Assignment Plan: Enhance Wrangler with Byte Size and Time Duration Parsers

---

Step to Complete the Assignment

1. Fork & Setup

- Fork the repo: https://github.com/data-integrations/wrangler
- Clone it locally and set up using Maven:

```
git clone https://github.com/YOUR_USERNAME/wrangler.git
cd wrangler
mvn clean install
```

---

2. Grammar Modifications – ANTLR Lexer & Parser

Add Lexer Rules:

- Byte units

```
BYTE_UNIT: ('B' | 'KB' | 'MB' | 'GB' | 'TB' | 'PB');
BYTE_SIZE: DIGITS BYTE_UNIT;
```

- Time units

```
TIME_UNIT: ('ns' | 'us' | 'ms' | 's' | 'm' | 'h');
TIME_DURATION: DIGITS TIME_UNIT;
```

- Helper

```
fragment DIGITS: [0-9]+ ('.' [0-9]+)?;
```

Modify Parser Rules:

```
byteSizeArg: BYTE_SIZE;
timeDurationArg: TIME_DURATION;
```

Regenerate Parser:

```
mvn compile
```

---

3. Create Token Classes (wrangler-api)

ByteSize.java

```
public class ByteSize extends Token {
    private final long bytes;
    public ByteSize(String value) {
        this.bytes = parseBytes(value);
    }
    private long parseBytes(String value) {
        value = value.trim().toUpperCase();
        if (value.endsWith("KB")) return (long)(Double.parseDouble(value.replace("KB", "")) *
1024);
        if (value.endsWith("MB")) return (long)(Double.parseDouble(value.replace("MB", "")) *
1024 * 1024);
        GB, TB
        return Long.parseLong(value.replace("B", ""));
    }
    public long getBytes() {
        return bytes;
    }
}
```

TimeDuration.java

```
public class TimeDuration extends Token {
```

```
    private final long nanoseconds;
    public TimeDuration(String value) {
        this.nanoseconds = parseDuration(value);
    }
    private long parseDuration(String value) {
        value = value.trim().toLowerCase();
        if (value.endsWith("ms")) return (long)(Double.parseDouble(value.replace("ms", "")) *
1_000_000);
        if (value.endsWith("s")) return (long)(Double.parseDouble(value.replace("s", "")) *
1_000_000_000);
        return Long.parseLong(value);
    }
    public long getNanoseconds() {
        return nanoseconds;
    }
}
```

Register New Token Types

In TokenType.java, add:

- BYTE_SIZE,
- TIME_DURATION

---

4. Core Parser Updates (wrangler-core)

In visitor (e.g., RecipeVisitor.java):

```
@Override
```

```
public Token visitByteSizeArg(DirectivesParser.ByteSizeArgContext ctx) {
    return new ByteSize(ctx.getText());
}

@Override
public Token visitTimeDurationArg(DirectivesParser.TimeDurationArgContext ctx) {
    return new TimeDuration(ctx.getText());
}
```

---

5. Implement aggregate-stats Directive

AggregateStats.java *(implements Directive)*

- Read 4 arguments:
    - Input: **data_transfer_size, response_time**
    - Output: **total_size_mb, total_time_sec**
- Store totals in ExecutorContext.Store
- In execute(...):
    - Sum converted values per row
        - Finalize:
        - Convert total bytes → MB
        - Convert total time → seconds

---

6. Testing

Unit Tests:

- ByteSizeTest.java and TimeDurationTest.java
- Example cases: 10KB, 1.5MB, 2s, 100ms

Parser Tests:

- Update GrammarBasedParserTest.java with new syntax

Directive Test:

- Use TestingRig and assert the aggregate values:

```
Assert.assertEquals(expectedSizeMB, row.getValue("total_size_mb"), 0.001);
Assert.assertEquals(expectedTimeSec, row.getValue("total_time_sec"), 0.001);
```

---

7. Final Touches

- prompts.txt: Save prompts you used with AI tools
- README.md: Add usage for:

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