Chen Yuqiu

Amol P Pathak

Rachana Ramshetty

Ambati Yamini Praneetha

Mandeep Singh Rahi

MGMT 58200 GrouP 14

Project Report

2022

# Background

Kung Fu Tea is a chain of milk tea shop founded in Queens, NY on April 30, 2010. As the largest American bubble tea brand, Kung Fu Tea has placed their stores all around the world. Although not all their stores manage their data well. Our client is Kung Fu Tea located in Chauncey Avenue West Lafayette. They have been operated for 3 years as an outlet of Kung Fu Tea, while they still face supply chain issue and high customer churn rate due to unmet customer needs.

The data table they currently have is as follows:

**Sales**: Contains the sale volume for each product each month.

The attributes in this entity are

**Sales\_ID**: Primary key that identifies the sales

**Product\_ID**: Primary key that identifies the product

**Product\_Name**: Name of the product

**Product\_series:** Category of each product

**ProductPrice**: Corresponding price for each product

**Sale\_Vol**: Sale volume for each product

**Outlet\_ID**: Foreign Key identifies the outlet that the record belongs to

**Shift**: There are two shifts per day which are morning shift and evening shift. Each shift has exactly two employees. This table contains all shifts and their corresponding employees.

The attributes in this entity are

**Shift\_ID**: Primary key that identifies the shifts

**Date**: The date of each shift

**Morning\_Night**: Whether the shift is a morning shift or night shift

**Employee\_ID**: Foreign key that identifies employee who worked in a specific shift

**Employee\_Name**: Name of the employee who worked in a specific shift

**Employee2\_ID**: Foreign key that identifies employee who worked in a specific shift

**Employee2\_Name**: Name of the employee who worked in a specific shift

**Outlet\_ID**: Foreign Key identifies the outlet that the record belongs to

**Employee**: Contains all employee information and number of shift they take each month

The attributes in this entity are

**Employee\_ID**: Primary Key identifies employees

**Employee\_Name**: Name of the employee

**Hourly\_Wage**: The hourly salary of the employee

**Status**: Whether the employee is currently available for work

**Outlet\_ID**: Foreign Key identifies the outlet that the record belongs to

**Order**: Contains the information about ordering raw material.

The attributes in the entity are

**Order\_ID**: Primary key identifies orders

**Material\_ID**: Foreign key identifies Material

**Supplier\_Address**: Adress where the material shipping from

**Contact\_number**: Phone number of the supplier

**Order\_quantity**: Number of each material in a specific order

**Material\_Cost**: Unit cost of each material in the order

**Outlet\_ID**: Foreign Key identifies the outlet that the record belongs to

**Material**: Recorded material usage.

The attributes are

**Material\_ID**: Primary key identifies materials

**Material\_Name**: Name of the material

**Quantity\_Usage** : number of units used

**Outlet\_ID**: Foreign Key identifies the outlet that the record belongs to

**Outlets:** There are thousands of outlets under the brand Kung Fu Tea. It is not the table provided by our client. We only focus on the outlet located in Chauncy Avenue West Lafayette. By creating this table and adding Outlet\_ID as a foreign key in all previous tables, we can easily connect all the tables together. If they plan to operate another outlet, they can easily insert the latest

information in the database by updating the information in this table.

The attributes in this entity are

**Outlet\_ID**: Primary Key identifies outlets

**Outlet\_City**: City of the outlet

**Outlet\_Street**: street the outlet locates

**Outlet\_State**: State the outlet locates

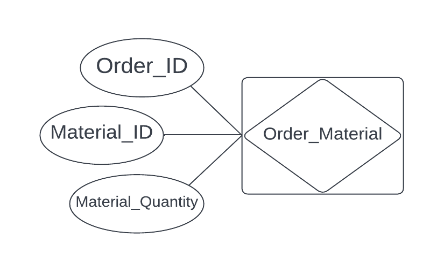
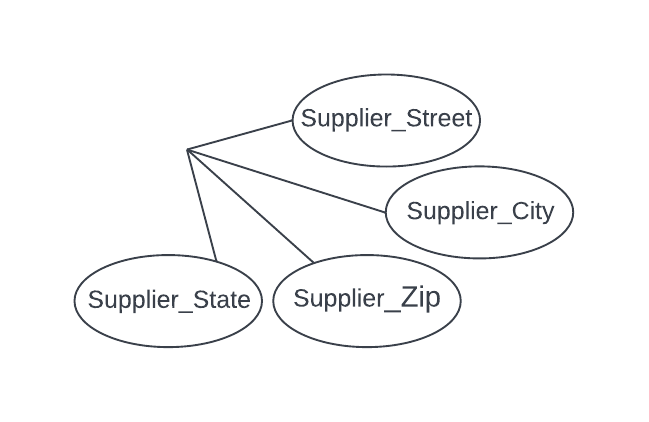
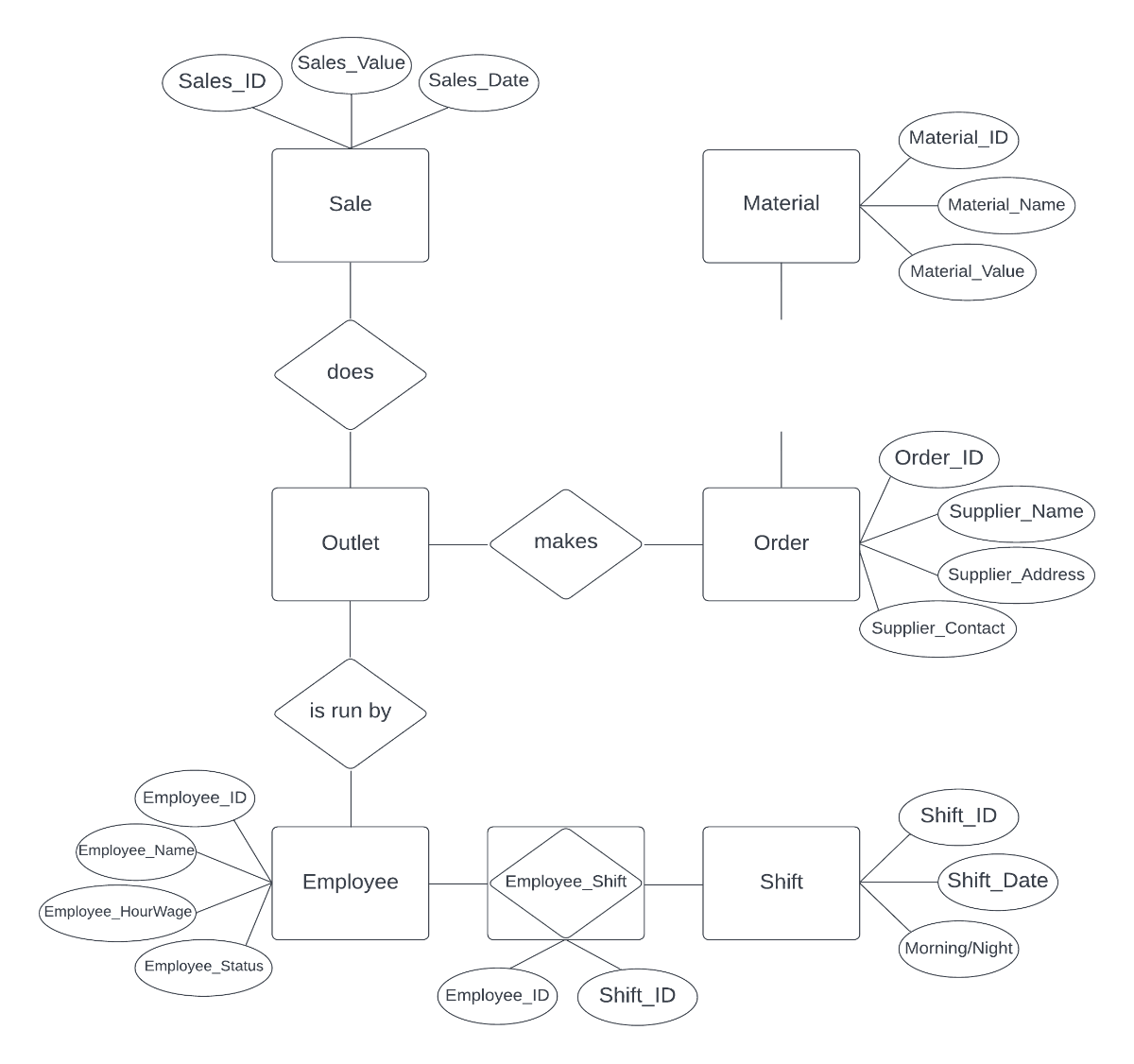
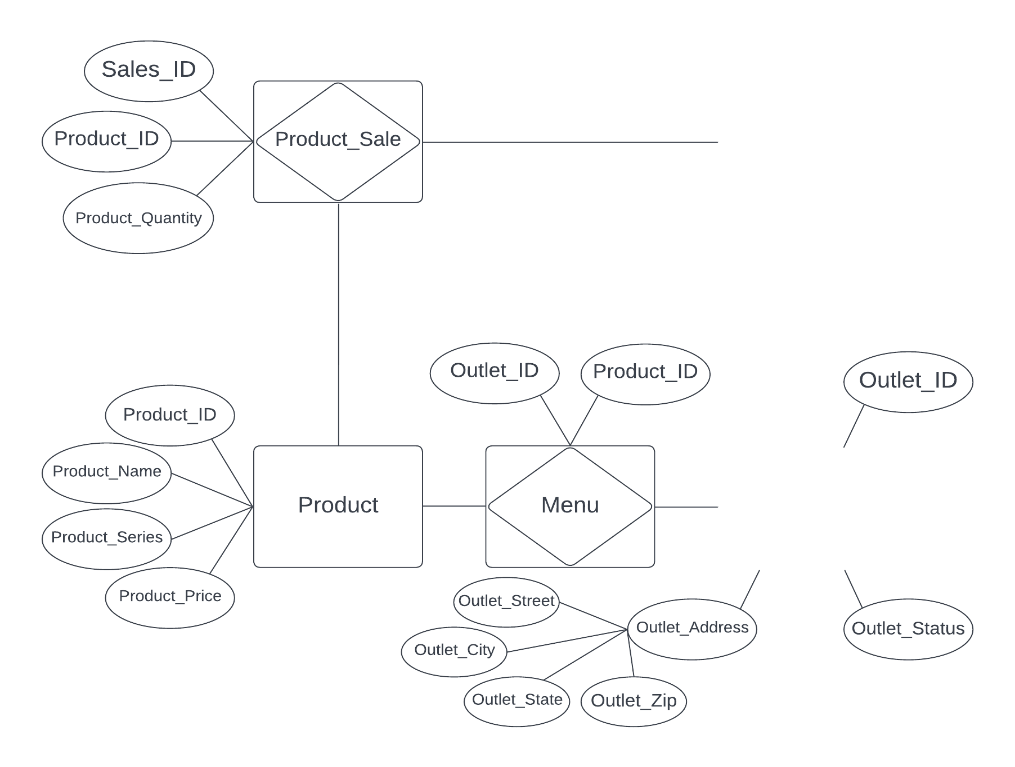
**Outlet\_Zip**: Corresponding zip code

**Status**: If the store is currently operating by the database owner

# Introduction

Their various data such as product sale volume, expenses, material usage, are recorded separately in different tables and unorganized. It is difficult for the manager to combine different data for analysis to predict the demand of each product and the order quantity of various materials. It also often exposes them to supply chain issues. Our group want to design a database for them so they can easily combine all kinds of data for analysis.

# Conceptual Data Modelling: ERD



* Outlets can have zero to many Products, whereas Products must have one to many Outlets.
* Outlets does zero to many Sales, whereas Sale belong to only one Outlet.
* Products can have zero to many Sales, whereas Sale can have one to many Products.
* Outlets are run by zero to many Employees, whereas Employees belong to only one Outlet.
* Employees can have zero to many Shifts, whereas Shifts can have one or many Employees.
* Outlets can have zero to many Orders, whereas Order belong to only one Outlet.
* Orders can have one to many Materials, whereas Materials can belong to zero to many Orders.

# Relational Data Model: Relational Schema

Outlet(Outlet\_ID, Outlet\_Status, Outlet\_Street, Outlet\_City, Outlet\_State, Outlet\_Zip)

Order(Order\_ID, Supplier\_Contact, Supplier\_Name, Supplier\_Street, Supplier\_City, Supplier\_State, Supplier\_Zip, Outlet\_ID)

Material(Material\_ID, Material\_Name, Material\_Value)

Employee(Employee\_ID, Employee\_Name, Employee\_HourWage, Employee\_Status, Outlet\_ID)

Shift(Shift\_ID, Shift\_Date, Morning\_Night)

Product(Product\_ID, Product\_Name, Product\_Series, Product\_Price)

Sale(Sales\_ID, Sales\_Date, Sales\_Value, Outlet\_ID)

*Associative Tables:*

Order\_Material(Order\_ID, Material\_ID, Material\_Quantity)

Shift\_Employee(Shift\_ID, Employee\_ID)

Menu(Outlet\_ID, Product\_ID)

Product\_Sale(Sales\_ID, Product\_ID, Product\_Quantity)

# Normalization Process

**No Normal Form:** None of the tables are 0NF, there is not any multi-valued and composite attribute

**First Normal Form:** All tables are in 1NF, each tuple contains exactly one value for each attribute.

**Second Normal Form:**

**Functional Dependency:** For the sales table, there are two primary keys which are Sales\_ID and Product\_ID. The Sale\_Vol depends on both Sales\_ID and Product\_ID, the Product\_Name, SalePrice and Product\_Series only depends on Product\_ID.

Therefore, The Sales table is not in 2NF because not all non-key attributes are fully functionally dependent on the entire primary key.

**Decomposition to 2NF:** To normalize the Sales table, we divide it into two tables: A Sale table contains only Sales\_ID and Sale\_Vol and Product table contains the rest of information.

**Third Normal Form:** All tables are in 3NF, there is no transitive dependency for non-key attributes.

# SQL Implementation

The following are a few of the queries we have implemented based on the business requirement:

1. Top Products sold (Top 5 products sold):

select p.\*,sum(sp.Product\_Quantity) as Total\_Sale from product p

left join sales\_product sp on sp.product\_ID = p.product\_ID

group by p.product\_ID

order by sum(sp.Product\_Quantity) desc

Limit 5;

Table

Description automatically generated

1. Most ordered material (Top 5)

select m.\*,sum(om.quantity) as Total\_Quantity from material m

left join order\_material om on om.material\_ID = m.material\_ID

group by m.material\_ID

order by sum(om.quantity) desc

Limit 5;

Graphical user interface, text, application

Description automatically generated

1. Cost of Materials ordered over the entire time period

select m.\*,sum(om.quantity) as Total\_Quantity, m.value\*sum(om.quantity) as Total\_Amount from material m

left join order\_material om on om.material\_ID = m.material\_ID

group by m.material\_ID

order by Total\_Amount desc;

Table

Description automatically generated

1. Total sales value grouped by Product\_ID

select p.\*,sum(sp.Product\_Quantity) as Total\_Sale from product p

left join sales\_product sp on sp.product\_ID = p.product\_ID

group by p.product\_ID

order by p.product\_ID;





1. Employee with max shifts

select e.employee\_ID, e.employee\_name, count(\*) as Shift\_Count from employee e

left join shift\_employee se on e.employee\_ID = se.employee\_ID

group by e.employee\_ID

order by count(\*) desc

limit 1;

Table

Description automatically generated

1. Are there any employees with no shift

select e.employee\_ID, e.employee\_name from employee e

left join shift\_employee se on e.employee\_ID = se.employee\_ID

group by e.employee\_ID

Having count(se.shift\_ID) =0

order by e.employee\_ID;

Table

Description automatically generated

1. Products not sold at all

select p.\*,sum(sp.Product\_Quantity) as Total\_Sale from product p

left join sales\_product sp on sp.product\_ID = p.product\_ID

group by p.product\_ID

Having sum(sp.Product\_Quantity) =0;

Table

Description automatically generated

1. No. of sales with sales value greater than $500 in a day

select count(sales\_ID) from sales

where sales\_value > 500;

Graphical user interface, application

Description automatically generated

1. No. of products sold starting with “coffee” in product\_name

select p.\*,sum(sp.Product\_Quantity) as Total\_Sale from product p

left join sales\_product sp on sp.product\_ID = p.product\_ID

where product\_name like 'coffee%'

group by p.product\_ID

order by sum(sp.Product\_Quantity) desc;

Graphical user interface, application, table

Description automatically generated with medium confidence

1. Total sale volume

select sum(sales\_value) as Total\_Sale from sales;

Graphical user interface, text, application

Description automatically generated

1. Sum of sales value grouped by month

select month(sales\_Date) as Sale\_Month,sum(sales\_Value) as Total\_sale\_Value

from sales

group by month(sales\_Date)

order by sum(sales\_Value) desc;

Table

Description automatically generated

1. Products sold and their quantities when there is maximum sale

select p.\*, s.\*, sp.product\_quantity from product p

left join sales\_product sp on sp.product\_ID = p.product\_ID

left join (select \* from sales order by sales\_value desc Limit 1) s on s.sales\_ID = sp.sales\_ID

where s.sales\_ID is not null;

Table

Description automatically generated

1. Maximum average value of material across all the orders

select avg(m.value) from orders o

left join order\_material om on o.order\_ID = om.order\_ID

left join material m on m.material\_ID = om.material\_ID

group by o.order\_ID

order by avg(m.value) desc

limit 1;

Graphical user interface, text, application

Description automatically generated

1. Most ordered product within each series

With top\_sold as

(select p.\*,sp.Total\_Quantity, Row\_Number() over(partition by p.product\_series order by sp.Total\_Quantity desc) as Max\_Quantity

from product p

left join (select product\_ID,sum(product\_quantity) as Total\_Quantity from sales\_product group by product\_ID) sp

on p.product\_ID = sp.product\_ID)

select product\_ID, Product\_Name, Price, Total\_Quantity from top\_sold where Max\_Quantity = 1;

Table

Description automatically generated

# Commentary

Our clients, Kung Fu Tea, were struggling with unorganized data. Our team understood their business scenario and designed a database solution that solves their business problem and is also future proof.

In order to do so, we prepared an ER Diagram and Relational Schema that represents their business scenario. We also ensured that the data is normalized to avoid any data redundancy. While designing the database, we added a new entity for Outlets to incorporate the possibility of future expansion of the company. The team organized the data and implemented the database on MySQL Server. Later, we executed several queries on the database as testing.

Our solution will improve the organization of data for our clients by providing normalized database structure. With the implementation of database on MySQL Server, in future, our clients could also integrate additional applications to easily interact with this data that could solve business challenges such as demand planning, sales analysis, employee information tracking, etc.