Item 307
git_comments:

1. this is super hackey, https://issues.apache.org/jira/browse/ARROW-7855

git_commits:

1. **summary:** Catch TypeError on PyArrow array instantiation (#9139)
   message: Catch TypeError on PyArrow array instantiation (#9139)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Catch TypeError on PyArrow array instantiation
   body: ### CATEGORY Choose one - [x] Bug Fix - [ ] Enhancement (new features, refinement) - [ ] Refactor - [ ] Add tests - [ ] Build / Development Environment - [ ] Documentation ### SUMMARY The following (Postgres) query fails with `TypeError: an integer is required (got type str)`: ``` SELECT json_build_object('test', json_build_array(123456, 'test')) AS data; ``` This is also raised when columnar data from Presto is in the following format (mixed array data types): ``` {'TestKey': [123456, 'foo']} ``` See also: https://github.com/apache/incubator-superset/pull/9096#issuecomment-584449351 *NOTE: this is meant to be a temporary fix until the [Arrow ticket](https://issues.apache.org/jira/browse/ARROW-7855) is resolved.* ### BEFORE/AFTER SCREENSHOTS OR ANIMATED GIF <!--- Skip this if not applicable --> ### TEST PLAN Ensure the following query against Postgres succeeds: ``` SELECT json_build_object('test', json_build_array(123456, 'test')) AS data; ``` ### ADDITIONAL INFORMATION <!--- Check any relevant boxes with "x" --> <!--- HINT: Include "Fixes #nnn" if you are fixing an existing issue --> - [ ] Has associated issue: - [ ] Changes UI - [ ] Requires DB Migration. - [ ] Confirm DB Migration upgrade and downgrade tested. - [ ] Introduces new feature or API - [ ] Removes existing feature or API ### REVIEWERS @john-bodley @serenajiang @nytai @willbarrett @dpgaspar @suddjian

github_pulls_comments:

1. # [Codecov](https://codecov.io/gh/apache/incubator-superset/pull/9139?src=pr&el=h1) Report > Merging [#9139](https://codecov.io/gh/apache/incubator-superset/pull/9139?src=pr&el=desc) into [master](https://codecov.io/gh/apache/incubator-superset/commit/30b7064e309c8f6d080e1c81ab3684b4a949a496?src=pr&el=desc) will **not change** coverage. > The diff coverage is `n/a`. [![Impacted file tree graph](https://codecov.io/gh/apache/incubator-superset/pull/9139/graphs/tree.svg?width=650&token=KsB0fHcx6l&height=150&src=pr)](https://codecov.io/gh/apache/incubator-superset/pull/9139?src=pr&el=tree) ```diff @@ Coverage Diff @@ ## master #9139 +/- ## ========== Coverage 59.1% 59.1% ========== Files 372 372 Lines 11922 11922 Branches 2919 2919 ========== Hits 7046 7046 Misses 4694 4694 Partials 182 182 ``` ------ [Continue to review full report at Codecov](https://codecov.io/gh/apache/incubator-superset/pull/9139?src=pr&el=continue). > **Legend** - [Click here to learn more](https://docs.codecov.io/docs/codecov-delta) > `Δ = absolute <relative> (impact)`, `ø = not affected`, `? = missing data` > Powered by [Codecov](https://codecov.io/gh/apache/incubator-superset/pull/9139?src=pr&el=footer). Last update [30b7064...a6c7892](https://codecov.io/gh/apache/incubator-superset/pull/9139?src=pr&el=lastupdated). Read the [comment docs](https://docs.codecov.io/docs/pull-request-comments).
2. Does this result in `123456` being returned as a string?
3. @serenajiang it results in the entire column value being returned as a JSON string: `{"TestKey": [123456, "foo"]}`. `123456` is an integer when the JSON is parsed.

github_pulls_reviews:

jira_issues:

1. **summary:** [Python] Always raise ArrowTypeError instead of TypeError inside pyarrow.array
   description: The following data structure passed to `pa.array` raises a generic `TypeError`: {code:java} import pyarrow as pa pa.array([{'TestKey': [123456, 'foo']}]) {code} {code:java} Traceback (most recent call last): File "pyarrow_list_test.py", line 30, in <module> pa_array = pa.array([\{'TestKey': [123456, 'foo']}]) File "pyarrow/array.pxi", line 269, in pyarrow.lib.array File "pyarrow/array.pxi", line 38, in pyarrow.lib._sequence_to_array TypeError: an integer is required (got type str) {code} I understand there may be a way to overcome this by setting the `type` value as an argument to `pa.array`, but the use case here is storing results of a SQL query where the structure/type of the column is unknown. If Arrow is ultimately unable to handle this data structure without a predefined `type` passed to `pa.array`, can the exception at least us the PyArrow namespace (e.g. `pa.lib.ArrowTypeError` or `pa.lib.ArrowNotImplementedError). Any other workaround suggestions welcome.

jira_issues_comments:

1. What would you exactly expect in this case? Such a dict will be interpreted as a StructType, and the values of a struct can be lists (as you have here), but Arrow only supports homogeneously typed lists (unless you use a UnionType, but that is not something that can be inferred I think)
2. [~robdiciuccio] I am curious what you want to do with the data after you create the array. If you're interacting with SQL, at some point you need to homogenize the data to all be of the same type. There are issues about inferring unions (see ARROW-2774) but this will probably be something that has to be opted in to explicitly like {{pa.array(data, allow_unions=True)}} once it is even implemented
3. **body:** [~wesm] We're ultimately creating a PyArrow Table from SQL query (DB-API) results. In some cases (with databases such as Presto, Postgres, etc.) there may be nested data structures such as the one above in a column. We're using PyArrow to infer column data types, as DB-API drivers vary greatly in the metadata that's returned, and numpy/pandas lack proper support for nullable data types. We're also using Arrow as a means to serialize data to disk and rehydrate to a new Table instance. We've been working around some of the issues with nested data types by serializing to JSON strings, which is not ideal, but is a functional workaround. Here's some additional context on how we're handling this: https://github.com/apache/incubator-superset/pull/9139 Any suggestions on how to better handle nested data types of unknown shape is appreciated.
   label: code-design
4. I'm not sure what API would be helpful to you, if there is one other than what is already available we can take a closer look. I agree that raising {{ArrowTypeError}} (inheriting from TypeError) consistently or {{ArrowInvalid}} (inheriting from ValueError) (https://github.com/apache/arrow/blob/master/python/pyarrow/error.pxi#L27) would be better.
5. While ArrowTypeError exists, it doesn't seem useful to me to mandate that PyArrow never raises a plain TypeError. Closing.