VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANASANGAMA, BELGAVI-590018

DATABASE MANAGEMENT SYSTEM MINI PROJECT
(18CSL58)

ON
"MINIMIZING FOOD LEFTOVERS SYSTEM"
Submitted in partial fulfillment of the requirements for the 5th Semester

INFORMATION SCIENCE AND ENGINEERING

Submitted by

SHREEDATTA NASIK (1BI20IS092)

SNEHIT KINAGI (1BI20IS103)

Under the guidance of

Mrs. Anupama K.C   Dr. Vani V
Assistant Professor   Assistant Professor
Dept. of ISE   Dept. of ISE
BIT, Bangalore-04   BIT, Bangalore-04

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
BANGALORE INSTITUTE OF TECHNOLOGY

K. R. Road, V. V. Puram, Bengaluru-560004

2022-2023

DECLARATION

I, SNEHIT KINAGI bearing USN **1BI20IS092** student of Fifth Semester **Bachelor of Engineering** in **Information Science Engineering, Bangalore Institute of Technology, Bangalore** declare that the project work has been carried out by me and submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Engineer**ing in **Information Science Engineering** of **Visvesvaraya Technological University, Belagavi** during the year 2022-2023. The matter embodied in this report has not been submitted to any university or institute for the award of any other degree.

Place: Bangalore   **Name: SNEHIT KINAGI**

Date: **USN: 1BI20IS103**

ACKNOWLEDGMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. So, with gratitude, I acknowledge all those whose guidance and encouragement crowned my effort with success.

It's an immense pleasure to thank principal **Dr. Ashwath M.U**, BIT, Bangalore, for providing an opportunity to present this project as a part of my curriculum in the partial fulfillment of the degree.

I express sincere gratitude to **Dr. Roopa H**, Head of the Department, Information Science and Engineering for her co-operation and encouragement at all moments of my approach.

It is pleasant duty to place on record my deepest sense of gratitude to my guide **Mrs. Anupama K C**, Asst. Professor for the constant encouragement, and **Dr. Vani V**, Asst. Professor for valuable help and assistance in every possible way.

I would like to thank all **teaching and non-teaching staff** of ISE Department for providing their valuable guidance and for being there at all the stages of my world.

SNEHIT KINAGI (1BI20IS103)

ABSTRACT

Food Leftovers is one of the main forms loss in restaurants, as the leftovers cannot be used and they would have to either have to let it go to waste or donate it. The purpose of this project is to solve this dilemma by predicting the food items that will be consumed in the specified amount of time.

This project strives to give the owner of a given restaurant a specified number of food items from which he can make further accommodations. This project identifies the parameters that are responsible for the consumption of food items like timings, day of the week, holidays etc. which are specific to a given restaurant business. It then tries to collect data on those parameters and how it affects the specified food item. After enough data, it processes those data predict the its future consumption

The owner inputs the parameters that the food item is dependent upon and gets the approximate consumption of that food item for those parameters. This project works as a feature to restaurant management system. The current system is manual and it is time consuming. This project requires large amount data to make passable prediction. The project is designed with a specific restaurant in mind and this system cannot be generalized as specific restaurants have their own specific parameters.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this one can define DBMS as a collection of interrelated data and a set of programs to store & access those data in an easy and effective manner.

1.1 The need for DBMS

Database systems are basically developed for large amounts of data. When dealing with huge amounts of data, there are two things that require optimization: Storage of data and retrieval of data.

 It can be used for:

- Creation of a database
- Retrieval of information from the database.
- Updating the database.
- Managing a database.

Storage: According to the principles of database systems, the data is stored in such a way that it acquires a lot less space as the redundant data (duplicate data) has been removed before storage.

Fast Retrieval of data: Along with storing the data in an optimized and systematic manner, it is also important to retrieve the data quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

The choice of a database product is often influenced by factors such as:

- Cost
- the computing platform (i.e., hardware, operating system)
- The volume of data to be managed.
- The number of transactions required per second.

1.2 Design and Modeling

The first task of a database designer is to produce a conceptual data model that reflects the structure of the information to be held in the database. A common approach to this is to develop an entity-relationship model, often with the aid of drawing tools. Another popular approach is the Unified Modeling Language. A successful data model will accurately reflect the possible state of the external world being modeled: for example, if people can have more than one phone number, it will allow this information to be captured as shown in the Figure 1.1.
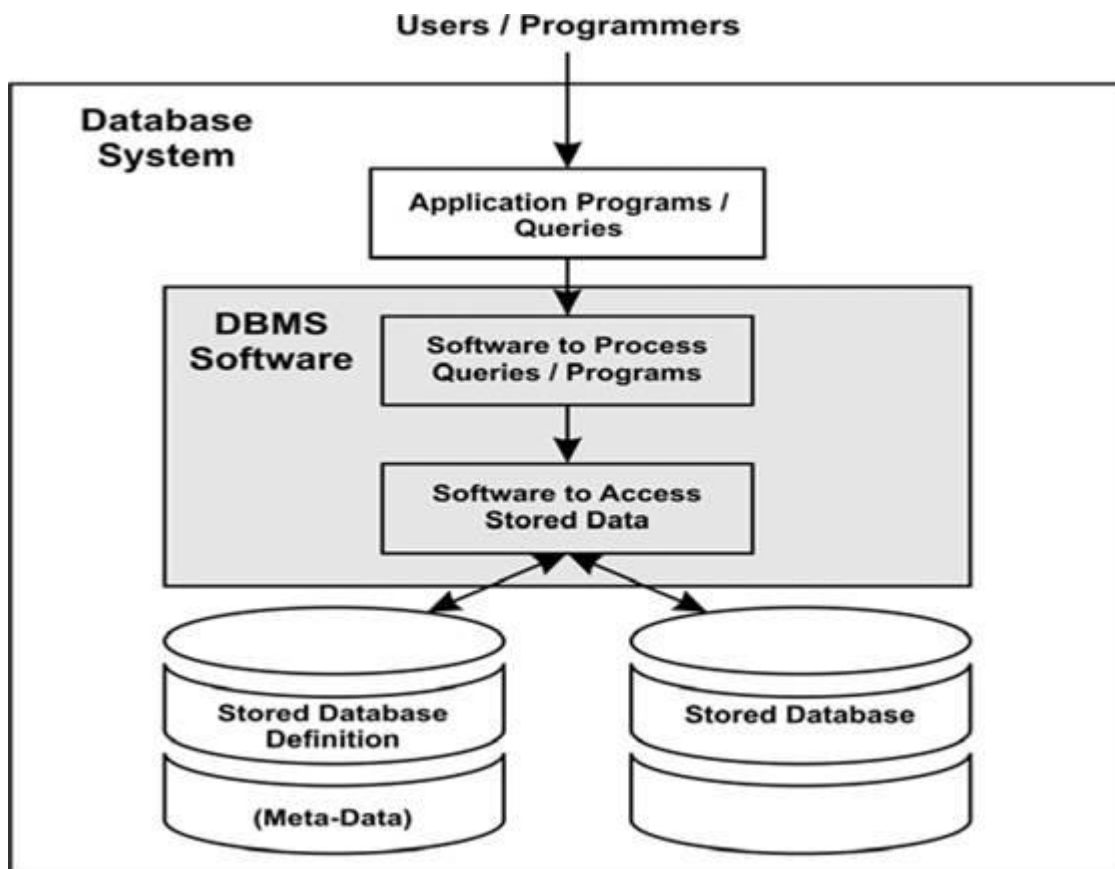
Figure 1.1 Simplified database environments

1.3 Problem Statement

In the past few years machine learning has progressed in a substantial manner. Day by day machine learning concepts are being integrated to fields that are not at all related and getting

desirable result. But not many projects are being done on the restaurant field which houses a large amount of data. The major aspect in which the restaurants lose their money and resources is due to the leftovers. The purpose is to apply the concepts of machine learning and develop a system that predicts the food items that would be consumed and hence reducing the leftovers as it is not completely accurate. Still the owner can use this data to his advantage and make the necessary arrangements in cooking food items.

1.4 Objectives

- Identify the parameters on which each food items are dependent upon
- Store the data on those parameters in the database on a certain manner specified by the schema
- Collect the data until required amount.
- Processing of data and fitting it with a predicting model
- Building an application program which predicts the consumption of food item specified by the parameters passed.

CHAPTER 2

SOFTWARE AND HARDWARE REQUIREMENT

The program works on Desktop PC and is executed using a PHP which interacts with a SQL database running on localhost.

2.1 Software Requirements

- **Operating System:**   Windows 7/8/10/11
- **Back End:**   python
- **Front End:**   HTML, CSS, flask
- **Application required:** Visual Studio and the database used is MYSQL

2.2 Hardware Requirements

- **Hard disk:** 15GB/Above
- **RAM:**   1GB/Above
- **Processor:** Intel core i3/i4/i5/i7 or equivalent AMD processors

CHAPTER 3

DESIGN

In the below section, a brief overview of how the project is designed is given, including the Entity Relation diagram and Schema diagram which describes the various attributes used in the design and their types.

3.1 ER Analysis: Identifying Entity Sets and Relationship Sets

Entity Relationship Model (ER Modeling) is a graphical approach to database design. It is a high-level data model that defines data elements and their relationship for a specified software system. An ER model is used to represent real-world objects.

3.1.1 Entities and Attributes

An **Entity** is a thing or object in the real world that is distinguishable from the surrounding environment. An Entity is an object of Entity Type and the set of all entities is called an **Entity set. Attributes** are the properties which define the entity type.

The section of the document explains the entities used in the project, their attributes and how they will work together. Basically, this is intended to make the design easier and understandable for everyone.

ENTITIES

1. Food_item
2. Order_Dates
3. Order
4. Date_table
5. Total_items

1. Food_item: Attributes of this entity are food_id, max_quantity, min_quantity, unit_price as shown in the Table 3.1.

Table 3.1: Food_item Database Table

ATTRIBUTE	DATATYPE	DESCRIPTION
FOOD_ID	INT	PRIMARY KEY
MAX_QUAN	INT	NO KEY ATTRIBUTE
MIN_QUAN	INT	NO KEY ATTRIBUTE
UNIT_PRICE	INT	NO KEY ATTRIBUTE

2. ORDER_DATES: Attributes of this entity are order_id, date1, time as shown in the Table 3.2.

Table 3.2: Order_dates Database Table

ATTRIBUTE	DATATYPE	DESCRIPTION
ORDER_ID	INT	PRIMARY KEY
DATE1	INT	FOREIGN KEY
TIME	INT	FOREIGN KEY

3. ORDERS: Attributes of this entity are order_id, food_id, quantity as shown in the Table 3.3.

Table 3.3: Orders Database Table

ATTRIBUTE	DATATYPE	DESCRIPTION
ORDER_ID	INT	PRIMARY KEY
FOOD_ID	INT	PRIMARY KEY
QUANTITY	INT	NO KEY ATTRIBUTE

4. DATE_TABLE: Attributes of this entity are date1, day as shown in the Table 3.4.

Table 3.4: Date_Table Database Table

ATTRIBUTE	DATATYPE	DESCRIPTION
DATE1	DATE	PRIMARY KEY
DAY	INT	NO KEY ATTRIBUTE

5. TOTAL_ITEMS: Attributes of this entity are date1, time, id_1, id_2, id_3,id_4 as shown in the table 3.5.

Table 3.5:Total_items Database Table

ATTRIBUTE	DATA_TYPE	DESCRIPTION
DATE1	INT	PRIMARY KEY
TIME	INT	PRIMARY KEY
ID_1	INT	NO KEY ATTRIBUTE
ID_2	INT	NO KEY ATTRIBUTE
ID_3	INT	NO KEY ATTRIBUTE
ID_4	INT	NO KEY ATTRIBUTE

3.1.2 Relationship Sets

A relationship type represents the **association between entity types.** A set of relationships of the same type is known as a relationship set.

The following are the relationships between the various entities used in the project.

- Has permissions: to check the permissions various groups have in the management system
- Authenticates: for one group, usually the admin, to grant permission to another group
- Verifies: for one group, usually the admin, to verify the credentials and to give access to something
- Accesses: usually, the admin has access to the main cart items, to add, delete, update information on various items
- Gets added: the admin gets to add items to be displayed in the home page of the management system
- Adds: here, the authenticated user, usually a supplier, add details and items to be displayed and accessed on the home page

3.2 Entity Relationship Diagram

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems

3.2.1 ER Diagram with relationships and cardinality ratio

The cardinality or fundamental principle of one data aspect with respect to another is a critical feature. The relationship of one to the other must be precise and exact between each other in order to explain how each aspect links together. In simple words Cardinality is a way to define the relationship between two entities. All the ER diagram notations used are shown in the Figure
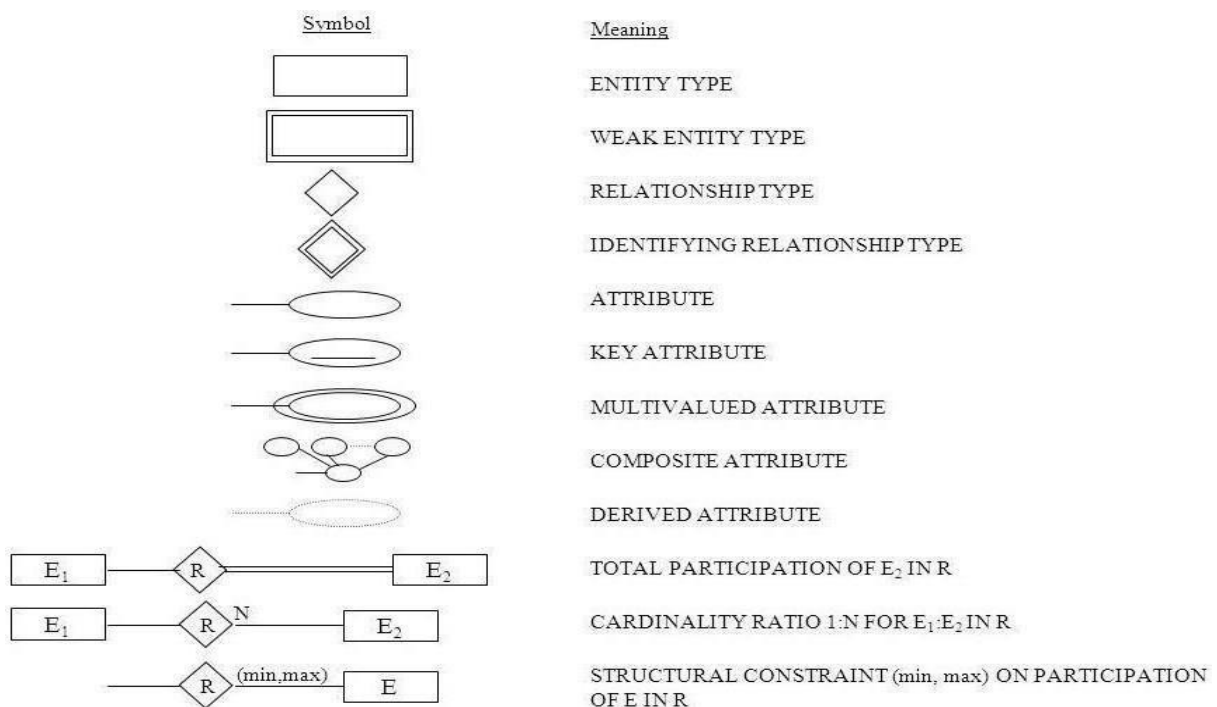
Figure 3.1 ER Diagram notations

The Figure 3.2 ER diagram shows the relationship between the many entities that exist in database for:

Food Leftover minimizing system
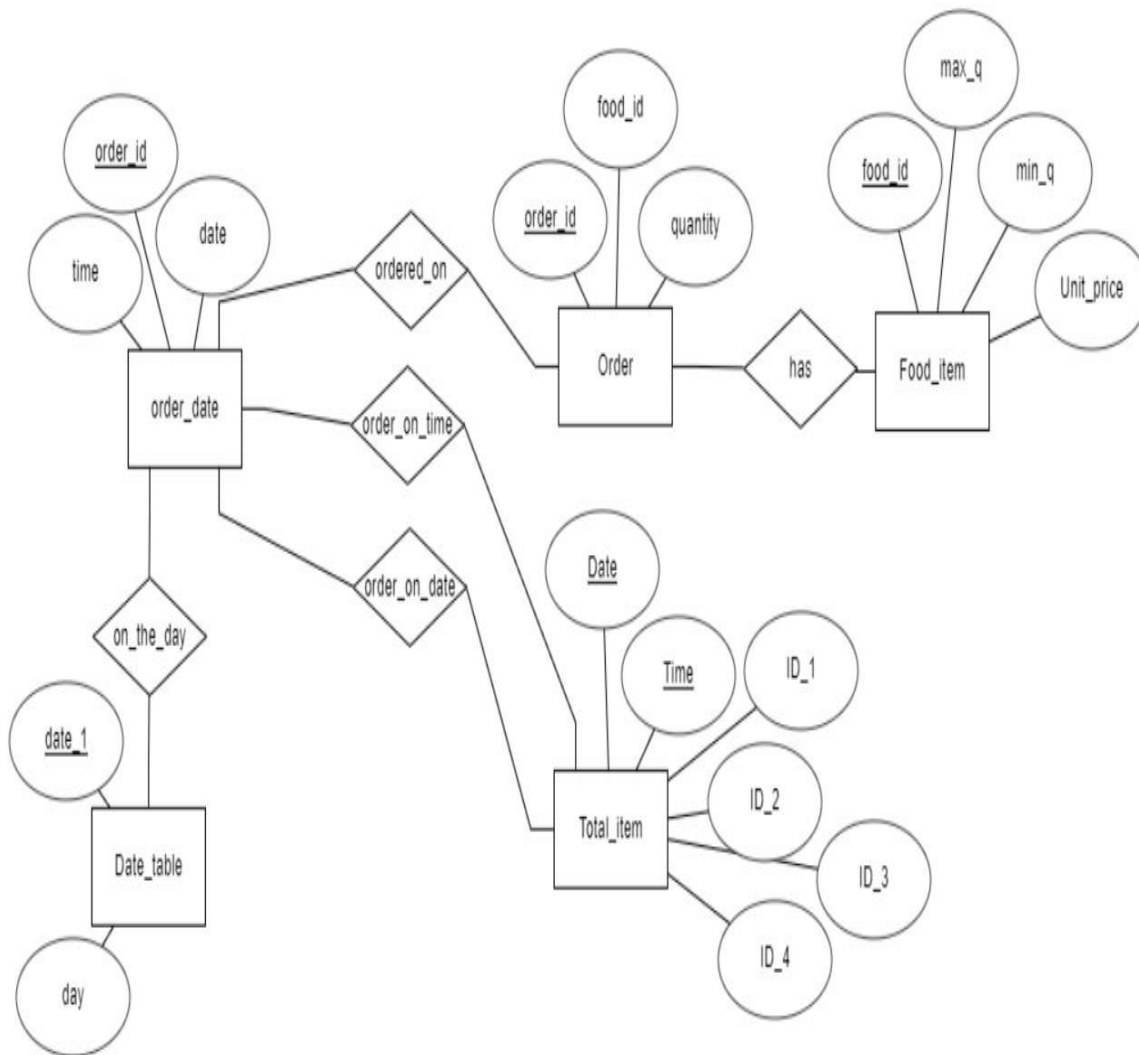
Figure 3.2: ER Diagram of Leftover Minimization

3.3 Schema Diagram

In any data model it is important to distinguish between the description of the database and the database itself. The description of a database is called the database schema, which is specified during database design and is not expected to change frequently. A displayed schema is called a schema diagram. A schema diagram displays only some aspects of a

schema, such as the names of record types and data items, and some types of constraints as shown in the Figure 3.3.

FOOD_ITEMS:

FOOD_ID	MAX_QUAN	MIN_QUAN	UNIT_PRICE

ORDER_DATE:

ORDER_ID	DATE1	TIME

ORDERS:

ORDER_ID	FOOD_ID	QUANTITY

DATE_TABLE:

DATE1	DAY

TOTAL_ITEMS:

DATE1	TIME	ID_1	ID_2	ID_3	ID_4
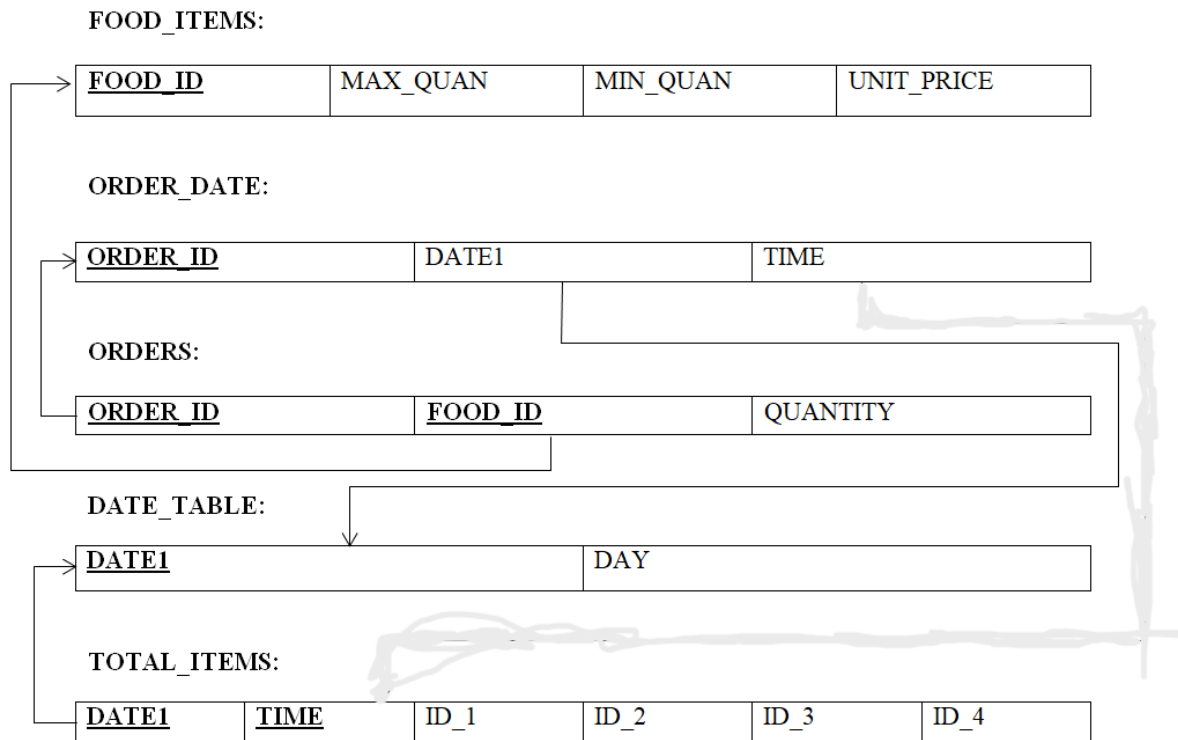
Figure 3.3: Schema diagram

3.4 Tables and Functional Dependencies

Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute in a Database Management System (DBMS). Functional Dependency helps to maintain the quality of data in the database. It plays a vital role to find the difference between good and bad database design.

A functional dependency is denoted by an arrow "→". The functional dependency of X on Y is represented by X → Y

1. Food_items(food_Id,max_quan,min_quan,unit_price)

   FD={food_id->max_quan,min_quan,unit_price)

2. Order_dates(order_id,date1,time)

   FD={order_id->date1,time}

3. Orders(order_id,food_id,quantity)

   FD={order_id,food_id->quantity}

4. Date_table(date1,day)

   FD={date1->day}

5. Total_items(date1,time,id_1,id_2,id_3,id_4)

   FD={date1,time-> id_1,id_2,id_3,id_4}

3.5 Normalization

Normalization is a process of organizing the data in a database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normalization is used for mainly two purposes,

1. Eliminating redundant(useless) data.

2. Ensuring data dependencies make sense i.e data is logically stored.

3.5.1 First Normal form

For a table to be in the First Normal Form, it should follow the following 4 rules

    1. It should only have single(atomic) valued attributes/columns.

    2. Values stored in a column should be of the same domain

    3. All the columns in a table should have unique names.

    4. And the order in which data is stored, does not matter.

    .

3.5.2 Second Normal form

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.
3. All the tables which are part of this project are in 2NF as they have at most one primary key, so no partial dependency.

3.5.3 Third Normal form

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

All the used tables satisfy both these conditions and hence are in 3NF.

3.5.4 Other Normalization Forms

BCNF (Boyce Codd Normal Form) is the advanced version of 3NF. A table is in BCNF if every functional dependency X->Y, X is the super key of the table. For BCNF, the table should be in 3NF, and for every FD. LHS is super key.

Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce-Codd Normal Form (BCNF).

Fifth normal form (5NF) is a level of database normalization where a table is in 4NF and does not contain any join dependency and joining should be lossless.

All the tables that we have used satisfied the conditions for 1NF, 2NF and 3NF.

CHAPTER 4

IMPLEMENTATION

This section focuses on the implementation of leftover minimizing system. The steps and the technologies used are listed below.

4.1 Basic Steps in Implementation:

- The data in the database is imported into python ide through the package sqlconnector.
- The data is read into a dataframe through pandas package.
- A linear regression model with multi variable is fitted to all the food items
- The parameters are given from the front end.
- The regression objects predicts the consumption of food_item based on the sent parameters
- The predicted values are then sent back to front end.
- Flask frame work is used to connect backend and front end

4.2 Technologies and Frameworks

HTML

Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. Functional Modules.

FLASK

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy[3].

LINAER REGRESSION

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a "least squares" method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable)[1].

4.3 Frontend Code:

Left1.html
```
<html lang="en" dir="ltr">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="static/style2.css">
```

```
    <title>Admin Panel</title>
</head>
<body>
    <div class="side-menu">
      <div class="brand-name">
        <h1>FOOD PREDICTION</h1>
      </div>
       <ul>
        <li><a href="#"></a><span>HOME</span> </li>
        <li                    class="active"><a                    href="http://127.0.0.1:5000/"
target="_blank"></a> <span>PREDICT</span> </li>
        <li><a href="#"></a> <span>HISTORY</span> </li>
        <li><a href="#"></a> <span>MENU</span> </li>


       </ul>
      </div>
    </div>
    <div class="container">
      <div class="title">Prediction</div>
        <div class="content">
          <form action="{{url_for('result')}}" method=post>
           <div class="user-details">
             <div class="input-box">
               <span class="details">Date</span>
               <input type="date" name="mydate">
             </div>
             <div class="t1">
              <span class="details">Time</span>
                <div class="t2">
                  <select name="mytime" id="time">
                  <option value="60" selected>60</option>
                  <option value="70">70</option>
```

```
                    <option value="80">80</option>

                    <option value="90">90</option>

                    </select>

                  </div>

              </div>

              <div class="button">

                <input type="submit" value="Predict">

              </div>

              <div class="button1"><input type="reset" value="Reset"></div>

               <div class="message">

                 Food_id1:{{ msg1 }}

                 <br>

                 Food_id1:{{ msg2 }}

                 <br>

                 Food_id1:{{ msg3 }}

                 <br>

                 Food_id1:{{ msg4 }}

               </div><br>

            </form>

          </div>

        </div>

     </body>

</html>


Left2.css

* {

   margin:0;

   padding: 0;

   box-sizing: border-box;

   font-family:Georgia, 'Times New Roman', Times, serif;

}

header{
```

```css
    background-image: url(minimal-geometric-stripe-shape-background_1409-1014.webp);

    height: 100vh;

    background-size: cover;

    background-position:center;

}

body {

    min-height: 100vh;

}

a {

    text-decoration: none;

}

li {

    list-style: none;

}

h1,

h2 {

    color:firebrick;

}

h3 {

    color: #999;

}

.btn {

    background:black;

    color: white;

    padding: 5px 10px;

    text-align: center;

}

.btn:hover {

    color: #f05462;

    background: white;

    padding: 3px 8px;

    border: 2px solid #f05462;
```

```
}
.title {
   display: flex;
   align-items: center;
   justify-content: space-around;
   padding: 15px 10px;
   border-bottom: 2px solid #999;
}
table {
   padding: 10px;
}
th,
td {
   text-align: left;
   padding: 8px;
}
.side-menu {
   position: relative;
   right: 165px;
   background-image: url(minimal-geometric-stripe-shape-background_1409-1014.webp);
   background-size: cover;
   background-position:center;
   width: 20vw;
   min-height: 100vh;
   display: flex;
   flex-direction: column;
}
.sidemenu .ul li.active a{
   background-color: #fff;
   color:black;
}
.side-menu .brand-name {
```

```css
    height: 10vh;
    display: flex;
    align-items: center;
    justify-content: center;
}
.side-menu li {
    font-size: 24px;
    padding: 10px 40px;
    color: white;
    display: flex;
    align-items: center;
    transition: 0.6s ease;
    padding: 5px 30px;
}
.side-menu li:hover {
    background: white;
    color:darkblue;
}
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family:Georgia, 'Times New Roman', Times, serif;
 }
 body{
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 10px;
    background-color: antiquewhite;
 }
```

```css
.container{
 height: 60%;
 max-width: 900px;
 width: 100%;
 background-color: #fff;
 padding: 25px 30px;
 border-radius: 5px;
 box-shadow: 0 5px 10px rgba(0,0,0,0.15);
}
.container .title{
 font-size: 25px;
 font-weight: 500;
 position: center;
}
.container .title::before{
 content: "";
 position: absolute;
 left: 0;
 bottom: 0;
 height: 3px;
 width: 30px;
 border-radius: 5px;
 background-color: antiquewhite;
}
.content form .user-details{
 display: flex;
 flex-wrap: wrap;
 justify-content: space-between;
 margin: 20px 0 12px 0;
}
form .user-details .input-box{
 margin-bottom: 15px;
```

```
  width: calc(100% / 2 - 20px);
 }
 form .input-box span.details{
  display: block;
  font-weight: 500;
  margin-bottom: 5px;
 }
 .user-details .input-box input{
  height: 45px;
  width: 100%;
  outline: none;
  font-size: 16px;
  border-radius: 5px;
  padding-left: 15px;
  border: 1px solid #ccc;
  border-bottom-width: 2px;
  transition: all 0.3s ease;
 }

 .form .t1 {
  margin: 25px;
  padding: 20px;
 }
 .form .t2{
   position: relative;
   left: 1px;
   height: 45px;
   width: 100%;
   outline: none;
   font-size: 16px;
   border-radius: 5px;
   padding-left: 15px;
```

```
    border: 1px solid #ccc;

    border-bottom-width: 2px;

    transition: all 0.3s ease;

    position:static;

  }


 .user-details .input-box input:focus,
 .user-details .input-box input:valid{

  border-color: black;

 }
 form .button{

  height: 45px;

  position: relative;

  top: 58px;

  right: 450px;

  margin: 35px 0;

}


  form .button1{

   position: relative;

   top: 96px;

   right: 420px;

  }
  form .button input{

   height: 130%;

   width: 125%;

   border-radius: 5px;

   border: none;

   color: #fff;

   font-size: 18px;

   font-weight: 500;

   letter-spacing: 1px;
```

```
    cursor: pointer;

    transition: all 0.3s ease;

    background-color: darkblue;


   }
  form .button1 input{

    height: 48%;

    width: 170%;

    border-radius: 5px;

    border: none;

    color: #fff;

    font-size: 18px;

    font-weight: 500;

    letter-spacing: 1px;

    cursor: pointer;

    transition: all 0.3s ease;

    background-color: darkblue;

   }
  form .button input:hover{

  /* transform: scale(0.99); */

  background-color:blue;

  }
  form .button1 input:hover{

   /* transform: scale(0.99); */

   background-color: blue;

    }
 form .message{

   position: relative;

   top: 160px;

   right:600px;

 }
  @media(max-width: 584px){
```

```
   .container{
   max-width: 100%;
  }
 form .user-details .input-box{
    margin-bottom: 15px;
    width: 100%;
  }
  form .category{
   width: 100%;
  }
  .content form .user-details{
   max-height: 300px;
   overflow-y: scroll;
  }
  .user-details::-webkit-scrollbar{
   width: 5px;
  }
  }
  @media(max-width: 459px){
  .container .content .category{
   flex-direction: column;
  }
  }
```

4.4 Back- end code:

```python
from flask import Flask
from pyngrok import ngrok
from flask import render_template,request
from datetime import datetime
import calendar
import mysql.connector
import pandas as pd
import sqlalchemy
```

```python
import numpy as np
from sklearn import linear_model
mydb=mysql.connector.connect(host="127.0.0.1",user="root",        passwd="ramanujabatta457",
database='left_over1')
my=mydb.cursor()
my.callproc('callview')
for i in my.stored_results():
    print(i.fetchall())
my.callproc('callview')
for i in my.stored_results():
    print(i.fetchall())
my.execute("SELECT * FROM left1;")
for i in my:
    print(i)
engine=sqlalchemy.create_engine('mysql+pymysql://root:ramanujabatta457@localhost:3306/left
_over1')
dp=pd.read_sql_table("left1",engine)
display(pd.DataFrame(dp))
reg0 = linear_model.LinearRegression()
reg0.fit(dp[['TIME','DAY']],dp.ID_1)
reg1 = linear_model.LinearRegression()

reg1.fit(dp[['TIME','DAY']],dp.ID_2)
reg2 = linear_model.LinearRegression()
reg2.fit(dp[['TIME','DAY']],dp.ID_3)
reg3 = linear_model.LinearRegression()
reg3.fit(dp[['TIME','DAY']],dp.ID_4)
app =Flask(__name__, template_folder='template')

def finday(date):
    born = datetime.strptime(date,'%Y-%m-%d').weekday()
    return (calendar.day_name[born])
```

```python
def finddate(date1):
  if date1=='Monday':
    d=0
  elif date1=='Tuesday':
    d=1
  elif date1=='Wednesday':
    d=2
  elif date1=='Thursday':
    d=3
  elif date1=='Friday':
    d=4
  elif date1=='Saturday':
    d=5
  elif date1=='Monday':
    d=6
  return d

def findtime(time):
  if time=='60':
    t=60
  elif time=='70':
    t=70
  elif time=='80':
    t=80
  elif time=='90':
    t=90
  return t
@app.route("/",methods = ['POST','GET'])
def result():
  if request.method=='POST':
    date=request.form['mydate']
    time=request.form['mytime']
```

```
    print(date)
    print(time)
    date1=finday(date)
    print(date1)
    d=finddate(date1)
    print(d)
    t=findtime(time)
    print(t)
    pre1=reg0.predict([[t,d]])
    print(pre1)
    pre2=reg1.predict([[t,d]])
    print(pre2)
    pre3=reg2.predict([[t,d]])
    print(pre3)
    pre4=reg3.predict([[t,d]])
    print(pre4)
    return render_template('input.html', msg1 =pre1,msg2=pre2,msg3=pre3,msg4=pre4)
  return render_template('input.html', message = "")


if __name__ =='__main__':
    app.run(debug = True, use_reloader=False)
```

4.4.1. Table Creation

Creating a basic table involves naming the table and defining its columns and each column's data type.

The SQL CREATE TABLE statement is used to create a new table.

Syntax

The basic syntax of the CREATE TABLE statement is as follows − CREATE TABLE table_name(

column1 datatype, column2 datatype, column3 datatype,

.....

columnN data type,

PRIMARY KEY (one or more columns)

);

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is.

1. Creation of table food_items

```
CREATE TABLE `food_items` (
  `FOOD_ID` int NOT NULL,
  `MAX_Q` int DEFAULT NULL,
  `MIN_Q` int DEFAULT NULL,
  `UNIT_PRICE` int DEFAULT NULL,
  PRIMARY KEY (`FOOD_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

2. Creation of table order_date

```
CREATE TABLE `order_date` (
  `ORDER_ID` int NOT NULL,
  `DATE1` date DEFAULT NULL,
  `TIME` int DEFAULT NULL,
  PRIMARY KEY (`ORDER_ID`),
  KEY `DATE1_idx` (`DATE1`),
  KEY `TIME_idx` (`TIME`),
  CONSTRAINT `TIME` FOREIGN KEY (`TIME`) REFERENCES `total_items` (`TIME`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

3. Creation of table orders

CREATE TABLE `orders` (

  `ORDER_ID` int NOT NULL,

  `FOOD_ID` int NOT NULL,

  `QUANTITY` int DEFAULT NULL,

  PRIMARY KEY (`ORDER_ID`,`FOOD_ID`),

  KEY `FOOD_ID_idx` (`FOOD_ID`),

  CONSTRAINT `FOOD_ID` FOREIGN KEY (`FOOD_ID`) REFERENCES `food_items` (`FOOD_ID`),

  CONSTRAINT `ORDER_ID` FOREIGN KEY (`ORDER_ID`) REFERENCES `order_date` (`ORDER_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

4. Creation of table date_table

CREATE TABLE `date_table` (

  `DATE1` date NOT NULL,

  `DAY` int DEFAULT NULL,

  PRIMARY KEY (`DATE1`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

5. Creation of table total_items

CREATE TABLE `total_items` (

  `DATE1` date NOT NULL,

  `TIME` int NOT NULL,

  `ID_1` text,

  `ID_2` text,

  `ID_3` text,

  `ID_4` text,

  PRIMARY KEY (`DATE1`,`TIME`),

  KEY `TIME_idx` (`TIME`),

CONSTRAINT `DATE1` FOREIGN KEY (`DATE1`) REFERENCES `date_table` (`DATE1`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

4.4.3 Insertion of values in the table

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below. INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)

VALUES (value1, value2, value3,... valueN);

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The **SQL INSERT INTO** syntax will be as follows −

INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

VALUES INSERTED INTO food_items:

INSERT INTO ` food_items ` (`FOOD_ID`,`MAX_Q`,`MIN_Q`,`UNIT_PRICE`) VALUES (1,100,2,8);

INSERT INTO ` food_items ` (`FOOD_ID`,`MAX_Q`,`MIN_Q`,`UNIT_PRICE`) VALUES (2,100,2,8);

INSERT INTO ` food_items ` (`FOOD_ID`,`MAX_Q`,`MIN_Q`,`UNIT_PRICE`) VALUES (3,5000,100,20);

INSERT INTO ` food_items ` (`FOOD_ID`,`MAX_Q`,`MIN_Q`,`UNIT_PRICE`) VALUES (4,5000,100,20);

VALUES INSERTED INTO order_date :

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (1,'2022-11-25',60);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (2,'2022-11-25',60);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (3,'2022-11-25',60);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (4,'2022-11-25',60);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (5,'2022-11-25',60);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (6,'2022-11-25',60);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (7,'2022-11-25',70);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (8,'2022-11-25',70);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (9,'2022-11-25',70);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (10,'2022-11-25',70);

INSERT INTO ` order_date ` (`ORDER_ID`,`DATE1`,`TIME`) VALUES (11,'2022-11-25',70);

VALUES INSERTED INTO orders:

INSERT INTO `orders` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (1,2,7);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (1,3,200);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (1,4,400);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (2,1,4);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (3,1,4);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (3,3,300);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (4,1,8);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (5,1,2);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (5,2,3);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (6,1,10);

INSERT INTO ` orders ` (`ORDER_ID`,`FOOD_ID`,`QUANTITY`) VALUES (6,3,1100);

VALUES INSERTED INTO date_table:

INSERT INTO ` date_table ` (`DATE1`,`DAY`) VALUES ('2022-11-25',6);

INSERT INTO ` date_table ` (`DATE1`,`DAY`) VALUES ('2022-11-26',7);

INSERT INTO ` date_table ` (`DATE1`,`DAY`) VALUES ('2022-11-27',1);

INSERT INTO ` date_table ` (`DATE1`,`DAY`) VALUES ('2022-11-28',2);

INSERT INTO ` date_table ` (`DATE1`,`DAY`) VALUES ('2022-11-29',3);

INSERT INTO ` date_table ` (`DATE1`,`DAY`) VALUES ('2022-11-30',4);\

4.4.4 TRIGGERS

A trigger is a stored procedure in the database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Trigger to trigger total_items table whenever user gets inserted to orders table:
CREATE DEFINER=`root`@`localhost` TRIGGER `orders_AFTER_INSERT` AFTER INSERT ON `orders` FOR EACH ROW BEGIN

call uall();

END

Trigger to trigger date_tabel table whenever user gets inserted to order_date table:
CREATE DEFINER=`root`@`localhost` TRIGGER `order_date_AFTER_INSERT` AFTER INSERT ON `order_date` FOR EACH ROW BEGIN

insert into `date_table` (DATE1) values (date1);

END

Trigger to trigger date_tabel table whenever user gets inserted to DATE1 value:
CREATE DEFINER=`root`@`localhost` TRIGGER `date_table_AFTER_INSERT` AFTER INSERT ON `date_table` FOR EACH ROW BEGIN

call calldate_table();

END

4.4.5 STORED PROCEDURES :

CREATE DEFINER=`root`@`localhost` PROCEDURE `u1`()

BEGIN

UPDATE total_items M

      JOIN orders N ON N.FOOD_ID=1

      JOIN order_date O ON M.DATE1=O.DATE1 AND M.TIME=O.TIME

SET M.ID_1=(SELECT SUM(X.QUANTITY)

      FROM orders X,order_date Y

      WHERE X.FOOD_ID=1 AND

X.ORDER_ID=Y.ORDER_ID AND

Y.DATE1=M.DATE1 AND

Y.TIME=M.TIME)

WHERE N.ORDER_ID=O.ORDER_ID AND

M.TIME=O.TIME;

END

2. CREATE DEFINER=`root`@`localhost` PROCEDURE `u2`()

CREATE DEFINER=`root`@`localhost` PROCEDURE `u2`()

BEGIN

UPDATE total_items M

      JOIN orders N ON N.FOOD_ID=2

      JOIN order_date O ON M.DATE1=O.DATE1 AND M.TIME=O.TIME

SET M.ID_2=(SELECT SUM(X.QUANTITY)

      FROM orders X,order_table Y

      WHERE X.FOOD_ID=2 AND

X.ORDER_ID=Y.ORDER_ID AND

Y.DATE1=M.DATE1 AND

Y.TIME=M.TIME)

WHERE N.ORDER_ID=O.ORDER_ID AND

M.TIME=O.TIME;

END

3. CREATE DEFINER=`root`@`localhost` PROCEDURE `u3`()

CREATE DEFINER=`root`@`localhost` PROCEDURE `u3`()

BEGIN

UPDATE toatal_items M

     JOIN orders N ON N.FOOD_ID=3

     JOIN order_table O ON M.DATE1=O.DATE1 AND M.TIME=O.TIME

SET M.ID_3=(SELECT SUM(X.QUANTITY)

    FROM orders X,order_tabel Y

    WHERE X.FOOD_ID=3 AND

X.ORDER_ID=Y.ORDER_ID AND

Y.DATE1=M.DATE1)

WHERE N.ORDER_ID=O.ORDER_ID AND

M.TIME=O.TIME

;

END

4. CREATE DEFINER=`root`@`localhost` PROCEDURE `u4`()

CREATE DEFINER=`root`@`localhost` PROCEDURE `u4`()

BEGIN

UPDATE total_items M

     JOIN orders N ON N.FOOD_ID=4

     JOIN order_table O ON M.DATE1=O.DATE1 AND M.TIME=O.TIME

SET M.ID_4=(SELECT SUM(X.QUANTITY)

    FROM orders X,order_table Y

    WHERE X.FOOD_ID=4 AND

X.ORDER_ID=Y.ORDER_ID AND

Y.DATE1=M.DATE1)

WHERE N.ORDER_ID=O.ORDER_ID AND

M.TIME=O.TIME

;

END

5. CREATE DEFINER=`root`@`localhost` PROCEDURE `UALL`()

BEGIN

CALL u1();

CALL u2();

CALL u3();

CALL u4();

END

6. CREATE DEFINER=`root`@`localhost` PROCEDURE `callview`()

BEGIN

SELECT * FROM `new_view`;

END

7. CREATE DEFINER=`root`@`localhost` PROCEDURE `calldate_table`()

BEGIN

update `date_table` set DAY=day(DATE1);

END

4.4.6 VIEWS:

CREATE

    ALGORITHM = UNDEFINED

    DEFINER = `root`@`localhost`

    SQL SECURITY DEFINER

VIEW `new_view` AS

    SELECT

        `d`.`DAY` AS `DAY`,

        `l`.`TIME` AS `TIME`,

        `l`.`ID_1` AS `ID_1`,

        `l`.`ID_2` AS `ID_2`,

        `l`.`ID_3` AS `ID_3`,

        `l`.`ID_4` AS `ID_4`

    FROM

        (`date_table` `d`

        JOIN `total_items` `l`)

    WHERE

        (`d`.`DATE1` = `l`.`DATE1`)

CHAPTER 5

SOFTWARE TESTING

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

5.1 Testing Objectives

- Testing is the process of executing a program with the intent of finding an error.
- A good test case design is one that has a probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.
- These above objectives imply a dramatic change in view port.
- Testing cannot show the absence of defects, it can only show that software errors are present

5.2 Testing Principles

- All tests should be traceable to end user requirements.
- Tests should be planned long before testing begins.
- Testing should begin on a small scale and progress towards testing in large.
- Exhaustive testing is not possible.

5.3 Validation Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed in an

appropriate environment.

5.4 Unit Testing

In this test the entire system is divided into smaller entities and each entity is known as a unit and each unit is tested one at a time. The test ensures the working of each module as per the requirement in functionality point of view rather than actual module visible to the user.

5.5 Integration Testing

In this test each unit is looped together to form a long and continuous chain of modules till it does not cover the entire system. The test checks the behavior of the interface. This test is generally performed after a unit test.

5.6 Test Cases

The test cases considered are shown in the Table 5.6

Table 5.6: Test cases

Test No.	Test Name	Actual Output	Expected Output	Result
1.	Accessing database	Connection established	Connection established	Pass
2.	Reading the created view	Display view	Display view	Pass
3.	Fitting a linear model	A model has been fitted	A model has been fitted	Pass
4.	Prediction	Values are predicted	Values are predicted	Pass
5.	Actual consumption	Not accurate but close	Not accurate but close	Pass

5.7 Regression Testing

This test is performed after the integrating test where the system is checked if it gives the desired output for the expected input data from the user.

The view that has been called from jupyter notebook and the output of it is shown in the Figure 5.1.

```
(251122, 60, '4', 28, 21, 5600, 5150)
(251122, 70, '4', 24, 43, 5600, 5150)
(251122, 80, '4', 34, 101, 5600, 5150)
(251122, 90, '4', 43, 41, 5600, 5150)
(261122, 60, '5', 19, 15, 4450, 3850)
(261122, 70, '5', 16, 25, 4450, 3850)
(261122, 80, '5', 43, 73, 4450, 3850)
(261122, 90, '5', 20, 31, 4450, 3850)
(271122, 60, '6', 20, 26, 4900, 4550)
(271122, 70, '6', 36, 21, 4900, 4550)
(271122, 80, '6', 73, 37, 4900, 4550)
(271122, 90, '6', 12, 3, 4900, 4550)
(281122, 60, '0', 19, 12, 6050, 6800)
(281122, 70, '0', 47, 22, 6050, 6800)
(281122, 80, '0', 36, 88, 6050, 6800)
(281122, 90, '0', 49, 25, 6050, 6800)
(291122, 60, '1', 19, 7, 5600, 4100)
(291122, 70, '1', 16, 23, 5600, 4100)
(291122, 80, '1', 62, 43, 5600, 4100)
(291122, 90, '1', 20, 17, 5600, 4100)
(301122, 60, '2', 24, 31, 6700, 7350)
(301122, 70, '2', 19, 32, 6700, 7350)
(301122, 80, '2', 54, 52, 6700, 7350)
...
(31222, 60, '5', 45, 22, 6000, 5800)
(31222, 70, '5', 19, 6, 6000, 5800)
(31222, 80, '5', 71, 17, 6000, 5800)
(31222, 90, '5', 14, 28, 6000, 5800)
```

Figure 5.1: View

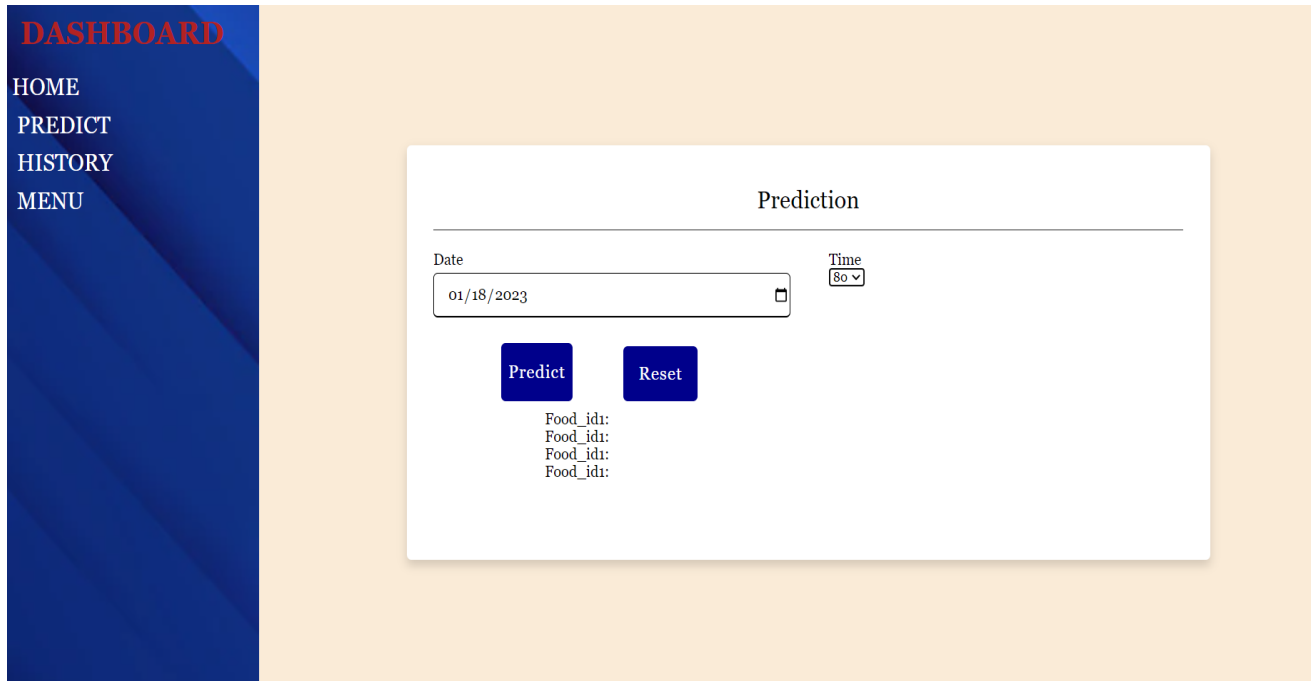Dataframe read from the database is shown in the Figure 5.2.

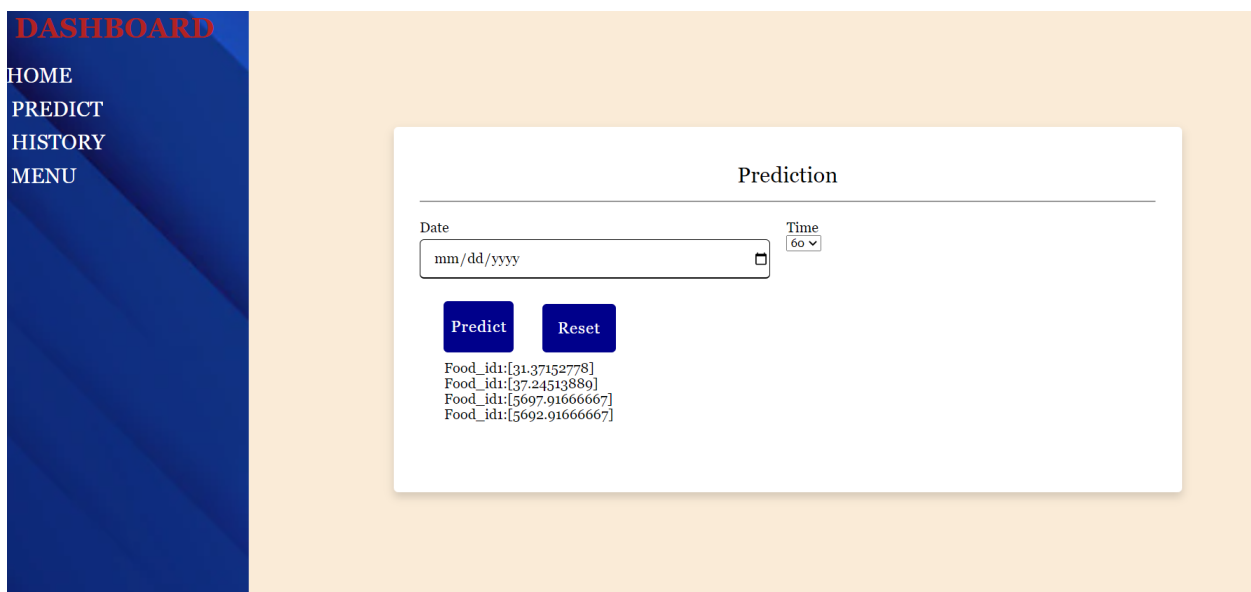|	DATE	TIME	DAY	ID_1	ID_2	ID_3	ID_4
0	251122	60	4	28	21	5600	5150
1	251122	70	4	24	43	5600	5150
2	251122	80	4	34	101	5600	5150
3	251122	90	4	43	41	5600	5150
4	261122	60	5	19	15	4450	3850
5	261122	70	5	16	25	4450	3850
6	261122	80	5	43	73	4450	3850
7	261122	90	5	20	31	4450	3850
8	271122	60	6	20	26	4900	4550
9	271122	70	6	36	21	4900	4550
10	271122	80	6	73	37	4900	4550
11	271122	90	6	12	3	4900	4550
12	281122	60	0	19	12	6050	6800
13	281122	70	0	47	22	6050	6800
14	281122	80	0	36	88	6050	6800
15	281122	90	0	49	25	6050	6800
16	291122	60	1	19	7	5600	4100
17	291122	70	1	16	23	5600	4100
18	291122	80	1	62	43	5600	4100
19	291122	90	1	20	17	5600	4100
20	301122	60	2	24	31	6700	7350
21	301122	70	2	19	32	6700	7350
22	301122	80	2	54	52	6700	7350
23	301122	90	2	19	39	6700	7350

Figure 5.2: Dataframe

A linear regression model has been fitted to it as shown in the Figure 5.3.
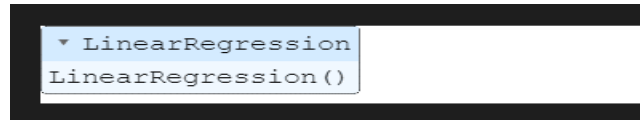
Figure 5.3: linear regression model

The input page where the parameters are prompted is shown in the Figure 5.4

Figure 5.4: Input page

The output page where the predictions are displayed in the Figure 5.5

Figure 5.5: Output page

CHAPTER 6

SNAPSHOTS

The input page where the parameters are prompted is shown in the Figure 6.1

Figure 6.1:Input page

The output page where the predictions are displayed in the Figure 6.2.

Figure 6.2: Output page

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

Food is a essential need for human and It should never go to waste. In this project we used both machine learning and database management to get results. Even though these results are not accurate but it is still a close race.

Even though the actual consumption depends upon the parameters which can't be quantified, the restaurant owner can make use of this data as a good reference and make the necessary accommodations to get the optimal result.

As the data are being collected for the prediction process, it makes the restaurant more organized. They also get to keep track of the orders and also holds good for inventory management

7.3 Further enhancement

With each passing the system collects more data about the business and with each passing day the predicted values will be more accurate. In this project only two parameters are being considered, but if other parameters like holidays, festive season, cricket season etc. are taken the predicted values will be very much close to the actual values. If we take into account the error in the predicted values as a parameter it can further enhance its prediction.

REFERENCES

[1]. www.ibm.com Working of Linear Regression

[2]. www.geeksforgeeks.org Python packages

[3]. www.tutorialspoint.com Flask concepts