Item 349
git_comments:

1. Fixed-size, non-intersecting time-based windows, for example: every hour aggregate elements from the previous hour; SQL Syntax: TUMBLE(monotonic_field, window_size [, window_offset]) Example: TUMBLE(event_timestamp_field, INTERVAL '1' HOUR)
2. Sliding, fixed-size, intersecting time-based windows, for example: every minute aggregate elements from the previous hour; SQL Syntax: HOP(monotonic_field, emit_frequency, window_size [, window_offset]) Example: HOP(event_timestamp_field, INTERVAL '1' MINUTE, INTERVAL '1' HOUR)
3. Session windows, for example: aggregate events after a gap of 1 minute of no events; SQL Syntax: SESSION(monotonic_field, session_gap) Example: SESSION(event_timestamp_field, INTERVAL '1' MINUTE)
4. * * Creates {@link WindowFn} wrapper based on HOP/TUMBLE/SESSION call in a query.
5. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
6. * * Returns a {@link WindowFn} based on the SQL windowing function defined by {#code operatorKind}. * Supported {@link SqlKind}s: * * {@link SqlKind#TUMBLE}, mapped to {@link FixedWindows}; * {@link SqlKind#HOP}, mapped to {@link SlidingWindows}; * {@link SqlKind#SESSION}, mapped to {@link Sessions}; * * * <p>For example: * <pre>{@code * SELECT event_timestamp, COUNT(*) * FROM PCOLLECTION * GROUP BY TUMBLE(event_timestamp, INTERVAL '1' HOUR) * }</pre> * * <p>SQL window functions support optional window_offset parameter which indicates a * how window definition is offset from the event time. Offset is zero if not specified. * * <p>Beam model does not support offset for session windows, so this method will throw * {@link UnsupportedOperationException} if offset is specified * in SQL query for {@link SqlKind#SESSION}.
7. * * Returns optional of {@link AggregateWindowField} which represents a * windowing function specified by HOP/TUMBLE/SESSION in the SQL query. * * <p>If no known windowing function is specified in the query, then {@link Optional#empty()} * is returned. * * <p>Throws {@link UnsupportedOperationException} if it cannot convert SQL windowing function * call to Beam model, see {@link #getWindowFieldAt(RexCall, int)} for details.
8. * * For internal use only; no backwards compatibility guarantees. * * <p>Represents a field with a window function call in a SQL expression.
9. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
10. * * Performs the same check as {@link GroupByKey}, provides more context in exception. * * <p>Verifies that the input PCollection is bounded, or that there is windowing/triggering being * used. Without this, the watermark (at end of global window) will never be reached. * * <p>Throws {@link UnsupportedOperationException} if validation fails.
11. * * Event time field, defines watermark.
12. * * Builds an unbounded {@link PCollection} in {@link Pipeline} * set by {@link #inPipeline(Pipeline)}. * * <p>If timestamp field was set with {@link #withTimestampField(String)} then * watermark will be advanced to the values from that field.

git_commits:

1. **summary:** Merge pull request #4546: [SQL] Inherit windowing strategy from the input in Aggregate operation
   message: Merge pull request #4546: [SQL] Inherit windowing strategy from the input in Aggregate operation

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SQL] Inherit windowing strategy from the input in Aggregate operation
   body: Aggregate operation (GROUP BY + aggregate functions) was overriding input's windowing strategy even when it was not explicitly specified in SQL. This change makes it inherit input's configuration (window and trigger) unless windowing is explicitly specified using 'HOP/TUMBLE/SESSION' windowing functions. Details in the doc: https://docs.google.com/document/d/1RmyV9e1Qab-axsLI1WWpw5oGAJDv0X7y9OSnPnrZWJk/edit#heading=h.q943qvxj4tt6 Follow this checklist to help us incorporate your contribution quickly and easily: - [ ] Make sure there is a [JIRA issue] (https://issues.apache.org/jira/projects/BEAM/issues/) filed for the change (usually before you start working on it). Trivial changes like typos do not require a JIRA issue. Your pull request should address just this issue, without pulling in other changes. - [x] Each commit in the pull request should have a meaningful subject line and body. - [ ] Format the pull request title like `[BEAM-XXX] Fixes bug in ApproximateQuantiles`, where you replace `BEAM-XXX` with the appropriate JIRA issue. - [x] Write a pull request description that is detailed enough to understand what the pull request does, how, and why. - [x] Run `mvn clean verify` to make sure basic checks pass. A more thorough check will be performed on your pull request automatically. - [ ] If this contribution is large, please file an Apache [Individual Contributor License Agreement](https://www.apache.org/licenses/icla.pdf). ---

github_pulls_comments:

1. R: @kennknowles , can you take a look?
2. retest this please
3. retest this please
4. retest this please
5. Reviewed 6 of 6 files at r1. Review status: all files reviewed at latest revision, 13 unresolved discussions, some commit checks failed. --- * [sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 70 at r1] (https://reviewable.io:443/reviews/apache/beam/4546#-L4EalKCrH5RZVocmItf:-L4EalKCrH5RZVocmItg:b-551d4r) ([raw file] (https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp > ```Java > List<ImmutableBitSet> groupSets, > List<AggregateCall> aggCalls, > @Nonnull AggregateWindowField windowField) { > ``` We should assume every object is non-null, and put the findbugs `DefaultAnnotation` on the package. (separately from this PR) --- * [sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 132 at r1] (https://reviewable.io:443/reviews/apache/beam/4546#-L4EdXpXI9r0vQU5sTA6:-L4EdXpXI9r0vQU5sTA7:b-iz8kne) ([raw file] (https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp > ```Java > // If GlobalWindows is used with unbounded input, then DefaultTrigger > // does not produce meaningful results. Reject this case. > if (isUnbounded(upstream) > ``` We should go farther than this, perhaps. You don't have to do it in this PR, but maybe file a ticket. I believe we can also analyze some other triggers that are not meaningful. Or you could argue that if the user sets it themselves then we should respect their request to never have output... --- * [sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 158 at r1] (https://reviewable.io:443/reviews/apache/beam/4546#-L4Ecx-pQb4v_ecbhmEl:-L4Ecx-pQb4v_ecbhmEm:b-g7a565) ([raw file] (https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp > ```Java > } > > private boolean isUnbounded(PCollection<BeamRecord> upstream) { > ``` This one seems a bit silly --- * [sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 162 at r1] (https://reviewable.io:443/reviews/apache/beam/4546#-L4Ed0MTZTJhbbog3oII:-L4Ed0MTZTJhbbog3oIJ:b-c50yjd) ([raw file] (https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp > ```Java > } > > private boolean isGlobalWindowing(PCollection<BeamRecord> upstream) { > ``` This one is maybe silly maybe not --- * [sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 166 at r1] (https://reviewable.io:443/reviews/apache/beam/4546#-L4Ed3BO8JPOzEXPQrUd:-L4Ed3BO8JPOzEXPQrUe:b-nv7cdt) ([raw file] (https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp

> ```Java > } > > private boolean isDefaultTrigger(WindowingStrategy windowingStrategy) { > ``` This one is fine :-) --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowFactory.java, line 40 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EbkphEoAj2-gp5rhM:-L4EbkphEoAj2-gp5rhN:btos0ry) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
> ```Java > static AggregateWindowField getWindowFieldAt(RexCall call, int groupField) { > > Optional<WindowFn> window = createWindowFn(call.operands, call.op.kind); > ``` Side note - wrote the other comments about `Optional` before I saw this. So I guess you decided it was more readable to not use it in the other case? --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowFactory.java, line 57 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EbwYiA42tMLJ-5O04:-L4EbwYiA42tMLJ-5O05:bzf1j39) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
> ```Java > switch (operatorKind) { > case TUMBLE: > FixedWindows fixedWindows = FixedWindows.of(durationParameter(parameters, 1)); > ``` Nice to have a comment about the syntax here, just an illustration how the parameters show up in SQL language. And you might also make them constants, though I think with a comment it is less important. Same for the other cases. --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowFactory.java, line 59 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4Ecigv89qYyhuF3UgG:-L4Ecigv89qYyhuF3UgH:b-prgw5d) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
> ```Java > FixedWindows fixedWindows = FixedWindows.of(durationParameter(parameters, 1)); > if (parameters.size() == 3) { > fixedWindows = fixedWindows.withOffset(durationParameter(parameters, 2)); > ``` Are you sure the Calcite and Beam chose the same reference point for the offset? --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowField.java, line 36 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EbMw3L0Jx0E4saRhv:-L4EbMw3L0Jx0E4saRhw:byzocv5) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
> ```Java > private static final AggregateWindowField ABSENT = builder().build(); > public abstract int fieldIndex(); > public abstract @Nullable WindowFn<BeamRecord, ? extends BoundedWindow> windowFn(); > ``` Now that we are on Java 8, you can use `Optional` anywhere that it is not a known problem allocating too many wrappers. Not sure how much they are optimized away to behave as well as nullables but I would use them by default until you find a problem. --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowField.java, line 51 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EbCFdCRLQt5NszKRb:-L4EbCFdCRLQt5NszKRc:bl9isgr) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
> ```Java > } > > public boolean isPresent() { > ``` Now that we are on Java 8, you could use `Optional<AggregateWindowField>` to decouple the data you want to store from the present/absent. --- *[sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/BeamSqlDslAggregationTest.java, line 472 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4Ee6Zzdy24t8aYaiDW:-L4Ee6Zzdy24t8aYaiDX:b-phfdkb) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/Bean
> ```Java > String sql = "SELECT f_int2, COUNT(*) AS `size` FROM PCOLLECTION GROUP BY f_int2"; > > input = input.apply("testInheritsWindowingStrategy", BeamSql.query(sql)); > ``` This is minor, but when I skimmed this I saw the `assertEquals` below and the declaration of `input` above and thought the test was wrong. There's no perf benefit to reassigning here, so use another variable for clarity. --- *[sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/BeamSqlDslAggregationTest.java, line 475 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EeHtXE6dYlev2lAwH:-L4EeHtXE6dYlev2lAwI:b-wel2h0) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/Bean
> ```Java > > assertEquals(fixedWindows, input.getWindowingStrategy().getWindowFn()); > assertEquals(trigger.getContinuationTrigger(), input.getWindowingStrategy().getTrigger()); > ``` To be honest, I don't really care that much about equality of continuation trigger. What this confirms is that there was one GBK and the trigger wasn't rewritten, which is fine. What is better to test would actually be outputs. You could use `TestStream` to provide interesting inputs to the SQL and then `PAssert` to check what comes out in each window. --- *[sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/BeamSqlDslBase.java, line 46 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4Eegdri7czMoGIRDkG:-L4EegdsDQaipf7i9clY:b-gy4fuq) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/Bean
> ```Java > * prepare input records to test {@link BeamSql}. > * > * <p>Note that, any change in these records would impact tests in this package. > ``` This class seems like generally a bad idea. Tests are almost always read one at a time, and should essentially be just read in a straight line. It is the opposite of usual software engineering - reuse makes them less maintainable. --- *Comments from [Reviewable](https://reviewable.io:443/reviews/apache/beam/4546)* <!-- Sent from Reviewable.io -->

6. Review status: all files reviewed at latest revision, 13 unresolved discussions. --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 70 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EalKCrH5RZVocmItf:-L4F-eo4NDEKmqW88yn3:bb7f2mf) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> We should assume every object is non-null, and put the findbugs `DefaultAnnotation` on the package. (separately from this PR) </blockquote></details> done --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 132 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EdXpXI9r0vQU5sTA6:-L4EnRtO5AF4f56gJla1:bu18h98) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> We should go farther than this, perhaps. You don't have to do it in this PR, but maybe file a ticket. I believe we can also analyze some other triggers that are not meaningful. Or you could argue that if the user sets it themselves then we should respect their request to never have output... </blockquote></details> We can look into that. Filed [BEAM-3588](https://issues.apache.org/jira/browse/BEAM-3588). It's fine to handle only defaults here because it's the same check as in GBK, we're just providing more context in the exception. If new validations are added to basic PTransofrms, then in the worst case SQL will expose those without wrapping. My main concern about performing more complex validation is that I'm not sure that we can solve it in general case (or even limited case). And without it it can become confusing if you can define 2 equivalent triggers and one will be rejected while the other will not. --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rel/BeamAggregationRel.java, line 158 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4Ecx-pQb4v_ecbhmEl:-L4EtNcwNzj5e1aeAhhz:b-tw1l6x) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> This one seems a bit silly </blockquote></details> Inlined these :) I wrote the invocations first to make sure`if` looks sane, and then implementations just ended up this way :) Should have just copied the GBK version from the start --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowFactory.java, line 40 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EbkphEoAj2-gp5rhM:-L4FISdCztxez_DMCMoh:bq4wpo1) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> Side note - wrote the other comments about `Optional` before I saw this. So I guess you decided it was more readable to not use it in the other case? </blockquote></details> replied in the other comment --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowFactory.java, line 57 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EbwYiA42tMLJ-5O04:-L4EuQ_8BwHE-QKZBSq_:b1uae3a) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> Nice to have a comment about the syntax here, just an illustration how the parameters show up in SQL language. And you might also make them constants, though I think with a comment it is less important. Same for the other cases. </blockquote></details> added the syntax comment --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowFactory.java, line 59 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4Ecigv89qYyhuF3UgG:-L4IC2s4s3zcgnIcOgFP:bx3phea) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> Are you sure the Calcite and Beam chose the same reference point for the offset? </blockquote></details> discussed offline, this seems correct --- *[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowField.java, line 36 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EbMw3L0Jx0E4saRhv:-L4EvkUyZJ91drVArRXQ:b-nmdh20) ([raw file](https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> Now that we are on Java 8, you can use `Optional`

anywhere that it is not a known problem allocating too many wrappers. Not sure how much they are optimized away to behave as well as nullables but I would use them by default until you find a problem. </blockquote></details> Wrapped the whole `AggregateWindowField` into `Optional`. My impression was that using `java.util.Optional` in class fields is discouraged (e.g. `java.util.Optional` is not serializable), so I thought that domain-specific logic would be better, and it just happens to have similar semantics with `Maybe`. It could have more state and logic, in my case it probably saved an `Optional.map()`call when extracting `windowFieldIndex`. Don't have a strong opinion in this case. --- *
[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowField.java, line 51 at r1]
(https://reviewable.io:443/reviews/apache/beam/4546#-L4EbCFdCRLQt5NszKRb:-L4FIIwsVaLKY788H2Jx:bb7f2mf) ([raw file]
(https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> Now that we are on Java 8, you could use `Optional<AggregateWindowField>` to decouple the data you want to store from the present/absent. </blockquote></details> done --- *
[sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/BeamSqlDslAggregationTest.java, line 472 at r1]
(https://reviewable.io:443/reviews/apache/beam/4546#-L4Ee6Zzdy24t8aYaiDW:-L4IL5P_kmZOXEyJVFoS:bb7f2mf) ([raw file]
(https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/Bean
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> This is minor, but when I skimmed this I saw the `assertEquals` below and the declaration of `input` above and thought the test was wrong. There's no perf benefit to reassigning here, so use another variable for clarity. </blockquote></details> done --- *[sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/BeamSqlDslAggregationTest.java, line 475 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4EeHtXE6dYlev2lAwH:-L4F-CU-SJiSeyMPs-1I:b-fzcjgm) ([raw file]
(https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/Bean
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> To be honest, I don't really care that much about equality of continuation trigger. What this confirms is that there was one GBK and the trigger wasn't rewritten, which is fine. What is better to test would actually be outputs. You could use `TestStream` to provide interesting inputs to the SQL and then `PAssert` to check what comes out in each window. </blockquote>
</details> I agree, updated the tests to validate results --- *[sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/BeamSqlDslBase.java, line 46 at r1](https://reviewable.io:443/reviews/apache/beam/4546#-L4Eegdri7czMoGIRDkG:-L4EyKoHrby2Tu4_TOLH:b-rsc81d) ([raw file]
(https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/test/java/org/apache/beam/sdk/extensions/sql/Bean
<details><summary>Previously, kennknowles (Kenn Knowles) wrote…</summary><blockquote> This class seems like generally a bad idea. Tests are almost always read one at a time, and should essentially be just read in a straight line. It is the opposite of usual software engineering - reuse makes them less maintainable. </blockquote></details> I agree. This class has only 5 subclasses, should be easy to refactor. There are similar classes `BeamSqlBuiltinFunctionsIntegrationTestBase` with 7 and `BeamSqlFnExecutorTestBase` with 24. Probably there are others. I'd prefer to do this in another PR :) ---
Comments from [Reviewable](https://reviewable.io:443/reviews/apache/beam/4546) <!-- Sent from Reviewable.io -->
7. retest this please
8. this is great to get in ---
Reviewed 9 of 9 files at r2. Review status: all files reviewed at latest revision, 2 unresolved discussions. --- *
[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowFactory.java, line 116 at r2]
(https://reviewable.io:443/reviews/apache/beam/4546#-L4JPNCJzuq1qkTBBJiB:-L4JPNCJzuq1qkTBBJiC:b2nh4j5) ([raw file]
(https://github.com/apache/beam/blob/ecd89b856ffd85baa769ecbe27615ffaf79f2195/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
> ```Java > > SlidingWindows slidingWindows = SlidingWindows > .of(durationParameter(parameters, 2)) > ``` :-) --- *
[sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/impl/rule/AggregateWindowField.java, line 36 at r1]
(https://reviewable.io:443/reviews/apache/beam/4546#-L4EbMw3L0Jx0E4saRhv:-L4JPbZzQUY-VEGDM0bf:b-833xbc) ([raw file]
(https://github.com/apache/beam/blob/33eeedfeae4d7534265f237549ff9eb4b5f3d58f/sdks/java/extensions/sql/src/main/java/org/apache/beam/sdk/extensions/sql/imp
<details><summary>Previously, akedin (Anton Kedin) wrote…</summary><blockquote> Wrapped the whole `AggregateWindowField` into `Optional`. My impression was that using `java.util.Optional` in class fields is discouraged (e.g. `java.util.Optional` is not serializable), so I thought that domain-specific logic would be better, and it just happens to have similar semantics with `Maybe`. It could have more state and logic, in my case it probably saved an `Optional.map()`call when extracting `windowFieldIndex`. Don't have a strong opinion in this case. </blockquote></details> Wow, didn't realize the `Optional` was not serializable. Found https://stackoverflow.com/questions/24547673/why-java-util-optional-is-not-serializable-how-to-serialize-the-object-with-suc Pretty arrogant for them to speak as though they were "designing" `Optional` since it predates even the first version of Java (same with generics, so I guess there's a pattern). And pretty annoying for them to take a purist stance, since Java is a pragmatic compromise language. This `Serializable` issue is a good argument for not including it in fields. And since that is a stupid thing to have to remember, we might just want to ban it (pretty easy to use static analysis to ban classes) and go back to using Guava's `Optional` or rolling our own. Sheesh. --- *Comments from [Reviewable](https://reviewable.io:443/reviews/apache/beam/4546#-:-L4J_yNoGn4Z-sF51pD7:b-ikyoki)* <!-- Sent from Reviewable.io -->
9. retest this please
10. tis green!

github_pulls_reviews:

jira_issues:

jira_issues_comments: