

### Defining Schema

```

from pyspark.sql.types import *
Schema = StructType([
    StructField('Store', StringType(), nullable=True),
    StructField('Store Type', StringType(), nullable=True),
    StructField('Assortment', StringType(), nullable=True),
    StructField('Competition Distance', FloatType(), nullable=True),
    StructField('Competition Opponent Since Month', IntegerType(), nullable=True),
    StructField('Competition Opponent Since Year', IntegerType(), nullable=True),
    StructField('Promo 2', IntegerType(), nullable=True),
    StructField('Promo 2 Since Week', IntegerType(), nullable=True),
    StructField('Promo 2 Since Year', IntegerType(), nullable=True),
    StructField('Promo Interval', StringType(), nullable=True)
])
df = spark.read.option("header", True).schema(Schema).csv('store.csv')
# We can drop invalid rows while reading the dataset by setting the read mode as "DROPMALFORMED"
df_1 = spark.read.option("header", True).option("mode", 'DROPMALFORMED').csv('store.csv')
df.show()

```

Spark does not detect schema itself properly, so we need to define the schema as well for the data set.

### PySpark DataTypes

Type	Size (Byte)	Default	Range (Digits)
byte	1	0	3 Ints
short	2	0	5
int	4	0	10
long	8	0	Lots
floats	4	0.0f	Lots floats
double	8	0.0d	Lots
DecimalType	32	0.0	Lots

### Filtering Data

### String Data Types

StringType	Description
VarcharType(-length)	A variant of StringType which has a length limitation. Data writing will fail if the input string exceeds the length limitation
CharType(-length)	Reading column of type CharType(n) always returns string values of length n. Char type column comparison will pad the short one to the longer length.

### Adding, renaming and removing columns

### Complex Data Types

ArrayType(elementType, containsNull)	Arrays values comprising a sequence of elements
MapType(keyType, valueType, valueContainsNull)	Represents values comprising a set of key-value pairs. The data type of keys is described by keyType and the data type of values is described by valueType. For a MapType value, keys are not allowed to have null values. valueContainsNull is used to indicate if values of a MapType value can have null values.
StructType(fields)	Represents values with the structure described by a sequence of StructFields (fields)

If, elif, else equivalent

```

voter_df.filter(voter_df['name'].isNone)
OR
voter_df.where(~ voter_df._c1.isNone)
voter_df.filter(voter_df.date.year == 1800)
voter_df.where(voter_df['_c0'].contains('VOTE'))
#Multiple Conditions
whereDF = flatte nDF.wh ere ((col("voter_df.d ro p(' unu sed _co -
stN ame ") == " xia ngr ui") | (col("funns) -
tNa me") == " mic hae l")).so rt( ascf
stN ame ")
whereD F.s how (tr unc ate =False)
#Unique Values
voter_df = df.sel ect (df ["VOTER
NAME"]).di sti nct()
# Show the rows with 10 highest IDs in a set
voter_df.o rd erB y(v ote r_d f.R OW
es c() ).s how(10)

```

### User Defined Functions

1. Define a Python method

```
def reverseString(mystr):
return mystr[::-1]
```
2. Wrap the function and store as a variable

```
udfReverseString = udf(reverseString, String Type())
```
3. Use with Spark

```
user_df = user_df.with Column ('Reverse Name',
udfReverseString(user_df.Name))
```

### Using SQL to clean script

```

df.createOrReplaceTempView("table1")
df2 = spark.sql("SELECT field1, field2
FROM table1")

```

```

add1(withColumn
voter_df.withColumn(' -
Near')voter_df.date.year)
year - withColumnRenamed(' Gender',
'Sex')
drop
whereDF = flatte nDF.wh ere ((col("voter_df.d ro p(' unu sed _co -
stN ame ") == " xia ngr ui") | (col("funns) -
tNa me") == " mic hae l")).so rt( ascf
stN ame ")
functions as F
add_n = udf(lambda x, y: x + y,
IntegerType())
# We register a UDF that adds a
column to the DataFrame,
and we cast the id column to an
Integer type.
df.withColumn(' id_ -
offset', add_n( F.l it( 1000),
df.id.c as t(Integer Typ -
e()))

```

### Validating with Joins

```

parsed_df =
spark.read.parquet('parsed_data.parquet')
company_df = spark.r ea d.p arq uet ('c -
omp ani es.p arquet')
verifi ed_df = parsed_df.jo in( com -
pan y_df, parsed_df.co mpany ==
compan y_d f.c ompany)
# This automa tically removes any rows
with a company not in the valid_df !

```

### View data/actions:

```

printSchema(), head(), show(), count(),
columns and describe()

```

show() - Displays/Prints a number of rows in a tabular format. By default it displays 20 rows and to change the default number, you can pass a value to show(n).

where as take(n) returns first n rows as Array of row objects. It is an alias for first().

count() - total rows

```

.when(<if condition>, <then x>)
df.sel ect (df.Name, df.Age,
.when( df.Age >= 18, " Adu lt")
.when( df.Age < 18, " Min or")
.other wise() is like else
df.sel ect (df.Name, df.Age,
.when( df.Age >= 18, " Adu lt")
.other wis e("M ino r")

```

### Remove duplicate rows & replace values

```

dropDuplicates()
test_d f_n o_dup =
tes t_d f.s ele ct( 'Us er_ -
ID' , 'G ender', 'Age').dr opD -
upl ica tes()
fillna()
used to replace null value with
any other value
df.fil lna (va lue ==-9 9,s -
ubset=

```



By datamansam

Published 3rd September, 2022.

Last updated 12th September, 2022.

Page 2 of 3.

Sponsored by Readable.com

Measure your website readability!

<https://readable.com>

### Remove duplicate rows & replace values (cont)

```
> ["Promo2SinceWeek", "Promo2SinceYear").show()
.withColumn(), when()
creating a new column, with value equal to
1 if
Promo2SinceYear > 2000 otherwise 0
df.withColumn("greater_than_2000",
when(df.CompetitionDistance==2000,1).ot-
herwise(0)
.alias('value_desc')).show()
```



By **datamansam**

[cheatography.com/datamansam/](https://cheatography.com/datamansam/)

Published 3rd September, 2022.

Last updated 12th September, 2022.

Page 3 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>