Unit -Test Cases:-

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

import io.cdap.wrangler.api.Arguments;
import io.cdap.wrangler.api.DirectiveContext;
import io.cdap.wrangler.api.ExecutorContext;
import io.cdap.wrangler.api.Row;
import org.junit.jupiter.api.Test;

import java.util.*;

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

class AggregateSizeTimeTest {

  private Arguments mockArgs(Map<String, String> args) {
    Arguments arguments = mock(Arguments.class);
    for (Map.Entry<String, String> entry : args.entrySet()) {
      when(arguments.value(entry.getKey())).thenReturn(entry.getValue());
      when(arguments.optional(eq(entry.getKey()), anyString())).thenReturn(entry.getValue());
    }
    return arguments;
  }

  private List<Row> sampleRows() {
    List<Row> rows = new ArrayList<>();
    rows.add(new Row().add("size", "1MB").add("duration", "1s"));
    rows.add(new Row().add("size", "512KB").add("duration", "500ms"));
```

```
    rows.add(new Row().add("size", "1024B").add("duration", "250000000ns")); // 250ms
    return rows;
}

@Test
void testTotalAggregation() {
    AggregateSizeTime directive = new AggregateSizeTime();

    Map<String, String> args = new HashMap<>();
    args.put("sizeColumn", "size");
    args.put("timeColumn", "duration");
    args.put("resultSizeColumn", "totalSize");
    args.put("resultTimeColumn", "totalTime");
    args.put("sizeUnit", "MB");
    args.put("timeUnit", "s");
    args.put("aggType", "total");

    directive.initialize(mock(DirectiveContext.class), mockArgs(args));
    List<Row> result = directive.execute(sampleRows(), mock(ExecutorContext.class));

    Row row = result.get(0);
    double totalSizeMB = (1 * 1024 * 1024 + 512 * 1024 + 1024) / (1024.0 * 1024);
    double totalTimeSec = (1_000_000_000 + 500_000_000 + 250_000_000) / 1_000_000_000.0;

    assertEquals(1, result.size());
    assertEquals(totalSizeMB, row.getValue("totalSize"));
    assertEquals(totalTimeSec, row.getValue("totalTime"));
}

@Test
void testAverageAggregation() {
```

```
        AggregateSizeTime directive = new AggregateSizeTime();

        Map<String, String> args = new HashMap<>();
        args.put("sizeColumn", "size");
        args.put("timeColumn", "duration");
        args.put("resultSizeColumn", "avgSize");
        args.put("resultTimeColumn", "avgTime");
        args.put("sizeUnit", "KB");
        args.put("timeUnit", "ms");
        args.put("aggType", "average");

        directive.initialize(mock(DirectiveContext.class), mockArgs(args));
        List<Row> result = directive.execute(sampleRows(), mock(ExecutorContext.class));

        Row row = result.get(0);

        long totalBytes = 1 * 1024 * 1024 + 512 * 1024 + 1024;
        long totalNanos = 1_000_000_000 + 500_000_000 + 250_000_000;

        double avgSizeKB = (totalBytes / 3.0) / 1024.0;
        double avgTimeMs = (totalNanos / 3.0) / 1_000_000.0;

        assertEquals(1, result.size());
        assertEquals(avgSizeKB, row.getValue("avgSize"));
        assertEquals(avgTimeMs, row.getValue("avgTime"));
    }
}
```

Result :-

For testTotalAggregation():

- Sizes:

    - 1MB = 1048576 bytes
    - 512KB = 524288 bytes
    - 1024B = 1024 bytes
      → Total = **1573888 bytes** = **1.501 MB**
- Durations:
    - 1s = 1,000,000,000 ns
    - 500ms = 500,000,000 ns
    - 250,000,000 ns
      → Total = **1.75 s**

For testAverageAggregation():

- Avg size = **512.32 KB**
- Avg time = **583.33 ms**

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

import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class TimeDurationTest {

  @Test
  void testMillisecondsParsing() {
    TimeDuration duration = new TimeDuration("500ms");
    assertEquals(500, duration.getMilliseconds());
    assertEquals(500, duration.value());
    assertEquals(TokenType.TIME_DURATION, duration.type());

    JsonElement json = duration.toJson();
    assertEquals(500, json.getAsJsonObject().get("milliseconds").getAsLong());
```

```
}

@Test
void testSecondsParsing() {
    TimeDuration duration = new TimeDuration("2s");
    assertEquals(2000, duration.getMilliseconds());
}

@Test
void testMinutesParsing() {
    TimeDuration duration = new TimeDuration("3m");
    assertEquals(180000, duration.getMilliseconds());
}

@Test
void testHoursParsing() {
    TimeDuration duration = new TimeDuration("1h");
    assertEquals(3600000, duration.getMilliseconds());
}

@Test
void testWhitespaceAndCaseHandling() {
    TimeDuration duration = new TimeDuration("  4H ");
    assertEquals(14400000, duration.getMilliseconds());
}

@Test
void testInvalidFormatThrowsException() {
    assertThrows(IllegalArgumentException.class, () -> new TimeDuration("10days"));
    assertThrows(IllegalArgumentException.class, () -> new TimeDuration("xyz"));
    assertThrows(IllegalArgumentException.class, () -> new TimeDuration(""));
```

```
    assertThrows(IllegalArgumentException.class, () -> new TimeDuration(null));
  }
}
```

Input	Output (ms)
500ms	500
2s	2000
3m	180000
1h	3600000
4H	14400000

JUnit Test Class for ByteSize

java

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

import com.google.gson.JsonElement;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class ByteSizeTest {

  @Test
  void testBytesParsing() {
    ByteSize size = new ByteSize("512B");
    assertEquals(512, size.getBytes());
    assertEquals(512, size.value());
    assertEquals(TokenType.BYTE_SIZE, size.type());

    JsonElement json = size.toJson();
```

```
    assertEquals(512, json.getAsJsonObject().get("bytes").getAsLong());
}

@Test
void testKBParsing() {
    ByteSize size = new ByteSize("1KB");
    assertEquals(1024, size.getBytes());
}

@Test
void testMBParsing() {
    ByteSize size = new ByteSize("2MB");
    assertEquals(2 * 1024 * 1024, size.getBytes());
}

@Test
void testGBParsing() {
    ByteSize size = new ByteSize("3GB");
    assertEquals(3L * 1024 * 1024 * 1024, size.getBytes());
}

@Test
void testWhitespaceAndCaseInsensitive() {
    ByteSize size = new ByteSize(" 4gb ");
    assertEquals(4L * 1024 * 1024 * 1024, size.getBytes());
}

@Test
void testInvalidFormatThrowsException() {
    assertThrows(IllegalArgumentException.class, () -> new ByteSize("10TB"));
    assertThrows(IllegalArgumentException.class, () -> new ByteSize("xyz"));
```

```
    assertThrows(IllegalArgumentException.class, () -> new ByteSize(""));
    assertThrows(IllegalArgumentException.class, () -> new ByteSize(null));
  }
}
```

Input	Output (bytes)
512B	512
1KB	1024
2MB	2,097,152
3GB	3,221,225,472
4gb	4,294,967,296

Unit Test Class: TokenParserTest

Using JUnit 5:

java

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

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class TokenParserTest {

  @Test
  void testDescribeToken() {
    assertEquals("Represents a directive name.",
TokenParser.describeToken(TokenType.DIRECTIVE_NAME));
    assertEquals("Represents a column name.",
TokenParser.describeToken(TokenType.COLUMN_NAME));
    assertEquals("Represents a quoted text.", TokenParser.describeToken(TokenType.TEXT));
```

Input	Output (bytes)

```
   assertEquals("Represents a numeric value.",
TokenParser.describeToken(TokenType.NUMERIC));

   assertEquals("Represents a boolean value.",
TokenParser.describeToken(TokenType.BOOLEAN));

   assertEquals("Represents a list of column names.",
TokenParser.describeToken(TokenType.COLUMN_NAME_LIST));

   assertEquals("Represents a list of quoted texts.",
TokenParser.describeToken(TokenType.TEXT_LIST));

   assertEquals("Represents a list of numeric values.",
TokenParser.describeToken(TokenType.NUMERIC_LIST));

   assertEquals("Represents a list of boolean values.",
TokenParser.describeToken(TokenType.BOOLEAN_LIST));

   assertEquals("Represents a logical or mathematical expression.",
TokenParser.describeToken(TokenType.EXPRESSION));

   assertEquals("Represents key=value properties.",
TokenParser.describeToken(TokenType.PROPERTIES));

   assertEquals("Represents value ranges.", TokenParser.describeToken(TokenType.RANGES));

   assertEquals("Represents a restricted identifier string.",
TokenParser.describeToken(TokenType.IDENTIFIER));

   assertEquals("Represents a byte size (e.g., 10KB).",
TokenParser.describeToken(TokenType.BYTE_SIZE));

   assertEquals("Represents a time duration (e.g., 1s).",
TokenParser.describeToken(TokenType.TIME_DURATION));

  }


  @Test

  void testInvalidToken() {

   Exception exception = assertThrows(IllegalArgumentException.class, () -> {

    TokenParser.describeToken(null);

   });

   assertTrue(exception.getMessage().contains("Unknown TokenType"));

  }
```

Input	Output (bytes)

}

---

src/test/java/io/cdap/wrangler/api/parser/TokenParserTest.java

mvn test