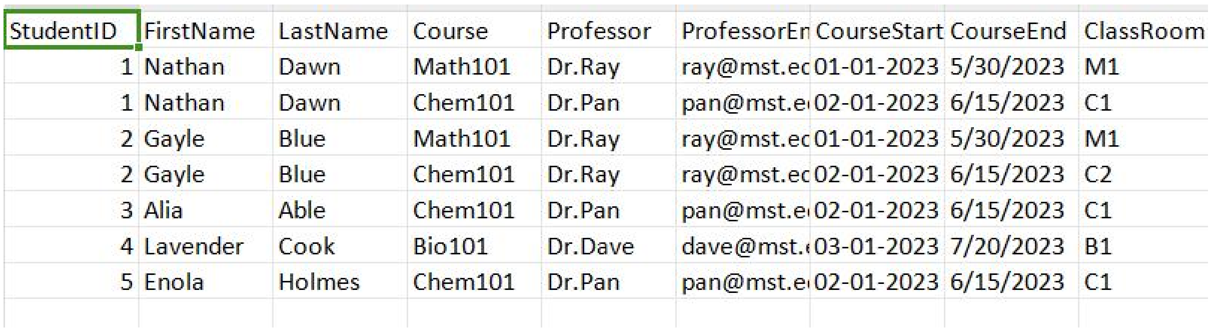
NAME : Kirthi Vaterira Kariappa

ID : 12634251

DATABASE PROJECT REPORT

Input table:

Code for this is in Normalization_codes -> code2 folder.



StudentID	FirstName	LastName	Course	Professor	ProfessorEn	CourseStart	CourseEnd	ClassRoom
1	Nathan	Dawn	Math101	Dr.Ray	ray@mst.ec	01-01-2023	5/30/2023	M1
1	Nathan	Dawn	Chem101	Dr.Pan	pan@mst.e	02-01-2023	6/15/2023	C1
2	Gayle	Blue	Math101	Dr.Ray	ray@mst.ec	01-01-2023	5/30/2023	M1
2	Gayle	Blue	Chem101	Dr.Ray	ray@mst.ec	02-01-2023	6/15/2023	C2
3	Alia	Able	Chem101	Dr.Pan	pan@mst.e	02-01-2023	6/15/2023	C1
4	Lavender	Cook	Bio101	Dr.Dave	dave@mst.	03-01-2023	7/20/2023	B1
5	Enola	Holmes	Chem101	Dr.Pan	pan@mst.e	02-01-2023	6/15/2023	C1

Assumptions:

Each class can be taught by multiple professors.

Each professor can teach multiple courses.

A class can have multiple sections (multiple classroom).

A student can take multiple courses but not the same course more than once.

A single course with multiple sections has same start and end date.

Functional dependencies:

This should be mentioned in the dependencies.txt file.

StudentID -> FirstName, LastName

Course, Professor -> ClassRoom

Course -> CourseStart, CourseEnd

Professor -> ProfessorEmail

Multivalued dependencies:

This should be mentioned in mvd_dependencies.txt file.

Course ->> Professor

Course ->> ClassRoom

StudentID ->> Course

StudentID ->> Professor

Running the code in terminal:

- ➔ Python main.py

- ➔ Choice of the highest normal form to reach (1: 1NF, 2: 2NF, 3: 3NF, B: BCNF, 4: 4NF, 5: 5NF)
- ➔ Find the highest normal form of the input table? (1: Yes, 2: No)
- ➔ Enter Primary Key values separated by comma : StudentID, Course

First normal form (1NF):

-The table already satisfies 1NF. Every attribute contains atomic values.
-A cell must never contain more than 1 value each row much be unique- potential primary key.
-each column name should be unique
-there should be no repeating groups
-The table already satisfies 1NF. Every attribute contains atomic values.
-This table doesn't handle nested relations in 1NF
-OUTPUT QUERIES AFTER 1NF:

```
CREATE TABLE StudentID_Course_table (
  StudentID VARCHAR(255) PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255),
  Course VARCHAR(255) PRIMARY KEY,
  Professor VARCHAR(255),
  ProfessorEmail VARCHAR(255),
  CourseStart VARCHAR(255),
  CourseEnd VARCHAR(255),
  ClassRoom VARCHAR(255)
);
```

Second normal form (2NF):

-It should be in 1NF.
-All data must depend on primary key.
-There should be no partial dependency.
-Composite key is (StudentID, Course)
-Course -> CourseStart, CourseEnd here CourseStart and CourseEnd are dependent on Course. So, there is partial dependency.
-StudentID ->FirstName, LastName here FirstName and Lastname depends only on StudentID. So, partial dependencies exist.
-The input table is not in 2NF and it was in 1NF, We have to remove the partial dependencies.
-OUTPUT QUERIES AFTER 2NF:

```
CREATE TABLE StudentID_table (
  StudentID VARCHAR(255) PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255)
);
CREATE TABLE Course_table (
  Course VARCHAR(255) PRIMARY KEY,
  CourseStart VARCHAR(255),
  CourseEnd VARCHAR(255)
);
CREATE TABLE StudentID_Course_table (
 FOREIGN KEY (StudentID) REFERENCES StudentID_table(StudentID),
 FOREIGN KEY (Course) REFERENCES Course_table(Course),
  Professor VARCHAR(255),
  ProfessorEmail VARCHAR(255),
  ClassRoom VARCHAR(255)
);
```

Third normal form (3NF):

-It should be in 1nf and 2nf.
-PK must fully define all columns and columns must not depend on any other key.
-There should be no transitive dependency.
-In table Enrollment as Professor -> ProfessorEmail transitively depends on the primary key, we should remove this transitive dependency to get 3NF form.
-OUTPUT QUERIES AFTER 3NF:

```
CREATE TABLE StudentID_table (
  StudentID VARCHAR(255) PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255)
);
CREATE TABLE Course_table (
  Course VARCHAR(255) PRIMARY KEY,
  CourseStart VARCHAR(255),
  CourseEnd VARCHAR(255)
);
CREATE TABLE Course_Professor_table (
 FOREIGN KEY (Course) REFERENCES Course_table(Course),
 FOREIGN KEY (Professor) REFERENCES Professor_table(Professor),
  ClassRoom VARCHAR(255)
);
CREATE TABLE StudentID_Course_table (
  Professor VARCHAR(255),
  ProfessorEmail VARCHAR(255),
 FOREIGN KEY (StudentID) REFERENCES StudentID_table(StudentID),
 FOREIGN KEY (Course) REFERENCES Course_table(Course)
);
```

Boyce Codd normal form (BCNF):

-It should be in 3NF
-a->b, a should be a super key
-Non-prime attributes derives prime attributes
-OUTPUT QUERIES AFTER BCNF:

```
CREATE TABLE StudentID_table (
  StudentID VARCHAR(255) PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255)
);
CREATE TABLE Course_table (
  Course VARCHAR(255) PRIMARY KEY,
  CourseStart VARCHAR(255),
  CourseEnd VARCHAR(255)
);
CREATE TABLE Course_Professor_table (
 FOREIGN KEY (Course) REFERENCES Course_table(Course),
 FOREIGN KEY (Professor) REFERENCES Professor_table(Professor),
  ClassRoom VARCHAR(255)
);
CREATE TABLE Professor_table (
  Professor VARCHAR(255) PRIMARY KEY,
  ProfessorEmail VARCHAR(255)
);
CREATE TABLE StudentID_Course_table (
 FOREIGN KEY (Course) REFERENCES Course_table(Course),
 FOREIGN KEY (StudentID) REFERENCES StudentID_table(StudentID),
  Professor VARCHAR(255)
);
```

Fourth normal form (4NF):

-It should be in BCNF
-There should be no multivalued dependency
-a->b for single value of a, more than 1 value of b exsist.
OUTPUT QUERIES AFTER 4NF:

```
CREATE TABLE StudentID_table (
  StudentID VARCHAR(255) PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255)
);
CREATE TABLE Course_table (
  Course VARCHAR(255) PRIMARY KEY,
  CourseStart VARCHAR(255),
  CourseEnd VARCHAR(255)
);
CREATE TABLE Course_Professor_table (
 FOREIGN KEY (Course) REFERENCES Course_table(Course),
 FOREIGN KEY (Professor) REFERENCES Professor_table(Professor),
  ClassRoom VARCHAR(255)
);
CREATE TABLE Professor_table (
  Professor VARCHAR(255) PRIMARY KEY,
  ProfessorEmail VARCHAR(255)
);
CREATE TABLE StudentID_Course_table (
 FOREIGN KEY (StudentID) REFERENCES StudentID_table(StudentID),
 FOREIGN KEY (Course) REFERENCES Course_table(Course),
  Professor VARCHAR(255)
);
```

Fifth normal form (5NF):

OUTPUT QUERIES AFTER 5NF:

```
CREATE TABLE StudentID_table (
  StudentID VARCHAR(255) PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255)
);
CREATE TABLE Course_table (
  Course VARCHAR(255) PRIMARY KEY,
  CourseStart VARCHAR(255),
  CourseEnd VARCHAR(255)
);
CREATE TABLE Course_Professor_table (
 FOREIGN KEY (Course) REFERENCES Course_table(Course),
 FOREIGN KEY (Professor) REFERENCES Professor_table(Professor),
  ClassRoom VARCHAR(255)
);
CREATE TABLE Professor_table (
  Professor VARCHAR(255) PRIMARY KEY,
  ProfessorEmail VARCHAR(255)
);
CREATE TABLE StudentID_Course_table (
  Professor VARCHAR(255),
 FOREIGN KEY (StudentID) REFERENCES StudentID_table(StudentID),
 FOREIGN KEY (Course) REFERENCES Course_table(Course)
);
```

Domain key normal form (DKNF):

I tried to add a section of code to get DKNF form. The extended code can be found in Normalization_codes -> code3 folder.

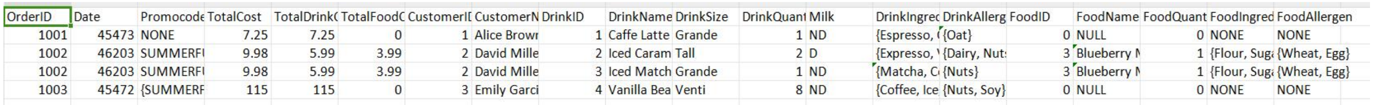
Domain constraints:

StudentID : Integer, Non-null
FirstName : Non-null, alphabetic characters only
LastName : Non-null, alphabetic characters only
Course : Non-null, values must be valid course codes
Professor : Non-null, must be an alphabetic string
ProfessorEmail : Text, must follow email format
CourseStart : Non-null, must be before CourseEnd
CourseEnd : Non-null, must be after CourseStart
ClassRoom : Non-null, values must follow classroom naming conventions

IMPLEMENTATION OF THE CODE TO THE GIVEN TESTING TABLE

Input table:

Code for this is in Normalization_codes -> code1 folder.



OrderID	Date	Promocode	TotalCost	TotalDrink(	TotalFood(	CustomerI[	CustomerN	DrinkID	DrinkName	DrinkSize	DrinkQuant	Milk	DrinkIngred	DrinkAllerg	FoodID	FoodName	FoodQuant	FoodIngred	FoodAllergen
1001	45473	NONE	7.25	7.25	0	1	Alice Browr	1	Caffe Latte	Grande	1	ND	{Espresso, (	{Oat}	0	NULL	0	NONE	NONE
1002	46203	SUMMERFI	9.98	5.99	3.99	2	David Mille	2	Iced Caram	Tall	2	D	{Expresso, \	{Dairy, Nut:	3	Blueberry N	1	{Flour, Suga	{Wheat, Egg}
1002	46203	SUMMERFI	9.98	5.99	3.99	2	David Mille	3	Iced Match	Grande	1	ND	{Matcha, C	{Nuts}	3	Blueberry N	1	{Flour, Suga	{Wheat, Egg}
1003	45472	{SUMMERF	115	115	0	3	Emily Garci	4	Vanilla Bea	Venti	8	ND	{Coffee, Ice	{Nuts, Soy}	0	NULL	0	NONE	NONE

Primary key:

OrderID, DrinkID, FoodID

Functional dependencies:
OrderID -> PromocodeUsed
DrinkID -> DrinkIngredient
DrinkID -> DrinkAllergen
FoodID -> FoodIngredient
FoodID -> FoodAllergen
OrderID -> Date, TotalCost, TotalDrinkCost, TotalFoodCost, CustomerID, CustomerName
OrderID, DrinkID -> DrinkSize, DrinkQuantity, Milk
OrderID, FoodID -> FoodQuantity
CustomerID -> CustomerName
DrinkID -> DrinkName
FoodID -> FoodName

Multivalued dependencies:
OrderID ->> PromocodeUsed
DrinkID ->> DrinkIngredient
DrinkID ->> DrinkAllergen
FoodID ->> FoodIngredient
FoodID ->> FoodAllergen

First normal form (1NF) :

OUTPUT QUERIES AFTER 1NF:
```
CREATE TABLE OrderID_DrinkID_FoodID_table (
  OrderID VARCHAR(255) PRIMARY KEY,
  Date VARCHAR(255),
  PromocodeUsed VARCHAR(255),
  TotalCost VARCHAR(255),
  TotalDrinkCost VARCHAR(255),
  TotalFoodCost VARCHAR(255),
  CustomerID VARCHAR(255),
  CustomerName VARCHAR(255),
  DrinkID VARCHAR(255) PRIMARY KEY,
  DrinkName VARCHAR(255),
  DrinkSize VARCHAR(255),
  DrinkQuantity VARCHAR(255),
  Milk VARCHAR(255),
  DrinkIngredient VARCHAR(255),
  DrinkAllergen VARCHAR(255),
  FoodID VARCHAR(255) PRIMARY KEY,
  FoodName VARCHAR(255),
  FoodQuantity VARCHAR(255),
  FoodIngredient VARCHAR(255),
  FoodAllergen VARCHAR(255)
);
```

Second normal form (2NF) :

OUTPUT QUERIES AFTER 2NF:
```
CREATE TABLE OrderID_table (
  OrderID VARCHAR(255) PRIMARY KEY,
  Date VARCHAR(255),
  TotalCost VARCHAR(255),
  TotalDrinkCost VARCHAR(255),
  TotalFoodCost VARCHAR(255),
  CustomerID VARCHAR(255),
  CustomerName VARCHAR(255)
);
CREATE TABLE DrinkID_table (
  DrinkID VARCHAR(255) PRIMARY KEY,
  DrinkName VARCHAR(255)
);
CREATE TABLE FoodID_table (
  FoodID VARCHAR(255) PRIMARY KEY,
  FoodName VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkSize VARCHAR(255),
  DrinkQuantity VARCHAR(255),
  Milk VARCHAR(255)
);
CREATE TABLE OrderID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodQuantity VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
  PromocodeUsed VARCHAR(255),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkIngredient VARCHAR(255),
  DrinkAllergen VARCHAR(255),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodIngredient VARCHAR(255),
  FoodAllergen VARCHAR(255)
);
```

Third normal form (3NF) :

OUTPUT QUERIES AFTER 1NF:

```
CREATE TABLE CustomerID_table (
  CustomerID VARCHAR(255) PRIMARY KEY,
  CustomerName VARCHAR(255)
);
CREATE TABLE OrderID_table (
  TotalCost VARCHAR(255),
  Date VARCHAR(255),
  TotalFoodCost VARCHAR(255),
  TotalDrinkCost VARCHAR(255),
  CustomerID VARCHAR(255),
  OrderID VARCHAR(255) PRIMARY KEY
);
CREATE TABLE DrinkID_table (
  DrinkID VARCHAR(255) PRIMARY KEY,
  DrinkName VARCHAR(255)
);
CREATE TABLE FoodID_table (
  FoodID VARCHAR(255) PRIMARY KEY,
  FoodName VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkSize VARCHAR(255),
  DrinkQuantity VARCHAR(255),
  Milk VARCHAR(255)
);
CREATE TABLE OrderID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodQuantity VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
  PromocodeUsed VARCHAR(255),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkIngredient VARCHAR(255),
  DrinkAllergen VARCHAR(255),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodIngredient VARCHAR(255),
  FoodAllergen VARCHAR(255)
);
```

Boyce Codd normal form (BCNF) :

OUTPUT QUERIES AFTER BCNF:

```
CREATE TABLE CustomerID_table (
  CustomerID VARCHAR(255) PRIMARY KEY,
  CustomerName VARCHAR(255)
);
CREATE TABLE OrderID_table (
  OrderID VARCHAR(255) PRIMARY KEY,
  TotalFoodCost VARCHAR(255),
  TotalDrinkCost VARCHAR(255),
  CustomerID VARCHAR(255),
  Date VARCHAR(255),
  TotalCost VARCHAR(255)
);
CREATE TABLE DrinkID_table (
  DrinkID VARCHAR(255) PRIMARY KEY,
  DrinkName VARCHAR(255)
);
CREATE TABLE FoodID_table (
  FoodID VARCHAR(255) PRIMARY KEY,
  FoodName VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkSize VARCHAR(255),
  DrinkQuantity VARCHAR(255),
  Milk VARCHAR(255)
);
CREATE TABLE OrderID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodQuantity VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
  PromocodeUsed VARCHAR(255),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkIngredient VARCHAR(255),
  DrinkAllergen VARCHAR(255),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodIngredient VARCHAR(255),
  FoodAllergen VARCHAR(255)
);
```

Fourth normal form (4NF) :

OUTPUT QUERIES AFTER 4NF:

```
CREATE TABLE CustomerID_table (
  CustomerID VARCHAR(255) PRIMARY KEY,
  CustomerName VARCHAR(255)
);
CREATE TABLE OrderID_table (
  Date VARCHAR(255),
  TotalDrinkCost VARCHAR(255),
  CustomerID VARCHAR(255),
  TotalCost VARCHAR(255),
  TotalFoodCost VARCHAR(255),
  OrderID VARCHAR(255) PRIMARY KEY
);
CREATE TABLE DrinkID_table (
  DrinkID VARCHAR(255) PRIMARY KEY,
  DrinkName VARCHAR(255)
);
CREATE TABLE FoodID_table (
  FoodID VARCHAR(255) PRIMARY KEY,
  FoodName VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkSize VARCHAR(255),
  DrinkQuantity VARCHAR(255),
  Milk VARCHAR(255)
);
CREATE TABLE OrderID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodQuantity VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
  PromocodeUsed VARCHAR(255),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkIngredient VARCHAR(255),
  DrinkAllergen VARCHAR(255),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodIngredient VARCHAR(255),
  FoodAllergen VARCHAR(255)
);
```

Fifth normal form (5NF) :

OUTPUT QUERIES AFTER 5NF:

```
CREATE TABLE CustomerID_table (
  CustomerID VARCHAR(255) PRIMARY KEY,
  CustomerName VARCHAR(255)
);
CREATE TABLE OrderID_table (
  TotalFoodCost VARCHAR(255),
  TotalDrinkCost VARCHAR(255),
  OrderID VARCHAR(255) PRIMARY KEY,
  TotalCost VARCHAR(255),
  Date VARCHAR(255),
  CustomerID VARCHAR(255)
);
CREATE TABLE DrinkID_table (
  DrinkID VARCHAR(255) PRIMARY KEY,
  DrinkName VARCHAR(255)
);
```

```
CREATE TABLE FoodID_table (
  FoodID VARCHAR(255) PRIMARY KEY,
  FoodName VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkSize VARCHAR(255),
  DrinkQuantity VARCHAR(255),
  Milk VARCHAR(255)
);
CREATE TABLE OrderID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodQuantity VARCHAR(255)
);
CREATE TABLE OrderID_DrinkID_FoodID_table (
 FOREIGN KEY (OrderID) REFERENCES OrderID_table(OrderID),
  PromocodeUsed VARCHAR(255),
 FOREIGN KEY (DrinkID) REFERENCES DrinkID_table(DrinkID),
  DrinkIngredient VARCHAR(255),
  DrinkAllergen VARCHAR(255),
 FOREIGN KEY (FoodID) REFERENCES FoodID_table(FoodID),
  FoodIngredient VARCHAR(255),
  FoodAllergen VARCHAR(255)
);
```