Error handling in Athena Connector CDK

SUMMARY:

In order to use the existing athena connector, we need better understanding for the current Runtime Exception in athena connectors and classify them into the following error code.

EXCEPTION CLASS

AthenaConnectorException: https://github.com/awslabs/aws-athena-query-federation/blob/master/athena-federation-sdk/src/main/java/com/amazonaws/athena/connector/lambda/exceptions/AthenaConnectorException.java#L46

```
FederationSourceErrorCode {
    AccessDeniedException("AccessDeniedException"),
    EntityNotFoundException("EntityNotFoundException"),
    InvalidCredentialsException("InvalidCredentialsException"),
    InvalidInputException("InvalidInputException"),
    InvalidResponseException("InvalidResponseException"),
    OperationTimeoutException("OperationTimeoutException"),
    OperationNotSupportedException("OperationNotSupportedException"),
    InternalServiceException("InternalServiceException"),
    PartialFailureException("PartialFailureException"),
    ThrottlingException("ThrottlingException");
}
```

Sample Exception Snippet:

```
try {
  // this block might throw a NullPointerException
} catch (NullPointerException e) {
  // assume this null pointer exception corresponds to an entity not found exception
  throw new AthenaConnectorException(e.getMessage(),
                new ErrorDetails(FederationSourceErrorCode.EntityNotFoundException, e.
}
```

1. ACCESSDENIEDEXCEPTION

When to Use: This exception should be used when a user tries to perform an operation or access a resource that they do not have permission for.

Common Scenarios: A user tries to access or perform an operation that is restricted based on the user's iam role

2. ENTITYNOTFOUNDEXCEPTION

When to Use: This exception is thrown when an entity (such as a record, table, or schea) is not found in a data store or database when it is expected to be present.

Common Scenarios: Querying a database for an item that has been deleted or never existed.

3. INVALIDCREDENTIALSEXCEPTION

When to Use: This exception should be used when the credentials provided (such as username and password) are invalid or incorrect.

Common Scenarios: User attempt to use Username and Password to access DB Server and authentication fails due to incorrect credentials.

4. INVALIDINPUTEXCEPTION

When to Use: This exception is thrown when the input provided to a method or function is invalid or does not meet the expected format or constraints.

Common Scenarios: receives parameters that do not adhere to required constraints or formats.

5. INVALIDRESPONSEEXCEPTION

When to Use: This exception should be used when the response from an external service or API is not in the expected format or is otherwise invalid.

Common Scenarios: Receiving a malformed JSON response or response does not contain expected fields or values.

6. OPERATIONTIMEOUTEXCEPTION

When to Use: This exception indicates that an operation or request took longer than the allowed time limit or timeout period.

Common Scenarios: A network request or database query exceeds the maximum allowed time.

7. OPERATIONNOTSUPPORTEDEXCEPTION

When to Use:This exception is thrown when an operation is requested that is not supported by the current context or implementation.

Common Scenarios: Performing an operation that is not supported by the current version or configuration.

8. INTERNALSERVICEEXCEPTION

When to Use: This exception is used when there is an internal error or issue within a service or component that prevents it from completing the request.

Common Scenarios: Service encounters an unexpected error or exception internally.

9. THROTTLINGEXCEPTION

When to Use: This exception indicates that a request has been throttled due to exceeding rate limits or quotas imposed by a service.

Common Scenarios: Making too many API requests in a short period or exceeding allowed resource usage or request quotas.

GENERIC EXCEPTION:

NullPointerException: **InvalidInputException**

IllegalArgumentException: **InvalidInputException**

TimeoutException: **OperationTimeoutException**

CONNECTOR SPECIFIC EXCEPTION:

We will use DynamoDB as an example to demonstrate the use case of each exception class. We need to look through code path for especially the following APIs:

- doListSchema
- doListTables
- doGetTables
- doGetTableLayout
- GetDataSourceCapabilitiesRequest
- doGetSplits
- doReadRecord

DynamoDbException: User: arn:aws:sts::xxxxxxxx:assumed-role/some-aws-role/dynamodbathena is not authorized to perform: dynamodb:Scan on resource

AccessDeniedException: AccessDeniedException

Caused by: software.amazon.awssdk.services.dynamodb.model.DynamoDbException: 1 validation error detected: Value '1000' at 'limit' failed to satisfy constraint: Member must have value less than or equal to 100

DynamoDbException: **InvalidInputException (code ref)**

COLUMN_NOT_FOUND: line 1:8: Column 'test' cannot be resolved or requester is not authorized to access requested resources

InvalidInputException("InvalidInputException"),

TimeoutException: OperationTimeoutException

software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException: Requested resource not found

EntityNotFoundException

IllegalStateException: "No metadata partition_type defined in Schema"

InvalidInputException("InvalidInputException") (code ref)

IOException (code ref)

InternalServiceException