Big Data II Project: Analyzing the Influence of Weather, Unemployment, Mental Health, Race, Budget, and Staffing on Crime Rates

Course:
Big Data Tools and Technologies II
CRN-52350-202302

Group Members:
Dabas, Rashika
Dimaculangan, Roan Kathrina
Ip, Chak Leung Vincent
Muthusamy, Karthikeyan
Himani.
Bhumika Pravinkumar Shukla
Erinda Kapllani
Kaushal Ghelani
Ahmed Ali
Kumud

Table of Contents

I. Introduction

Overview of the Project

This project looks into the intricate relationships between various environmental and socioeconomic factors and their impact on crime rates. By leveraging a comprehensive set of datasets, including weather patterns, unemployment rates, mental health incident reports, race-based data, government budget allocations for law enforcement, staffing levels within law enforcement agencies, victims of crime statistics, and detailed crime reports (specifically break and enter, homicide, and traffic collisions), this research aims to uncover the multifaceted influences that these variables exert on criminal activities. The ultimate goal is to provide a holistic understanding of how external conditions and societal issues contribute to crime dynamics within communities.

Objectives and Goals

Based on the required activities needed to be performed in the project – data integration and pre-processing, Machine Learning and creation of reports to analyze impact of one dataset to another, below are the primary Objectives and Goals identified by the team:

- **Advanced Data Integration and Processing:** Utilize Hadoop, MongoDB, MariaDB, Azure SQL, and Azure SQL Server Express for data ingestion and storage. Employ Snowflake, Alteryx, MS SQL Server Integration Services, and Azure Analysis Services for processing to create a unified, comprehensive database. This database will integrate weather, unemployment, mental health, racial demographics, law enforcement budgets, and staffing levels to analyze their impact on crime rates.

- **Predictive Analysis and Machine Learning Implementation:** Develop predictive models, including two advanced deep learning models, to forecast crime trends based on the integrated data. The project aims to provide actionable insights for preemptive and strategic law enforcement and policy-making, thereby enhancing public safety and community well-being.

- **Crime Rate vs. Weather Conditions Analysis:** Investigate the correlation between various weather conditions and crime rates. Use time series analysis to track how crime trends fluctuate with seasonal weather patterns, providing a foundation for weather-adaptive crime prevention strategies.

- **Economic Indicators and Crime Dynamics Exploration:** Examine the linkage between unemployment rates and crime frequencies across different regions. Conduct a comparative analysis to highlight the influence of economic conditions on crime, guiding resource allocation for economic and social interventions.

- **Mental Health and Crime Correlation Study:** Correlate mental health incident data with crime statistics to identify possible connections. This analysis will support

the development of mental health-focused preventive measures and law enforcement strategies.

- **Law Enforcement Resource Allocation Efficiency Analysis:** Compare law enforcement budget allocations with crime statistics to assess the efficiency of financial resource distribution in crime prevention and resolution. This will inform budgetary decisions for maximizing public safety outcomes.

- **Staffing Impact on Law Enforcement Effectiveness Study:** Analyze how staffing levels within law enforcement agencies influence crime rates and response times. Insights from this study will aid in optimizing staffing strategies to enhance law enforcement efficiency and effectiveness.

II. Data Sources Overview

The project leverages a diverse array of datasets, carefully selected to provide a comprehensive understanding of the influences on crime rates. Below are the eight (8) datasets selected by the team:

Table 1 Data Source Overview Diagram

Dataset Name/ Source Link	Description	Is Big Data?
combined_break_enter_open_homicide_traffic_collisions_file.csv https://data.torontopolice.on.ca/datasets/040ead448df2412da252cfbb532e77ac_0/explore https://data.torontopolice.on.ca/datasets/d96bf5b67c1c49879f354dad51cf81f9_0/explore?location=43.722845%2C-79.374074%2C10.33 https://data.torontopolice.on.ca/datasets/bc4c72a793014a55a674984ef175a6f3_0/explore	A compilation of various crime incidents, offering a detailed view of crime patterns and trends.	**Yes**
weather_data.csv https://www.kaggle.com/datasets/jasonzxho/toronto-city-centre-hourly-weather-data-20172020/code	Records of weather conditions, essential for analyzing the impact of environmental factors on crime rates. This is. crucial for examining how environmental conditions might affect crime dynamics.	**Yes**

unemployment_canada.csv https://www.kaggle.com/datasets/pienik/unemployment-in-canada-by-province-1976-present	Statistics on unemployment rates across different regions, to examine economic influences on crime.	No
mental_health_act_apprehensions_open_data.csv https://data.torontopolice.on.ca/datasets/333c4e1c96314741a83425045b6a7642_0/explore	Data on mental health-related law enforcement interventions, crucial for understanding the relationship between mental health issues and crime.	**Yes**
race_based_data.csv https://data.torontopolice.on.ca/datasets/TorontoPS::use-of-force-location-of-occurrences-rbdc-uof-tbl-003/explore	Information on the racial demographics of individuals involved in criminal incidents, to explore racial disparities in crime rates.	No
budget_by_command.csv https://data.torontopolice.on.ca/datasets/TorontoPS::budget-by-command/explore	Financial data detailing law enforcement budgets, to assess the effect of resource allocation on crime prevention and resolution.	No
staffing_by_command.csv https://data.torontopolice.on.ca/datasets/2c5dcbc96e65479389d75f7e65001658_0/explore	Details on law enforcement staffing levels, to investigate the impact of manpower on crime rates and response times.	No
victims_of_crime.csv https://data.torontopolice.on.ca/datasets/victims-of-crime-asr-vc-tbl-001/explore	Data on crime victims, providing insights into victimology and crime impact on different demographics.	No

III. Data Flow Diagram

The Data Flow Diagram encapsulates the lifecycle of data processing from collection to visualization. Below is a detailed breakdown of the data journey:

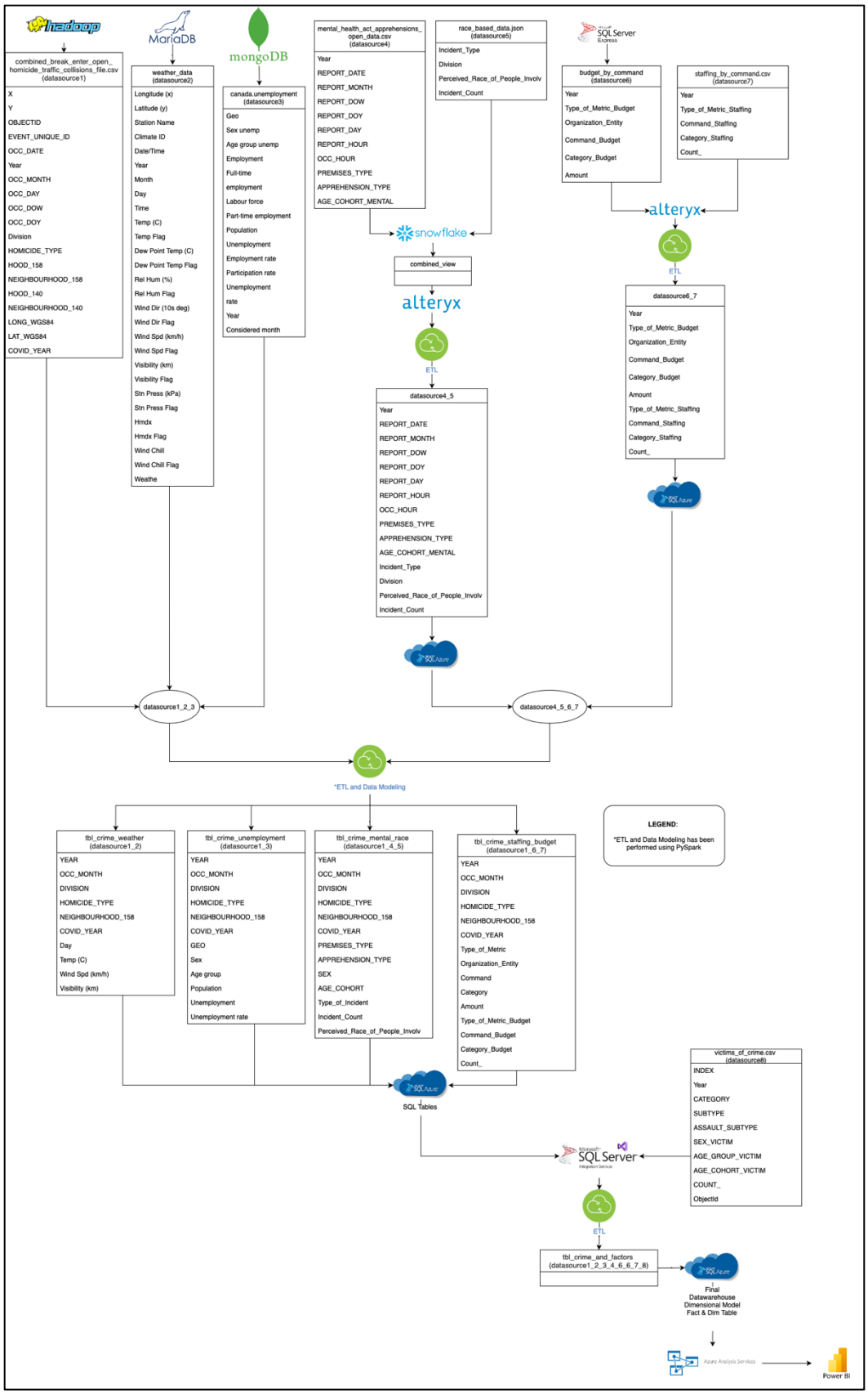

Figure 1 Data Flow Diagram

A. Data Sources:

- **Datasource1:** combined_break_enter_open_homicide_traffic_collisions_file.csv loaded into Hadoop HDFS

```
azureuser@myVM:~/data/crimedata$ hadoop fs  -ls /user/input/crimedata
Found 8 items
-rw-r--r--   1 azureuser supergroup   41451375 2024-03-08 19:44 /user/input/crimedata/1.csv
-rw-r--r--   1 azureuser supergroup    3949534 2024-03-08 19:44 /user/input/crimedata/2.csv
-rw-r--r--   1 azureuser supergroup     346004 2024-03-08 19:44 /user/input/crimedata/3.csv
-rw-r--r--   1 azureuser supergroup    9747551 2024-03-08 19:44 /user/input/crimedata/4.csv
-rw-r--r--   1 azureuser supergroup      14410 2024-03-08 19:44 /user/input/crimedata/5.csv
-rw-r--r--   1 azureuser supergroup      62854 2024-03-08 19:44 /user/input/crimedata/6.csv
-rw-r--r--   1 azureuser supergroup       9592 2024-03-08 19:44 /user/input/crimedata/7.csv
-rw-r--r--   1 azureuser supergroup      40675 2024-03-08 19:44 /user/input/crimedata/8.csv
azureuser@myVM:~/data/crimedata$ exit
logout
Connection to 172.191.18.208 closed.
```

Figure 2 Crime Data (Datasource1) loaded in Hadoop

- **Datasource2:** weather_data.csv stored in MariaDB

- **Datasource3:** unemployment_canada.csv data is integrated into MongoDB under below specified fields:
    - Cluster Name – Cluster0
    - Database Name – unemployment_canada_database
    - Collection Name – unemployment_canada

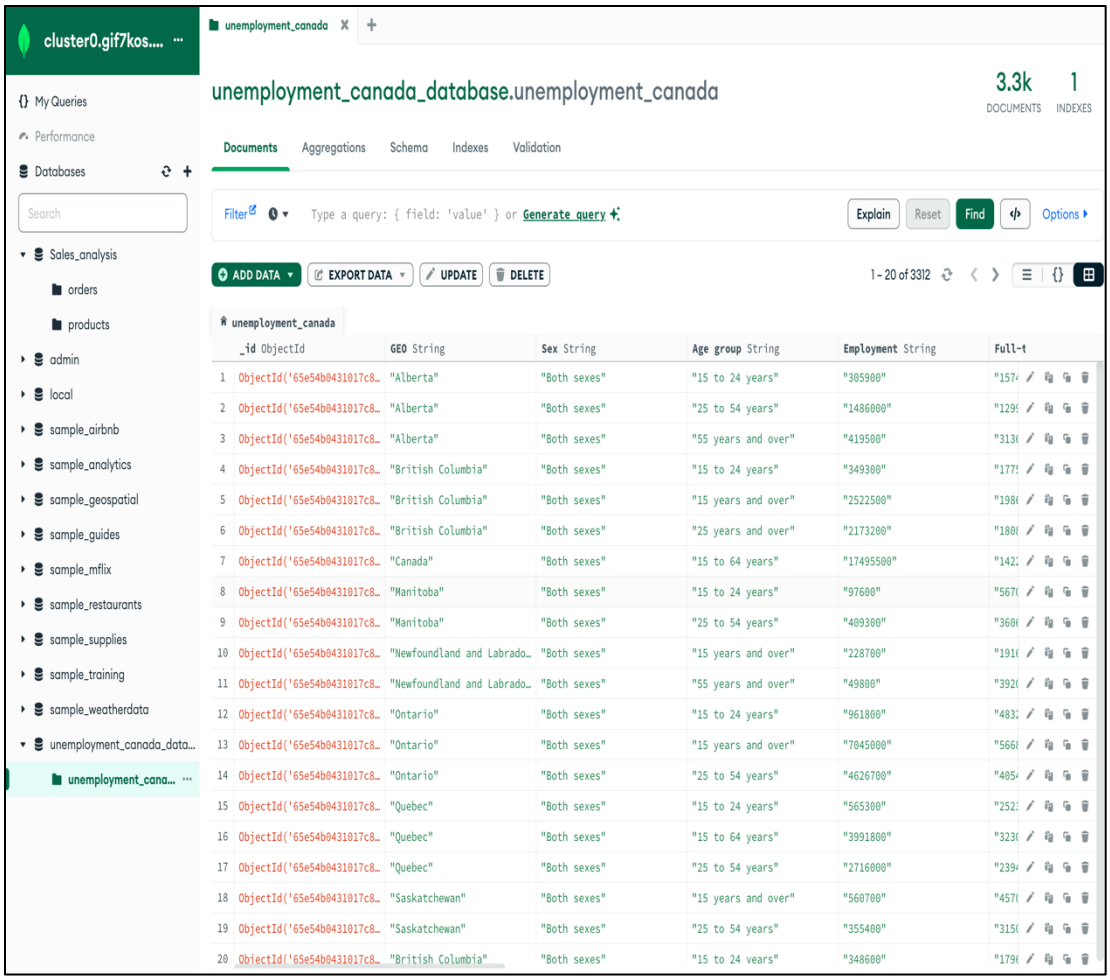

Figure 3 Unemployment Canada collections in MongoDB

- **Datasource4 & 5:** mental_health_act_apprehensions_open_data.csv and race_based_data.json are processed through the Snowflake-Alteryx-Azure SQL pipeline
    - Objectives:
        - Snowflake: Utilize Snowflake as a cloud-based data warehousing solution to store and query large volumes of data efficiently.
        - Alteryx: Employ Alteryx for data preparation, blending, and analytics, enabling the transformation of raw data into actionable insights.
        - Azure SQL: Use Azure SQL as a cloud-based relational database service for storing processed data, ensuring it is structured and ready for analysis.
        -
    - Configuration Steps:
        - Snowflake Setup:
            - Account Creation: Sign up for a Snowflake account and select the appropriate computing resources based on the project's data volume and processing needs.

- Warehouse Configuration: Create a data warehouse in Snowflake. This is where data will be loaded, transformed, and queried.
- Database and Schema Setup: Define a database and schema within Snowflake to organize the data effectively.

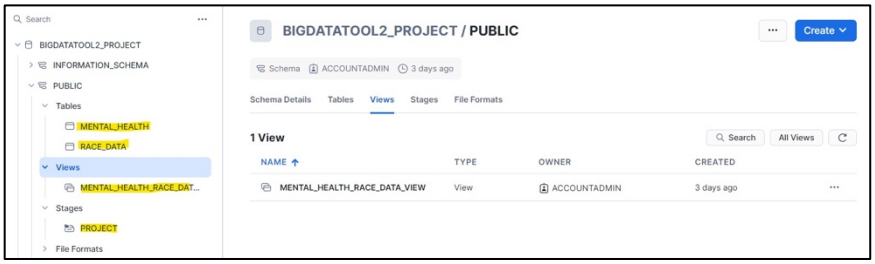

Figure 4 Combined View in Snowflake

- <u>Alteryx Integration:</u>
  - Install Alteryx Designer: Ensure Alteryx Designer is installed on your system to design, execute, and automate the data workflow.
  - Snowflake Connectivity: Use the Snowflake Input and Output tools in Alteryx to connect to the Snowflake database. This allows Alteryx workflows to read from and write to Snowflake.
  - Data Preparation and Transformation: Create Alteryx workflows to clean, transform, and prepare data for analysis. This includes joining datasets, filtering rows, and transforming columns.

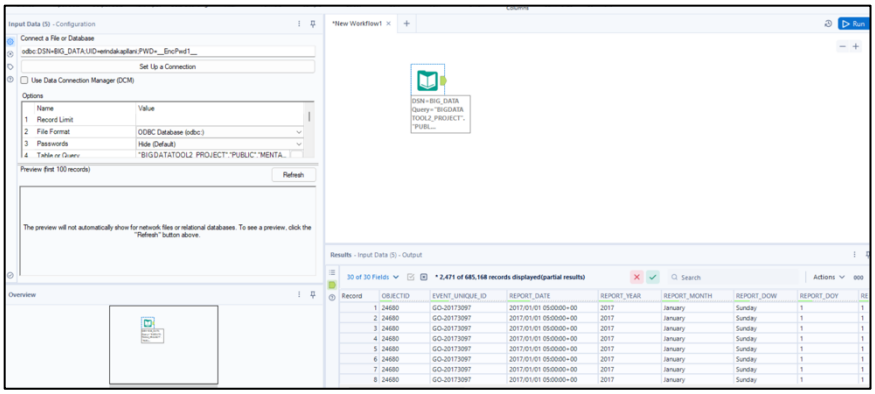

Figure 5 Snowflake's View Results displayed in Alteryx

- <u>Azure SQL Configuration</u>
  - Azure Subscription: Ensure you have an active Azure subscription. Create an Azure SQL Database instance through the Azure Portal.
  - Database Setup: Configure the Azure SQL database, defining the size, location, and performance settings as per the project requirements.

- Connectivity and Security: Set up the firewall rules and secure access to the Azure SQL Database to ensure data protection and privacy.

▪ Integration Workflow:

- Data Ingestion: Load raw data from various sources into Snowflake for initial storage and preliminary queries.
- Data Processing with Alteryx: Use Alteryx to pull data from Snowflake, perform complex transformations, and analytics.
- Loading Processed Data into Azure SQL: After processing in Alteryx, load the transformed data into Azure SQL for further analysis or visualization.

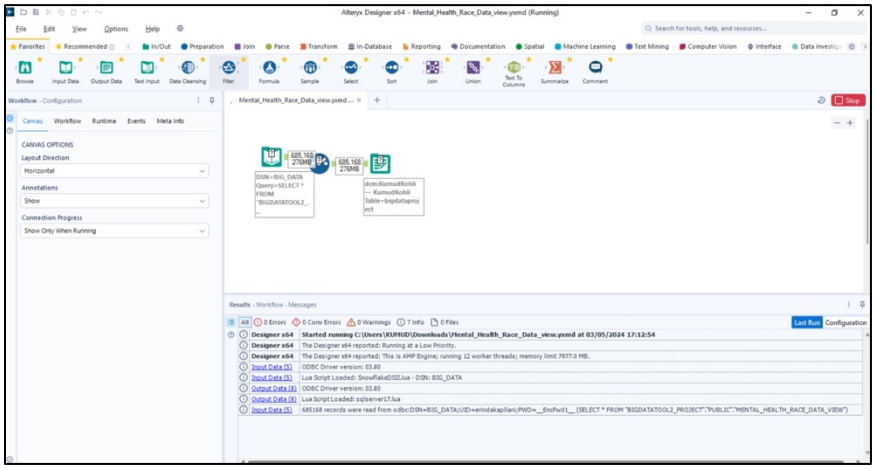

Figure 5: Workflow in Alteryx - Create an Output Table in Azure SQL

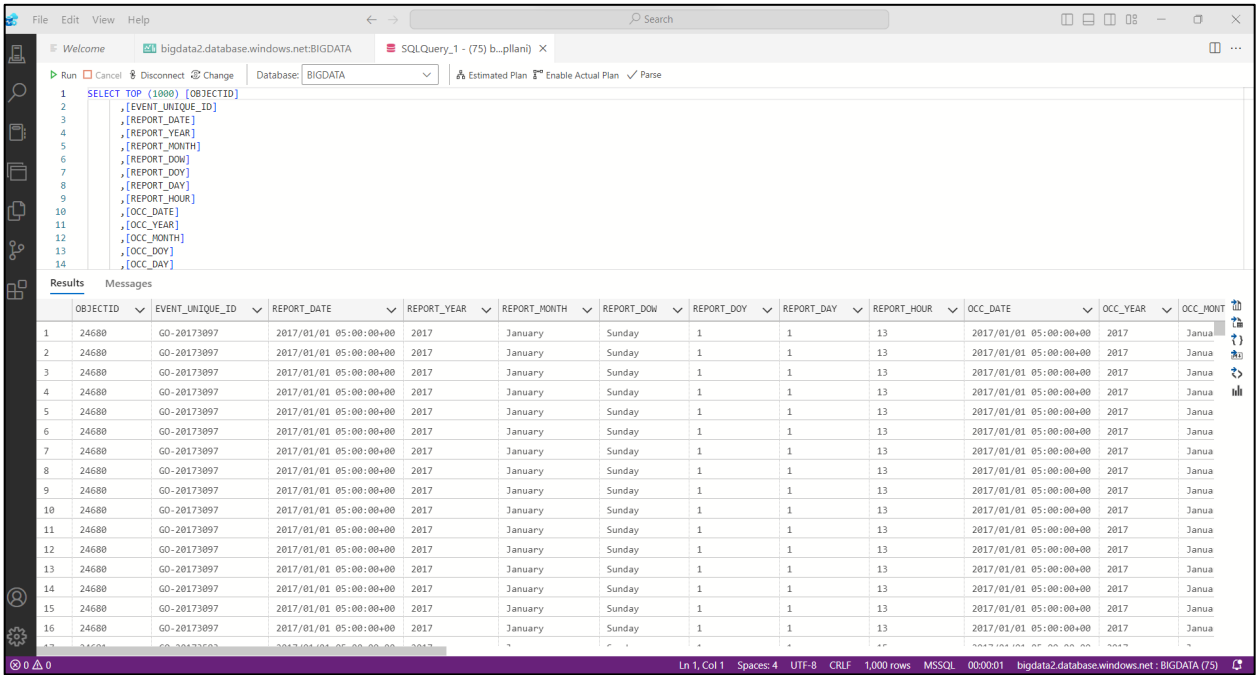

Figure 6 Mental Health Data (Datasource4) and Race Data (Datasource5) loaded in Azure SQL

- **Datasource6 & 7:** budget_by_command.csv and staffing_by_command.csv are channeled through the MS SQL Server Express-Alteryx-Azure SQL pipeline

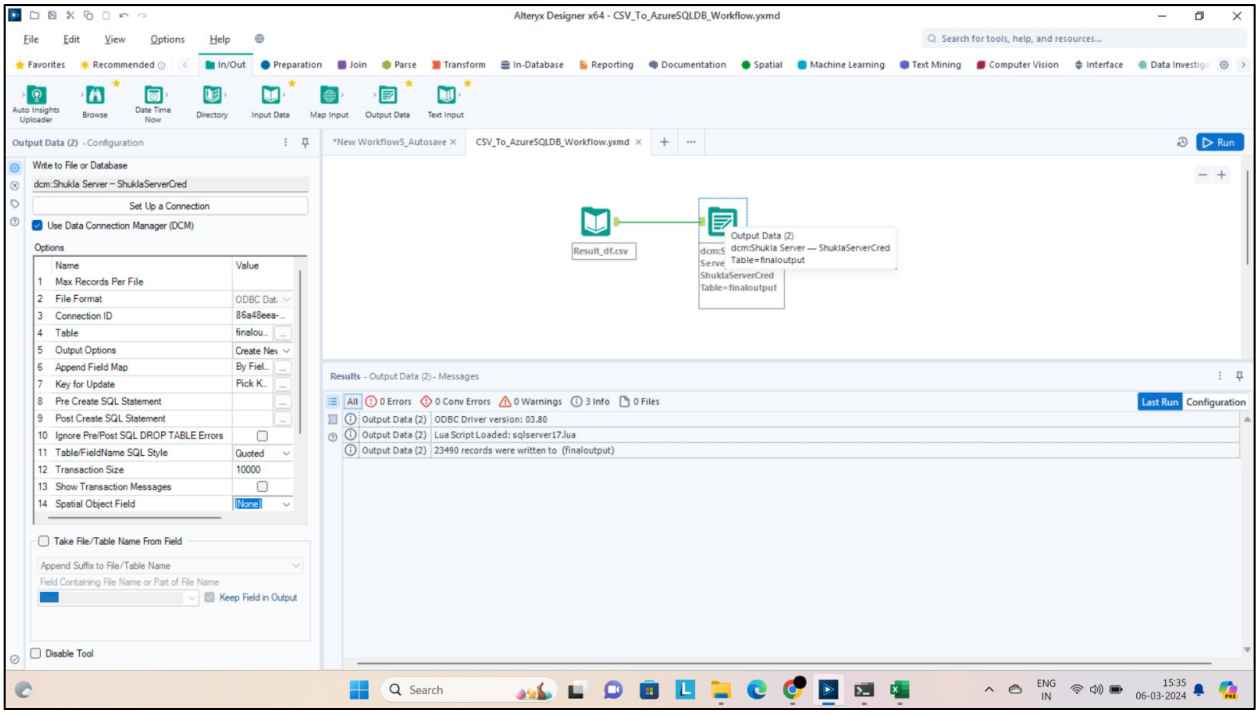

Figure 7 Budget by Command (Datasource6) and Staffing by Command (Datasource7) loaded into Azure SQL from Alteryx

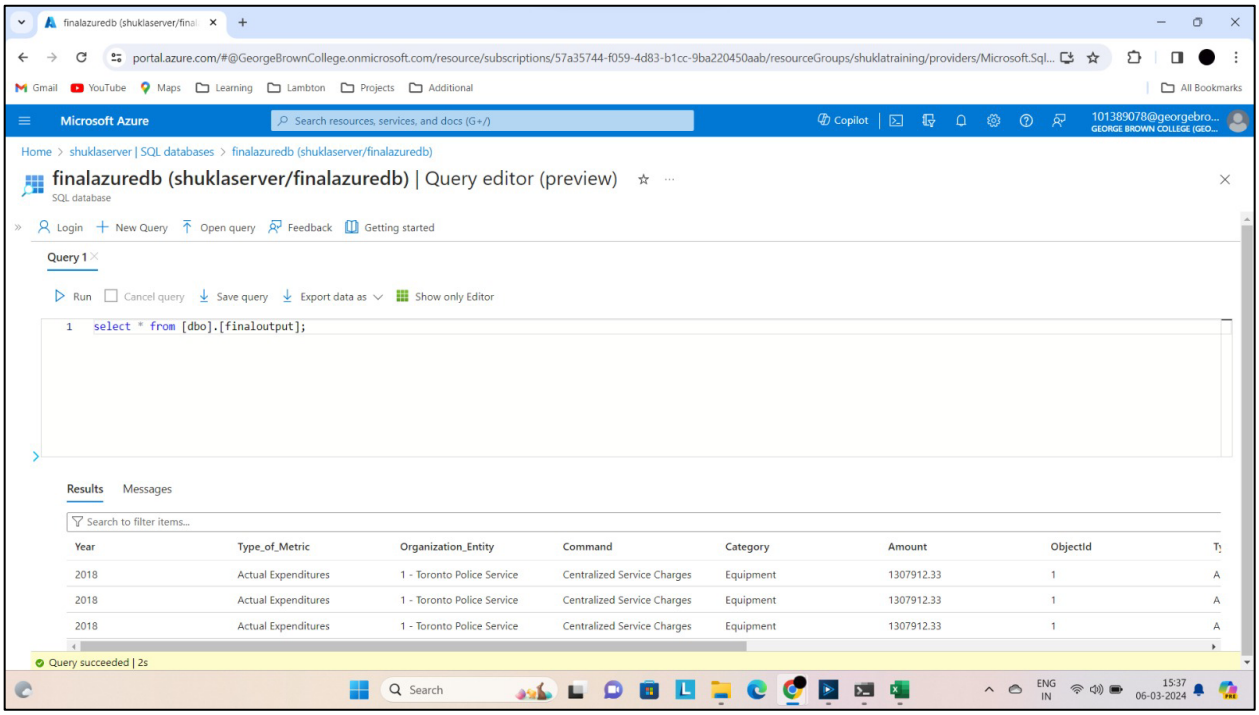

Figure 8 Staffing by Command (Datasource6) and Budget by Command (Datasource7) loaded in Azure SQL

- **Datasource8:** victims_of_crime.csv serves as additional data for various analyses

B. Data Ingestion and Storage:

- Hadoop acts as a scalable and distributed storage repository for crime-related data.
- MongoDB provides a flexible schema for dynamic data like unemployment rates.
- MariaDB offers reliable SQL storage for structured weather data.
- Azure SQL and Azure SQL Server Express serve as cloud-based SQL databases for efficient data management and accessibility.

C. Data Processing Engines:

- Snowflake offers a cloud-based data warehouse solution, which is paired with Alteryx for data preparation and blending.
- MS SQL Server Integration Services (SSIS) is utilized for orchestrating the ETL processes.
- Azure Analysis Services provides an enterprise-grade analytics engine.

D. Analysis and Machine Learning:

PySpark is utilized for both ETL processes and the development of machine learning models. Two advanced deep learning models are developed:

- Model 1: LSTM for Crime Prediction Based on Weather Data
- Model 2: CNN for Crime Prediction Based on Race Data

E. Data Visualization and Reporting:

PowerBI is utilized to transform the analyzed data into interactive and accessible reports. Batches executed for analysis are the following:

- Crime Rate vs. Weather Conditions Analysis: Utilizing Datasource1, Datasource2, and Datasource8.
- Economic Indicators and Crime Rates Analysis: Utilizing Datasource1, Datasource3, and Datasource8.
- Mental Health and Crime Rates Analysis: Utilizing Datasource1, Datasource4, and Datasource8.
- Race and Crime Rates Analysis: Utilizing Datasource1, Datasource5, and Datasource8.
- Law Enforcement Resource Allocation Efficiency Analysis: Utilizing Datasource1, Datasource6, and Datasource8.
- Staffing Impact on Law Enforcement Effectiveness Analysis: Utilizing Datasource1, Datasource7, and Datasource8.

IV. PySpark ETL Code

A. Configuration of Data Sources

MongoDB

Below are the configuration steps for integrating MongoDB with PySpark

```python
from pyspark.sql import SparkSession

# Create a SparkSession with MongoDB Spark connector package
spark = SparkSession.builder \
    .appName("Read MongoDB JSON") \
    .config("spark.mongodb.input.uri", "mongodb+srv://Pmk:desrufpmk@cluster0.gif7kos.mongodb.net/unemployment_canada_database.unemployment_canada") \
    .config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:3.0.0") \
    .getOrCreate()

# Read JSON data from MongoDB into a DataFrame
df = spark.read.format("com.mongodb.spark.sql.DefaultSource").load()

# Group by year and count records
result = df.groupBy("YEAR").count().orderBy("YEAR")

# Show the result
result.show()
```

Python

Figure 9 MongoDB Configuration

MariaDB

Below are the configuration steps for integrating MariaDB with PySpark

```python
from pyspark.sql import SparkSession

jar_path = "/opt/spark/jars/mariadb-java-client-3.3.2.jar"
print("Create a Spark Session -Mariadb ")
spark_mariadb = SparkSession.builder \
    .appName("PySpark Example") \
    .master("local") \
    .config("spark.jars", jar_path) \
    .config("spark.driver.extraClassPath", jar_path) \
    .config("spark.executor.extraClassPath", jar_path) \
    .getOrCreate()

server = "52.186.72.211"
port = 3306
database = "weather"
user = "azureuser"
password = "azureuser"
jdbc_url = f"jdbc:mysql://{server}:{port}/{database}?user={user}&password={password}&permitMysqlScheme"
jdbc_driver = "org.mariadb.jdbc.Driver"
properties = {
    "user": user,
    "password": password,
    "driver": jdbc_driver
}
```

Python

Figure 10 MariaDB Configuration

Azure SQL

Below are the configuration steps for integrating Azure SQL with PySpark

- **For Snowflake-Alteryx-Azure SQL pipeline**

```python
from pyspark.sql import SparkSession

# Create spark session
spark = SparkSession.builder \
    .appName("BigDataProject") \
    .config("spark.jars.packages", "com.microsoft.sqlserver:mssql-jdbc:7.4.1.jre8") \
    .getOrCreate()

# Defining the JDBC connection properties
jdbc_url = "jdbc:sqlserver://bigdata2.database.windows.net:1433;database=BIGDATA"
jdbc_properties = {
    "user": "erindakapllani",
    "password": "Bigdata123!",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

# Reading data from Azure SQL Server
df = spark.read.jdbc(url=jdbc_url, table="bigdataproject", properties=jdbc_properties)
```

Python

Figure 11 Azure SQL Configuration for Snowflake-Alteryx-Azure SQL pipeline

- **For MS SQL Server Express-Alteryx-Azure SQL pipeline**

```python
from pyspark.sql import SparkSession

# Initialize SparkSession
spark = SparkSession.builder \
    .appName("AzureSQLDataLoad") \
    .getOrCreate()

# Define JDBC connection properties
url = "jdbc:sqlserver://shuklaserver.database.windows.net:1433;database=finalazuredb"
properties = {
    "user": "azureuser",
    "password": "Ilove85workWonder69",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

# Read data from Azure SQL Database table
df = spark.read.jdbc(url=url, table="finaloutput", properties=properties)

# Show the loaded data
df.show()
```

Python

Figure 12 Azure SQL Configuration for MS SQL Server Express-Alteryx-Azure SQL pipeline

B. ETL - Data Extraction, Data Transformation and Data Loading

Since the team performed four (4) batches of PySpark-MS Azure SQL-MS SQL Server Integration Services-MS Azure SQL pipeline, below are the data extraction, data transformation and data loading to MS Azure SQL for each batch:

- **Crime Rate vs. Weather Conditions Analysis:**
  Utilizing Crime Data (Datasource1) from Hadoop and Weather Data (DataSource2) from MariaDB, joining via 'Year' column and saving the joined data to MS Azure SQL (tbl_crime_weather)

```python
from pyspark.sql import SparkSession

# Initialize SparkSession with MariaDB JDBC jar path
jar_path = "/opt/spark/jars/mariadb-java-client-3.3.2.jar"
spark = SparkSession.builder \
    .appName("PySpark ETL Example") \
    .master("local") \
    .config("spark.jars", jar_path) \
    .config("spark.driver.extraClassPath", jar_path) \
    .config("spark.executor.extraClassPath", jar_path) \
    .getOrCreate()

# Data Extraction from MariaDB
server = "52.186.72.211"
port = 3306
database = "weather"
user = "azureuser"
password = "azureuser"
jdbc_url = f"jdbc:mysql://{server}:{port}/{database}"
jdbc_driver = "org.mariadb.jdbc.Driver"
properties = {
    "user": user,
    "password": password,
    "driver": jdbc_driver
}

weather_df = spark.read.jdbc(url=jdbc_url, table="weather_data", properties=properties)

# Load data from Hadoop
crimes_df = spark.read.csv("/home/azureuser/crimedata/combined_break_enter_open_homicide_traffic_collisions_file.csv", header=True, inferSchema=True)
```

Python

Figure 13 Data Extraction of Crime Data (Datasource1) from Hadoop and Weather Data (Datasource2) from MariaDB

```python
# Join using common 'Year' column in both DataFrames
joined_df = crimes_df.join(weather_df, crimes_df.Year == weather_df.Year, "inner")

# Columns to retain
retain_columns = [
    'Year', 'OCC_MONTH', 'DIVISION', 'HOMICIDE_TYPE', 'NEIGHBOURHOOD_158',
    'COVID_YEAR', 'Day', 'Temp', 'Wind Spd ', 'Visibility'
]

# Drop columns not in the list of columns to retain
for col in joined_df.columns:
    if col not in retain_columns:
        joined_df = joined_df.drop(col)
```

Python

Figure 14 Joining Crime Data (Datasource1) and Weather Data (Datasource2) via common 'Year' column

```python
# Data Loading to Azure SQL Database
# Using provided Azure SQL database details
azure_jdbc_url = "jdbc:sqlserver://rashikaserver.database.windows.net:1433;database=myfirstdatabase"
azure_properties = {
    "user": "azureuser",
    "password": "Rashika19",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

# Writing the joined DataFrame to Azure SQL Database
joined_df.write.jdbc(url=azure_jdbc_url, table="tbl_crime_weather", mode="overwrite", properties=azure_properties)

# Stop the Spark session
spark.stop()
```

Python

Figure 15 Loading of the combined data to Azure SQL tbl_crime_weather table

- **Economic Indicators and Crime Rates Analysis**
  Utilizing Crime Data (Datasource1) from Hadoop and Unemployment Data (DataSource3) from MongoDB, joining via 'Year' column and saving the joined data to MS Azure SQL (tbl_crime_unemployment)

```python
from pyspark.sql import SparkSession

# Initialize SparkSession with MongoDB package
spark = SparkSession.builder \
    .appName("ETL Pipeline with MongoDB and Azure SQL") \
    .config("spark.mongodb.input.uri", "mongodb+srv://Pmk:desrufpmk@cluster0.gif7kos.mongodb.net/unemployment_canada_database.unemployment_canada") \
    .config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:3.0.0") \
    .getOrCreate()

# Data Extraction from Hadoop
crime_data = spark.read.csv("/home/azureuser/crimedata/combined_break_enter_open_homicide_traffic_collisions_file.csv", header=True, inferSchema=True)

# Data Extraction from MongoDB
unemployment_data = spark.read.format("com.mongodb.spark.sql.DefaultSource").load()
```

Python

Figure 16 Data Extraction of Crime Data (Datasource1) from Hadoop and Unemployment Data (Datasource3) from MongoDB

```python
# Specify the columns to retain after the join
columns_to_retain = [
    'Year', 'OCC_MONTH', 'DIVISION', 'HOMICIDE_TYPE', 'NEIGHBOURHOOD_158',
    'COVID_YEAR', 'GEO', 'Sex', 'Age group', 'Population',
    'Unemployment', 'Unemployment rate'
]

# Perform the join and select only the specified columns
joined_data = crime_data.join(unemployment_data, "Year", "inner").select(columns_to_retain)
```

Python

Figure 17 Joining Crime Data (Datasource1) and Unemployment Data (Datasource3) via common 'Year' column

```python
# Data Loading to Azure SQL
azure_jdbc_url = "jdbc:sqlserver://rashikaserver.database.windows.net:1433;database=myfirstdatabase"
azure_properties = {
    "user": "azureuser",
    "password": "Rashika19",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

# Write the joined DataFrame to Azure SQL Database
joined_data.write.jdbc(url=azure_jdbc_url, table="tbl_crime_unemployment", mode="overwrite", properties=azure_properties)

# Stop the Spark session
spark.stop()
```

Python

Figure 18 Loading of the combined data to Azure SQL tbl_crime_unemployment

- **Mental Health and Crime Rates Analysis <u>and</u> Race and Crime Rates Analysis**
  Utilizing Crime Data (Datasource1) from Hadoop and, Mental Health and Race Data (DataSource4 and 5) from MS Azure SQL, joining via 'Year' column and saving the joined data to MS Azure SQL (tbl_crime_mental_race)

```python
from pyspark.sql import SparkSession

# Initialize Spark session for Azure SQL JDBC
spark = SparkSession.builder \
    .appName("BigDataProject ETL") \
    .config("spark.jars.packages", "com.microsoft.sqlserver:mssql-jdbc:7.4.1.jre8") \
    .getOrCreate()

# Data Extraction from Hadoop (or local file system in this context)
crime_data = spark.read.csv("/home/azureuser/crimedata/combined_break_enter_open_homicide_traffic_collisions_file.csv", header=True, inferSchema=True)

# Data Extraction for Joined Mental Health and Race Data from Azure SQL
jdbc_url_read = "jdbc:sqlserver://bigdata2.database.windows.net:1433;database=BIGDATA"
read_properties = {
    "user": "erindakapllani",
    "password": "Bigdata123!",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}
```

Python

Figure 19 Data Extraction of Crime Data (Datasource1) from Hadoop and merged Mental Health Data (Datasource4) and Race Data (Datasource5) from MS Azure SQL

```python
# Loading MentalRaceJoinedData from Azure SQL Database
mental_race_joined_df = spark.read.jdbc(url=jdbc_url_read, table="MentalRaceJoinedData", properties=read_properties)

# Specify the columns to retain from both DataFrames, considering potential column name overlaps
columns_to_retain = [
    crime_data["Year"], crime_data["OCC_MONTH"], crime_data["DIVISION"],
    crime_data["HOMICIDE_TYPE"], crime_data["NEIGHBOURHOOD_158"],
    crime_data["COVID_YEAR"], mental_race_joined_df["PREMISES_TYPE"],
    mental_race_joined_df["APPREHENSION_TYPE"], mental_race_joined_df["SEX"].alias("SEX_MENTAL"),
    mental_race_joined_df["AGE_COHORT"], mental_race_joined_df["Type_of_Incident"],
    mental_race_joined_df["Perceived_Race_of_People_Involv"], mental_race_joined_df["Incident_Count"]
]

# Perform the join, resolving the 'Year' column ambiguity directly within the join condition
joined_df = crime_data.join(mental_race_joined_df, crime_data["Year"] == mental_race_joined_df["Year"], "inner") \
    .select([column for column in columns_to_retain])
```

Python

Figure 20 Joining Crime Data (Datasource1) and Mental Health Race Data (Datasource4 and 5) via common 'Year' column

```python
# Data Loading back to Azure SQL (for the transformed and joined data)
azure_jdbc_url = "jdbc:sqlserver://rashikaserver.database.windows.net:1433;database=myfirstdatabase"
azure_properties = {
    "user": "azureuser",
    "password": "Rashika19",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

# Writing the joined DataFrame to Azure SQL Database
joined_df.write.jdbc(url=azure_jdbc_url, table="tbl_crime_mental_health_race", mode="overwrite", properties=azure_properties)

# Stop Spark session
spark.stop()
```

Python

Figure 21 Loading of the combined data to Azure SQL tbl_crime_mental_health_race

- **Law Enforcement Resource Allocation Efficiency Analysis and Staffing Impact on Law Enforcement Effectiveness Analysis**
  Utilizing Crime Data (Datasource1) from Hadoop and, Staffing by Command and Budget by Command (DataSource6 and 7) from MS Azure SQL, joining via 'Year' column and saving the joined data to MS Azure SQL (tbl_crime_staffing_budget)

```python
from pyspark.sql import SparkSession

# Initialize Spark session for Azure SQL JDBC with necessary package for SQL Server
spark = SparkSession.builder \
    .appName("BigDataProject ETL") \
    .config("spark.jars.packages", "com.microsoft.sqlserver:mssql-jdbc:7.4.1.jre8") \
    .getOrCreate()

# Data Extraction from Hadoop (or local file system in this context)
crime_data = spark.read.csv("/home/azureuser/crimedata/combined_break_enter_open_homicide_traffic_collisions_file.csv", header=True

# Data Extraction for Joined Mental Health and Race Data from Azure SQL
# Adjusting to match the updated JDBC URL and properties formatting
jdbc_url_read = "jdbc:sqlserver://shuklaserver.database.windows.net:1433;database=finalazuredb"
read_properties = {
    "user": "azureuser",
    "password": "Ilove85workWonder69",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}
```

Python

Figure 22 Data Extraction of Crime Data (Datasource1) from Hadoop and merged Staffing by Command Data (Datasource6) and Budget By Command Data (Datasource7) from MS Azure SQL

```python
# Loading MentalRaceJoinedData from Azure SQL Database
budget_staffing_joined_df = spark.read.jdbc(url=jdbc_url_read, table="BudgetStaffingJoinedData", properties=read_properties)

# Define a list of column expressions to retain, handling potential name overlaps by specifying the DataFrame
columns_to_retain = [
    crime_data["Year"],
    crime_data["OCC_MONTH"],
    crime_data["DIVISION"],
    crime_data["HOMICIDE_TYPE"],
    crime_data["NEIGHBOURHOOD_158"],
    crime_data["COVID_YEAR"],
    budget_staffing_joined_df["Type_of_Metric"],
    budget_staffing_joined_df["Organization_Entity"],
    budget_staffing_joined_df["Command"],
    budget_staffing_joined_df["Category"],
    (variable) budget_staffing_joined_df: Any
    budget_staffing_joined_df["Command_Staffing"],
    budget_staffing_joined_df["Category_Staffing"],
    budget_staffing_joined_df["Count_"]
]

# Perform the join between crime_data and budget_staffing_joined_df on the "Year" column
joined_df = crime_data.join(
    budget_staffing_joined_df,
    crime_data.Year == budget_staffing_joined_df.Year,
    "inner"
)

# After joining, select only the columns specified in columns_to_retain
# To avoid ambiguity with the "Year" column or others, ensure each selected column is referenced with its DataFrame
joined_df = joined_df.select(columns_to_retain)
```

Python

Figure 23 Joining Crime Data (Datasource1) and Staffing Budget Data (Datasource6 and 7) via common 'Year' column

```python
# Data Loading back to Azure SQL (for the transformed and joined data)
azure_jdbc_url = "jdbc:sqlserver://rashikaserver.database.windows.net:1433;database=myfirstdatabase"
azure_properties = {
    "user": "azureuser",
    "password": "Rashika19",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

# Writing the joined DataFrame to Azure SQL Database
joined_df.write.jdbc(url=azure_jdbc_url, table="tbl_crime_staffing_budget", mode="overwrite", properties=azure_properties)

# Stop Spark session
spark.stop()
```

Python

Figure 24 Loading of the combined data to Azure SQL tbl_crime_staffing_budget

V. PySpark Deep Learning Code

A. Model 1: LSTM for Crime Prediction Based on Weather Data

The selection of Long Short-Term Memory (LSTM) networks for predicting crime occurrences based on weather data is grounded in LSTM's superior capability to process and analyze time-series data. Unlike traditional neural networks, LSTMs are designed to remember and utilize historical data over long sequences, making them exceptionally suitable for capturing the temporal relationships inherent in weather patterns and their potential impact on crime rates. This predictive modeling approach acknowledges the complex interplay between weather conditions and crime occurrences, leveraging LSTM's strengths in handling variable-length sequences and overcoming the vanishing gradient problem that often plagues standard RNNs.

Data Preparation and Model Implementation

- **Data Loading and Feature Engineering: The** process begins with loading a combined dataset of crime and weather data into a DataFrame, followed by the selection of relevant features (e.g., temperature, wind speed) and binary encoding of the target variable ('CRIME_TYPE'). This step is crucial for preparing the dataset for LSTM processing.

```python
df = pd.read_csv('/Users/roankathrinadimaculangan/Downloads/random_crime_weather_df_v3.0.csv')
```

✓ 0.3s Python

```python
df['Crime_Binary'] = df['CRIME_TYPE'].apply(lambda x: 1 if x == 'Traffic Collision' else 0)
target = df['Crime_Binary'].values
features = df[['Temp', 'Wind Spd', 'Visibility', 'Month_Weather', 'Year']]
```

✓ 0.0s Python

Figure 25 Identifying Features and Target

- **Preprocessing:** Numerical features are scaled, and categorical variables are one-hot encoded to ensure data uniformity and compatibility with LSTM input requirements. The data is then reshaped into a three-dimensional array to align with the LSTM model's expectations for samples, timesteps, and features.

```
# One-hot encode 'Month' and 'DIVISION' and scale other features
preprocessor = ColumnTransformer(
    transformers=[
        ('num', MinMaxScaler(feature_range=(0, 1)), ['Year', 'Temp', 'Wind Spd', 'Visibility']),
        ('cat', OneHotEncoder(), ['Month_Weather'])
    ])
```

Figure 26 Data Preprocessing using OneHotEncoder and MinMaxScaler

- **Model Architecture:** The LSTM model comprises layers tailored for sequential data processing, including dropout layers for regularization to mitigate overfitting. The model is compiled with a binary crossentropy loss function and Adam optimizer, emphasizing its role in binary classification tasks.

```
# Define the LSTM model
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(1, n_features)),
    Dropout(0.2),
    LSTM(50),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train_reshaped, y_train, epochs=20, batch_size=64, validation_data=(X_test_reshaped, y_test), verbose=2)
```

✓ 3m 27.3s Python

Figure 27 Model Training

Evaluation and Insights

```
# Evaluate the model
loss, accuracy = model.evaluate(X_test_reshaped, y_test, verbose=0)
print(f'Test Loss: {loss:.4f}, Test Accuracy: {accuracy:.4f}')
```

✓ 1.9s Python

```
Test Loss: nan, Test Accuracy: 0.6333
```

Figure 28 Accuracy Score of the Model

The LSTM model achieved an accuracy of 0.6333 in predicting crime occurrences based on weather data, highlighting its potential in understanding complex temporal patterns. There is room for improvement in the model's predictive capabilities. Enhancements could be made through advanced feature engineering, hyperparameter tuning, incorporating external data sources, and employing advanced LSTM architectures or regularization techniques. These refinements aim to improve the model's accuracy and underscore the impactful role of deep learning in crime prevention and public safety strategies.

B. Model 2: CNN for Crime Prediction Based on Race Data

The unconventional application of Convolutional Neural Networks (CNNs) to crime prediction based on race data is predicated on CNN's superior capability for feature extraction and pattern recognition within structured data. Despite CNNs being traditionally associated with image processing tasks, their adaptability allows for the extraction of local and hierarchical features from one-dimensional sequence data. This model leverages CNN's strengths to

analyze the complex interplay between crime occurrences and demographic factors, aiming to uncover underlying patterns that contribute to crime predictions.

Data Preparation and Model Implementation

- The dataset, incorporating variables such as 'Year', 'Month', 'Type_of_Incident', 'DIVISION', and 'Perceived_Race_of_People_Involv', undergoes rigorous preprocessing. Categorical variables are one-hot encoded, and numerical variables are scaled to ensure uniformity and optimality for neural network processing. This preprocessing phase transforms the dataset into a suitable format for CNN input, involving the reshaping of data into a three-dimensional array to fit the one-dimensional convolutional model architecture.

```python
# Features and target variable
features = df[['Year', 'Month', 'Type_of_Incident', 'DIVISION', 'Perceived_Race_of_People_Involv']]
target = df['CRIME_TYPE']
```

Python

Figure 29 Identifying Features and Target

```python
# Preprocessing: Encoding categorical variables and scaling numerical variables
categorical_features = ['Month', 'Type_of_Incident', 'Perceived_Race_of_People_Involv']
numerical_features = ['Year', 'DIVISION']

# One-hot encode categorical features
onehotencoder = OneHotEncoder()
categorical_encoded = onehotencoder.fit_transform(features[categorical_features]).toarray()

# Scale numerical features
scaler = StandardScaler()
numerical_scaled = scaler.fit_transform(features[numerical_features])

# Concatenate processed categorical and numerical features
processed_features = np.hstack((numerical_scaled, categorical_encoded))
# processed_features = np.hstack((categorical_encoded))

# Encode the target variable
target_encoded = OneHotEncoder(sparse=False).fit_transform(target.values.reshape(-1, 1))

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(processed_features, target_encoded, test_size=0.2, random_state=42)

# Reshape input data for the CNN
X_train_reshaped = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_reshaped = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
```

(parameter) random_state: Int | RandomState | None

Python

Figure 30 Data Preprocessing through OneHotEncoder and StandardScaler

- **CNN Architecture and Training:** The CNN model's architecture comprises a sequence of Conv1D and MaxPooling1D layers, designed specifically for one-dimensional data processing. This setup is followed by Flatten and Dense layers, culminating in a softmax output layer for multi-class classification. The model is compiled with an 'adam' optimizer and 'categorical_crossentropy' loss function, reflecting its classification task. Training is conducted over several epochs, with a validation split to monitor performance on unseen data.

```python
# Define the CNN model
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train_reshaped.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(y_train.shape[1], activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Fit the model on the reshaped training data
model.fit(X_train_reshaped, y_train, epochs=20, validation_split=0.2, verbose=2)
```

Python

Figure 31 Model Training

Evaluation and Insights

```python
# Evaluate the model
loss, accuracy = model.evaluate(X_test_reshaped, y_test)
print(f"Test Loss: {loss:.4f}, Test Accuracy: {accuracy:.4f}")
```

Python

```
2493/2493 [==============================] - 2s 920us/step - loss: 0.1750 - accuracy: 0.8916
Test Loss: 0.1750, Test Accuracy: 0.8916
```

Figure 32 Accuracy Score of the Model

Achieving high accuracy in predicting crime rates based on race, as demonstrated by the CNN model's 0.8916 accuracy score, highlights the model's ability to uncover significant patterns affecting crime dynamics. The selection and processing of features like 'Year', 'Month', 'Type_of_Incident', and 'Perceived_Race_of_People_Involv' are crucial for this success, underscoring the importance of comprehensive data and sophisticated feature engineering. This model's application could revolutionize crime prevention strategies, allowing for targeted interventions and resource allocation to at-risk communities. However, it's vital to employ such models ethically, ensuring they serve to enhance public safety without perpetuating biases.

VI. Data Visualization

A. MS Azure SQL Configuration

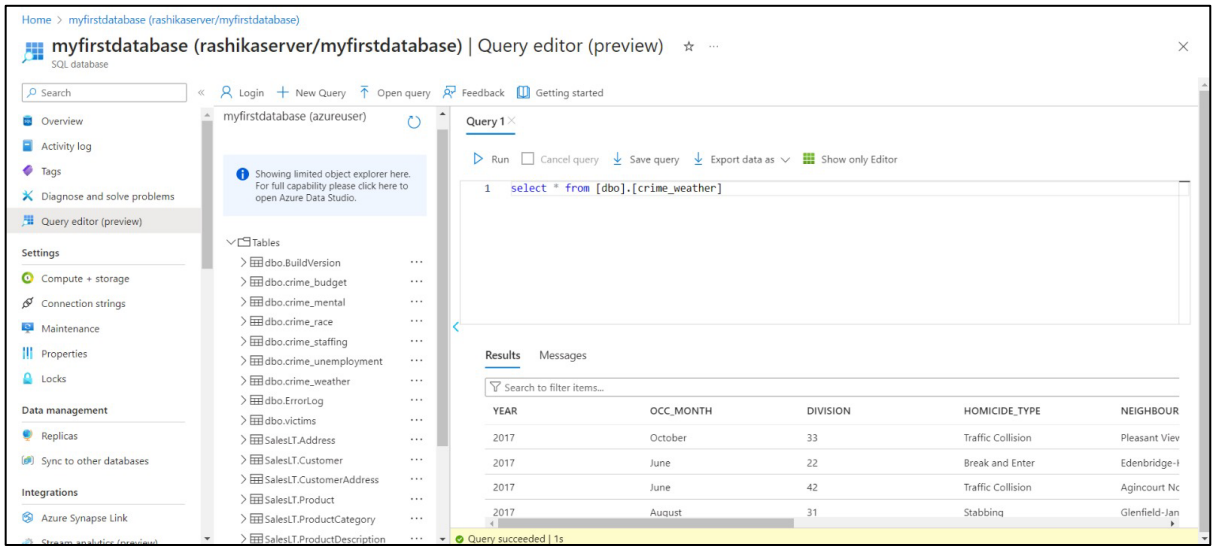

Figure 33 MS Azure SQL Configuration

B. Azure Analysis Services (SSAS)

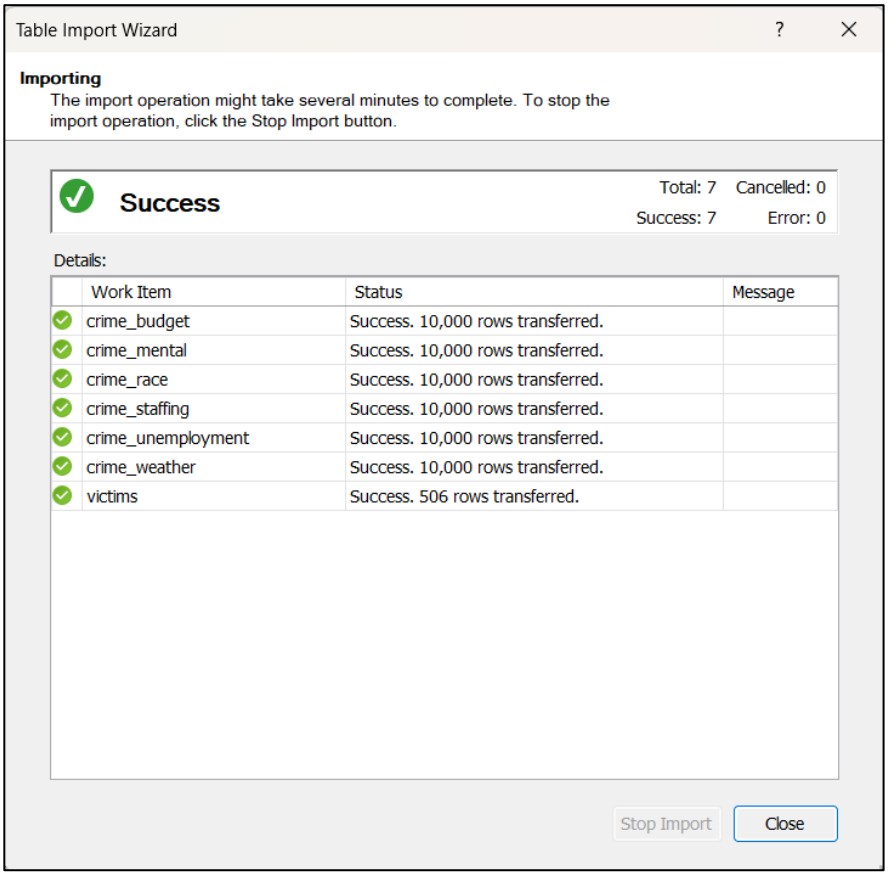

Figure 34 Loading data to SSAS

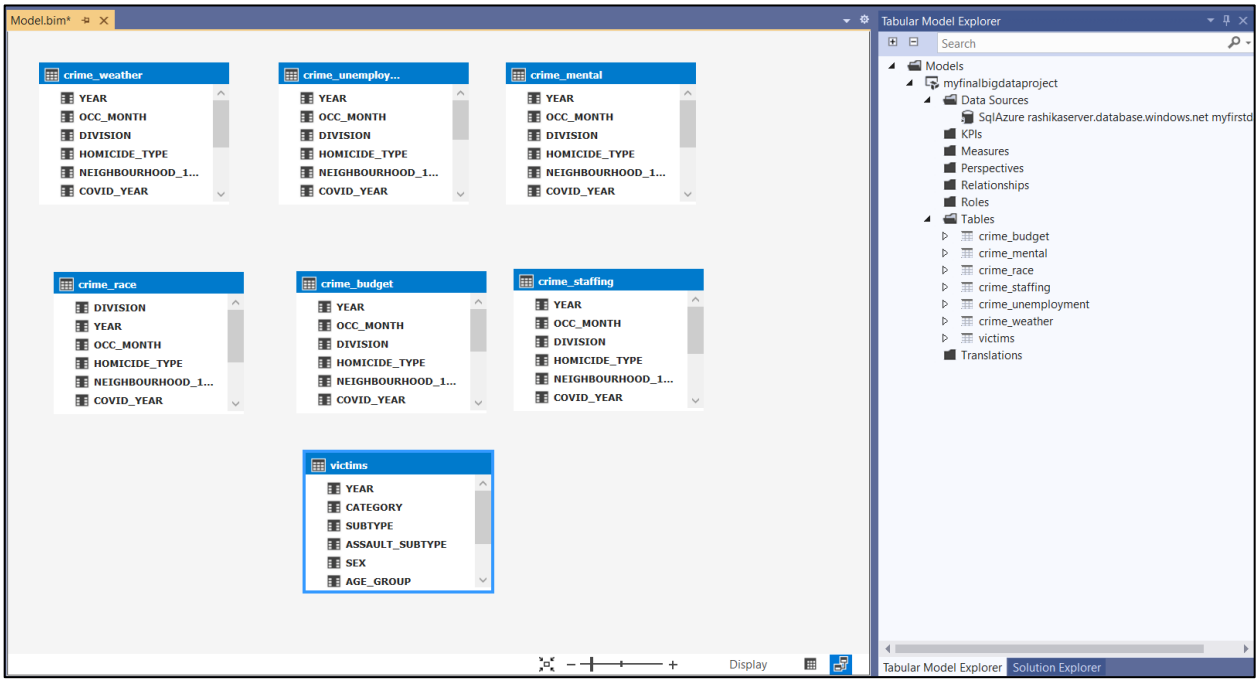

Figure 35 Fact and Dim Tables in SSAS (A)

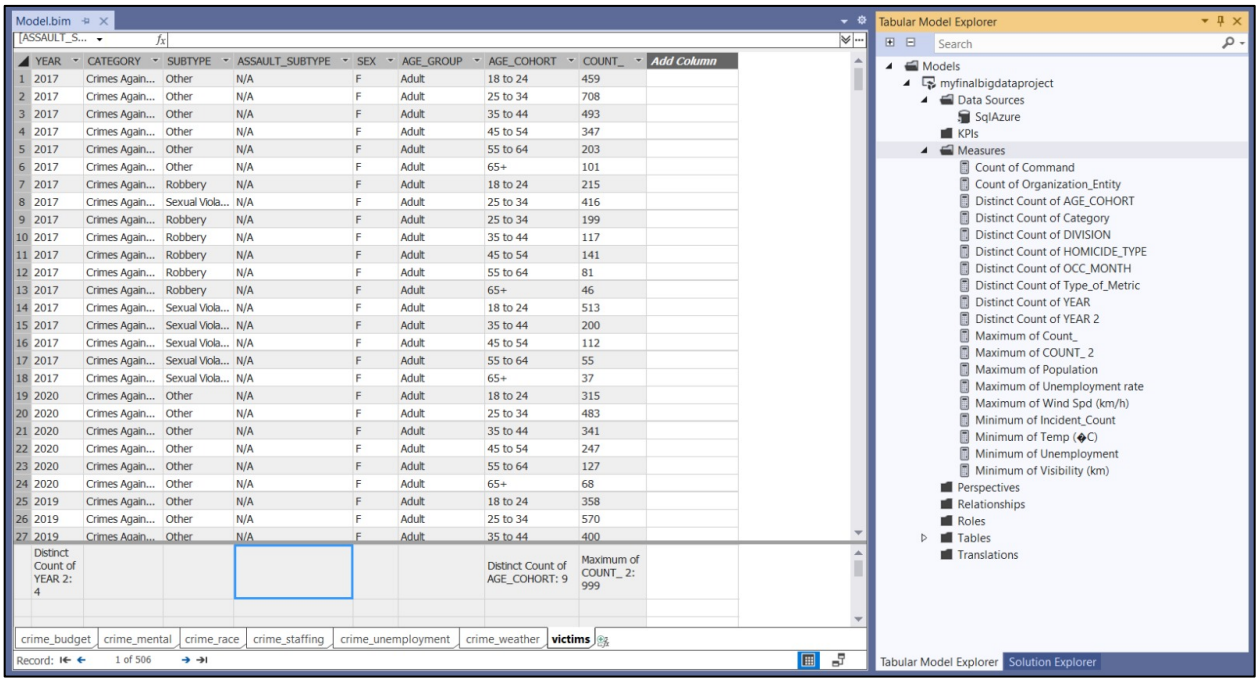

Figure 36 Fact and Dim Tables in SSAS (B)

A. Reporting (using PowerBI)

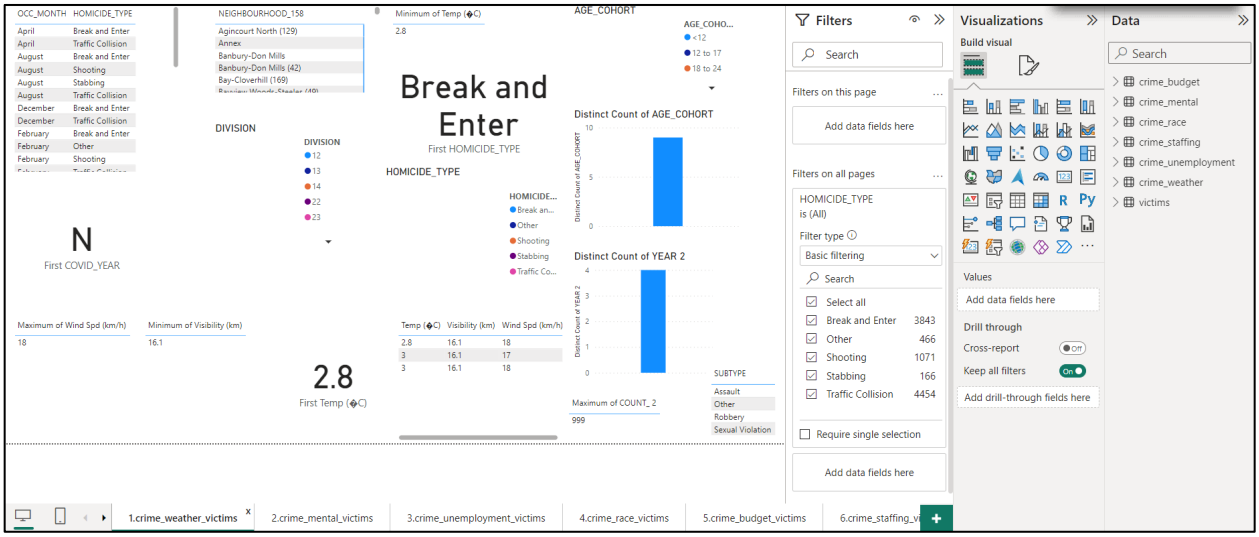

Figure 37 Crime Rates, Weather and Victims Data Analysis

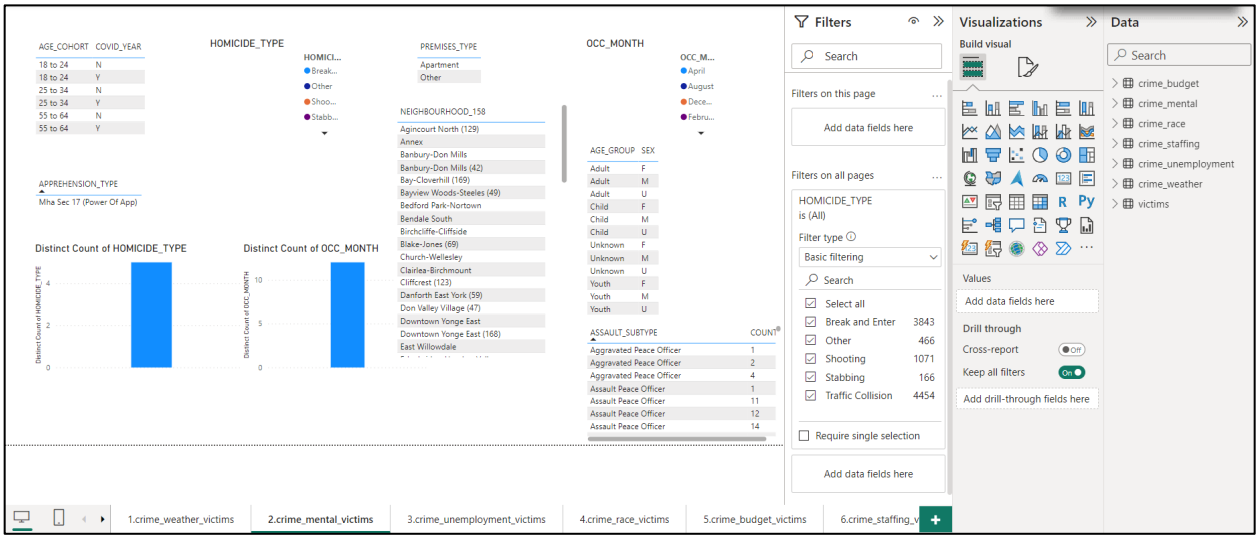

Figure 38 Crime Rates, Mental Health and Victims Data Analysis

Figure 39 Crime Rates, Unemploymnet and Victims Data Analysis

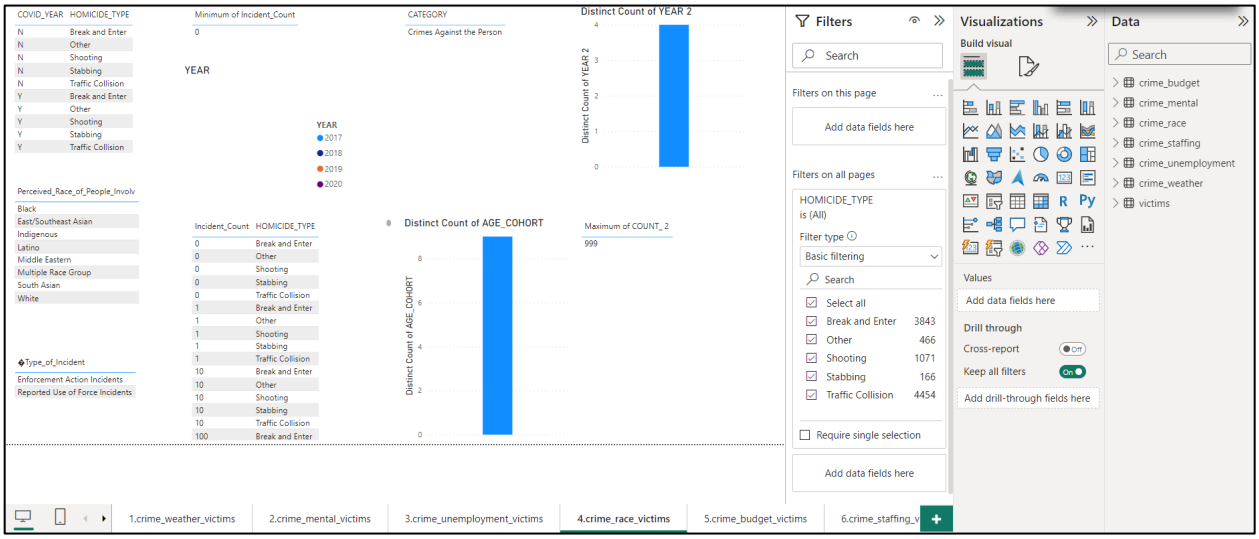

Figure 40 Crime Rates, Race and Victims Data Analysis

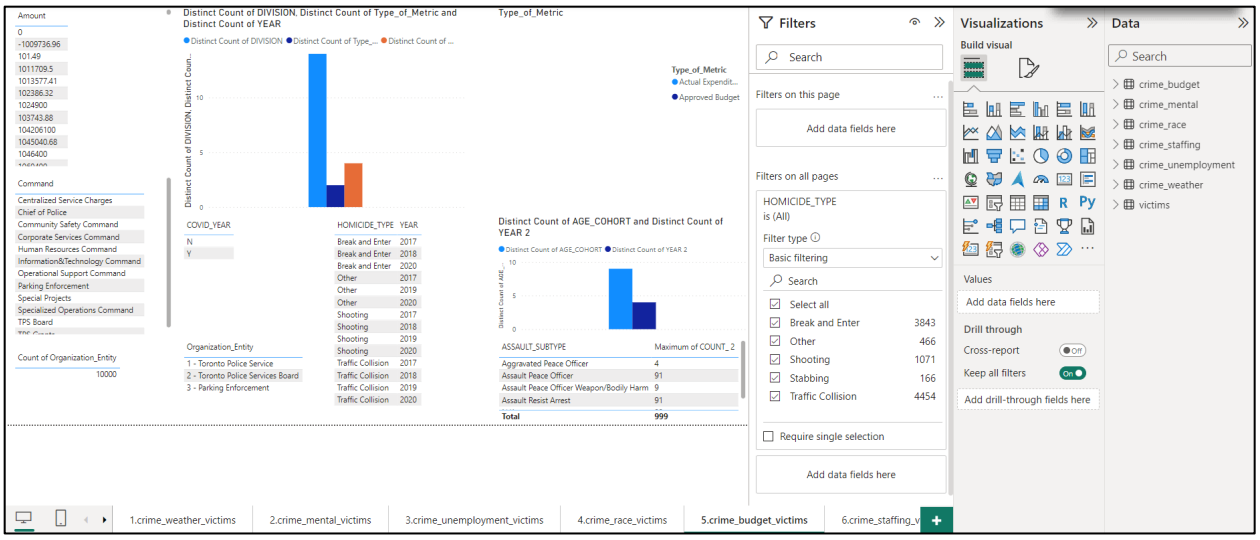

Figure 41 Crime Rates, Budget by Command and Victims Data Analysis

Figure 42 Crime Rates, Staffing By Command and Victims Data Analysis

VII. Analysis and Findings

- Based on the first report relating crime with weather and the affected people, following were the observations:
  1. April and December only have 2 homicide types, which are, Break and Enter & Traffic Collision.
  2. First Year in the Dataset is not a Covid Year.
  3. Minimum Temperature Recorded = 2.8°C
  4. Maximum Wind Speed = 18 km/h
  5. Minimum Visibility = 16.1 km
  6. Maximum Victim Count = 999
- Based on the second report relating crime with mental health and the affected people, there are 3 age groups in mental health data, namely, 18 to 24, 25 to 34 and 55 to 64

which implies that the age group 35 to 54 isn't affecting the crime rate due to mental health issues.

- Based on the third report relating crime with unemployment and the affected people, following were the observations:
  1. Maximum Number of People Contributing to Crime Rate based on Unemployment = 999600
  2. The maximum unemployment rate is 9.9%, which can be improved in the future if proper measures are taken.
  3. Minimum Number of People who are Unemployed = 10000
- Based on the fourth report relating crime with race and the affected people, there are 8 race groups in race-based data and 2 types of incidents caused. Also, the only category is crimes against the person which implies no crime against animals or environment is considered in the dataset and hence the dataset can be improvised considering crimes against animals or environment. Moreover, it is good to observe that the incident count is 0 as well for certain records.
- Based on the fifth report relating crime with allocated budget and the affected people, following were the observations:
  1. Year 2019 doesn't have Break and Enter Crime Type and Year 2018 doesn't have Other Crime Type.
  2. There are 3 organizations, namely, Toronto Police Services, Toronto Police Services Board and Parking Enforcement which are allocated budget for crimes.
  3. Two metrics used to allocate the budget for crimes are actual expenditures and approved budget.
  4. Maximum count of victims under assault subtype Assault Peace Officer is 91.
- Based on the sixth report relating crime with staffing and the affected people, following were the observations:
  1. There are 2 categories of staffing, namely, Civilian and Uniform and, 10 commands of staffing.
  2. Two metrics used to allocate the staff for crimes are actual staffing and approved staffing.
  3. The maximum count of staffing is 990 and the maximum count of victims is 999.