

# BikeFlow Analytics: Data Engineering Project

A comprehensive data warehouse solution for optimizing bike-sharing operations

By Layashree Adepu





# Project Objective

## Primary Goal

Analyze the relationship between bike availability and utilization patterns in popular urban areas

## Key Insights

Determine if popular locations have enough bikes to meet demand and identify underutilized areas

## Business Impact

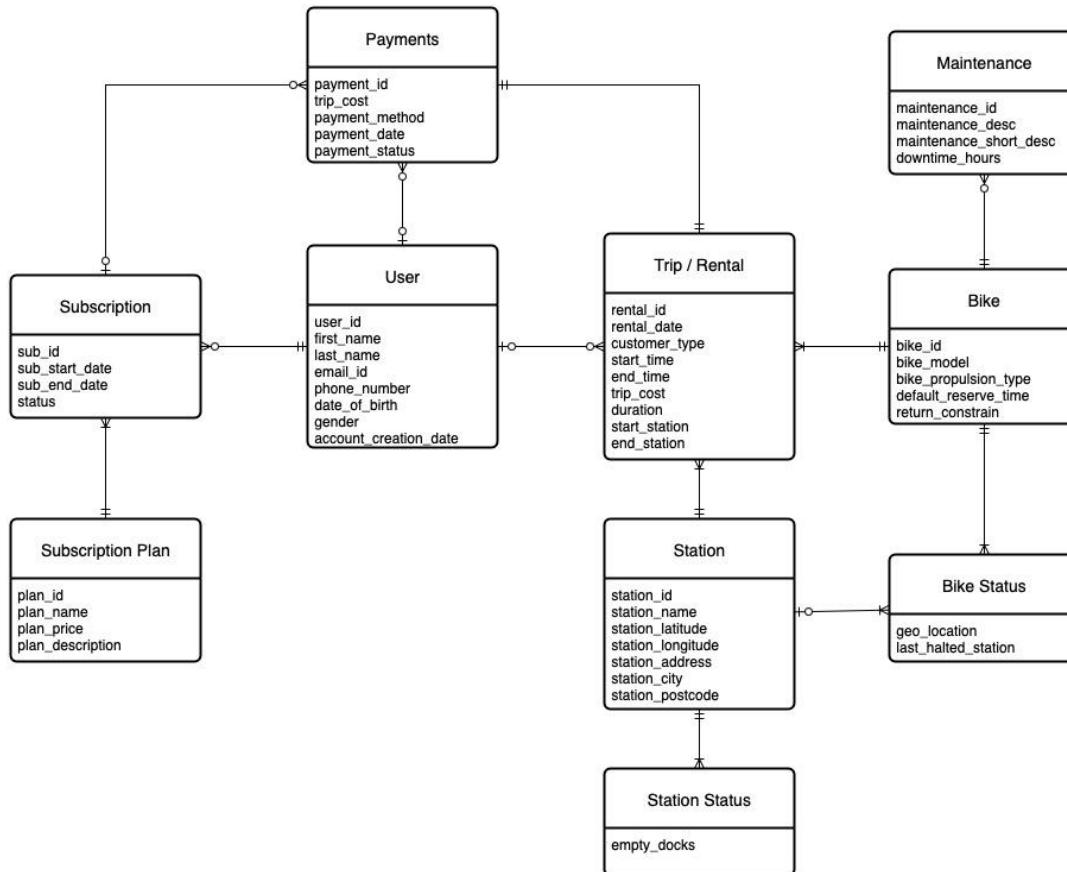
Optimize bike distribution strategies to maximize operational efficiency and user satisfaction

# Conceptual Diagram

wheels | lyft

BIKES

Overview Register/Sign In Map [Meet Our Bikes](#) Plans & pricing



# System Data

Here you'll find Bay Wheels's trip data for public use. So whether you're a designer, developer or just curious, bring it to life. This data is provided according to the [Bay Wheels License Agreement](#).

## The Data

Each trip is anonymized and includes:

- Trip Duration (seconds)
- Start Time and Date
- End Time and Date
- Start Station ID
- Start Station Name
- Start Station Latitude
- Start Station Longitude
- End Station ID
- End Station Name
- End Station Latitude
- End Station Longitude
- Ride ID
- User Type (Subscriber or Customer – “Subscriber” = Member or “Customer” = Casual)

[Download Bay Wheels trip history data](#)

## Real-Time Data

Bay Wheels publishes real-time system data in General Bikeshare Feed Specification form.

# Design Implementation

## Steps

### 01 Conceptual Diagram

The goal is to create a high-level representation focuses on identifying key entities, their attributes, and the relationships between them, independent of any technology

### 0 Logical Diagram

It involves mapping the entities and relationships to logical constructs like tables and columns, defining primary and foreign keys, and incorporating data types.

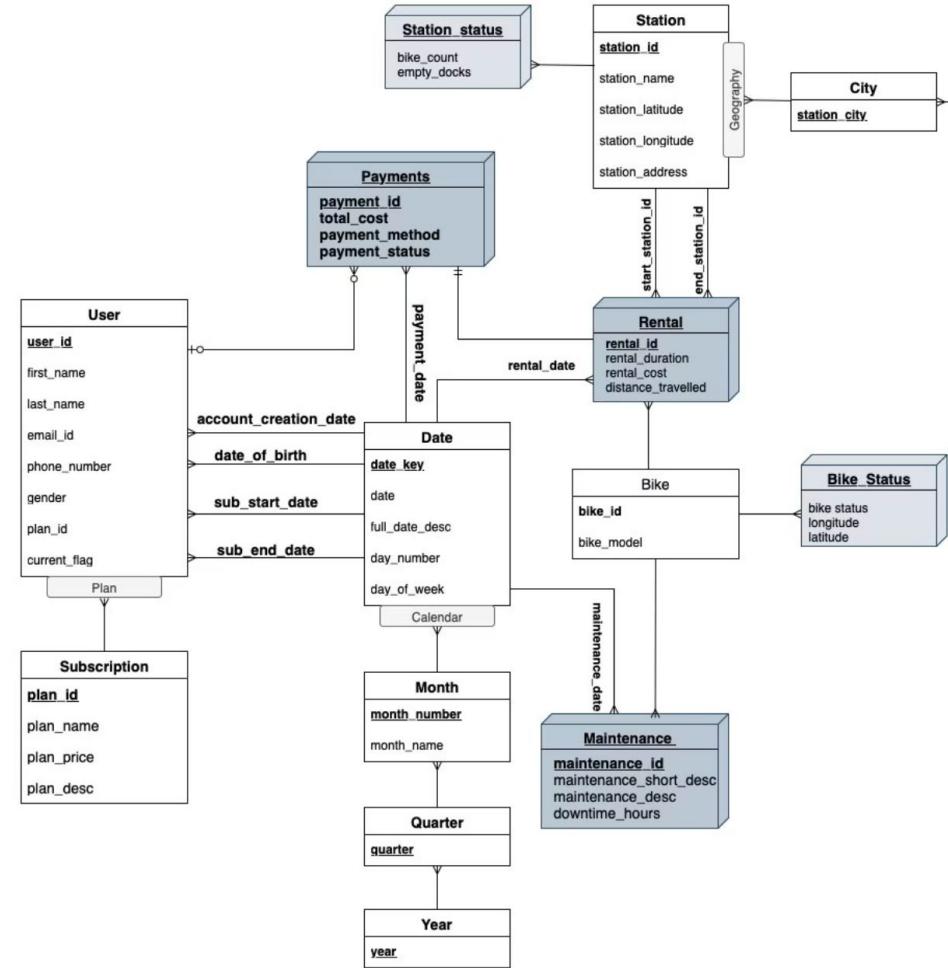
### 0 MultiDim Conceptual Diagram

It involves identifying key business metrics (facts) and the dimensions (e.g., time, location, user, bike) by which these metrics can be analyzed.

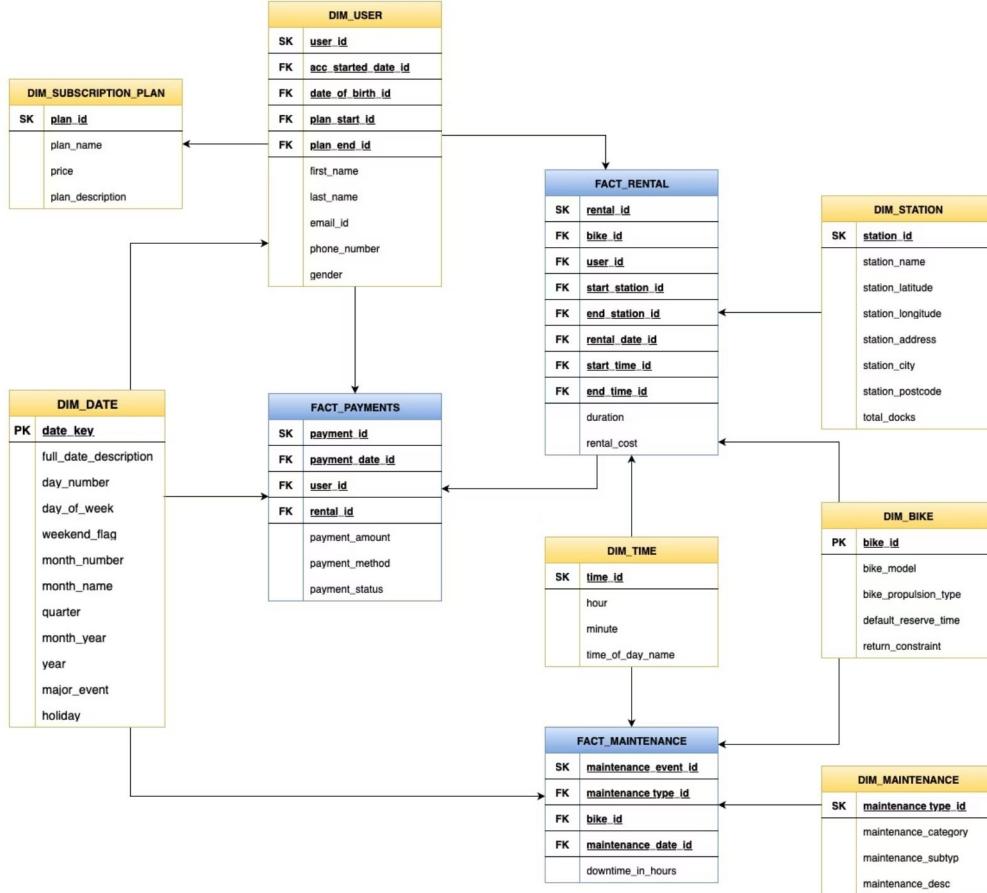
The goal is to create a star or snowflake schema that optimizes data retrieval.

### 0 MultiDim Logical Diagram

This includes defining the exact structure of fact and dimension tables, specifying aggregation rules, and planning for data transformations.



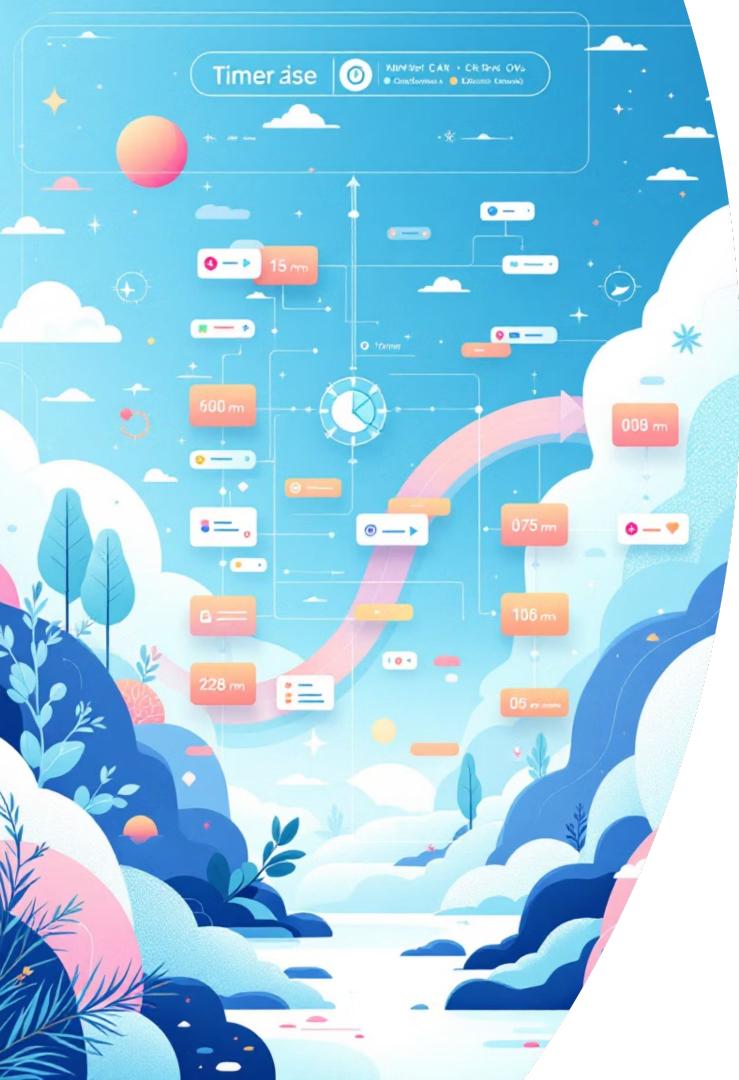
# Data Warehouse Model



## Star Schema Design

Our dimensional model features a central fact table surrounded by dimension tables for optimal query performance and business intelligence reporting.

- Fact table: Trip and availability metrics
- Time dimensions: Date and time hierarchies
- Location dimensions: Station and geographic data
- Bike dimensions: Vehicle characteristics



# Time Dimension Implementation

## DIM\_TIME Structure

Granular time tracking with minute-level precision, including time-of-day categorization for pattern analysis

- Hour and minute extraction
- Time period classification
- Peak hour identification

## DIM\_DATE Features

Comprehensive date dimension spanning decades with business calendar support and holiday tracking

- Weekend and holiday flags
- Quarter and month hierarchies
- Custom business periods

DIM_TIME	DIM_DATE
<pre>INSERT INTO DIM_TIME (     time_of_day,     hour,     minute,     time_of_day_name ) SELECT     (t::timestamp)::time AS time_of_day,     EXTRACT(HOUR FROM t) AS hour,     EXTRACT(MINUTE FROM t) AS minute,     CASE         WHEN EXTRACT(HOUR FROM t) BETWEEN 5 AND 11 THEN             'Morning'         WHEN EXTRACT(HOUR FROM t) BETWEEN 12 AND 16 THEN             'Afternoon'         WHEN EXTRACT(HOUR FROM t) BETWEEN 17 AND 20 THEN             'Evening'         ELSE 'Night'     END AS time_of_day_name FROM     GENERATE_SERIES('2025-01-01 00:00:00'::timestamp,     '2025-01-01 23:59:00'::timestamp, '1 minute'::interval) AS t;</pre>	<pre>INSERT INTO DIM_DATE ( date_key, full_date, day_number, day_of_week, weekend_flag, month_number, month_name, quarter, month_year, year, holiday ) SELECT     CAST(TO_CHAR(date_series, 'MMDDYYYY') AS INT) AS date_key,     date_series AS full_date,     EXTRACT(DAY FROM date_series) AS day_number,     TRIM(TO_CHAR(date_series, 'Day')) AS day_of_week,     CASE         WHEN EXTRACT(DOW FROM date_series) IN (0, 6) THEN             TRUE         ELSE FALSE     END AS weekend_flag,     EXTRACT(MONTH FROM date_series) AS month_number,     TRIM(TO_CHAR(date_series, 'Month')) AS month_name,     'Q'    EXTRACT(QUARTER FROM date_series) AS quarter,     TO_CHAR(date_series, 'YYYY-MM') AS month_year,     EXTRACT(YEAR FROM date_series) AS year,     CASE         WHEN date_series = '2025-01-01' THEN 'New Year's Day'         WHEN date_series = '2025-12-25' THEN 'Christmas Day'     END AS holiday;</pre>

# Other Dimensions

## Implementation

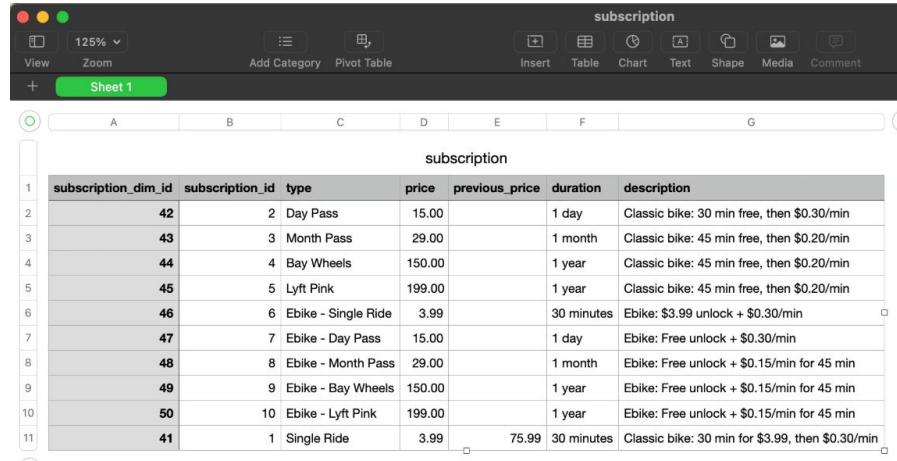
### Compare your options

They're all good ones.

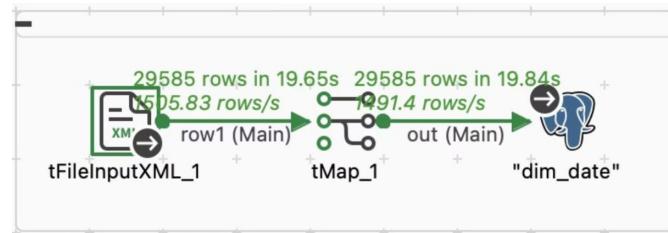
	Single ride	Day pass	Month pass	Bay Wheels	Lyft Pink
	\$3.99	\$15/day	\$29/month	\$150/year	\$199/year
	Get the app →	Get a day pass →	Get a month pass →	Join →	Join →
Classic bike prices	30 min for \$3.99, then \$0.30/min	30 min free, then \$0.30/min	45 min free, then \$0.20/min	45 min free, then \$0.20/min	45 min free, then \$0.20/min
Ebike prices	\$3.99 unlock + \$0.30/min	Free unlocks + \$0.30/min	Free unlocks + \$0.15/min for 45 min	Free unlocks + \$0.15/min for 45 min	Free unlocks + \$0.15/min for 45 min

Pricing and operational data sourced from Lyft Bay Wheels API

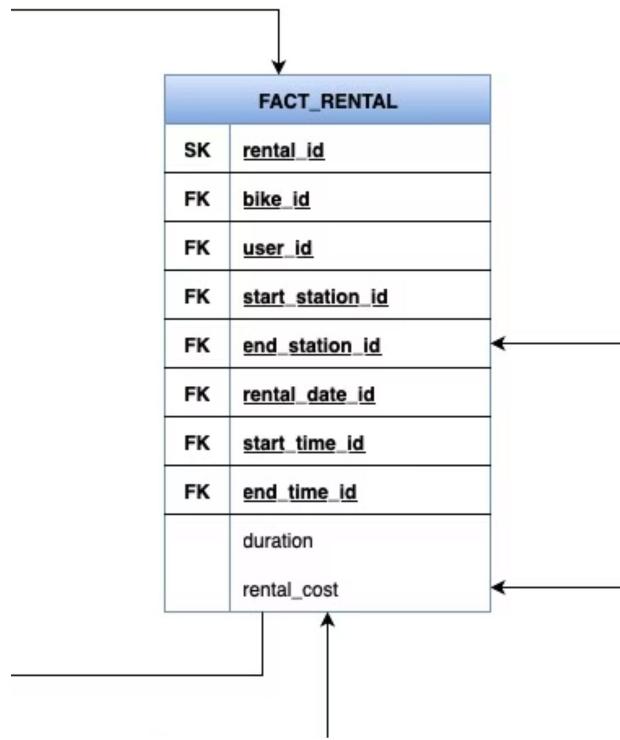
<https://www.lyft.com/bikes/bay-wheels/pricing>



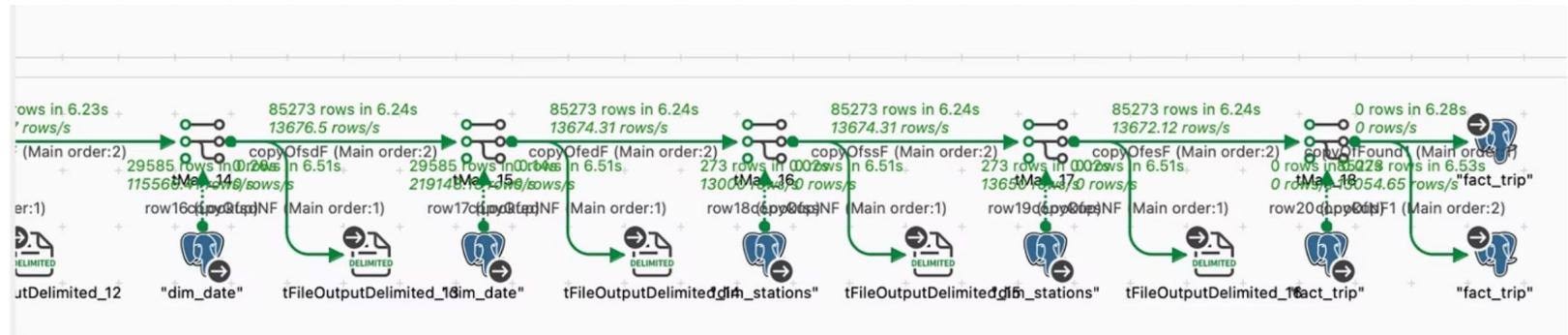
subscription						
	subscription_dim_id	subscription_id	type	price	previous_price	duration
1	42	2	Day Pass	15.00		1 day
2	43	3	Month Pass	29.00		1 month
3	44	4	Bay Wheels	150.00		1 year
4	45	5	Lyft Pink	199.00		1 year
5	46	6	Ebike - Single Ride	3.99		30 minutes
6	47	7	Ebike - Day Pass	15.00		1 day
7	48	8	Ebike - Month Pass	29.00		1 month
8	49	9	Ebike - Bay Wheels	150.00		1 year
9	50	10	Ebike - Lyft Pink	199.00		1 year
10	41	1	Single Ride	3.99	75.99	30 minutes
11						Classic bike: 30 min for \$3.99, then \$0.30/min



# FACT TABLE Focus



# FACT TABLE Implementation



**Expression Builder**

Expression

Wrap   Undo(Ctrl + Z)   Clear

```
out1.subscription_plan_sk == 1 ? (out1.duration_min > 60 ?
(out1.duration_min - 60) * 0.30 : 0.00) :

out1.subscription_plan_sk == 2 ? (out1.duration_min > 1440 ?
(out1.duration_min - 1440) * 0.30 : 0.00) :

(out1.subscription_plan_sk == 3 || out1.subscription_plan_sk == 4 ||
out1.subscription_plan_sk == 5) ? (out1.duration_min > 45 ?
(out1.duration_min - 45) * 0.20 : 0.00) :

out1.subscription_plan_sk == 6 ? 3.99 + (out1.duration_min *
0.30) :

out1.subscription_plan_sk == 7 ? out1.duration_min * 0.30 :

(out1.subscription_plan_sk == 8 || out1.subscription_plan_sk == 9 ||
out1.subscription_plan_sk == 10) ? (out1.duration_min *
0.15) : 0.00
```

**Test**

Var   Value

Var	Value
out1.end_time	null
out1.start_station_null	
out1.start_station_null	
out1.start_station_null	
out1.start_station_null	
out1.end_station_i	null
out1.bike_id	null
out1.user_type	null
out1.subscription_	null
row3.user_sk	null
row3.user_id	null
row3.first_name	null
row3.last_name	null

**Expression Builder**

out1

Expression	Column
row4.duration_sec	duration_min

Expression

Wrap   Undo(Ctrl + Z)   Clear

```
row4.duration_sec / 60
```

# SCD TYPE 3 implementation

Talend Studio

Feature Manager

LOCAL: BikeFlow

- > dim\_bike\_insurance\_status
- > dim\_bike\_insurance\_status\_scd
- > dim\_bike\_status
- > dim\_city
- > dim\_date
- > dim\_maintenance
- > dim\_payment
- > dim\_state
- > dim\_stations
- > dim\_subscription
- > dim\_subscription\_scd
- > dim\_ticket
- > dim\_time
- > dim\_trip
- > dim\_user\_subscription
- > fact\_trip
- > fact\_users
- > View schemas

OLTP 0.1

- > Queries
- > Synonym schemas

Table schemas

- > bike
- > dim\_bike\_insurance\_status
- > dim\_maintenance
- > dim\_payment
- > dim\_stations
- > dim\_subscription
- > dim\_ticket
- > dim\_trip
- > dim\_user\_subscription

Code Viewer

Job SCD\_Price ended at 22:58 11/11/2024. [Exit code = 0]

SCD component editor

Type 0 fields

Unused

description  
duration  
type

Type 1 fields

Source keys

subscription\_id

Type 2 fields

Surrogate keys

name subscription\_dim\_id  
creation DB Sequence  
complement SCDsub\_seq

Versioning

name start end creation complement

type scd\_start scd\_end Job start time scd\_complement

type 3 fields

current value previous value

price previous\_price

Cancel OK

Palette

Favorites

Recently Used

- 1DBSCD(Post...)
- 1DBInput(Post...)
- 1DBOutput
- iRunJob
- iFileOutputDe...
- iMap
- iLogRow
- iDBComm
- iDBConnection
- iFileInputDeli...
- iFileInputXML
- AggregateRow

Big Data

- Google BigQuery
- Google Storage
- Hive

Business Intelligence

- Charts
- DB SCD
- Jasper

Business Cloud

- Amazon
- Azure
- Box

subscription_id	type	price	previous_price
2	Day Pass	15.00	[null]
3	Month Pass	29.00	[null]
4	Bay Wheels	150.00	[null]
5	Lyft Pink	199.00	[null]
6	Ebike - Single Ride	3.99	[null]
7	Ebike - Day Pass	15.00	[null]
8	Ebike - Month Pass	29.00	[null]
9	Ebike - Bay Wheels	150.00	[null]
10	Ebike - Lyft Pink	199.00	[null]
1	Single Ride	5.99	5.00

# Bike Flow Analytics

1739

No of active users

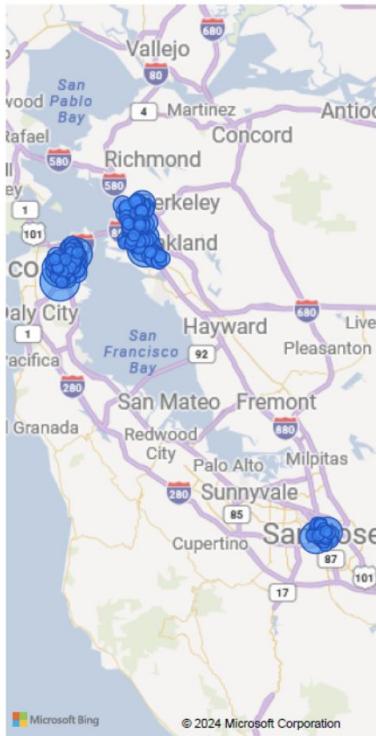
73.55K

Total amount earned till date

85.27K

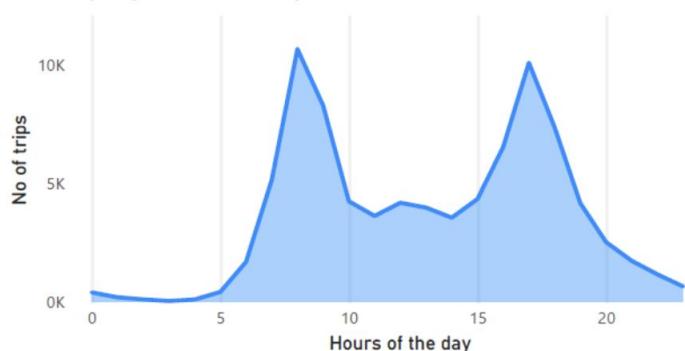
Num of trips till date

## Most popular stations

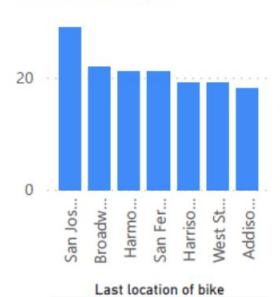


Stations
West St at 40th St
West Oakland BART Station
Webster St at O'Farrell St
Webster St at Grove St
Webster St at 19th St
Washington St at Kearny St
W St John St at Guadalupe River Trail
Valencia St at 24th St
Valencia St at 22nd St
Valencia St at 16th St
University Ave at Oxford St
Union St at 10th St

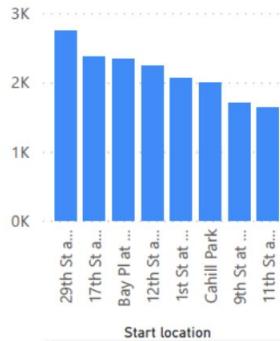
## Total Trips by Hour of the Day



## No of bikes under maintenance by Last location of bike



## No of trips by Start location



## Sum of user\_id by gender



## Sum of amount by type

