IST 659 M005 Data Admin Concepts and DB Mgmt

Project Implementation Report

Citi Bike Analysis & Insights

Group Members:

Ruchak Vira
Karthik Ramanathan
Pranav Sheth

Table of Content

Project Summary

Background

Citi Bike is a bike sharing program run in New York City(NYC), Jersey city, & Hoboken. The purpose of the program is to create an ecosystem where commuters and riders are able to save time and money to get around the town with convenience and ease. The program's mission is also to provide health benefits through cycling and zero down on the emissions and the carbon footprint on the environment.

Users can unlock a bike from the docking stations that are spread across the entire cities listed in the previous paragraph. Post unlocking, a user can take rides for any amount of duration and finally return the bike once the ride is completed to any docking station present in the vicinity. With increasing traffic congestion in hustling cities, Citi bikes are proving to be a preference for all sorts of daily commuters and are also turning out to be an exciting way for travelers to explore the city.

Problem Statement & Solution

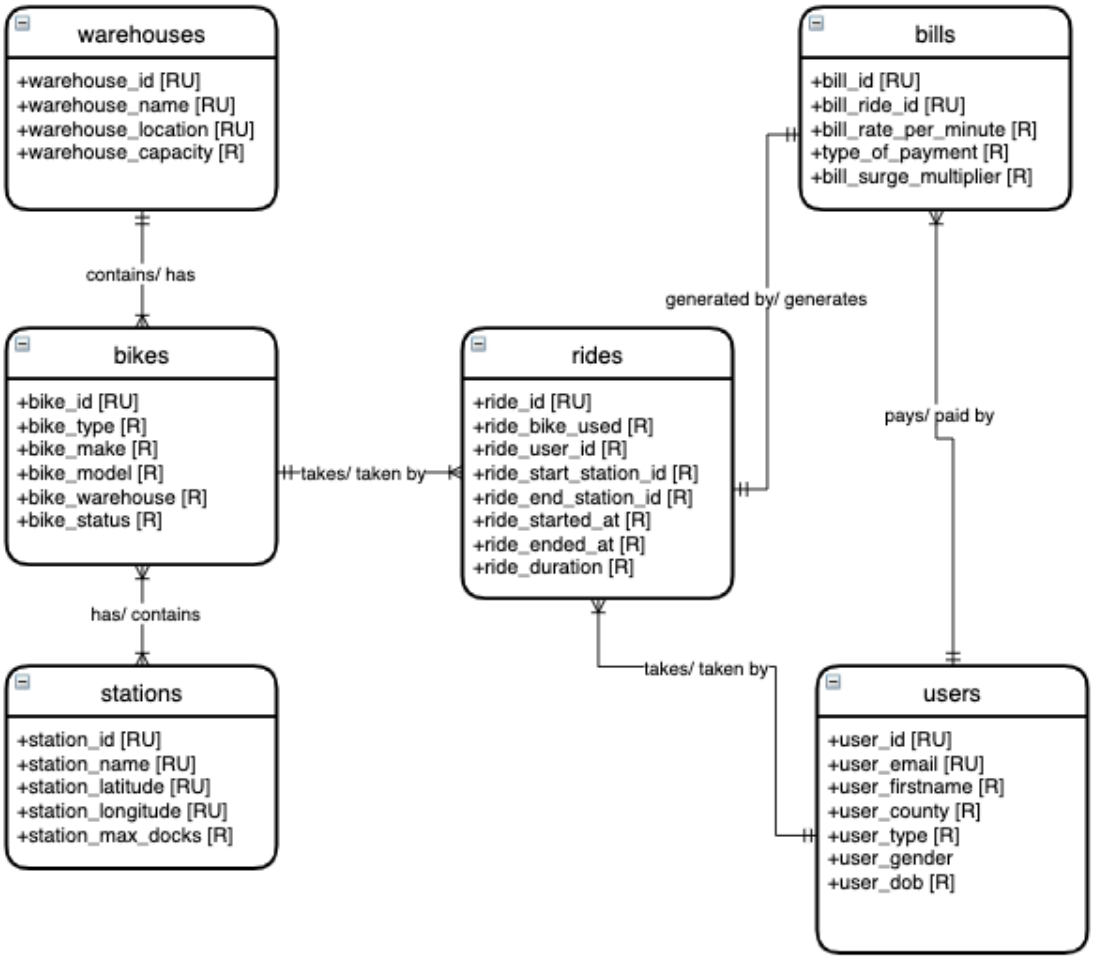
For a program that clocks in 100,000 trips in a single day, there are infinite data points that get captured but may not prove to be insightful for the organization. Additionally, for large scale use cases like Citi Bike, the data that is being captured needs to be mapped and processed accurately in order identify opportunities and ensure the progress of the program. Hence, it becomes extremely important to maintain and manage the data being collected and processed well. This will not only help in taking informed decisions but also minimize uncertainty and uncover potential gaps for improvement.

For the above reasons, we decided to take up the Citi Bike Database in order to analyze and identify insights that could potentially highlight scope for new initiatives and improvements to the existing setup. Our goal for this project was to provide actionable insights to the Business teams for certain problem statements by cleaning, processing, modifying and managing the data through the raw data dump that we sourced from the Citi Bike official website. Through this project, we would highlight the data points that would indicate growth opportunities, streamline operations and eventually have a positive impact on the revenue for the Citi Bike business.

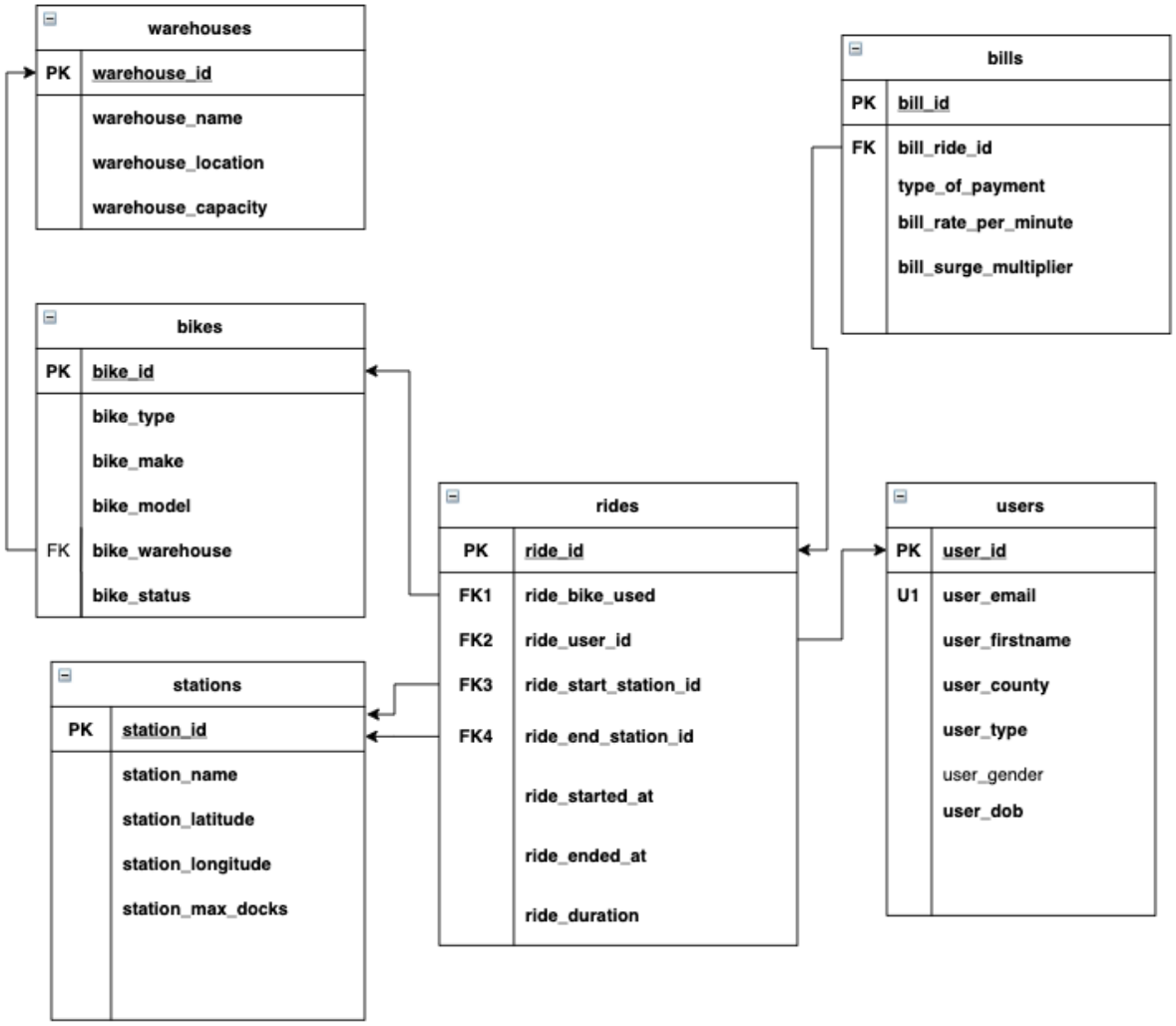
Post implementation of our system, we will be able to provide data-backed answers to key Business use cases like:

a) Average trip duration - This will help the Business teams to streamline their operations for the allocation of bikes on docking stations that will eventually optimize the number of trips per bike
b) Popular bike make & model based on trip duration - Identifying top bike make & model based on trip duration to assist the Business teams in deciding to capitalize more on a particular make and model of the bike. This will also help them identify what bikes are more efficient and trustworthy for longer/shorter trip durations thus ensuring better customer experience
c) Average age of the customer riding bike - This will help the management in deciding whether the bikes need certain age-specific modifications to enhance rider comfort and experience. This will also help in planning duration per trip and thereby help in improving the turnaround time per ride.

Conceptual Model Diagram



Logical Data Model Diagram



Data Dictionary

Warehouses

Consists information about the warehouses for the bikes

Entity Name: warehouses	Entity and Attributes	Field Type	Nullable	Foreign Key constraints	Description
Primary Key	warehouse_id	int	Not Null		Unique number of every warehouse
	warehouse_name	varchar(50)	Not Null		Unique name of every warehouse
	warehouse_location	varchar(50)	Not Null		Unique location of every warehouse
	warehouse_capacity	int	Not Null		Capacity of every warehouse

Bikes

Consists information about the bikes that are part of the program

Entity Name: bikes	Entity and Attributes	Field Type	Nullable	Foreign Key constraints	Description
Primary Key	bike_id	int	Not Null		Unique id of every bike
	bike_type	varchar(20)	Not Null		Type of bike
	bike_make	varchar(20)	Not Null		Make of bike
	bike_model	int	Not Null		Model of bike
Foreign Key	bike_warehouse	int	Not Null	Table warehouses on (warehouse_id)	Warehouse of bike

	bike_status	varchar(20)	Not Null		Status of bike

Stations

Contains information about the docking stations

Entity Name: stations	Entity and Attributes	Field Type	Nullable	Foreign Key constraints	Description
Primary Key	station_id	varchar(20)	Not Null		Id of station
	station_name	varchar(50)	Not Null		Name of station
	station_latitude	float	Not Null		Latitude of station
	station_longitude	float	Not Null		Longitude of station
	station_max_docks	int	Not Null		Number of docks at each station

Rides

Contains information about the details of the rides

Entity Name: rides	Entity and Attributes	Field Type	Nullable	Foreign Key constraints	Description
Primary Key	ride_id	varchar(30)	Not Null		Unique id of ride
Foreign Key	ride_bike_used	int	Not Null	Table bikes on (bike_id)	Id of bike used for ride
Foreign Key	ride_user_id	int	Not Null	Table users on (user_id)	User id taking ride
Foreign Key	ride_start_station_id	varchar(20)	Not Null	Table stations on (station_id)	Start station id
Foreign Key	ride_end_station_id	varchar(20)	Not Null	Table stations on	End station id

				(station_id)	
	ride_started_at	datetime2(7)	Not Null		Ride starting time
	ride_ended_at	datetime2(7)	Not Null		Ride ending time
	ride_duration	time(7)	Not Null		Duration of the ride

Users

Contains information about the users of Citi Bike

Entity Name: users	Entity and Attributes	Field Type	Nullable	Foreign Key constraints	Description
Primary Key	user_id	int	Not Null		Unique id of user
Unique Key	user_email	varchar(50)	Not Null		Unique email id of user
	user_firstname	varchar(50)	Not Null		First name of user
	user_county	varchar(50)	Not Null		County of user
	user_type	varchar(50)	Not Null		Type of user
	user_gender	varchar(5)			Gender of user
	user_dob	date	Not Null		Date of birth of user

Bills

Contains information about the billing for the rides

Entity Name: bills	Entity and Attributes	Field Type	Nullable	Foreign Key constraints	Description
Primary Key	bill_id	int	Not Null		Unique id of bill
Foreign Key	bill_ride_id	varchar(30)	Not Null	Table rides on (ride_id)	Unique ride id
	type_of_payment	varchar(50)	Not Null		Type of payment method
	bill_rate_per_minute	float	Not Null		Per minute ride rate
	bill_surge_multiplier	int	Not Null		Surge multiplier per ride

Business Rules

- A customer can book multiple rides from his account
- A ride once booked can be canceled by the customer
- Customer pays for the ride after the trip is over and the bike is docked back at the station
- Customer can pay for the ride from the available options and from the one selected while booking the ride
- Pick up and drop off station for a ride must not necessarily be the same and can be at different locations
- Surge multiplier is levied on normal rates during peak hours
- User gender can have null values as well
- Every customer has a unique user id, ride id
- At a given time, only one customer can ride the bike

Database System Infrastructure

We used the following tools to create and implement this project:

1) Draw.io : We created the entity-relationship diagram for the conceptual model and the logical data model diagram using the draw.io tool. We used this tool to create the entities and for defining their attributes and keys for our implementation.

2) SQL Server : We utilized the SQL server for storing all the tables and its corresponding data. We created tables in the database using SQL queries.

3) Power Apps: Used for building the front-end of the application

4) SQL Server Import : Due to the high volume of the data set, we utilized this Azure Data Studio wizard for importing data from CSV file into SQL Server.

5) Azure Data Studio : We used the ADS tool for creating the database and for querying the data to provide insights and analysis for the problem statements.

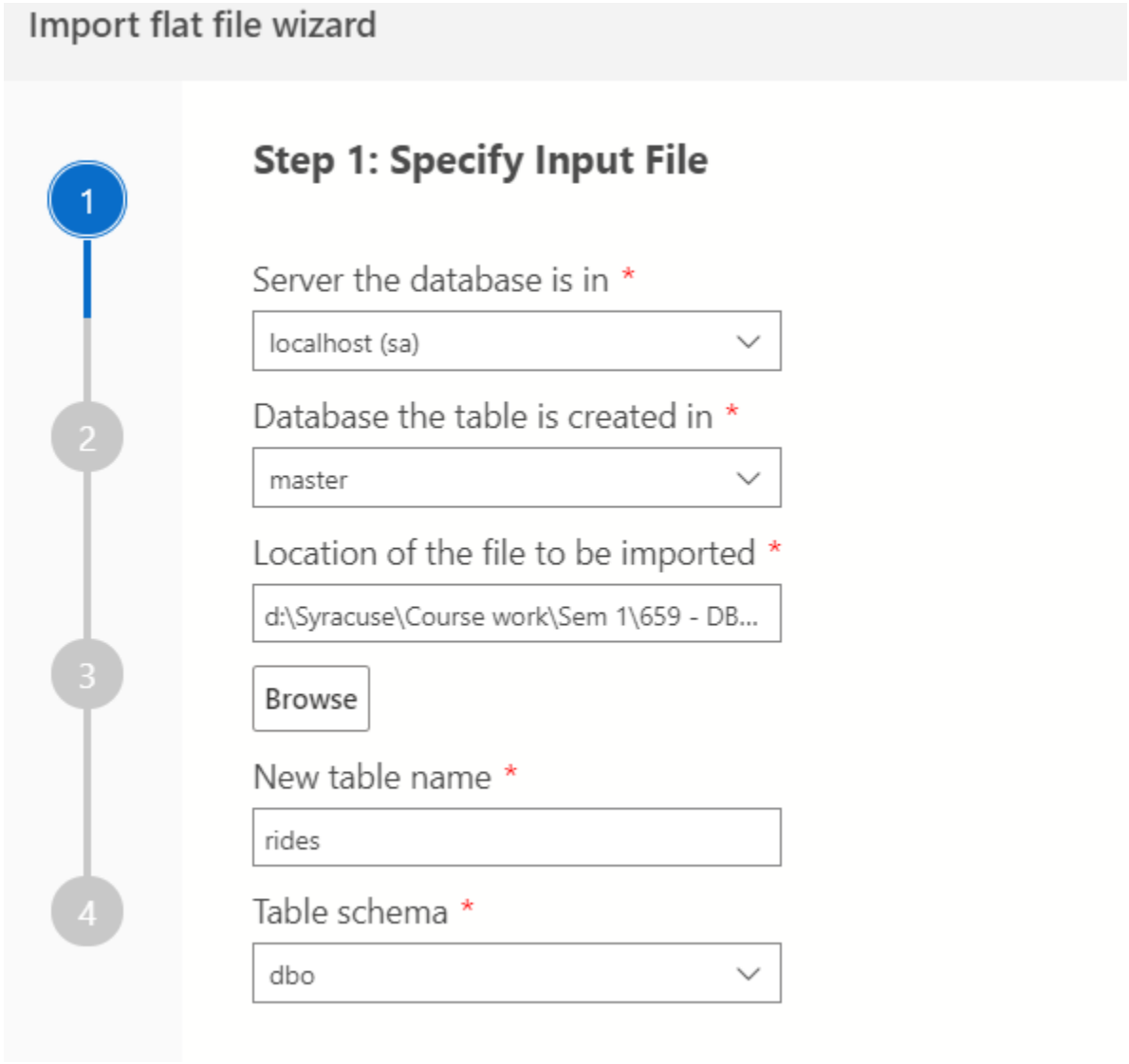
Creating Tables and Inserting Sample Data

```
create DATABASE citibikes

use citibikes
GO

--DOWN
if exists (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where CONSTRAINT_NAME='fk_bills_ride_id')
    alter table bills drop constraint fk_bills_ride_id
if exists (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where CONSTRAINT_NAME='fk_bikes_bike_warehouse')
    alter table bikes drop constraint fk_bikes_bike_warehouse
if exists (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where CONSTRAINT_NAME='fk_rides_ride_user_id')
    alter table rides drop constraint fk_rides_ride_user_id
if exists (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where CONSTRAINT_NAME='fk_rides_ride_bike_used')
    alter table rides drop constraint fk_rides_ride_bike_used
if exists (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where CONSTRAINT_NAME='fk_rides_ride_end_station_id')
    alter table rides drop constraint fk_rides_ride_end_station_id
if exists (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where CONSTRAINT_NAME='fk_rides_ride_start_station_id')
    alter table rides drop constraint fk_rides_ride_start_station_id
if exists (select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where CONSTRAINT_NAME='u_users_user_email')
    alter table users drop constraint u_users_user_email


drop table if exists warehouses
drop table if exists bills
drop table if exists bikes
drop table if exists stations
drop table if exists rides
drop table if exists users
GO

--UP
create table users
(
    user_id int identity not null,
    user_firstname varchar(50) not null,
    user_county varchar(50) not null,
    user_type varchar(50) not null,
```

```
    user_gender varchar(5) null,
    user_dob date not null,
    user_email varchar(50) not null,
    constraint pk_users_user_id primary key(user_id),
    constraint u_users_user_email unique (user_email)
)

CREATE TABLE rides
(
    ride_id varchar(30) not null,
    ride_started_at DATETIME2(7) not null,
    ride_ended_at DATETIME2(7) not null,
    ride_start_station_id varchar(20) not null,
    ride_end_station_id varchar(20) not null,
    ride_duration TIME(7) not null,
    ride_bike_used int not null,
    ride_user_id int not null,
    constraint pk_rides_ride_id primary key(ride_id)
)

create table stations
(
    station_id varchar(20) not null,
    station_latitude FLOAT not null,
    station_longitude FLOAT not null,
    station_name varchar(50) not null,
    station_max_docks int not null,
    constraint pk_stations_station_id primary key(station_id)
)

create table bikes
(
    bike_id int identity not null,
    bike_type varchar(20) not null,
    bike_make varchar(20) not null,
    bike_model int not null,
    bike_status varchar(20) not null,
    bike_warehouse int not null,
    constraint pk_bikes_bike_id primary key(bike_id)
)

create table bills
(
    bill_id int identity not null,
    bill_ride_id varchar(30) not null,
    bill_rate_per_minute float not null,
    bill_surge_multiplier int not null,
```

```
    bill_type_of_payment varchar(50) not null,
    constraint bills_bill_id primary key(bill_id)
)

create table warehouses
(
    warehouse_id int identity not null,
    warehouse_location varchar(50) not null,
    warehouse_name varchar(50) not null,
    warehouse_capacity int not null,
    constraint warehouses_warehouse_id
        primary key(warehouse_id)
)

alter table rides
    add constraint fk_rides_ride_start_station_id foreign key (ride_start_station_id)
        references stations(station_id)
alter table rides
    add constraint fk_rides_ride_end_station_id foreign key (ride_end_station_id)
        references stations(station_id)
alter table rides
    add constraint fk_rides_ride_bike_used foreign key (ride_bike_used)
        references bikes(bike_id)
alter table rides
    add constraint fk_rides_ride_user_id foreign key (ride_user_id)
        references users(user_id)

alter table bikes
    add constraint fk_bikes_bike_warehouse foreign key (bike_warehouse)
        references warehouses(warehouse_id)

alter table bills
    add constraint fk_bills_bill_ride_id foreign key (bill_ride_id)
        references rides(ride_id)
alter table users
    add constraint u_users_user_email
        unique (user_email)
```

Steps Showing Insertion of Data from CSV File



Import flat file wizard

1
2
3
4

Step 2: Preview Data

This operation analyzed the input file structure to generate the preview below for up to the first 50 rows.

ride_id	ride_started_at	ride_ended_at	ride_start_st...	ride_end_sta...	ride_duration	ride_bike_us...	ride_user_id
0FA0B0B972...	19-10-2021 0...	19-10-2021 0...	JC094	JC056	00:02:40	236	451
0702BED6EB...	16-10-2021 1...	16-10-2021 1...	JC094	JC056	00:05:00	237	622
A8810ECCC6...	29-10-2021 1...	29-10-2021 1...	JC094	JC056	00:05:00	31	15
C9A0FF2B822...	08-10-2021 1...	08-10-2021 1...	JC094	JC056	00:04:34	446	411
7DFDEB8EF4...	14-10-2021 0...	14-10-2021 0...	JC094	JC056	00:06:23	469	344
D5EC49F129...	17-10-2021 1...	17-10-2021 1...	JC094	JC056	00:03:43	446	5
C3AE382C29...	08-10-2021 1...	08-10-2021 1...	HB202	HB305	00:06:36	264	183
C3825383AB...	26-10-2021 1...	26-10-2021 1...	HB202	HB305	00:08:16	263	509
B614161AEB4...	05-10-2021 1...	05-10-2021 1...	HB202	HB305	00:07:28	21	552
7DF6438A67...	20-10-2021 1...	20-10-2021 1...	JC099	JC055	00:12:31	423	2
57933E92CCE...	22-10-2021 1...	22-10-2021 1...	HB202	HB502	00:03:36	476	638
78AF1615F97...	28-10-2021 1...	28-10-2021 1...	HB202	HB502	00:02:11	196	421
17D6977715E...	12-10-2021 0...	12-10-2021 0...	HB202	HB502	00:03:26	107	237
5B62FBC878F...	24-10-2021 0...	24-10-2021 0...	JC099	JC020	00:05:05	49	329
DA1D1E6BF5...	02-10-2021 1...	02-10-2021 2...	JC099	JC020	01:08:20	357	586

Import flat file wizard

1
2
3
4

Step 3: Modify Columns

Column Name	Data Type	Primary Key ☐	Allow Nulls ☐
ride_id	varchar(30)	☑	☐
ride_started_at	datetime2(7)	☐	☐
ride_ended_at	datetime2(7)	☐	☐
ride_start_station_id	varchar(20)	☐	☐
ride_end_station_id	varchar(20)	☐	☐
ride_duration	time(7)	☐	☐
ride_bike_used	int	☐	☐
ride_user_id	int	☐	☐

Answering Data Questions for Insights

Below listed are the major data questions that can be answered to provide valuable insights to the Business teams that can uncover potential gaps, scope for improvements and opportunities for new initiatives:

1) What is the Average Duration of Each Ride?

   This will assist the Business teams in streamlining their operations for the allocation of bikes on docking stations and also help in optimizing the number of trips per bike

```
select
  cast(cast(avg(cast(cast(ride_duration as datetime) as float)) as datetime) as
time) AvgDuration
from rides;
```

   Output Screenshot:

	AvgDuration
1	00:17:12.5333333

2) What is the Average Age of Riders using Citi Bikes ?

   This will help the Business teams in deciding whether the bikes need certain age-specific modifications to enhance rider comfort and experience and also give insights on the duration per trip and thereby help in improving the turnaround time per ride.

```
select AVG(DATEDIFF(hour,user_dob,GETDATE())/8766) as AverageRiderAge
from users
GO
```

   Output Screenshot:

	AverageRiderAge
1	51

3) Pivot Table of Current Bike Inventory - categorization basis manufacturer & functionality

This query displays the current inventory of bikes based on a particular model and manufacturer and depending on whether they are functional or non-functional. This will help the Business teams brainstorm whether they need to increase the inventory of a particular bike manufacturer/model.

```
with bikes_by_make as (
    select bike_id, bike_make, bike_status
    from bikes
)
select *
from bikes_by_make pivot(
    count(bike_id)
        for bike_status in (functional, non_functional)
) pivot_query
GO
```

Output Screenshot:

	bike_make	functional	non_functional
1	Aventure	125	30
2	Guardian	125	25
3	Kent	125	19
4	Schwinn	125	26

```
with bikes_by_make_model as (
    select bike_id, bike_make, bike_model, bike_status
    from bikes
)
select *
from bikes_by_make_model pivot(
    count(bike_id)
        for bike_status in (functional, non_functional)
) pivot_query
Go
```

Output Screenshot:

	bike_make	bike_model	functional	non_functional
1	Aventure	1	41	8
2	Guardian	1	42	9
3	Kent	1	42	6
4	Schwinn	1	42	10
5	Aventure	2	42	11
6	Guardian	2	41	6
7	Kent	2	42	7
8	Schwinn	2	42	7
9	Aventure	3	42	11
10	Guardian	3	42	10
11	Kent	3	41	6
12	Schwinn	3	41	9

4) Categorization of bikes based on trip duration:

This will help the think-tank identify the bike make that is preferred by customers depending on trip duration i.e for this current question we have considered bikes used for trips >15 minutes and trips <15 minutes. Depending on this data, Business teams can decide whether they want to increase or decrease the inventory of certain bike models.

Trips >15 minutes (longer duration)

```
select b.bike_make,
      count(r.ride_id) as ride_count
   from rides as r
      join bikes as b on b.bike_id=r.ride_bike_used
   where datepart(mi,(cast(r.ride_duration as datetime)))>15
   group by b.bike_make
   order by count(r.ride_id) desc
```

Output Screenshot:

	bike_make	ride_count
1	Aventure	835
2	Schwinn	134
3	Guardian	134
4	Kent	128

Trips <15 minutes (shorter duration)

```
select b.bike_make,
      count(r.ride_id) as ride_count
   from rides as r
      join bikes as b on b.bike_id=r.ride_bike_used
   where datepart(mi,(cast(r.ride_duration as datetime)))<15
   group by b.bike_make
   order by count(r.ride_id) desc
GO
```

Output Screenshot:

	bike_make	ride_count
1	Guardian	1902
2	Aventure	628
3	Schwinn	582
4	Kent	530

5) Customer Lifetime Value(LTV) on Citi Bike:

Identifying the Customer LTV will assist the organization in implementing retention and targeting strategies to ensure smoother and efficient customer experience for users who have a higher LTV in the program

```
select distinct top 10 u.user_id,
      u.user_firstname,
      u.user_type,
      count(r.ride_user_id) over (partition by u.user_id) as total_rides,
```

```
        sum(datepart(mi,(cast(r.ride_duration as datetime)))) over (partition by
u.user_id) as total_minutes,
        CASE
            WHEN u.user_type='Subscriber'
            THEN sum(cast(sum(datepart(mi,(cast(r.ride_duration as
datetime)))*b.bill_rate_per_minute*b.bill_surge_multiplier) as decimal(6,2))/1.5)
                OVER (PARTITION BY u.user_id)
            WHEN u.user_type='Customer'
            THEN sum(cast(sum(datepart(mi,(cast(r.ride_duration as
datetime)))*b.bill_rate_per_minute*b.bill_surge_multiplier) as decimal(6,2)))
                OVER (PARTITION BY u.user_id)
        END AS [Life_Time_Value ($)]
        from bills as b
join rides as r on b.bill_ride_id=r.ride_id
join users as u on u.user_id = r.ride_user_id
group by u.user_id,
         u.user_id,
         u.user_type,
         u.user_firstname,
         r.ride_duration,
         b.bill_rate_per_minute,
         b.bill_surge_multiplier,
         r.ride_user_id
order by [Life_Time_Value ($)] desc
```

Output Screenshot:

	user_id	user_firstname	user_type	total_rides	total_minutes	Life_Time_Value ($)
1	333	MARYAM	Customer	15	323	381.72
2	669	MARY	Customer	10	217	261.40
3	304	SANTIAGO	Customer	16	212	227.00
4	557	JAYDA	Customer	14	166	226.06
5	493	SHAINA	Customer	11	256	221.86
6	425	FAIGY	Subscriber	12	326	221.43
7	403	RAYMOND	Customer	10	211	205.40
8	390	MARIO	Customer	10	172	201.32
9	448	MARTIN	Customer	5	103	197.60
10	635	JORDYN	Customer	13	148	193.94

6) Station wise Payment method usage analysis:

We analyzed the payment methods that were used at each station to identify the usage of each payment type that could potentially help the Business teams in implementing any specific optimizations/marketing offers for a specific payment type at a particular docking station.

```
with paymentType as (
    select s.station_name,
    b.type_of_payment from
    bills as b
    join rides as r on b.bill_ride_id=r.ride_id
    join stations as s on s.station_id=r.ride_start_station_id
)
select *
from paymentType pivot(
    count(type_of_payment)
        for type_of_payment in ([cash], [Debit/Credit Card],[Digital
Wallet/ApplePay])
) pivot_query
```

Output Screenshot:

	station_name	cash	Debit/Credit Card	Digital Wallet/ApplePay
1	Hilltop	3	6	5
2	Paulus Hook	45	49	46
3	Lincoln Park	24	28	21
4	Hoboken Terminal - Hudson St…	12	20	13
5	Communipaw & Berry Lane	2	2	0
6	Grove St PATH	6	6	6
7	2 St HBLR - 2 St & Marshall …	4	1	2
8	Madison St & 10 St	12	12	15
9	14 St Ferry - 14 St & Shipya…	39	29	39
10	Dey St	32	38	33
11	Adams St & 11 St	63	52	49
12	Riverview Park	2	2	2

7) Dashboard displaying count of rides at each station:

This will help the Business teams in identifying the frequency of bikes that are utilized more or less in a particular geographical area eventually making scope for better planning of resources and leading to higher revenue.

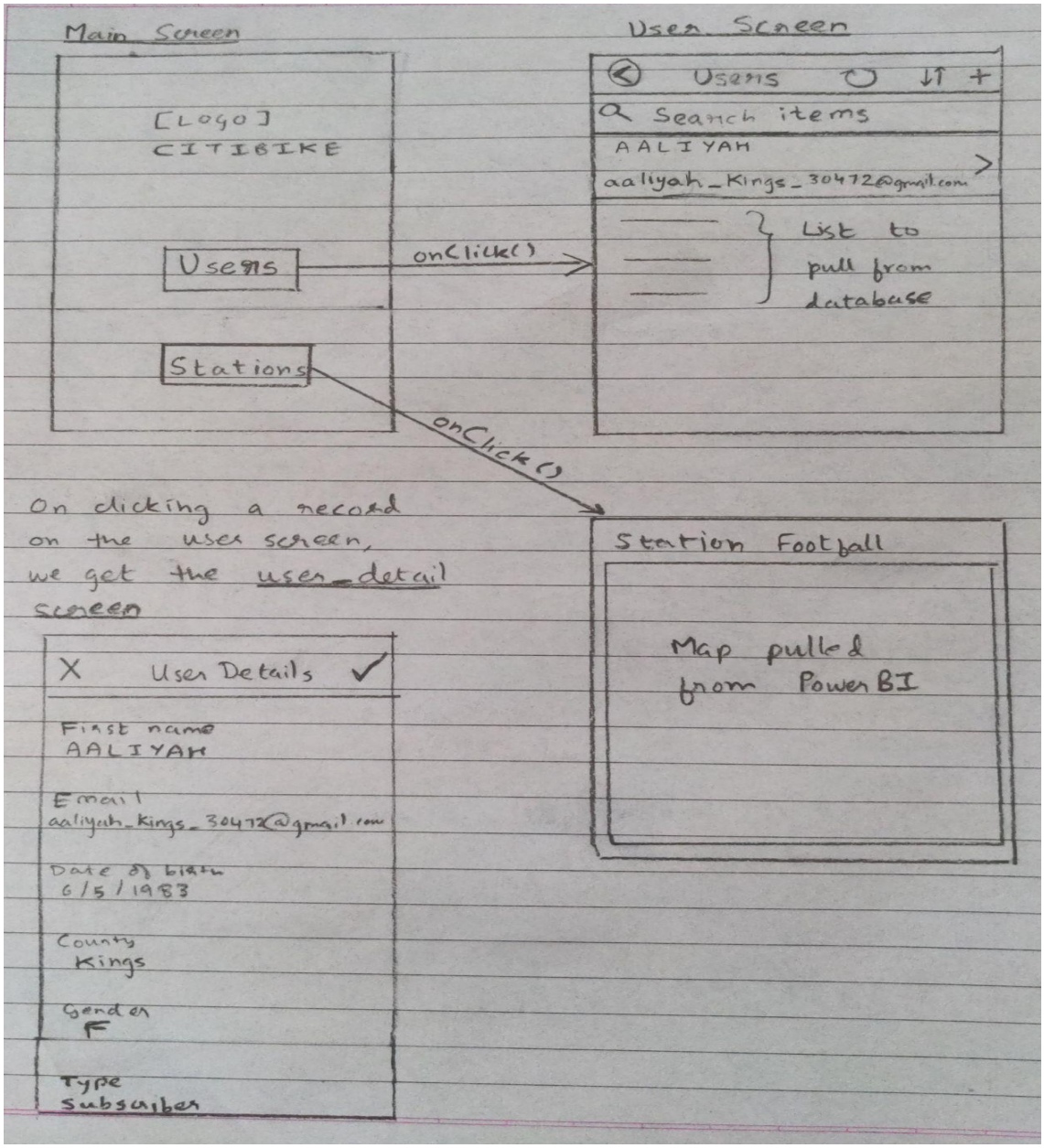
```
drop view if exists stations_visited
drop view if exists station_footfall_data
GO

create view stations_visited as(
   select ride_id, ride_start_station_id as ride_station_id
      from rides as strt
   UNION
   select ride_id, ride_end_station_id from rides as dest
)
GO

create view station_footfall_data as (
   select station_id, station_latitude, station_longitude, count(*) as ride_count
   from stations_visited
      join stations as s on station_id=ride_station_id
   group by station_id, station_latitude, station_longitude
)
GO

select * from station_footfall_data
GO
```

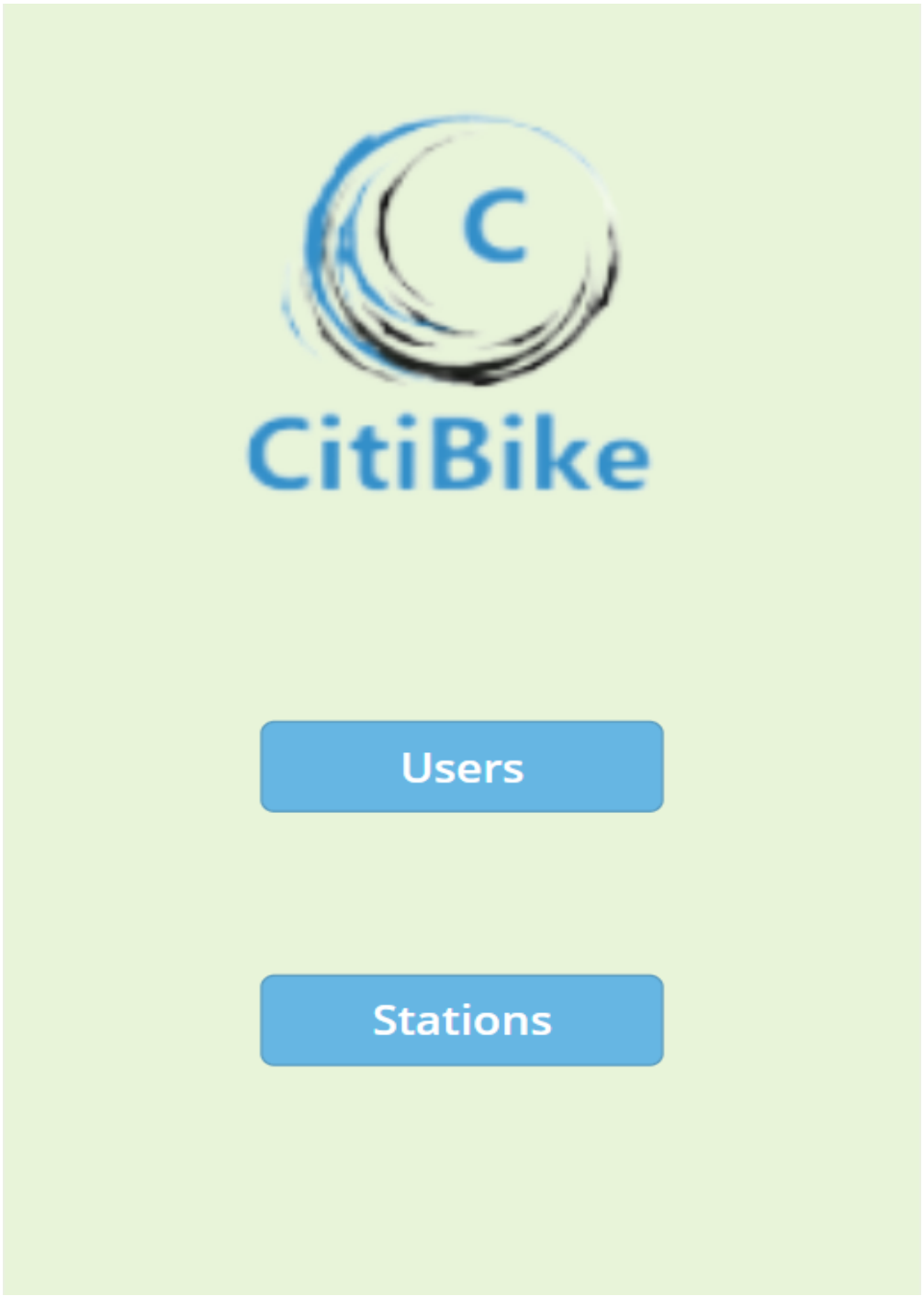
Output Screenshot

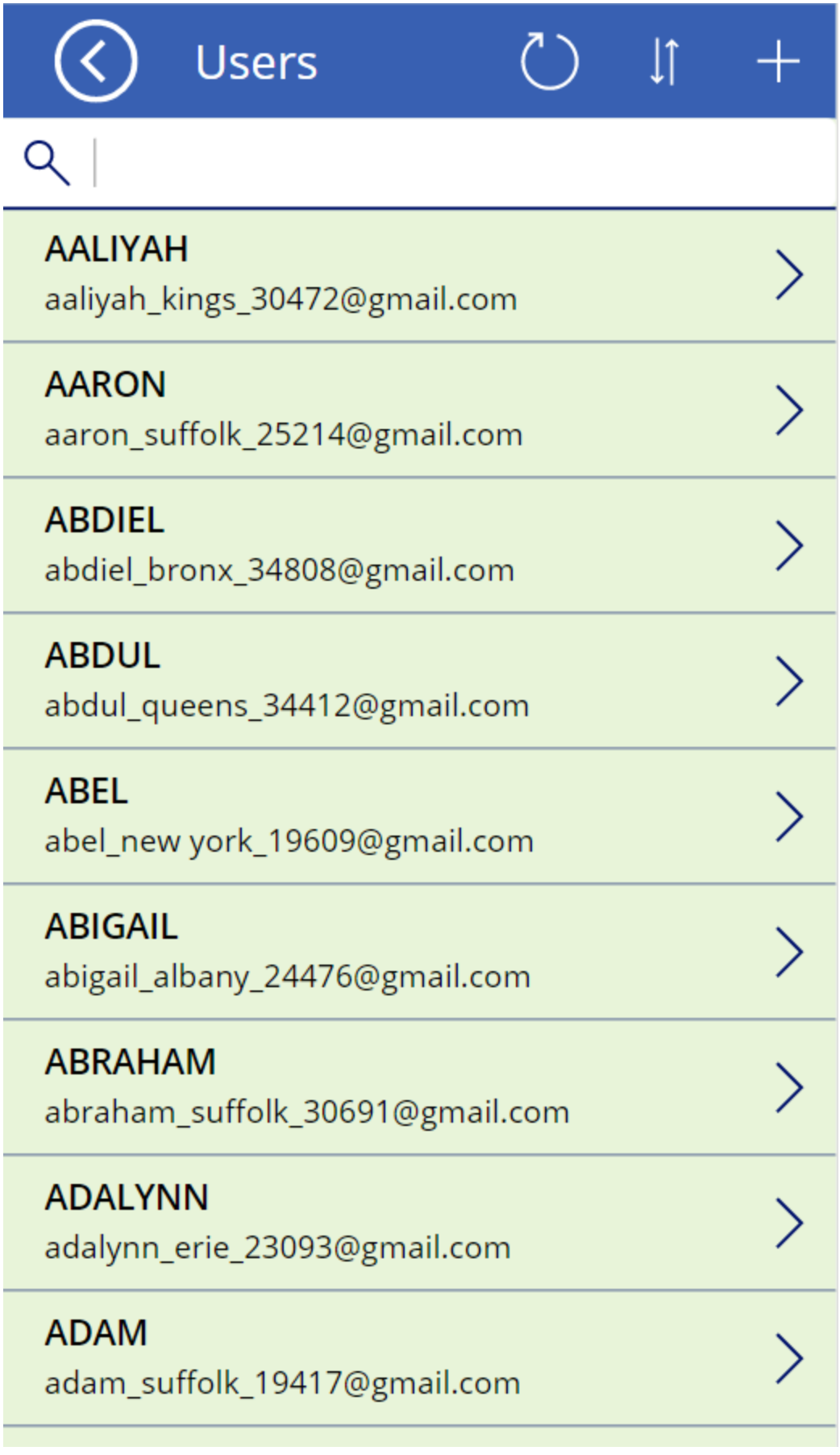

STATION FOOTFALL
Wood-Ridge
Fort Lee
East Rutherford
Eight Day Swamp
Ridgefield
Edgewater
Rutherford
Walden Swamp
Cliffside Park
MetLife Stadium
Fairview
Hackensack
North Bergen
West New York
Secaucus
Union City
Central Park
Weehawken
Manhattan
Manhattan
WEST END
Lincroft
TRIBECA
BAYSIDE-SOUTH
FDR Dr S
COUNTRY VILLAGE
LSP INDUSTRIAL
Brooklyn
GOVERNORS ISLAND
GREENVILLE YARDS
RED HOOK
Bayonne
Upper New York Bay
GREENWOOD
Staten Island
SUNSET PARK
PARKVILLE
WEST NEW BRIGHTON
BAY RIDGE
BOROUGH PARK
FOX HILLS
NEW UTRECHT

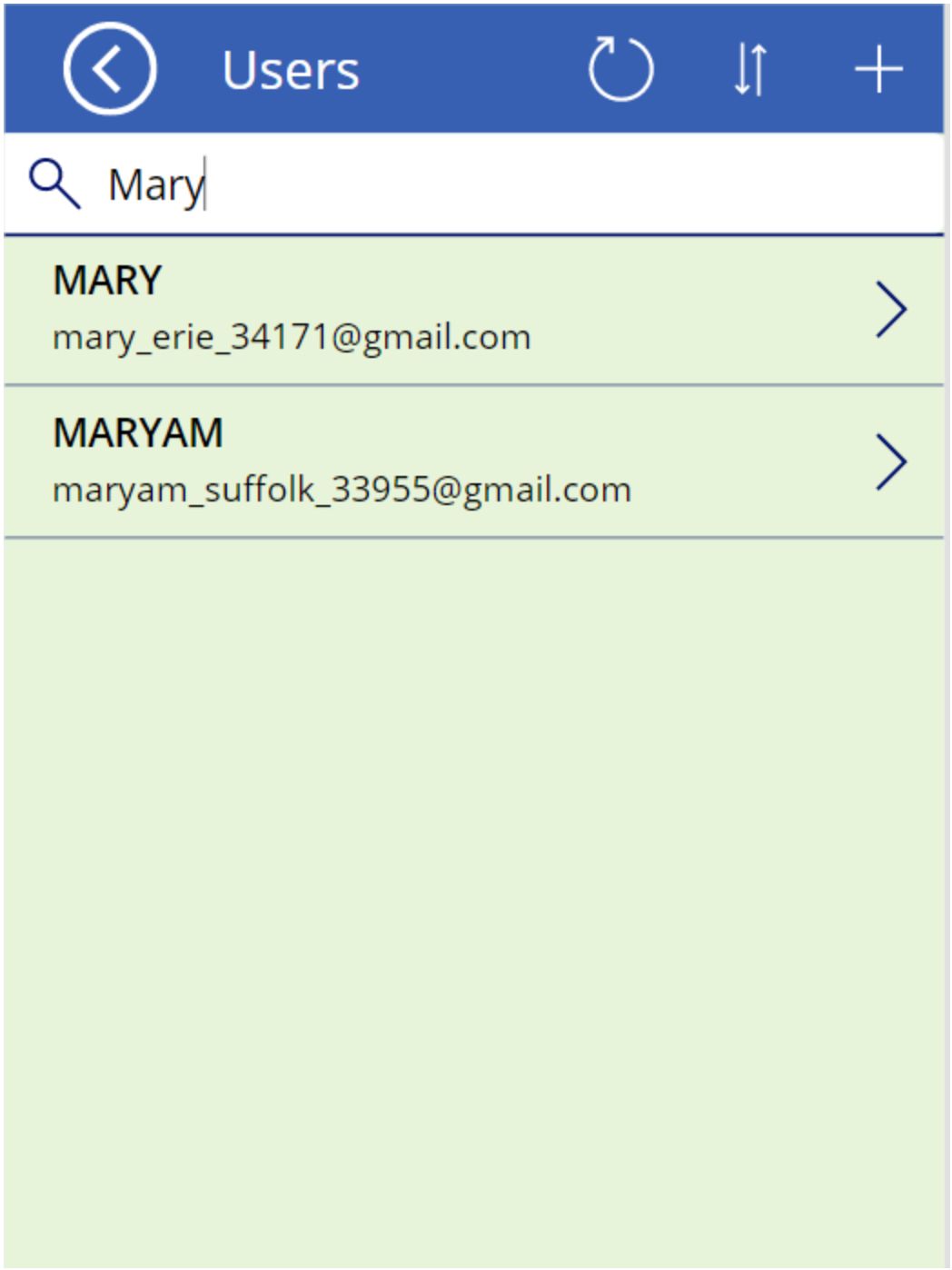
Interface Implementation - Power Apps

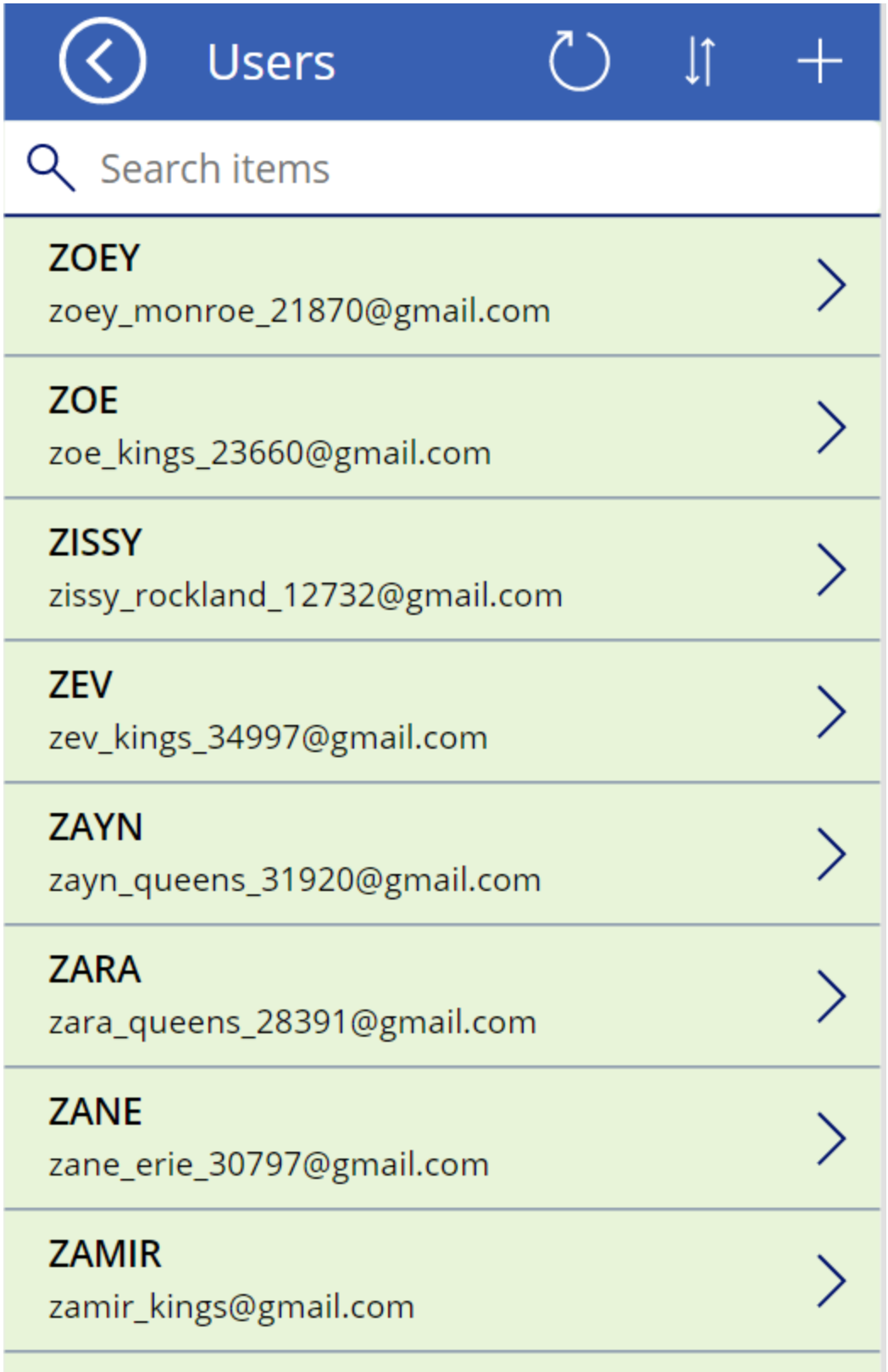
User Interface Design Wireframe

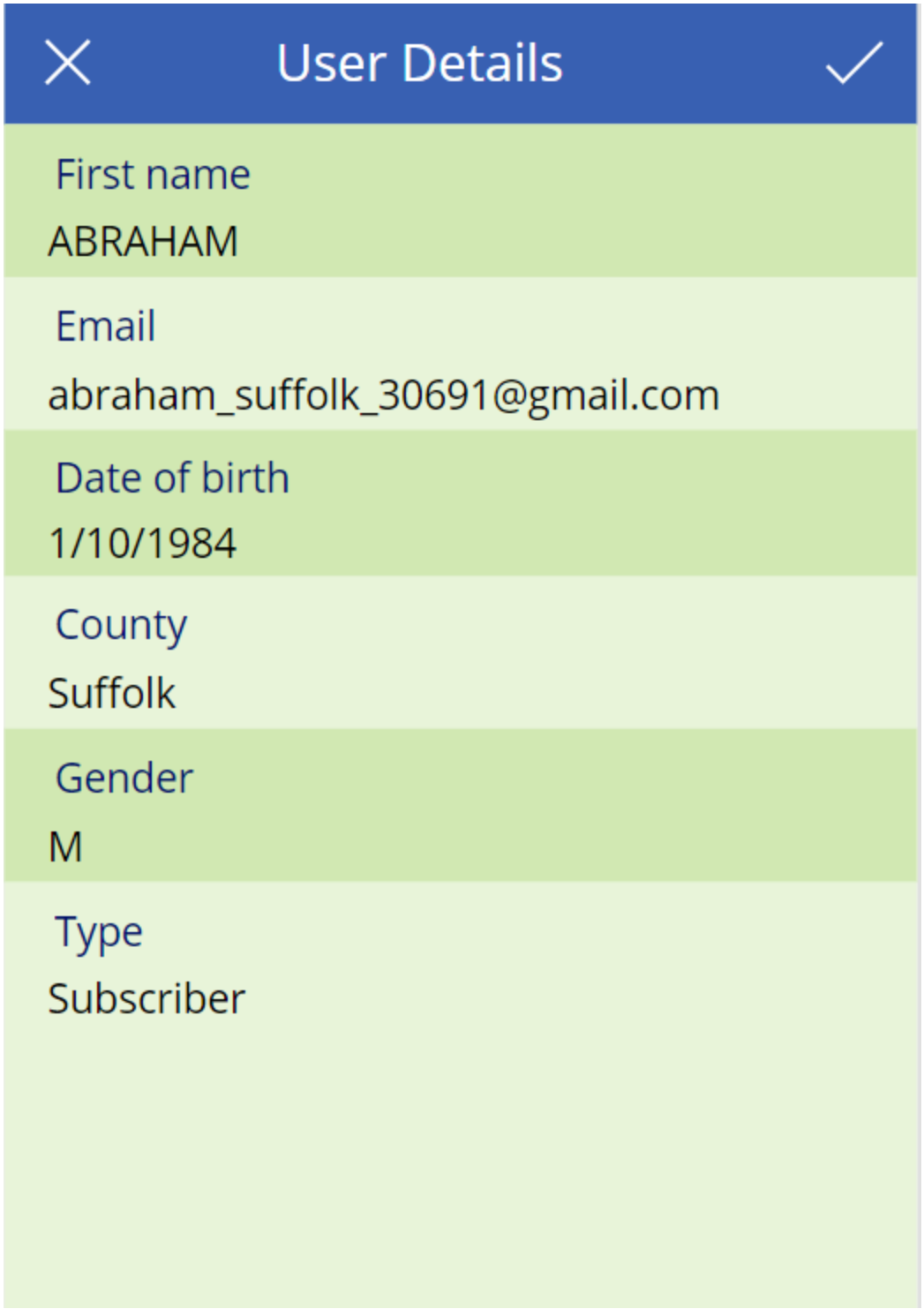



CitiBike
Users
Stations

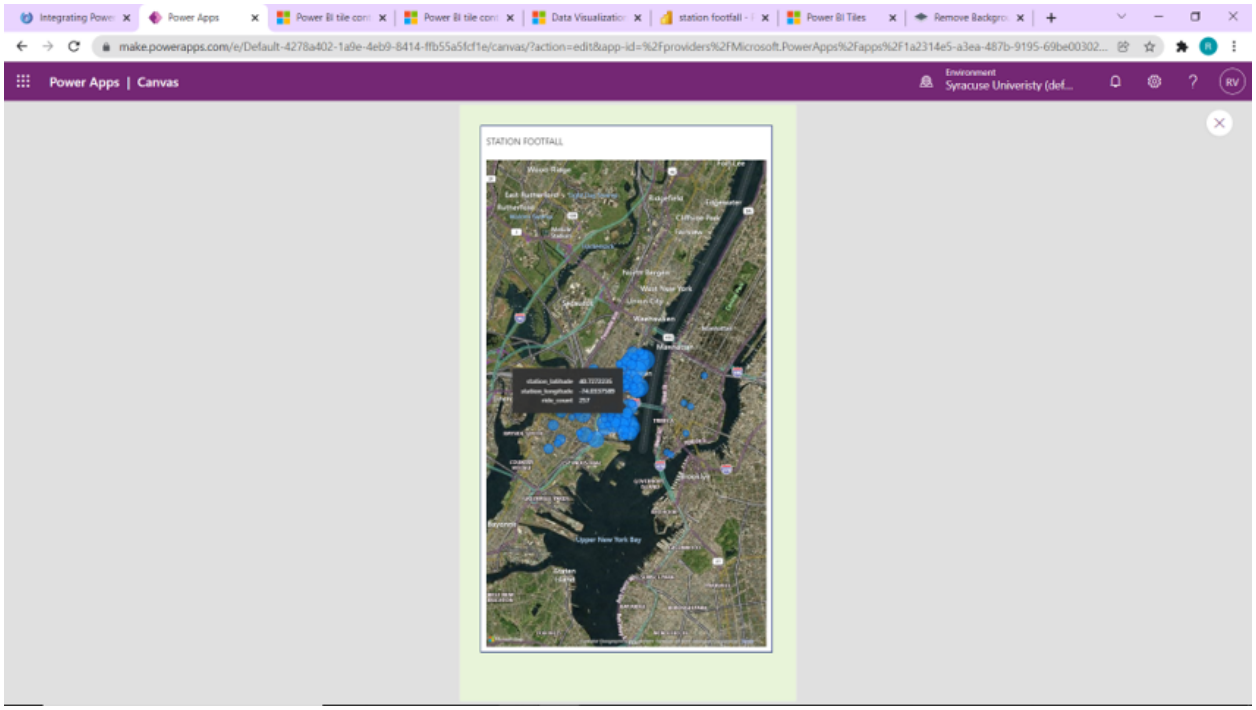

Users
AALIYAH
aaliyah_kings_30472@gmail.com
AARON
aaron_suffolk_25214@gmail.com
ABDIEL
abdiel_bronx_34808@gmail.com
ABDUL
abdul_queens_34412@gmail.com
ABEL
abel_new york_19609@gmail.com
ABIGAIL
abigail_albany_24476@gmail.com
ABRAHAM
abraham_suffolk_30691@gmail.com
ADALYNN
adalynn_erie_23093@gmail.com
ADAM
adam_suffolk_19417@gmail.com

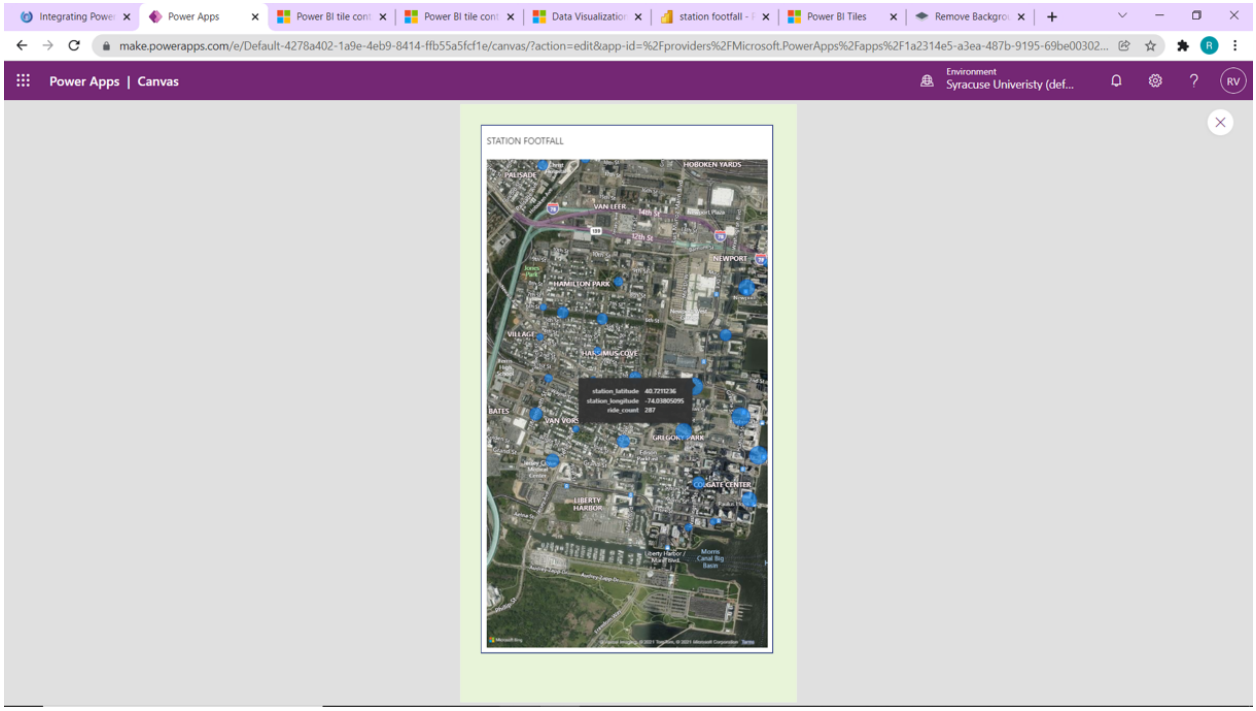

Users
Mary
MARY
mary_erie_34171@gmail.com
MARYAM
maryam_suffolk_33955@gmail.com


Users
Search items
ZOEY
zoey_monroe_21870@gmail.com
ZOE
zoe_kings_23660@gmail.com
ZISSY
zissy_rockland_12732@gmail.com
ZEV
zev_kings_34997@gmail.com
ZAYN
zayn_queens_31920@gmail.com
ZARA
zara_queens_28391@gmail.com
ZANE
zane_erie_30797@gmail.com
ZAMIR
zamir_kings@gmail.com


User Details
First name
ABRAHAM
Email
abraham_suffolk_30691@gmail.com
Date of birth
1/10/1984
County
Suffolk
Gender
M
Type
Subscriber


STATION FOOTFALL
Wood-Ridge
Fort Lee
East Rutherford
Eight Day Swamp
Ridgefield
Edgewater
Rutherford
Walden Swamp
Cliffside Park
MetLife Stadium
Fairview
Hackensack
North Bergen
West New York
Secaucus
Union City
Central Park
Weehawken
Manhattan
Manhattan
WEST END
Lincroft
TRIBECA
BAYSIDE-SOUTH
FDR Dr S
COUNTRY VILLAGE
LSP INDUSTRIAL
Brooklyn
GOVERNORS ISLAND
GREENVILLE YARDS
RED HOOK
Bayonne
Upper New York Bay
GREENWOOD
Staten Island
SUNSET PARK
PARKVILLE
WEST NEW BRIGHTON
BAY RIDGE
BOROUGH PARK
FOX HILLS
NEW UTRECHT


Power Apps | Canvas
Environment
Syracuse Univeristy (def...
STATION FOOTFALL


Power Apps | Canvas
Environment
Syracuse Univeristy (def...
STATION FOOTFALL
HOBOKEN YARDS
VAN LEER
NEWPORT
HAMILTON PARK
HARRISON COVE
GRISCOM PARK
GATE CENTER
LIBERTY HARBOR