Item 119
git_comments:

git_commits:

1. **summary:** [SPARK-22829] Add new built-in function date_trunc()
   message: [SPARK-22829] Add new built-in function date_trunc() ## What changes were proposed in this pull request? Adding date_trunc() as a built-in function. `date_trunc` is common in other databases, but Spark or Hive does not have support for this. `date_trunc` is commonly used by data scientists and business intelligence application such as Superset (https://github.com/apache/incubator-superset). We do have `trunc` but this only works with 'MONTH' and 'YEAR' level on the DateType input. date_trunc() in other databases: AWS Redshift: http://docs.aws.amazon.com/redshift/latest/dg/r_DATE_TRUNC.html PostgreSQL: https://www.postgresql.org/docs/9.1/static/functions-datetime.html Presto: https://prestodb.io/docs/current/functions/datetime.html ## How was this patch tested? Unit tests (Please explain how this patch was tested. E.g. unit tests, integration tests, manual tests) (If this patch involves UI changes, please attach a screenshot; otherwise, remove this) Please review http://spark.apache.org/contributing.html before opening a pull request. Author: Youngbin Kim <ykim828@hotmail.com> Closes #20015 from youngbink/date_trunc.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-22829] Add new built-in function date_trunc()
   body: ## What changes were proposed in this pull request? Adding date_trunc() as a built-in function. `date_trunc` is common in other databases, but Spark or Hive does not have support for this. `date_trunc` is commonly used by data scientists and business intelligence application such as Superset (https://github.com/apache/incubator-superset). We do have `trunc` but this only works with 'MONTH' and 'YEAR' level on the DateType input. date_trunc() in other databases: AWS Redshift: http://docs.aws.amazon.com/redshift/latest/dg/r_DATE_TRUNC.html PostgreSQL: https://www.postgresql.org/docs/9.1/static/functions-datetime.html Presto: https://prestodb.io/docs/current/functions/datetime.html ## How was this patch tested? Unit tests (Please explain how this patch was tested. E.g. unit tests, integration tests, manual tests) (If this patch involves UI changes, please attach a screenshot; otherwise, remove this) Please review http://spark.apache.org/contributing.html before opening a pull request.

github_pulls_comments:

1. @gatorsmile @cloud-fan
2. ok to test
3. we need to create a JIRA ticket
4. @cloud-fan and @youngbink how about reviving https://github.com/apache/spark/pull/14788 with a configuration to control this? AWS Redshift seems having `TRUNC` which just converts a timestamp to a date whereas we have Spark's `trunc` which supports date formats. This is not quite equivalent. I think Spark's `trunc` is more like Redshift's `DATE_TRUNC`. PostgreSQL does not have `trunc` but has `date_trunc` where we can specify the format and returns a timestamp always. Presto also looks not having a duplicated functionality. I think we can simply introduce an alias for `trunc` after resolving https://github.com/apache/spark/pull/14788 if the naming matters. Did I maybe miss something?
5. @HyukjinKwon Just took a look at this PR #14788. My point of mentioning those databases was just to give examples of the function that Spark doesn't support but other databases commonly do. (They all have this `date_trunc` which takes `timestamp` and output `timestamp`) As you said, we could extend `trunc` and simply create an alias `date_trunc`, but it's actually not as simple. For e.g, PR #14788 won't be able to handle the following command collectly on PySpark: ``` df = spark.createDataFrame([('1997-02-28 05:02:11',)], ['d']) df.select(functions.trunc(df.d, 'year').alias('year')).collect() df.select(functions.trunc(df.d, 'SS').alias('SS')).collect() ``` This is because `trunc(string, string)` isn't correctly handled. We could find a way around this and get it working, but after having a discussion with

@cloud-fan, @gatorsmile, @rednaxelafx and Reynold, we thought adding `date_trunc` is the simplest way for now.

6. > after having a discussion with @cloud-fan, @gatorsmile, @rednaxelafx and Reynold Where did the discussion happen? Was this offline discussion? I also want to actively join in the discussion. Many implementations of the trunc works differently and I think we decide the "right" behaviour after sufficient discussion. If we don't fix the stuff about #14788 in 2.3.0 timeline, it could be even more difficult because we need to keep the previous behaviour.
7. **[Test build #85083 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/85083/testReport)** for PR 20015 at commit [`f94f401`] (https://github.com/apache/spark/commit/f94f401bcfd765b21c3fb466041b42a605d6a814). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
8. OK. I am fine if you all guys strongly feel about this.
9. Yeah keep any substantive discussion on the public lists. Sometimes a side conversation happens; summarize the points here. We've rejected a lot of other functions that other DBs, but not Hive, support. Spark mostly follows Hive, and for everything else, there are UDFs. I'm not against this so much as not clear why it's exceptional
10. We had an offline discussion and wanna send this out to get more feedbacks. So generally just adding `date_trunc` is pretty straightforward and makes Spark consistent with other databases about this function, while extending `trunc` to support timestamp type is a better API design.
11. If we haven't get a similar function, I would have gone +1 but what I am less sure is `date_trunc` actually quite sounds a better version of `trunc` to be honest. Seems both also extend the same parent here `TruncTime`. I feel like we are trying to add this better version alone by working around because it takes a relatively larger change to update other related functions consistently.
12. I get `date_trunc` is common in other DBMS. I can see that this can be done now and we can still proceed `trunc`, etc. later. So, I am fine but still less sure tho.
13. hmm...even if we decide to change this later, I honestly think merging `trunc` and `date_trunc` would be simple, only touching a couple of files (mostly `datetimeExpressions.scala`). This PR isn't too small as you said, but most of the codes here can be used without modification if we are to merge `date_trunc`.
14. SPARK-17174 originally described few functions related with hour, min, etc. but I received an advice to fix up other related functions too even though they could also be done alone too. I agreed with doing other functions too at that time and I tried to propose as so. I am saying I think this PR actually more targets adding another (better) version of `trunc` to support day, hour, min, etc. in the format. In this case, I think we should deduplicate/support the logics with related functions too. Ah, so, I think I am less sure about why this should be done alone leaving out other related changes, and other functions we (I) usually reject. and I think you and @cloud-fan say the reasons are, it's common and this PR targets a separate functionality consistent with other DBMS.
15. - The API proposed by this PR is consistent with the other DBs. - The implementation does not introduce the behavior changes. The implementation is clean and the PR quality is pretty good.
16. **[Test build #85131 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/85131/testReport)** for PR 20015 at commit [`3547b7c`] (https://github.com/apache/spark/commit/3547b7c0bf8caa972e018f6875e4b0f599c4e12f). * This patch **fails Spark unit tests**. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `trait TruncInstant extends BinaryExpression with ImplicitCastInputTypes `
17. **[Test build #85132 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/85132/testReport)** for PR 20015 at commit [`b12ba92`] (https://github.com/apache/spark/commit/b12ba92add942c087dd45933464937479fc24bcd). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `trait TruncInstant extends BinaryExpression with ImplicitCastInputTypes `
18. **[Test build #85140 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/85140/testReport)** for PR 20015 at commit [`80a1959`] (https://github.com/apache/spark/commit/80a195989bd138fade5910cbf495b472a57ce445). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `trait TruncInstant extends BinaryExpression with ImplicitCastInputTypes `
19. **[Test build #85141 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/85141/testReport)** for PR 20015

at commit [`0d1a8cb`] (https://github.com/apache/spark/commit/0d1a8cbc922bc410d8d4a69c26b16290773a197c). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `trait TruncInstant extends BinaryExpression with ImplicitCastInputTypes `
20. LGTM
21. Thanks! Merged to master
22. **[Test build #85146 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/85146/testReport)** for PR 20015 at commit [`238d7d4`] (https://github.com/apache/spark/commit/238d7d470c583c910bccbca8bbcaa681b67d6025). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `trait TruncInstant extends BinaryExpression with ImplicitCastInputTypes `

github_pulls_reviews:

1. Can we use a timestamp string like `1997-02-28 05:02:11` to show the difference from `trunc` a bit more clearly?
2. Seems `input` and `truncFunc` descriptions missing.
3. nit: Since this is a time, it can be `val t = ...`
4. Maybe `TruncInstant`? I received this advice before and I liked it too. Not a big deal tho.
5. `// unknown format or too small level`?
6. nit: one space each more.
7. Could we make those lowercased too?
8. Maybe, `'second', 'minute', 'hour', 'week', 'month' and 'quarter'`
9. Why do we need a type parameter `T`?
10. Can we bring quarter and week forward, maybe to 3 and 4? Then it's more conform to the order of time granularity and max-level design is not influenced.
11. Maybe `truncFunc: (Any, Int) => Any` is enough? So we don't need to use the `T`, but I'm not sure if this is better...
12. Let us use the lower case and also update the other functions in this file. For example, `ToUnixTimestamp`
13. nit: d -> ts or t
14. Nit: `YYYY` -> `yyyy`
15. Also update the original `trunc`
16. Maybe, `time` -> `instant`.
17. Remove `@return`
18. Remove this line.
19. `date` -> `ts`.
20. `d` -> `t` or `ts`.
21. `test` -> `testTrunc` ?

jira_issues:

jira_issues_comments: