

# Incident Response

---

Team 10

Team Member: Xi Sun, Austin Wang

# Datasets provided:

---

Weather dataset for Davidson County (2010 - 2022):

- Precipitation, snow, wind speed, clouds, temperature, etc.

Traffic dataset for Davidson County (2017 - 2022):

- Time stamp, speed, congestion, travel time seconds, etc.

Incident dataset:

- Road segment ID, location(latitude, longitude), type of incidents (emdcardnumber), time, response time, etc.

# Analysis Objective:

---

Temporal Analysis on incidents

Spatial Analysis on incidents

Effect of time and space on response time (effect of time\_local on avg response\_time)

Join Incident with traffic datasets for better analysis

Join Incident with weather datasets for better analysis

Graphical Visualization for the Query

# Approach:

---

- Provided with over 20 GB of data → Upload to AWS S3
  - Parquet
  - GeoJSON
- Hard to run on local with big data → Establish connection with Athena and run all the queries in cloud
- Data visualization → use Plotly

# S3 (Dataset Bucket)

The screenshot shows the AWS S3 console interface for the 'vandy-bucket'. At the top, there's a navigation bar with the AWS logo, a 'Services' dropdown, a search bar containing 'Search for services, features, blogs, docs, and more', and a user account section. Below the navigation bar, a message says 'We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.'

The main area shows the 'vandy-bucket' details. The breadcrumb navigation indicates 'Amazon S3 > Buckets > vandy-bucket'. The bucket name 'vandy-bucket' is displayed with an 'Info' link. Below the bucket name are tabs for 'Objects' (which is selected), 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'.

The 'Objects (4)' section displays a table of objects. The table has columns for Name, Type, Last modified, Size, and Storage class. The objects listed are:

Name	Type	Last modified	Size	Storage class
nfd_incidents_xd_seg.parquet/	Folder	-	-	-
traffic.parquet/	Folder	-	-	-
USA_Tennessee.geojson/	Folder	-	-	-
weather_tn.parquet/	Folder	-	-	-

Below the table are buttons for Actions (with options like Copy S3 URI, Copy URL, Download, Open, Delete, Create folder, and Upload), a search bar for 'Find objects by prefix', and navigation controls (back, forward, and refresh).

At the bottom of the page, there are links for Feedback, Looking for language selection? Find it in the new United Settings, © 2022, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

# S3 (Results Bucket)

The screenshot shows the AWS S3 console interface for the 'athena-vandy' bucket. At the top, there's a blue header bar with the AWS logo, a search bar, and a feedback link. Below the header, a message encourages users to provide feedback to improve the console. The main area displays the 'Objects' tab, which lists 16 items. Each item has a checkbox, a preview icon, a name, a type (e.g., csv, metadata, txt), a last modified date, a size, and a storage class (Standard). The objects are mostly CSV files and metadata files, with some text files interspersed. The table is paginated at the bottom.

Name	Type	Last modified	Size	Storage class
08f7700a-b708-47a1-8461-91d94dd4c22e.csv	csv	May 2, 2022, 13:04:09 (UTC-05:00)	7.6 KB	Standard
08f7700a-b708-47a1-8461-91d94dd4c22e.csv.metadata	metadata	May 2, 2022, 13:04:10 (UTC-05:00)	334.0 B	Standard
2b19f7f2-542a-4a4d-ace5-7b934627ad0c.txt	txt	May 2, 2022, 13:34:55 (UTC-05:00)	0 B	Standard
50112ff8-58ae-4c84-ac10-75fb63938f.csv	csv	May 2, 2022, 13:04:42 (UTC-05:00)	7.6 KB	Standard
50112ff8-58ae-4c84-ac10-75fb63938f2.csv.metadata	metadata	May 2, 2022, 13:04:43 (UTC-05:00)	334.0 B	Standard
f5d74690-27b6-413c-a424-b6af53201e52.txt	txt	May 2, 2022, 13:38:06 (UTC-05:00)	0 B	Standard
79a0dd5a-4503-4eb0-9e60-214865ac67f3.txt	txt	May 2, 2022, 13:37:39 (UTC-05:00)	0 B	Standard
8af64ff4-3e72-4ef3-a161-5a27f0312c80.csv	csv	May 2, 2022, 13:05:36 (UTC-05:00)	5.6 KB	Standard
8af64ff4-3e72-4ef3-a161-5a27f0312c80.csv.metadata	metadata	May 2, 2022, 13:05:36 (UTC-05:00)	334.0 B	Standard
b5cb9e48-c3c9-4130-8450-406f94953776.csv	csv	May 2, 2022, 13:15:09 (UTC-05:00)	3.1 KB	Standard
b5cb9e48-c3c9-4130-8450-406f94953776.csv.metadata	metadata	May 2, 2022, 13:15:09 (UTC-05:00)	1.5 KB	Standard
b96f2709-c87b-4760-baee-f88bcb27469.csv	csv	May 2, 2022, 13:07:45 (UTC-05:00)	645.0 B	Standard
b96f2709-c87b-4760-baee-f88bcb27469.csv.metadata	metadata	May 2, 2022, 13:07:45 (UTC-05:00)	1.5 KB	Standard
f12282e3-1ee4-4d76-ae93-65c358dca6dc.csv	csv	May 2, 2022, 13:11:07 (UTC-05:00)	1.2 KB	Standard
f12282e3-1ee4-4d76-ae93-65c358dca6dc.csv.metadata	metadata	May 2, 2022, 13:11:07 (UTC-05:00)	610.0 B	Standard
f618f41c-f46c-47ea-9c9f-ee1e99c96ff2.txt	txt	May 2, 2022, 13:37:35 (UTC-05:00)	0 B	Standard

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

# Athena

The screenshot shows the Amazon Athena Query editor interface. The top navigation bar includes the AWS logo, Services, a search bar, and account information for N. Virginia and user vclabs/user1796666=austin.w.wang@vanderbilt.edu.

The main area has tabs for Editor, Recent queries, Saved queries, and Settings. The Workgroup is set to primary. On the left, the Data panel shows the Data Source (AwsDataCatalog) and Database (traffic-db). Below it are sections for Tables and views, showing nfд\_incidents\_xd\_seg, traffic-data, and weather-data; and Views (0).

The central pane displays a SQL query:

```
1 SELECT
2     emdcardnumber,
3     COUNT(*) AS "count",
4     AVG(response_time_sec) AS "avg_response_time_sec",
5     AVG(dist_to_seg) AS "avg_dist_to_seg",
6     (AVG(response_time_sec) / AVG(dist_to_seg)) AS "avg_response_to_dist_ratio"
7 FROM "traffic-db"."nfd_incidents_xd_seg"
8 WHERE
9     emdcardnumber IS NOT NULL
10    AND response_time_sec > 0
11    AND dist_to_seg > 0
12 GROUP BY emdcardnumber;
```

The status bar indicates the query was completed in 0.15 sec, with a run time of 1.104 sec and 345.39 KB scanned data.

The Results section shows 79 rows of data:

#	emdcardnumber	count	avg_response_time_sec	avg_dist_to_seg	avg_response_to_dist_ratio
1	29A2V	530	384.18490566037735	3.523172691011396	109.04515314862115
2	29B1	7476	385.58681112894595	3.1614990289033083	121.96328627774467
3	29D8V	19	316.8421052631579	3.137267320658179	100.99302127581733
4	29D7	160	384.5875	3.502680991734246	109.798037819477

At the bottom, there are links for Feedback, Unified Settings, and a footer with copyright information for 2022, Amazon Web Services, Inc. or its affiliates, and links for Privacy, Terms, and Cookie preferences.

# Sample Queries

# Creating Tables in Athena

```
# Schema
<pyarrow._parquet.ParquetSchema object at 0x132f9dd40>
required group field_id=-1 spark_schema {
    optional int32 field_id=-1 xd_id;
    optional int96 field_id=-1 measurement_tstamp;
    optional binary field_id=-1 measurement_tstamp_str (String);
    optional double field_id=-1 speed;
    optional double field_id=-1 average_speed;
    optional double field_id=-1 reference_speed;
    optional double field_id=-1 travel_time_seconds;
    optional double field_id=-1 confidence_score;
    optional binary field_id=-1 cvalue (String);
    optional double field_id=-1 congestion;
    optional double field_id=-1 extreme_congestion;
}

[austin@Austins-MacBook-Pro-2 month=4 % parq part-00184-427c2bcd-90e5-41f1-a1d3-c]
4c35772a251.c000.snappy.parquet --head 1
    xd_id measurement_tstamp measurement_tstamp_str speed \
0 1524378121 2017-04-30 19:00:00 2017-04-30 19:00:00 64.36

    average_speed reference_speed travel_time_seconds confidence_score \
0          62.0           61.0            19.0             28.0

    cvalue congestion extreme_congestion
0   98.50        0.0          0.0
[austin@Austins-MacBook-Pro-2 month=4 % parq /Users/austin/Downloads/emergencyres]
```

### 🔗 traffic-data

```
CREATE EXTERNAL TABLE IF NOT EXISTS `traffic-db`.`traffic-data` (
    `xd_id` int,
    `measurement_tstamp` bigint,
    `measurement_tstamp_str` string,
    `speed` double,
    `average_speed` double,
    `travel_time_seconds` double,
    `confidence_score` double,
    `cvalue` string,
    `congestion` double,
    `extreme_congestion` double
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
    'serialization.format' = '1'
) LOCATION 's3://vandy-bucket/traffic.parquet/'
TBLPROPERTIES ('has_encrypted_data'='false');
```

```
# Schema
<pyarrow._parquet.ParquetSchema object at 0x135673700>
required group field_id=-1 spark_schema {
    optional binary field_id=-1 station_id (String);
    optional binary field_id=-1 start_date_st (String);
    optional binary field_id=-1 end_date_st (String);
    optional int96 field_id=-1 timestamp_local;
    optional double field_id=-1 rh;
    optional double field_id=-1 wind_spd;
    optional int96 field_id=-1 timestamp_utc;
    optional binary field_id=-1 pod (String);
    optional double field_id=-1 slp;
    optional double field_id=-1 app_temp;
    optional double field_id=-1 elev_angle;
    optional double field_id=-1 solar_rad;
    optional double field_id=-1 pres;
    optional double field_id=-1 h_angle;
    optional double field_id=-1 dewpt;
    optional double field_id=-1 snow;
    optional double field_id=-1 uv;
    optional double field_id=-1 azimuth;
    optional double field_id=-1 wind_dir;
    optional double field_id=-1 ghi;
    optional double field_id=-1 dhi;
    optional double field_id=-1 vis;
    optional double field_id=-1 dni;
    optional binary field_id=-1 datetime (String);
    optional double field_id=-1 temp;
    optional double field_id=-1 precip;
    optional double field_id=-1 clouds;
    optional double field_id=-1 ts;
    optional binary field_id=-1 icon (String);
    optional double field_id=-1 code;
    optional binary field_id=-1 description (String);
    optional double field_id=-1 gps_coordinate_latitude;
    optional double field_id=-1 gps_coordinate_longitude;
    optional binary field_id=-1 spatial_id (String);
    optional int32 field_id=-1 days (Date);
}

[austin@Austins-MacBook-Pro-2 month=4 % parq /Users/austin/Downloads/emergencyres]
ponse/weather_tn.parquet/year\=2010/month\=1/part-00038-0c8142a9-ffb0-4186-8d5f-
060d42afe7d.c000.snapy.parquet --head 1
    station_id start_date_st end_date_st timestamp_local    rh   wind_spd \
0  720259-63844  2016-02-05  2016-02-12      2016-01-01  75.0       2.1

    timestamp_utc pod     slp   app_temp elev_angle solar_rad   pres \
0  2010-01-01 05:00:00.000  n  1019.0        4.4      -75.42      0.0  935.0005

    h_angle  dewpt  snow  uv azimuth  wind_dir  ghi  dhi  vis  dni \
0      NaN     1.0    0.0  0.0  324.18     350.0    0.0  0.0  16.0  0.0

    datetime  temp  precip  clouds      ts  icon  code \
0  2010-01-01 05:00:00.000  6.0     0.0  100.0  1.262322e+09  c04n  804.0

    description  gps_coordinate_latitude  gps_coordinate_longitude \
0  Overcast clouds                  35.223                   -83.419

    spatial_id  days
0  Franklin  2010-01-01
```

```
austin@Austins-MacBook-Pro-2 month=4 %
```

## weather-data

```
CREATE EXTERNAL TABLE IF NOT EXISTS `traffic-db`.`weather-data` (
    `station_id` string,
    `start_date_st` string,
    `end_date_st` string,
    `timestamp_local` bigint,
    `rh` double,
    `wind_spd` double,
    `timestamp_utc` bigint,
    `pod` string,
    `slp` double,
    `app_temp` double,
    `elev_angle` double,
    `solar_rad` double,
    `pre` double,
    `h_angle` double,
    `dewpt` double,
    `snow` double,
    `uv` double,
    `azimuth` double,
    `wind_dir` double,
    `ghi` double,
    `dhi` double,
    `vis` double,
    `dni` double,
    `datetime` string,
    `temp` double,
    `precip` double,
    `clouds` double,
    `ts` double,
    `icon` string,
    `code` double,
    `description` string,
    `gps_coordinate_latitude` double,
    `gps_coordinate_longitude` double,
    `spatial_id` string,
    `days` date
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
    'serialization.format' = '1'
)
LOCATION 's3://vandy-bucket/weather_tn.parquet/'
TBLPROPERTIES ('has_encrypted_data'='false');
```

```
month=4 --zsh-- 80x36
ponse/nfd_incidents_xd_seg.parquet --schema

# Schema
<pyarrow._parquet.ParquetSchema object at 0x135278ec0>
required group field_id=-1 schema {
    optional binary field_id=-1 ID_Original (String);
    optional double field_id=-1 latitude;
    optional double field_id=-1 longitude;
    optional binary field_id=-1 emdCardNumber (String);
    optional int64 field_id=-1 time_utc (Timestamp(isAdjustedToUTC=false, timeUnit
=microseconds, is_from_converted_type=false, force_set_converted_type=false));
    optional int64 field_id=-1 time_local (Timestamp(isAdjustedToUTC=false, timeUn
it=microseconds, is_from_converted_type=false, force_set_converted_type=false));
    optional double field_id=-1 response_time_sec;
    optional int64 field_id=-1 day_of_week;
    optional int32 field_id=-1 weekend_or_not;
    optional binary field_id=-1 geometry (String);
    optional int32 field_id=-1 Incident_ID;
    optional double field_id=-1 Dist_to_Seg;
    optional double field_id=-1 XDSegID;
}

[austin@Austins-MacBook-Pro-2 month=4 % parq /Users/austin/Downloads/emergencyres
ponse/nfd_incidents_xd_seg.parquet --head 1
      ID_Original  latitude  longitude emdCardNumber \
0  ObjectId(59d3a81908f47311c891f8e2)  36.037228 -86.783243  29B5
          time_utc        time_local  response_time_sec \
0  2017-01-01 07:59:29.507 2017-01-01 01:59:29.507       268.0
      day_of_week  weekend_or_not           geometry  Incident_ID \
0            6                  1 POINT (-86.78324314 36.03722849)      10
      Dist_to_Seg      XDSegID
0   13.55037  1.524394e+09
austin@Austins-MacBook-Pro-2 month=4 %
```

### nfd\_incidents\_xd\_seg

```
CREATE EXTERNAL TABLE IF NOT EXISTS `traffic-db`.`nfd_incidents_xd_seg` (
  `id_original` string,
  `latitude` double,
  `longitude` double,
  `emdcardnumber` string,
  `time_utc` bigint,
  `time_local` bigint,
  `response_time_sec` double,
  `day_of_week` int,
  `weekend_or_not` int,
  `geometry` string,
  `incident_id` int,
  `dist_to_seg` double,
  `xdsegid` double
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1'
)
LOCATION 's3://vandy-bucket/nfd_incidents_xd_seg.parquet/'
TBLPROPERTIES ('has_encrypted_data'='false');
```

## Count number of entries in each table

### nfd\_incidents\_xd\_seg

```
SELECT COUNT(*) FROM "traffic-db"."nfd_incidents_xd_seg";
-- result: 29765
-- time in queue: 0.175 sec
-- run time: 0.672 sec
```

### traffic-data

```
SELECT COUNT(*) FROM "traffic-db"."traffic-data";
-- result: 2949783075
-- time in queue: 0.206 sec
-- run time: 0.799 sec
```

### weather-data

```
SELECT COUNT(*) FROM "traffic-db"."weather-data";
-- result: 20792532
-- time in queue: 0.255 sec
-- Run time: 1.175 sec
```

## Order by types of incidents

```
1 SELECT
2     emdcardnumber,
3     COUNT(*) AS "count",
4     AVG(response_time_sec) AS "avg_response_time_sec",
5     AVG(dist_to_seg) AS "avg_dist_to_seg",
6     (AVG(response_time_sec) / AVG(dist_to_seg)) AS
7         "avg_response_to_dist_ratio"
8 FROM "traffic-db"."nfd_incidents_xd_seg"
9 WHERE
10    emdcardnumber IS NOT NULL
11    AND response_time_sec > 0
12    AND dist_to_seg > 0
13 GROUP BY emdcardnumber
14 ORDER BY COUNT(*) DESC;
```

Results (79)						
#	emdcardnumber	count	avg_response_time_sec	avg_dist_to_seg	avg_response_to_dist_ratio	
1	29B1	7476	385.58681112894595	3.1614990289033083	121.96328627774467	
2	29B5	2678	426.7176997759522	3.7595676700519376	113.50180053284085	
3	29A2	1937	382.8451213216314	3.3377674738187095	114.70095635021026	
4	29D2P	1421	452.96692470091483	3.8825778410146508	116.66654044018833	
5	29B1V	890	402.8	3.3581716348128676	119.94622187392928	
6	29D2M	868	335.2741935483871	4.752741127079406	70.5433316445778	
7	29B5U	667	420.2608695652174	3.6008681331512005	116.71098580259246	
8	29A2V	530	384.18490566037735	3.523172691011396	109.04515314862115	
9	29D2L	469	381.727078891258	3.6399152441404428	104.87251853068955	
10	29D5	452	459.1305309734513	3.7366575388136694	122.87198551227632	
11	29B1U	306	392.73202614379085	3.4164657075244156	114.95272008111739	
12	29A1	300	377.81333333333333	3.333745297148055	113.32999364304892	
13	29D3	215	443.6046511627907	3.53825842802923	125.3737283994469	
14	29B5V	179	401.6368715083799	3.153663592920708	127.35564833546854	
15	29D5V	169	460.33727810650885	3.154763813310736	145.91814327312343	
16	29D7	160	384.5875	3.502680991734246	109.798037819477	
17	29B3	147	392.421768707483	3.959533833703557	99.1080731189082	
18	29D5U	111	454.990990990991	2.7354898286471854	166.3288915301883	

## Types of incidents with high response time

WHERE filters out bad data

```
1 SELECT
2   emdcardnumber,
3   COUNT(*) AS "count",
4   AVG(response_time_sec) AS "avg_response_time_sec",
5   AVG(dist_to_seg) AS "avg_dist_to_seg",
6   (AVG(response_time_sec) / AVG(dist_to_seg)) AS "avg_response_to_dist_ratio"
7 FROM "traffic-db"."fd_incidents_xd_seg"
8 WHERE
9   emdcardnumber IS NOT NULL
10  AND response_time_sec > 0
11  AND dist_to_seg > 0
12 GROUP BY emdcardnumber
13 ORDER BY AVG(response_time_sec) DESC;
```

#	emdcardnumber	count	avg_response_time_sec	avg_dist_to_seg	avg_response_to_dist_ratio
1	29D9Y	3	807.6666666666666	4.400833792109867	183.5258282452543
2	29D2S	6	706.6666666666666	3.3099958218075387	213.49473072167405
3	29D1F	4	699.25	7.87716802339159	88.76921222494518
4	29D7V	11	662.8181818181819	4.558864290073672	145.39107541792401
5	29B3Y	6	568.0	3.059336271348967	185.66118583281767
6	29D2K	1	564.0	12.776684631855687	44.14290688476394
7	29D6U	25	515.4	3.779929741271957	136.35174071425106
8	29D7Y	1	498.0	1.097216878954357	453.8756280112933
9	29D5Y	32	494.59375	3.333776397871856	148.35840529548653
10	29B5Y	8	489.75	3.194528804816955	153.30899482312304
11	29D3Y	6	486.0	4.292936376511636	113.20922496291803
12	29D8U	2	483.0	0.710973555605364	679.3501617493296
13	29D3U	78	460.9230769230769	4.097681999203102	112.48385721798691
14	29D4U	5	460.6	7.9613078132785144	57.854816168742275
15	29D5V	169	460.33727810650885	3.1547638133310736	145.91814327312343
16	29D5	452	459.1305309734513	3.7366575388136694	122.87198551227632
17	29B2Y	6	457.6666666666667	0.9676738607290486	472.9554917623373
18	29D8Y	1	457.0	1.8736019477791932	243.91520330222144

## Hour with the most incidents

#	hour	_col1
1	17	2303
2	16	2081
3	15	1984
4	18	1874
5	14	1670
6	13	1613
7	12	1609
8	19	1554
9	11	1403
10	20	1307
11	10	1231
12	21	1197
13	7	1190
14	8	1163
15	9	1105
16	22	1032
17	6	931
18	23	878



## Epoch & Unix Timestamp Conversion Tools

The current Unix epoch time is **1651525383**

### Convert epoch to human-readable date and vice versa

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **milliseconds**:

**GMT** : Sunday, January 1, 2017 7:29:12 AM

**Your time zone** : Sunday, January 1, 2017 1:29:12 AM [GMT-06:00](#)

**Relative** : 5 years ago

Mon Day Yr Hr Min Sec

**Epoch timestamp:** 1651453270

Timestamp in milliseconds: 1651453270000

**Date and time (GMT):** Monday, May 2, 2022 1:01:10 AM

Date and time (your time zone): Sunday, May 1, 2022 8:01:10 PM GMT-05:00

```
1 SELECT HOUR(from_unixtime(time_local/1000)) AS "hour", COUNT(*)  
2 FROM "traffic-db"."nfd_incidents_xd_seg"  
3 GROUP BY HOUR(from_unixtime(time_local/1000))  
4 ORDER BY COUNT(*) DESC;
```

```

1 SELECT
2     HOUR(from_unixtime(time_local/1000)) AS "hour",
3     COUNT(*) AS "count",
4     AVG(response_time_sec) AS "avg_response_time",
5     AVG(dist_to_seg) AS "avg_dist_to_seg",
6     (AVG(response_time_sec) / AVG(dist_to_seg)) AS "avg_response_to_dist_ratio"
7 FROM "traffic-db"."nfd_incidents_xd_seg"
8 GROUP BY HOUR(from_unixtime(time_local/1000))
9 ORDER BY AVG(response_time_sec) DESC;

```

## Hours with long response time

#	hour	count	avg_response_time	avg_dist_to_seg	avg_response_to_dist_ratio
1	2	638	447.84602917341977	3.778233790810532	118.53317025078664
2	3	582	432.2536496350365	4.029446882839729	107.27369343814459
3	1	631	423.0238500851789	3.6953603886625976	114.47431524757917
4	4	469	416.38137472283813	3.3544287388752054	124.12884790107587
5	5	659	406.78343949044586	3.463554060833292	117.44682841548556
6	6	931	404.27659574468083	3.3443887305931046	120.88205896836195
7	0	661	403.0271132376395	3.7431274657949727	107.67122330739112
8	17	2303	395.8197701149425	3.5040504295669734	112.96063743119629
9	8	1163	395.6030330062444	3.4374118666987163	115.08746939486794
10	23	878	393.0719424460432	3.8952080369031146	100.91166857381899
11	7	1190	390.1402278702892	3.401522727242076	114.69575809261501
12	22	1032	389.10766045548655	3.7528051914190845	103.6844815033816
13	9	1105	388.8916116870877	3.3175795611533494	117.22148768962464
14	11	1403	388.7903703703704	3.1629427600310964	122.92045726636799
15	18	1874	382.4417693169093	3.39152380793513	112.76399370162531
16	16	2081	381.3605236656596	3.5280765646022703	108.0930406930246
17	21	1197	380.4735013032146	3.266796723011731	116.46684307692331
18	15	1984	375.8127637130802	3.4656756327737335	108.43852787581872

```

1 SELECT
2   "description",
3   MIN(app_temp) AS "temp (min)",
4   MAX(app_temp) AS "temp (max)",
5   MIN(clouds) AS "clouds (min)",
6   MAX(clouds) AS "clouds (max)",
7   MIN(snow) AS "snow (min)",
8   MAX(snow) AS "snow (max)",
9   MIN(precip) AS "precip (min)",
10  MAX(precip) AS "precip (max)",
11  MIN(uv) AS "uv (min)",
12  MAX(uv) AS "uv (max)",
13  COUNT(*) AS "count"
14 FROM "traffic-db"."weather-data"
15 GROUP BY "description";

```

1	description	temp (min)	temp (max)	clouds (min)	clouds (max)	snow (min)	snow (max)	precip (min)	precip (max)	uv (min)	uv (max)	count
2	Overcast clouds	-73.2	58.7	71.0	100.0	0.0	12.5	0.0	0.5	0.0	11.2	8112938
3	Clear Sky	-78.6	81.0	0.0	1.0	0.0	12.5	0.0	0.5	0.0	11.3	6043266
4	Moderate rain	-4.6	50.3	0.0	100.0	0.0	12.0	4.1	8.0	0.0	10.0	87065
5	Snow	-50.8	2.3	0.0	100.0	16.099999666214	100.0	1.7	12.0	0.0	3.0	10359
6	Scattered clouds	-76.7	76.8	21.0	40.0	0.0	12.5	0.0	0.5	0.0	11.2	1822752
7	Light snow	-75.9	3.7	0.0	100.0	3.5560001134872	37.5	0.508	5.0	0.0	4.6	30052
8	Heavy snow	-24.3	3.8	0.0	100.0	36.399998664856	487.5	4.3	39.0	0.0	1.5	771
9	Broken clouds	-75.9	69.5	41.0	70.0	0.0	12.5	0.0	0.5	0.0	11.2	2588518
10	Few clouds	-76.9	52.7	2.0	20.0	0.0	12.5	0.0	0.5	0.0	11.2	961175
11	Light rain	-5.9	62.6	0.0	100.0	0.0	8.0	0.508	4.0	0.0	10.1	1096212
12	Mix snow/rain	-7.0	3.4	0.0	100.0	1.0160000324249	37.0	0.508	18.5	0.0	4.3	11133
13	Heavy rain	-4.0	45.7	0.0	100.0	0.0	24.0	8.1	292.1	0.0	9.2	28291

```
✓ [36] !pip install pandas fiona shapely pyproj rtree
```

```
✓ [37] !pip install geopandas
```

```
✓ [61] results = roads.merge(incidents, on="XDSegID")
```

```
✓ [62] results.head(1)
```

	OID	XDSegID	PreviousXD	NextXDSegI	FRC	ToExport	Miles	Lanes	RoadNumber	RoadName	...	longitude	emdCardNumber	time_utc	time_local	response
0	8073455	155711459	NaN	NaN	5	None	0.028749	NaN	BERRY RD	...	-86.770009	29D2L	2020-07-04 03:42:22.387	2020-07-03 22:42:22.387		

1 rows x 40 columns



```
✓ [79] df = results.groupby(['XDSegID']).size().reset_index(name='counts')
```

```
✓ [80] df
```

	XDSegID	counts
0	155711459	1
1	155742183	2
2	155795795	5
3	155800999	1
4	155839691	1
...	...	...
3333	1524643823	15
3334	1524644550	9
3335	1524644585	6
3336	1524645372	4
3337	1524646899	12

3338 rows x 2 columns

```
✓ [81] df.loc[df['counts'].idxmax()]
```

```
XDSegID    441552606
counts      103
Name: 808, dtype: int64
```

## Joining incidents and roads



# Sample Visualization

# Analysis results

---

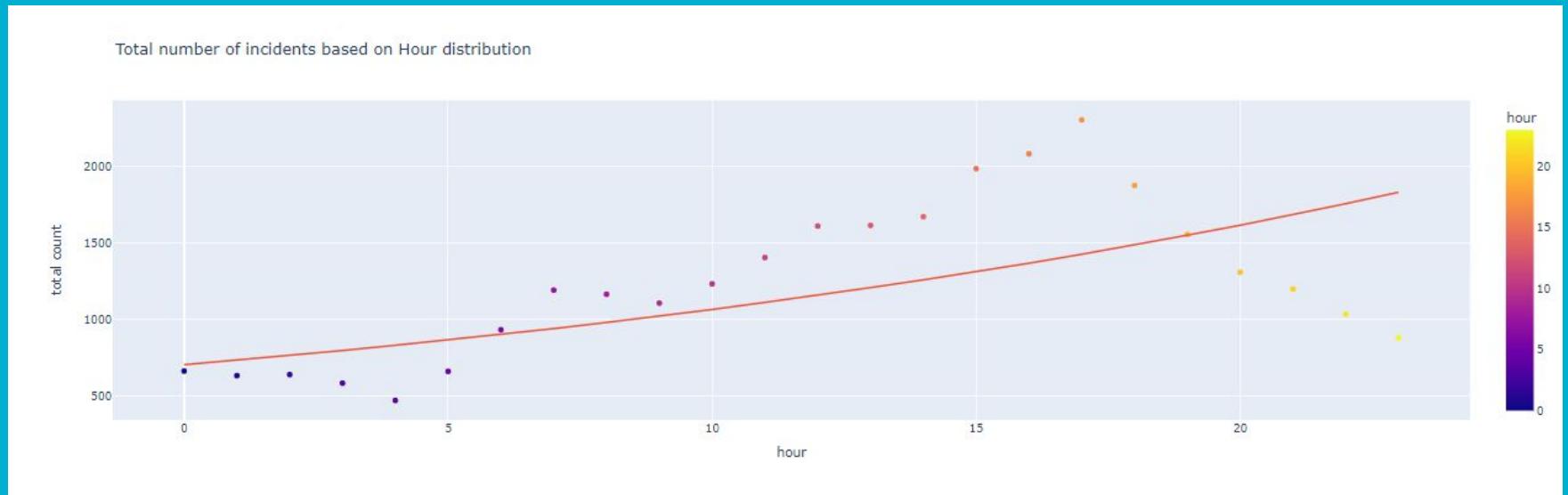
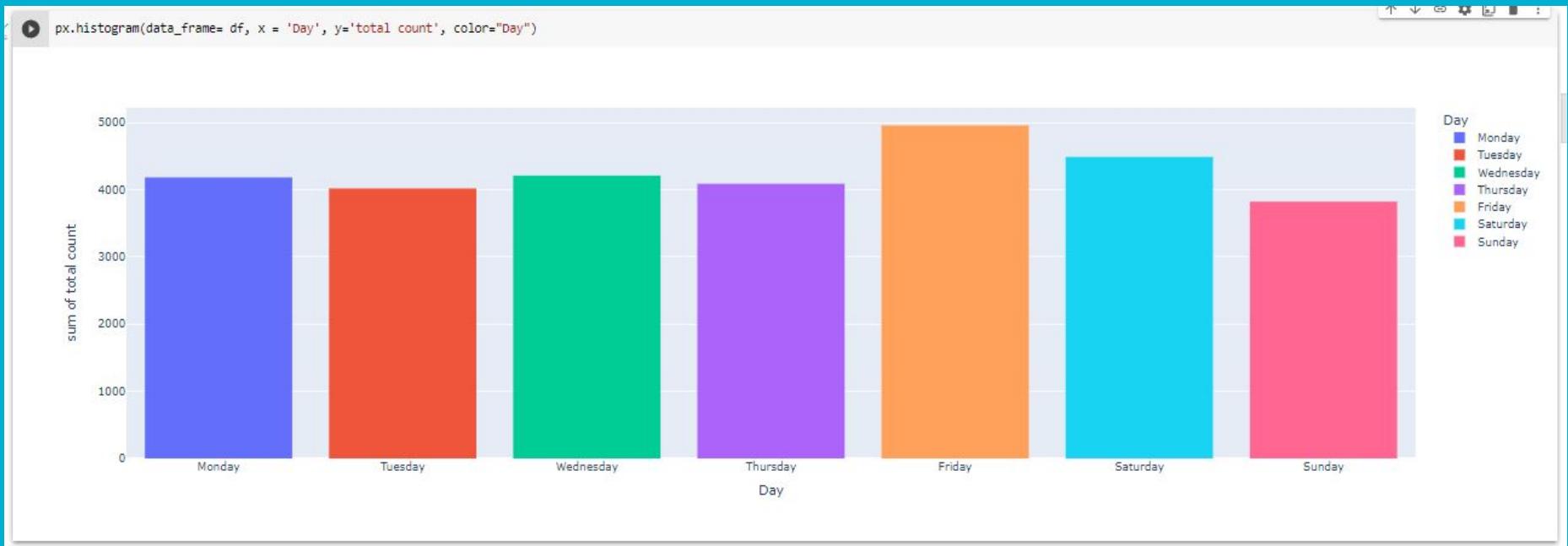
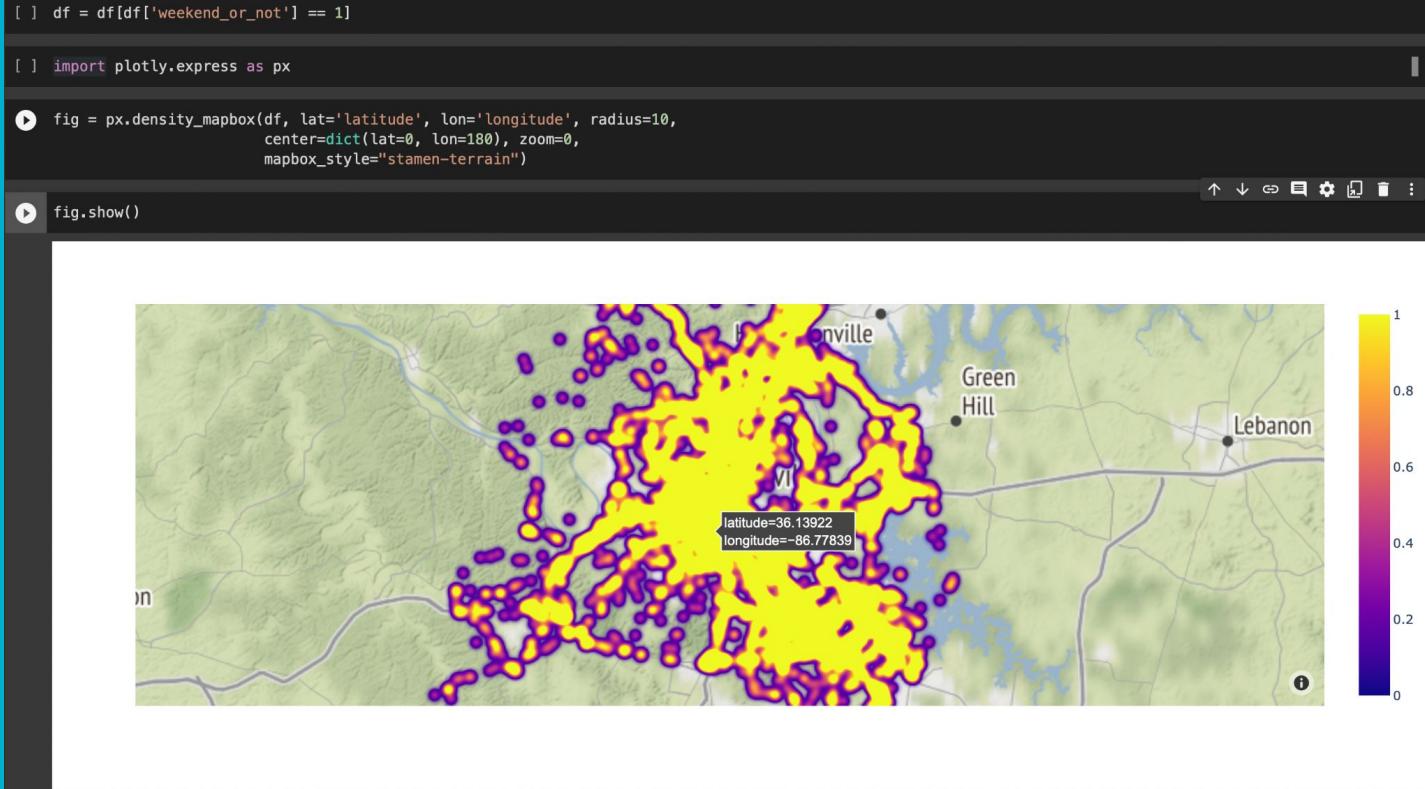


Figure 1. Distribution of total number of incidents based on Hour (0 - 23 hr)

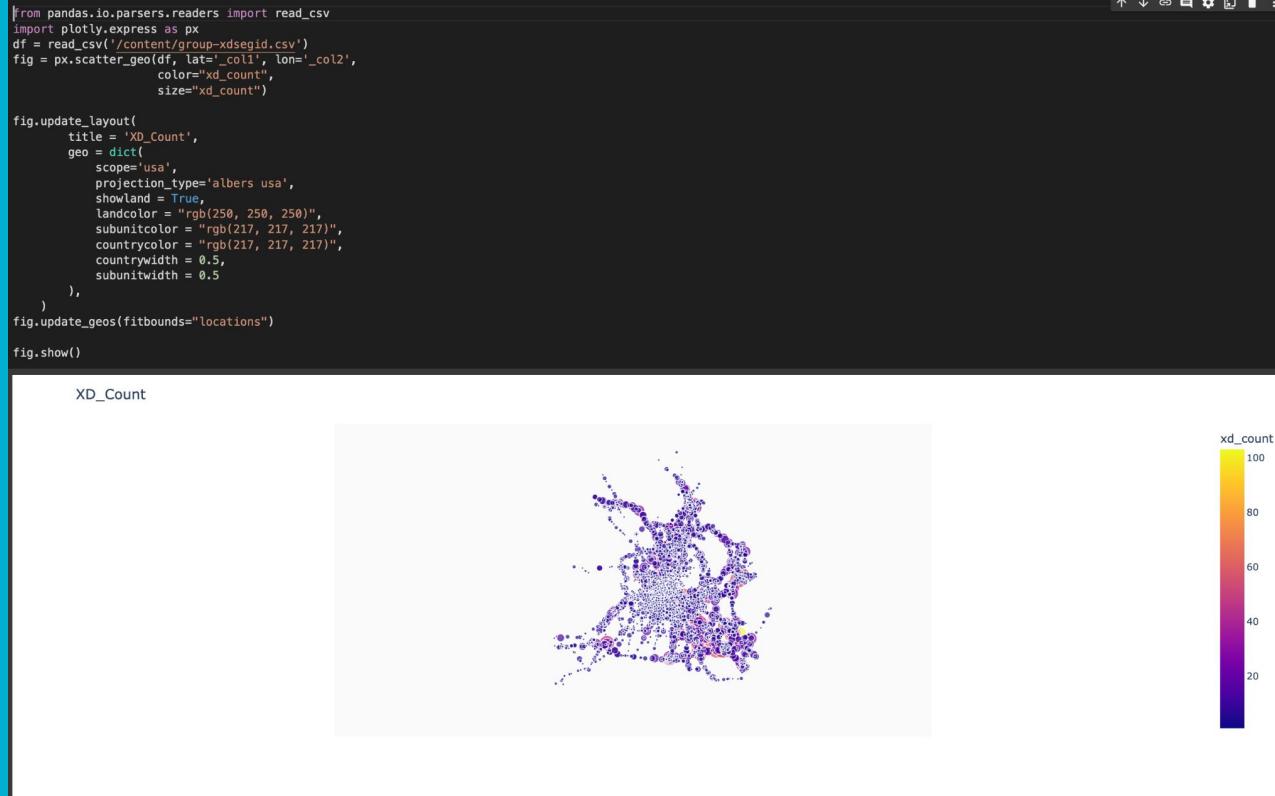
## Figure 2. Total number of incidents in different days of week



## Fig 3. Density Graph (Incidents)



# Fig. 4 Density Graph (roads)



**Thank you!**