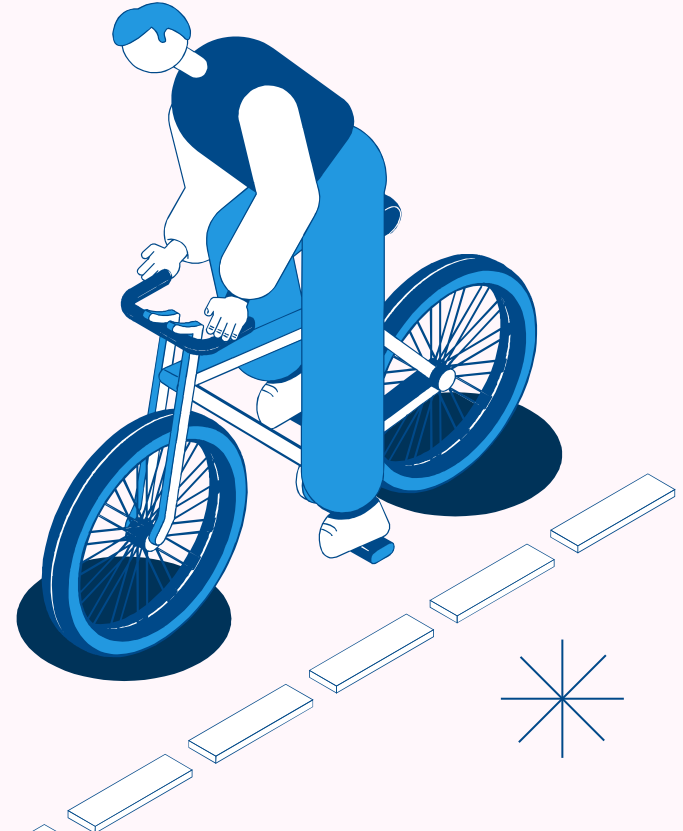
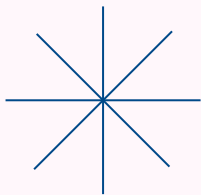
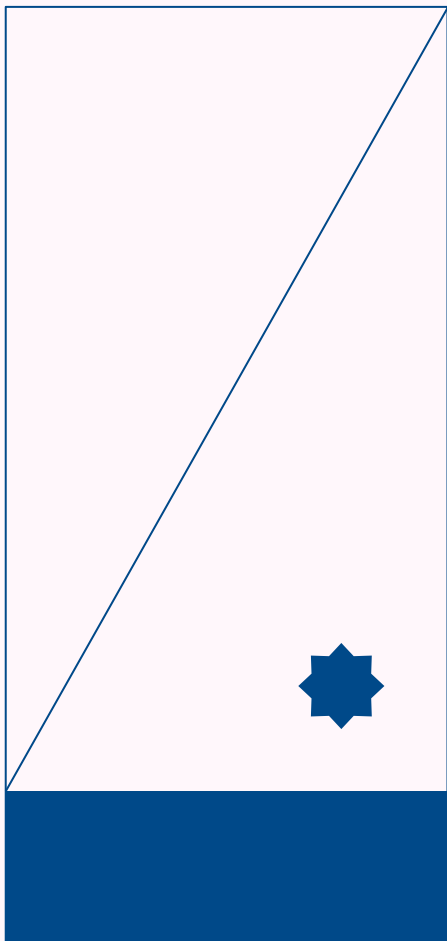


Bicycle Theft Analysis

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.

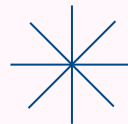
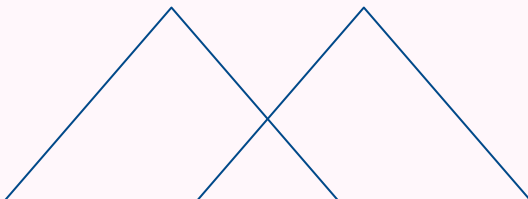
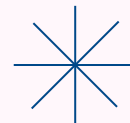




Table of Contents



01

Data Collection

Collection of data being used for analysis

02

Data Exploration

Identifying gaps and defining problems.

03

Data Transformation

Transforming the data for analysis. Involved cleaning data, combining tables, ETL

04

Data Visualization

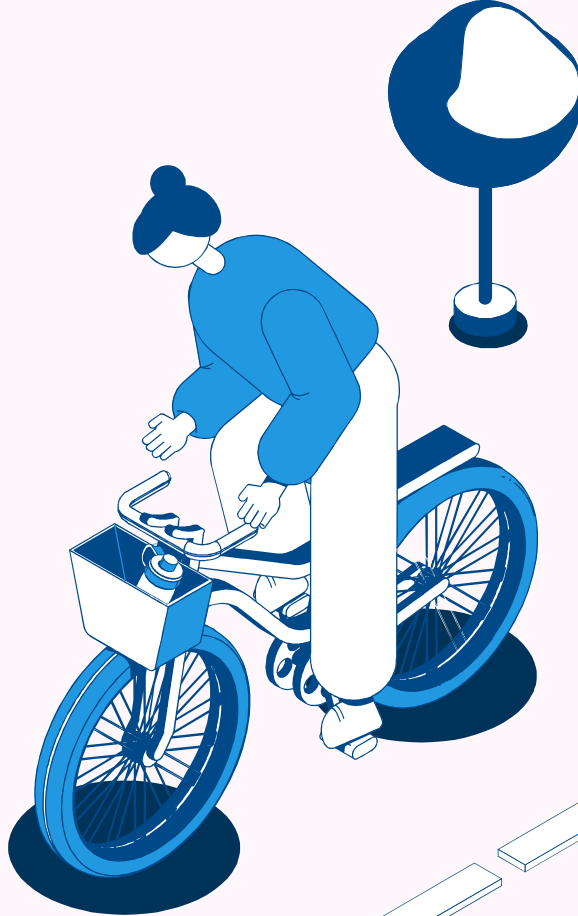
Visualizing the analysis done on the data and gaining insights



01

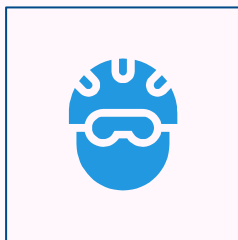
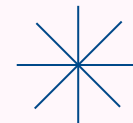
⋮ Data Collection

Collecting data from various sources





Datasets



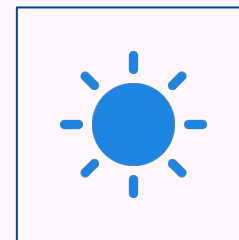
Bicycle Theft Dataset

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

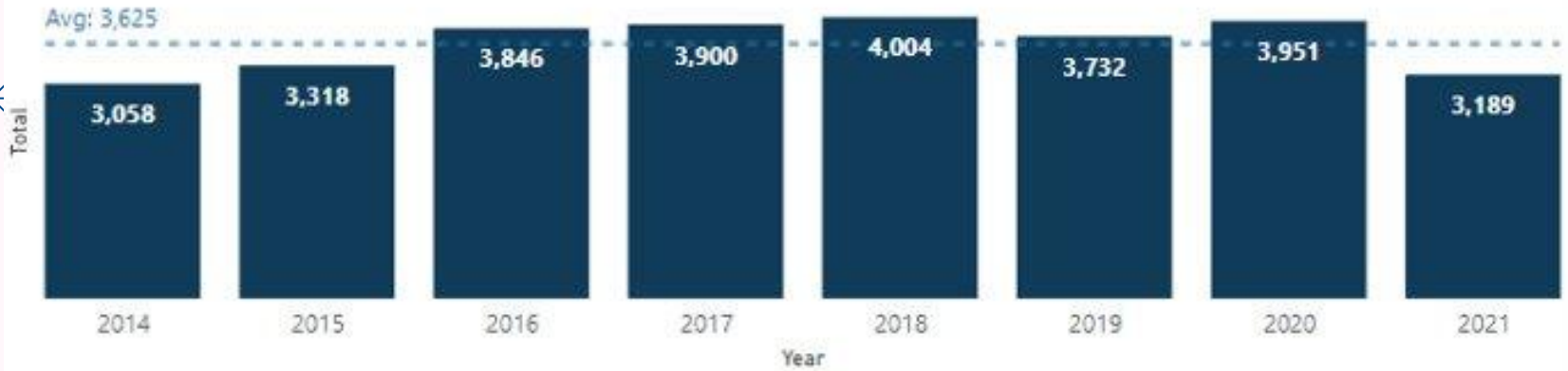


Data Collection

- 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



Bicycle Thefts by Year

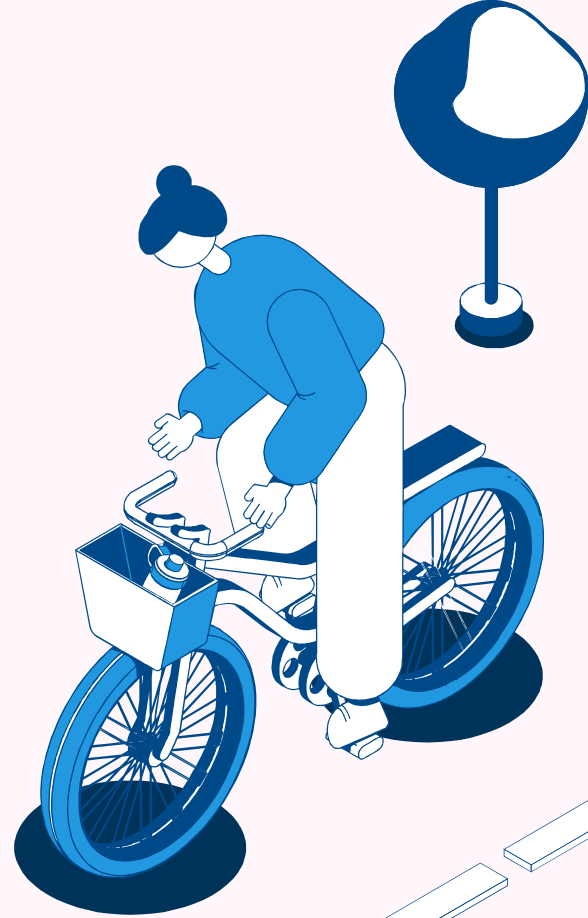


On average **3,625 bicycles** were stolen over a period of 7 years. The maximum(4,004) being in year 2018 and minimum(3,058) being in 2014

02

⋮ Data Exploration

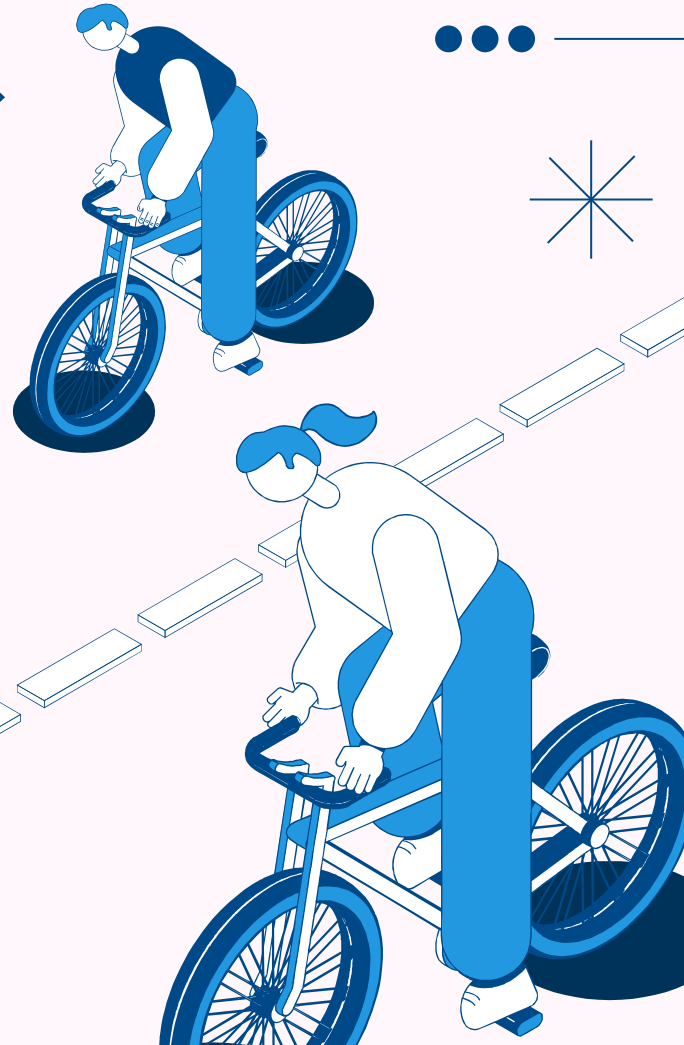
Identifying gaps and
defining problems



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.

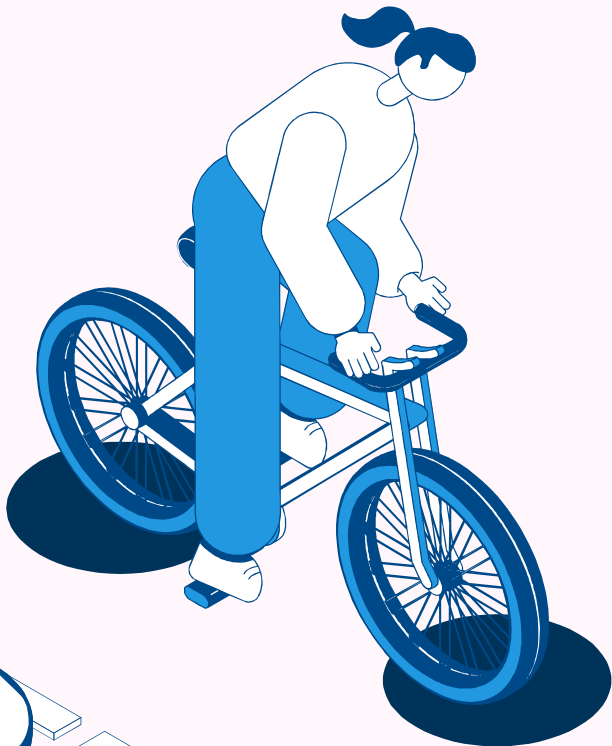
Bicycle theft can have a significant impact on individuals and the community as a whole. In addition to the cost of replacing a stolen bike, theft can also discourage people from using bicycles as a mode of transportation, which has negative environmental and health impacts.

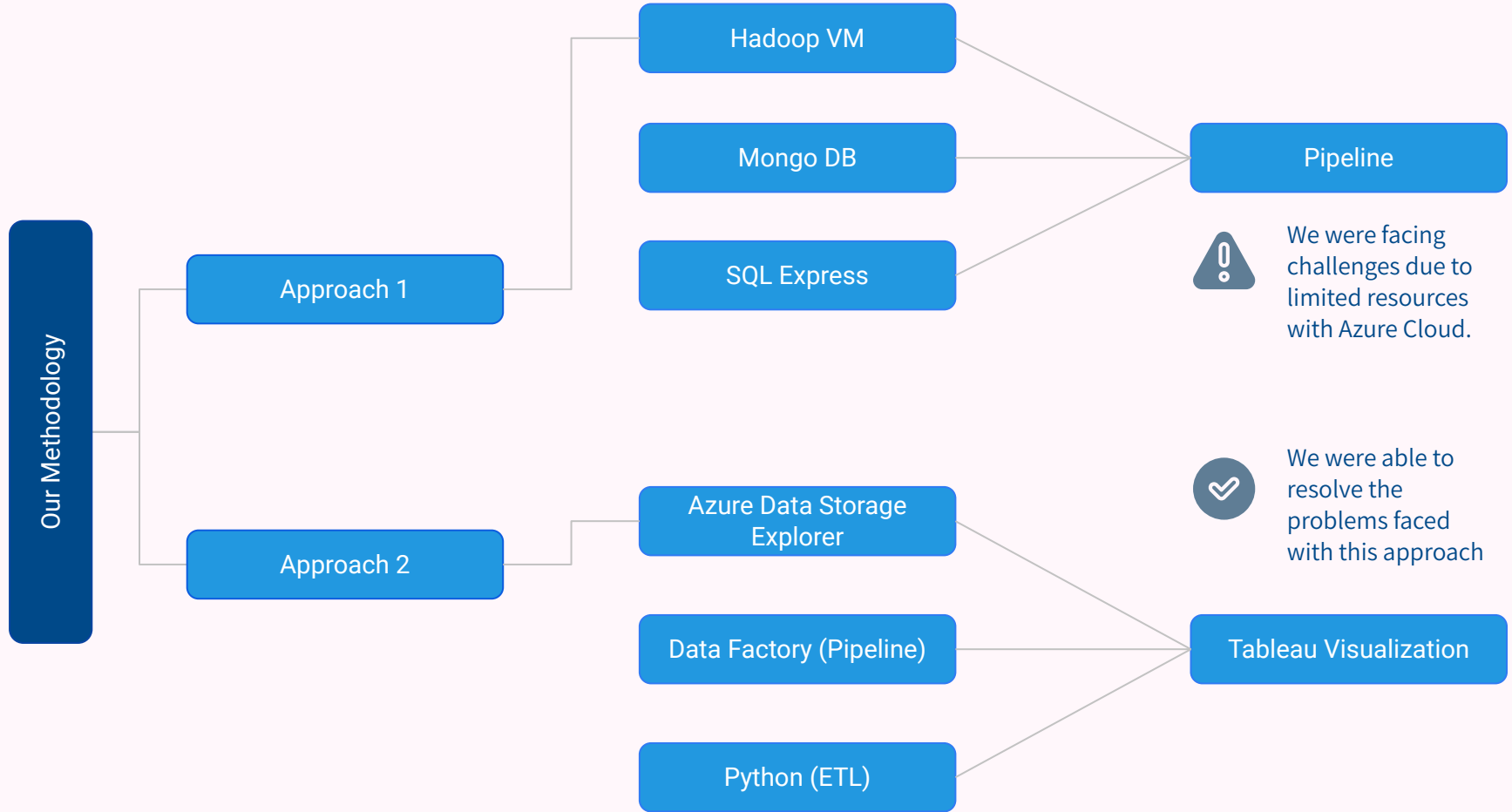


03

Data Transformation

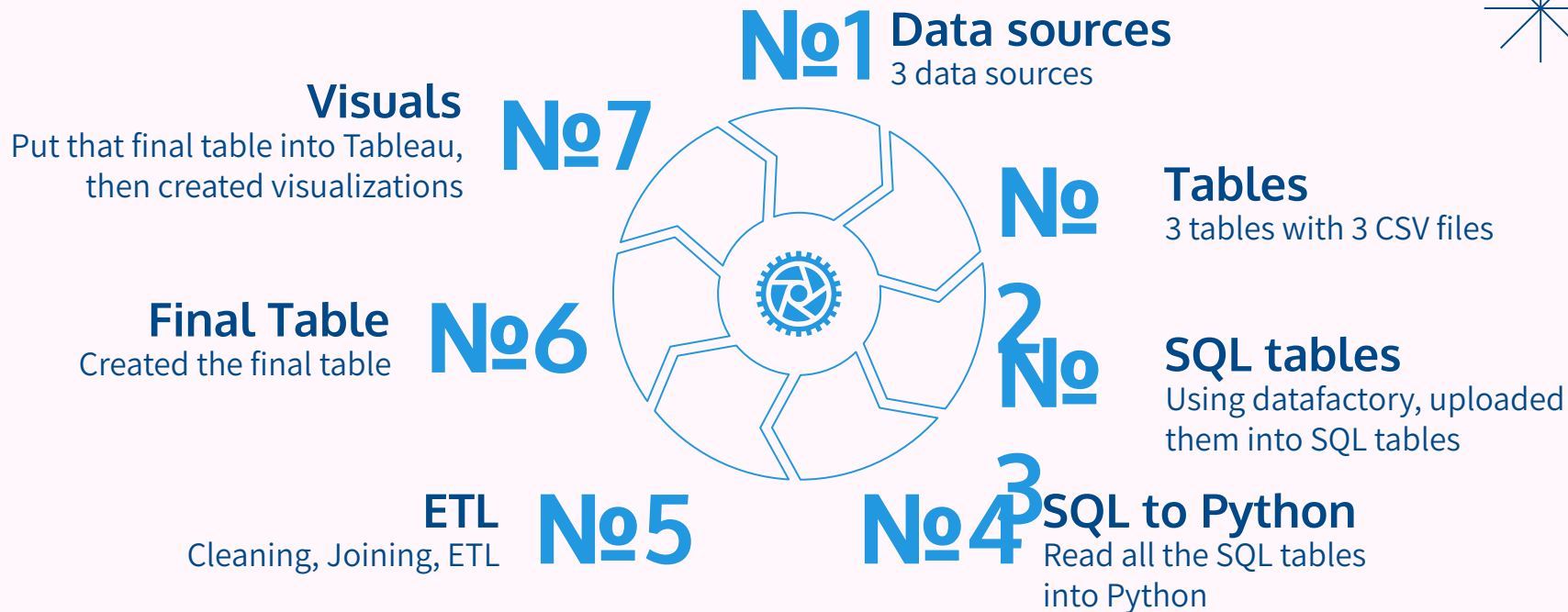
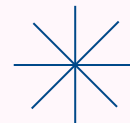
Converting data to a
standard workable format





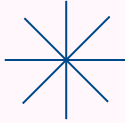


Sequential Process of Development





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

The screenshot shows a web browser window with the URL `127.0.0.1:9870/explorer.html#/user/hive/warehouse/theft.db`. The page title is "Browse Directory". Below the title, there is a search bar and a "Go" button. A table lists the contents of the directory:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwx-r-x	azureser	supergroup	0 B	Mar 18 18:44	0	0 B	bi_thets
drwx-r-x	azureser	supergroup	0 B	Mar 18 18:35	0	0 B	parking_indoor
drwx-r-x	azureser	supergroup	0 B	Mar 18 18:47	0	0 B	parking_map

Showing 1 to 3 of 3 entries

Previous 1 Next

Hadoop, 2022.

Hadoop VM : jps command

The screenshot shows the same Hadoop VM web interface as the previous image. A terminal window is open, displaying the output of the `jps` command:

```
azureser@ubuntuVM: ~$ jps
25777 RunJar
23567 DataNode
23300 NameNode
2283 Jps
23157 SecondaryNameNode
azureser@ubuntuVM: ~$
```

The terminal window is titled "azureser@ubuntuVM: ~". The output shows the following processes:

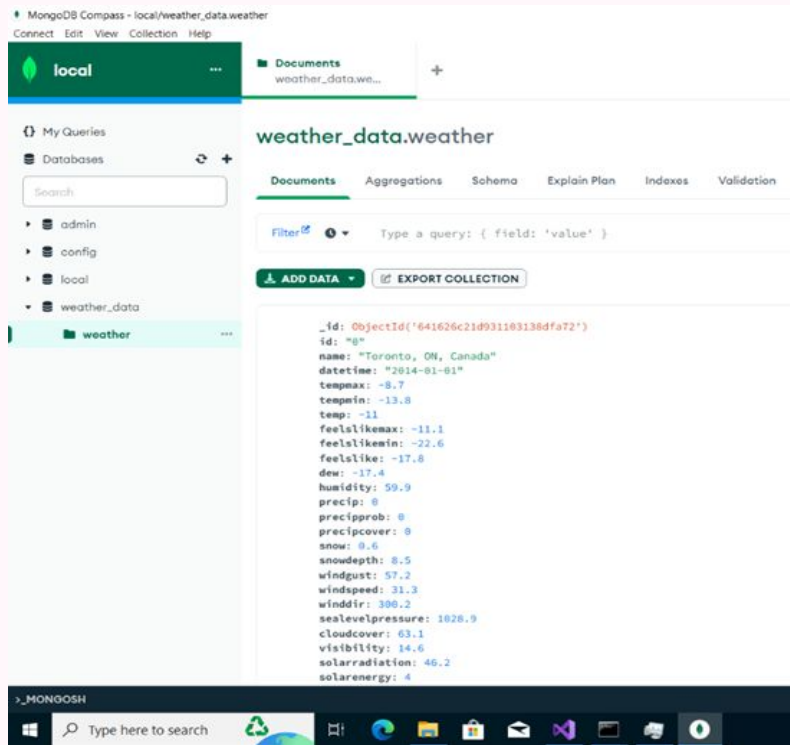
- 25777 RunJar
- 23567 DataNode
- 23300 NameNode
- 2283 Jps
- 23157 SecondaryNameNode

Showing 1 to 3 of 3 entries

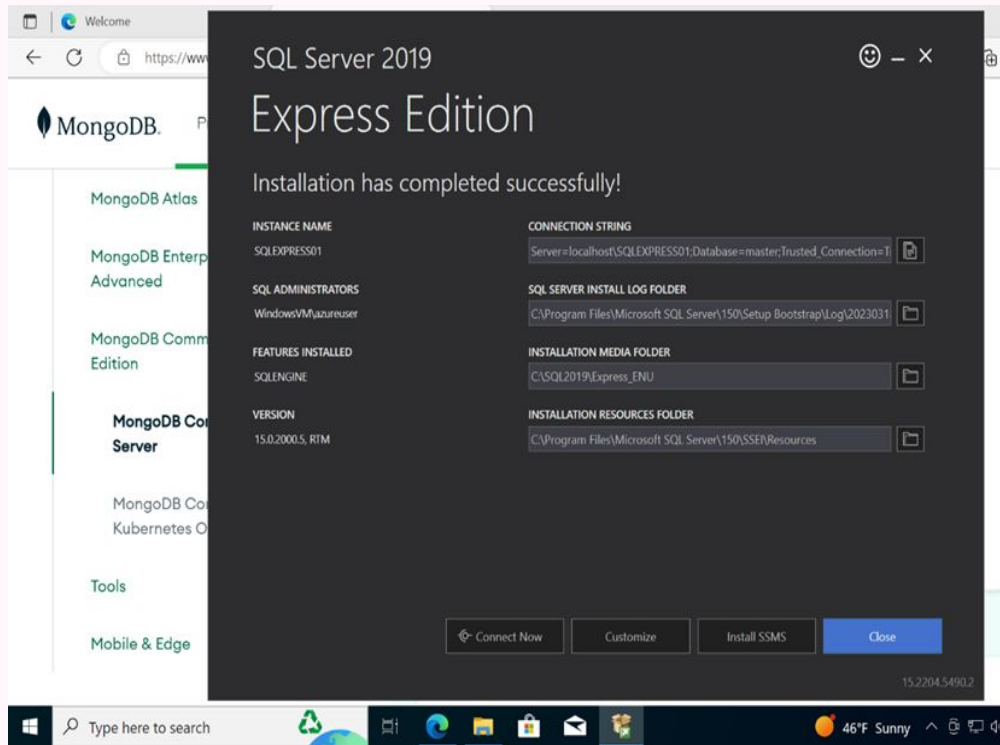
Previous 1 Next

Hadoop, 2022.

Windows VM: MongoDB



Windows VM : SQL Express



Pipeline (1 of 3)

The screenshot displays the Microsoft SQL Server Data Tools (SSDT) environment. The main window shows a Data Flow Task named 'FactTable'. The pipeline consists of the following tasks:

- Sources:** 'Mongo_Weather' and 'HDFS_HDFS_Thefts'.
- Transformations:** 'Data Conversion' (receiving input from 'Mongo_Weather'), 'Data Conversion 1' (receiving input from 'HDFS_HDFS_Thefts'), 'Sort' (receiving input from 'Data Conversion'), 'Sort 1' (receiving input from 'Data Conversion 1'), 'Merge Join' (receiving input from both 'Sort' and 'Sort 1'), and 'OLE DB Destination' (receiving input from 'Merge Join').

The Solution Explorer on the right shows the project structure for 'BigDataProject', including 'Project.params', 'Connection Managers', 'SSIS Packages', 'Package.dtsx', 'Package Parts', 'Control Flow', 'Miscellaneous', 'Linked Azure Resources', 'Azure-SSIS Integration Runtime', and 'Azure Storage'.

The Output window at the bottom shows the following messages:

```
Information: 0x00013009 at FactTable, SSIS.Pipeline: Cleanup phase is beginning.
Warning: 0x00013002 at FactTable: SSIS Warning Code DTS_W_MAXIMIZERREACHED. The Execution method succeeded, but the number of errors reached the maximum.
Task failed: FactTable
Warning: 0x00013002 at Package: SSIS Warning Code DTS_W_MAXIMIZERREACHED. The Execution method succeeded, but the number of errors reached the maximum.
SSIS package "C:\Users\azureuser\Source\Repos\BigDataProject\BigDataProject\Package.dtsx" finished: Failure.
The program "[10064] DtsDebugHost.exe: DTS" has exited with code 0 (0x0).
```

The bottom status bar shows the system clock as 5:03 AM on 3/19/2023, and the weather as 37°F Cloudy.

Dim Table (1 of 2)

SQLQuery_4 - bigdataprog1903.database.windows.net.BigDataDB (azureuser) - Azure Data Studio

```

1 SELECT distinct
2 ROW_NUMBER() OVER(ORDER BY Mongo_Weather_datetime) as dimOrderDateID
3 ,Mongo_Weather_datetime
4 ,FORMAT(Mongo_Weather_datetime,'dd-MM-yyyy') as DDMYYYY
5 ,FORMAT(Mongo_Weather_datetime,'dd MMM yyyy','en-us') as Date_En
6 ,FORMAT(Mongo_Weather_datetime,'dd MMM yyyy','no') as Date_Norwegian
7 ,FORMAT(Mongo_Weather_datetime,'dd MMM yyyy','sv-Sw') as Date_Switzerland
8 ,FORMAT(Mongo_Weather_datetime,'dd MMM yyyy','es') as Date_Spanish
9 ,FORMAT(Mongo_Weather_datetime,'dd MMM yyyy','fr') as Date_French
10 ,FORMAT(Mongo_Weather_datetime,'dd MMM yyyy','it') as Date_Italian
11 ,FORMAT(Mongo_Weather_datetime,'MMMM') as MMMYYYY
12 ,FORMAT(Mongo_Weather_datetime,'YYYY') as YYYY
13 ,FORMAT(Mongo_Weather_datetime,'MM') as MM
14 ,FORMAT(Mongo_Weather_datetime,'MMMM') as NonName
15 ,FORMAT(Mongo_Weather_datetime,'MMMM','no') as NonName_Norwegian
16 ,FORMAT(Mongo_Weather_datetime,'MMMM','sv-Sw') as NonName_Switzerland
17 ,FORMAT(Mongo_Weather_datetime,'MMMM','es') as NonName_Spanish
18 ,FORMAT(Mongo_Weather_datetime,'MMMM','fr') as NonName_French
19 ,FORMAT(Mongo_Weather_datetime,'MMMM','it') as NonName_Italian
20 from
21 dbo.vfactThrift
  
```

Results Messages

dimOrderDateID	Mongo_Weather_d...	DDMMYYYY	Date_En	Date_Norwegian	Date_Switzerland
1					

Ln 1, Col 16 Spaces: 4 UTF-8 CR/LF 0 rows MSSQL 00:00:00 bigdataprog

Fact Table

SQLQuery_2 - bigdataprog1903.database.windows.net.BigDataDB (azureuser) - Azure Data Studio

```

1 SELECT TOP (1000) Mongo_Weather_id
2 ,Mongo_Weather_cloudcover
3 ,Mongo_Weather_conditions
4 ,Mongo_Weather_datetime
5 ,Mongo_Weather_description
6 ,Mongo_Weather_id
7 ,Mongo_Weather_conditions
8 ,Mongo_Weather_conditions
9 ,Mongo_Weather_conditions
10 ,Mongo_Weather_conditions
11 ,Mongo_Weather_conditions
12 ,Mongo_Weather_conditions
13 ,Mongo_Weather_conditions
14 ,Mongo_Weather_conditions
15 ,Mongo_Weather_conditions
16 ,Mongo_Weather_conditions
17 ,Mongo_Weather_conditions
18 ,Mongo_Weather_conditions
19 ,Mongo_Weather_conditions
  
```

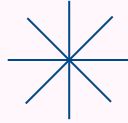
Results Messages

Mongo_Weather_id	Mongo_Weather_cloudcover	Mongo_Weather_conditions	Mongo_Weather_datetime	Mongo_Weather_description
1	641629114931103138e0b7	63.1	2014-01-01 00:00:00.000	Partly cloudy throughout the
2	641629114931103138e0b8	99	2014-01-02 00:00:00.000	Cloudy skies throughout the
3	641629114931103138e0b9	19.2	2014-01-03 00:00:00.000	Clear conditions throughout
4	641629114931103138e0ba	65.8	2014-01-04 00:00:00.000	Partly cloudy throughout the
5	641629114931103138e0bb	95.8	2014-01-05 00:00:00.000	Cloudy skies throughout the
6	641629114931103138e0bc	68.9	2014-01-06 00:00:00.000	Partly cloudy throughout the
7	641629114931103138e0bd	42.2	2014-01-07 00:00:00.000	Becoming cloudy in the after
8	641629114931103138e0be	67.4	2014-01-08 00:00:00.000	Partly cloudy throughout the
9	641629114931103138e0bf	25.5	2014-01-09 00:00:00.000	Partly cloudy throughout the
10	641629114931103138e0c0	81	2014-01-10 00:00:00.000	Partly cloudy throughout the
11	641629114931103138e0c1	83.3	2014-01-11 00:00:00.000	Partly cloudy throughout the
12	641629114931103138e0c2	35	2014-01-12 00:00:00.000	Partly cloudy throughout the
13	641629114931103138e0c3	59.2	2014-01-13 00:00:00.000	Partly cloudy throughout the
14	641629114931103138e0c4	42.7	2014-01-14 00:00:00.000	Partly cloudy throughout the

Ln 1, Col 1 Spaces: 4 UTF-8 CR/LF 1000 rows MSSQL 00:00:01 bigdataprog1903.database.windows.net: BigDataDB



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=bigdatapipeline;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

Factory Resources ⌵ ⌵

+

▲ Pipelines 4

▶ pipeline1

▶ pipeline2

▶ pipeline3

▶ pipeline4

▲ Change Data Capture (preview) 0

▶ Datasets 8

▶ AzureSqlTable1

▶ AzureSqlTable1_indoor_parking

▶ AzureSqlTable1_parking_map_data

▶ AzureSqlTable_bicycl_theft

▶ DelimitedText1

▶ DelimitedText2












▶ DelimitedText3

▶ DelimitedText4

▶ Data flows 0

▶ Power Query 0

Showing 11 results.

<input type="checkbox"/>	Name	Type	Location	Resource Group
<input type="checkbox"/>	 bigdata23	... Data factory (V2)	East US	final_project
<input type="checkbox"/>	 DataPrep	... Azure Databricks Service	Australia East	finalproject
<input type="checkbox"/>	 bigdataproject	... SQL database	East US	final_project
<input type="checkbox"/>	 BigDataProject	... Resource group	East US	BigDataProject
<input type="checkbox"/>	 BigDataDB	... SQL database	East US	BigDataProject
<input type="checkbox"/>	 ubuntuVM	... Virtual machine	East US	BigDataProject
<input type="checkbox"/>	 WindowsVM	... Virtual machine	East US	BigDataProject
<input type="checkbox"/>	 Azure for Students	... Subscription		
<input type="checkbox"/>	 finalproject	... Resource group		finalproject
<input type="checkbox"/>	 bigdata23	... SQL server	East US	final_project
<input type="checkbox"/>	 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

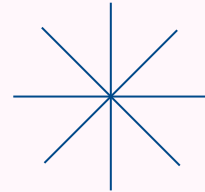
Copy data

Copy data1

🗑️ {} 📄 + ➡️

General Source Sink Mapping Settings User properties

<input type="checkbox"/>	Source	Type		Destination	Type		
<input type="checkbox"/>	_id	abc String	➔	id	abc nvarchar	+	🗑️
<input type="checkbox"/>	event_unique_id	abc String	➔	event_unique_id	abc nvarchar	+	🗑️
<input type="checkbox"/>	Primary_Offence	abc String	➔	Primary_Offence	abc nvarchar	+	🗑️
<input type="checkbox"/>	Occurrence_Date	abc String	➔	Occurrence_Date	abc nvarchar	+	🗑️
<input type="checkbox"/>	Occurrence_Year	abc String	➔	Occurrence_Year	abc nvarchar	+	🗑️
<input type="checkbox"/>	Occurrence_Month	abc String	➔	Occurrence_Month	abc nvarchar	+	🗑️
<input type="checkbox"/>	Occurrence_DayOfWe...	abc String	➔	Occurrence_DayOfWe...	abc nvarchar	+	🗑️
<input type="checkbox"/>	Occurrence_DayOfMo...	abc String	➔	Occurrence_DayOfMo...	abc nvarchar	+	🗑️
<input type="checkbox"/>	Occurrence_DayOfYear	abc String	➔	Occurrence_DayOfYear	abc nvarchar	+	🗑️
<input type="checkbox"/>	Occurrence_Hour	abc String	➔	Occurrence_Hour	abc nvarchar	+	🗑️
<input type="checkbox"/>	Report_Date	abc String	➔	Report_Date	abc nvarchar	+	🗑️
<input type="checkbox"/>	Report_Year	abc String	➔	Report_Year	abc nvarchar	+	🗑️



SQL Tables



Azure SQL Database

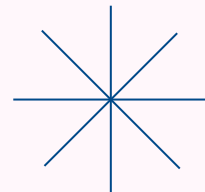
AzureSqlTable1

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)



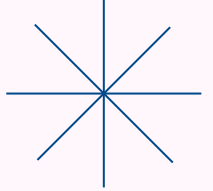
DelimitedText
DelimitedText1

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**

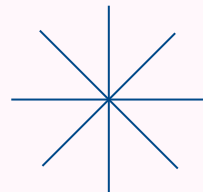
 Refresh

 **Pipeline**



SUCCEEDED RUNS

1





04

Data Visualisation

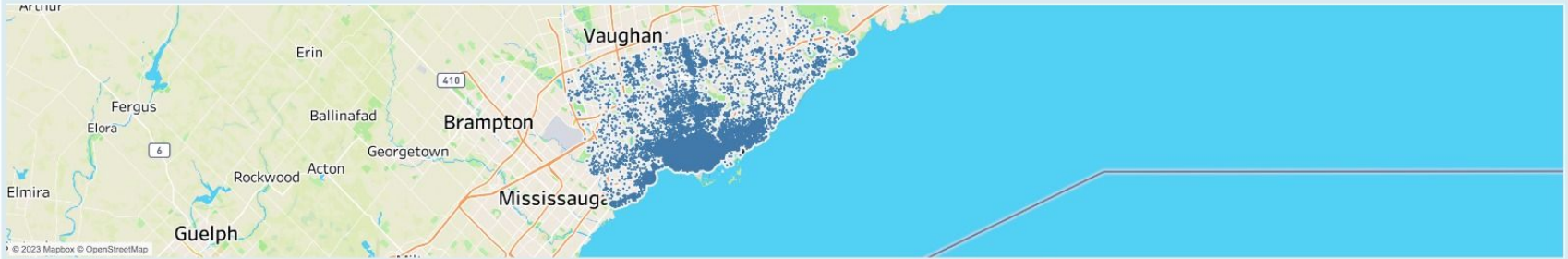
Visualizing the analysis done on the
data and gaining insights



Bicycle Theft Analysis

1/1/2014-6/30/2022

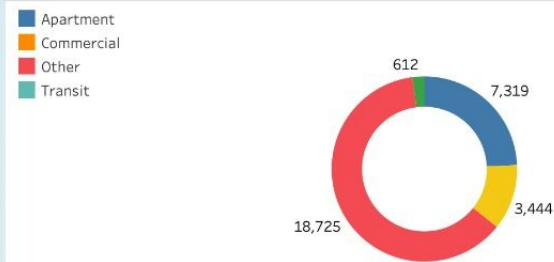
Map View



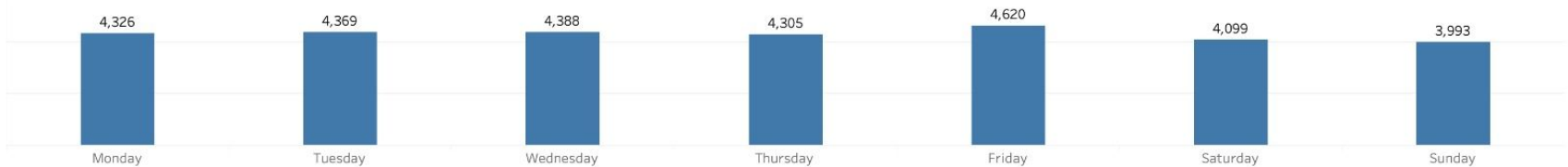
Time Series



Theft distribution by Premisis Type



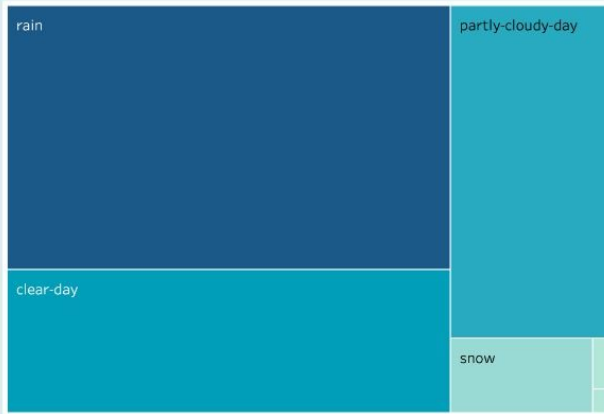
Comparison of Thefts by Days



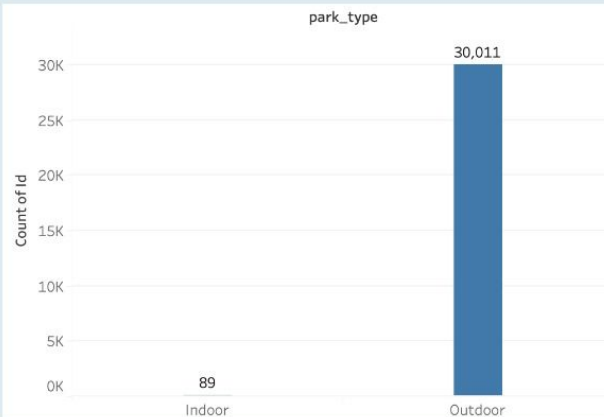
Bicycle Theft Analysis

1/1/2014-6/30/2022

Theft Comparison by Weather



Theft by Parking Type

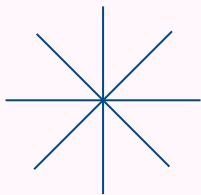
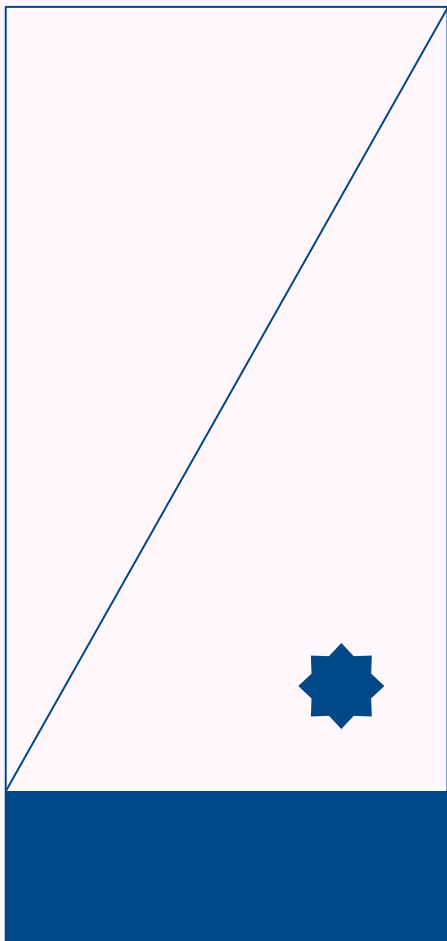


Top 10 Neighborhoods of Theft

Neighbourhood Name	Count
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

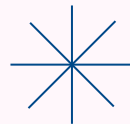
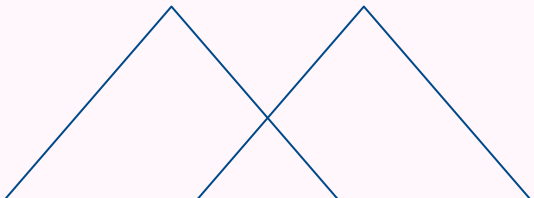
Bike Make	Count
OT	5,875
UK	2,807
GI	1,895
OTHER	1,782
TR	1,618
NO	1,089
CC	740
GIANT	705
UNKNOWN MAKE	682
SU	647



CONCLUSION

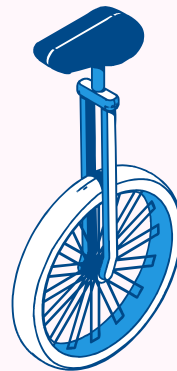
Bike theft is quite common on Fridays, than on other days. It's common to see that outdoor storage is more prone to theft than indoor. There's a strange relation with bike theft on rainy days.

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

