MISM6213 - PROJECT 01
GROUP 10- Teja Padarthi, Rodrigo Silva, Shivani Avasarala.

NU Brewery – Final Report

Problem Statement

A brewery deals with a large amount of data, such as information on ingredients, production processes, inventory levels, sales data, and customer information. Managing this data effectively is essential for a brewery to operate efficiently and make informed decisions.
A database is crucial for a brewery company because it allows for the organization and management of data in a structured and efficient manner. For example, a brewery can use its database to manage inventory levels and ensure that it has enough raw materials to meet production demands without overstocking or wasting resources.
This can also be used to optimize production schedules and track sales data across multiple branches, allowing the brewery to analyze trends and make informed decisions about pricing, marketing, and distribution. Furthermore, a DBMS can help the brewery to manage customer information and personalize marketing efforts to increase customer loyalty and drive sales.
Overall, a database system is essential for a brewery company because it allows for the effective management and organization of large volumes of data, enabling the company to make informed decisions and operate efficiently. It ensures consistency, eliminates data redundancy, and provides efficient execution of queries and high-performance applications for the domain.

Functionality:

The Brewery Database is designed to store and manage information pertaining to diverse aspects of the brewery, including its various branches, employees, suppliers, and product offerings.

This database will enable users to retrieve required information easily and quickly through efficient queries from a well-designed and easily maintainable database.

Entities

1. *Employee*
2. *Customer*
3. *Orders*
4. *Branch*
5. *Raw Material*
6. *Supplier*
7. *Batch*
8. *Product*

Associative Entities:

9. *Served*
10. *Visit*
11. *Stock*
12. *Production*
13. *Bill*

Disjoint Entities

14. *Brewer*
15. *Server*

Relationship between Entities

Employee ⇄ Employee	*Employee ⇄ Brewer*	
Employee ⇄ Branch	*Branch ⇄ Visit*	
Served ⇄ Server	*Served ⇄ Customer*	
Customer ⇄ Visit	*Customer ⇄ Orders*	
Product ⇄ Batch	*Batch ⇄ Stock*	
Batch ⇄ Production	*Stock ⇄ Brewer*	
Production ⇄ Raw Material	*Raw Material ⇄ Bill*	
Bill ⇄ Supplier	*Employee ⇄ Server*	

Cardinalities of Relationships among Entities

Employee (Mandatory One) ⇄ Employee (Mandatory Many)	*Product (Mandatory One) ⇄ Batch (Mandatory Many)*
Employee (Mandatory One) ⇄ Brewer	*Batch (Mandatory One) ⇄ Stock (Mandatory Many)*
Employee (Mandatory Many) ⇄ Branch (Mandatory One)	*Batch (Mandatory One) ⇄ Production (Mandatory Many)*
Branch (Mandatory One) ⇄ Visit (Optional Many)	*Stock (Optional Many) ⇄ Brewer (Mandatory One)*
Served (Mandatory Many) ⇄ Server Mandatory One)	*Production (Optional Many) ⇄ Raw Material (Mandatory One)*
Served (Mandatory Many) ⇄ Customer (Mandatory One)	*Raw Material (Mandatory One) ⇄ Bill (Optional Many)*
Customer (Mandatory One) ⇄ Visit (Mandatory Many)	*Bill (Optional Many) ⇄ Supplier (Mandatory One)*
Customer (Mandatory One) ⇄ Orders (Mandatory Many)	*Employee (Mandatory One) ⇄ Server*

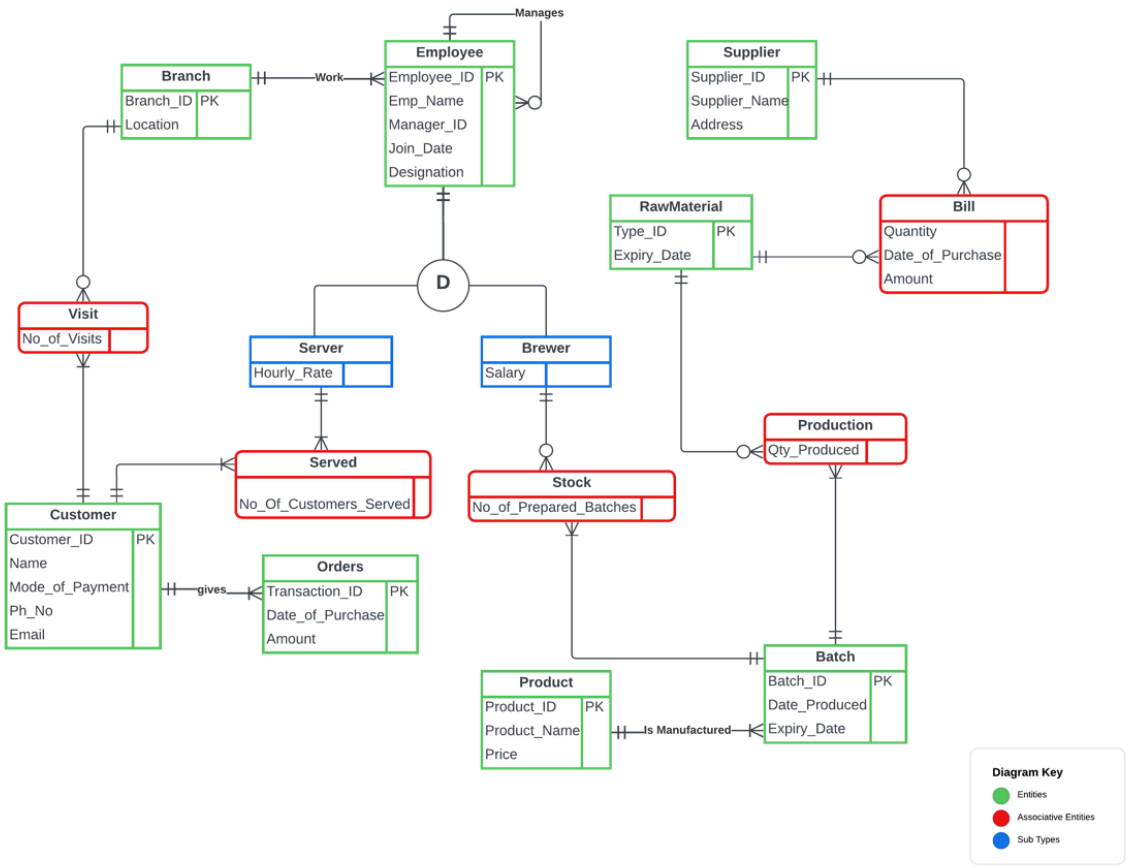
MISM6213 - PROJECT 01
GROUP 10- Teja Padarthi, Rodrigo Silva, Shivani Avasarala.

Attributes of All Entities

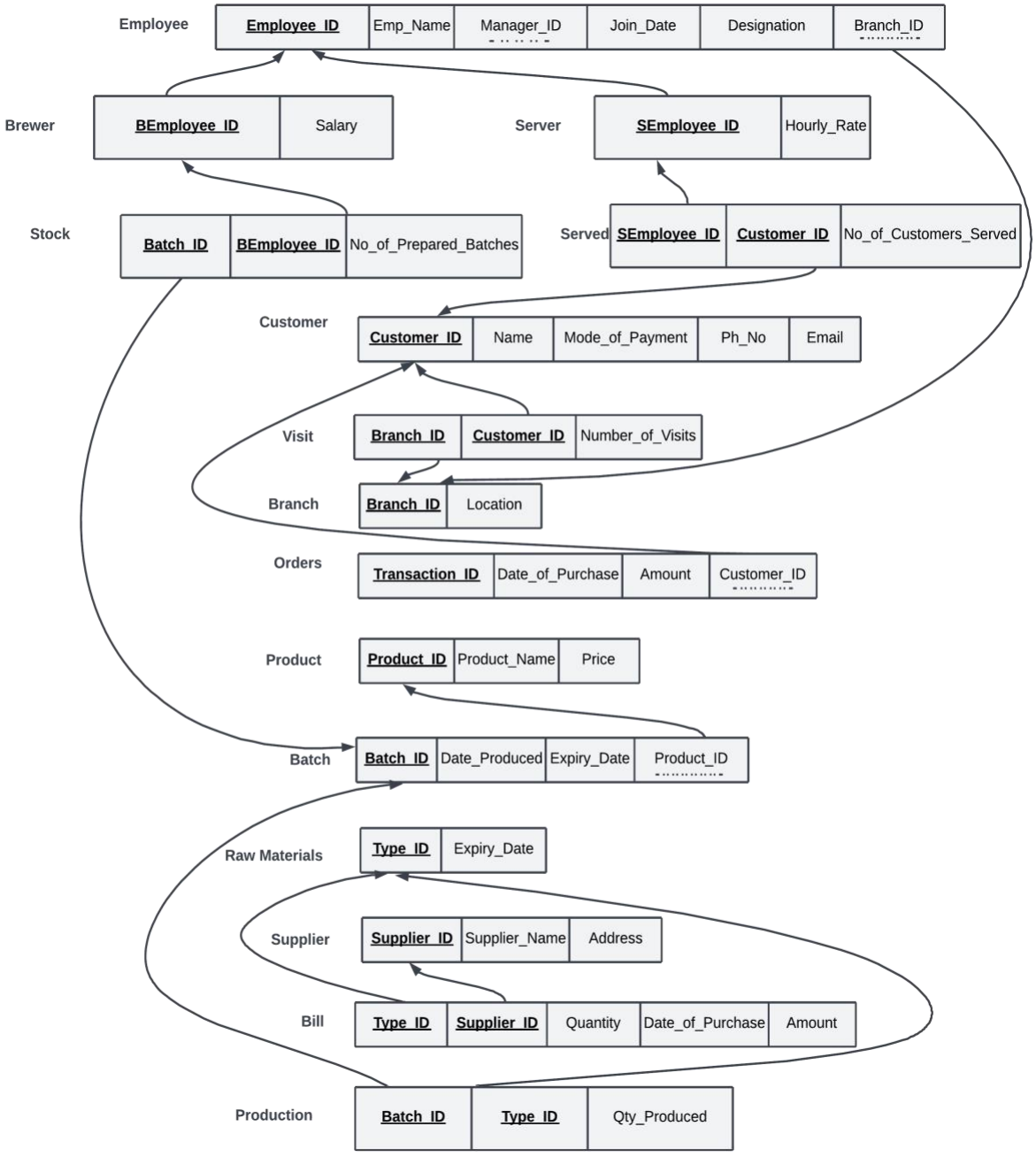
Below are the attributes, we as a group thought it is important to get an overall picture for running a Brewery and get good insights.

Table	**Attributes**
Employee	*Employee ID, Emp Name, Manager ID, Join Date, Designation (Discriminator)*
Server	*B_Employee ID, Hourly Rate*
Brewer	*S_Employee ID, Salary*
Branch	*Branch ID, Location*
Customer	*Customer ID, Name, Mode of Payment, Ph. No, Email*
Orders	*Transaction ID, Date of Purchase, Amount*
Product	*Product ID, Product Name, Price*
Batch	*Batch ID, Date Produced, Expiry Date*
Raw Material	*Type ID, Expiry Date*
Supplier	*Supplier ID, Supplier Name, Address*
Production	*Qty Produced, Qty of Raw Materials Used*
Bill	*Quantity, Date of Purchase, Amount*
Stock	*No. of Prepared Batches*
Served	*Customers Served*
Visit	*No. of Visits*

Entity Relationship Diagram



Relational Schema of NU Brewery

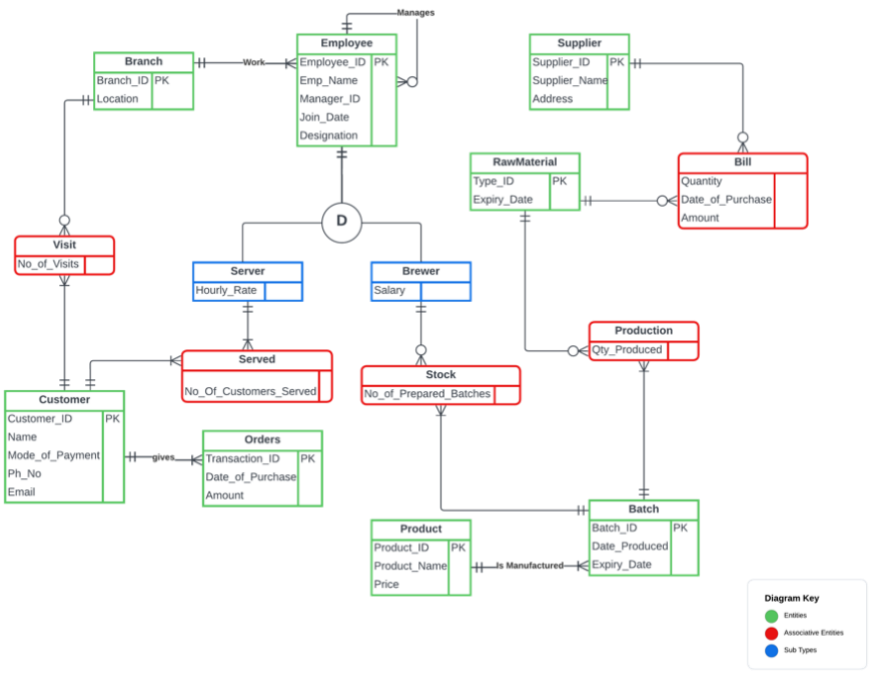


MISM6213 - PROJECT 01
GROUP 10- Teja Padarthi, Rodrigo Silva, Shivani Avasarala.

Data Normalization

Partial Dependencies existed which were removed and converted into 3NF.

Brewery	
Employee_ID	
Customer_ID	
Branch_ID	
Supplier_ID	**PK**
Transaction_ID	
Type_ID	
Product_ID	
Emp_Name	
Join_Date	
Designation	
Salary	
Hourly_Rate	
Manager_ID	
Location	
Supplier_Name	
Address	
Name	
Mode_of_Payment	
Ph_No	
Email	
Date_of_Purchase (Orders)	
Amount (Orders)	
Product_Name	
Price	
Date_Produced	
Expiry_Date (Batch)	
Expiry_Date (Raw Material)	
Amount (Bill)	
Date_of_Purchase (Bill)	
Quantity	
No_of_Visits	
No_of_Customers_Served	
No_of_Prepared_Batches	
Qty_Produced	



Summary Table for each entity

Employee Table

Employee (Employee_ID, Emp_Name, Manager_ID, Join_Date, Designation, Branch_ID)
Data Type: VARCHAR (6), VARCHAR (20), VARCHAR (6), DATE, VARCHAR (25) respectively.
Additional Details: Employee_ID is Unique, and all fields are required. Manager_ID & Branch_ID are foreign keys.

Customer Table

Customer (Customer_ID, Name, Mode_of_Payment, Ph_No, Email)

Data Type: VARCHAR (6), VARCHAR (20), VARCHAR (6), INTEGER (10), VARCHAR (30) respectively

Additional Details: Customer_ID is Unique.

Supplier Table

Supplier (Supplier_ID, Supplier_Name, Address)

Data Type: VARCHAR (6), VARCHAR (20), VARCHAR (50) respectively

Additional Details: Supplier_ID is Unique, and all fields are required.

Orders Table

Orders (Transaction_ID, Date_of_Purchase, Amount, Customer_ID)

Data Type: VARCHAR (10), DATE, INTEGER (5), VARCHAR (6) respectively

Additional Details: Transaction_ID is Unique, and all fields are required. Customer_ID is a foreign key.

Branch Table

Branch (Branch_ID, Location)
Data Type: VARCHAR (5), VARCHAR (10) respectively.
Additional Details: Branch_ID is Unique, and all fields are required.

Product Table

Product (Product_ID, Product_Name, Price)
Data Type: VARCHAR (5), VARCHAR (20), INTEGER (4) respectively.
Additional Details: Product_ID is Unique, and all fields are required.

Batch Table

Batch (Batch_ID, Date_Produced, Expiry_Date, Product_ID)
Data Type: VARCHAR (7), DATE, DATE, VARCHAR (5) respectively.
Additional Details: Batch_ID is Unique, and all fields are required. Product_ID is a foreign key.

Raw Material Table

Raw Material (Type_ID, Expiry_Date)
Data Type: VARCHAR (5), DATE respectively.
Additional Details: Type_ID is Unique, and all fields are required.

Production Table

Production (Qty_Produced, Batch_ID, Type_ID)
Data Type: INTEGER (5), VARCHAR (7), VARCHAR (5) respectively.
Additional Details: Batch_ID & Type_ID are Unique, and all fields are required. Batch_ID & Type_ID are foreign keys.

Bill Table

Bill (Quantity, Date_of_Purchase, Amount, Type_ID, Supplier_ID)
Data Type: INTEGER (5), DATE, INTEGER (5), VARCHAR (5), VARCHAR (6)
Additional Details: Type_ID & Supplier_ID together are Unique, and all fields are required. Type_ID & Supplier_ID are foreign keys.

Stock Table

Stock (No_of_Prepared_Batches, Batch_ID, BEmployee_ID)
Data Type: INTEGER (3), VARCHAR (7), VARCHAR (6)
Additional Details: Batch_ID & BEmployee_ID together are Unique, and all fields are required. Batch_ID & BEmployee_ID are foreign keys.

Served Table

Served (No_of_Customers_Served, Customer_ID, SEmployee_ID)
Data Type: INTEGER (3), VARCHAR (6), VARCHAR (6)
Additional Details: Customer_ID & SEmployee_ID together are Unique, and all fields are required. Customer_ID & SEmployee_ID are foreign keys.

Visit Table

Visit (Number_of_Visits, Customer_ID, Branch_ID)
Data Type: INTEGER (3), VARCHAR (6), VARCHAR (5)
Additional Details: Customer_ID & Branch_ID together are Unique, and all fields are required. Customer_ID & Branch_ID are foreign key

Brewer Table

Brewer (BEmployee_ID, Salary)
Data Type: VARCHAR (6), INTEGER (8)
Additional Details: BEmployee_ID is Unique, and all fields are required. BEmployee_ID is a foreign key.

Server Table

Server (SEmployee_ID, Hourly_Rate)
Data Type: VARCHAR (6), INTEGER (3)
Additional Details: SEmployee_ID is Unique, and all fields are required. SEmployee_ID is a foreign key.

Creation of Tables

Employee Table

Employee (Employee_ID, Emp_Name, Manager_ID, Join_Date, Designation, Branch_ID)
Data Type: VARCHAR (6), VARCHAR (20), VARCHAR (6), DATE, VARCHAR (25) respectively
Additional Details: Employee_ID is Unique, and all fields are required. Manager_ID & Branch_ID are foreign keys.

```
DROP TABLE IF EXISTS Employee;
 CREATE TABLE Employee
(
Employee_ID VARCHAR (6) NOT NULL,
Emp_Name VARCHAR (20) NOT NULL,
 Join_Date DATE NOT NULL,
 Designation VARCHAR (25) NOT NULL,
 Branch_ID VARCHAR (5) NOT NULL,
 Manager_ID VARCHAR (6),
 CONSTRAINT Employee_ID PRIMARY KEY
(Employee_ID),
 FOREIGN KEY (Branch_ID)
 REFERENCES Branch(Branch_ID),
```

Brewer Table

Brewer (BEmployee_ID, Salary)
Data Type: VARCHAR (6), INTEGER (8)
Additional Details: BEmployee_ID is Unique, and all fields are required. BEmployee_ID is a foreign key.

```
DROP TABLE IF EXISTS Brewer;
 CREATE TABLE Brewer
(
BEmployee_ID VARCHAR (6) NOT NULL,
Salary INT (8) NOT NULL,
 CONSTRAINT BEmployee_ID PRIMARY KEY
(BEmployee_ID),
 FOREIGN KEY (BEmployee_ID)
 REFERENCES Employee (Employee_ID)
 );
```

```
 FOREIGN KEY (Manager_ID)
 REFERENCES Employee (Employee_ID)
);
```

Server Table

Server (SEmployee_ID, Hourly_Rate)
Data Type: VARCHAR (6), INTEGER (3)
Additional Details: SEmployee_ID is Unique, and all fields are required. SEmployee_ID is a foreign key.

```
DROP TABLE IF EXISTS Server;
 CREATE TABLE Server
(
SEmployee_ID VARCHAR (6) NOT NULL,
Hourly_Rate INT (3) NOT NULL,
 CONSTRAINT SEmployee_ID PRIMARY KEY
(SEmployee_ID),
 FOREIGN KEY (SEmployee_ID)
REFERENCES Employee (Employee_ID)
 );
```

Customer Table

Customer (Customer_ID, Name, Mode_of_Payment, Ph_No, Email)
Data Type: VARCHAR (6), VARCHAR (20), VARCHAR (6), INTEGER (10), VARCHAR (30) respectively
Additional Details: Customer_ID is Unique.

```
DROP TABLE IF EXISTS Customer;
 CREATE TABLE Customer
(
Customer_ID VARCHAR (6) NOT NULL,
Name VARCHAR (20) NOT NULL,
 Mode_of_Payment VARCHAR (6) NOT NULL,
 Ph_No VARCHAR (10) NOT NULL,
Email VARCHAR (30),
 CONSTRAINT Customer_ID PRIMARY KEY
(Customer_ID)
 );
```

Supplier Table

Supplier (Supplier_ID, Supplier_Name, Address)
Data Type: VARCHAR (6), VARCHAR (20), VARCHAR (50) respectively
Additional Details: Supplier_ID is Unique, and all fields are required.

```
DROP TABLE IF EXISTS Supplier;
 CREATE TABLE Supplier
(
Supplier_ID VARCHAR (6) NOT NULL,
Supplier_Name VARCHAR (20) NOT NULL,
 Address VARCHAR (50) NOT NULL,
 CONSTRAINT Supplier_ID PRIMARY KEY
(Supplier_ID)
 );
```

Branch Table

Branch (Branch_ID, Location)
Data Type: VARCHAR (5), VARCHAR (10) respectively.
Additional Details: Branch_ID is Unique, and all fields are required.

```
DROP TABLE IF EXISTS Branch;
 CREATE TABLE Branch
(
Branch_ID VARCHAR (5) NOT NULL,
Location VARCHAR (10) NOT NULL,
 CONSTRAINT Branch_ID PRIMARY KEY
(Branch_ID)
 );
```

Product Table

Product (Product_ID, Product_Name, Price)
Data Type: VARCHAR (5), VARCHAR (20), INTEGER (4) respectively.
Additional Details: Product_ID is Unique, and all fields are required.

```
DROP TABLE IF EXISTS Product;
 CREATE TABLE Product
(
Product_ID VARCHAR (5) NOT NULL,
Product_Name VARCHAR (20) NOT NULL,
 Price INT (4) NOT NULL,
 CONSTRAINT Product_ID PRIMARY KEY
(Product_ID)
 );
```

Batch Table

Batch (Batch_ID, Date_Produced, Expiry_Date, Product_ID)
Data Type: VARCHAR (7), DATE, DATE, VARCHAR (5) respectively.
Additional Details: Batch_ID is Unique, and all fields are required. Product_ID is a foreign key.

```
DROP TABLE IF EXISTS Batch;
 CREATE TABLE Batch (
     Batch_ID VARCHAR (7) NOT NULL,
     Date_Produced DATE NOT NULL,
     Expiry_Date DATE NOT NULL,
     Product_ID VARCHAR (5) NOT NULL,
     CONSTRAINT Batch_ID PRIMARY KEY
(Batch_ID),
     FOREIGN KEY (Product_ID)
         REFERENCES Product (Product_ID)
 );
```

Raw Material Table

Raw Material (Type_ID, Expiry_Date)
Data Type: VARCHAR (5), DATE respectively.
Additional Details: Type_ID is Unique, and all fields are required.

```
DROP TABLE IF EXISTS RawMaterial;
 CREATE TABLE RawMaterial (
     Type_ID VARCHAR (5) NOT NULL,
     Expiry_Date DATE NOT NULL,
     CONSTRAINT Type_ID PRIMARY KEY
(Type_ID)
 );
```

Production Table

Production (Qty_Produced, Batch_ID, Type_ID)
Data Type: INTEGER (5), VARCHAR (7), VARCHAR (5) respectively.
Additional Details: Batch_ID & Type_ID are Unique, and all fields are required. Batch_ID & Type_ID are foreign keys.

```
DROP TABLE IF EXISTS Production;
 CREATE TABLE Production (
     Batch_ID VARCHAR (7) NOT NULL,
     Type_ID VARCHAR (5) NOT NULL,
     Qty_Produced INT (5) NOT NULL,
     CONSTRAINT BT_ID PRIMARY KEY
(Batch_ID, Type_ID),
     FOREIGN KEY (Batch_ID)
         REFERENCES Batch (Batch_ID),
    FOREIGN KEY (Type_ID)
         REFERENCES RawMaterial (Type_ID)
 );
```

Bill Table

Bill (Quantity, Date_of_Purchase, Amount, Type_ID, Supplier_ID)
Data Type: INTEGER (5), DATE, INTEGER (5), VARCHAR (5), VARCHAR (6)
Additional Details: Type_ID & Supplier_ID together are Unique, and all fields are required. Type_ID & Supplier_ID are foreign keys.

```
DROP TABLE IF EXISTS Bill;
CREATE TABLE Bill (
     Date_Of_Purchase DATE NOT NULL,
     Quantity INT(5) NOT NULL,
     Amount INT(5) NOT NULL,
     Type_ID VARCHAR(5) NOT NULL,
     Supplier_ID VARCHAR(6) NOT NULL,
     CONSTRAINT TS_ID PRIMARY KEY
(Type_ID , Supplier_ID),
     FOREIGN KEY (Type_ID)
         REFERENCES RawMaterial
(Type_ID),
     FOREIGN KEY (Supplier_ID)
         REFERENCES Supplier
(Supplier_ID)
 );
```

Stock Table

Stock (No_of_Prepared_Batches, Batch_ID, BEmployee_ID)
Data Type: INTEGER (3), VARCHAR (7), VARCHAR (6)
Additional Details: Batch_ID & BEmployee_ID together are Unique, and all fields are required. Batch_ID & BEmployee_ID are foreign keys.

```
DROP TABLE IF EXISTS Stock;
CREATE TABLE Stock (
     No_Of_Prepared_Batches INT(3) NOT
NULL,
     Batch_ID VARCHAR(7) NOT NULL,
     BEmployee_ID VARCHAR(6) NOT NULL,
     CONSTRAINT Stock PRIMARY KEY
(Batch_ID , BEmployee_ID),
     FOREIGN KEY (Batch_ID)
         REFERENCES Batch (Batch_ID),
     FOREIGN KEY (BEmployee_ID)
         REFERENCES Brewer (BEmployee_ID)
 );
```

Served Table

Served (No_of_Customers_Served, Customer_ID, SEmployee_ID)
Data Type: INTEGER (3), VARCHAR (6), VARCHAR (6)
Additional Details: Customer_ID & SEmployee_ID together are Unique, and all fields are required. Customer_ID & SEmployee_ID are foreign keys.

Visit Table

Visit (Number_of_Visits, Customer_ID, Branch_ID)
Data Type: INTEGER (3), VARCHAR (6), VARCHAR (5)
Additional Details: Customer_ID & Branch_ID together are Unique, and all fields are required. Customer_ID & Branch_ID are foreign keys.

```
DROP TABLE IF EXISTS Served;
 CREATE TABLE Served (
     No_Of_Customers_Served INT(3) NOT
NULL,
     SEmployee_ID VARCHAR(12) NOT NULL,
     Customer_ID VARCHAR(6) NOT NULL,
     CONSTRAINT SC_ID PRIMARY KEY
(SEmployee_ID , Customer_ID),
     FOREIGN KEY (SEmployee_ID)
         REFERENCES Server
(SEmployee_ID),
     FOREIGN KEY (Customer_ID)
         REFERENCES Customer
(Customer_ID)
 );
```

```
DROP TABLE IF EXISTS Visit;
 CREATE TABLE Visit (
     Branch_ID VARCHAR(5) NOT NULL,
     Customer_ID VARCHAR(6) NOT NULL,
     No_Of_Visits INT(3) NOT NULL,
     CONSTRAINT BC_ID PRIMARY KEY
(Branch_ID , Customer_ID),
     FOREIGN KEY (Branch_ID)
         REFERENCES Branch (Branch_ID),
     FOREIGN KEY (Customer_ID)
         REFERENCES Customer
(Customer_ID)
 );
```

Orders Table

Orders (Transaction_ID, Date_of_Purchase, Amount, Customer_ID)
Data Type: VARCHAR (10), DATE, INTEGER (5), VARCHAR (6) respectively
Additional Details: Transaction_ID is Unique, and all fields are required. Customer_ID is a foreign key.

```
DROP TABLE IF EXISTS Orders;
CREATE TABLE Orders (
    Transaction_ID VARCHAR (10) NOT NULL,
    Date_Of_Purchase DATE NOT NULL,
    Amount INT (5) NOT NULL,
    Customer_ID VARCHAR (6) NOT NULL,
    CONSTRAINT Transaction_ID PRIMARY KEY
(Transaction_ID),
    FOREIGN KEY (Customer_ID)
        REFERENCES Customer (Customer_ID)
);
```

Insertion of Data in Tables

Employee Table

```
INSERT INTO Employee (Employee_ID,
Emp_Name, Manager_ID, Join_Date,
Designation, Branch_ID)
 VALUES
   ('E00001', 'John Doe', NULL, '2020-
01-01', 'Manager', 'B00001'),
   ('E00002', 'Jane Smith', 'E00001',
'2020-01-01', 'Assistant Manager',
'B00001'),
   ('E00003', 'David Lee', 'E00001',
'2020-02-01', 'Supervisor', 'B00001'),
   ('E00004', 'Amy Johnson', 'E00003',
'2020-02-01', 'Team Lead', 'B00002'),
   ('E00005', 'Sarah Kim', 'E00003',
```

Customer Table

```
INSERT INTO Customer (Customer_ID, Name,
Mode_Of_Payment, Ph_No, Email)
 VALUES
   ('C00001', 'Emily Chen', 'Cash',
'2345678901', 'emily.chen@example.com'),
   ('C00002', 'Jake Kim', 'Credit',
'3456789012', 'jake.kim@example.com'),
   ('C00003', 'Grace Park', 'Cash',
'4567890123', 'grace.park@example.com'),
   ('C00004', 'Lucy Lee', 'Credit',
'5678901234', 'lucy.lee@example.com'),
   ('C00005', 'Michael Smith', 'Cash',
'6789012345',
'michael.smith@example.com'),
```

```
'2020-03-01', 'Team Lead', 'B00002'),
   ('E00006', 'Adam Park', 'E00002',
'2020-04-01', 'Server', 'B00001'),
   ('E00007', 'Peter Chen', 'E00002',
'2020-04-01', 'Server', 'B00001'),
   ('E00008', 'Mary Wong', 'E00003',
'2020-05-01', 'Server', 'B00002'),
   ('E00009', 'Linda Wang', 'E00003',
'2020-05-01', 'Brewer', 'B00002'),
   ('E00010', 'Tom Chang', 'E00002',
'2020-06-01', 'Brewer', 'B00001');
```

```
   ('C00006', 'Olivia Johnson',
'Credit', '7890123456',
'olivia.johnson@example.com'),
   ('C00007', 'Henry Wong', 'Cash',
'8901234567', 'henry.wong@example.com'),
   ('C00008', 'Ava Wang', 'Credit',
'9012345678', 'ava.wang@example.com'),
   ('C00009', 'Ethan Liu', 'Cash',
'1234567890', 'ethan.liu@example.com'),
   ('C00010', 'Sophia Chang', 'Credit',
'2345678901',
'sophia.chang@example.com');
```

Supplier Table

```
INSERT INTO Supplier (Supplier_ID,
Supplier_Name, Address)
 VALUES
 ('S00001', 'ABC Company', '123 Main
St'),
   ('S00002', 'XYZ Corporation', '456
Elm St'),
   ('S00003', 'Acme Inc.', '789 Oak
St'),
   ('S00004', 'Smith & Sons', '1010
Maple Ave'),
   ('S00005', 'Jones Industries', '1111
Cedar Blvd'),
   ('S00006', 'Globex Corporation',
'1313 Mockingbird Ln'),
   ('S00007', 'Initech', '555 Office
Park Dr'),
   ('S00008', 'Vandelay Industries',
'2468 Broadway'),
   ('S00009', 'Stark Industries', '10880
Malibu Point'),
   ('S00010', 'Wayne Enterprises', '1007
Mountain Dr');
```

Orders Table

```
INSERT INTO Orders (Transaction_ID,
Date_of_Purchase, Amount, Customer_ID)
 VALUES
   ('T000001', '2022-01-01', 50,
'C00001'),
   ('T000002', '2022-01-02', 75,
'C00002'),
   ('T000003', '2022-01-03', 100,
'C00003'),
   ('T000004', '2022-01-04', 125,
'C00004'),
   ('T000005', '2022-01-05', 150,
'C00005'),
   ('T000006', '2022-01-06', 175,
'C00006'),
   ('T000007', '2022-01-07', 200,
'C00007'),
   ('T000008', '2022-01-08', 225,
'C00008'),
   ('T000009', '2022-01-09', 250,
'C00009'),
   ('T000010', '2022-01-10', 275,
'C00002'),
   ('T000011', '2022-01-11', 375,
'C00010'),
   ('T000012', '2022-01-12', 515,
'C00005'),
   ('T000013', '2022-01-13', 200,
'C00005'),
   ('T000014', '2022-01-14', 750,
'C00001'),
    ('T000015', '2022-01-15', 398,
'C00004');
```

Branch Table

```
INSERT INTO Branch (Branch_ID, Location)
 VALUES
  ('B0001', 'Boston'),
  ('B0002', 'London');
```

Product Table

```
  INSERT INTO Product (Product_ID,
Product_Name, Price)
VALUES
('P0001', 'IPA', 15),
('P0002', 'Lager', 14),
('P0003', 'Stout', 16),
('P0004', 'Wheat Beer', 15),
('P0005', 'Pilsner', 12),
('P0006', 'Brown Ale', 20),
```

```
('P0007', 'Porter', 19),
('P0008', 'Belgian Tripel', 10),
('P0009', 'Hefeweizen', 13),
('P0010', 'Sour Ale', 17);
```

Batch Table

```
INSERT INTO Batch (Batch_ID,
Date_Produced, Expiry_Date, Product_ID)
VALUES
('B001001', '2022-01-01', '2023-01-01',
'P0010'),
('B001002', '2022-01-01', '2023-01-01',
'P0001'),
('B002003', '2022-01-01', '2023-02-01',
'P0002'),
('B002004', '2022-01-01', '2022-07-01',
'P0006'),
('B001005', '2022-01-01', '2023-01-01',
'P0003'),
('B002006', '2022-01-01', '2023-01-01',
'P0007'),
('B001007', '2022-01-01', '2022-09-31',
'P0004'),
('B002008', '2022-01-01', '2023-01-01',
'P0008'),
('B001009', '2022-01-01', '2023-01-01',
'P0005'),
('B002010', '2022-01-01', '2023-01-01',
'P0009');
```

Raw Material Table

```
INSERT INTO RawMaterial (Type_ID,
Expiry_Date)
VALUES
('RM001', '2023-05-20'),
('RM002', '2023-07-15'),
('RM003', '2023-06-30'),
('RM004', '2023-08-01'),
('RM005', '2023-07-05'),
('RM006', '2023-09-10'),
('RM007', '2023-07-30'),
('RM008', '2023-08-25'),
('RM009', '2023-09-20'),
('RM010', '2023-10-15');
```

Brewer Table

```
INSERT INTO Brewer (BEmployee_ID,
Salary)
 VALUES
('E00009', 50000),
 ('E00010', 60000);
```

Server Table

```
INSERT INTO Server (SEmployee_ID,
Hourly_Rate)
 VALUES
('E00006', 30),
 ('E00008',25),
 ('E00007', 20);
```

Production Table

```
INSERT INTO Production (Qty_Produced,
Batch_ID, Type_ID)

VALUES
(1000, 'B001001', 'RM001'),
 (2000, 'B001002', 'RM002'),
 (1500, 'B002003', 'RM003'),
 (3000, 'B002004', 'RM004'),
 (2500, 'B001005', 'RM005'),
 (500, 'B002006', 'RM006'),
 (800, 'B001007', 'RM007'),
 (1200, 'B002008', 'RM008');
```

Bill Table

```
INSERT INTO Bill (Quantity,
Date_of_Purchase, Amount, Type_ID,
Supplier_ID)
 VALUES
 (1000, '2021-09-15', 2000,
'RM001','S00001'),
 (715, '2022-02-10', 1500, 'RM002',
'S00001'),
 (250, '2022-02-12', 400, 'RM001',
'S00002'),
 (320, '2022-03-21', 800, 'RM002',
'S00003'),
 (1570, '2022-03-01', 3200, 'RM001',
'S00010');
```

Served Table

```
INSERT INTO Served
(No_of_Customers_Served, Customer_ID,
SEmployee_ID)
 VALUES
 (3,'C00001', 'E00006'),
 (2,'C00002', 'E00006'),
 (4,'C00003', 'E00007'),
 (1,'C00004', 'E00007'),
 (2,'C00005', 'E00008'),
 (2,'C00006', 'E00008'),
 (6,'C00006', 'E00006');
```

Visit Table

```
INSERT INTO Visit (No_of_Visits,
Customer_ID, Branch_ID)
 VALUES
 (4, 'C00001', 'B0001'),
 (2, 'C0002', 'B0002'),
 (2, 'C0003', 'B0002'),
 (2, 'C0004', 'B0001'),
 (2, 'C0005', 'B0002'),
 (2, 'C0006', 'B0001'),
 (2, 'C0006', 'B0002');
```

Stock Table

```
INSERT INTO Stock
(No_of_Prepared_Batches, Batch_ID,
BEmployee_ID)
 VALUES
 (1, 'B001001', 'E00009'),
 (2, 'B001002', ' E00009'),
 (1, 'B002003', ' E00010'),
 (3, 'B002004', ' E00010');
```

Data loaded into Database

```
1 •  SELECT * FROM Project.Visit;
```

100% 1:1

Result Grid Filter Rows: Search

Branch_ID	Customer_ID	No_Of_Visits
B0001	C00001	4
B0001	C00004	2
B0001	C00006	2
B0002	C00002	2
B0002	C00003	2
B0002	C00005	2
B0002	C00006	2

```
1 •  SELECT * FROM Project.Supplier;
```

100% 1:1

Result Grid Filter Rows: Search

Supplier_ID	Supplier_Name	Address
S00001	ABC Company	123 Main St
S00002	XYZ Corporation	456 Elm St
S00003	Acme Inc.	789 Oak St
S00004	Smith & Sons	1010 Maple Ave
S00005	Jones Industries	1111 Cedar Blvd
S00006	Globex Corporation	1313 Mockingbird Ln
S00007	Initech	555 Office Park Dr
S00008	Vandelay Industries	2468 Broadway
S00009	Stark Industries	10880 Malibu Point
S00010	Wayne Enterprises	1007 Mountain Dr

```
1 •  SELECT * FROM Project.Stock;
```

100% 1:1

Result Grid Filter Rows: Search

No_Of_Prepared_Batch...	Batch_ID	BEmployee_ID
1	B001001	E00009
2	B001002	E00009
1	B002003	E00010
3	B002004	E00010

```
1 •  SELECT * FROM Project.Server;
```

100% 1:1

Result Grid Filter Rows: Search

SEmployee_ID	Hourly_Rate
E00006	30
E00007	20
E00008	25

```
1 •  SELECT * FROM Project.Served;
```

100% 1:1

Result Grid Filter Rows: Search

No_Of_Customers_Serv...	SEmployee_ID	Customer_ID
3	E00006	C00001
2	E00006	C00002
6	E00006	C00006
4	E00007	C00003
1	E00007	C00004
2	E00008	C00005
2	E00008	C00006

```
1 •  SELECT * FROM Project.RawMaterial;
```

100% 1:1

Result Grid Filter Rows: Search

Type_ID	Expiry_Date
RM001	2023-05-20
RM002	2023-07-15
RM003	2023-06-30
RM004	2023-08-01
RM005	2023-07-05
RM006	2023-09-10
RM007	2023-07-30
RM008	2023-08-25
RM009	2023-09-20
RM010	2023-10-15

```
1 • SELECT * FROM Project.Production;
```

100% 1:1

Result Grid Filter Rows: Search

Batch_ID	Type_ID	Qty_Produced
B001001	RM001	1000
B001002	RM002	2000
B001005	RM005	2500
B001007	RM007	800
B002003	RM003	1500
B002004	RM004	3000
B002006	RM006	500
B002008	RM008	1200

```
1 • SELECT * FROM Project.Product;
```

100% 1:1

Result Grid Filter Rows: Search

Product_ID	Product_Name	Price
P0001	IPA	15
P0002	Lager	14
P0003	Stout	16
P0004	Wheat Beer	15
P0005	Pilsner	12
P0006	Brown Ale	20
P0007	Porter	19
P0008	Belgian Tripel	10
P0009	Hefeweizen	13
P0010	Sour Ale	17
NULL	NULL	NULL

```
1 • SELECT * FROM Project.Orders;
```

100% 1:1

Result Grid Filter Rows: Search

Transaction_ID	Date_Of_Purchase	Amount	Customer_ID
T000001	2022-01-01	50	C00001
T000002	2022-01-02	75	C00002
T000003	2022-01-03	100	C00003
T000004	2022-01-04	125	C00004
T000005	2022-01-05	150	C00005
T000006	2022-01-06	175	C00006
T000007	2022-01-07	200	C00007
T000008	2022-01-08	225	C00008
T000009	2022-01-09	250	C00009
T000010	2022-01-10	275	C00002
T000011	2022-01-11	375	C00010
T000012	2022-01-12	515	C00005
T000013	2022-01-13	200	C00005
T000014	2022-01-14	750	C00001
T000015	2022-01-15	398	C00004

```
1 • SELECT * FROM Project.Batch;
```

100% 29:1

Result Grid Filter Rows: Search

Batch_ID	Date_Produced	Expiry_Date	Product_ID
B001001	2022-01-01	2023-01-01	P0010
B001002	2022-01-01	2023-01-01	P0001
B001005	2022-01-01	2023-01-01	P0003
B001007	2022-01-01	2022-09-30	P0004
B001009	2022-01-01	2023-01-01	P0005
B002003	2022-01-01	2023-02-01	P0002
B002004	2022-01-01	2022-07-01	P0006
B002006	2022-01-01	2023-01-01	P0007
B002008	2022-01-01	2023-01-01	P0008
B002010	2022-01-01	2023-01-01	P0009
NULL	NULL	NULL	NULL

```
1 • SELECT * FROM Project.Customer;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/

Customer_ID	Name	Mode_of_Payment	Ph_No	Email
C00001	Emily Chen	Cash	2345678901	emily.chen@example.com
C00002	Jake Kim	Credit	3456789012	jake.kim@example.com
C00003	Grace Park	Cash	4567890123	grace.park@example.com
C00004	Lucy Lee	Credit	5678901234	lucy.lee@example.com
C00005	Michael Smith	Cash	6789012345	michael.smith@example.com
C00006	Olivia Johnson	Credit	7890123456	olivia.johnson@example.com
C00007	Henry Wong	Cash	8901234567	henry.wong@example.com
C00008	Ava Wang	Credit	9012345678	ava.wang@example.com
C00009	Ethan Liu	Cash	1234567890	ethan.liu@example.com
C00010	Sophia Chang	Credit	2345678901	sophia.chang@example.com

```
1 • SELECT * FROM Project.Brewer;
```

100% 1:1

Result Grid Filter Rows: Search

BEmployee_ID	Salary
E00009	50000
E00010	60000

```
1 • SELECT * FROM Project.Employee;
```

100% 1:1

Result Grid Filter Rows: Search Edit:

Employee_ID	Emp_Name	Join_Date	Designation	Branch_ID	Manager_ID
E00001	John Doe	2020-01-01	Manager	B0001	NULL
E00002	Jane Smith	2020-01-01	Assistant Manager	B0001	E00001
E00003	David Lee	2020-02-01	Supervisor	B0001	E00001
E00004	Amy Johnson	2020-02-01	Team Lead	B0002	E00003
E00005	Sarah Kim	2020-03-01	Team Lead	B0002	E00003
E00006	Adam Park	2020-04-01	Server	B0001	E00002
E00007	Peter Chen	2020-04-01	Server	B0001	E00002
E00008	Mary Wong	2020-05-01	Server	B0002	E00003
E00009	Linda Wang	2020-05-01	Brewer	B0002	E00003
E00010	Tom Chang	2020-06-01	Brewer	B0001	E00002

```
1 • SELECT * FROM Project.Branch;
```

100% 1:1

Result Grid Filter Rows: Search

Branch_ID	Location
B0001	Boston
B0002	London

```
1 • SELECT * FROM Project.Bill;
```

100% 1:1

Result Grid Filter Rows: Search

Date_Of_Purchase	Quantity	Amount	Type_ID	Supplier_ID
2021-09-15	1000	2000	RM001	S00001
2022-02-12	250	400	RM001	S00002
2022-03-01	1570	3200	RM001	S00010
2022-02-10	715	1500	RM002	S00001
2022-03-21	320	800	RM002	S00003

Querying in the Database

1. ***Which Branch of NU Brewery has the most visited customers?***

```
298 •   SELECT Branch_ID, COUNT(Customer_ID) AS Total_Visits
299     FROM Visit
300     GROUP BY Branch_ID;
301
```

100% 20:300

Result Grid | Filter Rows: Search | Export:

Branch_ID	Total_Visits
B0001	3
B0002	4

```
302 •   SELECT Location
303     FROM Branch
304     WHERE Branch_ID = 'B0002'
```

100% 26:304

Result Grid | Filter Rows: Searcl

Location
London

London Branch has the most visited customers.

2. ***Get the names of Suppliers and the Total Amount where Quantity ordered is more than 500 from each.***

```
311 •   SELECT s.Supplier_ID, s.Supplier_Name, SUM(b.Amount) AS Total_Amount
312     FROM Bill b
313     JOIN Supplier s ON b.Supplier_ID = s.Supplier_ID
314     WHERE b.Quantity > 500
315     GROUP BY b.Supplier_ID, s.Supplier_Name
316     ORDER BY s.Supplier_Name ASC;
317
```

00% 30:316

Result Grid | Filter Rows: Search | Export:

Supplier_ID	Supplier_Name	Total_Amount
S00001	ABC Company	3500
S00010	Wayne Enterprises	3200

ABC Company & Wayne Enterprises.

3. Find the most underperforming Server with his details?

```
318 •  SELECT a.SEmployee_ID, SUM(a.No_Of_Customers_Served) AS Total_Served, b.Manager_ID, b.Branch_ID, b.Emp_Name, b.Join_Dat
319    FROM Served a
320    JOIN Employee b ON a.SEmployee_ID = b.Employee_ID
321    GROUP BY a.SEmployee_ID
322    ORDER BY Total_Served ASC;
```

00% 27:322

Result Grid | Filter Rows: Search | Export:

SEmployee_ID	Total_Served	Manager_ID	Branch_ID	Emp_Name	Join_Date
E00008	4	E00003	B0002	Mary Wong	2020-05-01
E00007	5	E00002	B0001	Peter Chen	2020-04-01
E00006	11	E00002	B0001	Adam Park	2020-04-01

Mary Wong from London Branch is the underperforming Server.

Assumptions

1. *Only one Phone Number is taken from a customer, and it is not a multivalued attribute.*
2. *All other employees other than Brewer and Server do not have Salary. They have Stake in the company.*

Learnings from Project

1. *Proper Plan and entity establishment must be done before we continue to Relational Model.*
2. *Attribute names should not have space in between. We re-did the ERD Charts & Relational Model after we got error in SQL.*
3. *Sequence of Table creation is crucial. So, we thought of creating Tables without Foreign key first and then create those tables which are dependent.*
4. *It takes time, effort and especially Team work to build a database and design where data must be stored.*
5. *Normalizing the data into 3NF really showed us an effective way to store the data.*