City of Austin

Group 5: Harris Azerf, Timothy Wong, Malavika Prasad, Kinjal Doshi, Nadia Florez

OVERVIEW

01 MOTIVATION + DATA STRATEGY

02 DATA ARCHITECTURE + ETL

03 ANALYTICS

04 DATA GOVERNANCE

05 REFLECTIONS

MOTIVATION

Facilitate future city projects and operations and create the most efficient use of resources by leveraging publicly available datasets

DATA STRATEGY

City of Austin needs to provide **accurate** information to allow people and decision-makers to make **informed decisions** about city matters

- A mainly offensive strategy provides timely information for decision-making and a robust infrastructure for varied analytics
- Defensive components protect private government and citizen information
- Evidence-based decisions about city matters ultimately impact the everyday of Austinites

COMPONENTS

DATA

- Weather, traffic, bus stop and sensor locations
- Sensor and device measurements
- Bus route schedule

TRANSACTION MANAGEMENT APPLICATIONS

ENTERPRISE DATA WAREHOUSE

DATA LAKE

Transaction Management Applications

WEATHER

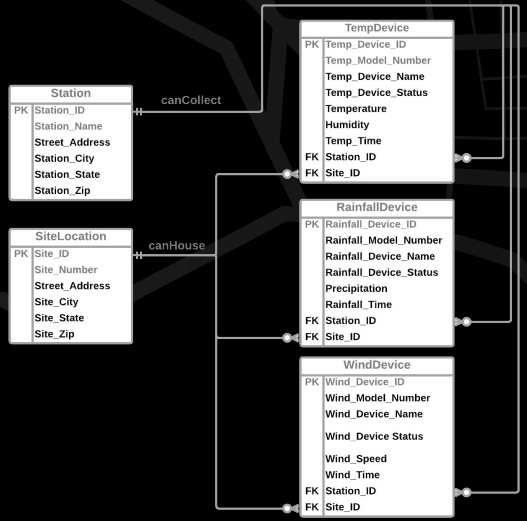
- Rainfall Devices
- Temperature Devices
- Wind Devices

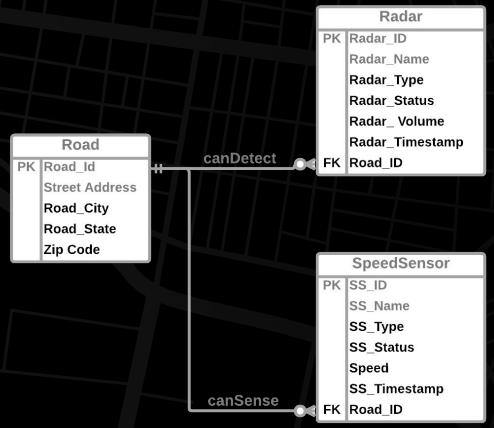
TRAFFIC

- Speed Sensor
- Radar
- Road

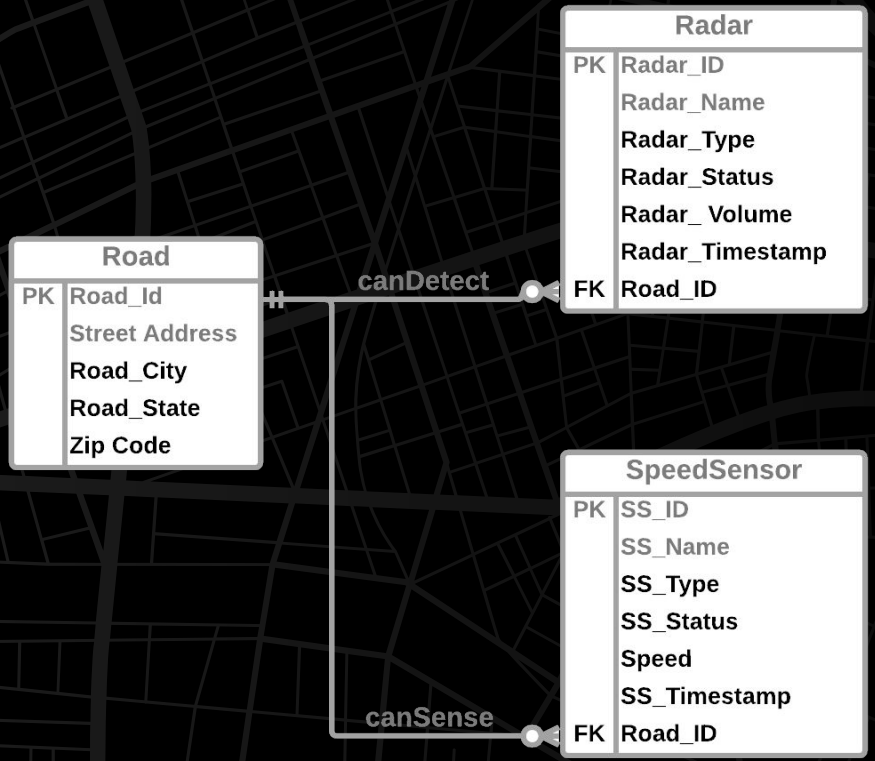
TRANSPORTATION

- Bus Stops
- Bus
- Routes





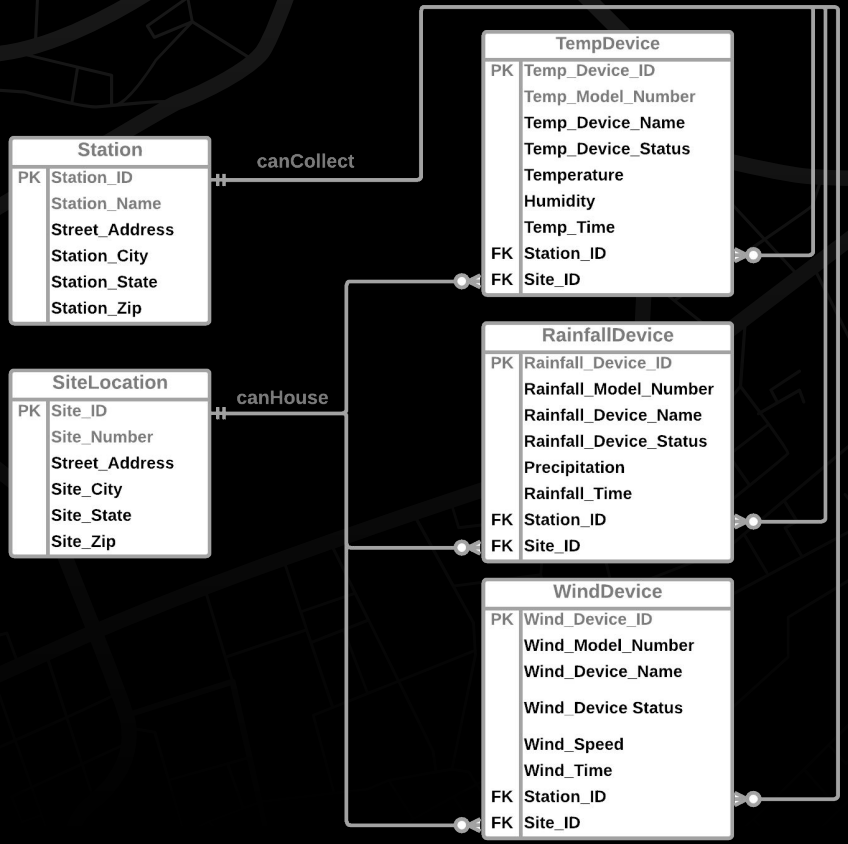
Transaction Management Applications



Traffic

- Devices provide information in different time intervals from different locations
  - Speed Sensor
  - Radar
- Road Location is always the same, so that is placed into its own table

Transaction Management Applications



Weather

- Devices provide information in different time intervals from different locations
  - Rainfall
  - Temperature
  - Wind
- Device Location is always the same
- The Weather Station location that each device is going to is always the same

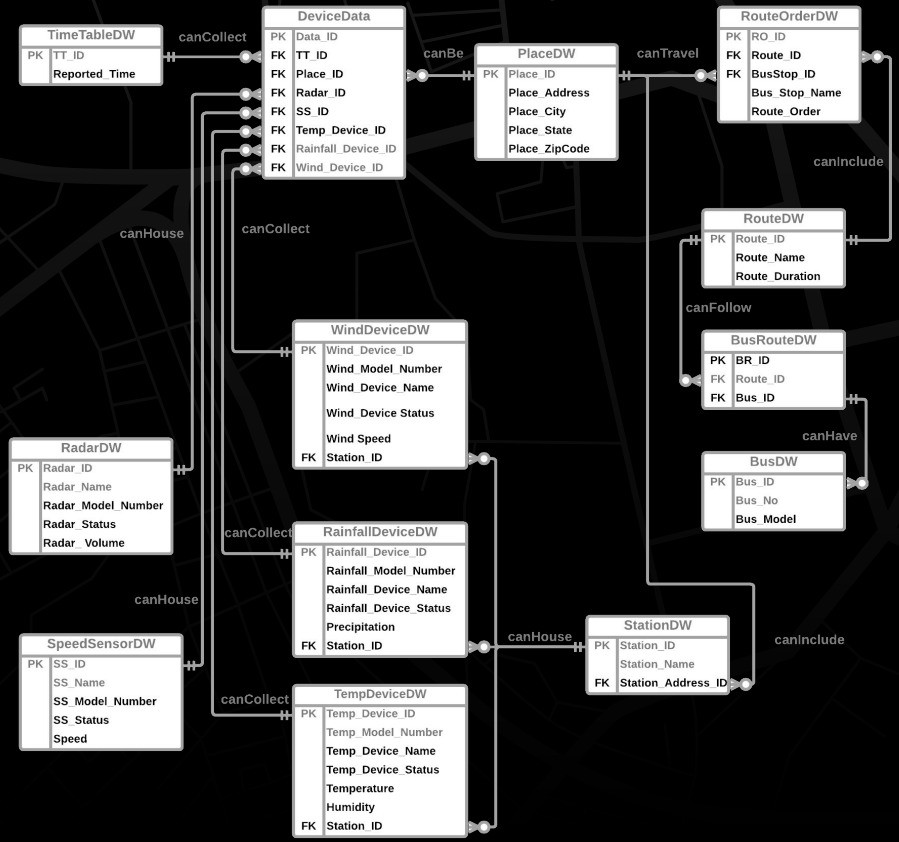
Transaction Management Applications

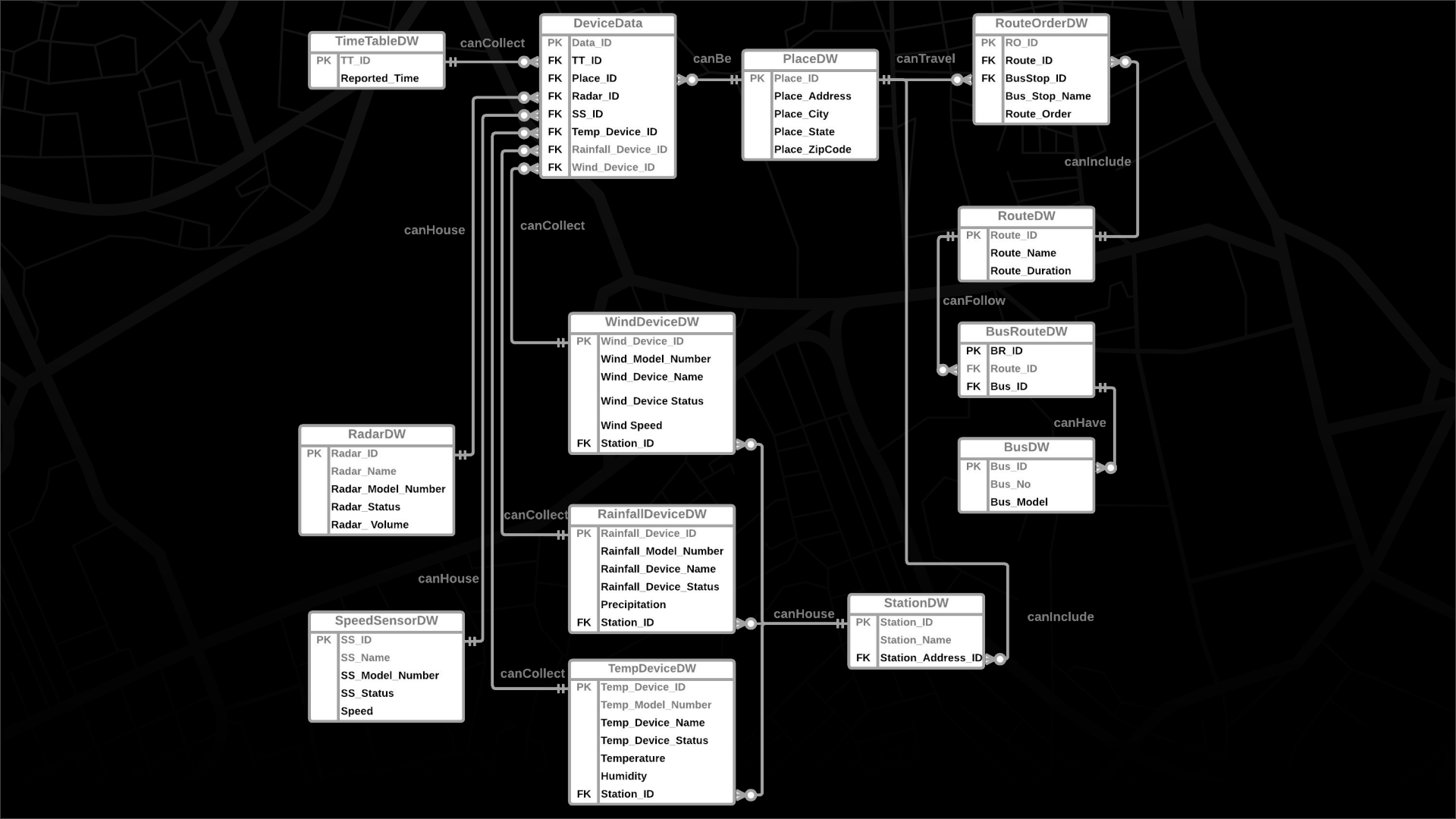
Transportation (Bus)

- Add a bus route schedule
- Find a specific bus route
- Includes information:
  - Which Bus
  - Which Bus Stop
  - Which Route
  - Where in the Route

Enterprise Data Warehouse

- Created a Data Warehouse containing all the information
- 3NF
  - Places (all the different locations)
  - Timestamp




TimeTableDW
PK TT_ID
Reported_Time
canCollect
DeviceData
PK Data_ID
FK TT_ID
FK Place_ID
FK Radar_ID
FK SS_ID
FK Temp_Device_ID
FK Rainfall_Device_ID
FK Wind_Device_ID
canBe
PlaceDW
PK Place_ID
Place_Address
Place_City
Place_State
Place_ZipCode
canTravel
RouteOrderDW
PK RO_ID
FK Route_ID
FK BusStop_ID
Bus_Stop_Name
Route_Order
canInclude
RouteDW
PK Route_ID
Route_Name
Route_Duration
canFollow
BusRouteDW
PK BR_ID
FK Route_ID
FK Bus_ID
canHave
BusDW
PK Bus_ID
Bus_No
Bus_Model
canHouse
canCollect
WindDeviceDW
PK Wind_Device_ID
Wind_Model_Number
Wind_Device_Name
Wind_Device Status
Wind Speed
FK Station_ID
RadarDW
PK Radar_ID
Radar_Name
Radar_Model_Number
Radar_Status
Radar_ Volume
canCollect
RainfallDeviceDW
PK Rainfall_Device_ID
Rainfall_Model_Number
Rainfall_Device_Name
Rainfall_Device_Status
Precipitation
FK Station_ID
canHouse
StationDW
PK Station_ID
Station_Name
FK Station_Address_ID
canInclude
canHouse
SpeedSensorDW
PK SS_ID
SS_Name
SS_Model_Number
SS_Status
Speed
canCollect
TempDeviceDW
PK Temp_Device_ID
Temp_Model_Number
Temp_Device_Name
Temp_Device_Status
Temperature
Humidity
FK Station_ID

ETL Process

Extract

Created Python classes and objects for each transaction application and data warehouse table

Transform

Mapped data for each transaction application to corresponding data warehouse object

Load

Used SQL commands to load data from warehouse objects into the database

ETL Sample Code

```
class SpeedSensor:
  def __init__(self, ss_id, ss_name, ss_type, ss_status, speed, ss_timestamp, road_id):
    self.ss_id = ss_id
    self.ss_name = ss_name
    self.ss_type = ss_type
    self.ss_status = ss_status
    self.speed = speed
    self.ss_timestamp = ss_timestamp
    self.road_id = road_id
```

```
speedsensor_query = 'SELECT * FROM SpeedSensor'
speedsensor_query_list = [speed_sensor for speed_sensor in cursor.execute(speedsensor_query)]
speedsensor_obj_list = []

for row_speedsensor in range(len(speedsensor_query_list)):
  result = speedsensor_query_list[row_speedsensor]
  speedsensor_oltp = SpeedSensor(result[0], result[1], result[2], result[3], result[4], result[5], result[6])
  speedsensor_obj_list.append(speedsensor_oltp)
```

ETL Sample Code

```
def speed_transform(obj, cursor):

    ## get SpeedSensor attributes
    ss_id = getattr(obj, 'ss_id')
    ss_name = getattr(obj, 'ss_name')
    ss_model_num = getattr(obj, 'ss_type')
    ss_status = getattr(obj, 'ss_status')
    ss_volume = getattr(obj, 'speed')
    ss_timestamp = getattr(obj, 'ss_timestamp')
    road_id = getattr(obj, 'road_id')

    ##sql where we get road information
    sql_road_oltp = "SELECT * FROM Road WHERE Road_ID = {}".format(road_id)


    try:

        cursor.execute(sql_road_oltp)
        result = cursor.fetchone() #have to check format of output

    except cx_Oracle.Error as error:
        print('Error occurred when fetching Road information:')
        print(error)

    obj_road = Road(result[0], result[1], result[2], result[3], result[4])
    # get Road attributes
    street_address = getattr(obj_road, 'street_address')
    road_city = getattr(obj_road, 'road_city')
    road_state = getattr(obj_road, 'road_state')
    zip_code = getattr(obj_road, "zip_code")


    speed_dw = SpeedSensorDW(ss_id, ss_name, ss_model_num, ss_status, ss_volume)
    tt_dw = TimeTableDW(ss_timestamp) #we can have that id but not use it
    place_dw = PlaceDW(street_address, road_city, road_state, zip_code)

    return speed_dw, tt_dw, place_dw
```

```
def speed_load(speed, tt, place, cursor):
    #sql for speed
    sql_speed = ('insert into SpeedSensorDW(SS_ID, SS_Name, SS_Model_Number, SS_Status, Speed)'
        'values(:SS_ID,:SS_Name,:SS_Model_Number,:SS_Model_Number,:SS_Volume)')

    sql_place = ('insert into PlaceDW(Place_Address, Place_City, Place_State, Place_ZipCode)'
    'values(:Place_Address,:Place_City,:Place_State, :Place_ZipCode)')

    sql_time = ('insert into TimeTableDW(Reported_Time)'
    'values(:Reported_Time)')



    ############ Loading speed, place and timetable#####################


    try:
      # execute the insert statement
      cursor.execute(sql_speed, [speed.ss_id, speed.ss_name, speed.ss_model_number,
                                 speed.ss_status, speed.speed])
      print("SpeedSensorDW row inserted")


     except cx_Oracle.Error as error:
        print('Error occurred when inserting SpeedSensorDW row:')
        print(error)

    try:
        # execute the insert statement

        cursor.execute(sql_place, [place.place_address, place.place.place_address,
                                   place.place_state, place.place_zip_code])
        print("PlaceDW row inserted")

    except cx_Oracle.Error as error:
            print('Error occurred when inserting placeDW row:')
            print(error)
```

Data Lake

Purpose:

- Help build efficiency for Public Projects
  - Improve road conditions
  - Locating areas affected by weather
  - Replacing multiple defective sensors at the same time
- Give public job functions better access to utilize public data for analytics
  - Reduce Traffic
  - Identify best fuel efficient bus routes

Data Lake Structure:

- Included Location, Weather, and Traffic Data

```
CREATE VIEW Data_Lake AS (
SELECT
    Place_address, place_city, place_state, place_zipcode,
    bus_stop_name, route_order, route_name, route_duration, bus_no, bus_model,
    reported_time,
    radar_name, radar_model_number, radar_status, radar_volume,
    ss_name, ss_model_number, ss_status, speed,
    wind_device_name, wind_model_number, wind_device_status, wind_speed,
    rainfall_device_name, rainfall_model_number, rainfall_device_status, precipitation,
    temp_device_name, temp_model_number, temp_device_status, temperature, humidity

FROM
    placedw        p
    FULL OUTER JOIN routeorderdw   ro ON p.place_id = ro.busstop_id
    FULL OUTER JOIN routedw r ON ro.route_id = r.route_id
    FULL OUTER JOIN busroutedw br ON r.route_id = br.route_id
    FULL OUTER JOIN busdw b ON b.bus_id = br.bus_id
    FULL OUTER JOIN devicedata dd ON p.place_id = dd.place_id
    FULL OUTER JOIN timetabledw tt ON dd.tt_id = tt.tt_id
    FULL OUTER JOIN radardw rad ON rad.radar_id = dd.radar_id
    FULL OUTER JOIN speedsensordw ss ON ss.ss_id = dd.ss_id
    FULL OUTER JOIN winddevicedw w ON w.wind_device_id = dd.wind_device_id
    FULL OUTER JOIN rainfalldevicedw rf ON rf.rainfall_device_id = dd.rainfall_device_id
    FULL OUTER JOIN tempdevicedw td ON td.temp_device_id = dd.temp_device_id
    );
```

Analytics

How are bus routes affected by flooding?

What weather causes the most traffic congestion?

Which devices are down, including both weather and traffic?

Analytics

```
SELECT  bus_stop_name, bus_no, route_name, route_order, route_duration,
        place_address, Place_zipcode,
        precipitation, radar_volume, speed, reported_time
FROM data_lake
WHERE precipitation > 40;
```

How are bus routes affected by flooding?

- Bus Stop Name
- Bus No
- Route Name
- Route Order
- Route Duration
- Place Address
- Place Zip Code
- Precipitation
- Radar Volume
- Speed
- Reported Time

- Routes changes for buses during bad weather conditions
- Find routes that take the longest time during heavy rainfall based on the traffic volume and average speed
- Optimize bus routes for better efficiency

Analytics

What weather causes the most traffic congestion?

- Radar Volume
- Speed
- Place Address
- Place Zip Code
- Precipitation
- Wind Speed
- Temperature
- Humidity
- Reported Time

```
SELECT  radar_volume,  speed,
        place_address, Place_zipcode,
        precipitation, wind_speed, temperature, humidity,
        reported_time
FROM data_lake
Where radar_volume > 30;
```

- Identify areas where severe weather conditions cause traffic
- Build wind barriers based on areas with highest wind speeds
- Locate and fix roads that are flooding

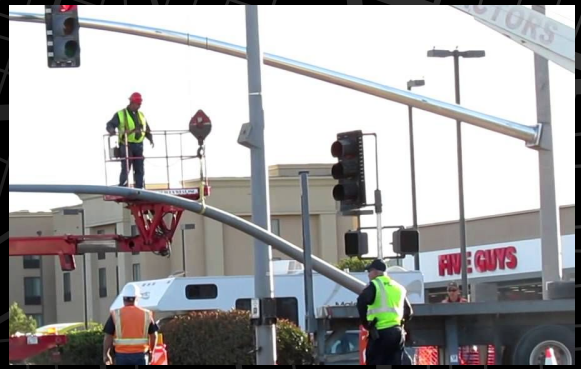
Analytics

Figure out which devices are down, including both weather and traffic

- Place Address
- Place Zip Code
- Device Name
- Device Model Number
- Device Status
- Radar
- Speed Sensor
- Temperature
- Wind Sensor
- Rainfall

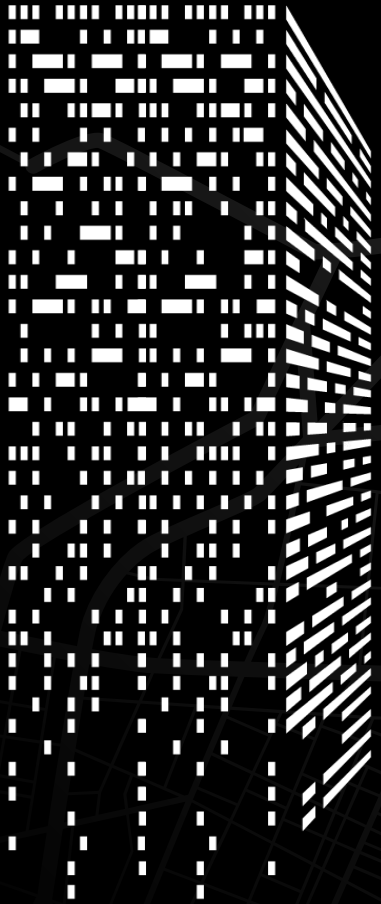
```
SELECT  Place_address, place_city, place_state, place_zipcode,
        radar_name, radar_model_number, radar_status,
        ss_name, ss_model_number, ss_status,
        wind_device_name, wind_model_number, wind_device_status,
        rainfall_device_name, rainfall_model_number, rainfall_device_status,
        temp_device_name, temp_model_number, temp_device_status
FROM data_lake
WHERE   radar_status = 'N'
        OR ss_status = 'N'
        OR wind_device_status = 'N'
        OR rainfall_device_status = 'N'
        OR temp_device_status = 'N';
```

- Improve efficiency of device repairs or replacements by locating similarly placed devices
- Reduce downtime of devices



Data Governance Strategy

- Data is open source and different
  - Agencies are responsible for the data
  - Communication and coordination will help with data
- Location data needs to be consistent, and standardized
  - Automated checks and flags for review



PROJECT REFLECTIONS

Learnings:

- ETL and Data Warehouse is more complicated in terms of consistency and the order of creation since the team didn't have previous experience with it
- Data Modeling is an important step in any organization and it is vital to fully understand the end goal before starting any data structuring

Application & Feedback:

- A real world problem and learning the intricacies of system integration
- Dived deeper into the "analytics" portion
- Reasonable instructions but need more time

THANKS!

Does anyone have any questions?