# Deliverables

## Deliverable 1.1

- Team 20: Blake Crockett, Bowen Fan, Madhav Dahal, Nishan Budhathoki
- Team 21: Brennan Graham, Connor Day, Jose Cruz Guerrero, Keaton Martin

## Deliverable 2.1

To load in the other databases we initially used two methods depending on whether we had MySQL locally. Those of us who could only use mysql.cs.uky.edu renamed the other team's databases to have something like '_team_20' after each name to avoid conflicting with our own tables. Those of us who had MySQL locally could instead load the other databases separately (into the databases team_3, team_20, and team_21). During our final integration one of us set up an internet accessible MySQL server and used the second method to load the three databases as it was more reproducible (and less prone to error) than loading all three team's tables into one database. This also mirrors much more closely how an actual database integration would work.

To merge the databases together we wrote a Python script that loaded both of the other team's databases into Pandas dataframes which were used to reformat and check the data. The script then output the Pandas dataframe into an SQL `INSERT` statement which added the data to our database.

Additionally, our database only accepts entities that include both a name and address. Therefore, values from other databases that did not have values for both characteristics are dropped.

## Deliverable 2.2

We used python with pandas to flag duplicates. If the Name and Address for an entity matched another entity in the table, both were flagged as duplicates. For deliverables 5 and 6, only the information for the first value for each duplicate pair was used to generate the output.

## Deliverable 2.3

We used strings and python to update the tables. We Used the code:

```
cursor.execute(
    "INSERT INTO Entity_Table (Entity_ID, Street_Name, Zip, City, StateName, EntityName, Pri
    + e21
```

```
    + e20[:-1]
    + ";"
)
```

where e20 were the entities from team20 and e21 were the values from team 21.
Some of the values associated with these strings were

```
e20 :(41,'300 Rose Street Room 102 Hardymon Building','40506','Lexington','KY','Mr. Ray L. H
(42,'301 Hilltop Avenue, Room 102','40506','Lexington','KY','Mr. Ray L. Hyatt Jr.',NULL)

e21:(113,'666 Chestnut Road','40514','Lexington','KY','Richard Jackson','859-345-6782'),(114
```

Of note: some of the duplicate entries were copied into the other databases with different addresses. Given that we were importing the data as if we were a business who had acquired these customers, we did not modify the addresses as this could indicate that these are separate people with the same name.

## Deliverable 3.1

```
+--------------------+
| Tables_in_team3    |
+--------------------+
| Entity_Table       |
| Receipt_Table      |
| Telephone_Numbers  |
+--------------------+
```

## Deliverable 3.2

```
+-----------+-----------------------------------------------+-------+-----------------+----
| Entity_ID | Street_Name                                   | Zip   | City            | Sta
+-----------+-----------------------------------------------+-------+-----------------+----
|         1 | 300 Rose Street Room 102 Hardymon Building    | 40506 | Lexington       | KY
|         2 | 301 Hilltop Avenue, Room 102                  | 40506 | Lexington       | KY
|         3 | 82 Beaver St Room 1301                        | 10005 | New York        | NY
|         4 | 200 Park Avenue Penthouse                     | 10001 | New York        | NY
|         5 | 117A Bleecker Street                          | 10001 | New York        | NY
|         6 | 200 Park Avenue Apartment 221                 | 10001 | New York        | NY
|         7 | #1 Avenue of Champions                        | 40506 | Lexington       | KY
|         8 | 200 Park Avenue                               | 40507 | Lexington       | KY
|         9 | 1 Dead End Row Room 200                       | 12347 | Dallas          | TX
|        10 | 1 Dead End Row Room 200                       | 12347 | Dallas          | TX
|        11 | 2299 Richmond Rd                              | 40502 | Lexington       | KY
|        12 | 471 Rose St                                   | 40508 | Lexington       | KY
|        13 | 150 W Lowry Ln #190                           | 40503 | Lexington       | KY
|        14 | 2161 Paul Jones Way                           | 40509 | Lexington       | KY
```

| | | | | |
|---|---|---|---|---|
| 15 | 1180 Seven Seas Dr | 32830 | Lake Buena Vista | FL |
| 16 | 500 W New Circle Rd | 40511 | Lexington | KY |
| 17 | 123 NotAReal St | 40502 | Lexington | KY |
| 18 | 1600 Pennsylvania Ave NW | 20500 | Washington | DC |
| 19 | 1400 Defense Pentagon | 20301 | Washington | DC |
| 20 | 101 Cochran Rd | 40502 | Lexington | KY |
| 21 | 867 South Broadway | 40504 | Lexington | KY |
| 22 | 700 Clark Ave | 63102 | St. Louis | MO |
| 23 | Antarctic Support Contract 7400 S. Tucson Way | 80112 | Centennial | CO |
| 24 | 1386 Lexington Rd | 40206 | Louisville | KY |
| 25 | 867 South Broadway | 40504 | Lexington | KY |
| 26 | 899 Manchester St | 40508 | Lexington | KY |
| 27 | 1649 M-32 | 49735 | Gaylord | MI |
| 28 | 7345 Delridge Way SW | 98106 | Seattle | WA |
| 29 | 500 S Upper St STE 110 | 40508 | Lexington | KY |
| 30 | 211 NE Revere Avenue | 97701 | Bend | OR |
| 31 | 1837 Plaudit Pl | 40509 | Lexington | KY |
| 32 | 525 Alakawa St | 96817 | Honolulu | HI |
| 33 | 2021 Harrodsburg Rd | 40504 | Lexington | KY |
| 34 | 2039 El Camino Real | 95050 | Santa Clara | CA |
| 35 | 918 Natural Bridge Rd | 40376 | Slade | KY |
| 36 | 867 S Broadway | 40504 | Lexington | KY |
| 37 | 3400 Vine St | 45220 | Cincinnati | OH |
| 38 | 417 E Maxwell St Unit B | 40508 | Lexington | KY |
| 39 | 4055 Nichols Park Dr | 40503 | Lexington | KY |
| 40 | 4081 Finn Way | 40503 | Lexington | KY |
| 41 | 300 Rose Street Room 102 Hardymon Building | 40506 | Lexington | KY |
| 42 | 301 Hilltop Avenue, Room 102 | 40506 | Lexington | KY |
| 43 | 82 Beaver St Room 1301 | 10005 | New York | NY |
| 44 | 200 Park Avenue Penthouse | 10001 | New York | NY |
| 45 | 117A Bleecker Street | 10001 | New York | NY |
| 46 | 200 Park Avenue Apartment 221 | 10001 | New York | NY |
| 47 | #1 Avenue of Champions | 40507 | Lexington | KY |
| 48 | 1 Dead End Row Room 200 | 12347 | Dallas | TX |
| 49 | 1 Dead End Row Room 200 | 12347 | Dallas | TX |
| 50 | 200 Park Avenue | 40507 | Lexington | KY |
| 51 | 1625 Pelham South | 36265 | Jacksonville | AL |
| 52 | 3371 S Alabama Ave | 36460 | Monroeville | AL |
| 53 | 103 North Caroline St | 13350 | Herkimer | NY |
| 54 | 1000 State Route 36 | 14843 | Hornell | NY |
| 55 | 1400 County Rd 64 | 14845 | Horseheads | NY |
| 56 | 135 Fairgrounds Memorial Pkwy | 14850 | Ithaca | NY |
| 57 | 2 Gannett Dr | 13790 | Johnson City | NY |
| 58 | 233 5th Ave Ext | 12095 | Johnstown | NY |
| 59 | 601 Frank Stottile Blvd | 12401 | Kingston | NY |
| 60 | 350 E Fairmount Ave | 14750 | Lakewood | NY |

| | | | | |
|---|---|---|---|---|
| 61 | 4975 Transit Rd | 14086 | Lancaster | NY |
| 62 | 579 Troy-Schenectady Road | 12110 | Latham | NY |
| 63 | 5783 So Transit Road | 14094 | Lockport | NY |
| 64 | 280 Washington Street | 1749 | Hudson | MA |
| 65 | 20 Soojian Dr | 1524 | Leicester | MA |
| 66 | 11 Jungle Road | 1453 | Leominster | MA |
| 67 | 301 Massachusetts Ave | 1462 | Lunenburg | MA |
| 68 | 780 Lynnway | 1905 | Lynn | MA |
| 69 | 70 Pleasant Valley Street | 1844 | Methuen | MA |
| 70 | 3138 Custer Drive, Suite 210 | 40517 | Lexington | KY |
| 71 | 1845 Goodpaster Way | 40505 | Lexington | KY |
| 72 | 721 W. Main St. | 40508 | Lexington | KY |
| 73 | 241 Meadow Valley Rd. | 40511 | Lexington | KY |
| 74 | 123 Means Drive | 40356 | Nicholasville | KY |
| 75 | 956 Enterprise Ct Suite 150 | 40510 | Lexington | KY |
| 76 | 1 Dell Way | 78682 | Round Rock | TX |
| 77 | 800 N Brand Blvd | 91203 | Glendale | CA |
| 78 | 242 W 41st St. | 10036 | New York | NY |
| 79 | 5000 South Broad St. | 19112 | Philadelphia | PA |
| 80 | One Bowerman Drive | 97005 | Beaverton | OR |
| 81 | 300 Rose Street Room 102 Hardymon Building | 40506 | Lexington | KY |
| 82 | 1 Dead End Row Room 200 | 12347 | Dallas | TX |
| 83 | 1 Dead End Row Room 200 | 12347 | Dallas | TX |
| 84 | 82 Beaver St Room 1301 | 10005 | New York | NY |
| 85 | 200 Park Avenue Penthouse | 10001 | New York | NY |
| 86 | 117A Bleecker Street | 10001 | New York | NY |
| 87 | #1 Avenue of Champions | 40506 | Lexington | KY |
| 88 | 200 Park Avenue Apartment 221 | 10001 | New York | NY |
| 89 | 200 Park Avenue Apartment 221 | 10001 | New York | NY |
| 90 | 301 Hilltop Avenue, Room 102 | 40506 | Lexington | KY |
| 91 | 2558 Larkin Rd #120 | 40503 | Lexington | KY |
| 92 | 10 Quality St | 40507 | Lexington | KY |
| 93 | 2333 Alexandria Drive | 40504 | Lexington | KY |
| 94 | 740 W New Circle Rd | 40511 | Lexington | KY |
| 95 | 1780 Lexington Road | 40505 | Lexington | KY |
| 96 | 1700 Lexington Rd | 40505 | Lexington | KY |
| 97 | 4000 East New Circle Rd | 40511 | Lexington | KY |
| 98 | 4201 Versailles Rd. | 40510 | Lexington | KY |
| 99 | 4201 Versailles Rd. | 40510 | Lexington | KY |
| 100 | 123 Elm Street | 40502 | Lexington | KY |
| 101 | 456 Oak Ave. Apt. 374 | 40503 | Lexington | KY |
| 102 | 456 Oak Ave. Apt. 374 | 40503 | Lexington | KY |
| 103 | 789 Maple Road Unit 3 | 40504 | Lexington | KY |
| 104 | 789 Maple Road Unit 3 | 40504 | Lexington | KY |
| 105 | 101 Pine Lane | 40505 | Lexington | KY |
| 106 | 234 Cedar Dr. | 40506 | Lexington | KY |

```
|       107 | 890 Birch Circle                                | 40508 | Lexington        | KY
|       108 | 111 Willow Drive                                | 40509 | Lexington        | KY
|       109 | 222 Spruce Avenue                               | 40510 | Lexington        | KY
|       110 | 333 Hickory Street                              | 40511 | Lexington        | KY
|       111 | 444 Sycamore Lane                               | 40512 | Lexington        | KY
|       112 | 555 Redwood Drive                               | 40513 | Lexington        | KY
|       113 | 666 Chestnut Road                               | 40514 | Lexington        | KY
|       114 | 777 Poplar Court                                | 40515 | Lexington        | KY
|       115 | 888 Birch Circle                                | 40516 | Lexington        | KY
|       116 | 999 Willow Drive                                | 40517 | Lexington        | KY
|       117 | 121 Pine Lane                                   | 40518 | Lexington        | KY
|       118 | 232 Cedar Drive                                 | 40519 | Lexington        | KY
|       119 | 343 Walnut Court                                | 40520 | Lexington        | KY
|       121 | 547 Euclid Ave Ste. 120                         | 40504 | Lexington        | KY
+-----------+-------------------------------------------------+-------+------------------+----
```

## Deliverable 4.1

```python
from connection import connection
from database import grab_all_entities
import random

def randomize_entity(entity_id:int):
    '''Randomizes the contact preferences for a given entity'''
    if not connection:
        print("Could not connect - no connection object")
    elif connection.is_connected():

        preference = random.randint(0, 255)
        print(entity_id, preference)

        # Grab our full Entity List
        with connection.cursor() as cursor:
            cursor.execute("UPDATE Entity_Table SET Contact_Preferences = (%s) WHERE Entity_

        connection.commit()
    else:
        print("Not connected - connection object is not connected")

peoples_list = {
    'John Wick': 8,
    'Bob': 1,
    'Tony Stark': 63,
    'Stephen Strange': 4,
    'Ray L. Hyatt': 64,
```

```
}

def set_specific():
    '''Sets the contact preferences for a given entity'''
    global peopls_list
    if not connection:
        print("Could not connect - no connection object")
    elif connection.is_connected():

        # Grab our full Entity List
        with connection.cursor() as cursor:

            for person, preference in peoples_list.items():
                print(person, preference)
                person_pattern = f"%{person}%"
                cursor.execute("UPDATE Entity_Table SET Contact_Preferences = (%s) WHERE Ent

        connection.commit()
    else:
        print("Not connected - connection object is not connected")

def randomize_contacts():
    '''Randomizes the contact preferences for each entity'''
    global peoples_list

    # Grab the entity list
    entities = grab_all_entities()

    '''
    entity = {
            "entity_id": entity[0],
            "address": entity[1],
            "zip": entity[2],
            "city": entity[3],
            "state": entity[4],
            "name": entity[5],
            "primary_phone": entity[6],
            "contact_preference": get_contact_info(entity[7]),
        }
    '''

    for entity in entities:
        randomize_entity(entity['entity_id'])

set_specific()
randomize_contacts()
```

## Deliverable 4.2

See 3.2, since the contact values are just booleans, we decided to use a bitmask to store them together in a single column. This is out logic for storing the value:

```
TEXT_BITMASK = 0b00000001
EMAIL_BITMASK = 0b00000010
MAIL_BITMASK = 0b00000100
ROBOCALL_BITMASK = 0b00001000
FAX_BITMASK = 0b00010000
PHONE_BITMASK = 0b00100000
DNC_BITMASK = 0b01000000
ALL_BITMASK = (
    TEXT_BITMASK | EMAIL_BITMASK | MAIL_BITMASK | ROBOCALL_BITMASK | PHONE_BITMASK
)


def contact_info(
    ALL=False,
    EMAIL=False,
    TEXT=False,
    DNC=False,
    MAIL=False,
    ROBOCALL=False,
    FAX=False,
    PHONE=False,
) -> int:
    """Returns a binary representation for ALL, EMAIL, TEXT, DNC, MAIL, ROBOCALL, FAX, and P
    bitmask = 0b00000000

    if ALL:
        bitmask |= ALL_BITMASK
    else:
        if EMAIL:
            bitmask |= EMAIL_BITMASK
        if TEXT:
            bitmask |= TEXT_BITMASK
        if MAIL:
            bitmask |= MAIL_BITMASK
        if ROBOCALL:
            bitmask |= ROBOCALL_BITMASK
        if PHONE:
            bitmask |= PHONE_BITMASK
    if DNC:
        bitmask |= DNC_BITMASK
    if FAX:
```

```
        bitmask |= FAX_BITMASK

    return bitmask
```

## Deliverable 5.1

```
import mysql.connector
import pandas as pd

from args import args
from connection import connection

# Create Dataframe
Entity_Table = pd.read_sql("Select * from Entity_Table", connection)

Entity_Table["duplicate"] = "N"

Entity_Table.loc[:5, "duplicate"] = "Y"

# Create duplicates dataframe
duplicates = Entity_Table[Entity_Table.duplicate == "Y"].copy()
duplicates.reset_index(inplace=True, drop=True)

# Combine name and Address to get identifier for possible duplicates
duplicates["Name_Address"] = (
    duplicates.EntityName
    + " "
    + duplicates.Street_Name
    + " "
    + duplicates.City
    + " "
    + duplicates.StateName
    + " "
    + duplicates.Zip
)
unique_names = duplicates.Name_Address.unique()

# Create postcard
# Only send postcard to first listing for likely duplicate. If after sending postcard it is
# adjust database to indicate that the likely duplicates were unique customers
for i in unique_names:
    row = duplicates[duplicates.Name_Address == i].iloc[0]
    print(
        (row.email or "No email provided")
        + "\n"
```

```
        + row.EntityName
        + "\n"
        + row.Street_Name
        + "\n"
        + row.City
        + ", "
        + row.StateName
        + " "
        + row.Zip
        + "\n\n"
        + row.EntityName
        + "\n"
        + "It is great to meet you again! Your favorite food delivery family is growing, loo
        + "\n\n"
        + "Looking forward to serving you again!"
        + "\n"
    )

connection.close()
```

## Deliverable 5.2

```
Mr.296@icloud.com
Mr. Ray L. Hyatt Jr.
300 Rose Street Room 102 Hardymon Building
Lexington, KY 40506

Mr. Ray L. Hyatt Jr.
It is great to meet you again! Your favorite food delivery family is growing, looks like you

Looking forward to serving you again!

Mr.922@gmail.com
Mr. Ray L. Hyatt Jr.
301 Hilltop Avenue, Room 102
Lexington, KY 40506

Mr. Ray L. Hyatt Jr.
It is great to meet you again! Your favorite food delivery family is growing, looks like you

Looking forward to serving you again!

Joh111@gmail.com
John Wick
82 Beaver St Room 1301
```

```
New York, NY 10005

John Wick
It is great to meet you again! Your favorite food delivery family is growing, looks like you

Looking forward to serving you again!

Ton795@gmail.com
Tony Stark
200 Park Avenue Penthouse
New York, NY 10001

Tony Stark
It is great to meet you again! Your favorite food delivery family is growing, looks like you

Looking forward to serving you again!

Dr.902@gmail.com
Dr. Stephen Strange
117A Bleecker Street
New York, NY 10001

Dr. Stephen Strange
It is great to meet you again! Your favorite food delivery family is growing, looks like you

Looking forward to serving you again!

Bob944@icloud.com
Bob C. Smith
200 Park Avenue Apartment 221
New York, NY 10001

Bob C. Smith
It is great to meet you again! Your favorite food delivery family is growing, looks like you

Looking forward to serving you again!
```

## Deliverable 6.1

```
import mysql.connector
import pandas as pd

from connection import connection
from contact import get_contact_info
```

```
# Create Dataframe
Entity_Table = pd.read_sql("Select * from Entity_Table", connection)

# Create duplicates dataframe
duplicates = Entity_Table[Entity_Table.duplicate == "Y"].copy()
duplicates.reset_index(inplace=True, drop=True)

# Combine name and Address to get identifier for possible duplicates
duplicates["Name_Address"] = (
    duplicates.EntityName
    + " "
    + duplicates.Street_Name
    + " "
    + duplicates.City
    + " "
    + duplicates.StateName
    + " "
    + duplicates.Zip
)
unique_names = duplicates.Name_Address.unique()

# Send Coupon codes. Our customers are listed as entities rather than as businesses and cust
# used for the name for the coupon.
# Customer can choose a contact preference that involves a phone number without providing a
for i in unique_names:
    row = duplicates[duplicates.Name_Address == i].iloc[0]
    row.contact_preferences = get_contact_info(row.Contact_Preferences)
    if row.contact_preferences["email"] == True:
        print("EMAIL: " + row.EntityName + ";" + row.email + ";" + "25% Coupon Code")
    if row.contact_preferences["text"] == True:
        if row.Primary_Telephone_Number != None:
            print(
                "TEXT: "
                + row.EntityName
                + "; "
                + row.Primary_Telephone_Number
                + "; "
                + "25% Coupon Code"
            )
        else:
            print("TEXT: " + row.EntityName + "; No phone number provided")
    if row.contact_preferences["robocall"] == True:
        if row.Primary_Telephone_Number != None:
            print(
                "ROBOCALL: "
                + row.EntityName
```

```
            + "; "
            + row.Primary_Telephone_Number
            + "; "
            + "25% Coupon Code"
        )
    else:
        print("ROBOCALL: " + row.EntityName + "; No phone number provided")
if row.contact_preferences["phone"] == True:
    if row.Primary_Telephone_Number != None:
        print(
            "PHONE: "
            + row.EntityName
            + "; "
            + row.Primary_Telephone_Number
            + "; "
            + "25% Coupon Code"
        )
    else:
        print("PHONE: " + row.EntityName + "; No phone number provided")
if row.contact_preferences["fax"] == True:
    if row.Primary_Telephone_Number != None:
        print(
            "FAX: "
            + row.EntityName
            + "; "
            + row.Primary_Telephone_Number
            + "; "
            + "25% Coupon Code"
        )
    else:
        print("FAX: " + row.EntityName + "; No phone number provided")
if row.contact_preferences["dnc"] == True:
    print("DNC: " + row.EntityName)
if row.contact_preferences["mail"] == True:
    print(
        "MAIL: "
        + row.EntityName
        + "; "
        + row.Street_Name
        + " "
        + row.City
        + ", "
        + row.StateName
        + " "
        + row.Zip
        + "; "
```

```
            + "25% Coupon Code"
        )
    print(" ")

connection.close()
```

## Deliverable 6.2

```
DNC: Mr. Ray L. Hyatt Jr.

DNC: Mr. Ray L. Hyatt Jr.

ROBOCALL: John Wick; 555-555-5555; 25% Coupon Code

EMAIL: Tony Stark;Ton795@gmail.com;25% Coupon Code
TEXT: Tony Stark; 555-555-3142; 25% Coupon Code
ROBOCALL: Tony Stark; 555-555-3142; 25% Coupon Code
PHONE: Tony Stark; 555-555-3142; 25% Coupon Code
FAX: Tony Stark; 555-555-3142; 25% Coupon Code
MAIL: Tony Stark; 200 Park Avenue Penthouse New York, NY 10001; 25% Coupon Code

MAIL: Dr. Stephen Strange; 117A Bleecker Street New York, NY 10001; 25% Coupon Code

TEXT: Bob C. Smith; No phone number provided

TEXT: Bob C. Smith; No phone number provided

TEXT: Bob Porter c/o Intech; No phone number provided

TEXT: Mr. Bob Sydell c/o Intech; No phone number provided
```

## Deliverable 7.1

```
mysqldump --host HOSTNAME -u USER -p team3 > cs405g.team3.sql
```

## Deliverable 8.1

One of the largest difficulties within this project stemmed from the differences in how the various groups formatted their databases, and figuring out how to format the data when importing their data, not only to make the data compatible, but to also allow our database to detect the duplicate entries. Because of this, a fair amount of time was spent developing ways to sanitize the imported data to make it both usable and recognizable to our own database.

In addition, we learned the importance of standardizing data. To this point, sometimes providing too much granularity for the data can make a database too cumbersome to use. However, if the data is not broken up sufficiently, more work will be needed later to perform certain queries. Because of this, a balance must be struck.