1.1 ./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier/HasuraSQLSimplifierTransformer.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.RegularExpressions.Transformer.HasuraSQLSimplifier
{
    /// <summary>
    /// <para>
    /// Represents the hasura sql simplifier transformer.
    /// </para>
    /// <para></para>
    /// </summary>
    /// <seealso cref="TextTransformer"/>
    public class HasuraSQLSimplifierTransformer : TextTransformer
    {
        /// <summary>
        /// <para>
        /// The to list.
        /// </para>
        /// <para></para>
        /// </summary>
        public static readonly IList<ISubstitutionRule> DefaultRules = new List<SubstitutionRule>
        {
            // HTML clean up
            (new Regex(@"<span class=""[^""]*"">([^<>]*)<\/span>"), "$1", 0),
            // ('describe')
            // 'describe'
            (new Regex(@"\([\s\n]*('[^']+')[\s\n]*\)"), "$1", int.MaxValue),
            // AND ('true' AND 'true')
            //
            (new Regex(@"[\s\n]*AND[\s\n]*\([\s\n]*'true'[\s\n]*AND[\s\n]*'true'[\s\n]*\)"), "",
            ↪  0),
            // AND ('true')
            //
            (new Regex(@"[\s\n]*AND[\s\n]*'true'"), "", 0),
            //  ::
            // ::
            (new Regex(@"[\s]*::[\s]*"), "::", 0),
            // ('describe'::text)
            // 'describe'::text
            (new Regex(@"\([\s\n]*('[^']+'::text)[\s\n]*\)"), "$1", 0),
            // ("_0__be_0_nodes"."target_id")
            // "_0__be_0_nodes"."target_id"
            (new Regex(@"\([\s\n]*(""[^""]+"")[\s\n]*\.[\s\n]*(""[^""]+"")[\s\n]*\)"), "$1.$2",
            ↪  0),
            // ("public"."nodes"."_id")
            // "public"."nodes"."_id"
            (new Regex(@"\([\s\n]*(""[^""]+"")[\s\n]*\.[\s\n]*(""[^""]+"")[\s\n]*\.[\s\n]*(""[^"」
            ↪  "]+"")[\s\n]*\)"), "$1.$2.$3",
            ↪  0),
            // LIMIT\n\t\t\t1
            // LIMIT 1
            (new Regex(@"(LIMIT)[\s\n]*(\d+)"), "$1 $2", 0),
            // ("_0__be_0_nodes"."type" = 'describe'::text)
            // "_0__be_0_nodes"."type" = 'describe'::text
            (new Regex(@"(\W)\([\s\n]*((?!SELECT)[^\s\n()][^()]*?)[\s\n]*\)"), "$1$2",
            ↪  int.MaxValue),
            // (EXISTS (...))
            // EXISTS (...)
            (new Regex(@"(\W)\([\s\n]*((?!SELECT)[^\s\n()][^()]*\([^()]*\)[^()]*?)[\s\n]*\)"),
            ↪  "$1$2", int.MaxValue),
            // ((EXISTS (...)))
            // (EXISTS (...))
            (new Regex(@"(\W)\([\s\n]*((?!SELECT)[^\s\n()][^()]*\([^()]*\([^()]*\)[^()]*\)[^()]*」
            ↪  ?)[\s\n]*\)"), "$1$2",
            ↪  int.MaxValue),
        }.Cast<ISubstitutionRule>().ToList();

        /// <summary>
        /// <para>
        /// Initializes a new <see cref="HasuraSQLSimplifierTransformer"/> instance.
        /// </para>
        /// <para></para>
        /// </summary>
```

```
69          public HasuraSQLSimplifierTransformer()
70              : base(DefaultRules)
71          {
72          }
73      }
74  }
```

1.2 ./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests/HasuraSQLSimplifierTransforme

```
1   using Xunit;
2
3   namespace Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests
4   {
5       /// <summary>
6       /// <para>
7       /// Represents the hasura sql simplifier transformer tests.
8       /// </para>
9       /// <para></para>
10      /// </summary>
11      public class HasuraSQLSimplifierTransformerTests
12      {
13          /// <summary>
14          /// <para>
15          /// Tests that empty line test.
16          /// </para>
17          /// <para></para>
18          /// </summary>
19          [Fact]
20          public void EmptyLineTest()
21          {
22              // This test can help to test basic problems with regular expressions like incorrect
                ↪  syntax
23              var transformer = new HasuraSQLSimplifierTransformer();
24              var actualResult = transformer.Transform("");
25              Assert.Equal("", actualResult);
26          }
27
28          /// <summary>
29          /// <para>
30          /// Tests that basic request test.
31          /// </para>
32          /// <para></para>
33          /// </summary>
34          [Fact]
35          public void BasicRequestTest()
36          {
37              var original = @"SELECT
38    coalesce(json_agg(""root""), '[]') AS ""root""
39  FROM
40    (
41      SELECT
42        row_to_json(
43          (
44            SELECT
45              ""_2_e""
46            FROM
47              (
48                SELECT
49                  ""_1_root.base"".""id"" AS ""id""
50              ) AS ""_2_e""
51          )
52        ) AS ""root""
53      FROM
54        (
55          SELECT
56            *
57          FROM
58            ""public"".""nodes""
59          WHERE
60            (
61              (
62                (""public"".""nodes"".""type"") = (('auth_token') :: text)
63              )
64              AND (
65                EXISTS (
66                  SELECT
67                    1
68                  FROM
69                    ""public"".""nodes"" AS ""_0__be_0_nodes""
70                  WHERE
71                    (
```

```
                    (
                      (
                        (""_0__be_0_nodes"".""_source_id"") = (""public"".""nodes"".""_id"")
                      )
                      AND ('true')
                    )
                    AND (
                      (
                        (
                          (""_0__be_0_nodes"".""type"") = (('describe') :: text))
                          AND ('true')
                        )
                        AND (
                          (
                            (
                              (""_0__be_0_nodes"".""target_id"") = (('X-Hasura-User-Id') :: text)
                            )
                            AND ('true')
                          )
                          AND ('true')
                        )
                      )
                      AND (
                        ('true')
                        AND ('true')
                      )
                    )
                  )
                )
              )
            )
        ) AS ""_1_root.base""
      LIMIT
        1
    ) AS ""_3_root""";

            var expected = @"SELECT
    coalesce(json_agg(""root""), '[]') AS ""root""
  FROM
    (
      SELECT
        row_to_json(
          (
            SELECT
              ""_2_e""
            FROM
              (
                SELECT
                  ""_1_root.base"".""id"" AS ""id""
              ) AS ""_2_e""
          )
        ) AS ""root""
      FROM
        (
          SELECT
            *
          FROM
            ""public"".""nodes""
          WHERE
            ""public"".""nodes"".""type"" = 'auth_token'::text
              AND EXISTS (
                  SELECT
                    1
                  FROM
                    ""public"".""nodes"" AS ""_0__be_0_nodes""
                  WHERE
                    ""_0__be_0_nodes"".""_source_id"" = ""public"".""nodes"".""_id""
                      AND ""_0__be_0_nodes"".""type"" = 'describe'::text
                          AND ""_0__be_0_nodes"".""target_id"" = 'X-Hasura-User-Id'::text
                )
        ) AS ""_1_root.base""
      LIMIT 1
    ) AS ""_3_root""";
            var transformer = new HasuraSQLSimplifierTransformer();
            var actual = transformer.Transform(original);
            Assert.Equal(expected, actual);
        }
    }
}
```

Index