# DATABASE DESIGN  FOR GET-TO-PEDALIN

**Josue Sosa** **June 2023**

MGMT 3140

Relevant files:

| | |
|---|---|
| ERD_06_11_23 | Visio ERD |
| Database6_14_23V2 | Access database |
| data_dictionary | Excel data dictionary |
| Optional | |

Version control:

| Version | Changes |
|---|---|
| Database6_14_23V2 | Referential integrity (Access) |
| ERD_06_11_23V2 | Removed Subscriber and bike subtypes (Visio) |
| | |

# Part I. Database Initial Study

## About the Model Company

**Get-to-Pedalin' is a bike-share company that offers both electric and non-electric bikes for daily to monthly rides. The company currently faces a problem of limited availability, resulting in customers seeking alternative bike-share services. To address this issue and attract more customers, Get-to-Pedalin' needs to increase the number of bikes and stations throughout the city. Additionally, they aim to stay ahead of the competition by implementing convenient bike booking systems.**

## Problems and Constraints

**The primary problem is the insufficient availability of bikes, which leads to a loss of customers to other bike-share companies. The constraints include limited resources to expand our fleet and the need to compete with other established bike-share services in the market.**

## Project Objectives and Business Needs

**The main objective of Get-to-Pedalin' is to increase its customer base by addressing the problem of limited bike availability. To achieve this:**

1. **Increasing the number of stations and bikes by expanding its infrastructure, Get-to-Pedalin' can ensure wider availability of bikes throughout the city. Making it more convenient for customers to access its services.**
2. **Implement a bike booking system. Offering a bike booking feature will enhance accessibility and convenience for customers, enabling them to reserve bikes in advance and have peace of mind knowing a bike will be available when needed.**
3. **Stay ahead of the competition. Get-to-Pedalin' needs to maintain a competitive edge by providing superior services and incentives. Offering discounts to specific demographics, such as students and veterans, and providing special offers during city events can help attract more customers and differentiate their services from competitors.**

## Scope and Boundaries

**Get-to-Pedalin' Scope is:**

4. **Increasing the number of stations and bikes. This involves identifying suitable locations for additional stations, procuring more bikes, and establishing maintenance and repair services to ensure a reliable and well-maintained fleet.**
5. **Implementing a bike booking system. Get-to-Pedalin' will need to develop or adopt a reservation platform that allows customers to book bikes in advance, manage bookings, and integrate it with their existing operations.**
6. **Offering discounts to students and veterans. Get-to-Pedalin' will need to establish a verification process to ensure eligibility for these discounts and communicate the offers effectively to the target demographic.**
7. **Providing exclusive offers during events. Get-to-Pedalin' should identify relevant events in the city and create promotional campaigns to attract participants and attendees, leveraging the increased footfall during such occasions.**

**The boundaries are limited to our ability to expand our fleet, establish new stations, develop or adopt suitable booking systems, and offer discounts and specials.**

## Deliverables

**Booking Form**

| | |
|---|---|
| Subscriber ID: | |
| Arrival Date: | |
| Arrival Time: | |
| BikeType: | |
| Station Address: | |

**Booking Report**

| | |
|---|---|
| Subscriber ID: | |
| Arrival Date: | |
| Arrival Time: | |
| BikeType: | |
| Station Address: | |
| Bike Status: | |
| Booking ID: | |

# Part II. Database Design

## Business Rules

**Trip:**

- A trip can have one and only one bike and

each bike can have Zero to many trips.

- A trip can have one to many trip_station

and each trip_station can have one and only one trip.

- A trip can have one and only one subscriber

and each subscriber can have one too many trips.

**Trip_ station:**

- A trip_station can have one and only one

station, and each station can have one to many trip_stations.

**Subscriber:**

- A subscriber can have one and only one

subscription type either weekly or monthly.

**Bike:**

- A bike can have one and only one

bike type either traditional or electric.

- A bike can have Zero to many repair orders and each

repair order can be associated with one or only one bike.

**Repair_Order:**

- A repair order can have one to many Employee Repairs and

each employee repair can have one and only one repair order.

**Employee _Repairs:**

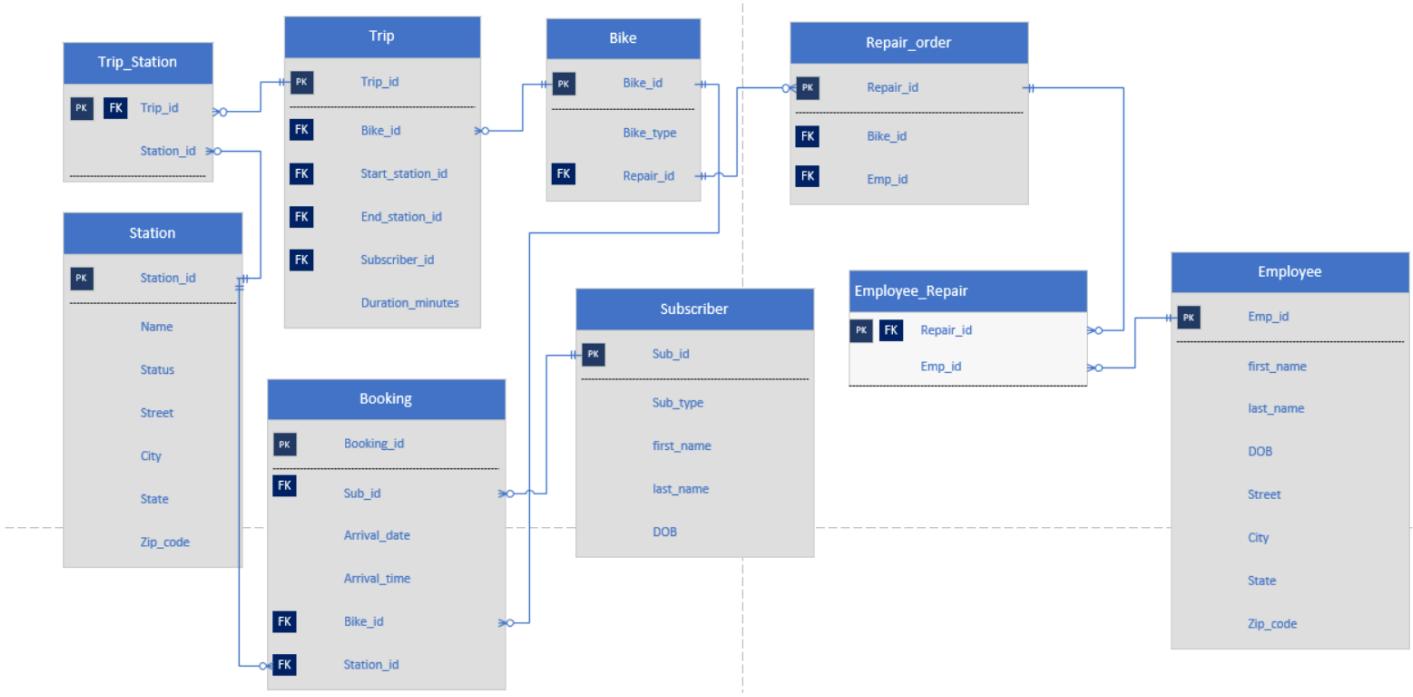
- An employee repair can have one and only one employee

and each employee can be associated with one-to-many employee repairs.

**Booking:**

- A subscriber can have zero-to-many bookings and each booking has one and only one subscriber.
- A bike can have zero-to-many bookings and each booking has one and only one bike.
- A station can be associated with one-to-many bookings and each booking has one and only station.

# Normalized ERD- Visio




# Simplified Data Dictionary

| Bike | |
|---|---|
| **Fields** | **Description** |
| Bike_id (PK) | Bike identification |
| Bike_type | If the bike is traditional or electric |
| Repair_id (FK) | Repair identification |

| Repair Order | |
|---|---|
| **Fields** | **Description** |
| repair_id (PK) | repair identification |
| repair_cost | cost of repair |
| Bike_ID (FK) | bike identification |

| Trips | |
|---|---|
| **Fields** | **Description** |
| trip_id (PK) | trip identification |
| bike_id (FK) | bike identification |
| Start_station_id(FK | Bike pickup station identification |
| end_station_id (FK | Bike dropoff station identification |
| duration_minutes | Length of trip(min) |
| subscriber_id(FK) | subscriber identification |

| Subscriber | |
|---|---|
| **Fields** | **Description** |
| sub_id(PK) | Subscriber identification |
| sub_type | Weekly or monthly subscription |
| first_name | Subscribe first name |
| last_name | Subscriber last name |
| DOB | Subscriber date of birth |

| Employee Repair | |
|---|---|
| **Fields** | **Description** |
| repair_id (PK,FK) | repair identification |
| emp_id (PK,FK) | Employee number |

| Trip_station | |
|---|---|
| **Fields** | **Description** |
| trip_id (PK, FK) | trip identification |
| station_id (PK, FK | station identification |

| Booking | |
|---|---|
| **Fields** | **Description** |
| Booking_id (PK) | Booking identification |
| Sub_id (FK) | Subscriber identification |
| Arrival_date | Date of arrival |
| Arrival_time | Time of arrival |
| Bike_id (FK) | Bike identification |
| Station_id (FK) | Bike station identification |

| Employee | |
|---|---|
| **Fields** | **Description** |
| Emp_id (PK) | Employee Number |
| Emp_Fname | Employee first name |
| Emp_Lname | Employee last name |
| DOB | Date of birth |
| Street_Address | Street name |
| City | city |
| State | state |
| Zip_code | Zip code |

| Station | |
|---|---|
| **Fields** | **Description** |
| station_id (PK) | Bike station identification |
| name | name of the station |
| status | If the station is closed or open |
| street_Address | street name |
| city | city |
| state | state |
| zip_code | zip code |

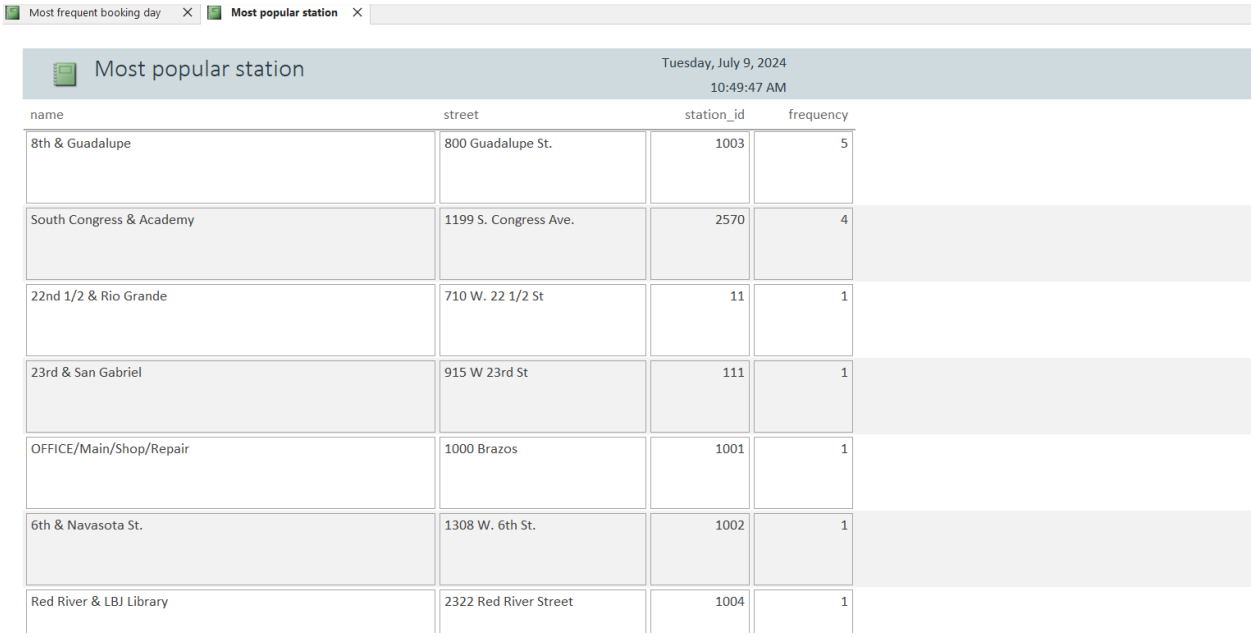
## Notes explaining design decisions and assumptions

**I also created two bridge entities named Trip_station and Employee_repair. To break the M:M relationship between the trip table to the station table, and the repair_order table to the employee table.**

# Part III. Implementation and Loading - Access

## Business Reports

1. This is to see which Station is our hottest ticket item.



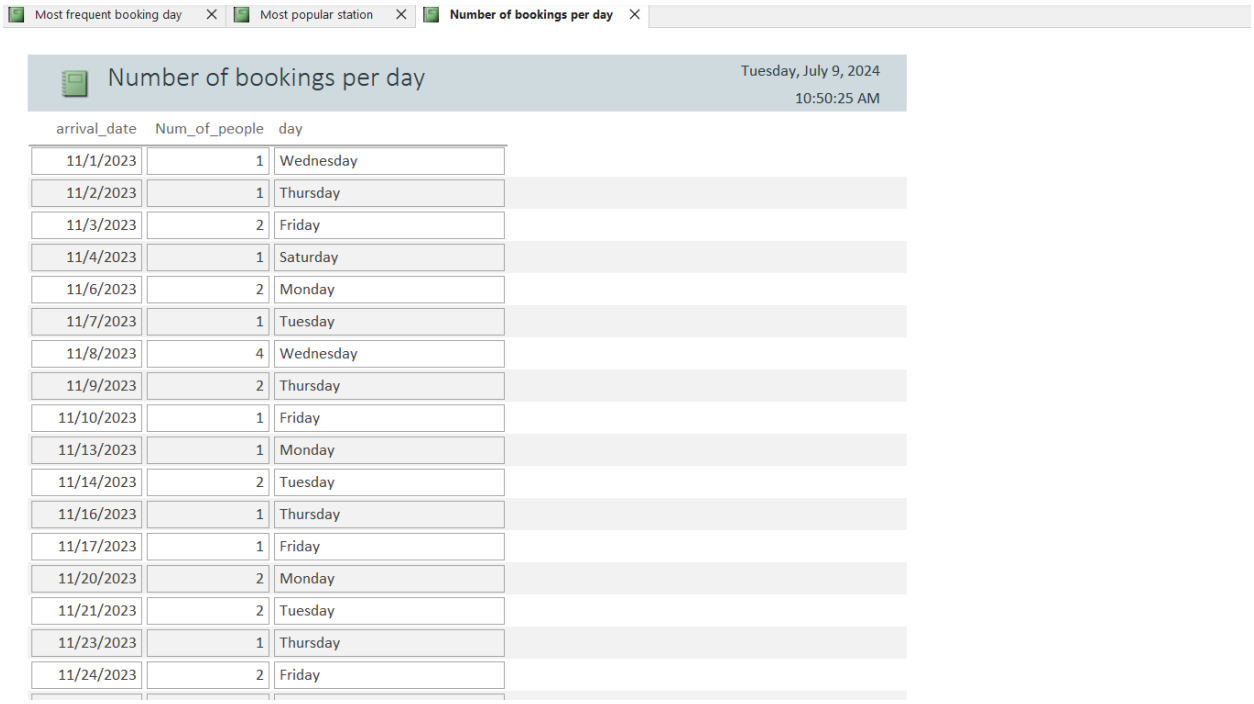

| name | street | station_id | frequency |
|---|---|---|---|
| 8th & Guadalupe | 800 Guadalupe St. | 1003 | 5 |
| South Congress & Academy | 1199 S. Congress Ave. | 2570 | 4 |
| 22nd 1/2 & Rio Grande | 710 W. 22 1/2 St | 11 | 1 |
| 23rd & San Gabriel | 915 W 23rd St | 111 | 1 |
| OFFICE/Main/Shop/Repair | 1000 Brazos | 1001 | 1 |
| 6th & Navasota St. | 1308 W. 6th St. | 1002 | 1 |
| Red River & LBJ Library | 2322 Red River Street | 1004 | 1 |

```
SELECT
        name,
        street,
        booking.station_id,
        COUNT(booking.station_id) AS frequency
FROM booking INNER JOIN station ON station.station_id = booking.station_id
```

GROUP BY booking.station_id, name, street

ORDER BY COUNT(booking.station_id) DESC;

2. This Report helps to see the trend of bookings by date.




| arrival_date | Num_of_people | day |
|---|---|---|
| 11/1/2023 | 1 | Wednesday |
| 11/2/2023 | 1 | Thursday |
| 11/3/2023 | 2 | Friday |
| 11/4/2023 | 1 | Saturday |
| 11/6/2023 | 2 | Monday |
| 11/7/2023 | 1 | Tuesday |
| 11/8/2023 | 4 | Wednesday |
| 11/9/2023 | 2 | Thursday |
| 11/10/2023 | 1 | Friday |
| 11/13/2023 | 1 | Monday |
| 11/14/2023 | 2 | Tuesday |
| 11/16/2023 | 1 | Thursday |
| 11/17/2023 | 1 | Friday |
| 11/20/2023 | 2 | Monday |
| 11/21/2023 | 2 | Tuesday |
| 11/23/2023 | 1 | Thursday |
| 11/24/2023 | 2 | Friday |

SELECT

arrival_date,

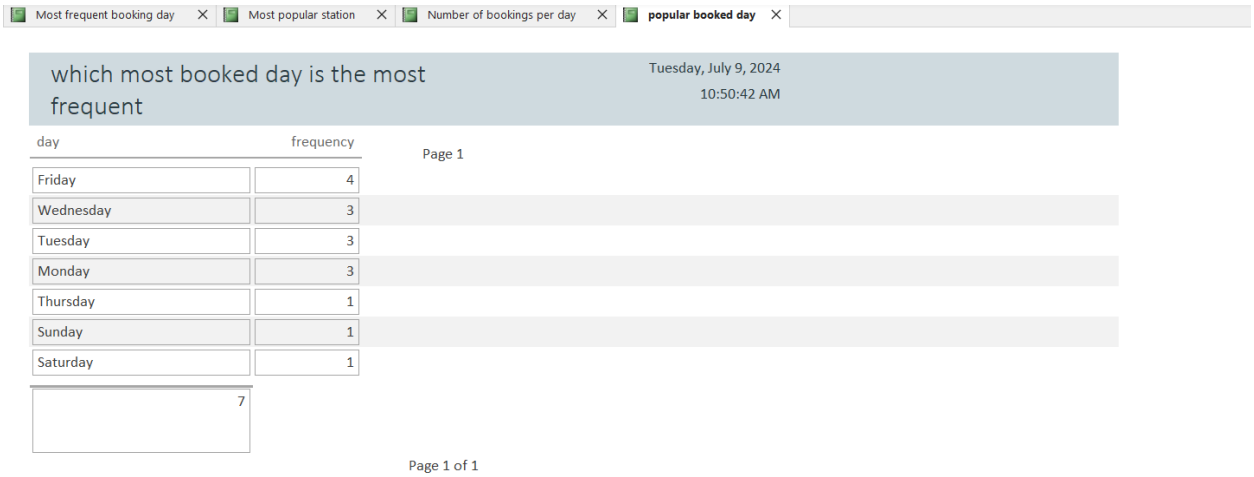
COUNT(arrival_date) AS Num_of_people,

FORMAT(arrival_date, 'dddd') AS [day]

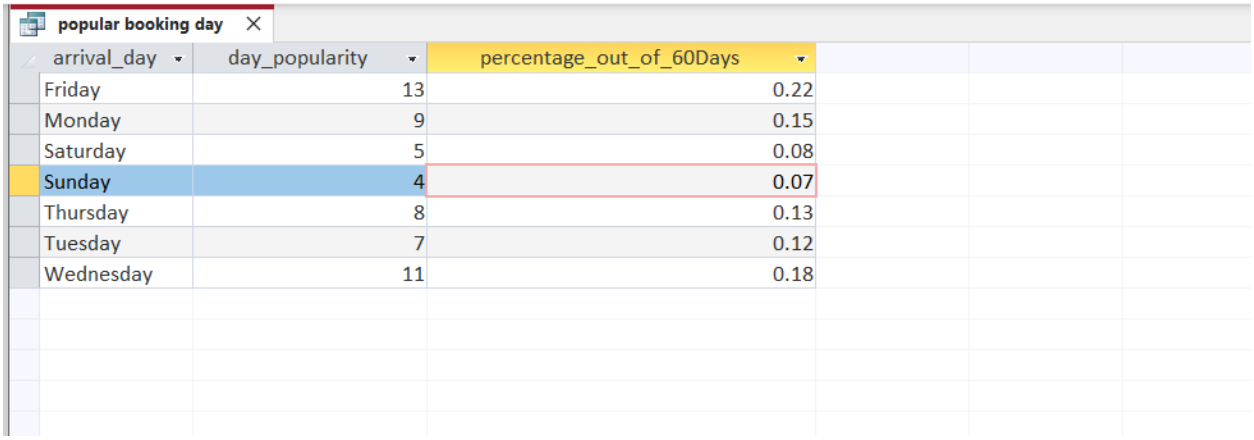
FROM booking

GROUP BY arrival_date;

3. Based on the bookings per day which day gets the most traffic.




4. This data spans two months. This helps us to see the percentage of popularity that builds more on the last report.




| arrival_day | day_popularity | percentage_out_of_60Days |
|---|---|---|
| Friday | 13 | 0.22 |
| Monday | 9 | 0.15 |
| Saturday | 5 | 0.08 |
| Sunday | 4 | 0.07 |
| Thursday | 8 | 0.13 |
| Tuesday | 7 | 0.12 |
| Wednesday | 11 | 0.18 |

SELECT

Format(arrival_date,'dddd') AS arrival_day,

Count(*) AS day_popularity,

Round((day_popularity/60),2) AS percentage_out_of_60Days

FROM booking

GROUP BY Format(arrival_date,'dddd');

# Conclusion

- I found that the Station on 8th and Guadelupe was the most popular
- Fridays are the most popular for people to book a bike.
- The order of popularity is Friday, Wednesday, Monday, Thursday, Tuesday, Saturday.