SF BAY AREA BIKE SHARE

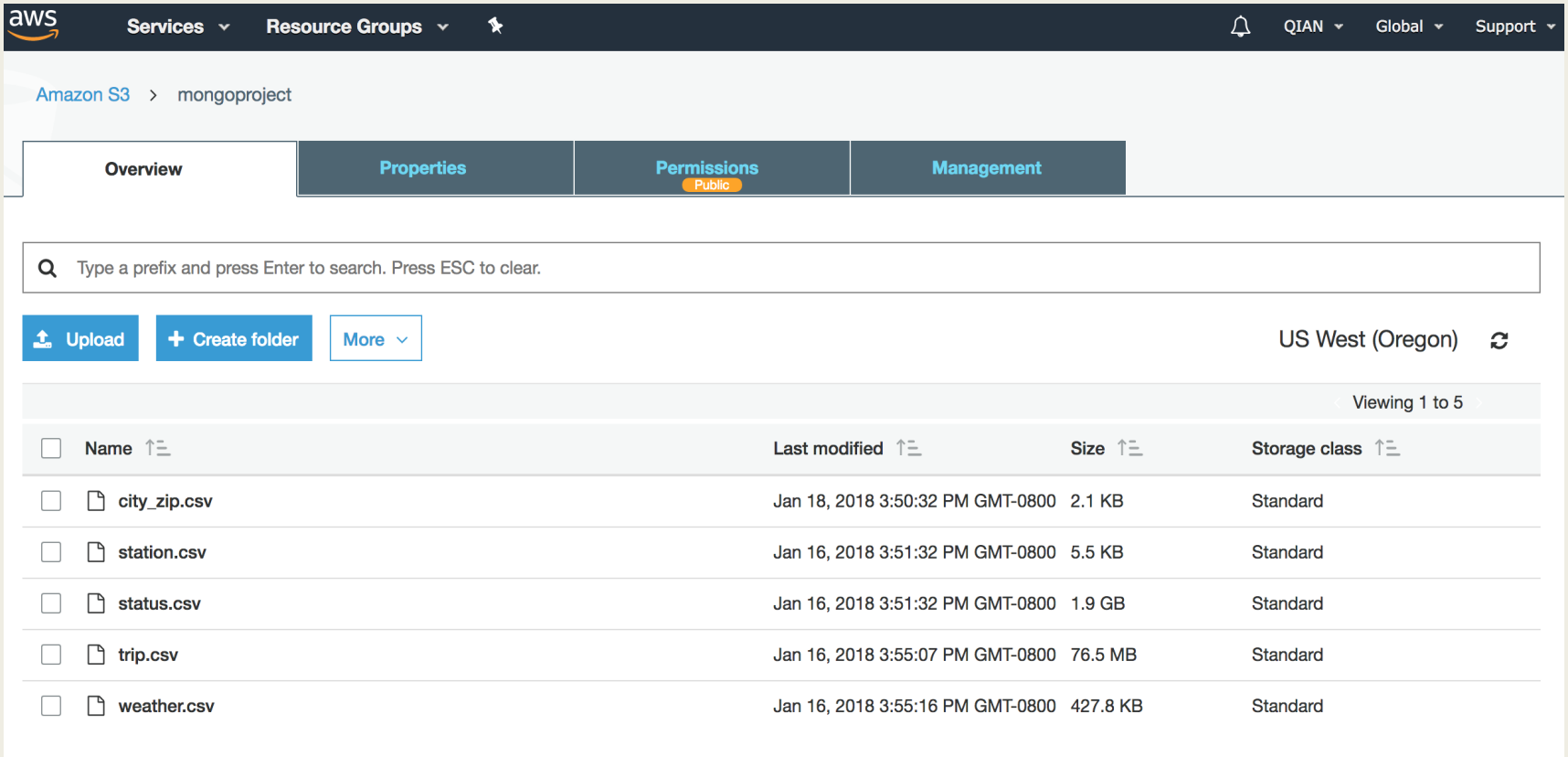
MSAN697

Cara Qin Qian Li Sangyu Shen

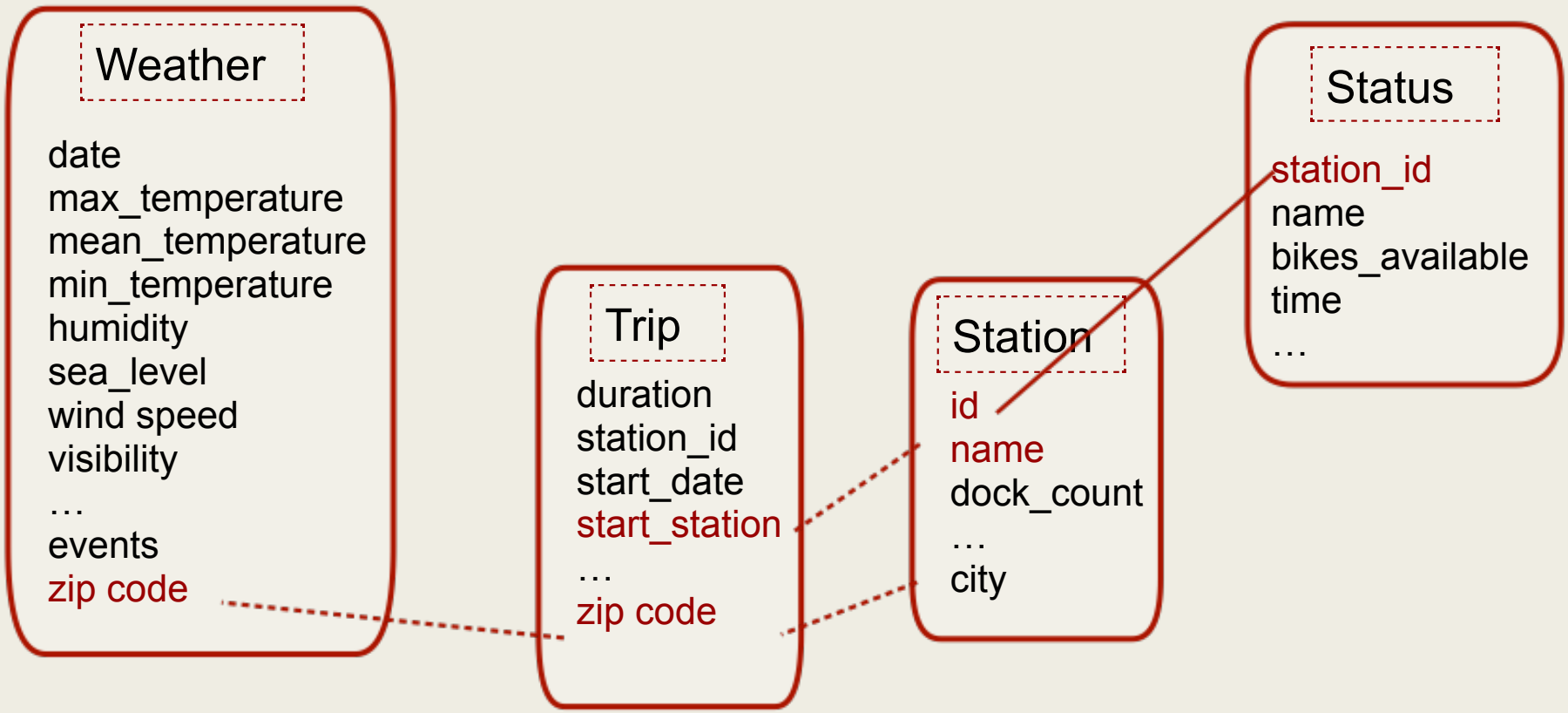
Overview

- Kaggle competition- “SF Bay Area Bike Share”
- The emergence of bike-share system transform the way we travel.
- We want to predict number of daily bike-share trips and analyze how much effect weather has on this number

Launch S3 to store data



Understanding Data Structure



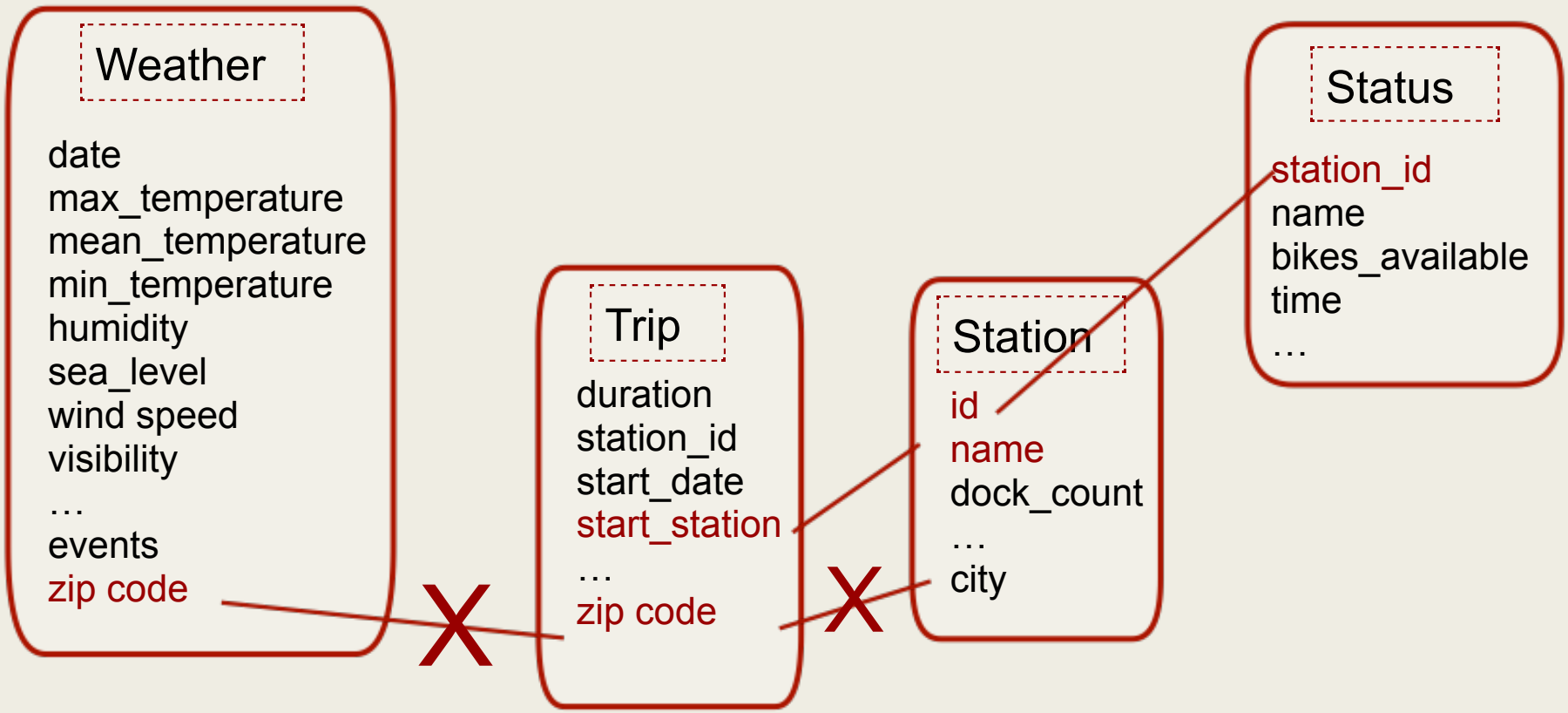
Imported data from S3 into MongoDB on EC2

AWS — ec2-user@ip-172-31-40-253:~ — ssh -i

```
> db.trip.findOne()
{
    "_id" : ObjectId("5a5fa3e005daa8eea48c197b"),
    "id" : 4576,
    "duration" : 63,
    "start_date" : "8/29/2013 14:13",
    "start_station_name" : "South Van Ness at Market",
    "start_station_id" : 66,
    "end_date" : "8/29/2013 14:14",
    "end_station_name" : "South Van Ness at Market",
    "end_station_id" : 66,
    "bike_id" : 520,
    "subscription_type" : "Subscriber",
    "zip_code" : 94127
}
```

94127
not accurate

Understanding Data Structure



Import data from MongoDB on EMR (1 master 2 workers)

jupyter project Last Checkpoint: 8 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Code

In [2]:
```
city_zip_raw = spark.read.format("com.mongodb.spark.sql.DefaultSource")\
                .option("uri","mongodb://54.244.14.54/msan697.city_zip").load()
```

In [3]:
```
city = city_zip_raw['city','zip_code']
```

In [4]:
```
city.show()
```

```
+-------------+--------+
|         city|zip_code|
+-------------+--------+
|San Francisco|   94102|
|San Francisco|   94107|
|San Francisco|   94110|
|San Francisco|   94109|
|San Francisco|   94112|
|San Francisco|   94111|
|San Francisco|   94115|
|San Francisco|   94114|
|San Francisco|   94117|
|San Francisco|   94116|
|San Francisco|   94118|
|San Francisco|   94121|
|San Francisco|   94123|
|San Francisco|   94122|
|San Francisco|   94124|
|San Francisco|   94108|
|San Francisco|   94127|
|San Francisco|   94129|
```

Create New DataFrame

- Column types should be changed

```
31]: trip = spark.read.format("com.mongodb.spark.sql.DefaultSource").

32]: trip.printSchema()

root
 |-- _id: struct (nullable = true)
 |     |-- oid: string (nullable = true)
 |-- bike_id: integer (nullable = true)
 |-- duration: integer (nullable = true)
 |-- end_date: string (nullable = true)
 |-- end_station_id: integer (nullable = true)
 |-- end_station_name: string (nullable = true)
 |-- id: integer (nullable = true)
 |-- start_date: string (nullable = true)
 |-- start_station_id: integer (nullable = true)
 |-- start_station_name: string (nullable = true)
 |-- subscription_type: string (nullable = true)
 |-- zip_code: string (nullable = true)
```

- Assigned defaultly
  - should be deleted

- Type
  - string -- timestamp

Create New DataFrame

step 1 : create schema

```
# trip
trip_raw = spark.read.format("com.mongodb.spark.sql.DefaultSource")\
                .option("uri","mongodb://54.244.14.54/mean697.trip").load()

trip_rdd = trip_raw.rdd

trip_new = trip_rdd.map(lambda x:[str(x[6]), str(x[2]),str(x[7]), \
                                  str(x[9]),str(x[8]),str(x[3]),str(x[5]),\
                                  str(x[4]),str(x[1]),str(x[10]),str(x[11])])

trip = trip_new.map(lambda x: stringToPost_trip(x))

tripdf = sqlContext.createDataFrame(trip, tripSchema)
```

step 2 : create new RDD

step 3 : create new DF

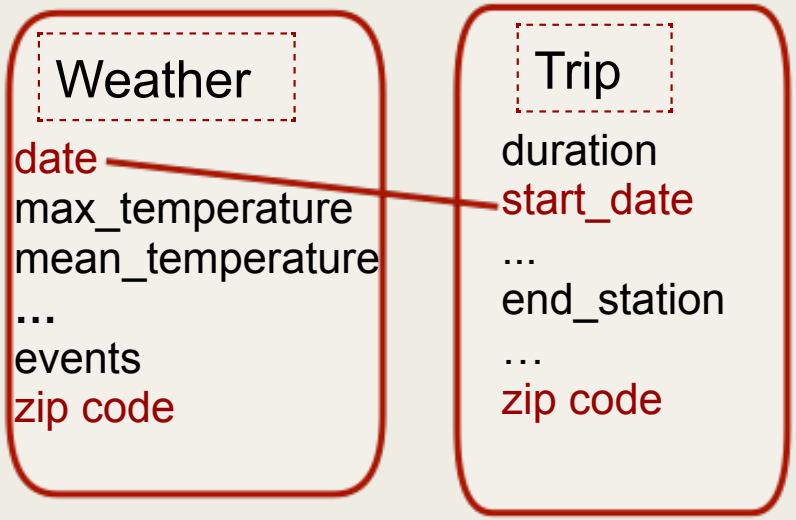
Create New DataFrame

```
tripdf.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- duration: double (nullable = true)
 |-- date: timestamp (nullable = true)
 |-- start_station_name: string (nullable = true)
 |-- start_station_id: long (nullable = true)
 |-- end_date: timestamp (nullable = true)
 |-- end_station_name: string (nullable = true)
 |-- end_station_id: long (nullable = true)
 |-- bike_id: long (nullable = true)
 |-- subscription_type: string (nullable = true)
 |-- zip_code: long (nullable = true)
```

```
weatherdf.printSchema()
```

```
root
 |-- date: date (nullable = true)
 |-- max_temperature_f: float (nullable = true)
 |-- mean_temperature_f: float (nullable = true)
 |-- min_temperature_f: float (nullable = true)
 |-- max_dew_point_f: float (nullable = true)
 |-- mean_dew_point_f: float (nullable = true)
 |-- min_dew_point_f: float (nullable = true)
 |-- max_humidity: float (nullable = true)
 |-- mean_humidity: float (nullable = true)
 |-- min_humidity: float (nullable = true)
 |-- max_sea_level_pressure_inches: float (nullable = true)
 |-- mean_sea_level_pressure_inches: float (nullable = true)
 |-- min_sea_level_pressure_inches: float (nullable = true)
 |-- max_visibility_miles: float (nullable = true)
 |-- mean_visibility_miles: float (nullable = true)
 |-- min_visibility_miles: float (nullable = true)
 |-- max_wind_Speed_mph: float (nullable = true)
 |-- mean_wind_Speed_mph: float (nullable = true)
 |-- max_gust_speed_mph: float (nullable = true)
 |-- precipitation_inches: float (nullable = true)
 |-- cloud_cover: float (nullable = true)
 |-- events: string (nullable = true)
 |-- wind_dir_degrees: float (nullable = true)
 |-- zip_code: long (nullable = true)
```



Extract day of week

```
tripdf.select('date').show()
```

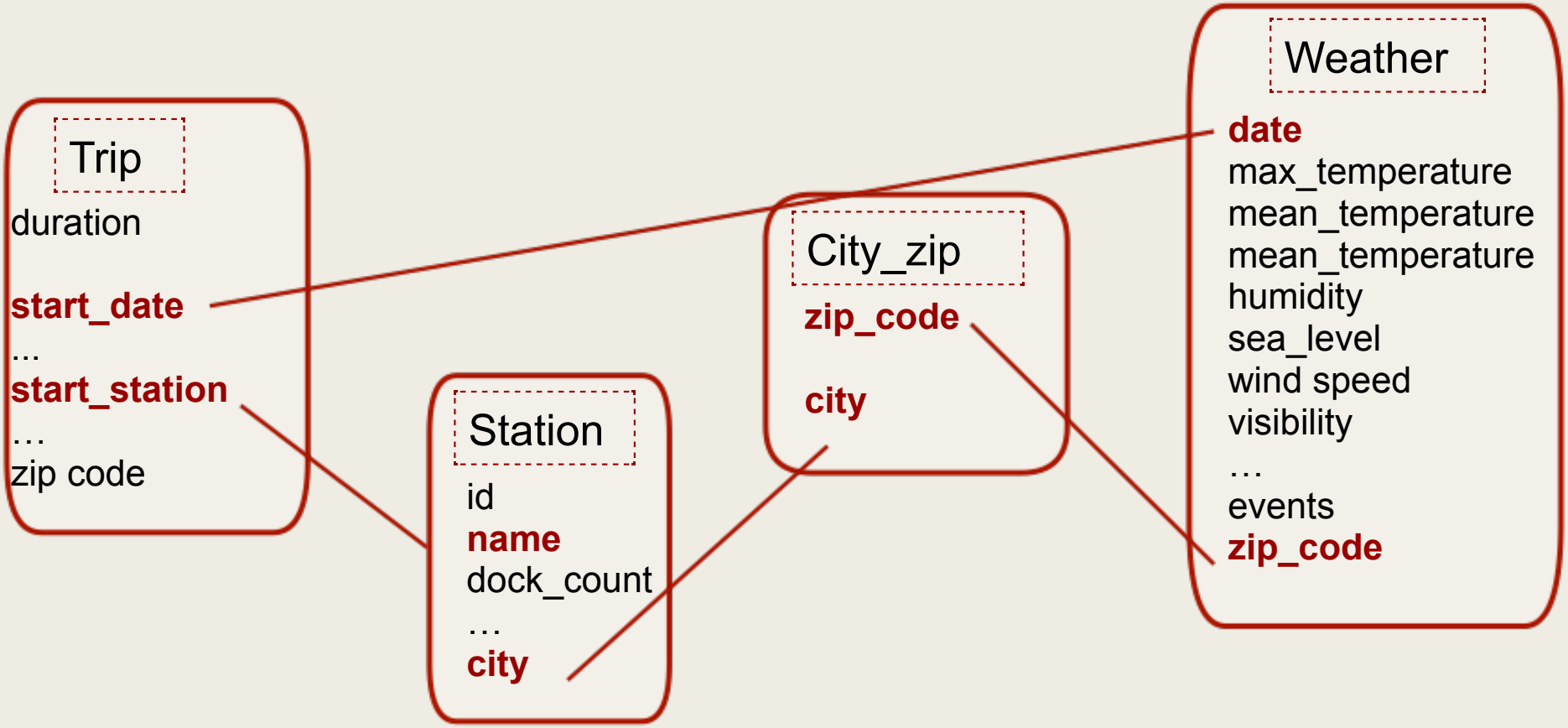
```
+-------------------+
|               date|
+-------------------+
|2013-08-29 14:13:00|
|2013-08-29 14:42:00|
|2013-08-29 10:16:00|
|2013-08-29 11:29:00|
|2013-08-29 12:02:00|
|2013-08-29 18:54:00|
|2013-08-29 13:25:00|
|2013-08-29 14:02:00|
|2013-08-29 17:01:00|
```

```
def dow(x):
    v = x.strftime('%w')
    return v

dow = udf(dow, StringType())
```

```
tripdf = tripdf.withColumn("day_of_week",dow(tripdf['date']))
```

Join DataFrame



SparkSQL

```
weather_city = weatherdf.join(city_zip, 'zip_code').drop('zip_code')
```

```
trip_station = tripdf.join(stationdf, (tripdf.start_station_name == stationdf.name))
```

```
complete_table = trip_station.join(weather_city, ['date', 'city'])
```

```
sqlContext.sql('select count(1) as num_trips, \
                date, city from raw_data\
                group by 2, 3').show(5)
```

```
+---------+----------+-------------+
|num_trips|      date|         city|
+---------+----------+-------------+
|        7|2013-09-04|    Palo Alto|
|        7|2013-09-16|Mountain View|
|      448|2013-10-26|San Francisco|
|      865|2013-10-28|San Francisco|
|       14|2014-01-17|Mountain View|
+---------+----------+-------------+
only showing top 5 rows
```

Response

Imputing & Encoding

OneHotEncode

```
data.printSchema()
```

```
root
 |-- num_trips: long (nullable = false)
 |-- day_of_week: string (nullable = true)
 |-- city: string (nullable = true)
 |-- max_temperature_f: float (nullable = true)
 |-- mean_temperature_f: float (nullable = true)
 |-- min_temperature_f: float (nullable = true)
 |-- max_dew_point_f: float (nullable = true)
 |-- mean_dew_point_f: float (nullable = true)
 |-- min_dew_point_f: float (nullable = true)
 |-- max_humidity: float (nullable = true)
 |-- mean_humidity: float (nullable = true)
 |-- min_humidity: float (nullable = true)
 |-- max_sea_level_pressure_inches: float (nullable = true)
 |-- mean_sea_level_pressure_inches: float (nullable = true)
 |-- min_sea_level_pressure_inches: float (nullable = true)
 |-- max_visibility_miles: float (nullable = true)
 |-- mean_visibility_miles: float (nullable = true)
 |-- min_visibility_miles: float (nullable = true)
 |-- max_wind_Speed_mph: float (nullable = true)
 |-- mean_wind_Speed_mph: float (nullable = true)
 |-- precipitation_inches: float (nullable = true)
 |-- cloud_cover: float (nullable = true)
 |-- events: string (nullable = true)
 |-- wind_dir_degrees: float (nullable = true)
```

Machine Learning

- Training set: 60%
- Validation set: 20%
- Test set: 20%

```
lr = LinearRegression()
evaluator = RegressionEvaluator()

cv = CrossValidator().setEstimator(lr).setEvaluator(evaluator).setNumFolds(5)
paramGrid = ParamGridBuilder().addGrid(lr.regParam, [0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5]).build()

cv.setEstimatorParamMaps(paramGrid)
cvmodel = cv.fit(train)
```

```
RegressionEvaluator().setMetricName("rmse").evaluate(cvmodel.bestModel.transform(valid))
```

```
310.1462084548694
```

```
rf = RandomForestRegressor(maxDepth=20)
evaluator = RegressionEvaluator()

cv = CrossValidator().setEstimator(lr).setEvaluator(evaluator).setNumFolds(5)
paramGrid = ParamGridBuilder().addGrid(rf.maxDepth, [10, 15, 20, 25]).build()

cv.setEstimatorParamMaps(paramGrid)
cvmodel = cv.fit(train)
```

```
RegressionEvaluator().setMetricName("rmse").evaluate(cvmodel.bestModel.transform(valid))
```

```
310.1760822665558
```

```
gbt = GBTRegressor()
evaluator = RegressionEvaluator()
gbtmodel = gbt.fit(train)
gbtpredicts = gbtmodel.transform(valid)
RegressionEvaluator().setMetricName("rmse").evaluate(gbtpredicts)
```

```
171.83903993201034
```

RMSE

Conclusion

The GBT model has a R2 score of 9.4

Most Important features

- San Francisco,
- Mean Wind Speed

```
feat_impo.sortBy(lambda x: x[1], ascending=False).take(10)
```

```
[('san_francisco', 0.2224607674272947),
 ('mean_wind_Speed_mph', 0.075666132179973408),
 ('min_dew_point_f', 0.062387996140445011),
 ('max_wind_Speed_mph', 0.054416037663545698),
 ('max_sea_level_pressure_inches', 0.046911619313796782),
 ('min_humidity', 0.043752369316787332),
 ('cloud_cover', 0.040819665641833432),
 ('max_dew_point_f', 0.039471005872858705),
 ('wind_dir_degrees', 0.037685852085101973),
 ('mean_humidity', 0.036768024362484095)]
```

Lesson learned

- Different operating systems have different characters representing the new line. We cannot discern difference except displaying in vim. The difference will interfere mongoimport command.

- AWS EC2 has its architecture and space distribution that we usually have to go to a memory space much larger than the data we need to store

  t2.small (2GB), t2.medium(4GB), t2.xlarge(16GB) - EBS

  m3.2xlarge, 30GB - 2*80 SSD

- s3 and ec2 put in the same region has higher connection success rate

Lesson Learned

Work with pyspark dataframe efficiently

- Create dataframe with RDD and schema
- *pros: safe, easy to change column type*
- *cons: heavy boring finger workout when the dataframe is wide*

- load data using spark.read.format()
- *pros: generate dataframe automatically without specifying schema*
- *cons: sometimes hard to coerce to desired column type*