```csharp
1   using System.Collections.Generic;
2   using System.Linq;
3   using System.Text.RegularExpressions;
4
5   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7   namespace Platform.RegularExpressions.Transformer.HasuraSQLSimplifier
8   {
9       /// <summary>
10      /// <para>
11      /// Represents the hasura sql simplifier transformer.
12      /// </para>
13      /// <para></para>
14      /// </summary>
15      /// <seealso cref="TextTransformer"/>
16      public class HasuraSQLSimplifierTransformer : TextTransformer
17      {
18          /// <summary>
19          /// <para>
20          /// The to list.
21          /// </para>
22          /// <para></para>
23          /// </summary>
24          public static readonly IList<ISubstitutionRule> DefaultRules = new List<SubstitutionRule>
25          {
26              // HTML clean up
27              (new Regex(@"<span class=""[^""]*"">([^<>]*)<\/span>"), "$1", 0),
28              // ('describe')
29              // 'describe'
30              (new Regex(@"\([\s\n]*('[^']+')[\s\n]*\)"), "$1", int.MaxValue),
31              // AND ('true' AND 'true')
32              //
33              (new Regex(@"[\s\n]*AND[\s\n]*\([\s\n]*'true'[\s\n]*AND[\s\n]*'true'[\s\n]*\)"), "",
                 0),
34              // AND ('true')
35              //
36              (new Regex(@"[\s\n]*AND[\s\n]*'true'"), "", 0),
37              //  ::
38              // ::
39              (new Regex(@"[\s]*::[\s]*"), "::", 0),
40              // ('describe'::text)
41              // 'describe'::text
42              (new Regex(@"\([\s\n]*('[^']+'::text)[\s\n]*\)"), "$1", 0),
43              // ("_0__be_0_nodes"."target_id")
44              // "_0__be_0_nodes"."target_id"
45              (new Regex(@"\([\s\n]*(""[^""]+"")[\s\n]*\.[\s\n]*(""[^""]+"")[\s\n]*\)"), "$1.$2",
                 0),
46              // ("public"."nodes"."_id")
47              // "public"."nodes"."_id"
48              (new Regex(@"\([\s\n]*(""[^""]+"")[\s\n]*\.[\s\n]*(""[^""]+"")[\s\n]*\.[\s\n]*(""[^"
                 "]+"")[\s\n]*\)"), "$1.$2.$3",
                 0),
49              // LIMIT\n\t\t\t1
50              // LIMIT 1
51              (new Regex(@"(LIMIT)[\s\n]*(\d+)"), "$1 $2", 0),
52              // ("_0__be_0_nodes"."type" = 'describe'::text)
53              // "_0__be_0_nodes"."type" = 'describe'::text
54              (new Regex(@"(\W)\([\s\n]*((?!SELECT)[^\s\n()][^()]*?)[\s\n]*\)"), "$1$2",
                 int.MaxValue),
55              // (EXISTS (...))
56              // EXISTS (...)
57              (new Regex(@"(\W)\([\s\n]*((?!SELECT)[^\s\n()][^()]*\([^()]*\)[^()]*?)[\s\n]*\)"),
                 "$1$2", int.MaxValue),
58              // ((EXISTS (...)))
59              // (EXISTS (...))
60              (new Regex(@"(\W)\([\s\n]*((?!SELECT)[^\s\n()][^()]*\([^()]*\([^()]*\)[^()]*\)[^()]*
                 ?)[\s\n]*\)"), "$1$2",
                 int.MaxValue),
61          }.Cast<ISubstitutionRule>().ToList();
62
63          /// <summary>
64          /// <para>
65          /// Initializes a new <see cref="HasuraSQLSimplifierTransformer"/> instance.
66          /// </para>
67          /// <para></para>
68          /// </summary>
```

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

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

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

```
72                                    )
73                                    AND ('true')
74                                )
75                            )
76                            AND (
77                                ('true')
78                                AND ('true')
79                            )
80                        )
81                    )
82                )
83            )
84        )
85    ) AS ""_1_root.base""
86    LIMIT
87        1
88 ) AS ""_3_root""";

90            var expected = @"SELECT
91  coalesce(json_agg(""root""), '[]') AS ""root""
92 FROM
93  (
94    SELECT
95      row_to_json(
96        (
97          SELECT
98            ""_2_e""
99          FROM
100            (
101              SELECT
102                ""_1_root.base"".""id"" AS ""id""
103            ) AS ""_2_e""
104        )
105    ) AS ""root""
106    FROM
107      (
108        SELECT
109          *
110        FROM
111          ""public"".""nodes""
112        WHERE
113          ""public"".""nodes"".""type"" = 'auth_token'::text
114            AND EXISTS (
115                SELECT
116                  1
117                FROM
118                  ""public"".""nodes"" AS ""_0__be_0_nodes""
119                WHERE
120                  ""_0__be_0_nodes"".""_source_id"" = ""public"".""nodes"".""_id""
121                    AND ""_0__be_0_nodes"".""type"" = 'describe'::text
122                    AND ""_0__be_0_nodes"".""target_id"" = 'X-Hasura-User-Id'::text
123            )
124      ) AS ""_1_root.base""
125    LIMIT 1
126 ) AS ""_3_root""";
127            var transformer = new HasuraSQLSimplifierTransformer();
128            var actual = transformer.Transform(original);
129            Assert.Equal(expected, actual);
130        }
131    }
132 }
```

Index