Python API

Load some data

```
df = spark \
  .read \
  .option('header','True') \
  .option('delimiter',',') \
  .option('inferSchema','True') \
  .csv('/databricks-datasets/airlines/part-00000')
```

```
display(df)
df.limit(5).show()
display(df.limit(5))
df.printSchema()
```

Reading several files at once: inefficient

```
df = spark \
  .read \
  .option('header','True') \
  .option('delimiter',',') \
  .option('inferSchema','True') \
  .csv('/databricks-datasets/airlines/part-0000*')
```

Reading several files at once: efficient

```
df = spark \
  .read \
  .option('header','True') \
  .option('delimiter',',') \
  .option('inferSchema','True') \
  .csv('/databricks-datasets/airlines/part-00000')

schema = df.schema


df = spark \
  .read \
  .option('header','True') \
  .option('delimiter',',') \
  .schema(schema) \
  .csv('/databricks-datasets/airlines/part-0000*')
```

Reading several files at once: explicit schema

```
from pyspark.sql.types import IntegerType, StringType, StructField,
StructType
schema = StructType([
        StructField("Year",IntegerType(),True),
        StructField("Month",IntegerType(),True),
        StructField("DayofMonth",IntegerType(),True),
        ...
        StructField("IsDepDelayed",StringType(),True)
        ])
```

DataFrame operations

```
display(df.select(['year', 'month', 'dayofmonth', 'arrdelay','depdelay']))
display(df['year', 'month', 'dayofmonth', 'arrdelay','depdelay'])
display(df.groupBy('Month').avg('arrdelay')) #error!
```

Type coercion

```
from pyspark.sql.types import DoubleType
df = df \
      .withColumn("ArrDelay", df['ArrDelay'].cast(DoubleType()))
```

Filtering and aggregating

```
display(df
.filter(df.Origin == 'SAN')
.groupBy('DayOfWeek')
.avg('ArrDelay'))
```

```
display(
  df \
    .filter(df.Origin != 'SAN')\
    .filter(df.DayOfWeek < 3)\
    .groupBy('DayOfWeek')\
    .avg('ArrDelay')
)
```

Filtering and aggregating (cont.)

```
from pyspark.sql.functions import mean, round
display(
  df \
    .filter(df.Origin != 'SAN')\
    .filter(df.DayOfWeek < 3)\
    .groupBy('DayOfWeek')\
    .agg(round(mean('ArrDelay'),1)) \
    .alias('AvgArrDelay')
)
```

Join

```
%sql
use taxidata;
```

```
trips = spark.read.table('yellow_tripdata')
zones = spark.read.table('taxi_zone_lookup')
```

Join (cont.)

```
%sql
SELECT
tz.Borough,
tz.Zone,
yt.tpep_pickup_datetime,
yt.tpep_dropoff_datetime
FROM
yellow_tripdata yt
LEFT JOIN taxi_zone_lookup tz
ON (yt.PULocationID = tz.LocationID);
```

```
result = trips \
            .join(zones, trips.PULocationID == zones.LocationID, 'left') \
            .select(zones.Borough, zones.Zone, trips.tpep_pickup_datetime, trips.tpep_dropoff_datetime)
display(result)
```