COMP-5000

2024

ASSIGNMENT bRIEF

**BY**

10892938, 10842878, 10882588, 10893087.

Comp 5000 Documentation

10842878, 10882588, 10892938, 10893087

2024-01-12

This Documentation was generated by R-markdown to aid visualisation of our programming process.

The goal of this project is to create an SQL database system in third normal form with a GUI that enables the user to perform a variety of operations including statistical analysis and data manipulation. The external datasets are orders.csv and restaurant\_info.csv respectively; sourced from the University of Plymouth database.

At the beginning of the project, our first action was to load both data-sets into a python session as a variable using the Pandas library.

# Import csv file using the pandas library--------------------------------------------------------------------------------------  
import sqlite3  
import pandas as pd  
from datetime import datetime  
  
# Read Orders CSV file into pandas dataframe  
Orders\_df = pd.read\_csv('Orders.csv')  
  
# Read restaurant\_info CSV file into pandas dataframe  
restaurant\_info\_df = pd.read\_csv('restaurant\_info.csv')

.Using R-Studio, we can view each of the columns below.

## # A tibble: 6 × 14  
## `Order ID` First\_Customer\_Name `Restaurant ID` `Order Date`   
## <chr> <chr> <dbl> <dttm>   
## 1 OD1 Srini 6 2022-01-01 23:15:00  
## 2 OD2 Revandh 13 2022-01-01 19:21:00  
## 3 OD3 David 9 2022-01-01 23:15:00  
## 4 OD4 Selva 4 2022-01-01 20:31:00  
## 5 OD5 Vinny 4 2022-01-01 11:10:00  
## 6 OD6 Dev 16 2022-01-01 14:22:00  
## # ℹ 10 more variables: `Quantity of Items` <dbl>, `Order Amount` <dbl>,  
## # `Payment Mode` <chr>, `Delivery Time Taken (mins)` <dbl>,  
## # `Customer Rating-Food` <dbl>, `Customer Rating-Delivery` <dbl>,  
## # `Credit Card` <dbl>, `Debit Card` <dbl>, `Card provider` <chr>,  
## # Last\_Customer\_Name <chr>

The original *Orders* dataset has 14 columns and 500 rows. It represents all the information relating to food orders from all the resaurants in the restaurant\_info database. The first column OrderID representsthe primary key of the dataset, which labels all the data auto-incrementally. The second column First\_Customer\_Name represents the first names of customers who ordered services from the restaurants. The third column is a foriegn key which references the restaurant\_info dataset. It represents what restaurants each of the orders were taken from. The fourth column which is the Order\_Date column represents what dates the orders were made. The Order\_Amount represents the amount paid for each order presumably in pounds. The payment\_mode represents what method of payment a customer used when purchasing goods from the restaurant. The delivery\_time\_taken shows the how long it took each order to arrive at the customer’s address. The customer\_rating\_food column shows how the customer rated the foodstuff produced by the firm, while the customer\_rating\_delivery column shows how the customer rates the delivery of the food. The credit\_card column shows the serial numbers of the credit card used for a transaction and the debit\_card column shows similar data for credit cards and the Last\_customer\_Name column shows the last name of customers whom purchased a product.

## # A tibble: 6 × 10  
## RestaurantID RestaurantName Cuisine Zone Category Store Manager   
## <dbl> <chr> <chr> <chr> <chr> <dbl> <chr>   
## 1 1 The Cave Hotel Continental Zone B Pro 1 Esther Hosea  
## 2 2 SSK Hotel North Indian Zone D Pro 2 Dolores Dome  
## 3 3 ASR Restaurant South Indian Zone D Ordinary 3 Jacquline S…  
## 4 4 Win Hotel South Indian Zone D Ordinary 4 Anne Mckinl…  
## 5 5 Denver Restaurant Continental Zone D Pro 5 Francisco D…  
## 6 6 Willies French Zone D Pro 6 Bonnie Some…  
## # ℹ 3 more variables: Years\_as\_manager <dbl>, Email <chr>, Address <chr>

While the restaurant\_info data-set has 20 rows. and 10 columns. RestaurantID represents the primary key of the database. The RestaurantName represents the name of the restaurant. The Cuisine represents the geographic origin of food groups served in the restaurant. The Zone represents what Zone the restaurant is located in. The Category column represents what the category of the restaurant is. The Store column is presumably an external ID to a database of stores, however, no other information is given. The manager column shows which manager controls which restaurant.The Years\_as\_manager column shows how long each manager has been a manager. The Email column shows the email address of the manager. The Address column shows the address of the restaurant.

## Data Cleaning

### ‘order’ Table

In the orders table there are at least two missing values. Entry 187 on the OrderID column seems to be missing.

A screenshot of a computer

AI-generated content may be incorrect.

However, according to Köhler et al (2013), the primary key entries of a database cannot be NULL. Given that each entry corresponds to its row number in the ‘OrderID’ column, the NULL value in the 187th entry was replaced with the value ‘OD187’ as shown below.

Orders\_df.iloc[186,0] = "OD187"

In the First\_Customer\_Name column in row 336, there seems to be a missing value. However the same row carries ‘Brooks’ in its last name column and also carries the same credit card number as existing entries labelled ‘Charlie’. This makes it safe to assume that the missing value is “Charlie”. Thus the NULL value was replaced as shown below.

Orders\_df.iloc[335,1] = "Charlie"

In the Payment\_info column, three amendments were made in response to the information given in the CreditCard and DebitCard columns. As such, three NULL values were amended as follows.

Orders\_df.iloc[381,6] = "Cash on Delivery"  
Orders\_df.iloc[486,6] = "Credit Card"  
Orders\_df.iloc[492,6] = "Credit Card"

All other NULL values within the table were filled with zeros in as shown below.

Orders\_df = Orders\_df.fillna(0)

### ‘restaurant\_info’ Table

The restaurant\_info table had missing information for one of its manager’s name, e-mail and experience which was filled in as follows:

restaurant\_info\_df.iloc[7,6] = "Adeleke Daniel"  
restaurant\_info\_df.iloc[7,7] = 5  
restaurant\_info\_df.iloc[7,8] = "dannydave1000@gmail.com"

Manager ‘Clara Owen’ was missing the amount of time spent as manager in her attributes. While Patrick Stacey is missing an email address. These values are filled in as follows.

restaurant\_info\_df.iloc[14,7] = 7  
restaurant\_info\_df.iloc[16,8] = "patrickstacey@gmail.com"

All other missing values were filled in with zeros.

restaurant\_info\_df = restaurant\_info\_df.fillna(0)

## The SQL schema for all the tables before Delivery Staff is added.

A screenshot of a computer

Description automatically generated

### The Manager Table

CREATE TABLE IF NOT EXISTS Manager (  
 ManagerID INTEGER PRIMARY KEY AUTOINCREMENT,  
 ManagerName VARCHAR(255) NOT NULL,  
 YearsAsManager INT NOT NULL,  
 Email VARCHAR(255) NOT NULL  
 );

### The Restaurant Table

CREATE TABLE IF NOT EXISTS restaurant\_info (  
 RestaurantID INTEGER PRIMARY KEY AUTOINCREMENT,  
 RestaurantName VARCHAR(255) NOT NULL,  
 Cuisine VARCHAR(255) NOT NULL,  
 Zone VARCHAR(255) NOT NULL,  
 Category VARCHAR(255) NOT NULL,  
 Store INT NOT NULL,  
 Address VARCHAR(255) NOT NULL,  
 ManagerID INTEGER NOT NULL,  
 StaffID INTEGER NOT NULL,  
 FOREIGN KEY (ManagerID) REFERENCES Manager(ManagerID);

### The Customer Table

CREATE TABLE IF NOT EXISTS Customers (  
 CustomerID INTEGER PRIMARY KEY AUTOINCREMENT,  
 First\_Customer\_Name VARCHAR(255) NOT NULL,  
 Last\_Customer\_Name VARCHAR(255) NOT NULL  
 );

### The Payment Info Table

CREATE TABLE IF NOT EXISTS Payment\_info (  
 PaymentID INTEGER PRIMARY KEY AUTOINCREMENT,  
 PaymentMode VARCHAR(255) NOT NULL,  
 DebitCard REAL NOT NULL,  
 CreditCard REAL NOT NULL,  
 CardProvider VARCHAR(255) NOT NULL  
 );

### The Orders Table

CREATE TABLE IF NOT EXISTS Orders (  
 ID INTEGER PRIMARY KEY AUTOINCREMENT,  
 OrderID VARCHAR(255) NOT NULL,  
 OrderDate DATETIME NOT NULL,  
 Quantity\_of\_Items INT,  
 OrderAmount INT NOT NULL,  
 DeliveryTimeTaken INT NOT NULL,  
 CustomerRatingFood INT NOT NULL,  
 CustomerRatingDelivery INT NOT NULL,  
 RestaurantsID INTEGER NOT NULL,  
 PaymentID INTEGER NOT NULL,   
 StaffID INTEGER NOT NULL,  
 CustomerID INTEGER NOT NULL,  
 FOREIGN KEY (RestaurantsID) REFERENCES restaurant\_info(RestaurantID),  
 FOREIGN KEY (PaymentID) REFERENCES Payment\_info(PaymentID),  
 FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
 );

## Delivery Staff

A screenshot of a computer

Description automatically generated

The values for the delivery staff were input into the data-set as demonstrated below.

delivery\_staff\_data = [  
 ("John", "Doe", "Bike"),  
 ("Jane", "Smith", "Car"),  
 ("Dave", "Mark", "Bike"),  
 ("Bill", "Huber", "Motorbike"),  
 ("Aryna", "Sabalenka", "Car"),  
 ("Gerlad", "Hold", "Car"),  
 ("Adam", "Smith", "Motorbike"),  
 ("Janet", "Rajesh", "Motorbike"),  
 ("Isaac", "Shaw", "Car"),  
 ("Benedict", "Fall", "Bike"),  
 ("Andrew", "Pears", "Bike"),  
 ("William", "Hive", "Bike"),  
 ("Fred", "Short", "Car"),  
 ("Helena", "Gates", "Motorbike"),  
 ("Gaius", "Finn", "Motorbike"),  
 ("Thomas", "Fire", "Car"),  
 ("Vivian", "Geld", "Car"),  
 ("Lucy", "Tor", "Car"),  
 ("James", "Gold", "Bike"),  
 ("David", "Sean", "Motorbike"),  
]  
  
for staff\_record in delivery\_staff\_data:  
 cursor.execute('''  
 INSERT INTO DeliveryStaff (FirstName, LastName, VehicleType)  
 VALUES (?, ?, ?)  
 ''', staff\_record)  
 cursor.execute('SELECT last\_insert\_rowid()')  
 staff\_id = cursor.fetchone()[0]

The schema for all tables affected are shown below.

CREATE TABLE IF NOT EXISTS DeliveryStaff (  
 StaffID INTEGER PRIMARY KEY AUTOINCREMENT,  
 FirstName VARCHAR(255) NOT NULL,  
 LastName VARCHAR(255) NOT NULL,  
 VehicleType VARCHAR(255) NOT NULL  
 );  
   
 CREATE TABLE IF NOT EXISTS Orders (  
 ID INTEGER PRIMARY KEY AUTOINCREMENT,  
 OrderID VARCHAR(255) NOT NULL,  
 OrderDate DATETIME NOT NULL,  
 Quantity\_of\_Items INT,  
 OrderAmount INT NOT NULL,  
 DeliveryTimeTaken INT NOT NULL,  
 CustomerRatingFood INT NOT NULL,  
 CustomerRatingDelivery INT NOT NULL,  
 RestaurantsID INTEGER NOT NULL,  
 PaymentID INTEGER NOT NULL,   
 StaffID INTEGER NOT NULL,  
 CustomerID INTEGER NOT NULL,  
 FOREIGN KEY (RestaurantsID) REFERENCES restaurant\_info(RestaurantID),  
 FOREIGN KEY (PaymentID) REFERENCES Payment\_info(PaymentID),  
 FOREIGN KEY (StaffID) REFERENCES DeliveryStaff(StaffID)  
 FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
 );

The entity relationship diagram after including the Delivery table into the dataset.

Manager Table before the information update  
  
  
A screenshot of a computer

Description automatically generated

Update of manager information at the first row with id 1  
  
A screenshot of a computer

Description automatically generated

Manager Table after the Information update  
  
  
A screenshot of a computer

Description automatically generated

Statistical Calculation of the mean of customer food rating  
  
  
A screenshot of a computer

Description automatically generated

Histogram of delivery time taken (mins)  
  
  
A screenshot of a computer

Description automatically generated

Delivery Staff Table

A screenshot of a delivery service table

Description automatically generated

Customers Table  
  
A screenshot of a computer

Description automatically generated

Payment Info Table  
  
  
A screenshot of a computer

Description automatically generated

Restaurant info Table   
  
A screenshot of a menu

Description automatically generated

Orders Table  
  
  
A screenshot of a computer

Description automatically generated

**Design Choice For The Database**

The design choice for the database is the relational model, which is widely used for its flexibility, simplicity, robust structure and support for operations such as selection, projection and join. The design choices for the database reflect a structured approach to managing a restaurant order system. The database schema is normalized, with separate tables for different entities such as Orders, DeliveryStaff, Customers, Payment\_info, Manager, and restaurant\_info. Each table is created with a primary key that uniquely identifies each record, and foreign keys are used to establish relationships between tables, ensuring referential integrity.

The Orders table is central to the system, capturing essential details about each order, including a unique OrderID, the date, quantity of items, order amount, delivery time taken, and customer ratings for both delivery and food. This table is linked to other tables through foreign keys, connecting orders to specific restaurants, payment methods, staff members, and customers.

The use of VARCHAR and INT data types for fields is appropriate for the data being stored, such as names, vehicle types, and numerical identifiers. The use of AUTOINCREMENT for primary keys simplifies the process of adding new records by automatically generating a unique identifier for each new entry.

The restaurant\_info table includes not only the name and cuisine of the restaurant but also the zone and category, which could be useful for analysis and marketing purposes. The inclusion of a store number and address provides additional granularity, potentially aiding in logistics and delivery planning.

The DeliveryStaff table includes first and last names, as well as the vehicle type, which is a practical consideration for delivery logistics. Knowing the vehicle type can help in assigning appropriate delivery tasks based on distance and order size.

The Manager and Customers tables are straightforward, capturing essential contact information and, for managers, their tenure. This could be useful for administrative purposes and customer service.

The Payment\_info table is designed to handle different payment methods and card providers, which is important for financial transactions and record-keeping.

Overall, the database design is well thought out, with a clear structure that supports the operational needs of a restaurant order system. It allows for efficient data retrieval and manipulation, which is essential for the accompanying GUI developed using Tkinter. The GUI facilitates interaction with the database, providing functions for inserting, updating, and fetching data, as well as statistical analysis, such as calculating the mean customer rating for food. This design supports a robust and user-friendly system for managing restaurant orders.