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
{
    public class HasuraSQLSimplifierTransformer : TextTransformer
    {
        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)\([\s\n]*(((?!SELECT)[^()])*?)[\s\n]*\)"), "$1$2", int.MaxValue),
        }.Cast<ISubstitutionRule>().ToList();

        public HasuraSQLSimplifierTransformer()
            : base(DefaultRules)
        {
        }
    }
}
```

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

```
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')
                          )
                          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