

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

```

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]*\\)"), "",
34             → 0),
35             // AND ('true')
36             //
37             (new Regex(@"\[\\s\\n]*AND\[\\s\\n]*'true'"), "", 0),
38             // ::
39             // ::
40             (new Regex(@"\[\\s]*::\[\\s]*"), "::", 0),
41             // ('describe'::text)
42             // 'describe'::text
43             (new Regex(@"\[\\s\\n]*('^[^']*')::text)\[\\s\\n]*\\)"), "$1", 0),
44             // ("_0__be_0_nodes"."target_id")
45             // "_0__be_0_nodes"."target_id"
46             (new Regex(@"\[\\s\\n]*(""[^"]*"")\[\\s\\n]*\\. \[\\s\\n]*(""[^"]*"")\[\\s\\n]*\\)"), "$1.$2",
47             → 0),
48             // ("public"."nodes"."_id")
49             // "public"."nodes"."_id"
50             (new Regex(@"\[\\s\\n]*(""[^"]*"")\[\\s\\n]*\\. \[\\s\\n]*(""[^"]*"")\[\\s\\n]*\\. \[\\s\\n]*(""[^"]*"")\[\\s\\n]*\\)"), "$1.$2.$3",
51             → 0),
52             // LIMIT\\n\\t\\t\\t1
53             // LIMIT 1
54             (new Regex(@"(LIMIT)\[\\s\\n]*([d+])"), "$1 $2", 0),
55             // ("_0__be_0_nodes"."type" = 'describe'::text)
56             // "_0__be_0_nodes"."type" = 'describe'::text
57             (new Regex(@"(\\W)\\([\\s\\n]*((?!SELECT) [^\\s\\n() [^()]*?) [\\s\\n]*\\)"), "$1$2",
58             → int.MaxValue),
59             // (EXISTS (...))
60             // EXISTS (...)
61             (new Regex(@"(\\W)\\([\\s\\n]*((?!SELECT) [^\\s\\n() [^()]*?) [^()]*?) [\\s\\n]*\\)"),
62             → "$1$2", int.MaxValue),
63             // ((EXISTS (...)))
64             // (EXISTS (...))
65             (new Regex(@"(\\W)\\([\\s\\n]*((?!SELECT) [^\\s\\n() [^()]*?) [^()]*?) [^()]*?) [^()]*? [^()]*? [^()]*?"),
66             → "? [\\s\\n]*\\)"), "$1$2",
67             → int.MaxValue),
68         }.Cast<ISubstitutionRule>().ToList();
69
70     /// <summary>
71     /// <para>
72     /// Initializes a new <see cref="HasuraSQLSimplifierTransformer"/> instance.
73     /// </para>
74     /// <para></para>
75     /// </summary>

```

```

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

```

1.2 ./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests/HasuraSQLSimplifierTransformerTests.cs

```
using Xunit;

namespace Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests
{
    public class HasuraSQLSimplifierTransformerTests
    {
        [Fact]
        public void EmptyLineTest()
        {
            // This test can help to test basic problems with regular expressions like incorrect
            ↪ syntax
            var transformer = new HasuraSQLSimplifierTransformer();
            var actualResult = transformer.Transform("");
            Assert.Equal("", actualResult);
        }

        [Fact]
        public void BasicRequestTest()
        {
            var original = @"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 ('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'))
```

```

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"";
89
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

./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier.Tests/HasuraSQLSimplifierTransformerTests.cs, 2
./csharp/Platform.RegularExpressions.Transformer.HasuraSQLSimplifier/HasuraSQLSimplifierTransformer.cs, 1