Milestone 1

0. Notebooks Description

Internal Data:

- `motive_basic_eda.ipynb` for any basic data exploratory that is Motive related or driving related; this serves as information points for the table definition and columns' description
- `motive_data_preprocessing.ipynb`: This is used to aggregated all the tables from Motive, `data_inspections_w_vehicle_map`, `data_driving_periods_w_vehicle_map`, `data_combined_events_w_vehicle_map`, `data_idle_events_w_vehicle_map`
  External Data:
- `state_crash_data_processing.ipynb`: Pre processing Data Exploratory for the `State Crash Data`. Aggregates data by crash year, month, state, and county to create a monthly county-level dataset and calculates number of crashes, total fatalities, total injuries, and total vhichles involved in crashes in each county each month .
- `accident_data_location_processing.ipynb` - Pre processing of `accident_record` and extracting relevant location information
- `weather_data_processing.ipynb` - Pre processing EDA for Precipitation Data
- `mapping_external_location_data_to_accidents.ipynb` - Joining in external datasets to accident data set using time and location variables`

1. Data Dictionary

`monthly_claims_insurance_driverid.csv`

The `Monthly Claims Report` table contains **major accident claims data**, including details about the policy, claim processing timeline, responsible adjusters, financials, and loss specifics. Each row represents **an individual claim** reported for a major accident within a given month. These records range from Jan 2023- March 2025

Column Name	Data Type	Description
Year of Policy Effect	`INTEGER`	The year in which the policy was issued or became effective.
Loss Date	`DATE`	The date when the accident or loss event occurred.
Reported Date	`DATE`	The date the claim was officially reported to the insurance company.

Column Name	Data Type	Description
Lag Time	`INTEGER`	The number of days between `Loss Date` and `Reported Date`.
Client Name	`VARCHAR`	The name of the insured entity (individual or company).
Claim Number	`VARCHAR`	A unique identifier assigned to the claim.
Adjuster	`VARCHAR`	The name or ID of the insurance adjuster handling the claim.
Loss Location	`VARCHAR`	The specific location where the accident occurred.
State Code	`CHAR(2)`	The two-letter state abbreviation (e.g., `CA` for California).
Claims Line	`VARCHAR`	The type of insurance claim (`Commercial Auto`).
Loss Cause	`VARCHAR`	General cause of loss (`Auto Accident` , `Fire` , `Theft` , etc.).
Loss Cause Detail	`VARCHAR`	Additional details specifying the cause of loss.
Current Claim Status	`VARCHAR`	The latest status of the claim (`Open` , `Closed` , `Pending`).
Claim Current Close	`DATE`	The date when the claim was officially closed.
Insured Driver	`VARCHAR`	The name of the insured driver involved in the accident.
Loss Description	`TEXT`	Detailed summary of the accident or loss event.
Gross Paid	`DECIMAL(15,2)`	Total amount paid by the insurer before recoveries.
Loss Reserve	`DECIMAL(15,2)`	Estimated amount set aside for future claim payments.
Recoveries	`DECIMAL(15,2)`	Amount recovered from subrogation, salvage, or other sources.
Net Incurred	`DECIMAL(15,2)`	The net cost of the claim (`Gross Paid + Loss Reserve - Recoveries`).
driver_id	`INTEGER`	The associated driver ID with Motive App

`accident_register_combined_driverid.csv

The `accident_records` table documents **all minor accident incidents** that occurred in **2025, 2024, and 2023**. It captures details such as **location, severity, involved personnel, claims, and financial impact**. These records range from Jan 2023- March 2025

- **Each row represents a single accident report** recorded within the given timeframe.
- **Primary Key:** `Claim Number` (unique identifier for each claim).

Column Name	Data Type	Description
Location of Accidents	`VARCHAR`	The physical location where the accident occurred (e.g., city, site, or address).
Region	`VARCHAR`	The regional classification associated with the accident location.
DBA	`VARCHAR`	"Doing Business As" name for the business entity associated with the accident.
General Manager	`VARCHAR`	The name of the General Manager overseeing the location where the accident occurred.
Date of Accident	`DATE`	The date (`YYYY-MM-DD`) when the accident took place.
Time of Accident	`TIME`	The exact time when the accident occurred.
Date Reported	`DATE`	The date (`YYYY-MM-DD`) when the accident was reported.
Fatality	`BOOLEAN`	Indicates whether the accident resulted in a fatality (`true` = yes, `false` = no).
Driver Name	`VARCHAR`	The name of the driver involved in the accident.
Copy of State or Insurance Report	`BOOLEAN`	Indicates whether a state or insurance report is available (`true` = yes, `false` = no).
Severity	`VARCHAR`	The severity level of the accident (e.g., `Minor` , `Moderate` , `Severe`).
Summary of Accident	`TEXT`	A brief description of how the accident occurred and contributing factors.
Claim Number	`VARCHAR`	Unique identifier assigned to the claim related to the accident.
Total Incurred	`DECIMAL(15,2)`	The total financial cost associated with the accident (e.g., damages, medical expenses).
Notes	`TEXT`	Additional details or remarks about the accident.
APMM Recordable	`BOOLEAN`	Indicates whether the accident is recordable under APMM safety compliance (`true` = yes, `false` = no).
driver_id	`INTEGER`	The associated driver ID with Motive App

`data_inspections_w_vehicle_map`

The `data_inspections_w_vehicle_map` table records truck inspections at the beginning of the day/ before trips and post-trip to ensure compliance with safety checks. These records range from Jan 2023- March 2025

- Each row represents an individual inspection log.
- **Primary Key:** `inspection_id` (unique identifier for each inspection record).
- **Unique Constraint:** `(vehicle_id, date, inspection_type)`, ensuring a vehicle is inspected only once per type per day.
- **Duplicate Handling:** Drivers may accidentally log multiple inspections on the same day, or omit `inspection_type`.

Column Name	Data Type	Description
id	`INTEGER`	Auto-incremented unique row identifier (optional primary key alternative).
inspection_id	`INTEGER`	Unique identifier for the inspection log.
vehicle_id	`INTEGER`	Unique ID assigned to the vehicle being inspected.
date	`DATE`	The date when the inspection occurred (`YYYY-MM-DD`).
location	`VARCHAR`	The physical location where the inspection took place.
status	`VARCHAR`	The inspection status (`acceptable`, `open`, `resolved`).
inspection_type	`VARCHAR`	Type of inspection performed (`pre-trip`, `post-trip`). May be missing in some cases.
driver_id	`INTEGER`	The unique identifier of the driver performing the inspection.
mechanic_id	`INTEGER`	Mechanic's ID, if the inspection required mechanic review.
reviewer_id	`INTEGER`	ID of the reviewer who validates or approves the inspection.
number	`VARCHAR`	The vehicle's unique number in the company's internal tracking system.
status-2	`VARCHAR`	The vehicle's operational status (`active`, `deactivated`).
make	`VARCHAR`	The manufacturer or main company of the vehicle.
model	`VARCHAR`	The specific model of the vehicle.

Notes

- **Each inspection is linked to a `vehicle_id` and `date`, ensuring tracking of daily vehicle inspections.**
- **Duplicate inspections may occur** due to drivers mistakenly logging multiple entries. Data validation should account for this.

- **Inspection type (`pre-trip` , `post-trip`) may be missing**, requiring data cleansing or defaulting.
- **Vehicles remain active (`status-2`) unless deactivated**, indicating whether the truck is still in service.
- **Foreign keys (`driver_id` , `mechanic_id` , `reviewer_id`)** should link to the respective user/employee database for consistency.
- This process is not rigorously being implemented; therefore, a lot of missing information

`data_idle_events_w_vehicle_map`

The `idle_events_log` table records idle events whenever a truck is idling during a trip, within the driver's clock-in and clock-out hours. These records range from Jan 2023- March 2025

- Each row represents a single idle event, capturing start time, end time, and duration.
- Idle events are logged per `event_id` whenever a truck is idling during a trip within clock-in/out hours.
- **Primary Key:** `event_id` (unique identifier for each idle event).

Column Name	Data Type	Description
id	`INTEGER`	Auto-incremented unique row identifier (optional).
event_id	`INTEGER`	Unique identifier for each idle event (Primary Key).
start_time	`TIMESTAMP`	The timestamp when idling began.
end_time	`TIMESTAMP`	The timestamp when idling ended.
driver_company_id	`INTEGER`	The driver’s company ID (from Motive's system).
driver_id	`INTEGER`	Unique identifier for the driver.
minutes_idling	`INTEGER`	Total duration (in minutes) the truck remained idle or the driver was resting.
number	`VARCHAR`	The vehicle's unique number in the company’s internal tracking system.
status-2	`VARCHAR`	The vehicle’s operational status (`active` , `deactivated`).
make	`VARCHAR`	The manufacturer or main company of the vehicle.
model	`VARCHAR`	The specific model of the vehicle.

`data_driving_periods_w_vehicle_map`

The `data_driving_periods_w_vehicle_map` table tracks driving sessions for truck drivers, linking them to vehicles and recording trip details such as distance, timestamps, and driving type. These records range from Jan 2023- March 2025

- Each row represents **a single driving period** (a continuous driving session).
- **Primary Key:** `event_id` (unique identifier for each driving period).
- **Unassigned Trips:** If a driver does not log into the Motive mobile app, the trip is recorded under `"unassigned"`.
- **Trip Classification:** The column `driving_period_type` categorizes trips as `driving`, `pc` (Personal Conveyance), or `ym` (Yard Move).

Column Name	Data Type	Description
id	`INTEGER`	Auto-incremented unique row identifier (optional).
event_id	`INTEGER`	Unique identifier for each driving period (Primary Key).
driver_id	`INTEGER`	Unique identifier for the driver. If unassigned, marked as `"unassigned"`.
driver_first_name	`VARCHAR`	The first name of the driver (if signed in).
driver_last_name	`VARCHAR`	The last name of the driver (if signed in).
vehicle_id	`INTEGER`	Unique identifier of the vehicle used in the driving session.
start_date	`TIMESTAMP`	Start time of the driving period.
end_date	`TIMESTAMP`	End time of the driving period.
driving_distance	`DECIMAL(10,2)`	Distance driven in miles or kilometers (depending on system settings).
driving_period_type	`VARCHAR`	Categorization of trip type: `driving`, `pc` (Personal Conveyance), or `ym` (Yard Move).
driver_company_id	`INTEGER`	The driver's company ID from Motive's system.
minutes_driving	`INTEGER`	Total time (in minutes) spent driving during the session.
month	`INTEGER`	The month (same as `start_date`) when the trip started.
created_at	`TIMESTAMP`	Timestamp when the driving period record was created.
updated_at	`TIMESTAMP`	Timestamp when the record was last updated.
unassigned	`BOOLEAN`	Indicates if the driver was logged in (`false`) or if the trip was recorded under `"unassigned"`

Column Name	Data Type	Description
		(`true`).
not_current	`BOOLEAN`	Indicates whether the record is outdated (`true`) or active (`null`).
number	`VARCHAR`	The vehicle's unique number in the company’s tracking system.
status-2	`VARCHAR`	The vehicle’s operational status (`active`, `deactivated`).
make	`VARCHAR`	The manufacturer or main company of the vehicle.
model	`VARCHAR`	The specific model of the vehicle.

`data_combined_events_w_vehicle_map`

The `data_combined_events_w_vehicle_map` table consolidates **safety and operational events** detected by the Motive system. This includes **hazard events, safety violations, coaching interventions, vehicle information, and status tracking**. These records range from Jan 2023- March 2025

- **Each row represents a recorded event** associated with a driver and vehicle.
- **Primary Key:** `event_id` (unique identifier for each event).
- **Contains both driving behavior events and vehicle-specific information**.

Column Name	Data Type	Description
id	`INTEGER`	Auto-incremented unique row identifier (optional).
event_id	`INTEGER`	Unique identifier for each event (Primary Key).
type	`VARCHAR`	The type of event recorded. One of: `camera_obstruction`, `cell_phone`, `crash`, `distraction`, `driver_facing_cam_obstruction`, `drowsiness`, `forward_collision_warning`, `manual_event`, `near_miss`, `ran_a_red_light`, `road_facing_cam_obstruction`, `seat_belt_violation`, `speeding`, `stop_sign_violation`, `tailgating`, `unsafe_lane_change`.

Column Name	Data Type	Description
driver_id	`INTEGER`	Unique identifier for the driver involved in the event.
driver_first_name	`VARCHAR`	The first name of the driver.
driver_last_name	`VARCHAR`	The last name of the driver.
vehicle_id	`INTEGER`	Unique identifier of the vehicle associated with the event.
coaching_status	`VARCHAR`	The coaching review status for the event. Options: `Pending Review`, `Coachable`, `Coached`, `Uncoachable`.
start_date	`TIMESTAMP`	The time when the driver start driving
severity	`VARCHAR`	The severity level of the event (`Low`, `Moderate`, `Severe`).
group_id	`INTEGER`	The group (e.g., department or fleet) associated with the driver.
group_name	`VARCHAR`	The name of the driver's assigned group.
month	`INTEGER`	The month (same as `start_date`) when the event occurred.
created_at	`TIMESTAMP`	Timestamp when the event record was created.
updated_at	`TIMESTAMP`	Timestamp when the record was last updated (typically when `coaching_status` changes).
max_over_speed_in_kph	`DECIMAL(5,2)`	Maximum recorded speed over the limit (in kilometers per hour) for speeding violations.
max_over_speed_in_mph	`DECIMAL(5,2)`	Maximum recorded speed over the limit (in miles per hour) for speeding violations.
number	`VARCHAR`	The vehicle's unique number in the company's tracking system.
status-2	`VARCHAR`	The vehicle's operational status (`active`, `deactivated`).
make	`VARCHAR`	The manufacturer or main company of the vehicle.
model	`VARCHAR`	The specific model of the vehicle.

`full_driver_details.csv`

The `truck_driver_records` table contains a **list of active and deactivated truck drivers**, along with their licensing, activity status, and device usage details.

- Each row represents a single driver.
- Primary Key: `Driver ID` (unique identifier for each driver).
- Tracks driver status (`active` or `deactivated`).

Column Name	Data Type	Description
Driver	`VARCHAR`	Full name of the truck driver.
Driver ID	`INTEGER`	Unique identifier assigned to the driver (Primary Key).
License State	`CHAR(2)`	The state where the driver's license was issued (e.g., `CA`, `TX`).
Groups	`VARCHAR`	The group(s) or fleet(s) to which the driver is assigned.
Driver Type	`VARCHAR`	Indicates whether the driver follows `ELD` (Electronic Logging Device) regulations or is classified as `Logs Not Required`.
Vehicle Gateway Enabled	`BOOLEAN`	Indicates whether a vehicle gateway is enabled (`true` = Yes, `false` = No).
Last Activity	`TIMESTAMP`	The last recorded activity of the driver (e.g., last login, last recorded trip).
Recent Vehicle	`VARCHAR`	The vehicle ID or description of the last vehicle assigned to the driver.
Vehicle Gateway S/N	`VARCHAR`	The serial number of the vehicle's logging gateway (if applicable).
Last Drive Time	`TIMESTAMP`	The timestamp of the last recorded driving session.
Cycle Rule	`VARCHAR`	The regulatory cycle rule the driver follows (e.g., `US 70-hour/8-day`, `Canada South 70-hour/7-day`).
Device Platform	`VARCHAR`	The type of mobile platform the driver uses for logging (e.g., `iOS`, `Android`).
App Version	`VARCHAR`	The version of the Motive app installed on the driver's device.
App Up To Date	`BOOLEAN`	Indicates whether the driver's app is updated to the latest version (`true` = Yes, `false` = No).
Time Zone	`VARCHAR`	The driver's configured time zone (e.g., `UTC-5`, `PST`).
driver_status	`VARCHAR`	The current status of the driver: `active` or `deactivated`.

External Data

State Crash Data

- There is one .xlsx for each state with the following sheets:
    - GENERAL DEFINITIONS
    - MCIS COLUMN NAMES
    - FARS COLUMN NAMES
    - FARS & MCMIS COMMON NAMES
    - {State Code} 2018-2025 MCMIS
        - {State Code} {Year} MCMIS
    - {State Code} 2018-2025 FARS
        - {State Code} {Year} FARS
- We use the Motor Carrier Management Information System (MCMIS) defined as:
    - "The MCMIS Crash data includes crashes that are reported by states to the FMCSA through the SAFETYNET computer reporting system. The Crash File includes the National Governors' Association (NGA) recommended data elements collected on trucks and buses involved in crashes that meet the NGA recommended crash threshold. A State reportable crash must involve a truck (a vehicle designed, used, or maintained primarily for carrying property, with a gross vehicle weight rating or gross combination weight rating of more than 10,000 lbs.) or bus (a vehicle with seats for at least nine people, including the driver). The crash must result in at least one fatality; one injury where the person injured is taken to a medical facility for immediate medical attention; or one vehicle having been towed from the scene as a result of disabling damage suffered in the crash."
    - We gather this data from the {State Code} {Year} MCMIS sheets for every state of interest between the years 2023 and 2025
- Our aim with this dataset is to develop location based risk scores so we will use this to get the moving averages of different crash statistics in each county. Specifically, we are interested in the number of crashes, number of fatalities, number of injuries and number of vehicles in accidents in each county for each month.
- For more information see FMCSA

Field Name	Data Type	Description
Crash ID	String	A unique identifier for each crash record.
Crash Year	Numeric	The year in which the crash occurred.
Quarter-CY	String	Calendar quarter of the crash. Calculated as: Q1 (Jan 01-Mar 31), Q2 (Apr 01-Jun 30), Q3 (Jul 01-Sep

Field Name	Data Type	Description
		30), Q4 (Oct 01-Dec 31).
Report State	String	The state in which the crash occurred.
Truck Bus Indicator	String	Indicates vehicle type involved: "T" for truck, "B" for bus.
Fatal Count	Boolean/Numeric	Calculated field indicating whether the crash record involves a fatality.
Non Fatal Count	Boolean/Numeric	Calculated field indicating whether the crash record is non-fatal.
Fatalities	Numeric	Calculated field showing the number of fatalities resulting from the crash.
Injuries	Numeric	Calculated field showing the number of injuries resulting from the crash.
Injury Count	Boolean/Numeric	Calculated field indicating whether the crash resulted in injuries.
Tow Away	Boolean/Numeric	Calculated field indicating whether the crash required vehicle(s) to be towed away.
Federal Recordable	String	"Yes" or "No" - Indicates if crash meets federal criteria: at least 1 fatality, 1 injury requiring immediate medical attention away from the scene, or a vehicle towed due to disability damage.
State Recordable	String	"Yes" or "No" - Indicates if crash meets state reportable criteria.
Vehicles In Accident	Numeric	The total number of vehicles involved in the crash.
City Code	String	Code that translates to city name.
City	String	Name of the city where the crash occurred.
County Code	String	Code that translates to county name.
County Name	String	Name of the county where the crash occurred.
State	String	Name of the state where the crash occurred.
Location	String	Text field with detailed location information.
Trafficway Desc	String	Description of the trafficway where the crash occurred.
Road Surface Condition Desc	String	Description of road surface conditions (e.g., dry, wet, snow, ice).
Weather Condition Desc	String	Description of weather conditions (e.g., snow, rain, no adverse conditions).

Field Name	Data Type	Description
Light Condition Desc	String	Description of lighting conditions (e.g., daylight, dark).
First Harmful Event	String	Description of the first harmful event in the crash (e.g., collision involving motor vehicle in transport).
USDOT Number	String	U.S. Department of Transportation identification number for the carrier.
Carrier Name	String	Name of the carrier involved in the crash.
Carrier Street	String	Street address of the carrier.
Carrier City	String	City of the carrier's address.
Carrier State	String	State of the carrier's address.
Carrier Zip Code	String	ZIP code of the carrier's address.
Carrier Domicile	String	"Yes" or "No" - Indicates if carrier is domiciled in the state where the crash occurred.
Carrier Add Date	Date	Date when the carrier obtained operating authority (Census data).
New Entrant	Boolean/String	Calculated field indicating if the carrier's crash occurred within 18 months of obtaining operating authority.
Mexican Location	String	Carrier address/Mexican neighborhood if applicable.
Interstate Carrier	String	"Yes" or "No" - Indicates if carrier operates as an interstate carrier.
State Number	String	State registration number of the carrier.
State Issuing Number	String	State that assigned the registration number.
Driver Age Category	String	Age category of the driver.
Driver License State	String	State that issued the driver's license.
Valid Driver License	String	"Yes" or "No" - Indicates if the driver had a valid license.
Driver License Class Desc	String	Description of the driver's license class.
Citation Issued Code Desc	String	Indicates if citation was issued: "Yes", "No", or "Warning".

Field Name	Data Type	Description
Vehicle Configuration Desc	String	Description of vehicle configuration (e.g., bus, tractor/semi-trailer).
Cargo Body Type Desc	String	Description of cargo body type (e.g., bus, cargo tank).
Axles	Numeric	Number of axles on the vehicle.
GVWR	Numeric	Gross Vehicle Weight Rating.
GVW Rating Desc	String	Description of the Gross Vehicle Weight Rating.
Vehicle License Number	String	License plate number of the vehicle.
Vehicle License State	String	State that issued the vehicle license.
Hazmat Placard	String	"Yes" or "No" - Indicates if vehicle displays a hazardous materials placard.
Crash Number	String	Report number for the crash.
Crash Date	Date	Date when the crash occurred.
Day of Week Category	String	Category indicating day of the week when the crash occurred.
Crash Time	Time	Time when the crash occurred (in military format).
Time of Day Category	String	Category indicating time of day when the crash occurred.
Crash Seq No	Numeric	Sequence number indicating vehicle order (1st or 2nd) in the crash; value is 1 unless multiple vehicles were involved.
Agency	String	Agency that collected information about the crash.
Record Status	String	Indicates if this is an "Add record" (added to MCMIS and not modified) or a "Change record" (modified since initial addition).
Matched Status	String	Indicates if the US DOT# successfully matched with MCMIS Motor carrier census information. Values include: complete, nonmatch, intratstate, fmcsa hold, potential resolution, to census search, non motor carrier, initial load.
SAFETYNET Input Date	Date	Date when the crash was added to SAFETYNET.

Field Name	Data Type	Description
MCMIS Original Upload Date	Date	Date when the crash record was first uploaded from SAFETYNET to MCMIS. May not be populated for older records.
MCMIS Upload Date	Date	Date when the current crash record was uploaded to MCMIS. For change records, this is the date of the change; for add records, this is the original upload date.
Number Days to SAFETYNET	Numeric	Calculated field showing the number of days between crash date and SAFETYNET input date.
Number Days to MCMIS	Numeric	Calculated field showing the number of days between crash date and MCMIS upload date.
Counter	Numeric	Field used to subtotal the number of records.

Precipitation Data

- This dataset provides daily precipitation data for the conterminous United States from January 1, 2023 to January 31, 2025. The data is spatially gridded at a 4km resolution using the PRISM model, which was developed by Dr. Christopher Daly of the PRISM Climate Group at Oregon State University.
- The PRISM model defines a "day" differently than the standard midnight-to-midnight period. Instead, each "day" represents the 24-hour period ending at 12:00 Greenwich Mean Time (7:00am Eastern Standard Time).
- The reliability of the data depends on its age. Data older than 6 months is considered "stable" and unlikely to change until a major new PRISM release. Data from 1-6 months ago is "provisional" and likely to change as reporting networks finalize their information. Data from the current month should be treated as "early results" that will certainly change as new reporting stations are added and quality control measures are applied.
- Each day of data includes seven different files:
  1. A CSV file contains a list of all stations that provided input data for the PRISM model run that produced the grid associated with this particular day.
     - Station
     - Name
     - Longitude
     - Latitude
     - Elevation (m)
     - Network
     - station_id

2. A BIL (Band Interleaved by Line) file serves as the main raster data file, which is a common format for storing spatial raster data.
3. A PRJ file defines the coordinate system and projection information for the spatial data.
4. An HDR file contains header information about the data structure and parameters.
5. An STX file is included with each day's data.
6. An XML auxiliary file contains metadata for the BIL format.
7. A TXT file provides additional metadata and information about the dataset.

- For more information see PRISM

Zip Code Data

- This dataset contains the 2020 Census 5-Digit ZIP Code Tabulation Area (ZCTA5) shapefiles from the TIGER/Line series, last updated on January 27, 2024. The data is derived from the U.S. Census Bureau's Master Address File / Topologically Integrated Geographic Encoding and Referencing (MAF/TIGER) Database.
- ZCTAs are geographic representations of U.S. Postal Service (USPS) ZIP Code service areas created by the Census Bureau specifically for statistical purposes. These boundaries are delineated once per decade following each decennial census for the United States, Puerto Rico, and U.S. territories. The ZCTA boundaries in this release reflect those established following the 2020 Census.
- Files:
  - `tl_2020_us_zcta520.cpg` - A code page file that specifies the character encoding used in the attribute table.
  - `tl_2020_us_zcta520.dbf` - The dBASE database file containing attribute data for each ZCTA, such as ZIP code, geographic identifiers, and area measurements.
  - `tl_2020_us_zcta520.prj` - The projection file defining the coordinate system and map projection used for the spatial data.
  - `tl_2020_us_zcta520.shp` - The main shapefile containing the actual geometric data (polygons) for each ZCTA.
  - `tl_2020_us_zcta520.shp.ea.iso.xml` and `tl_2020_us_zcta520.shp.iso.xml` - XML metadata files containing information about the data source, creation, specifications, and other details about the shapefile.
  - `tl_2020_us_zcta520.shx` - The shape index file that allows software to quickly locate the spatial features within the .shp file.
- For more information see Data.gov

US Cities States Data

- Free US Zip Code Database provided by https://simplemaps.com/data/us-zips

- All latitude and longitude coordinates for ZCTAs (and geographic relationships) are now based on ZCTA shapes from the 2020 Census.

Column Name	Data Type	Description
city	String	The primary city name associated with the ZIP code, using proper capitalization and spacing.
city_ascii	String	The city name in ASCII format, with special characters removed and standardized for easier processing.
state_id	String	The two-letter state abbreviation (e.g., "NY" for New York) following USPS standards.
state_name	String	The full name of the state (e.g., "New York").
county_fips	String	The Federal Information Processing Standards (FIPS) code for the county, a unique identifier used by the US government.
county_name	String	The full name of the county where the ZIP code is located.
lat	Numeric	The latitude coordinate of the approximate geographic center of the ZIP code area, in decimal degrees.
lng	Numeric	The longitude coordinate of the approximate geographic center of the ZIP code area, in decimal degrees.
population	Numeric	The estimated population residing within the ZIP code boundaries, based on recent Census or other demographic data.
density	Numeric	Population density, measured in people per square mile or square kilometer.
source	String	The source of the ZIP code data, which may include Census Bureau, USPS, or other demographic sources.
military	Boolean	Indicates whether the ZIP code is primarily associated with a military installation (true/false).
incorporated	Boolean	Indicates whether the area is within an incorporated city or town boundary (true/false).
timezone	String	The time zone in which the ZIP code is located (e.g., "America/New_York").
ranking	Numeric	A relative importance ranking assigned to the ZIP code, potentially based on population or other metrics.
zips	String	The 5-digit ZIP code itself.
id	String/Numeric	A unique identifier assigned to each record in the database.

`Fusion Site Regional Data`

- Information on Fusion Site Geographic footprint

Column Name	Data Type	Description
Brand	Text	Company brand name (e.g., "Arkansas Portable Toilets")
Region	Text	Geographic region designation (e.g., "Gulf South")
State	Text	U.S. state location (e.g., "Arkansas")
City	Text	City name (e.g., "Little Rock")
Address	Text	Complete address string
PR	Boolean	Indicator for Portable Restroom services (X = Yes, blank = No)
RR	Boolean	Indicator for Restroom Trailer services (X = Yes, blank = No)
T	Boolean	Unknown service type indicator
Shower	Boolean	Indicator for Shower services (X = Yes, blank = No)
RO	Boolean	Unknown service type indicator
Front Loader	Boolean	Indicator for Front Loader equipment (X = Yes, blank = No)
Septic	Boolean	Indicator for Septic services (X = Yes, blank = No)
Grease	Boolean	Indicator for Grease Trap services (X = Yes, blank = No)
Concrete	Boolean	Indicator for Concrete services (X = Yes, blank = No)
S&B	Boolean	Unknown service type indicator
Conex	Boolean	Indicator for Conex container services (X = Yes, blank = No)
Mobile Offices	Boolean	Indicator for Mobile Office services (X = Yes, blank = No)
Res Storage	Boolean	Indicator for Residential Storage services (X = Yes, blank = No)
Refrigeration	Boolean	Indicator for Refrigeration services (X = Yes, blank = No)
Latitude	Numeric	Geographic latitude coordinate (e.g., 34.768452)
Longitude	Numeric	Geographic longitude coordinate (e.g., -92.277291)
Number	Numeric	Street number portion of address (e.g., 924)
Street	Text	Street name portion of address (e.g., "W 15th St")
Unit Type	Text	Type of unit designation, if applicable
Unit Number	Text	Unit identifier, if applicable
City	Text	City name, parsed from address (e.g., "North Little Rock")
State	Text	State abbreviation (e.g., "AR")

Column Name	Data Type	Description
County	Text	County name (e.g., "Pulaski County")
Zip	Numeric	ZIP/Postal code (e.g., 72114)
Country	Text	Country code (e.g., "US")
FUSION SITE	Text	Contact information for regional manager

2. Define Outcomes (evaluation metrics)

Evaluation metrics: F1 Score, AUC, MCC, sensitivity and specificity curves depending on chosen threshold.
Target: binary classification (0-1) outcome of whether or not a driver has an accident on a given trip given historical information about them as well as recent precipitation and crash data for their specific location

3. Data Ingestion

- For Motive Data, large datasets— `data_inspections_w_vehicle_map`, `data_driving_periods_w_vehicle_map`, `data_combined_events_w_vehicle_map`, and `data_idle_events_w_vehicle_map`—each around 1GB, are loaded into DBFS and processed using PySpark. The processing steps include data cleaning and aggregation across all four tables.
- For crash incidents from insurance records, such as `accident_register_combined_driverid` and `monthly_insurance_claims_driverid.csv`, the data is small enough to be efficiently handled with pandas. Using pandas for these datasets avoids unnecessary resource overhead while maintaining ease of use.
- For External Data
    - **Multiple File Formats**
        - Excel (`.xlsx`) for business location data and state crash data
        - CSV (`.csv`) for accident records, claims data, and reference data
            - Used when data is small enough because they offer universal compatibility
        - Parquet (`.parquet`) for preprocessed data
            - Parquet stores data in a columnar format which allows for more efficient compression and faster reading of specific columns. These files include metadata about the schema so types do not need to be inferred and its optimized for Spark processing.
            - The precipitation data is stored in Parquet after complex spatial processing, preserving the precise relationships between ZIP codes and weather

readings.
    - Binary BIL (`.bil`) for precipitation raster data
    - Shapefile (`.shp`) for geographic boundary data
- **Storage Locations**
  - Databricks File System (`/dbfs/`) for most raw files
  - Google Cloud Storage for original state crash data
  - Local storage (`/data/` and `/tmp/`) for temporary processing

4. Summarized Relavant Datasets:

FusionSite Internal Data

a. Data Cleaning Steps:

All these data cleaning steps are being done in `motive_data_preprocessing.ipynb`
Before joining the tables together, some of the steps that we did to clean the data before joinn

1. **`data_combined_events_w_vehicle_map`**

- Drop NULL values from `driver_id`.
- Ensure `event_id` is unique and non-null (primary key); if so, drop the `id` column.
- Add a new column `main_event_type` with the value `"hazard"`.
- Extract `trip_date` from `start_date` timestamp.

2. **`data_driving_periods_w_vehicle_map`**

- Drop NULL values from `driver_id`.
- Filter `minutes_driving` to retain only non-null values greater than 0.
- Filter `driving_distance` to keep values between **0 and 10,000** (outliers above 10,000 are removed).
- Count trips where `driving_distance` exceeds **500**.
- Check the number of NULL values in `vehicle_id`.
- Ensure `event_id` is unique and non-null (primary key); if so, drop the `id` column.
- Add `main_event_type` column with the value `"driving"`.
- Extract `trip_date` from `start_date` timestamp.

3. **`data_inspections_w_vehicle_map`**

- Drop NULL values from `driver_id`.
- Rename `"status"` to `"inspection_status"` and `"status-2"` to `"status"`.

- Ensure `inspection_id` is unique and non-null (primary key); if so, drop the `id` column.

4. **`data_idle_events_w_vehicle_map`**

- Drop NULL values from `driver_id`.
- Remove idling events where `minutes_idling` exceeds **500 minutes**.
- Ensure `event_id` is unique and non-null (primary key); if so, drop `id_x` and `id_y` columns.
- Extract `trip_date` from `start_date` timestamp.

b. Data Aggregation

After cleaning, all 4 tables from above are joined to create a comprehensive dataframe that contains all events information related for a specific trip and on a specific date

1. **Create a Main DataFrame (`all_trips_df`)**
   - Extract distinct `(driver_id, vehicle_id, trip_date)` records from `cleaned_driving_period`.
   - Perform a **cross join** with a list of `main_event_type` values: `["hazard", "driving", "inspection", "idle"]`.
   - This ensures each trip-date entry has all event types available for later joins.
2. **Join `cleaned_driving_period`**
   - **Join Keys:** `driver_id`, `vehicle_id`, `trip_date`, `main_event_type`
   - **Join Type:** `LEFT JOIN`
   - **Preprocessing:**
     - Rename columns (`created_at_driving_period`, `event_id_driving_period`, etc.).
     - Adds driving-related event data to `all_trips_df`.
3. **Join `cleaned_combined_event_data` (Hazard Events)**
   - **Join Keys:** `driver_id`, `vehicle_id`, `trip_date`, `main_event_type`
   - **Join Type:** `LEFT JOIN`
   - **Preprocessing:**
     - Rename columns (`created_at_combined_event`, `event_id_combined_event`, etc.).
     - Drop unnecessary columns (`number`, `status`, `make`, `model`, `group_id`, `group_name`, `month`).
     - Links hazard-related events to trip records.
4. **Join `cleaned_inspection_data`**
   - **Join Keys:** `driver_id`, `vehicle_id`, `trip_date`, `main_event_type`

  - **Join Type:** `LEFT JOIN`
  - **Preprocessing:**
    - Rename `date` → `trip_date`, `location` → `inspection_location`.
    - Drop unnecessary columns.
    - Links inspection data to trip records.
5. **Join `cleaned_idle_data`**
  - **Join Keys:** `driver_id`, `vehicle_id`, `trip_date`, `main_event_type`
  - **Join Type:** `LEFT JOIN`
  - **Preprocessing:**
    - Rename columns (`start_time_idle`, `end_time_idle`, `event_id_idle`).
    - Drop unnecessary columns (`driver_company_id` and others).
    - Links idling events to trip records.
6. **Cache the final dataframe** for any further advanced data exploratory

External Location Data

Note: We have successfully created a comprehensive dataset that joins all accidents with both state crash statistics and precipitation data for their respective counties. This integrated dataset serves as proof of concept that our data pipeline can effectively combine these diverse data sources. However, we still need to finalize our approach for considering precipitation across broader geographic areas. The infrastructure for these joins is in place, allowing us to explore different spatial aggregation strategies during the feature engineering phase.

a. Data Cleaning Steps:

`accident_data_location_processing.ipynb`

- **Accident Location Parsing**
  - Extract zip codes state information using regex pattern matching for state abbreviations and full names
  - Extract city names using multiple methods:
    - Matching against known cities from FusionSite regional data
    - Pattern matching for city names before state names
    - Special case handling for specific patterns observed in the free text
- **DBA (Doing Business As) Standardization**
  - Remove accidents with non valid DBA
  - Forward-fill missing DBA values in csv
  - Use fuzzy matching to map non-standard DBA names to official brand names
- **Driver Information Cleaning**

    - Use fuzzy matching to standardize driver names
- Removed accidents with no valid location information or without valid driver information

`weather_data_processing`

- **Precipitation Data Processing**
    - Extract date information from BIL file names using regex
    - Read raster files with appropriate metadata and transformations
- **ZIP Code Point Generation**
    - Generate representative points for each ZIP code polygon
    - Store coordinates and ZIP code attributes for spatial processing

`state_crash_data_processing`

- **Standardize County Codes and State IDS**
    - Convert county codes to numeric, handling conversion errors with `pd.to_numeric(errors='coerce')`
    - Fill null values with zeroes
    - Ensure consistent 3-digit format for country codes
    - Convert all state identifiers to lowercase for consistent matching
- **Handle Date and Time Formats**
    - Convert string dates to timestamp format
    - Create `Reference_Date` by using `Date of Accident` when available, otherwise use `Date Reported` (will double check this logic business team)
    - Extract year, month, and day components
- **Fix Data Errors in State/County Information**
    - Apply manual corrections to specific ZIP codes with known issues
    - Create lookup dictionaries for managing state/county information based on ZIP codes
- There is no missing data in the data fields, but not all the zip codes present in the precipitation data are present in this data, so we use proximity based matching later on

b. Data Aggregation and EDA

`state_crash_data_processing.ipynb`

- **Multi-State Data Integration**
    - Load crash data from multiple state files (50 states plus territories)
    - Union the data from each state using `reduce`
- **Monthly County-Level Aggregation**
    - Group by `Crash Year`, `Crash Month`, `State`, and `County Code`

    - Compute key metrics:
        - `crash_count`: Count of crash incidents
        - `total_fatalities`: Sum of fatalities
        - `total_injuries`: Sum of injuries
        - `total_vehicles`: Sum of vehicles involved in accidents
- **Enhance County Data with Geographic Information**
    - Join county crash data with US city-state reference data
    - Link using `State`/`state_id` and `County Code`/`county_code`
    - Add geographic details like FIPS codes, county names, and ZIP codes
- **Calculate Correlation Matrix for Crash Metrics**
    - Generate statistical correlations between:
        - Crash count
        - Total fatalities
        - Total injuries
        - Total vehicles
    - Organize into a correlation matrix for analysis
- **Export Processed Data to CSV**
    - Convert final Spark DataFrame to pandas
    - Save to temporary local CSV file
    - Upload to Google Cloud Storage for further analysis
    - This makes the processed data (41,712 rows, 8 columns) accessible for downstream processes

`accident_data_location_processing.ipynb`

- **DBA-based Location Mapping**
    - Create location mapping from FusionSite regional data for DBAs with single locations
    - Apply location details (state, city, zip, county) to accidents based on DBA match
    - Joins
        - Accident Data with DBA Location Information
            - **Key:** DBA (Doing Business As)
            - **Purpose:** Add business location details to accident records
            - **Tables:** `accident_data`, `fusionsite_regional`
        - FusionSite Regional Data with Location Updates
            - **Type:** Iterative join/lookup
            - **Keys:** Multiple (County, City, or Zip - any can match)
            - **Purpose:** Fill missing location data by looking up partial matches and find ZIP codes for records with missing ZIP but known city and state

        - **Tables:** `accident data`, `fusionsite_regional`, `us_cities`
- **Driver-based Location Enhancement**
  - For multi-location DBAs with missing location data, extract location from driver information
  - Parse driver's "Groups" field to extract city information
  - Map extracted city to known FusionSite cities
- **General Manager-based Location Enhancement**
  - For remaining records with missing location information, use the manager information to identify the DBA site corresponding to that driver
- **Missing Data Backfilling**
  - Fill missing state/city/county based on ZIP code
  - Fill missing state/ZIP based on city
  - Use external city-state database to fill remaining gaps

`weather_data_processing`

- **ZIP Code to Precipitation Mapping**
  - Extract precipitation values at ZIP code representative points
    - `get_zip_points()` creates representative points for each ZIP code area
      - Calls the `representative_point()` method on each polygon which guarantees returning a point that falls *inside* the polygon which means the extracted precipitation value will actually represent that ZIP code area
  - Process BIL files in batches to manage memory consumption
  - Handle coordinate system transformations when needed
  - Joins
    - Precipitation Data with ZIP Code Information
      - **Type:** Left join
      - **Key:** zip code
      - **Purpose:** Add city, state, and county information to precipitation data by ZIP code
      - **Tables:** `precipitation_data`, `us cities states`
      - `map_bil_to_zip_codes()` creates a mapping between BIL data and ZIP codes
        - Uses `get_values_at_points()` to sample each raster at the ZIP code points
        - Organizes the results into a structured DataFrame with geographic attributes
        - Pivots the data so each (non location) column represents a

    - precipitation measurement
    - Creates a wide-format table suitable for analyzing precipitation patterns by ZIP code over time
- **Time Series Creation**
  - Map daily precipitation values to ZIP codes for each date
  - Join with location data to add state, county, and city information
- The daily data for each zip was processed and saved separately in parquet format for 2023, 2024, and 2025

`mapping_external_location_data_to_accidents`

- Join crash statistics with US city/state reference data by adding state FIPS codes, county names, and ZIP codes with a left join on state and county code
  - ZIP Code Consolidation
    - Group by state and county code
    - Combine multiple ZIP codes into single comma-separated lists
    - Create a comprehensive lookup dictionary mapping ZIP codes to state/county information
  - County Crash Statistics to US Cities Reference Data
    - **Type**: Left join
    - **Keys**: state_id/State, county_code/County Code
    - **Purpose**: Enrich crash statistics with geographic details
    - **Result**: Added ZIP codes, county names, and state FIPS codes to crash statistics
  - Accident Location to Crash Statistics
    - **Type**: Lookup-based join with fallback strategy
    - **Keys**: Primary - ZIP code, year, month, state, county code
    - **Fallback Strategy**:
      - Try exact match first
      - If no match, try nearby ZIP codes (+/-1, +/-2, +/-3)
      - If still no match, try matching just state and month
      - Further fallbacks to nearby months or most recent data for the state
    - Added county-level crash statistics to each accident record:
      - county_crash_count
      - county_total_fatalities
      - county_total_injuries
      - county_total_vehicles
- Accident Records to Precipitation Data
  - **Type**:left join with special handling for multiple ZIP codes

    - **Keys**: accident_date/date, joining_zip/zip_code
    - **Process**:
        1. Transform precipitation data from wide format (columns for dates) to long format
        2. Create a joining_zip field in accident data that uses either zip or dba_zip
        3. Handle single ZIP codes with direct join
        4. For multiple ZIP codes, average the precipitation values
    - **Result**: Added precipitation values to each accident record
    - Special handling
        - Multiple ZIP Codes: For accidents with multiple possible ZIP codes, the code averages precipitation across all matching ZIPs
        - Inexact Matches: Tracked 322 cases where exact county matches weren't found, with fallback to nearest county
        - Missing Values: Detailed logging of missing data percentages for each field
        - Date Handling: Used Date of Accident with fallback to Date Reported when needed
- EDA
    - Categorize precipitation amounts into meaningful ranges
    - Calculate statistics about accidents occurring during different weather conditions
- An important consideration for our predictive modeling will be ensuring we only use historical data rather than same-day information that wouldn't be available at prediction time. In the feature engineering phase, we will:
    - Implement various historical moving averages
    - Explore different time windows (e.g., 7-day, 30-day, seasonal)
    - Develop lagged variables to prevent data leakage

5. Data Checkpoint

- **Internal Data**: For Motive internal data, we cached each table when it is finished with cleaning processed. We cached the data once more when we aggregated all the Motive's event data; This aggregation would now be our final reference points for more advanced data exploratory analysis
- **External Data**
    - `accident_data_location_processing.ipynb` creates location-enhanced accident data
        - accidents_with_location: Processed accident data with enhanced location information
    - `weather_data_processing.ipynb` processes precipitation data by year and creates lookup tables

        - prism_rainfall_2023
        - prism_rainfall_2024
        - 2025_precipitation_data
    - `state_crash_data_processing.ipynb` aggregates crash statistics by county and month
        - monthly_county_counts.csv file containing county-level crash statistics
    - `mapping_external_location_data_to_accidents.ipynb` combines all these intermediate files to map crash statistics, weather data, and location information to individual accidents

6. How split train, valid, test

We will split the data into train/test split based on driver ID, stratified by DBA to ensure that no one site is overrepresented in the splits.
Our split ratios would be

- Train: 70%
- Valid: 15%
- Test: 15%

7. Credit Assignment

- Since Trang is working directly with FusionSite and has access to more domain context, she will take the lead on the datasets we are using from the company including those from Motive API. Isa will take ownership over the parts of the project where we are bringing in external data to understand the risk factors associated with weather and specific locations based on state crash data and precipitation data. So far, we have met 1-2/times a week to define the problem, discuss the problems we are trying to solve and align on the overall direction of the project. We have found this cadence has worked well and will continue throughout the remaining milestones.
- **Milestone 3:** Feature Engineering and hyperparameter tuning
    - Trang:
        - Feature engineering for FusionSite and Motive API datasets
        - Hyperparameter tuning for models using internal company data
        - Documentation of internal data preprocessing steps
    - Isa:
        - Feature engineering for external weather and location-based risk data
        - Hyperparameter tuning for models using external data sources
        - Documentation of external data preprocessing methodology

- Milestone 4: Final model selection and final report (due 11:30AM Monday 4/14)
  - Trang:
    - Write sections on internal data sources, methods, and corresponding results
    - Create visualizations for internal data insights
  - Isa:
    - Write sections on external data sources, methods, and corresponding results
    - Create visualizations for external data insights
- In class presentations
  - Trang:
    - Present sections on problem formulation and internal data sources
    - Discuss technical approaches for company data integration
    - Address questions related to FusionSite and Motive API
  - Isa:
    - Present sections on external data integration and potential extensions
    - Discuss insights from weather and location-based risk analysis
- While we've divided specific responsibilities, certain aspects of the project (such as hyperparameter tuning and model selection) will require collaborative effort. For these shared tasks, both team members will independently experiment with different approaches to gain hands-on experience. During our weekly meetings, we'll compare results, discuss findings, and decide which combination of our approaches yields the best performance for each specific task. This ensures both team members contribute to critical modeling decisions while maintaining our established areas of focus.