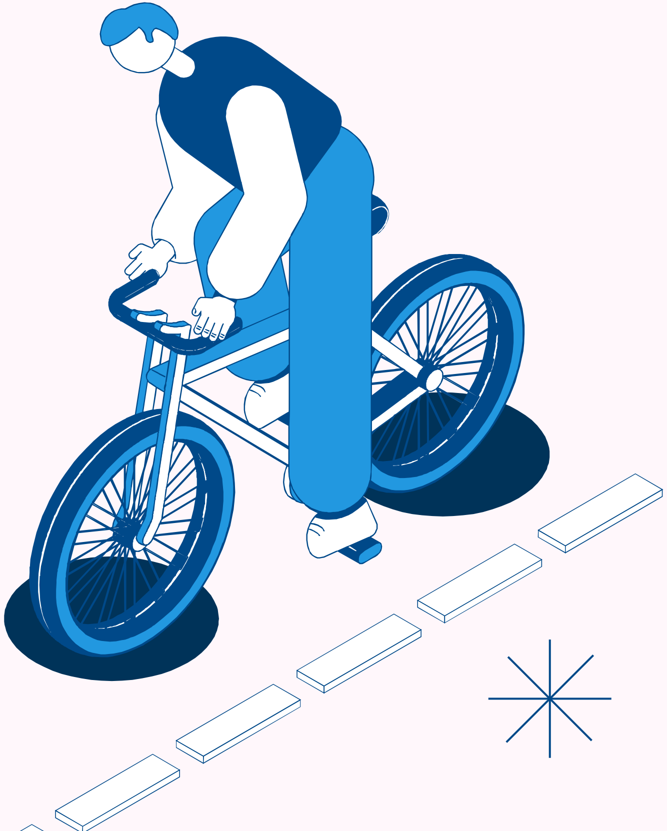
Bicycle Theft Analysis

Presented by Group-6
Arkesh, Eman, Harpreet, Garv, Rajanbir, Sukhsimar, Sarthak



INTRODUCTION

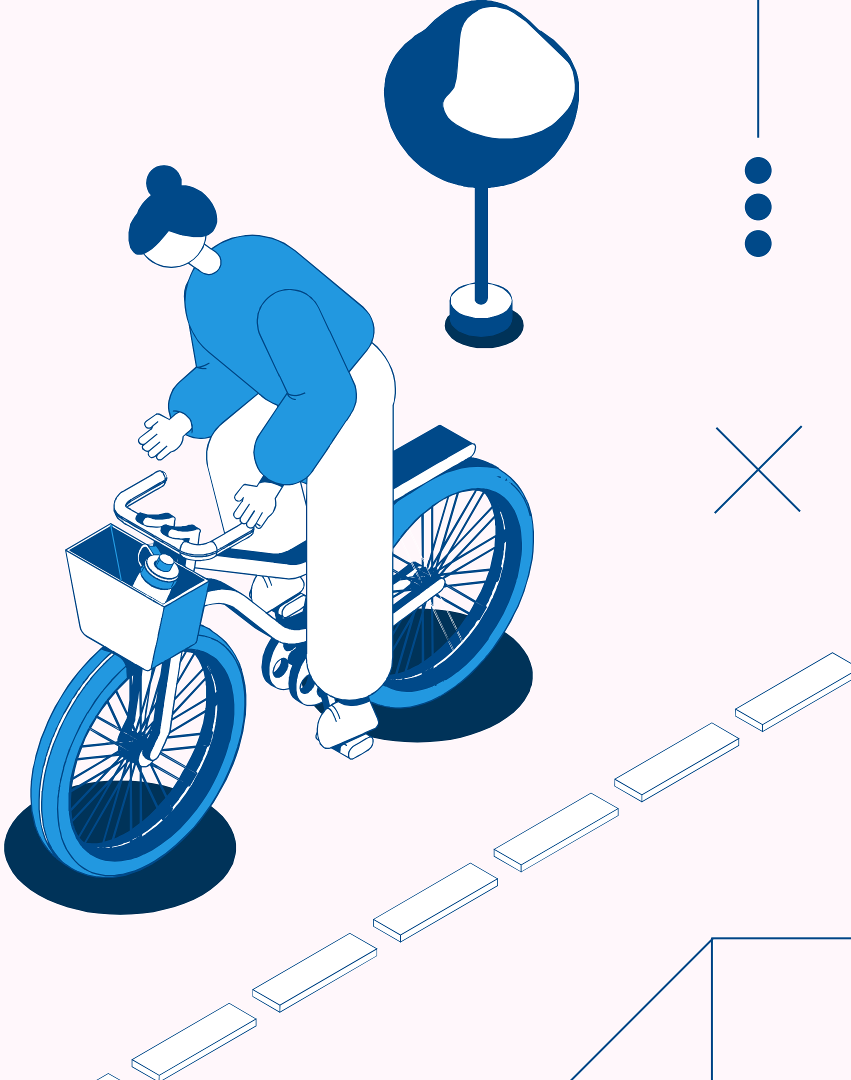
According to data from the Toronto Police Service, there were 3,474 reported cases of bicycle theft in 2019, 3,102 in 2020, and 3,312 in 2021. As of March 2023, there have been 637 reported cases of bicycle theft in Toronto.

Table of Contents

01

Data Collection

Collecting data from various sources



Datasets

Bicycle Theft Dataset

Contains info about neighbourhood, premises type, location (lat/long), occurrence time (based on reports), bike information (color, make)

Bicycle Parking Dataset

Shapefile of permanent and seasonal multiple-capacity bicycle parking racks installed and managed by the Cycling Infrastructure and Programs Unit

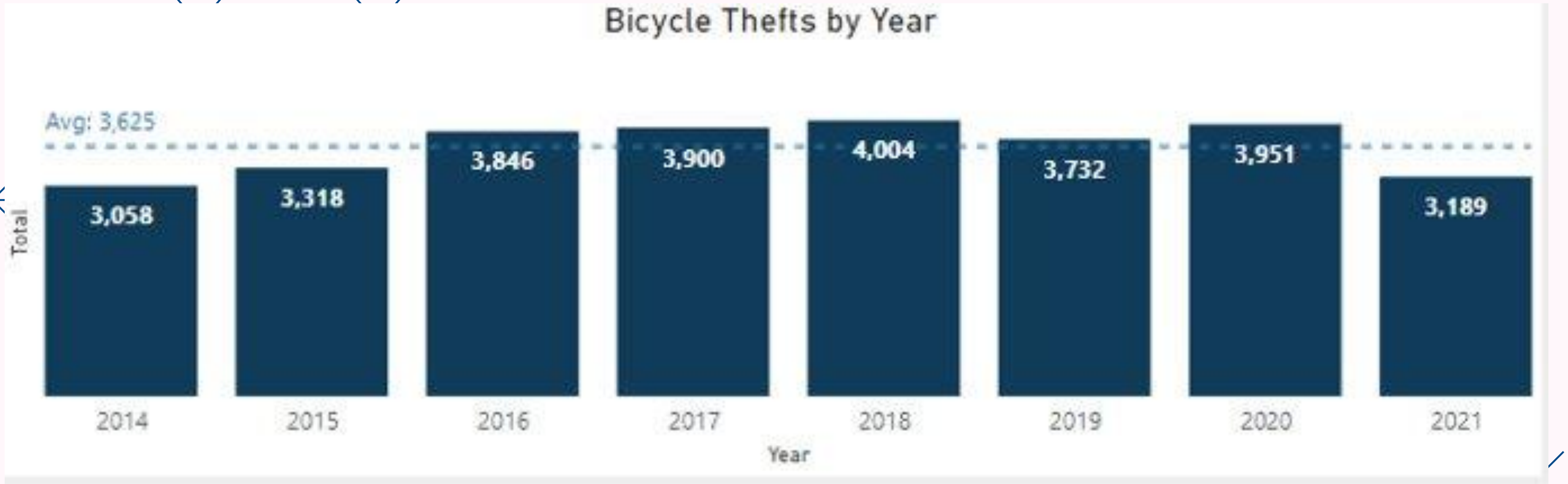
Weather Dataset

Obtained from Dashboard which provide updated information of Toronto weather

Data Collection

- Bicycle theft data was collected from Toronto open data resource along with which we found relating data to bicycle indoor and outdoor stands. Data was in CSV format.

- Weather data was collected from https://weather.visualcrossing.com as csv format

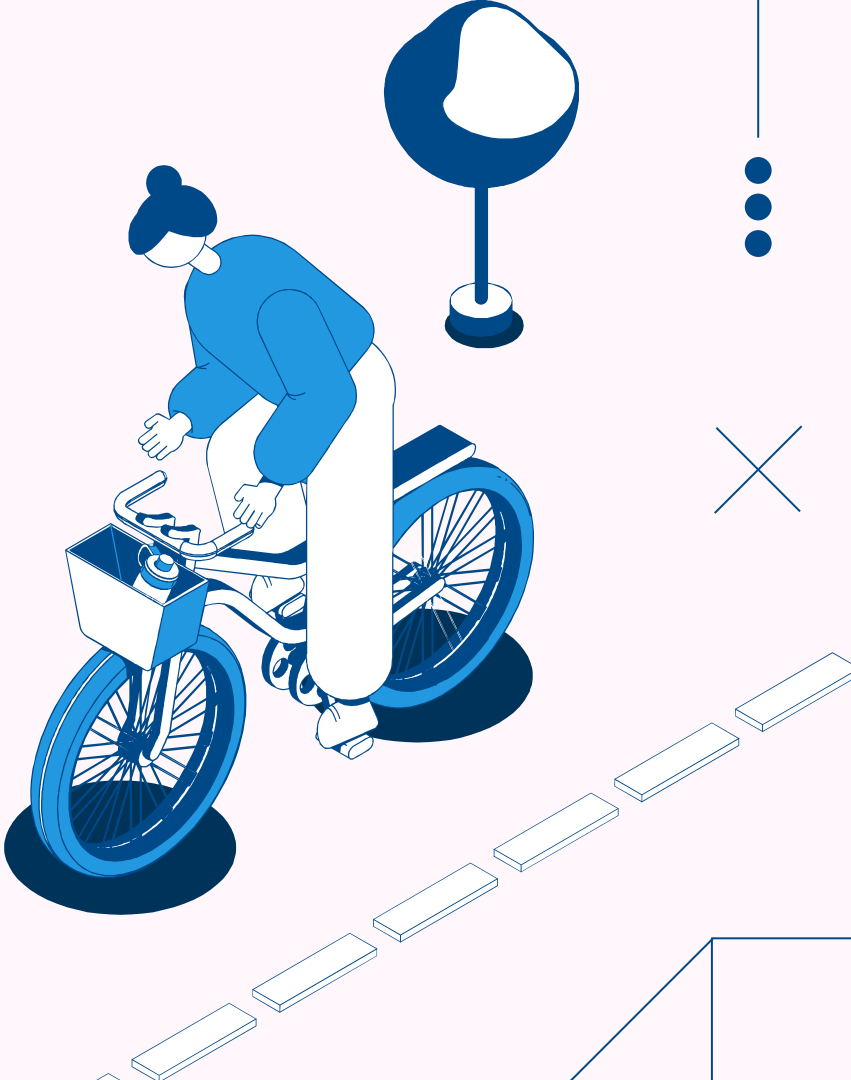


On average **3,625 bicycles** were stolen over a period of 7 years. The maximum(4,004) being in year 2018 and minimum(3,058) being in 2014

02

Data Exploration

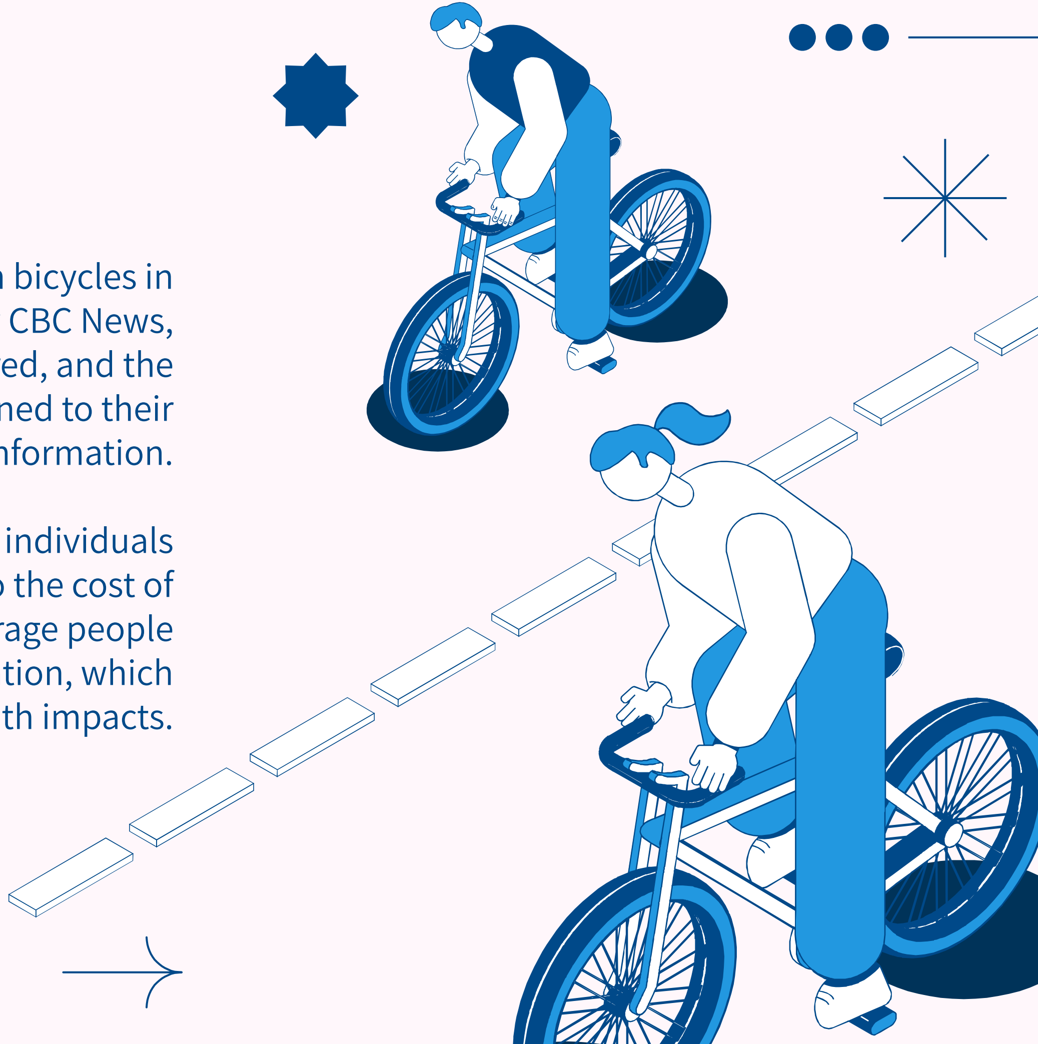
Identifying gaps and defining problems



Impact of Thefts

Unfortunately, the recovery rates for stolen bicycles in Toronto are low. According to a 2019 report by CBC News, only 10-15% of stolen bikes are recovered, and the majority of recovered bikes are not returned to their owners due to a lack of identifying information.

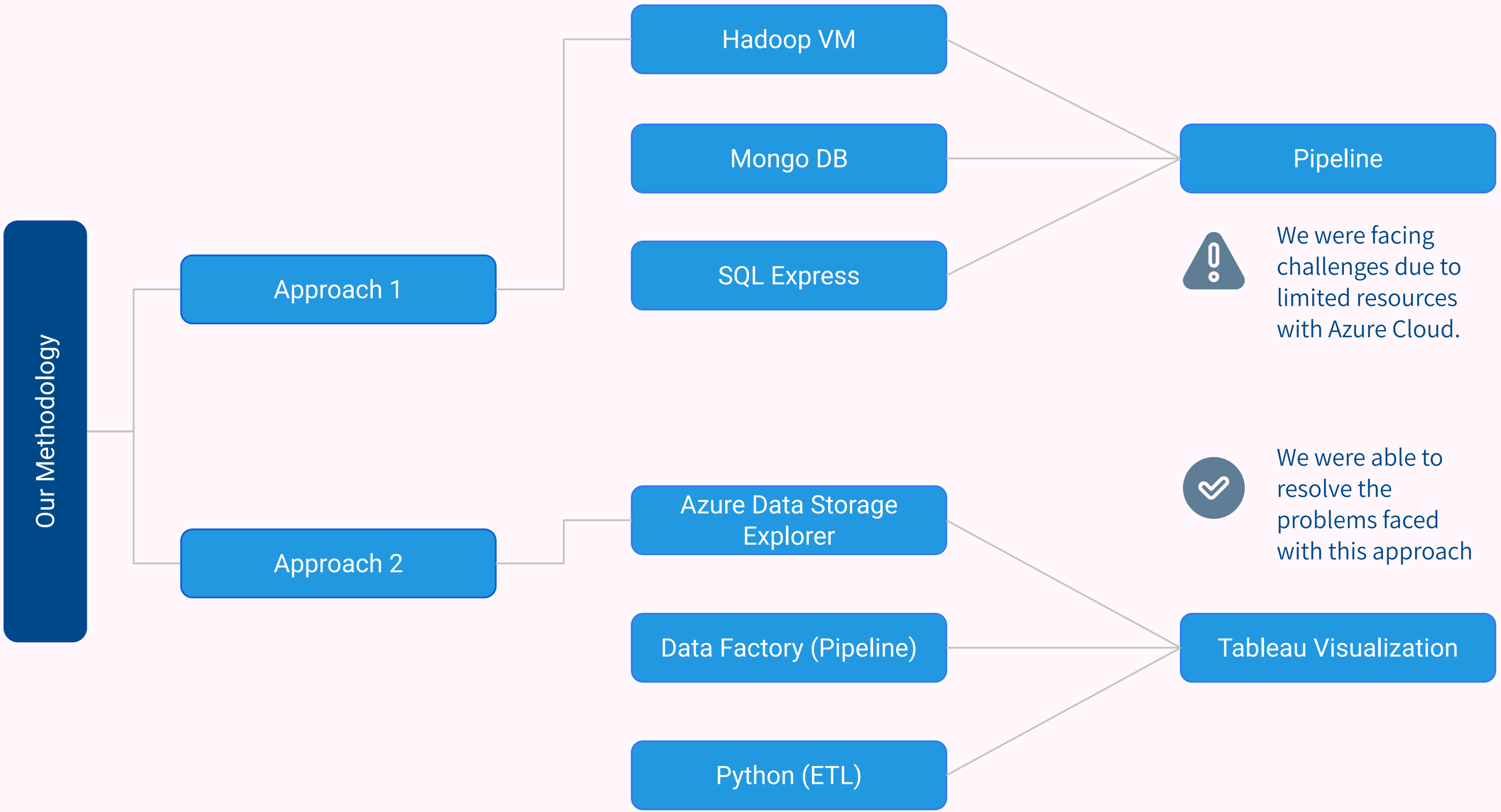
Bicycle theft can have a significant impact on individuals and the community as a whole. In addition to the cost of replacing a stolen bike, theft can also discourage people from using bicycles as a mode of transportation, which has negative environmental and health impacts.



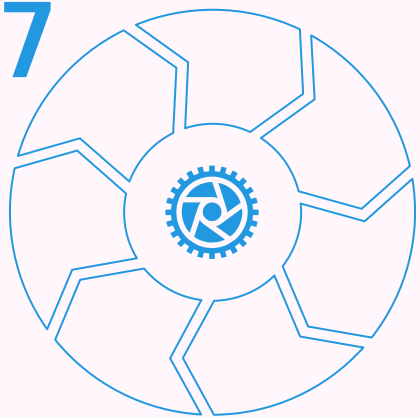
03

Data Transformation

Converting data to a standard workable format


Our Methodology
Approach 1
Approach 2
Hadoop VM
Mongo DB
SQL Express
Azure Data Storage Explorer
Data Factory (Pipeline)
Python (ETL)
Pipeline
We were facing challenges due to limited resources with Azure Cloud.
We were able to resolve the problems faced with this approach
Tableau Visualization

Sequential Process of Development



№1 Data sources
3 data sources

№2 Tables
3 tables with 3 CSV files

№3 SQL tables
Using datafactory, uploaded them into SQL tables

№4 SQL to Python
Read all the SQL tables into Python

№5 ETL
Cleaning, Joining, ETL

№6 Final Table
Created the final table

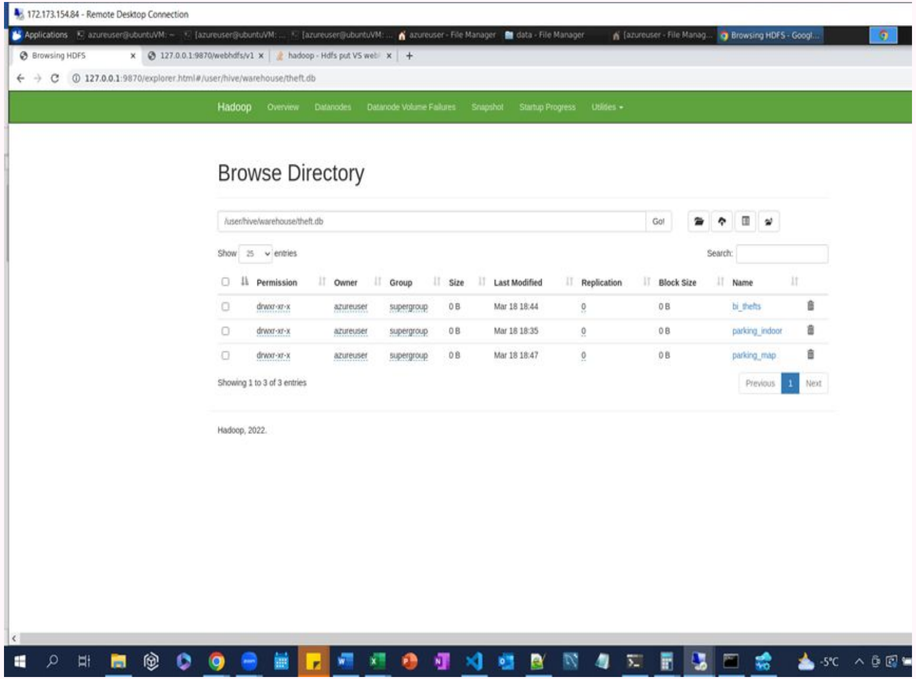
№7 Visuals
Put that final table into Tableau, then created visualizations

Approach 1

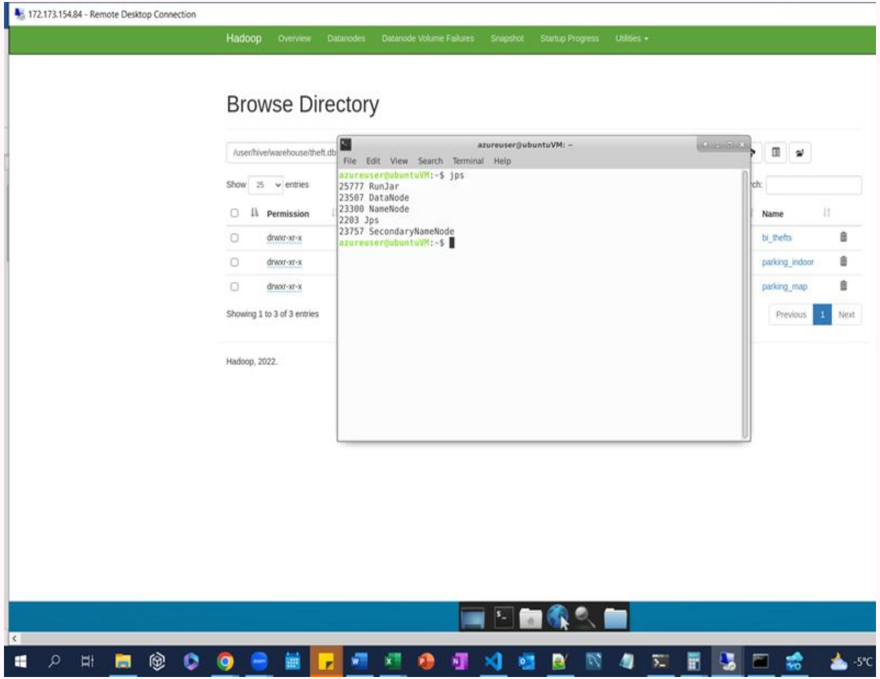
- **Azure :** Installed Hadoop on the Ubuntu VM and uploaded the bicycle theft data.
- **Windows VM:** Installed all the SQL tools (including SQL Express, SSMS, SSIS) , Visual Studio and MongoDB. Uploaded the weather data to MongoDB.
- **In SSIS in Windows VM**, created pipeline where connections were established with Hadoop as well as Mongo using the Connection Manager. 1 Fact and 2 Dimension tables were created for warehousing and passing the data to SQL Database in Azure.

However, due to having a student subscription with Azure and limited resources, we faced issues with this approach and tried a slightly different one with some different tools.

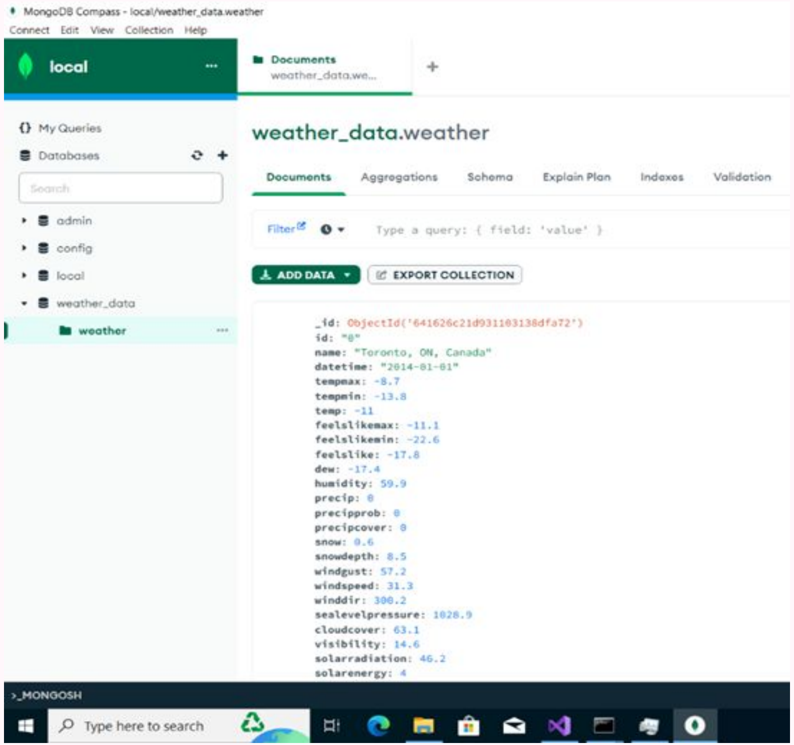
Hadoop VM



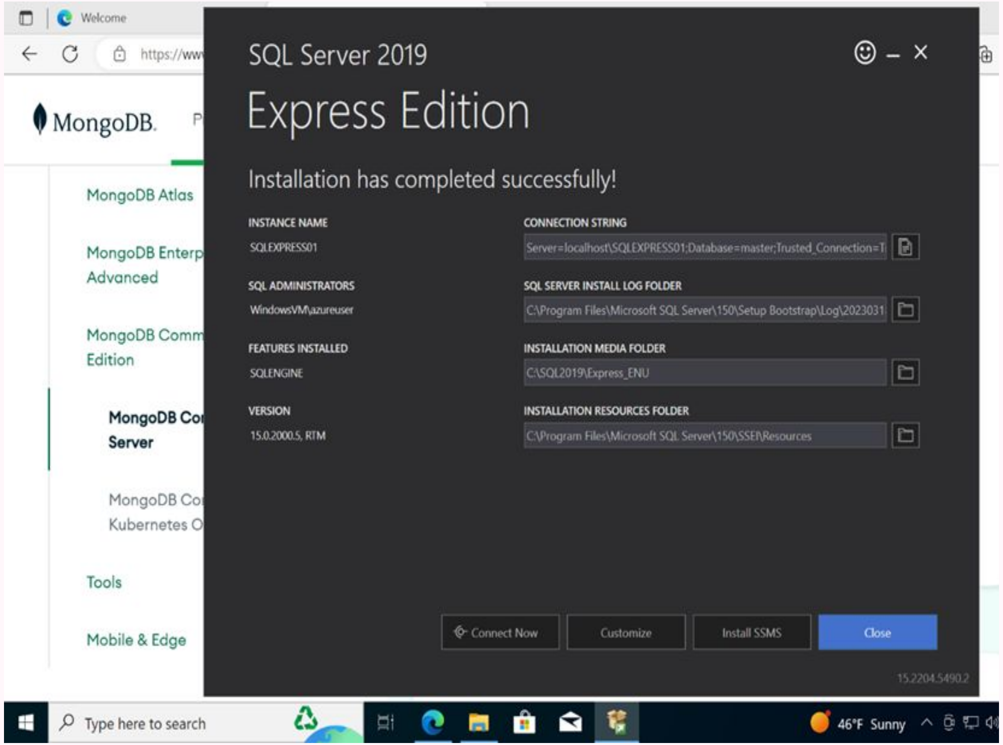
Hadoop VM : jps command



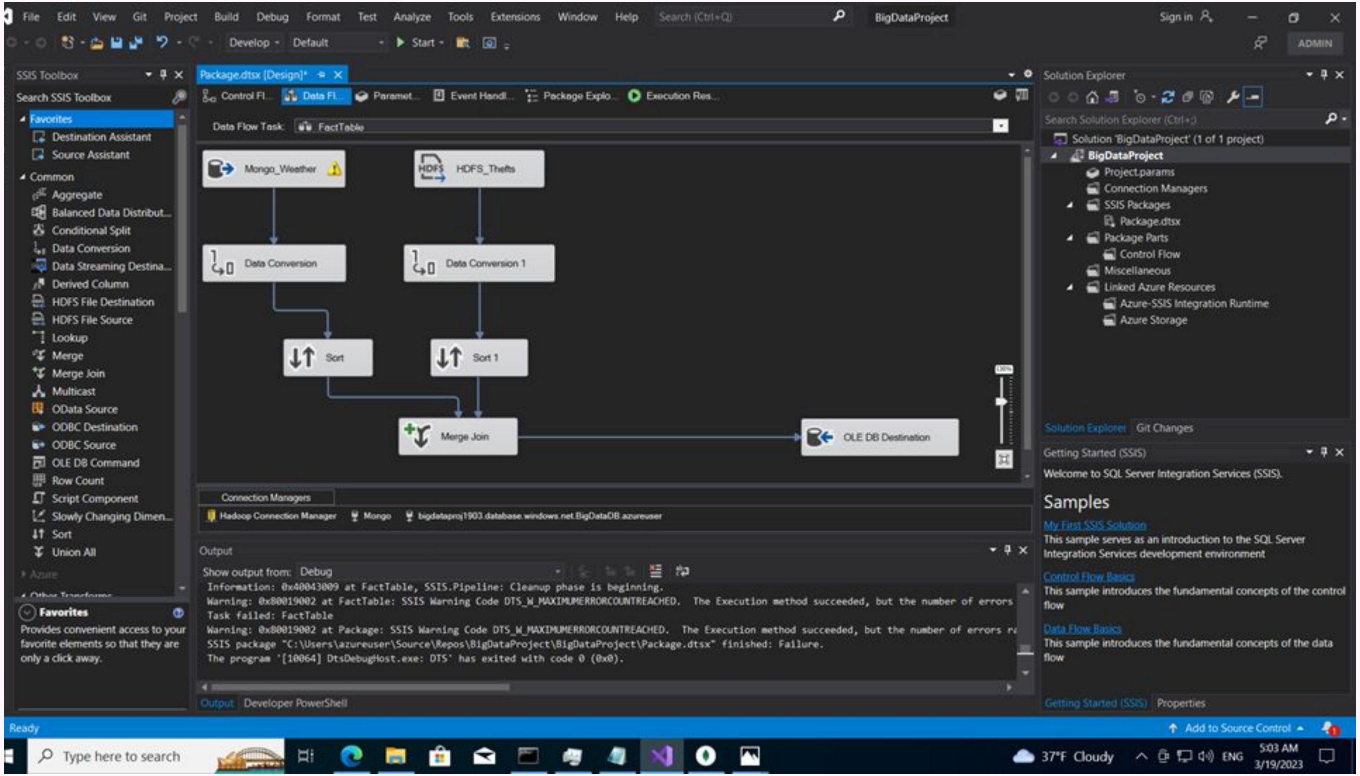
Windows VM: MongoDB



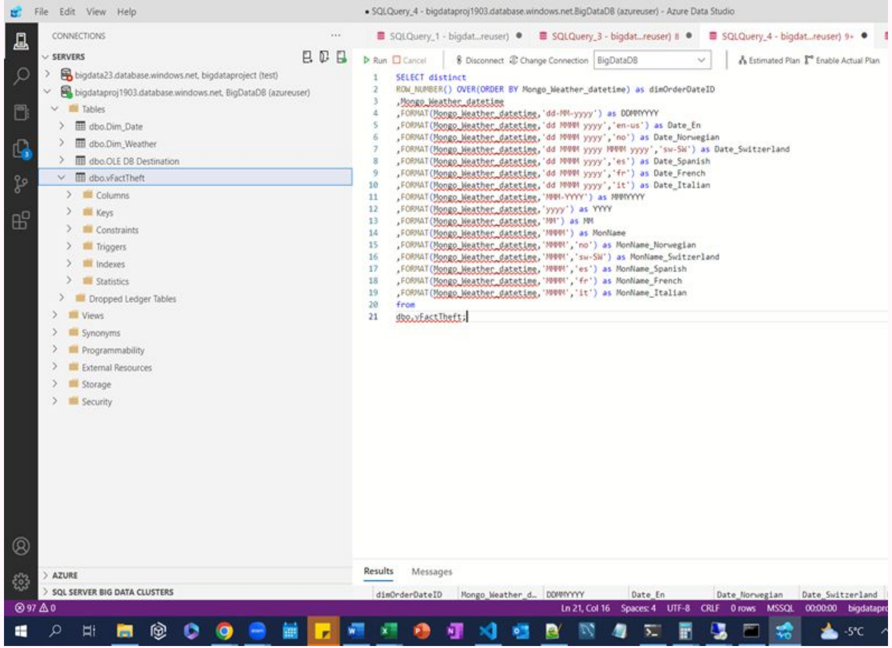
Windows VM : SQL Express



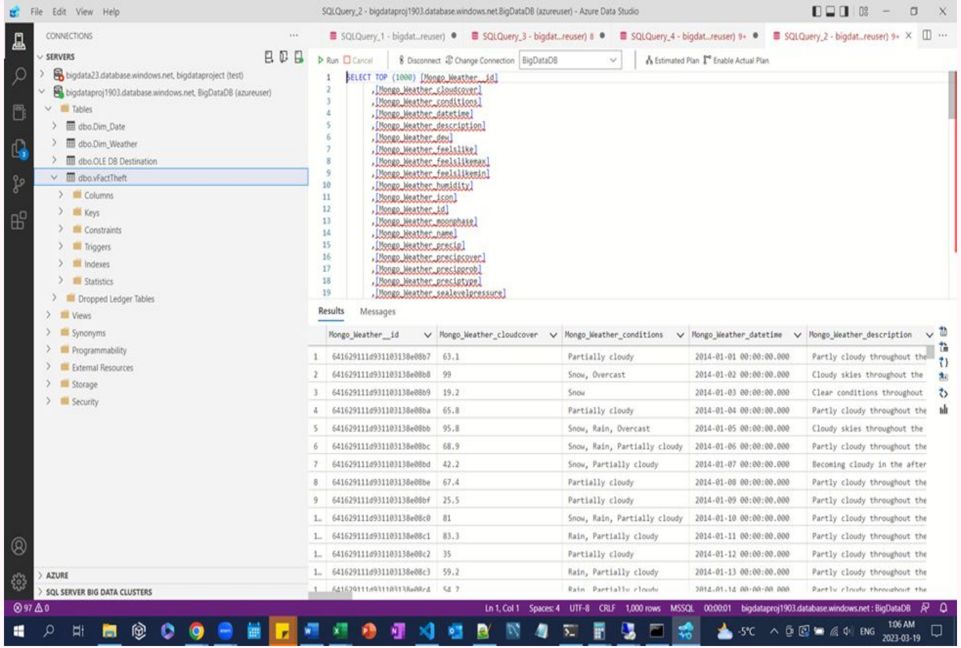
Pipeline (1 of 3)



Dim Table (1 of 2)



Fact Table



Approach 2

- **Azure Data Storage:** Storing the bicycle theft data and parking location data collected on cloud blob storage.
- **Azure Data Factory:** Creating pipelines to connect and push data from storage to SQL Server
- **SQL Server:** After pipeline runs successfully data should reflect in SQL server
- **Python(ETL):** Using python ETL is done on different tables and prepared for data analysis and visualization.
- **Visualizations using Tableau,** we imported the final table from Python to Tableau to visualize the data.

However, due to having a student subscription with Azure and limited resources, we faced issues with this approach and tried a slightly different one with some different tools.

Loading Data

We have erected tables within the Data Factory utilizing pipelines sourced from the CSVs of the datasets and leveraged the pandas library in Python to extract data from Data Factory pipeline.

```
def getConnection():
    global globalConn
    if globalConn != None:
        return globalConn

    str = "Driver={ODBC Driver 18 for SQL Server};Server=tcp:bigdata23.database.windows.net,1433;Database=bigdataproject;Uid=test;Pwd=password@123;Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;"
    params = urllib.parse.quote_plus(str)
    conn_str = 'mssql+pyodbc:///?odbc_connect={}'.format(params)
    globalConn = create_engine(conn_str,use_insertmanyvalues=True,)
    return globalConn

def dropConnection():
    global globalConn
    if globalConn != None:
        globalConn.dispose()
    globalConn = None
```

- Connecting weather data csv using SQL Server and indexing it on ID

```
def loadWeatherData():
    ctx = getConnection()
    df = pd.read_csv('./weather/all.csv',index_col='id')
    df.to_sql('weather', con=ctx.connect(), if_exists='replace')
    dropConnection()
```

- Connecting with SQL Server to import bicycle theft data and changing date format.

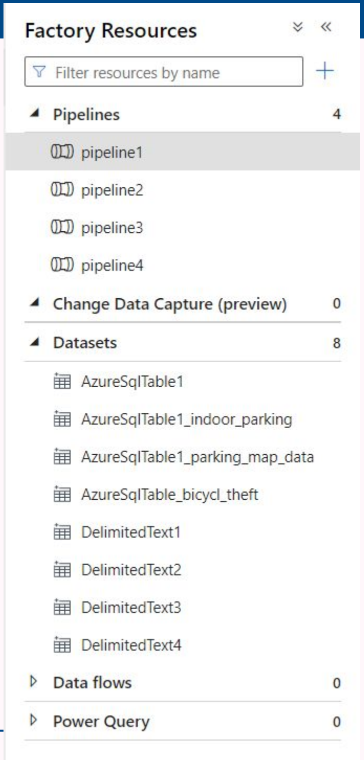
```
def loadBicycleTheftData():
    ctx = getConnection()
    df = pd.read_csv('./bicycle_thefts.csv')
    df = df.reset_index(drop=True)
    df.index.name = "id"
    df['o_date'] = df['Occurrence_Date'].apply(lambda x : x.split(' ')[0].replace("/","-"))
    with ctx.connect() as connection:
        df.to_sql('bicycle_thefts_new', con=ctx, if_exists='replace')
    print("End")
    dropConnection()
```

- After that, we meticulously processed the data by removing irrelevant entries and retaining only the pertinent ones.
- Subsequently, we integrated these tables into a single cohesive unit to generate a final, comprehensive table.
- Finally, we uploaded this master table back to the Data Factory, completing the task with proficiency and efficiency.

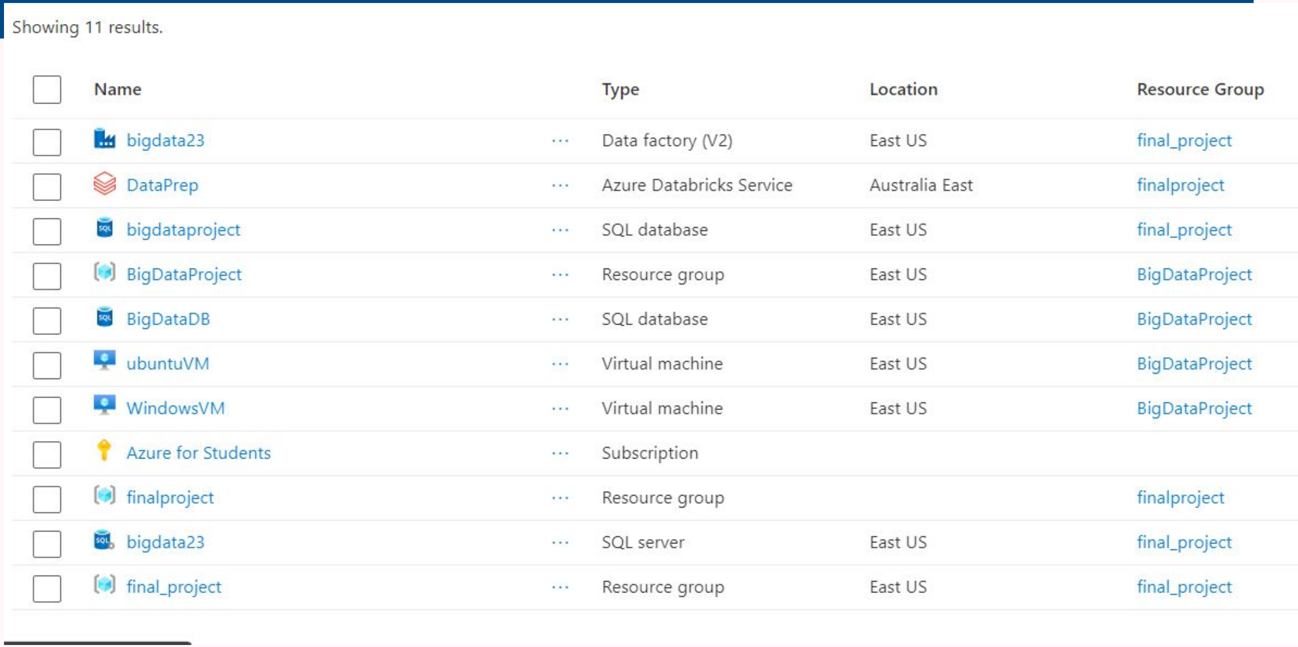
```
def prepareBicycleTheftData():
    theftData = getBicycleTheftData()
    parkingData = getBicycleParkingData()
    indoorParkingData = getBicycleIndoorParkingData()
    list = []
    for index, row in theftData.iterrows():
        coords = parseCoords(f"({row['Longitude']},{row['Latitude']})")
        parkingId, parkingDistance = getRowWithMinimumDistance(
            parkingData, 'geometry', coords)
        indoorParkingId, indoorParkingDistance = getRowWithMinimumDistance(
            indoorParkingData, 'geometry', coords)
        if (parkingId != None):
            parkingId = int(parkingId)
        if (indoorParkingId != None):
            indoorParkingId = int(indoorParkingId)

        list.append([index,parkingId,parkingDistance,indoorParkingId,indoorParkingDistance,row['o_date']])

    df = pd.DataFrame.from_records(data=list,columns=['theftId','parkingId','parkingDistance','indoorParkingId','indoorParkingDistance','o_date'])
    ctx = getConnection()
    with ctx.connect() as conn:
        df.to_sql('final_table',ctx,if_exists='replace')
```


```
    df = pd.DataFrame.from_records(data=list,columns=['theftId','parkingId','parkingDistance','indoorParkingId','indoorParkingDistance','o_date'])
    ctx = getConnection()
    with ctx.connect() as conn:
        df.to_sql('final_table',ctx,if_exists='replace')
```

DataFactory Resources

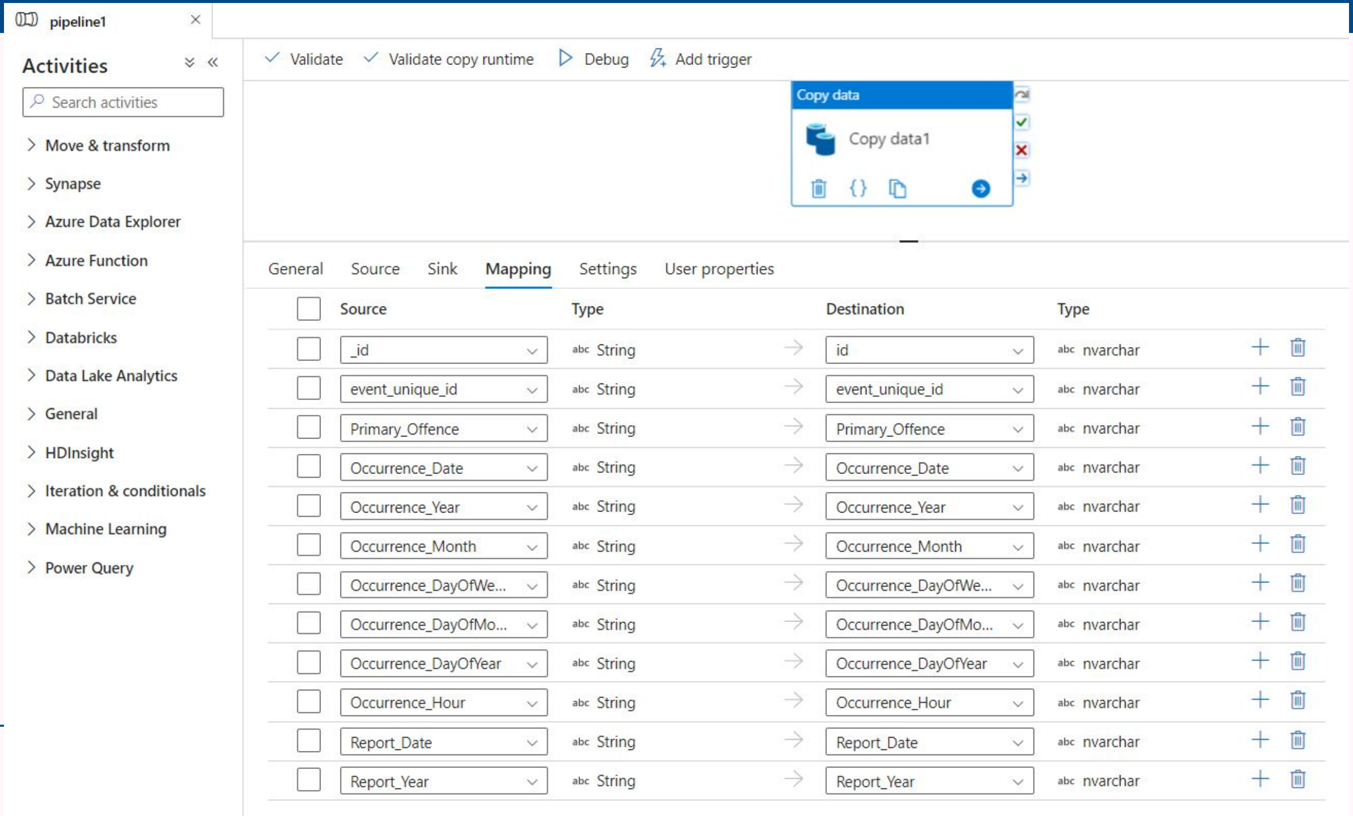


Azure Resources


Showing 11 results.

Name	Type	Location	Resource Group
bigdata23	Data factory (V2)	East US	final_project
DataPrep	Azure Databricks Service	Australia East	finalproject
bigdataproject	SQL database	East US	final_project
BigDataProject	Resource group	East US	BigDataProject
BigDataDB	SQL database	East US	BigDataProject
ubuntuVM	Virtual machine	East US	BigDataProject
WindowsVM	Virtual machine	East US	BigDataProject
Azure for Students	Subscription		
finalproject	Resource group		finalproject
bigdata23	SQL server	East US	final_project
final_project	Resource group	East US	final_project

Pipeline


pipeline1
Activities
Search activities
Move & transform
Synapse
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Validate
Validate copy runtime
Debug
Add trigger
Copy data
Copy data1
General
Source
Sink
Mapping
Settings
User properties
Source
Type
Destination
Type
_id
abc String
id
abc nvarchar
event_unique_id
abc String
event_unique_id
abc nvarchar
Primary_Offence
abc String
Primary_Offence
abc nvarchar
Occurrence_Date
abc String
Occurrence_Date
abc nvarchar
Occurrence_Year
abc String
Occurrence_Year
abc nvarchar
Occurrence_Month
abc String
Occurrence_Month
abc nvarchar
Occurrence_DayOfWe...
abc String
Occurrence_DayOfWe...
abc nvarchar
Occurrence_DayOfMo...
abc String
Occurrence_DayOfMo...
abc nvarchar
Occurrence_DayOfYear
abc String
Occurrence_DayOfYear
abc nvarchar
Occurrence_Hour
abc String
Occurrence_Hour
abc nvarchar
Report_Date
abc String
Report_Date
abc nvarchar
Report_Year
abc String
Report_Year
abc nvarchar

SQL Tables

Azure SQL Database
AzureSqlTable1

Connection Schema Parameters

Import schema Clear

Column name	Type
date	nvarchar
max_temperature	nvarchar
avg_hourly_temperature	nvarchar
avg_temperature	nvarchar
min_temperature	nvarchar
max_humidex	nvarchar
min_windchill	nvarchar
max_relative_humidity	nvarchar
avg_hourly_relative_humidity	nvarchar
avg_relative_humidity	nvarchar
min_relative_humidity	nvarchar
max_dew_point	nvarchar

SQL Tables (Delimited Text)

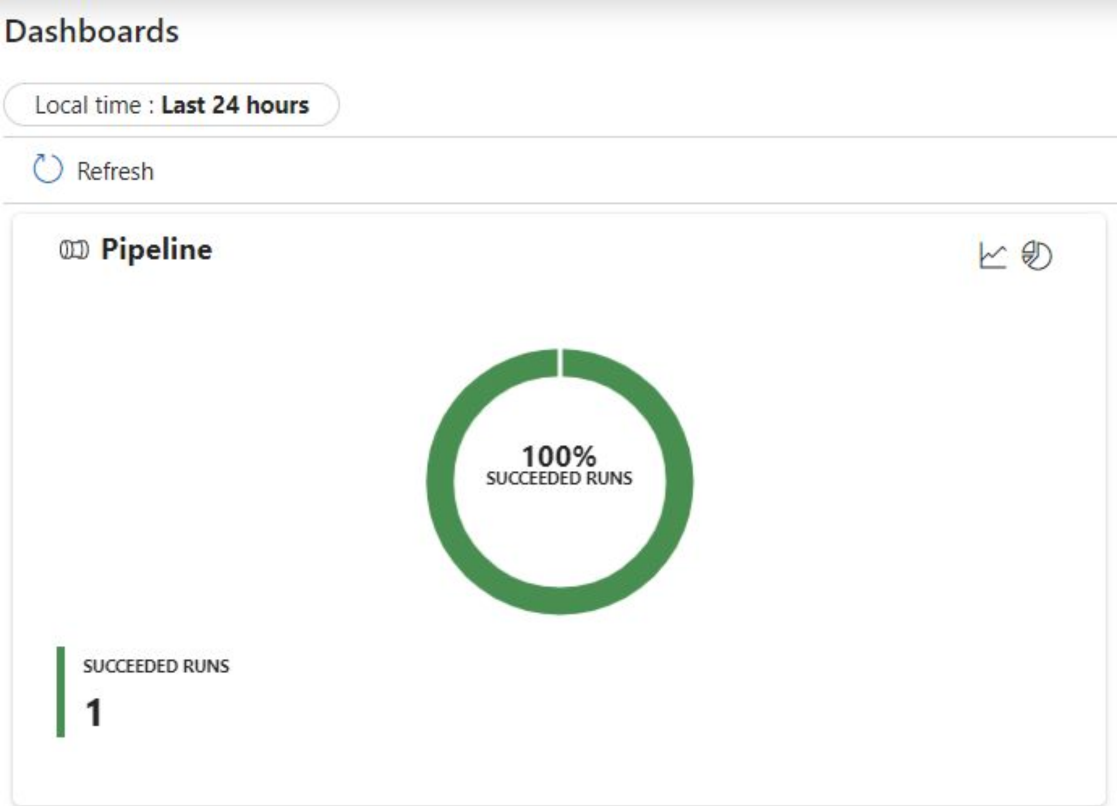
DelimitedText
DelimitedText1

Connection | **Schema** | Parameters

Import schema | Clear

Column name	Type
_id	String
event_unique_id	String
Primary_Offence	String
Occurrence_Date	String
Occurrence_Year	String
Occurrence_Month	String
Occurrence_DayOfWeek	String
Occurrence_DayOfMonth	String
Occurrence_DayOfYear	String
Occurrence_Hour	String
Report_Date	String
Report_Year	String

Dashboard


Dashboards
Local time : Last 24 hours
Refresh
Pipeline
100%
SUCCEEDED RUNS
SUCCEEDED RUNS
1

04

Data Visualisation

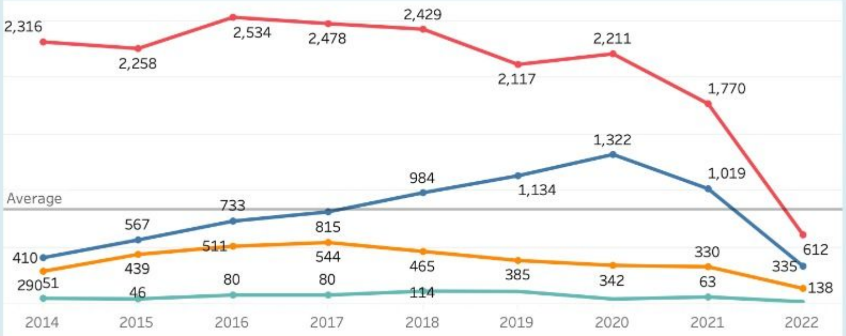
Visualizing the analysis done on the data and gaining insights

Bicycle Theft Analysis

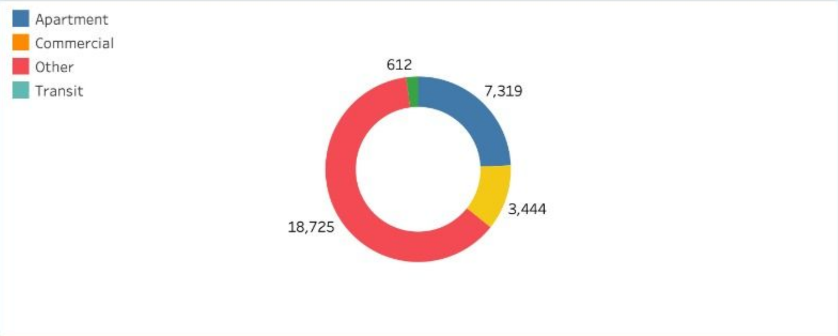
1/1/2014-6/30/2022

Map View

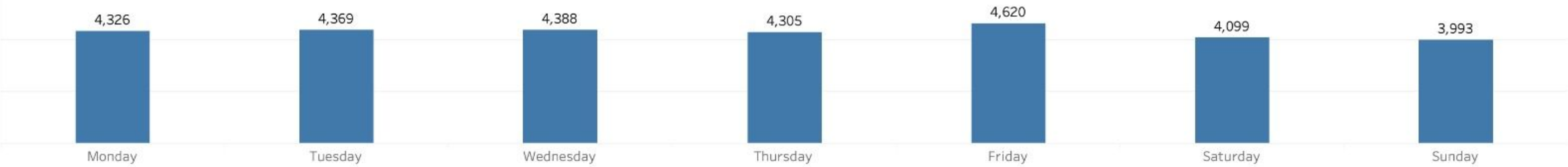
Time Series



Theft distribution by Premisis Type



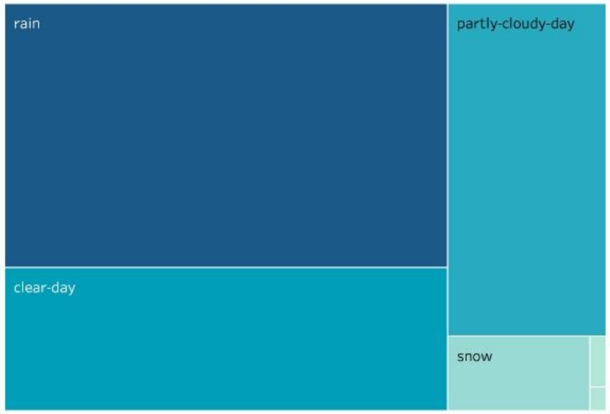
Comparison of Thefts by Days



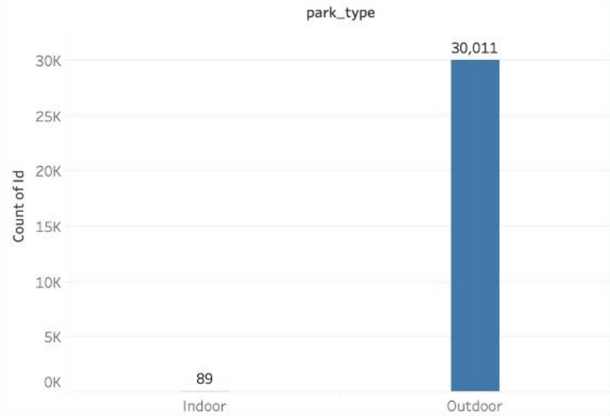
Bicycle Theft Analysis

1/1/2014-6/30/2022

Theft Comparison by Weather



Theft by Parking Type



Top 10 Neighborhoods of Theft

Neighbourhood Name	
Waterfront Communities-The Island	3,032
Bay Street Corridor	2,460
Church-Yonge Corridor	2,016
Niagara	1,115
Annex	1,100
Kensington-Chinatown	982
Moss Park	932
University	881
South Riverdale	824
Dovercourt-Wallace Emerson-Junction	732

Top 10 Bike Make of Theft

Bike Make	
OT	5,875
UK	2,807
GI	1,895
OTHER	1,782
TR	1,618
NO	1,089
CC	740
GIANT	705
UNKNOWN MAKE	682
SU	647

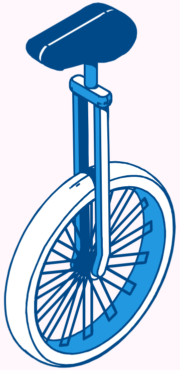
CONCLUSION

Bike theft is quite common on Fridays, than on other days. It's common to see that outdoor storage is more prone to theft than indoor. There's a strange relation with bike theft on rainy days.

Also some neighborhoods are less safe than others, we could also see some bike brands are more targeted than other. We believe this could be that some brands are more expensive than others, or that they are more easier to steal.

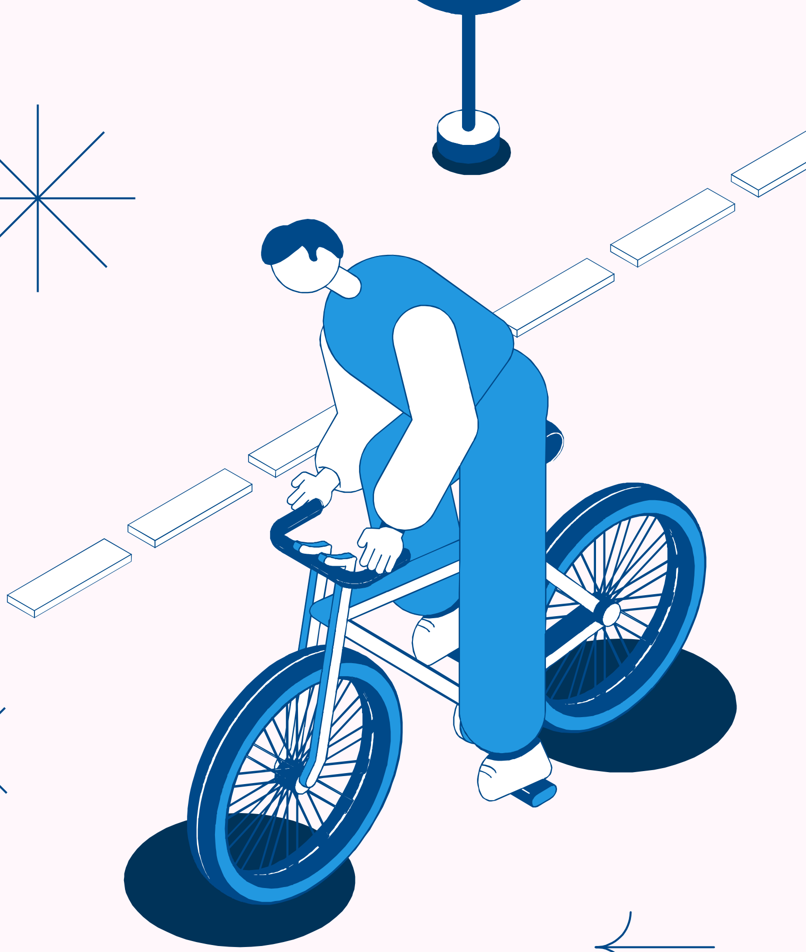
References

- Weather Dataset : https://weather.visualcrossing.com
- Toronto Bike Theft Dataset : https://open.toronto.ca/dataset/bicycle-thefts/
- Parking Dataset : https://open.toronto.ca/dataset/bicycle-parking-racks/



THANKS!

Does anyone have any questions?



CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution