DataSet join

```
case class Person(familyId: Long, name: String, cityId: Long)
case class City(cityId: Long, name: String)

val family = Seq(
  Person(0, "Agata", 0),
  Person(1, "Iweta", 0),
  Person(2, "Patryk", 2),
  Person(3, "Maksym", 0))

val cities = Seq(
  City(0, "Warsaw"),
  City(1, "Washington"),
  City(2, "Sopot"))
```

```
defined class Person
defined class City
family: Seq[Person] = List(Person(0,Agata,0), Person(1,Iweta,0), Person(2,Patryk,2), Person(3,Maksym,0))
cities: Seq[City] = List(City(0,Warsaw), City(1,Washington), City(2,Sopot))
```

```
val familyDS = family.toDS
val cityDS = cities.toDS
```

```
familyDS: org.apache.spark.sql.Dataset[Person] = [familyId: bigint, name: string ... 1 more field]
cityDS: org.apache.spark.sql.Dataset[City] = [cityId: bigint, name: string]
```

```
val cityFamily = cityDS.join(familyDS, cityDS("cityId") === familyDS("cityId"), "left")
```

```
cityFamily: org.apache.spark.sql.DataFrame = [cityId: bigint, name: string ... 3 more fields]
```

```
cityFamily.show(20)
```

```
+------+----------+--------+------+------+
|cityId|      name|familyId|  name|cityId|
+------+----------+--------+------+------+
|     0|    Warsaw|       3|Maksym|     0|
|     0|    Warsaw|       1| Iweta|     0|
|     0|    Warsaw|       0| Agata|     0|
|     1|Washington|    null|  null|  null|
|     2|     Sopot|       2|Patryk|     2|
+------+----------+--------+------+------+
```

```
cityDS.join(familyDS, cityDS("cityId") === familyDS("cityId"), "left").select(cityDS("cityId"), cityDS("name"), $"familyId", familyDS("name").as("fname"), familyDS("cityId").as("fcityId")).show(20)
```

```
+------+----------+--------+------+-------+
|cityId|      name|familyId| fname|fcityId|
+------+----------+--------+------+-------+
|     0|    Warsaw|       3|Maksym|      0|
|     0|    Warsaw|       1| Iweta|      0|
|     0|    Warsaw|       0| Agata|      0|
|     1|Washington|    null|  null|   null|
|     2|     Sopot|       2|Patryk|      2|
+------+----------+--------+------+-------+
```

```
val joined = cityDS.join(familyDS, cityDS("cityId") === familyDS("cityId"), "left").select(cityDS("cityId"), cityDS("name"), $"familyId", familyDS("name").as("fname"), familyDS("cityId").as("fcityId"))
```

```
joined: org.apache.spark.sql.DataFrame = [cityId: bigint, name: string ... 3 more fields]
```

```
val cityDF = cities.toDF
val familyDF = family.toDF
```

```
cityDF: org.apache.spark.sql.DataFrame = [cityId: bigint, name: string]
familyDF: org.apache.spark.sql.DataFrame = [familyId: bigint, name: string ... 1 more field]
```

```
val joinedDF = cityDF.join(familyDF, Seq("cityId"), "left")
```

```
joinedDF: org.apache.spark.sql.DataFrame = [cityId: bigint, name: string ... 2 more fields]
```

```
joinedDF.show(20)
```

```
+------+----------+--------+------+
|cityId|      name|familyId|  name|
+------+----------+--------+------+
|     0|    Warsaw|       3|Maksym|
|     0|    Warsaw|       1| Iweta|
|     0|    Warsaw|       0| Agata|
|     1|Washington|    null|  null|
|     2|     Sopot|       2|Patryk|
+------+----------+--------+------+
```

```
joinedDF.select($"cityDF.name")
```

```
org.apache.spark.sql.AnalysisException: cannot resolve '`cityDF.name`' given input columns: [cityId, name, familyId, name];;
'Project ['cityDF.name]
+- Project [cityId#10732L, name#10733, familyId#10740L, name#10741]
   +- Join LeftOuter, (cityId#10732L = cityId#10742L)
      :- LocalRelation [cityId#10732L, name#10733]
      +- LocalRelation [familyId#10740L, name#10741, cityId#10742L]

  at org.apache.spark.sql.catalyst.analysis.package$AnalysisErrorAt.failAnalysis(package.scala:42)
  at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAnalysis$1$$anonfun$apply$3.applyOrElse(CheckAnalysis.scala:110)
  at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAnalysis$1$$anonfun$apply$3.applyOrElse(CheckAnalysis.scala:107)
  at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$transformUp$1.apply(TreeNode.scala:278)
  at org.apache.spark.sql.catalyst.trees.TreeNode$$anonfun$transformUp$1.apply(TreeNode.scala:278)
  at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:70)
  at org.apache.spark.sql.catalyst.trees.TreeNode.transformUp(TreeNode.scala:277)
  at org.apache.spark.sql.catalyst.plans.QueryPlan$$anonfun$transformExpressionsUp$1.apply(QueryPlan.scala:93)
  at org.apache.spark.sql.catalyst.plans.QueryPlan$$anonfun$transformExpressionsUp$1.apply(QueryPlan.scala:93)
  at org.apache.spark.sql.catalyst.plans.QueryPlan$$anonfun$1.apply(QueryPlan.scala:105)
  at org.apache.spark.sql.catalyst.plans.QueryPlan$$anonfun$1.apply(QueryPlan.scala:105)
  at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:70)
  at org.apache.spark.sql.catalyst.plans.QueryPlan.transformExpression$1(QueryPlan.scala:104)
  at org.apache.spark.sql.catalyst.plans.QueryPlan.org$apache$spark$sql$catalyst$plans$QueryPlan$$recursiveTransform$1(QueryPlan.scala:116)
  at org.apache.spark.sql.catalyst.plans.QueryPlan$$anonfun$org$apache$spark$sql$catalyst$plans$QueryPlan$$recursiveTransform$1$2.apply(QueryPlan.scala:121)
  at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
  at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
  at scala.collection.mutable.ResizableArray$class.foreach(ResizableArray.scala:59)
  at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:48)
  at scala.collection.TraversableLike$class.map(TraversableLike.scala:234)
  at scala.collection.AbstractTraversable.map(Traversable.scala:104)
  at org.apache.spark.sql.catalyst.plans.QueryPlan.org$apache$spark$sql$catalyst$plans$QueryPlan$$recursiveTransform$1(QueryPlan.scala:121)
  at org.apache.spark.sql.catalyst.plans.QueryPlan$$anonfun$2.apply(QueryPlan.scala:126)
  at org.apache.spark.sql.catalyst.trees.TreeNode.mapProductIterator(TreeNode.scala:187)
  at org.apache.spark.sql.catalyst.plans.QueryPlan.mapExpressions(QueryPlan.scala:126)
  at org.apache.spark.sql.catalyst.plans.QueryPlan.transformExpressionsUp(QueryPlan.scala:93)
  at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAnalysis$1.apply(CheckAnalysis.scala:107)
  at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$$anonfun$checkAnalysis$1.apply(CheckAnalysis.scala:85)
  at org.apache.spark.sql.catalyst.trees.TreeNode.foreachUp(TreeNode.scala:127)
  at org.apache.spark.sql.catalyst.analysis.CheckAnalysis$class.checkAnalysis(CheckAnalysis.scala:85)
  at org.apache.spark.sql.catalyst.analysis.Analyzer.checkAnalysis(Analyzer.scala:95)
  at org.apache.spark.sql.catalyst.analysis.Analyzer$$anonfun$executeAndCheck$1.apply(Analyzer.scala:108)
  at org.apache.spark.sql.catalyst.analysis.Analyzer$$anonfun$executeAndCheck$1.apply(Analyzer.scala:105)
  at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper$.markInAnalyzer(AnalysisHelper.scala:201)
  at org.apache.spark.sql.catalyst.analysis.Analyzer.executeAndCheck(Analyzer.scala:105)
  at org.apache.spark.sql.execution.QueryExecution.analyzed$lzycompute(QueryExecution.scala:57)
  at org.apache.spark.sql.execution.QueryExecution.analyzed(QueryExecution.scala:55)
  at org.apache.spark.sql.execution.QueryExecution.assertAnalyzed(QueryExecution.scala:47)
  at org.apache.spark.sql.Dataset$.ofRows(Dataset.scala:79)
  at org.apache.spark.sql.Dataset.org$apache$spark$sql$Dataset$$withPlan(Dataset.scala:3407)
  at org.apache.spark.sql.Dataset.select(Dataset.scala:1335)
  ... 125 elided
```

```
joined.groupBy($"cityId").count.show(20)
```

```
+------+-----+
|cityId|count|
+------+-----+
|     0|    3|
|     1|    1|
|     2|    1|
+------+-----+
```

```
val cityFamilyJoinWith = cityDS.joinWith(familyDS, cityDS("id") === familyDS("cityID"), "left")
```

```
cityFamilyJoinWith: org.apache.spark.sql.Dataset[(City, Person)] = [_1: struct<id: bigint, name: string>, _2: struct<id: bigint, name: string ... 1 more field>]
```

```
cityDS.join(familyDS, Seq("cityId"), "left").show(20)
```

```
+------+----------+--------+------+
|cityId|      name|familyId|  name|
+------+----------+--------+------+
|     0|    Warsaw|       3|Maksym|
|     0|    Warsaw|       1| Iweta|
|     0|    Warsaw|       0| Agata|
|     1|Washington|    null|  null|
|     2|     Sopot|       2|Patryk|
+------+----------+--------+------+
```

```
cityFamilyJoinWith.show(20)
```

```
+---------------+--------------+
|             _1|            _2|
+---------------+--------------+
|    [0, Warsaw]|[3, Maksym, 0]|
|    [0, Warsaw]| [1, Iweta, 0]|
|    [0, Warsaw]| [0, Agata, 0]|
|[1, Washington]|          null|
|     [2, Sopot]|[2, Patryk, 2]|
+---------------+--------------+
```