**Data Sources**

**>>> df = spark.read.csv("Uber\_trips\_dataset.csv", inferSchema=True, header=True)**

**>>> product = spark.read.csv("product.csv",inferSchema=True, header=True)**

**>>> store = spark.read.csv("store.csv",inferSchema=True, header=True)**

**>>> tx = spark.read.csv("transactions.csv",inferSchema=True, header=True)**

**1. Aggregate, min,max. e.g. Longest trip**

agg(self, \*exprs)

>>> from pyspark.sql import functions as F

>>> df.agg(F.max("Trip\_distance\_m"))

**2. Column name “max\_distance”**

alias(self, alias)

>>> from pyspark.sql.functions import col

>>>df.agg(F.max("Trip\_distance\_m")).select(col("max(Trip\_distance\_m)").alias("max\_distance"))

**3. cache(self)**

>>> df.cache()

**4. Select column names based on a regular expression. E.g. all the columns starting with Long\_ keyword**

colRegex(self, colName)

>>> df.select(df.colRegex("`Long\_[a-z]\*`")).show()

**5. Get result on driver program**

collect(self)

>>> x=df.collect()

How would you print it nicely?

**6. Correlation between two columns**

corr(self, col1, col2, method=None)

>>> df.corr("Trip\_distance\_m", "trip\_fare")

*Note: Currently only supports the Pearson Correlation Coefficient.*

**7. Temporary views to treat a dataframe as table**

createGlobalTempView(self, name)

createTempView(self, name) vs createOrReplcaeTempView(self, name)

**8. Inner Join**

>>> tx.join(product, on=[‘product\_num’], how=’inner’)

OR

>>> tx.join(product, tx.product\_num==product.product\_num, how=’inner’)

**9. Cross Join**

crossJoin(self, other)

**Exercise**: Display the store\_name, product\_name, sales for all the possible combinations of store and product. If there is no sales then display zero.

>>> from pyspark.sql.functions import col

>>> tx.crossJoin(product.select(col("product\_num").alias("p\_num")))

**10. Cross tab**

crosstab(self, col1, col2)

Gives you the frequency of each combination

>>> tx.crosstab("store\_num", "product\_num").show()

**11. Describe a column stats**

>>> tx.describe("amount").show()

**12. Drop a column**

drop(self, \*cols)

>>> df1=tx.join(store, tx.store\_num==store.store\_num, how='inner')

>>> df1.drop(tx.store\_num)

**13. Drop null values**

dropna(self, how='any', thresh=None, subset=None)

>>> tx1 = tx.dropna(subset=["amount", "store\_num"])

**14. Subtract operation**

exceptAll(self,other)

Note: There is a subtract transformation too

>>> store1, store2 = store.randomSplit([0.2,0.8])

>>> store.exceptAll(store1).show()

**15. Execution plan of a dataframe**

explain(self, extended=False)

>>> store1.explain()

>>> store1.explain(extended=True)

**16. Replace null values**

fillna(self, value, subset=None)

>>> tx.fillna({'store\_num':'Unknown Store', ‘product\_num’:’Unknown Product’}).show()

**17. Filter**

filter(condition)

>>> tx.filter(tx.store\_num!="null").show()

>>> tx.where(tx.store\_num!="null").show()

**18. Intersect: to return exactly matching rows in both the dataframes**

intersect(self, other)

>>> store1.intersect(store).show()

**19. Repartition: e.g. store the result in a single file**

coalesce(self, numPartitions)

>>> df.coalesce(1).rdd.getNumPartitions()

>>> df.repartition(1).rdd.getNumPartitions()

**20. Replace a value with another**

replace(self, to\_replace, value=<no value>, subset=None)

>>> tx.replace("s1","store1", ["store\_num"])

**21. Combine two dataframes**

>>> store1.union(store2).show()

Note: There is a UNION ALL too. Whats the difference?

**22. How to use schema of one dataframe to produce another dataframe**

>>> tx\_schema=tx.schema

>>> txf=sc.textFile("transactions2.csv")

>>> from pyspark.sql import Row

>>> tx2=txf.map(lambda x: x.split(",")).map(lambda x: Row(int(x[0]), x[1], x[2], int(x[3])))

>>> tx2df = spark.createDataFrame(tx2,tx\_schema)

>>> tx2df.show()

*Blank doesn’t mean null. Lets replace blank with null*

>>> tx2df.replace("",”null”) # “null” is not null. Check using fillna

>>> tx2df.replace("",None).fillna("unknown").show()