Created by: Bryan Loo Chun Wai

Databases and Advanced Data Techniques

Mid Term Project Report

Introduction

The project focuses on analysing a dataset, creating a normalized relational database, and developing a simple web application to interact with the data. This report covers the dataset critique, E/R modelling, database implementation, and web application creation.

The chosen dataset is "Supermart Grocery Sales Retail Analytics Dataset." It was selected because it provides detailed insights into retail sales, enabling the exploration of sales patterns, profit trends, and regional performance through a database-driven approach.

The research questions derived from the dataset are designed to provide actionable insights into sales and operational trends, emphasizing the impact of discounts and performance variations across different regions.

This project aims to showcase the ability of relational databases to transform raw data into meaningful business intelligence, justifying its practical significance for retail analytics.

Stage 1 - Dataset Analysis and Critique

Dataset Overview

Link:

https://www.kaggle.com/datasets/mohamedharris/supermart-grocery-sales-retail-analytics-dataset

Description:

This fictional dataset was created to support exploratory data analysis and data visualization. It contains data on orders placed by customers in Tamil Nadu, India, via a grocery delivery application.

Dataset Assessment

Quality

The dataset contains 1000+ entries with well-structured fields. Missing values are minimal, and outliers are limited, ensuring reliability for analysis.

Level of Detail

Provides granular information, including customer details, order date, product categories, subcategories, financial metrics (sales, discounts, and profits), and geographic data (city and region).

Documentation

The dataset's accompanying documentation explains its structure, attribute definitions, and assumptions clearly, making it easy to interpret.

Interrelation

Key attributes such as "OrderID," "City," and "Category" establish relational connections, making the dataset suitable for a normalized database structure.

Use and Discoverability

The dataset is publicly accessible on Kaggle, with a Creative Commons license that facilitates use in academic projects.

Overall, the dataset is highly appropriate for this coursework due to its comprehensiveness, relational potential, and practical applicability in a business context.

Research Questions

The database application aims to address the following questions:

1. Which product categories contribute the most to sales and profits?
2. How do sales and profits vary across cities and regions?
3. What is the impact of discounts on overall profit margins?

These questions justify the need for a relational database and demonstrate its advantages over a flat-file structure.

Stage 2 - Data Modelling

Entity Relationship Model

The E/R model identifies key entities and their relationships in the dataset. The primary entities include:

Orders

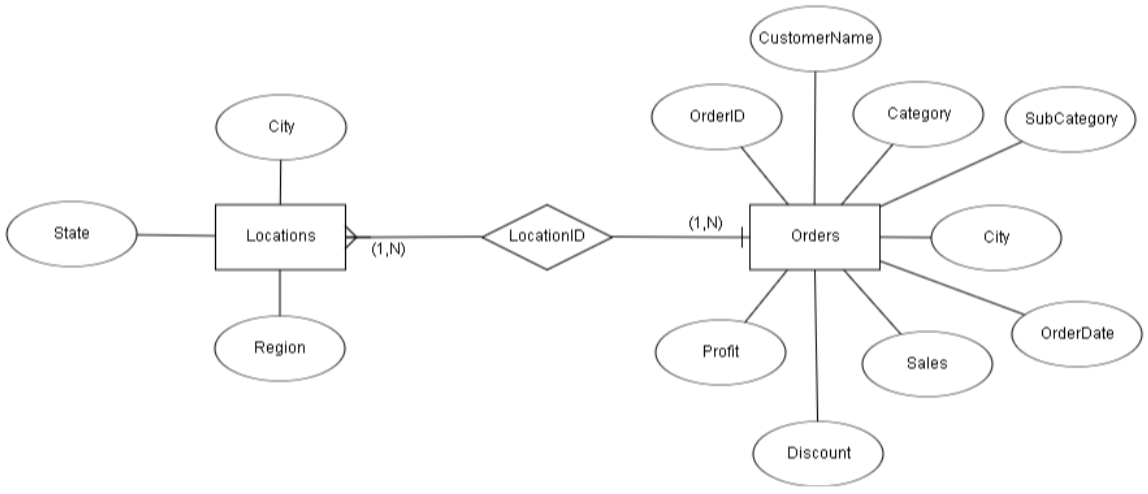
Represents individual transactions, including details such as order ID, customer name, and sales figures.

Locations

Stores geographic details such as city, state, and region.

Key relationships

Each order is placed in a specific location, creating a one-to-many relationship between "Locations" and "Orders."



Explanation

The diagram follows the standard notation with entities represented as rectangles and relationships as diamonds. Attributes are included as ovals, with primary and foreign keys clearly identified.

Relational Mapping

Orders table schema

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
OrderID	INT	NO	Primary Key	NULL	AUTO_INCREMENT
CustomerName	VARCHAR(100)	YES		NULL	
Category	VARCHAR(50)	YES		NULL	
SubCategory	VARCHAR(255)	YES		NULL	
City	VARCHAR(255)	YES		NULL	
OrderDate	VARCHAR(255)	YES		NULL	
Sales	FLOAT	YES		NULL	
Discount	FLOAT	YES		NULL	
Profit	FLOAT	YES		NULL	
LocationID	INT	YES	MUL	NULL	

Locations table schema

Field	Type	Null	Key	Default	Extra
LocationID	INT	NO	PRIMARY KEY	NULL	AUTO_INCREMENT
City	VARCHAR(100)	NO		NULL	
State	VARCHAR(100)	YES		NULL	
Region	VARCHAR(50)	NO		NULL	

Normalization

1NF (First Normal Form)

1NF Characteristics

- Each column contains atomic values (no repeating groups or arrays).
- Each row is unique, typically with a primary key (e.g., OrderID).

Your initial CSV dataset satisfies 1NF because:

- All data values are atomic.
- Each row can be uniquely identified by the OrderID column.
- These were the columns for the initial CSV file:

Order ID, Customer Name, Category, Sub Category, City, Order Date, Region, Sales, Discount, Profit, State

2NF (Second Normal Form)

2NF Characteristics:

- Satisfies 1NF.
- Removes partial dependencies: All non-key attributes must depend on the whole primary key, not just part of it.

In this case:

- In the initial CSV dataset, columns like State and Region depend only on City and not on Order ID, the primary key.
- To eliminate partial dependencies, we split the dataset into two tables:

Orders Table

- Stores details about the order.
- Columns: OrderID, CustomerName, Category, SubCategory, City, OrderDate, Sales, Discount, Profit, and a foreign key, LocationID linking to the Locations table.

Locations Table

- Stores location-specific information.
- Columns: LocationID, City, State, Region.

By splitting, we isolate location-specific attributes (State and Region) into the Locations table, where they depend only on City.

3NF (Third Normal Form)

3NF Characteristics:

- Satisfies 2NF.
- Removes transitive dependencies: Non-key attributes must depend only on the primary key and not on other non-key attributes.

In this case:

- In the Orders Table, all attributes are directly dependent on the primary key OrderID or the foreign key LocationID. No transitive dependencies exist.
- In the Locations Table, State and Region are dependent only on the primary key LocationID. No transitive dependencies exist here either.

Thus, the data structure now satisfies 3NF.

Database Implementation

Database Creation

The MySQL database structure was implemented using the following CREATE commands:

```
CREATE TABLE Orders (
  OrderID INT AUTO_INCREMENT PRIMARY KEY,
  CustomerName VARCHAR(100),
  Category VARCHAR(50),
  SubCategory VARCHAR(255),
  City VARCHAR(255),
  OrderDate VARCHAR(255),
  Sales FLOAT,
  Discount FLOAT,
  Profit FLOAT,
  LocationID INT,
  FOREIGN KEY (LocationID) REFERENCES Locations(LocationID)
);

CREATE TABLE Locations (
  LocationID INT AUTO_INCREMENT PRIMARY KEY,
  City VARCHAR(100) NOT NULL,
  State VARCHAR(100),
  Region VARCHAR(50) NOT NULL
);
```

Instance Data

Sample data entries were imported into the database using CSV imports and manual entry methods.

Critical Reflection

The database implementation successfully translates the E/R model into a relational structure. Key strengths include:

1) Accurate representation of relationships between orders and locations.
2) Efficient storage and retrieval mechanisms enabled by normalization.

Challenges encountered include:

Data Cleaning

Addressing inconsistencies in location names required additional preprocessing.

Missing Data

Certain orders lacked city information, requiring imputation or exclusion during import.

Further thinking:

Future improvements could involve automating data cleaning processes and integrating more robust validation mechanisms during import.

SQL Queries

```
// Loading orders data
LOAD DATA INFILE '/home/coder/project/Orders_Cleaned.csv'
INTO TABLE Orders
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS (CustomerName, Category, SubCategory, City, OrderDate, Sales, Discount,
Profit);

// Loading locations data
LOAD DATA INFILE '/home/coder/project/Locations_Cleaned.csv'
INTO TABLE Orders
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS (City, State, Region);

// Updating the locationID column with mapped locationID from location table
UPDATE Orders o JOIN Locations l ON o.City = l.City SET o.LocationID = l.LocationID;

// Insert a new order
```

```
INSERT INTO Orders (CustomerName, Category, SubCategory, City, OrderDate, Sales,
Discount, Profit)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)

// Update an existing order
UPDATE Orders SET CustomerName = ?, Category = ?, SubCategory = ?, City = ?, OrderDate =
?, Sales = ?, Discount = ?, Profit = ? WHERE OrderID = ?

// Delete order
DELETE FROM Orders WHERE OrderID = ?

// Fetch dashboard statistics
SELECT SUM(Sales) AS totalSales, AVG(Discount) AS avgDiscount, SUM(Profit) AS totalProfit
FROM Orders

// Fetch city report data
SELECT City, SUM(Sales) AS totalSales, SUM(Profit) AS totalProfit, AVG(Discount) AS
avgDiscount FROM Orders GROUP BY City

// Fetch category report data
SELECT Category, SUM(Sales) AS totalSales, SUM(Profit) AS totalProfit, AVG(Discount) AS
avgDiscount FROM Orders GROUP BY Category

// Fetch the first 10 orders
SELECT * FROM Orders ORDER BY OrderID ASC LIMIT 10

// Fetch the last 10 orders
SELECT * FROM Orders ORDER BY OrderID DESC LIMIT 10

// Search orders by orderID
SELECT * FROM Orders WHERE OrderID = ?
// Search orders by customerName
SELECT * FROM Orders WHERE CustomerName LIKE ?

// Search orders by both customerName and OrderID
SELECT * FROM Orders WHERE OrderID = ? AND CustomerName LIKE ?

// Identify top-performing categories:
SELECT Category, SUM(Sales) AS TotalSales, SUM(Profit) AS TotalProfit
FROM Orders
GROUP BY Category
ORDER BY TotalProfit DESC;
```

Stage 4 - Web Application

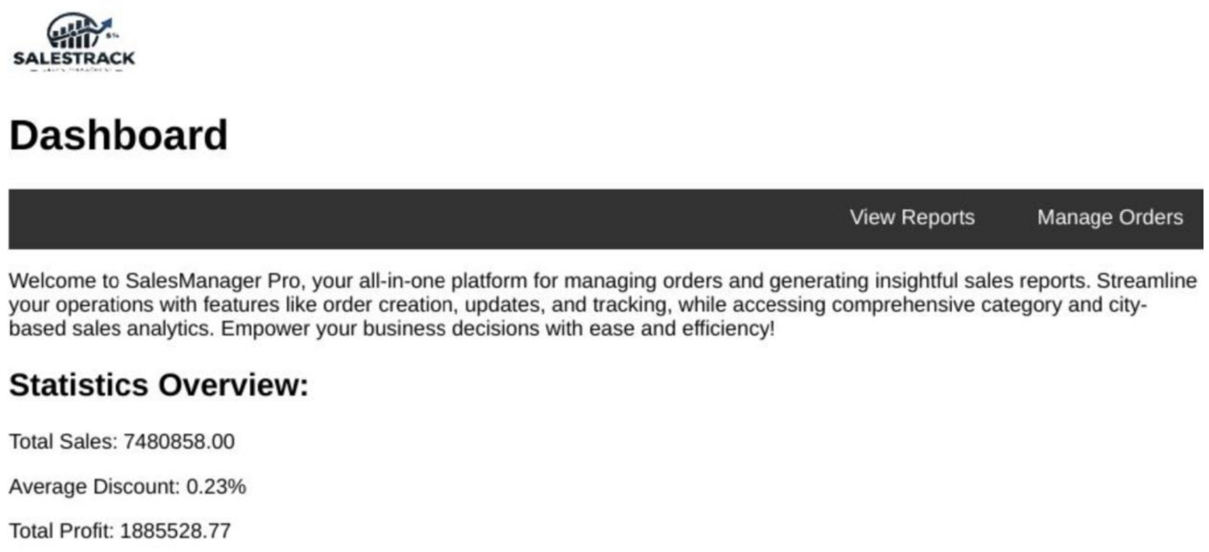
Application Overview

The web application, built using Node.js, provides the following features:

- Display sales and profit data by category.
- Generate reports by city or region.
- Add, update, and delete order records.

Screenshots

1) Home page



Dashboard

View Reports | Manage Orders

Welcome to SalesManager Pro, your all-in-one platform for managing orders and generating insightful sales reports. Streamline your operations with features like order creation, updates, and tracking, while accessing comprehensive category and city-based sales analytics. Empower your business decisions with ease and efficiency!

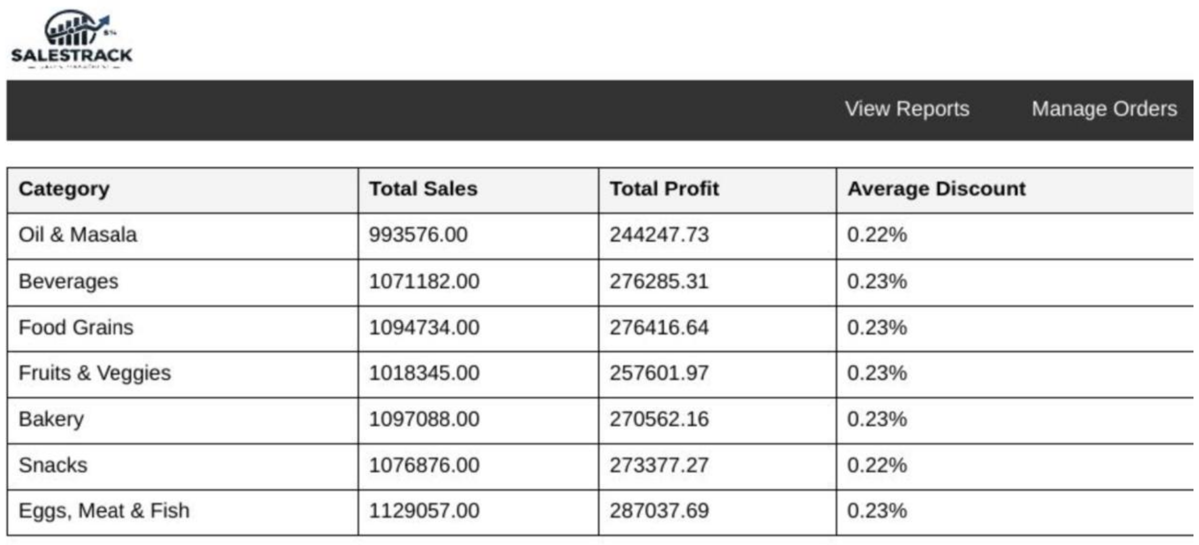
Statistics Overview:

Total Sales: 7480858.00

Average Discount: 0.23%

Total Profit: 1885528.77

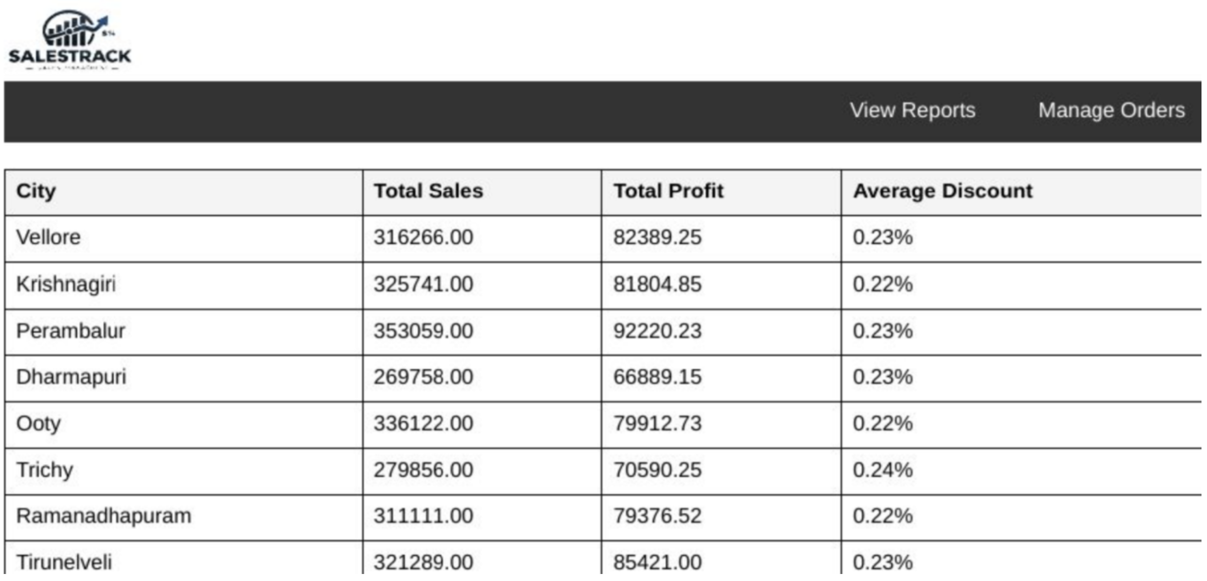
2) Category report page



View Reports | Manage Orders

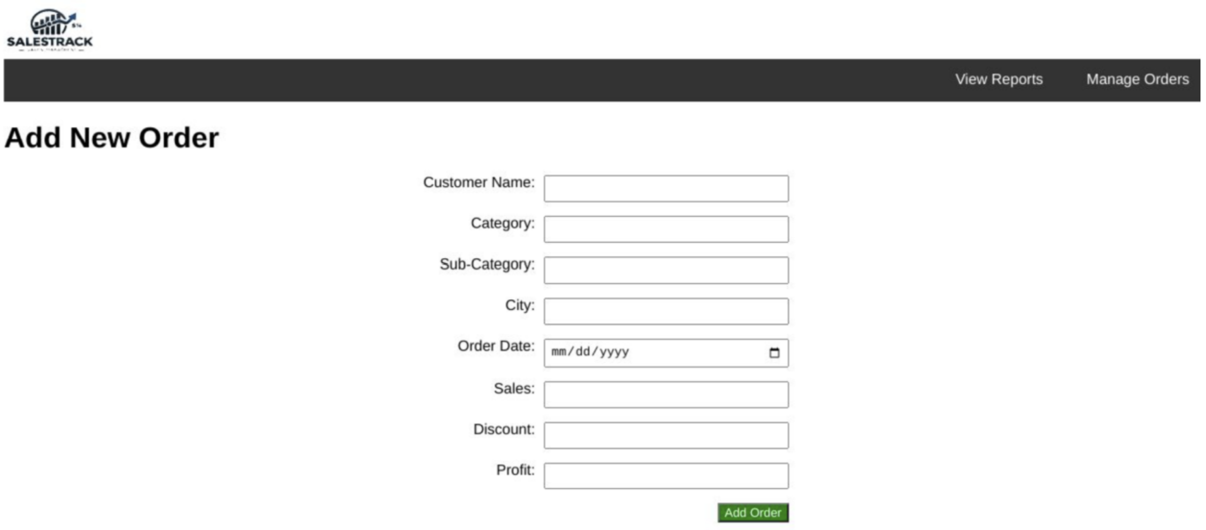
Category	Total Sales	Total Profit	Average Discount
Oil & Masala	993576.00	244247.73	0.22%
Beverages	1071182.00	276285.31	0.23%
Food Grains	1094734.00	276416.64	0.23%
Fruits & Veggies	1018345.00	257601.97	0.23%
Bakery	1097088.00	270562.16	0.23%
Snacks	1076876.00	273377.27	0.22%
Eggs, Meat & Fish	1129057.00	287037.69	0.23%

3) City report page


View Reports
Manage Orders

City	Total Sales	Total Profit	Average Discount
Vellore	316266.00	82389.25	0.23%
Krishnagiri	325741.00	81804.85	0.22%
Perambalur	353059.00	92220.23	0.23%
Dharmapuri	269758.00	66889.15	0.23%
Ooty	336122.00	79912.73	0.22%
Trichy	279856.00	70590.25	0.24%
Ramanadhapuram	311111.00	79376.52	0.22%
Tirunelveli	321289.00	85421.00	0.23%

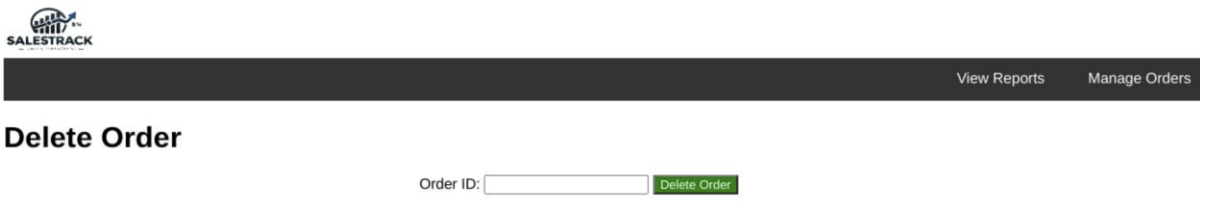
4) Add order page


View Reports
Manage Orders

Add New Order

Customer Name:	
Category:	
Sub-Category:	
City:	
Order Date:	mm/dd/yyyy
Sales:	
Discount:	
Profit:	

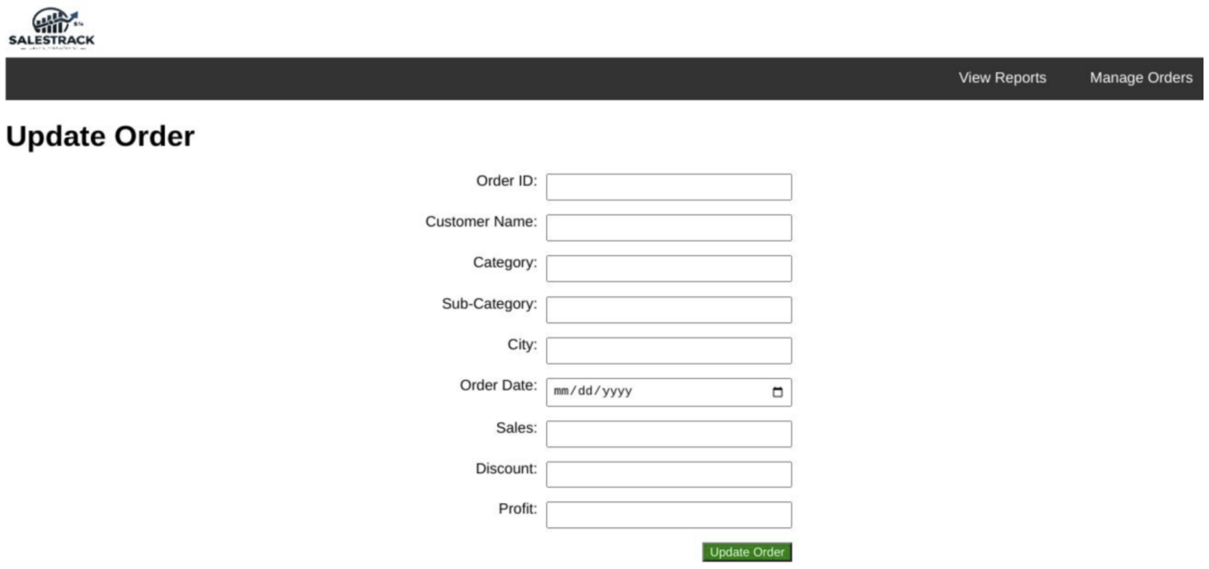
5) Remove order page


View Reports
Manage Orders

Delete Order

Order ID:

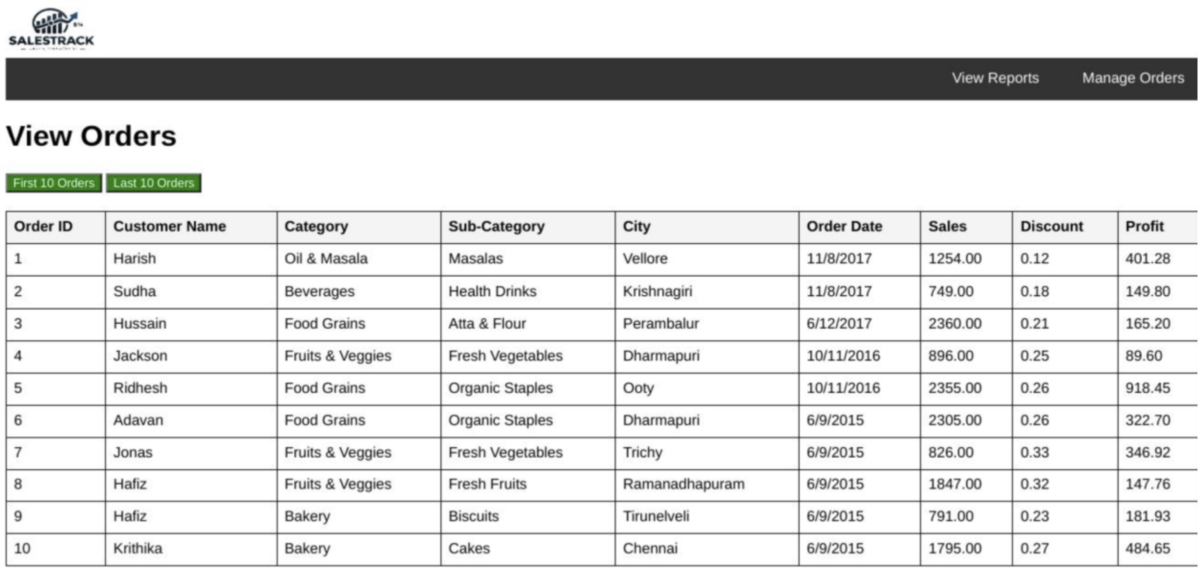
6) Update order page



Update Order

Order ID:	
Customer Name:	
Category:	
Sub-Category:	
City:	
Order Date:	mm/dd/yyyy
Sales:	
Discount:	
Profit:	

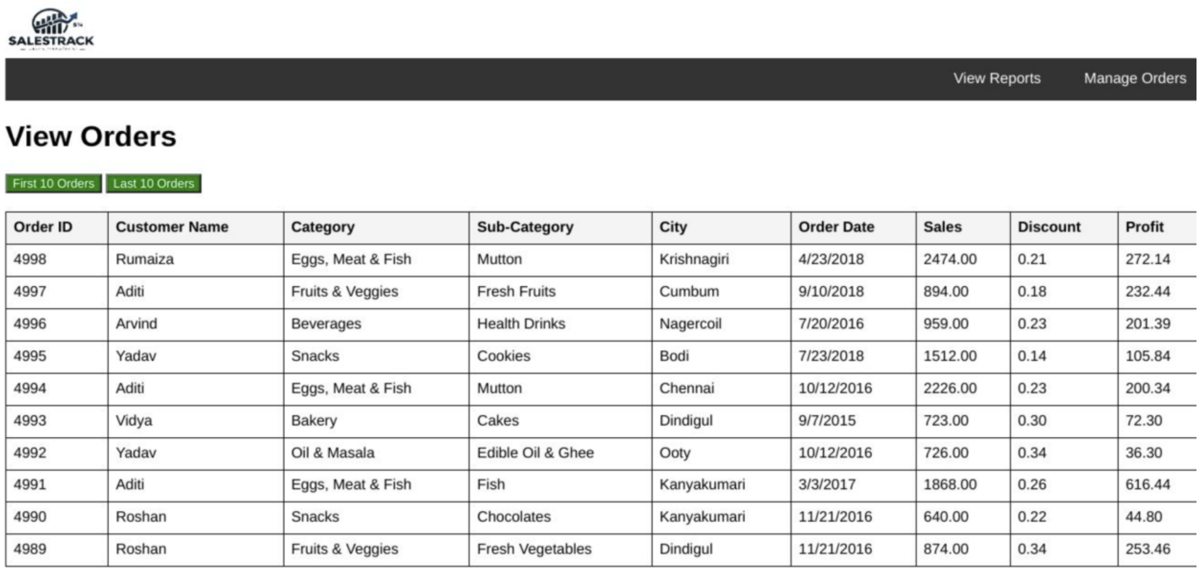
7) View order page (first 10)



View Orders

Order ID	Customer Name	Category	Sub-Category	City	Order Date	Sales	Discount	Profit
1	Harish	Oil & Masala	Masalas	Vellore	11/8/2017	1254.00	0.12	401.28
2	Sudha	Beverages	Health Drinks	Krishnagiri	11/8/2017	749.00	0.18	149.80
3	Hussain	Food Grains	Atta & Flour	Perambalur	6/12/2017	2360.00	0.21	165.20
4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10/11/2016	896.00	0.25	89.60
5	Ridhesh	Food Grains	Organic Staples	Ooty	10/11/2016	2355.00	0.26	918.45
6	Adavan	Food Grains	Organic Staples	Dharmapuri	6/9/2015	2305.00	0.26	322.70
7	Jonas	Fruits & Veggies	Fresh Vegetables	Trichy	6/9/2015	826.00	0.33	346.92
8	Hafiz	Fruits & Veggies	Fresh Fruits	Ramanadhapuram	6/9/2015	1847.00	0.32	147.76
9	Hafiz	Bakery	Biscuits	Tirunelveli	6/9/2015	791.00	0.23	181.93
10	Krithika	Bakery	Cakes	Chennai	6/9/2015	1795.00	0.27	484.65

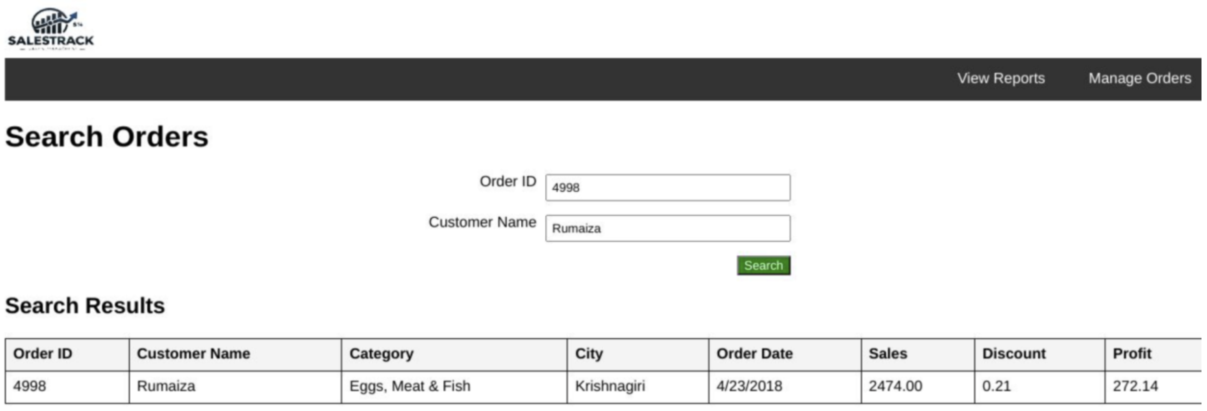
View orders page (Last 10)



View Orders

Order ID	Customer Name	Category	Sub-Category	City	Order Date	Sales	Discount	Profit
4998	Rumaiza	Eggs, Meat & Fish	Mutton	Krishnagiri	4/23/2018	2474.00	0.21	272.14
4997	Aditi	Fruits & Veggies	Fresh Fruits	Cumbum	9/10/2018	894.00	0.18	232.44
4996	Arvind	Beverages	Health Drinks	Nagercoil	7/20/2016	959.00	0.23	201.39
4995	Yadav	Snacks	Cookies	Bodi	7/23/2018	1512.00	0.14	105.84
4994	Aditi	Eggs, Meat & Fish	Mutton	Chennai	10/12/2016	2226.00	0.23	200.34
4993	Vidya	Bakery	Cakes	Dindigul	9/7/2015	723.00	0.30	72.30
4992	Yadav	Oil & Masala	Edible Oil & Ghee	Ooty	10/12/2016	726.00	0.34	36.30
4991	Aditi	Eggs, Meat & Fish	Fish	Kanyakumari	3/3/2017	1868.00	0.26	616.44
4990	Roshan	Snacks	Chocolates	Kanyakumari	11/21/2016	640.00	0.22	44.80
4989	Roshan	Fruits & Veggies	Fresh Vegetables	Dindigul	11/21/2016	874.00	0.34	253.46

8) Search orders page



Search Orders

Order ID: 4998

Customer Name: Rumaiza

Search Results

Order ID	Customer Name	Category	City	Order Date	Sales	Discount	Profit
4998	Rumaiza	Eggs, Meat & Fish	Krishnagiri	4/23/2018	2474.00	0.21	272.14

Critical Reflection

The application meets its objectives effectively, providing a user-friendly interface for interacting with the database. Notable strengths include:

- Seamless CRUD operations with clear feedback for users.
- Real-time data updates and accurate report generation.

Limitations and potential enhancements:

- Data visualization could be improved with graphical charts.
- Advanced filtering and search options would enhance usability for large datasets.

References

Supermart Grocery Sales **-** https://www.kaggle.com/datasets/mohamedharris/supermart-grocery-sales-retail-analytics-dataset

Commands to run the code

Run server:

- node server.js

Run python preprocessors:

- python LocationsPreprocessor.py
- python OrdersPreprocessor.py

Modules and libraries used

- MySQL
- Express
- Pandas

Full code implementation

```
// All code is original and written by me

// HTML files

//===========================================================================
// homepage.html
//===========================================================================

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dashboard</title>
  <link rel="stylesheet" href="/styles/style.css">
  <script defer src="/javascript-renderers/homepage.js"></script>
  <script defer src="/javascript-renderers/api.js"></script>
</head>

<body>
  <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
  <h1>Dashboard</h1>
  <nav id="navigation-bar">
    <div id="navigation-links">
      <div class="dropdown">
        <button class="dropbtn">View Reports</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/category-report">Category Report</a>
          <a href="http://localhost:3000/city-report">City Report</a>
        </div>
      </div>
      <div class="dropdown">
        <button class="dropbtn">Manage Orders</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/add-order">Add Orders</a>
          <a href="http://localhost:3000/delete-order">Remove Orders</a>
          <a href="http://localhost:3000/update-order">Update Orders</a>
          <a href="http://localhost:3000/view-orders">View Orders</a>
          <a href="http://localhost:3000/search-orders">Search Orders</a>
        </div>
      </div>
    </div>
  </nav>
  <main>
    <p>Welcome to SalesManager Pro, your all-in-
one platform for managing orders and generating insightful sales
      reports. Streamline your operations with features like order creation, updates, and track
ing, while
```

```
        accessing comprehensive category and city-
based sales analytics. Empower your business decisions with ease
        and efficiency!</p>
      <h2>Statistics Overview:</h2>
      <p>Total Sales: <span id="totalSales">Loading...</span></p>
      <p>Average Discount: <span id="avgDiscount">Loading...</span></p>
      <p>Total Profit: <span id="totalProfit">Loading...</span></p>
   </main>
</body>

</html>

//==========================================================================
// category-report.html
//==========================================================================

<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Category Report</title>
   <link rel="stylesheet" href="/styles/style.css">
   <script defer src="/javascript-renderers/api.js"></script>
   <script defer src="/javascript-renderers/category-report.js"></script>
</head>

<body>
   <a href="http://localhost:3000/">
      <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
   </a>
   <nav id="navigation-bar">
      <div id="navigation-links">
         <div class="dropdown">
            <button class="dropbtn">View Reports</button>
            <div class="dropdown-content">
               <a href="http://localhost:3000/category-report">Category Report</a>
               <a href="http://localhost:3000/city-report">City Report</a>
            </div>
         </div>
         <div class="dropdown">
            <button class="dropbtn">Manage Orders</button>
            <div class="dropdown-content">
               <a href="http://localhost:3000/add-order">Add Orders</a>
               <a href="http://localhost:3000/delete-order">Remove Orders</a>
               <a href="http://localhost:3000/update-order">Update Orders</a>
               <a href="http://localhost:3000/view-orders">View Orders</a>
               <a href="http://localhost:3000/search-orders">Search Orders</a>
            </div>
         </div>
      </div>
   </nav>
   <main>
      <table id="categoryTable">
         <thead>
            <tr>
               <th>Category</th>
               <th>Total Sales</th>
               <th>Total Profit</th>
               <th>Average Discount</th>
            </tr>
         </thead>
         <tbody>
         </tbody>
      </table>
   </main>
</body>

</html>

//==============================================================================
// city-report.html
//==============================================================================

<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>City Report</title>
   <link rel="stylesheet" href="/styles/style.css">
   <script defer src="/javascript-renderers/api.js"></script>
   <script defer src="/javascript-renderers/city-report.js"></script>
</head>

<body>
   <a href="http://localhost:3000/">
      <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
   </a>
   <nav id="navigation-bar">
      <div id="navigation-links">
         <div class="dropdown">
            <button class="dropbtn">View Reports</button>
            <div class="dropdown-content">
               <a href="http://localhost:3000/category-report">Category Report</a>
               <a href="http://localhost:3000/city-report">City Report</a>
            </div>
         </div>
         <div class="dropdown">
            <button class="dropbtn">Manage Orders</button>
```

```
          <div class="dropdown-content">
            <a href="http://localhost:3000/add-order">Add Orders</a>
            <a href="http://localhost:3000/delete-order">Remove Orders</a>
            <a href="http://localhost:3000/update-order">Update Orders</a>
            <a href="http://localhost:3000/view-orders">View Orders</a>
            <a href="http://localhost:3000/search-orders">Search Orders</a>
          </div>
        </div>
      </div>
    </nav>
    <main>
      <table id="cityTable">
        <thead>
          <tr>
            <th>City</th>
            <th>Total Sales</th>
            <th>Total Profit</th>
            <th>Average Discount</th>
          </tr>
        </thead>
        <tbody>
        </tbody>
      </table>
    </main>
</body>

</html>

//=======================================================================
// add-order.html
//========================================================================

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Add New Order</title>
    <link rel="stylesheet" href="/styles/style.css">
    <script defer src="/javascript-renderers/add-order.js"></script>
</head>

<body>
    <a href="http://localhost:3000/">
      <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
    </a>
    <nav id="navigation-bar">
      <div id="navigation-links">
        <div class="dropdown">
          <button class="dropbtn">View Reports</button>
          <div class="dropdown-content">
            <a href="http://localhost:3000/category-report">Category Report</a>
            <a href="http://localhost:3000/city-report">City Report</a>
          </div>
        </div>
        <div class="dropdown">
          <button class="dropbtn">Manage Orders</button>
          <div class="dropdown-content">
            <a href="http://localhost:3000/add-order">Add Orders</a>
            <a href="http://localhost:3000/delete-order">Remove Orders</a>
            <a href="http://localhost:3000/update-order">Update Orders</a>
            <a href="http://localhost:3000/view-orders">View Orders</a>
            <a href="http://localhost:3000/search-orders">Search Orders</a>
          </div>
        </div>
      </div>
    </nav>
    <main>
      <h1>Add New Order</h1>
      <form id="addOrderForm">
        <div class="form-group">
          <label for="customerName">Customer Name:</label>
          <input type="text" id="customerName" required>
        </div>
        <div class="form-group">
          <label for="category">Category:</label>
          <input type="text" id="category" required>
        </div>
        <div class="form-group">
          <label for="subCategory">Sub-Category:</label>
          <input type="text" id="subCategory" required>
        </div>
        <div class="form-group">
          <label for="city">City:</label>
          <input type="text" id="city" required>
        </div>
        <div class="form-group">
          <label for="orderDate">Order Date:</label>
          <input type="date" id="orderDate" required>
        </div>
        <div class="form-group">
          <label for="sales">Sales:</label>
          <input type="number" id="sales" step="0.01" required>
        </div>
        <div class="form-group">
          <label for="discount">Discount:</label>
          <input type="number" id="discount" step="0.01" required>
        </div>
        <div class="form-group">
          <label for="profit">Profit:</label>
          <input type="number" id="profit" step="0.01" required>
        </div>
        <div id="errorMessage" style="color: red; display: none;"></div>
        <div class="buttonPos">
          <button type="submit">Add Order</button>
        </div>
      </form>
    </main>
</body>
</html>

//==============================================================================
// delete-order.html
//==============================================================================

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Delete Order</title>
  <link rel="stylesheet" href="/styles/style.css">
  <script defer src="/javascript-renderers/delete-order.js"></script>
</head>

<body>
  <a href="http://localhost:3000/">
    <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
  </a>
  <nav id="navigation-bar">
    <div id="navigation-links">
      <div class="dropdown">
        <button class="dropbtn">View Reports</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/category-report">Category Report</a>
          <a href="http://localhost:3000/city-report">City Report</a>
        </div>
      </div>
      <div class="dropdown">
        <button class="dropbtn">Manage Orders</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/add-order">Add Orders</a>
          <a href="http://localhost:3000/delete-order">Remove Orders</a>
          <a href="http://localhost:3000/update-order">Update Orders</a>
          <a href="http://localhost:3000/view-orders">View Orders</a>
          <a href="http://localhost:3000/search-orders">Search Orders</a>
        </div>
      </div>
    </div>
  </nav>
  <main>
```

```
     <h1>Delete Order</h1>
     <form id="deleteOrderForm">
       <label for="orderId">Order ID:</label>
       <input type="text" id="orderId" required>
       <div id="errorMessage" style="color: red; display: none;"></div>
       <button type="submit">Delete Order</button>
     </form>
   </main>
</body>

</html>

//==========================================================================
// update-order.html
//==========================================================================

<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Update Order</title>
   <link rel="stylesheet" href="/styles/style.css">
   <script defer src="/javascript-renderers/update-order.js"></script>
</head>

<body>
   <a href="http://localhost:3000/">
     <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
   </a>
   <nav id="navigation-bar">
     <div id="navigation-links">
       <div class="dropdown">
         <button class="dropbtn">View Reports</button>
         <div class="dropdown-content">
           <a href="http://localhost:3000/category-report">Category Report</a>
           <a href="http://localhost:3000/city-report">City Report</a>
         </div>
       </div>
       <div class="dropdown">
         <button class="dropbtn">Manage Orders</button>
         <div class="dropdown-content">
           <a href="http://localhost:3000/add-order">Add Orders</a>
           <a href="http://localhost:3000/delete-order">Remove Orders</a>
           <a href="http://localhost:3000/update-order">Update Orders</a>
           <a href="http://localhost:3000/view-orders">View Orders</a>
           <a href="http://localhost:3000/search-orders">Search Orders</a>
         </div>
       </div>
     </div>
```

```
  </nav>
  <main>
    <h1>Update Order</h1>
    <form id="updateOrderForm">
      <div class="form-group">
        <label for="orderId">Order ID:</label>
        <input type="text" id="orderId" required>
      </div>
      <div class="form-group">
        <label for="customerName">Customer Name:</label>
        <input type="text" id="customerName" required>
      </div>
      <div class="form-group">
        <label for="category">Category:</label>
        <input type="text" id="category" required>
      </div>
      <div class="form-group">
        <label for="subCategory">Sub-Category:</label>
        <input type="text" id="subCategory" required>
      </div>
      <div class="form-group">
        <label for="city">City:</label>
        <input type="text" id="city" required>
      </div>
      <div class="form-group">
        <label for="orderDate">Order Date:</label>
        <input type="date" id="orderDate" required>
      </div>
      <div class="form-group">
        <label for="sales">Sales:</label>
        <input type="number" id="sales" step="0.01" required>
      </div>
      <div class="form-group">
        <label for="discount">Discount:</label>
        <input type="number" id="discount" step="0.01" required>
      </div>
      <div class="form-group">
        <label for="profit">Profit:</label>
        <input type="number" id="profit" step="0.01" required>
      </div>
      <div id="errorMessage" style="color: red; display: none;"></div>
      <div class="buttonPos">
        <button type="submit">Update Order</button>
      </div>
    </form>
  </main>
</body>

</html>
```

```
//==========================================================================
// view-orders.html
//==========================================================================

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>View Orders</title>
  <link rel="stylesheet" href="/styles/style.css">
  <script defer src="/javascript-renderers/view-orders.js"></script>
</head>

<body>
  <a href="http://localhost:3000/">
    <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
  </a>
  <nav id="navigation-bar">
    <div id="navigation-links">
      <div class="dropdown">
        <button class="dropbtn">View Reports</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/category-report">Category Report</a>
          <a href="http://localhost:3000/city-report">City Report</a>
        </div>
      </div>
      <div class="dropdown">
        <button class="dropbtn">Manage Orders</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/add-order">Add Orders</a>
          <a href="http://localhost:3000/delete-order">Remove Orders</a>
          <a href="http://localhost:3000/update-order">Update Orders</a>
          <a href="http://localhost:3000/view-orders">View Orders</a>
          <a href="http://localhost:3000/search-orders">Search Orders</a>
        </div>
      </div>
    </div>
  </nav>
  <main>
    <h1>View Orders</h1>
    <div>
      <button id="first10Orders">First 10 Orders</button>
      <button id="last10Orders">Last 10 Orders</button>
    </div>
    <table id="ordersTable">
      <thead>
        <tr>
          <th>Order ID</th>
          <th>Customer Name</th>
          <th>Category</th>
          <th>Sub-Category</th>
          <th>City</th>
          <th>Order Date</th>
          <th>Sales</th>
          <th>Discount</th>
          <th>Profit</th>
        </tr>
      </thead>
      <tbody>
      </tbody>
    </table>
  </main>
</body>

</html>

//============================================================================
// search-orders.html
//============================================================================

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Search Orders</title>
  <link rel="stylesheet" href="../styles/style.css">
  <script defer src="../javascript-renderers/search-orders.js"></script>
</head>

<body>
  <a href="http://localhost:3000/">
    <img src="/images/Sales.png" alt="Sales Image" width="100" height="50">
  </a>
  <nav id="navigation-bar">
    <div id="navigation-links">
      <div class="dropdown">
        <button class="dropbtn">View Reports</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/category-report">Category Report</a>
          <a href="http://localhost:3000/city-report">City Report</a>
        </div>
      </div>
      <div class="dropdown">
        <button class="dropbtn">Manage Orders</button>
        <div class="dropdown-content">
          <a href="http://localhost:3000/add-order">Add Orders</a>
          <a href="http://localhost:3000/delete-order">Remove Orders</a>
          <a href="http://localhost:3000/update-order">Update Orders</a>
```

```html
          <a href="http://localhost:3000/view-orders">View Orders</a>
          <a href="http://localhost:3000/search-orders">Search Orders</a>
        </div>
      </div>
    </div>
  </nav>
  <header>
    <h1>Search Orders</h1>
  </header>
  <main>
    <section>
      <form id="searchForm">
        <div class="form-group">
          <label for="orderID">Order ID</label>
          <input type="text" id="orderID" placeholder="Enter Order ID">
        </div>
        <div class="form-group">
          <label for="customerName">Customer Name</label>
          <input type="text" id="customerName" placeholder="Enter Customer Name">
        </div>
        <div class="buttonPos">
          <button type="submit">Search</button>
        </div>
      </form>
    </section>
    <section>
      <h2>Search Results</h2>
      <table id="ordersTable">
        <thead>
          <tr>
            <th>Order ID</th>
            <th>Customer Name</th>
            <th>Category</th>
            <th>City</th>
            <th>Order Date</th>
            <th>Sales</th>
            <th>Discount</th>
            <th>Profit</th>
          </tr>
        </thead>
        <tbody>
        </tbody>
      </table>
    </section>
  </main>
</body>

</html>
```

```
//=============================================================================
// Javascript files

// api.js
//=============================================================================

async function fetchData(url) {
  // Perform a GET request to the specified URL using the Fetch API
  const response = await fetch(url);

  // Check if the response status is not ok
  if (!response.ok) {
    // Throw an error with a message if the request fails
    throw new Error("Failed to fetch data from the server");
  }

  // Parse and return the response body as JSON
  return response.json();
}

//=============================================================================
// homepage.js
//=============================================================================

// Wait until the DOM content is fully loaded before executing the script
document.addEventListener("DOMContentLoaded", async () => {
  // Log a message to indicate that the retrieval of home page data is starting
  console.log("Attempt to retrieve home page data...");

  try {
    // Fetch data for the dashboard from the API endpoint
    const stats = await fetchData("/api/dashboard");

    // Update the DOM elements with the retrieved stats
    // Set the total sales value, formatted to two decimal places
    document.getElementById("totalSales").textContent = stats.totalSales.toFixed(2);

    // Set the average discount value, formatted to two decimal places and appended with a per
centage sign
    document.getElementById("avgDiscount").textContent = stats.avgDiscount.toFixed(2) + "%
";

    // Set the total profit value, formatted to two decimal places
    document.getElementById("totalProfit").textContent = stats.totalProfit.toFixed(2);

    // Log a success message to indicate that the data was retrieved and displayed successfully
    console.log("Home page data retrieved successfully!");
  } catch (error) {
    // Log an error message if there is an issue fetching or processing the data
    console.error("Error loading dashboard stats:", error);
  }
```

```
});

//==========================================================================
// category-report.js
//==========================================================================

// Wait until the DOM content is fully loaded before executing the script
document.addEventListener("DOMContentLoaded", async () => {
  try {
    // Fetch data from the API endpoint for category reports
    const categories = await fetchData("/api/category-report");

    // Select the table body element where rows will be appended
    const tableBody = document.getElementById("categoryTable").querySelector("tbody");

    // Iterate through each category object received from the API
    categories.forEach(category => {
      // Create a new table row element
      const row = document.createElement("tr");

      // Set the inner HTML of the row with category data
      // Format numeric values to two decimal places
      row.innerHTML = `
        <td>${category.Category}</td>  <!-- Category name -->
        <td>${category.totalSales.toFixed(2)}</td> <!-- Total sales, formatted to 2 decimals -->
        <td>${category.totalProfit.toFixed(2)}</td> <!-- Total profit, formatted to 2 decimals -->
        <td>${category.avgDiscount.toFixed(2)}%</td> <!-- Average discount as a percentage -->
      `;

      // Append the newly created row to the table body
      tableBody.appendChild(row);
    });
  } catch (error) {
    // Log an error message to the console if fetching or rendering data fails
    console.error("Error loading category report:", error);
  }
});

//==========================================================================
// city-report.js
//==========================================================================

// Wait for the DOM content to be fully loaded before executing the script
document.addEventListener("DOMContentLoaded", async () => {
  try {
    // Fetch city report data from the API
    const cities = await fetchData("/api/city-report");

    // Select the table body element where rows will be dynamically added
    const tableBody = document.getElementById("cityTable").querySelector("tbody");
```

```
   // Loop through the array of city data received from the API
   cities.forEach(city => {
    // Create a new table row element for each city
    const row = document.createElement("tr");

    // Populate the row with city data using template literals
    // Format numerical values to two decimal places
    row.innerHTML = `
     <td>${city.City}</td> <!-- City name -->
     <td>${city.totalSales.toFixed(2)}</td> <!-- Total sales for the city -->
     <td>${city.totalProfit.toFixed(2)}</td> <!-- Total profit for the city -->
     <td>${city.avgDiscount.toFixed(2)}%</td> <!-- Average discount as a percentage -->
    `;

    // Append the constructed row to the table body
    tableBody.appendChild(row);
   });
  } catch (error) {
   // Log an error message to the console if fetching or rendering data fails
   console.error("Error loading city report:", error);
  }
 });

//==========================================================================
// add-order.js
//==========================================================================

// Get the ID from add order form and listen for a submit request
document.getElementById("addOrderForm").addEventListener("submit", async (e) => {

 // Prevent the default form submission
 e.preventDefault();

 // Collect form data
 const newOrder = {
  customerName: document.getElementById("customerName").value,
  category: document.getElementById("category").value,
  subCategory: document.getElementById("subCategory").value,
  city: document.getElementById("city").value,
  orderDate: document.getElementById("orderDate").value,
  sales: parseFloat(document.getElementById("sales").value),
  discount: parseFloat(document.getElementById("discount").value),
  profit: parseFloat(document.getElementById("profit").value),
 };

 // Reference the error message element
 const errorMessageElement = document.getElementById("errorMessage");

 // Send data to the server
 try {
  const response = await fetch("/api/orders", {
    method: "POST",
    headers: {
     "Content-Type": "application/json",
    },
    body: JSON.stringify(newOrder), // Convert the object to JSON
   });

   // Handle the response
   if (response.ok) {
    // A confirmation message is returned upon successful inserted order
    alert("Order added successfully!");

    // Clear the form
    document.getElementById("addOrderForm").reset();

    // Upon successful inserted order user is redirected back to orders page
    window.location.href = "http://localhost:3000/view-orders";
   } else {
    // Display error message on the page
    const errorData = await response.json();

    // Update the error message in the web page
    errorMessageElement.innerText = errorData.message || "Failed to add order. Please check
the input and try again.";
    errorMessageElement.style.display = "block";
   }
  } catch (error) {
   // Display a generic error message
   console.error("Error adding order:", error);

   // Update the error message in the web page
   errorMessageElement.innerText = "An error occurred while adding the order. Please try agai
n later.";
   errorMessageElement.style.display = "block";
  }
});

//==============================================================================
// delete-order.js
//==============================================================================

// Attach an event listener to the delete order form, triggered when the form is submitted
document.getElementById("deleteOrderForm").addEventListener("submit", async (e) => {
 // Prevent the default form submission behavior to handle it manually with JavaScript
 e.preventDefault();

 // Retrieve the value entered in the order ID input field
 const orderId = document.getElementById("orderId").value;

 // Reference the element where error messages will be displayed
 const errorMessageElement = document.getElementById("errorMessage");
  try {
   // Send a DELETE request to the server to delete the order with the specified ID
   const response = await fetch(` /api/orders/${orderId}` , { method: "DELETE" });

   if (response.ok) {
    // If the response indicates success, clear any existing error messages
    errorMessageElement.style.display = "none";

    // Notify the user with an alert that the order was successfully deleted
    alert("Order deleted successfully!");

    // Redirect the user to the 'View Orders' page
    window.location.href = "http://localhost:3000/view-orders";
   } else {
    // If the response indicates a failure, display an error message on the page
    const errorData = await response.json(); // Attempt to parse the server's error message
    errorMessageElement.innerText = errorData.message || "Failed to delete order. Please che
ck the input and try again.";
    errorMessageElement.style.display = "block"; // Make the error message visible
   }
  } catch (error) {
   // Catch any unexpected errors during the process, such as network issues
   console.error("Error deleting order:", error); // Log the error for debugging purposes

   // Display a generic error message on the page for the user
   errorMessageElement.innerText = "An error occurred while deleting the order. Please try aga
in later.";
   errorMessageElement.style.display = "block"; // Make the error message visible
  }
});

//============================================================================
// update-order.js
//============================================================================

// Attach an event listener to the update order form that triggers when the form is submitted
document.getElementById("updateOrderForm").addEventListener("submit", async (e) => {
  // Prevent the default form submission behavior
  e.preventDefault();

  // Collect the updated order details from the form inputs
  const updatedOrder = {
   orderId: document.getElementById("orderId").value,
   customerName: document.getElementById("customerName").value,
   category: document.getElementById("category").value,
   subCategory: document.getElementById("subCategory").value,
   city: document.getElementById("city").value,
   orderDate: document.getElementById("orderDate").value,
   sales: parseFloat(document.getElementById("sales").value),
   discount: parseFloat(document.getElementById("discount").value),
   profit: parseFloat(document.getElementById("profit").value),
  };

  // Reference the error message element to display errors on the page
  const errorMessageElement = document.getElementById("errorMessage");

  try {
   // Send a PUT request to the server to update the order with the specified details
   const response = await fetch(`/api/orders/${updatedOrder.orderId}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(updatedOrder), // Convert the object to JSON format for the request
   });

   if (response.ok) {
    // Clear any previous error message
    errorMessageElement.style.display = "none";

    // Notify the user with an alert that the order was updated successfully
    alert("Order updated successfully!");

    // Redirect the user to the 'View Orders' page
    window.location.href = "http://localhost:3000/view-orders";
   } else {
    // If the response indicates failure, parse and display the error message
    const errorData = await response.json(); // Attempt to parse the server's error message
    errorMessageElement.innerText = errorData.message || "Failed to update order. Please che
ck the input and try again.";
    errorMessageElement.style.display = "block"; // Make the error message visible
   }
  } catch (error) {
   // Catch any unexpected errors, such as network issues, and log them for debugging
   console.error("Error updating order:", error);

   // Display a generic error message on the page for the user
   errorMessageElement.innerText = "An error occurred while updating the order. Please try ag
ain later.";
   errorMessageElement.style.display = "block"; // Make the error message visible
  }
});

//==========================================================================
// view-orders.js
//==========================================================================

// Function to fetch and display orders from a given endpoint
async function fetchOrders(endpoint) {
  try {
   // Send a GET request to the specified API endpoint
   const response = await fetch(endpoint);
```

```
    // Parse the JSON response containing the orders
    const orders = await response.json();

    // Select the table body element where the orders will be displayed
    const tableBody = document.getElementById("ordersTable").querySelector("tbody");

    // Clear any existing rows in the table to prepare for new data
    tableBody.innerHTML = "";

    // Iterate through each order in the response and create a table row
    orders.forEach(order => {
      // Create a new table row element
      const row = document.createElement("tr");

      // Populate the row with order data using template literals
      // Use bracket notation for fields with spaces or special characters
      // Format numerical fields (Sales, Discount, Profit) to two decimal places
      row.innerHTML = `
        <td>${order["OrderID"]}</td> <!-- Order ID -->
        <td>${order["CustomerName"]}</td> <!-- Customer Name -->
        <td>${order.Category}</td> <!-- Order Category -->
        <td>${order["SubCategory"]}</td> <!-- Subcategory -->
        <td>${order.City}</td> <!-- City -->
        <td>${order["OrderDate"]}</td> <!-- Order Date -->
        <td>${order.Sales.toFixed(2)}</td> <!-- Sales -->
        <td>${order.Discount.toFixed(2)}</td> <!-- Discount -->
        <td>${order.Profit.toFixed(2)}</td> <!-- Profit -->
      `;

      // Append the constructed row to the table body
      tableBody.appendChild(row);
    });
  } catch (error) {
    // Log the error to the console for debugging purposes
    console.error("Error fetching orders:", error);

    // Notify the user with an alert if fetching orders fails
    alert("Failed to fetch orders. Please try again.");
  }
}

// Add an event listener for the "First 10 Orders" button
document.getElementById("first10Orders").addEventListener("click", () => {
  // Fetch and display the first 10 orders using the appropriate API endpoint
  fetchOrders("/api/orders/first10");
});

// Add an event listener for the "Last 10 Orders" button
document.getElementById("last10Orders").addEventListener("click", () => {
  // Fetch and display the last 10 orders using the appropriate API endpoint
  fetchOrders("/api/orders/last10");
```

```
});

//==========================================================================
// search-orders.js
//==========================================================================

// Attach an event listener to the search form that triggers when the form is submitted
document.getElementById("searchForm").addEventListener("submit", async (e) => {
 // Prevent the default form submission behavior to handle it manually with JavaScript
 e.preventDefault();

 // Retrieve the values entered in the order ID and customer name input fields
 const orderID = document.getElementById("orderID").value;
 const customerName = document.getElementById("customerName").value;

 // Initialize an array to build query parameters for the search
 const queryParams = [];

 // If an order ID is provided, add it to the query parameters
 if (orderID) queryParams.push(`orderID=${orderID}`);

 // If a customer name is provided, add it to the query parameters
 if (customerName) queryParams.push(`customerName=${customerName}`);

 // Build the query string by joining query parameters with '&'
 // If no parameters exist, keep the query string empty
 const query = queryParams.length ? `?${queryParams.join("&")}` : "";

 // Send a GET request to the search endpoint with the constructed query string
 const response = await fetch(`/api/orders/search${query}`);

 // Parse the JSON response to get the search results
 const results = await response.json();

 // Select the table body element where the search results will be displayed
 const tableBody = document.getElementById("ordersTable").querySelector("tbody");

 // Clear any existing rows in the table to prepare for new search results
 tableBody.innerHTML = "";

 // Iterate through the results and create a new table row for each order
 results.forEach(order => {
  // Create a new table row element
  const row = document.createElement("tr");

  // Populate the row with order data using template literals
  // Format numerical fields like sales, discount, and profit to two decimal places
  row.innerHTML = `
   <td>${order.OrderID}</td> <!-- Order ID -->
   <td>${order.CustomerName}</td> <!-- Customer Name -->
   <td>${order.Category}</td> <!-- Order Category -->
```

```
    <td>${order.City}</td> <!-- City of the order -->
    <td>${order.OrderDate}</td> <!-- Date of the order -->
    <td>${order.Sales.toFixed(2)}</td> <!-- Total Sales -->
    <td>${order.Discount.toFixed(2)}</td> <!-- Discount -->
    <td>${order.Profit.toFixed(2)}</td> <!-- Profit -->
   `;

   // Append the constructed row to the table body
   tableBody.appendChild(row);
  });
});

//===========================================================================
// server.js
//===========================================================================

const express = require("express");
const mysql = require("mysql");
const app = express();
const path = require("path");

// Middleware to parse JSON and URL-encoded data
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Serve static files from specified directories
app.use("/views", express.static(path.join(__dirname, "views")));
app.use("/javascript-renderers", express.static(path.join(__dirname, "javascript-renderers")));
app.use("/styles", express.static(path.join(__dirname, "styles")));
app.use("/images", express.static(path.join(__dirname, "images")));

// MySQL connection configuration
const db = mysql.createConnection({
 host: "localhost",
 user: "root",
 password: "",
 database: "Sales", // Ensure this matches your database name
});

// Establish the MySQL connection
db.connect((err) => {
 if (err) {
   console.error("Error connecting to MySQL:", err);
   process.exit(1); // Exit process if connection fails
 }
 console.log("MySQL Connected...");
});

// Utility function to serve static HTML files with error handling
const serveFile = (filePath) => (req, res) => {
 res.sendFile(filePath, (err) => {
    if (err) {
      console.error(`Error serving ${filePath}:`, err);
      res.status(500).send("Internal Server Error");
    }
  });
};

// Routes for serving static HTML pages
app.get("/", serveFile(path.join(__dirname, "views", "homepage.html")));
app.get("/category-report", serveFile(path.join(__dirname, "views", "category-report.html")));
app.get("/city-report", serveFile(path.join(__dirname, "views", "city-report.html")));
app.get("/add-order", serveFile(path.join(__dirname, "views", "add-order.html")));
app.get("/delete-order", serveFile(path.join(__dirname, "views", "delete-order.html")));
app.get("/update-order", serveFile(path.join(__dirname, "views", "update-order.html")));
app.get("/view-orders", serveFile(path.join(__dirname, "views", "view-orders.html")));
app.get("/search-orders", serveFile(path.join(__dirname, "views", "search-orders.html")));
app.get("/settings", serveFile(path.join(__dirname, "views", "settings.html")));

// Endpoint to add a new order
app.post("/api/orders", (req, res) => {
  const { customerName, category, subCategory, city, orderDate, sales, discount, profit } = req.
body;

  const sql = `
    INSERT INTO Orders (CustomerName, Category, SubCategory, City, OrderDate, Sales, Disc
ount, Profit)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?)`;

  db.query(sql, [customerName, category, subCategory, city, orderDate, sales, discount, profit
], (err) => {
    if (err) {
      console.error("Error inserting order:", err);
      return res.status(500).send("Failed to add order");
    }
    res.status(201).send("Order added successfully");
  });
});

// Endpoint to update an order
app.put("/api/orders/:id", (req, res) => {
  const { id } = req.params;
  const { customerName, category, subCategory, city, orderDate, sales, discount, profit } = req.
body;

  const sql = `
    UPDATE Orders
    SET CustomerName = ?, Category = ?, SubCategory = ?, City = ?, OrderDate = ?, Sales = ?, D
iscount = ?, Profit = ?
    WHERE OrderID = ?`;
  db.query(sql, [customerName, category, subCategory, city, orderDate, sales, discount, profit
, id], (err) => {
    if (err) {
      console.error("Error updating order:", err);
      return res.status(500).send("Failed to update order");
    }
    res.status(200).send("Order updated successfully");
  });
});

// Endpoint to delete an order
app.delete("/api/orders/:id", (req, res) => {
  const { id } = req.params;

  const sql = "DELETE FROM Orders WHERE OrderID = ?";
  db.query(sql, [id], (err) => {
    if (err) {
      console.error("Error deleting order:", err);
      return res.status(500).send("Failed to delete order");
    }
    res.status(200).send("Order deleted successfully");
  });
});

// API to fetch dashboard statistics
app.get("/api/dashboard", (req, res) => {
  const sql = `
    SELECT SUM(Sales) AS totalSales, AVG(Discount) AS avgDiscount, SUM(Profit) AS totalProfi
t
    FROM Orders`;

  db.query(sql, (err, results) => {
    if (err) {
      console.error("Error fetching dashboard stats:", err);
      return res.status(500).send("Failed to fetch dashboard stats");
    }
    res.status(200).json(results[0]);
  });
});

// API to fetch city-wise sales data
app.get("/api/city-report", (req, res) => {
  const sql = `
    SELECT City, SUM(Sales) AS totalSales, SUM(Profit) AS totalProfit, AVG(Discount) AS avgDis
count
    FROM Orders
    GROUP BY City`;

  db.query(sql, (err, results) => {
    if (err) {
      console.error("Error fetching city report:", err);
    return res.status(500).send("Failed to fetch city report");
   }
   res.status(200).json(results);
  });
});

// API to fetch category-wise sales data
app.get("/api/category-report", (req, res) => {
  const sql = `
   SELECT Category, SUM(Sales) AS totalSales, SUM(Profit) AS totalProfit, AVG(Discount) AS a
vgDiscount
   FROM Orders
   GROUP BY Category`;

  db.query(sql, (err, results) => {
   if (err) {
    console.error("Error fetching category report:", err);
    return res.status(500).send("Failed to fetch category report");
   }
   res.status(200).json(results);
  });
});

// API to fetch the first 10 orders
app.get("/api/orders/first10", (req, res) => {
  const sql = "SELECT * FROM Orders ORDER BY OrderID ASC LIMIT 10";

  db.query(sql, (err, results) => {
   if (err) {
    console.error("Error fetching first 10 orders:", err);
    return res.status(500).send("Failed to fetch first 10 orders");
   }
   res.status(200).json(results);
  });
});

// API to fetch the last 10 orders
app.get("/api/orders/last10", (req, res) => {
  const sql = "SELECT * FROM Orders ORDER BY OrderID DESC LIMIT 10";

  db.query(sql, (err, results) => {
   if (err) {
    console.error("Error fetching last 10 orders:", err);
    return res.status(500).send("Failed to fetch last 10 orders");
   }
   res.status(200).json(results);
  });
});

// API to search orders based on parameters
app.get("/api/orders/search", (req, res) => {
  const { orderID, customerName } = req.query;

  if (!orderID && !customerName) {
   return res.status(400).send("At least one search parameter is required.");
  }

  let sql = "SELECT * FROM Orders";
  const conditions = [];
  const params = [];

  if (orderID) {
   conditions.push("OrderID = ?");
   params.push(orderID);
  }
  if (customerName) {
   conditions.push("CustomerName LIKE ?");
   params.push(`%${customerName}%`);
  }

  if (conditions.length) {
   sql += ` WHERE ${conditions.join(" AND ")}`;
  }

  db.query(sql, params, (err, results) => {
   if (err) {
    console.error("Error searching orders:", err);
    return res.status(500).send("Failed to search orders");
   }
   res.status(200).json(results);
  });
});

// Start the server
const PORT = 3000;
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));

//==========================================================================
// Python preprocessor files

// LocationPreprocessor.py
//==========================================================================

import pandas as pd

# Load the CSV file
df = pd.read_csv('Locations.csv')

# Display basic information
print("Initial Dataset Information:")
print(df.info())
```

```
# Remove duplicate rows
df = df.drop_duplicates()

# Handle missing values in all columns
df.fillna({'City': 'Unknown', 'State': 'Unknown', 'Region': 'Unknown'}, inplace=True)

# Resolve inconsistent 'Region' values for the same city
df['Region'] = df.groupby('City')['Region'].transform(lambda x: x.mode()[0] if not x.mode().emp
ty else x)

# Standardize text formatting
df['City'] = df['City'].str.title()
df['State'] = df['State'].str.title()
df['Region'] = df['Region'].str.title()

# Check if there are any null values left
if df.isnull().any().any():
   print("Warning: Null values are still present in the dataset.")
else:
   print("No null values in the dataset.")

# Save the cleaned dataset
df.to_csv('Locations_Cleaned.csv', index=False)

print("Preprocessing complete. Cleaned data saved as 'Locations_Cleaned.csv'.")

//=======================================================================
// OrdersPreprocessor.py
//=======================================================================

import pandas as pd

# Load the CSV file
df = pd.read_csv('Orders.csv')

# Remove duplicate rows
df = df.drop_duplicates()

# Standardize text formatting for specific columns
text_columns = ['Customer Name', 'Category', 'Sub Category', 'City']
for col in text_columns:
   if col in df.columns:
      df[col] = df[col].str.title()

# Handle missing values for numeric columns
numeric_columns = ['Sales', 'Discount', 'Profit']
for col in numeric_columns:
   if col in df.columns:
      df[col] = df[col].fillna(0)

# Handle missing values for all columns
```

```
for col in df.columns:
   if df[col].isnull().any():
      print(f"Null values found in column: {col}")
      if df[col].dtype == 'object':
         # Fill missing values with 'Unknown' for text columns
         df[col] = df[col].fillna('Unknown')
      else:
         # Fill missing values with 0 for numeric columns
         df[col] = df[col].fillna(0)

# Verify that there are no null values remaining
if df.isnull().any().any():
   print("Warning: Null values are still present in the dataset.")
else:
   print("No null values in the dataset.")

# Save the cleaned dataset
df.to_csv('Orders_Cleaned.csv', index=False)

print("Preprocessing complete. Cleaned data saved as 'Orders_Cleaned.csv'.")

//======================================================================
// Stylesheet

// style.css
//======================================================================

body {
  font-family: Arial, sans-serif;
  margin: 20px;
  padding: 20px;
}

table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
  border-width: 1px;
  border-color: black;
}

th, td {
  border: 1px solid black;
  padding: 8px;
  text-align: left;
}

th {
  background-color: #f4f4f4;
}
```

```
#navigation-bar {
  display: flex;
  justify-content: flex-end;
  background-color: #333;
  overflow: hidden;
}

#navigation-links {
  display: flex;
  gap: 15px;
}

#navigation-links a {
  float: left;
  color: black;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

#navigation-links a:hover {
  background-color: #ddd;
  color: black;
}

.dropdown {
  float: left;
  overflow: hidden;
}

.dropdown .dropbtn {
  font-size: 16px;
  border: none;
  outline: none;
  color: white;
  padding: 14px 16px;
  background-color: inherit;
  font-family: inherit;
  margin: 0;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px rgba(0, 0, 0, 0.2);
  z-index: 1;
}

.dropdown-content a {
 float: none;
 color: black;
 padding: 12px 16px;
 text-decoration: none;
 display: block;
 text-align: left;
}

.dropdown-content a:hover {
 background-color: #ddd;
}

.dropdown:hover .dropdown-content {
 display: block;
}

.dropdown:hover .dropbtn {
 background-color: #ddd;
 color: black;
}

form {
 max-width: 400px;
 margin: 0 auto;
}

.form-group {
 display: flex;
 justify-content: space-between;
 margin-bottom: 15px;
}

.form-group label {
 flex: 1;
 margin-right: 10px;
 text-align: right;
}

.form-group input {
 flex: 2;
 padding: 5px;
}

button {
 background-color: green;
 color: white;
}

.buttonPos {
 display: flex;
 justify-content: flex-end;
```

```
}

.modal {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 1000;
}

.modal-content {
  background-color: white;
  padding: 20px;
  border-radius: 5px;
  width: 80%;
  max-width: 400px;
  text-align: center;
}

.close-btn {
  float: right;
  font-size: 20px;
  cursor: pointer;
  margin-top: -10px;
}

//============================================================================
// package.json
//============================================================================

{
  "name": "project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.3",
    "express": "^4.21.2",
    "mustache-express": "^1.3.2",
    "mysql": "^2.18.1"
  }
}
```