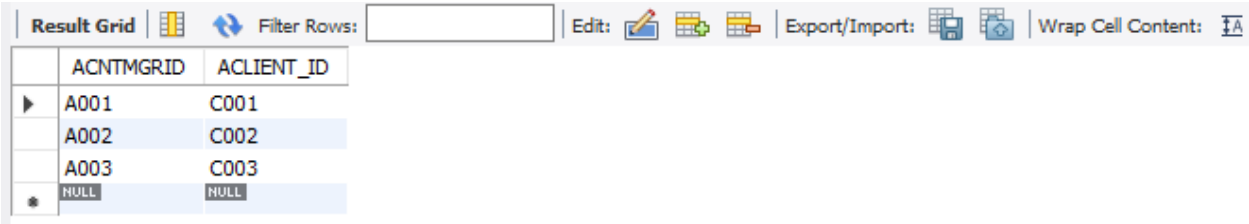
Employee_ID	Operations_ID	MKTGID	DEVID	ACNTMGRID	SUPERVISORID	EmpName	EmpRole	Hire_Date	Total_Compensation
E001	O001	NULL	D001	NULL	NULL	Alice Johnson	DEVELOPER	2022-01-15	85000.00
E002	O002	M001	NULL	NULL	E001	Bob Smith	MARKETING STRATEGIST	2021-05-20	75000.00
E003	O003	NULL	NULL	A001	E001	Charlie Brown	ACCOUNT MANAGER	2023-03-10	70000.00
E004	O004	NULL	D002	NULL	E001	David Lee	DEVELOPER	2021-12-01	82000.00
E005	O005	M002	NULL	NULL	E002	Emily Davis	MARKETING STRATEGIST	2022-06-15	76000.00
E006	O006	NULL	D003	NULL	E001	Frank Miller	DEVELOPER	2023-02-10	81000.00
E007	O007	NULL	NULL	A002	E003	Grace Hall	ACCOUNT MANAGER	2021-08-05	72000.00
E008	O008	M003	NULL	NULL	E002	Hannah Wilson	MARKETING STRATEGIST	2023-05-01	74000.00
E009	O009	NULL	D004	NULL	E001	Ian Brown	DEVELOPER	2021-11-20	83000.00
E010	O010	NULL	NULL	A003	E003	Jessica Green	ACCOUNT MANAGER	2022-04-25	71000.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
-- Create MARKETING_STRATEGIST table
CREATE TABLE MARKETING_STRATEGIST
(
MKTGID          VARCHAR(10) PRIMARY KEY,
CONSTRAINT MKTG_FK FOREIGN KEY (MKTGID) REFERENCES EMPLOYEE(MKTGID)
);

INSERT INTO MARKETING_STRATEGIST (MKTGID) VALUES
('M001'),
('M002'),
('M003');
```



MKTGID
M001
M002
M003
NULL

```
-- Create ACCOUNT_MGR table
CREATE TABLE ACCOUNT_MGR
(
ACNTMGRID               VARCHAR(10) PRIMARY KEY,
ACLIENT_ID              VARCHAR(10),
CONSTRAINT ACNTMGR_FK FOREIGN KEY (ACNTMGRID) REFERENCES
EMPLOYEE(ACNTMGRID),
CONSTRAINT ACLIENT_FK FOREIGN KEY (ACLIENT_ID) REFERENCES CLIENTS(Client_ID)
);

INSERT INTO ACCOUNT_MGR (ACNTMGRID, ACLIENT_ID)VALUES
('A001', 'C001'),
('A002', 'C002'),
('A003', 'C003');
```



ACNTMGRID	ACLIENT_ID
A001	C001
A002	C002
A003	C003
NULL	NULL

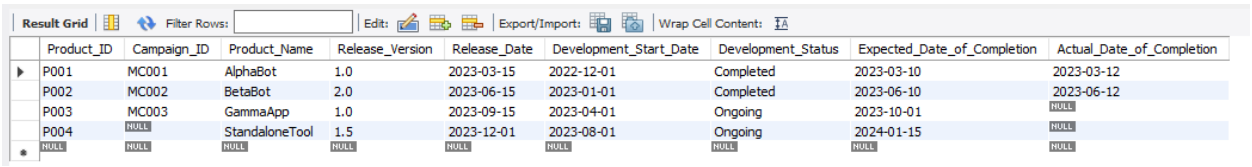
```
CREATE TABLE Marketing_Campaign
(
Campaign_ID       VARCHAR(10),
MKTGID            VARCHAR(10),
Campaign_Name  VARCHAR(50) NOT NULL UNIQUE,
Target_Audience   TEXT NOT NULL,
Budget            DECIMAL(10, 2) CHECK (Budget > 0),
Start_Date        DATE NOT NULL,
End_Date          DATE NOT NULL,
MChannel          ENUM('Social Media', 'TV', 'Email', 'Radio') NOT NULL,
Engagement_Produced   INT CHECK (Engagement_Produced >= 0),
CONSTRAINT MARKETING_PK PRIMARY KEY (Campaign_ID),
CONSTRAINT MARKETING_FK FOREIGN KEY (MKTGID) REFERENCES
MARKETING_STRATEGIST(MKTGID)
);

-- Adding a trigger to check for date validation

DELIMITER //

CREATE TRIGGER validate_dates
BEFORE INSERT ON Marketing_Campaign
FOR EACH ROW
BEGIN
   IF NEW.End_Date < NEW.Start_Date THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'End_Date must be greater than or equal to Start_Date.';
   END IF;
END;

//
DELIMITER ;

INSERT INTO Marketing_Campaign (Campaign_ID, MKTGID, Campaign_Name, Target_Audience,
Budget, Start_Date, End_Date, MChannel, Engagement_Produced) VALUES
('MC001', 'M001', 'Campaign Alpha', 'Tech Enthusiasts', 5000.00, '2023-01-01', '2023-03-01', 'Social
Media', 1500),
('MC002', 'M002', 'Campaign Beta', 'Young Adults', 6000.00, '2023-04-01', '2023-06-01', 'TV', 2000),
('MC003', 'M003', 'Campaign Gamma', 'Small Businesses', 7000.00, '2023-07-01', '2023-09-01', 'Email',
1800);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Campaign_ID	MKTGID	Campaign_Name	Target_Audience	Budget	Start_Date	End_Date	MChannel	Engagement_Produced
MC001	M001	Campaign Alpha	Tech Enthusiasts	5000.00	2023-01-01	2023-03-01	Social Media	1500
MC002	M002	Campaign Beta	Young Adults	6000.00	2023-04-01	2023-06-01	TV	2000
MC003	M003	Campaign Gamma	Small Businesses	7000.00	2023-07-01	2023-09-01	Email	1800
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
 CREATE TABLE Product
(
Product_ID                      VARCHAR(10) NOT NULL UNIQUE,
Campaign_ID                     VARCHAR(10),
Product_Name                    VARCHAR(50) NOT NULL UNIQUE,
Release_Version                 VARCHAR(10) NOT NULL,
Release_Date                    DATE NOT NULL,
Development_Start_Date          DATE NOT NULL,
Development_Status              ENUM('Ongoing', 'Completed', 'Paused') NOT NULL,
Expected_Date_of_Completion     DATE NOT NULL,
Actual_Date_of_Completion       DATE DEFAULT NULL,
CONSTRAINT PRODUCT_PK PRIMARY KEY PRODUCT(PRODUCT_ID),
CONSTRAINT PRODUCT_FK FOREIGN KEY (CAMPAIGN_ID) REFERENCES Marketing_Campaign
(CAMPAIGN_ID)
);

-- Create a trigger to add actual date of completion once the development status is set to completed

DELIMITER $$

CREATE TRIGGER UpdateActualDateOfCompletion
BEFORE UPDATE ON Product
FOR EACH ROW
BEGIN
   -- Check if Development_Status is being changed to 'Completed'
   IF NEW.Development_Status = 'Completed' AND OLD.Development_Status <> 'Completed' THEN
      SET NEW.Actual_Date_of_Completion = CURDATE();
   END IF;
END $$

DELIMITER ;

-- Insert triggers to enforce release_date and expected_date_of_completion constraints

DELIMITER $$

CREATE TRIGGER CheckReleaseDate
BEFORE INSERT ON Product
FOR EACH ROW
BEGIN
   IF NEW.Release_Date < NEW.Development_Start_Date THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Release_Date must be greater than or equal to
Development_Start_Date.';
   END IF;
END $$

DELIMITER ;
```

```
DELIMITER $$

CREATE TRIGGER CheckExpectedCompletionDate
BEFORE INSERT ON Product
FOR EACH ROW
BEGIN
   IF NEW.Expected_Date_of_Completion < NEW.Development_Start_Date THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Expected_Date_of_Completion must be greater than or equal to
Development_Start_Date.';
   END IF;
END $$

DELIMITER ;

-- Update triggers to enforce release_date and expected_date_of_completion constraints

DELIMITER $$

CREATE TRIGGER CheckReleaseDateUpdate
BEFORE UPDATE ON Product
FOR EACH ROW
BEGIN
   IF NEW.Release_Date < NEW.Development_Start_Date THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Release_Date must be greater than or equal to
Development_Start_Date.';
   END IF;
END $$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER CheckExpectedCompletionDateUpdate
BEFORE UPDATE ON Product
FOR EACH ROW
BEGIN
   IF NEW.Expected_Date_of_Completion < NEW.Development_Start_Date THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Expected_Date_of_Completion must be greater than or equal to
Development_Start_Date.';
   END IF;
END $$

DELIMITER ;
```

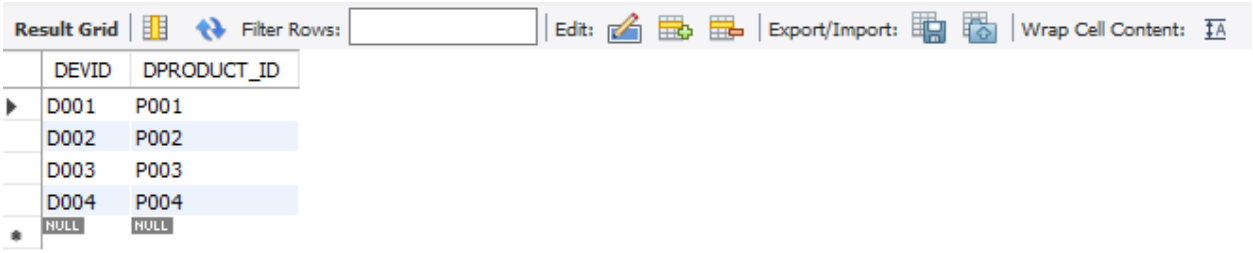


Product_ID	Campaign_ID	Product_Name	Release_Version	Release_Date	Development_Start_Date	Development_Status	Expected_Date_of_Completion	Actual_Date_of_Completion
P001	MC001	AlphaBot	1.0	2023-03-15	2022-12-01	Completed	2023-03-10	2023-03-12
P002	MC002	BetaBot	2.0	2023-06-15	2023-01-01	Completed	2023-06-10	2023-06-12
P003	MC003	GammaApp	1.0	2023-09-15	2023-04-01	Ongoing	2023-10-01	NULL
P004	NULL	StandaloneTool	1.5	2023-12-01	2023-08-01	Ongoing	2024-01-15	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
-- Create DEVELOPER table
CREATE TABLE DEVELOPER
(
   DEVID VARCHAR(10) PRIMARY KEY,
   DPRODUCT_ID VARCHAR(10),
   CONSTRAINT DEV_FK FOREIGN KEY (DEVID) REFERENCES EMPLOYEE(DEVID),
   CONSTRAINT DPRODUCT_FK FOREIGN KEY (DPRODUCT_ID) REFERENCES
PRODUCT(Product_ID)
);
```

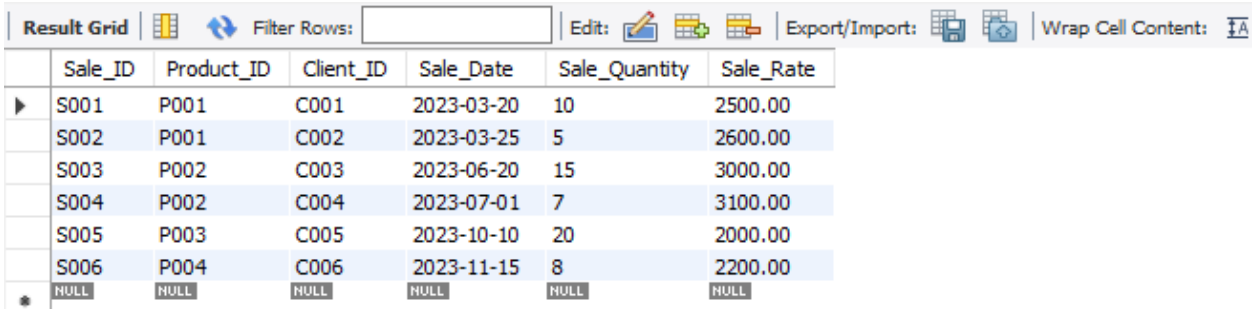
```
INSERT INTO Product (Product_ID, Campaign_ID, Product_Name, Release_Version, Release_Date,
Development_Start_Date, Development_Status, Expected_Date_of_Completion,
Actual_Date_of_Completion)VALUES
('P001', 'MC001', 'AlphaBot', '1.0', '2023-03-15', '2022-12-01', 'Completed', '2023-03-10', '2023-03-12'),
('P002', 'MC002', 'BetaBot', '2.0', '2023-06-15', '2023-01-01', 'Completed', '2023-06-10', '2023-06-12'),
('P003', 'MC003', 'GammaApp', '1.0', '2023-09-15', '2023-04-01', 'Ongoing', '2023-10-01', NULL),
('P004', NULL, 'StandaloneTool', '1.5', '2023-12-01', '2023-08-01', 'Ongoing', '2024-01-15', NULL);
```

```
INSERT INTO DEVELOPER (DEVID, DPRODUCT_ID)VALUES
('D001', 'P001'),
('D002', 'P002'),
('D003', 'P003'),
('D004', 'P004');
```



DEVID	DPRODUCT_ID
D001	P001
D002	P002
D003	P003
D004	P004
NULL	NULL

```
DROP TABLE IF EXISTS Sales;

CREATE TABLE Sales
(
   Sale_ID              VARCHAR(10) NOT NULL UNIQUE,
   Product_ID           VARCHAR(10),
   Client_ID            VARCHAR(10),
   Sale_Date            DATE NOT NULL,
   Sale_Quantity        INT NOT NULL,
   Sale_Rate            DECIMAL(10, 2) NOT NULL CHECK (Sale_Rate > 0),

   CONSTRAINT SALES_PK PRIMARY KEY (Sale_ID),
   CONSTRAINT SALES_FK FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
   CONSTRAINT SALES_FK2 FOREIGN KEY (Client_ID) REFERENCES Clients(Client_ID)
);
-- Adding the trigger for Sale Date validation
DELIMITER //

CREATE TRIGGER validate_sale_date
BEFORE INSERT ON Sales
FOR EACH ROW
BEGIN
   IF NEW.Sale_Date > CURDATE() THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Sale_Date cannot be greater than the current date.';
   END IF;
END;

//
DELIMITER ;

INSERT INTO Sales (Sale_ID, Product_ID, Client_ID, Sale_Date, Sale_Quantity, Sale_Rate)VALUES
('S001', 'P001', 'C001', '2023-03-20', 10, 2500.00),
('S002', 'P001', 'C002', '2023-03-25', 5, 2600.00),
('S003', 'P002', 'C003', '2023-06-20', 15, 3000.00),
('S004', 'P002', 'C004', '2023-07-01', 7, 3100.00),
('S005', 'P003', 'C005', '2023-10-10', 20, 2000.00),
('S006', 'P004', 'C006', '2023-11-15', 8, 2200.00);
```



Sale_ID	Product_ID	Client_ID	Sale_Date	Sale_Quantity	Sale_Rate
S001	P001	C001	2023-03-20	10	2500.00
S002	P001	C002	2023-03-25	5	2600.00
S003	P002	C003	2023-06-20	15	3000.00
S004	P002	C004	2023-07-01	7	3100.00
S005	P003	C005	2023-10-10	20	2000.00
S006	P004	C006	2023-11-15	8	2200.00
NULL	NULL	NULL	NULL	NULL	NULL

Retrieving Information from Tables:

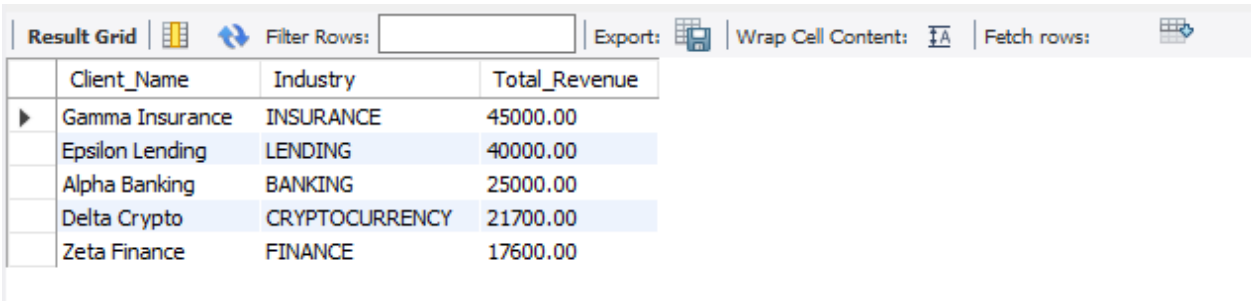
Queries

Query 1: Revenue-Driven Client and Industry Insights

```
SELECT C.Client_Name, C.Industry, SUM(S.Sale_Quantity * S.Sale_Rate) AS Total_Revenue
FROM Sales S JOIN Clients C ON S.Client_ID = C.Client_ID
GROUP BY C.Client_ID, C.Client_Name, C.Industry
ORDER BY Total_Revenue DESC
LIMIT 5;
```

Key Insights:

1. Top Clients: Identify the clients contributing the most revenue. These are our most valuable customers.
2. Industry Insights: By grouping data by industry, we are able to assess which sectors are most engaged with our products or services.
3. Strategic Focus: Understanding which clients and industries drive the highest revenue allows us to tailor our outreach and product development to meet their needs more effectively.
4. Feedback and Loyalty: Engaging with these top contributors can provide critical feedback for improving our offerings and strengthening business relationships.



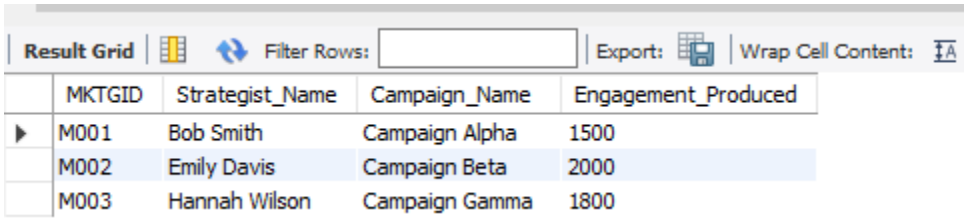
Client_Name	Industry	Total_Revenue
Gamma Insurance	INSURANCE	45000.00
Epsilon Lending	LENDING	40000.00
Alpha Banking	BANKING	25000.00
Delta Crypto	CRYPTOCURRENCY	21700.00
Zeta Finance	FINANCE	17600.00

Query 2: Top-Performing Marketing Campaign Insights

```
SELECT MC.MKTGID, E.EmpName AS Strategist_Name, MC.Campaign_Name,
MC.Engagement_Produced
FROM Marketing_Campaign MC JOIN Marketing_Strategist MS ON MC.MKTGID = MS.MKTGID JOIN
Employee E ON MS.MKTGID = E.MKTGID
WHERE Engagement_Produced = (
      SELECT MAX(Engagement_Produced)
      FROM Marketing_Campaign
      WHERE Marketing_Campaign.MKTGID = MC.MKTGID
  );
```

Key Insights:

1. High-Impact Campaign Identified: Highlights the marketing campaign with the highest engagement, showcasing a standout performer among all campaigns.
2. Strategist Accountability: The responsible strategist for the top campaign is clearly identified, recognizing individual contributions.
3. Focused Engagement Data: Engagement metrics are isolated for the highest performer, offering precise data to replicate success.
4. Campaign-Level Comparison: By using a subquery for maximum engagement, this approach ensures fair benchmarking across campaigns.


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

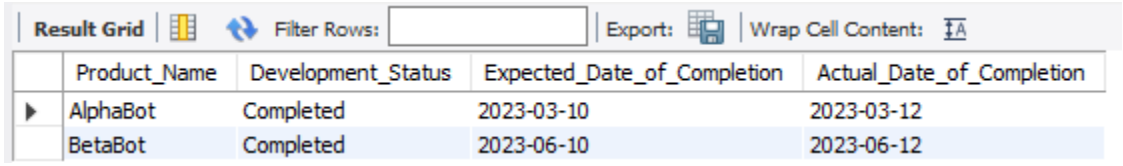
MKTGID	Strategist_Name	Campaign_Name	Engagement_Produced
M001	Bob Smith	Campaign Alpha	1500
M002	Emily Davis	Campaign Beta	2000
M003	Hannah Wilson	Campaign Gamma	1800

Query 3: Delayed Product Development Insights

```
SELECT P.Product_Name, P.Development_Status, P.Expected_Date_of_Completion,
P.Actual_Date_of_Completion
FROM Product P
WHERE P.Actual_Date_of_Completion > P.Expected_Date_of_Completion;
```

Key Insights:

1. Product Delays Identified: Pinpoints products that exceeded their expected completion dates, highlighting areas for improvement in project timelines.
2. Development Status Overview: Provides visibility into the current status of delayed products, aiding in prioritization and resource allocation.
3. Timeline Gaps: Captures both expected and actual completion dates, offering precise data to analyze the extent of delays.
4. Process Optimization Opportunity: Identify recurring bottlenecks or inefficiencies in the product development process.


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

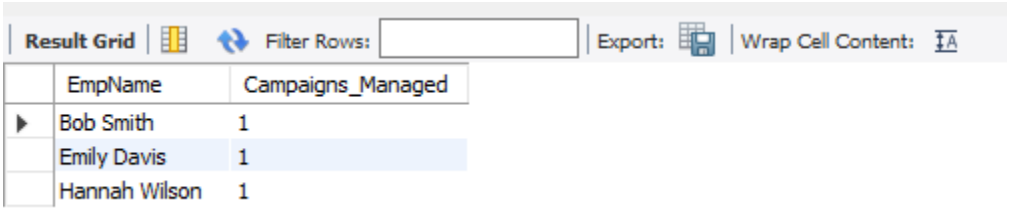
Product_Name	Development_Status	Expected_Date_of_Completion	Actual_Date_of_Completion
AlphaBot	Completed	2023-03-10	2023-03-12
BetaBot	Completed	2023-06-10	2023-06-12

Query 4: Employee with Highest Campaign Involvement

```
SELECT E.EmpName, COUNT(MC.Campaign_ID) AS Campaigns_Managed
FROM Employee E JOIN Marketing_Strategist MS ON E.MKTGID = MS.MKTGID JOIN
Marketing_Campaign MC ON MS.MKTGID = MC.MKTGID
GROUP BY E.Employee_ID
HAVING Campaigns_Managed = (
                                    SELECT MAX(Campaigns_Count)
                                    FROM (
                                    SELECT COUNT(MC.Campaign_ID) AS Campaigns_Count
                                    FROM Marketing_Strategist MS JOIN Marketing_Campaign MC ON
MS.MKTGID = MC.MKTGID
                                   GROUP BY MS.MKTGID) AS SubQuery
   );
```

Key Insights:

1. Top Contributor Identified: Highlights the employee who managed the most campaigns, showcasing exceptional involvement in marketing efforts.
2. Campaign Workload Analysis: Provides insights into workload distribution among employees, revealing potential over-reliance on key individuals.
3. Benchmark for Excellence: Establishes a benchmark for campaign management, setting performance standards for other strategists.
4. Recognition Opportunity: The identified employee can be acknowledged for their significant contribution, boosting morale and setting a standard for the team.



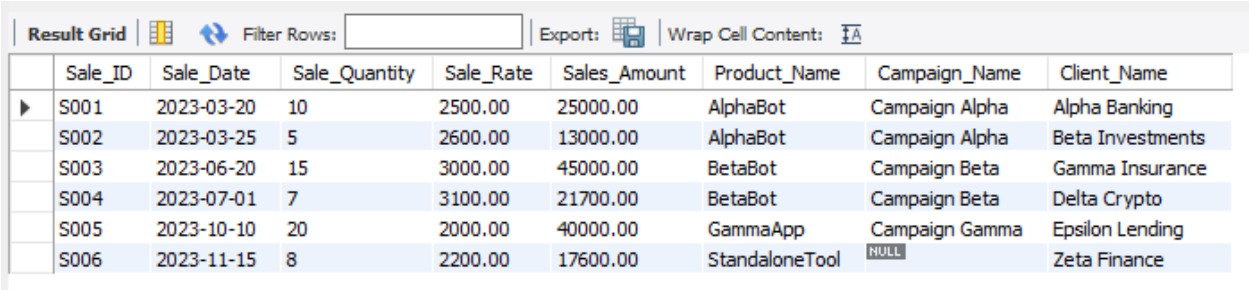
EmpName	Campaigns_Managed
Bob Smith	1
Emily Davis	1
Hannah Wilson	1

Query 5: Integrated Sales, Campaign, and Product Insights

```
SELECT S.Sale_ID, S.Sale_Date, S.Sale_Quantity, S.Sale_Rate, (S.Sale_Quantity * S.Sale_Rate) AS
Sales_Amount, P.Product_Name, MC.Campaign_Name, C.Client_Name
FROM Sales S JOIN Product P ON S.Product_ID = P.Product_ID LEFT JOIN Marketing_Campaign MC
ON P.Campaign_ID = MC.Campaign_ID JOIN Clients C ON S.Client_ID = C.Client_ID;
```

Key Insights:

1. Holistic Sales Data: Combines sales figures, product details, and campaign associations to offer a comprehensive view of revenue generation.
2. Sales Amount Calculation: Directly computes sales amounts for each transaction, simplifying financial analysis.
3. Client-Centric Insights: Links sales data to client names, enabling targeted client engagement strategies based on purchase patterns.
4. Campaign Impact Assessment: Associates sales with specific campaigns, providing valuable feedback on the effectiveness of marketing efforts.


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Sale_ID	Sale_Date	Sale_Quantity	Sale_Rate	Sales_Amount	Product_Name	Campaign_Name	Client_Name
S001	2023-03-20	10	2500.00	25000.00	AlphaBot	Campaign Alpha	Alpha Banking
S002	2023-03-25	5	2600.00	13000.00	AlphaBot	Campaign Alpha	Beta Investments
S003	2023-06-20	15	3000.00	45000.00	BetaBot	Campaign Beta	Gamma Insurance
S004	2023-07-01	7	3100.00	21700.00	BetaBot	Campaign Beta	Delta Crypto
S005	2023-10-10	20	2000.00	40000.00	GammaApp	Campaign Gamma	Epsilon Lending
S006	2023-11-15	8	2200.00	17600.00	StandaloneTool	NULL	Zeta Finance

---

Conclusion

In conclusion, the database system designed for Vanguard Industries demonstrates a robust and scalable architecture tailored to the company's operational and analytical needs. The structure aligns seamlessly with business goals, facilitating streamlined data management across entities like clients, employees, marketing campaigns, operations, products, and sales. By integrating logical relationships and enforcing domain constraints, the system ensures data integrity and consistency.

The queries implemented showcase the database's analytical capabilities, offering actionable insights. For instance, revenue-driven client and industry insights guide strategic decisions, while campaign and product performance queries identify areas of excellence and improvement. Employee workload and involvement analyses ensure optimal resource allocation, and integrated sales data connects client behavior with product and campaign effectiveness. These insights underscore the database's role as more than a storage tool—it’s an engine for innovation and strategy.

Furthermore, scalability and adaptability were core design principles, with data types and constraints prepared for future growth. The database supports both current business operations and advanced analytics, driving data-driven decision-making in a competitive AI/Fintech landscape. For a start-up, this organized and thoughtfully designed database lays the groundwork for long-term success, ensuring the company can adapt, innovate, and thrive in a dynamic environment.