Assignment 2

Prompt Ideas

Prompt:"GenerateANTLRgrammarlexerrulesforbytesizeunitslikeKB,MBandtimedurationlikems,s,m,h" Prompt: "Create a Java class that parses strings like '1.5MB' into bytes"
Prompt:"WriteaWranglerdirectivethataggregatestotalbytesizeandtimedurationfromtwocolumns" Prompt: "How to store values across rows in a Wrangler directive"
Prompt: "Write JUnit tests for parsing time durations like '1h', '30s'"
Prompt:"ExampleusageofTestingRiginWranglerforrecipeexecution"

README

Byte Size & Time Duration Parsers (New)

Wrangler now supports parsing and aggregating byte sizes and time durations using the `aggregate-stats` directive.

Example Recipe:

```wrangler
aggregate-stats :data_transfer_size :response_time total_size_mb total_time_sec
```

Supported Units:

- Byte Size: `B`, `KB`, `MB`, `GB`, `TB`
- Time Duration: `ms`, `s`, `m`, `h`, `d`

Example Values:
- `"1.5MB"` will be parsed as `1572864` bytes
- `"2m"` will be parsed as `120000` milliseconds

Output:
A single row with aggregate values in megabytes and seconds:
- `total_size_mb`: `sum(all sizes in MB)`
- `total_time_sec`: `sum(all times in seconds)`

ByteSize.java

```
package io.cdap.wrangler.api.parser;

publicclassByteSizeextendsToken{
private final long bytes;

publicByteSize(Stringvalue){
super(value);
this.bytes = parseBytes(value);
}

private long parseBytes(String input) {
String unit = input.replaceAll("[0-9.]", "").toLowerCase();
double num = Double.parseDouble(input.replaceAll("[^0-9.]", ""));

switch (unit) {
case "kb": return (long) (num * 1024);
case "mb": return (long) (num * 1024 * 1024);
case "gb": return (long) (num * 1024 * 1024 * 1024);
case"tb":return(long)(num*1024L*1024*1024*1024);
default: return (long) num;
}
}

publiclonggetBytes(){
return bytes;
}
}
```

TimeDuration.java

```
package io.cdap.wrangler.api.parser;

publicclassTimeDurationextendsToken{
private final long milliseconds;

publicTimeDuration(Stringvalue){
super(value);
this.milliseconds = parseMillis(value);
}

private long parseMillis(String input) {
String unit = input.replaceAll("[0-9.]", "").toLowerCase();
double num = Double.parseDouble(input.replaceAll("[^0-9.]", ""));

switch (unit) {
case "s": return (long) (num * 1000);
case "m": return (long) (num * 60 * 1000);
case"h":return(long)(num*3600*1000);
case"d":return(long)(num*24*3600*1000); default:
return (long) num; // assume ms
}
}

publiclonggetMilliseconds(){
return milliseconds;
}
}
```

AggregateStats.java

```
packageio.cdap.wrangler.steps.aggregate;

import io.cdap.wrangler.api.*;
import io.cdap.wrangler.api.parser.*;
import io.cdap.wrangler.api.Row;
import io.cdap.wrangler.api.annotations.Public;
importio.cdap.wrangler.api.executor.ExecutorContext;
import io.cdap.wrangler.api.executor.Store;
import io.cdap.wrangler.api.directive.Directive;
importio.cdap.wrangler.api.directive.DirectiveContext;
import io.cdap.wrangler.api.parser.Text;

import java.util.List;
importjava.util.ArrayList;

/**
* Directive for aggregating byte sizes and time durations.
*/
@Public
publicclassAggregateStatsimplementsDirective{
private String sizeSource;
privateStringtimeSource;
private String sizeTarget;
private String timeTarget;

privateStorestore;

@Override
public UsageDefinition define() {
return UsageDefinition.builder("aggregate-stats")
.addRequiredArg("sizeSource", TokenType.COLUMN_NAME)
.addRequiredArg("timeSource", TokenType.COLUMN_NAME)
.addRequiredArg("sizeTarget", TokenType.TEXT)
.addRequiredArg("timeTarget", TokenType.TEXT)
.build();
}

@Override
publicvoidinitialize(DirectiveContextcontext,List<Argument>args)throwsDirectiveParseException{
sizeSource = ((ColumnName) args.get(0)).value();
timeSource=((ColumnName)args.get(1)).value();
sizeTarget = ((Text) args.get(2)).value();
timeTarget = ((Text) args.get(3)).value();
store = context.getExecutorContext().getStore("aggregate-stats");
}

@Override
publicList<Row>execute(Rowrow,ExecutorContextcontext)throwsDirectiveExecutionException{ Object
sizeVal = row.getValue(sizeSource);
Object timeVal = row.getValue(timeSource);
```

```
long bytes = parseByteValue(sizeVal);
long millis = parseTimeValue(timeVal);

store.add(sizeTarget,bytes);
store.add(timeTarget, millis);

return new ArrayList<>(); // no rows emitted yet
}

@Override
publicList<Row>finalize(ExecutorContextcontext){
long totalBytes = store.get(sizeTarget, 0L);
long totalMillis = store.get(timeTarget, 0L);

Row output = new Row();
output.add(sizeTarget,totalBytes/(1024.0*1024));//MB
output.add(timeTarget, totalMillis / 1000.0); // Seconds

List<Row>results=newArrayList<>();
results.add(output);
return results;
}

privatelongparseByteValue(Objectvalue){ if
(value instanceof String) {
return new io.cdap.wrangler.api.parser.ByteSize((String) value).getBytes();
}
return Long.parseLong(value.toString());
}

privatelongparseTimeValue(Objectvalue){ if
(value instanceof String) {
return new io.cdap.wrangler.api.parser.TimeDuration((String) value).getMilliseconds();
}
return Long.parseLong(value.toString());
}
}
```

Assignment 1

wrangler-extension/prompts.txt

```
# AI-Generated Prompts Log
# This file records the prompts used to generate parts of the Wrangler extension project
using AI tooling.

## Grammar
Prompt:"GenerateANTLRgrammarlexerrulesforbytesizeunitslikeKB,MBandtime duration like ms, s, m,
h"

## Parser Classes
Prompt: "Create a Java class that parses strings like '1.5MB' into bytes"
Prompt:"CreateaJavaclassthatparsestimestringslike'2h','30m','45s'intomilliseconds"

## Directive
Prompt:"WriteaWranglerdirectivethataggregatestotalbytesizeandtimeduration from two
columns"

## Persistence
Prompt: "How to store and retrieve state across rows in a Wrangler directive"

## Unit Testing
Prompt:"WriteJUnittestsforparsingbytesizestringslike'1GB','512KB',etc." Prompt: "Write
JUnit tests for parsing time durations like '1h', '30s'"

## Integration Testing
Prompt: "Example usage of TestingRig in Wrangler for recipe execution"
Prompt:"Writeatesttoverifytheaggregate-statsdirectiveoutputscorrecttotals from rows"

## Documentation
Prompt:"WriteREADMEusageinstructionsforaWranglerdirectivethatparsesbyteand time units"
```

wrangler-extension/README.md

```
# Wrangler Extension

## Byte Size & Time Duration Parsers

Wranglernowsupportsparsingandaggregatingbytesizesandtimedurationsusingthe
`aggregate-stats`directive.

### Example Recipe

```
aggregate-stats :data_transfer_size :response_time total_size_mb total_time_sec
```

Supported Units
```

```
- Byte Size: B, KB, MB, GB, TB
- TimeDuration:ms,s,m,h,d ###

Output

A single row with aggregate values in megabytes and seconds.
- `total_size_mb`: total bytes converted to MB
- `total_time_sec`: total time in seconds
```

wrangler-extension/grammar/Directives.g4

```
//UpdatedANTLRgrammarrules
value
    : ... // existing alternatives
    | BYTE_SIZE
    | TIME_DURATION
    ;

BYTE_SIZE: DIGITS BYTE_UNIT;
TIME_DURATION:DIGITSTIME_UNIT;

fragment BYTE_UNIT: [kKmMgGtTpPeE]? [bB];
fragment TIME_UNIT: ('ms' | 's' | 'm' | 'h' | 'd');
fragment DIGITS: [0-9]+ ('.' [0-9]+)?;
```

wrangler-extension/wrangler-api/parser/ByteSize.java

```
package io.cdap.wrangler.api.parser;

publicclassByteSizeextendsToken{
  private final long bytes;

  publicByteSize(Stringvalue){ super(value);
    this.bytes = parseBytes(value);
  }

  private long parseBytes(String input) {
    String unit = input.replaceAll("[0-9.]", "").toLowerCase();
    doublenum=Double.parseDouble(input.replaceAll("[^0-9.]",""));

    switch (unit) {
      case "kb": return (long) (num * 1024);
      case "mb": return (long) (num * 1024 * 1024);
      case "gb": return (long) (num * 1024 * 1024 * 1024);
      case"tb":return(long)(num*1024L*1024*1024*1024); default: return
      (long) num;
    }
  }

  publiclonggetBytes(){ return
    bytes;
```

```
  }
}
```

wrangler-extension/wrangler-api/parser/TimeDuration.java

```
package io.cdap.wrangler.api.parser;

publicclassTimeDurationextendsToken{
  private final long milliseconds;

  publicTimeDuration(Stringvalue){
    super(value);
    this.milliseconds = parseMillis(value);
  }

  private long parseMillis(String input) {
    String unit = input.replaceAll("[0-9.]", "").toLowerCase();
    doublenum=Double.parseDouble(input.replaceAll("[^0-9.]",""));

    switch (unit) {
      case "s": return (long) (num * 1000);
      case "m": return (long) (num * 60 * 1000);
      case"h":return(long)(num*3600*1000);
      case"d":return(long)(num*24*3600*1000); default:
      return (long) num;
    }
  }

  publiclonggetMilliseconds(){ return
    milliseconds;
  }
}
```

wrangler-extension/wrangler-core/src/main/java/io/cdap/wrangler/steps/aggregate/AggregateStats.j

```
packageio.cdap.wrangler.steps.aggregate;

import io.cdap.wrangler.api.*;
importio.cdap.wrangler.api.parser.*;
import io.cdap.wrangler.api.Row;
import io.cdap.wrangler.api.annotations.Public;
importio.cdap.wrangler.api.executor.ExecutorContext;
import io.cdap.wrangler.api.executor.Store;
import io.cdap.wrangler.api.directive.Directive;
importio.cdap.wrangler.api.directive.DirectiveContext;
import io.cdap.wrangler.api.parser.Text;

import java.util.List;
importjava.util.ArrayList;

@Public
public class AggregateStats implements Directive {
```

```
  privateStringsizeSource;
  privateStringtimeSource;
  privateStringsizeTarget;
  privateStringtimeTarget;
  private Store store;

  @Override
  public UsageDefinition define() {
    return UsageDefinition.builder("aggregate-stats")
      .addRequiredArg("sizeSource", TokenType.COLUMN_NAME)
      .addRequiredArg("timeSource", TokenType.COLUMN_NAME)
      .addRequiredArg("sizeTarget", TokenType.TEXT)
      .addRequiredArg("timeTarget", TokenType.TEXT)
      .build();
  }

  @Override
     publicvoidinitialize(DirectiveContextcontext,List<Argument>args)throws
DirectiveParseException {
    sizeSource=((ColumnName)args.get(0)).value();
    timeSource=((ColumnName)args.get(1)).value();
    sizeTarget = ((Text) args.get(2)).value();
    timeTarget = ((Text) args.get(3)).value();
    store = context.getExecutorContext().getStore("aggregate-stats");
  }

  @Override
       public    List<Row>    execute(Row    row,    ExecutorContext    context)    throws
DirectiveExecutionException {
    ObjectsizeVal=row.getValue(sizeSource); Object
    timeVal = row.getValue(timeSource);

    long bytes = parseByteValue(sizeVal);
    longmillis=parseTimeValue(timeVal);

    store.add(sizeTarget, bytes);
    store.add(timeTarget,millis);

    return new ArrayList<>();
  }

  @Override
  publicList<Row>finalize(ExecutorContextcontext){ long
    totalBytes = store.get(sizeTarget, 0L);
    long totalMillis = store.get(timeTarget, 0L);

    Row output = new Row();
    output.add(sizeTarget,totalBytes/(1024.0*1024)); output.add(timeTarget,
    totalMillis / 1000.0);

    List<Row>results=newArrayList<>();
    results.add(output);
    return results;
  }
```

```
  privatelongparseByteValue(Objectvalue){ if
    (value instanceof String) {
      return new io.cdap.wrangler.api.parser.ByteSize((String) value).getBytes();
    }
    return Long.parseLong(value.toString());
  }

  privatelongparseTimeValue(Objectvalue){ if
    (value instanceof String) {
                        return    new    io.cdap.wrangler.api.parser.TimeDuration((String)
value).getMilliseconds();
    }
    return Long.parseLong(value.toString());
  }
}
```

wrangler-extension/wrangler-core/src/main/resources/META-INF/services/io.cdap.wrangler.api.dire

```
io.cdap.wrangler.steps.aggregate.AggregateStats
```

wrangler-extension/wrangler-core/src/test/java/io/cdap/wrangler/parser/ByteSizeTest.java

```
package io.cdap.wrangler.parser;

importio.cdap.wrangler.api.parser.ByteSize;
import org.junit.Assert;
import org.junit.Test;

publicclassByteSizeTest{ @Test
  public void testByteParsing() {
    Assert.assertEquals(1024, new ByteSize("1KB").getBytes());
    Assert.assertEquals(1572864, new ByteSize("1.5MB").getBytes());
    Assert.assertEquals(1, new ByteSize("1b").getBytes());
    Assert.assertEquals(1073741824,newByteSize("1GB").getBytes());
  }
}
```

wrangler-extension/wrangler-core/src/test/java/io/cdap/wrangler/parser/TimeDurationTest.java

```
package io.cdap.wrangler.parser;

importio.cdap.wrangler.api.parser.TimeDuration;
import org.junit.Assert;
import org.junit.Test;

publicclassTimeDurationTest{
  @Test
  public void testTimeParsing() {
    Assert.assertEquals(150,newTimeDuration("150ms").getMilliseconds());
    Assert.assertEquals(2000, new TimeDuration("2s").getMilliseconds());
```

```
    Assert.assertEquals(120000, new TimeDuration("2m").getMilliseconds());
    Assert.assertEquals(7200000,newTimeDuration("2h").getMilliseconds());
  }
}
```

wrangler-extension/wrangler-core/src/test/java/io/cdap/wrangler/steps/aggregate/AggregateStatsTe

```
package io.cdap.wrangler.steps.aggregate;

importio.cdap.wrangler.TestingRig;
import io.cdap.wrangler.api.Row;
import org.junit.Assert;
import org.junit.Test;

importjava.util.Arrays;
import java.util.List;

publicclassAggregateStatsTest{ @Test
  publicvoidtestAggregation()throwsException{
    List<Row> rows = Arrays.asList(
      new Row("data_transfer_size", "1MB").add("response_time", "1s"),
      newRow("data_transfer_size","2MB").add("response_time","500ms")
    );

    String[] recipe = new String[] {
      "aggregate-stats :data_transfer_size :response_time total_size_mb total_time_sec"
    };

    List<Row>result=TestingRig.execute(recipe,rows); Assert.assertEquals(1,

    result.size());
    Row agg = result.get(0);

    Assert.assertEquals(3.0, (Double) agg.getValue("total_size_mb"), 0.001);
    Assert.assertEquals(1.5,(Double)agg.getValue("total_time_sec"),0.001);
  }
}
```