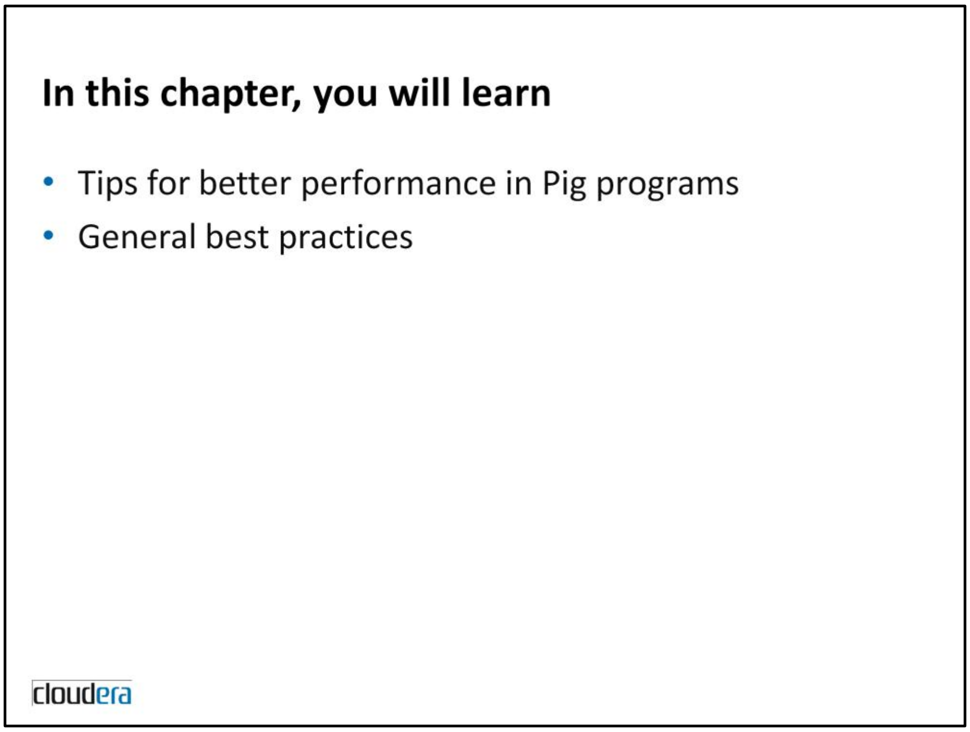
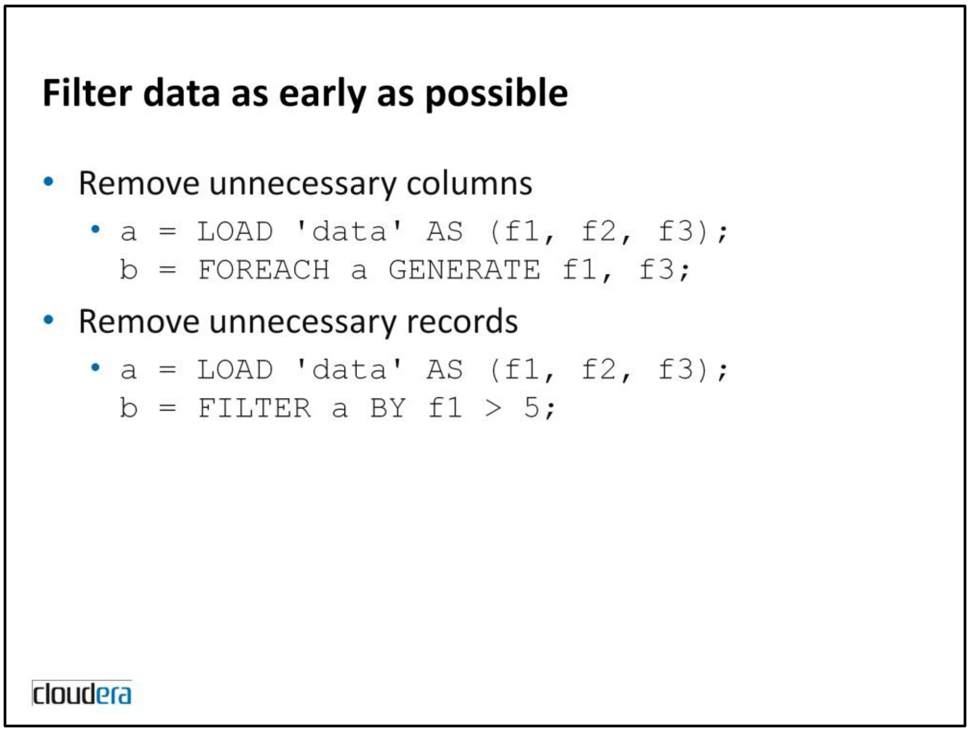
Best Practices



In this chapter, you will learn

- Tips for better performance in Pig programs
- General best practices



Filter data as early as possible

- Remove unnecessary columns

  -
    ```
    a = LOAD 'data' AS (f1, f2, f3);
    b = FOREACH a GENERATE f1, f3;
    ```
- Remove unnecessary records

  -
    ```
    a = LOAD 'data' AS (f1, f2, f3);
    b = FILTER a BY f1 > 5;
    ```

cloudera

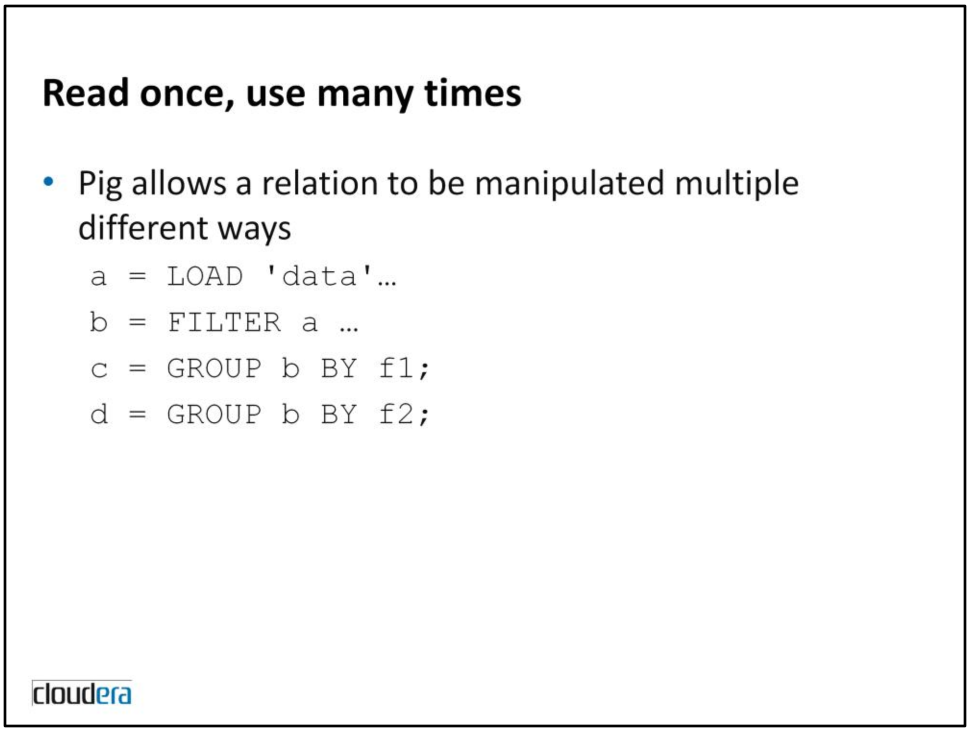
As early as possible, eliminate columns or records that are not needed so Pig does not have to carry along extra data throughout the pipeline.

To eliminate columns, use `FOREACH`:

```
a = LOAD 'data' AS (f1, f2, f3);
-- inform Pig that f2 is not needed:
b = FOREACH a GENERATE f1, f3;
```

To eliminate whole records, use `FILTER`:

```
a = LOAD 'data' AS (f1, f2, f3);
b = FILTER a BY f1 > 5;
```



Read once, use many times

- Pig allows a relation to be manipulated multiple different ways

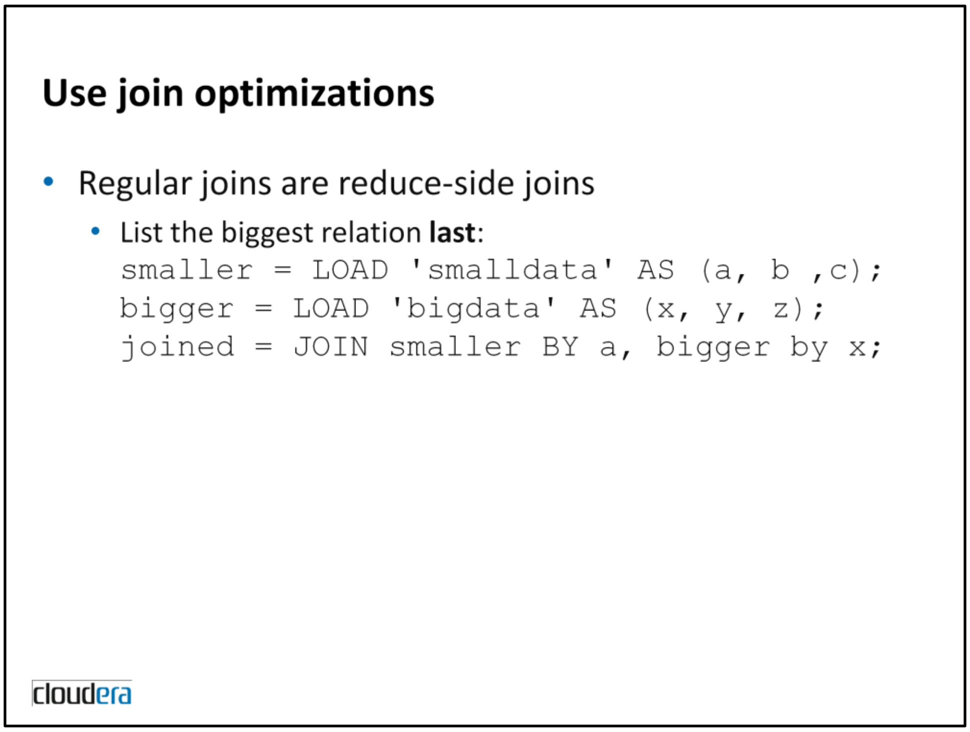
```
a = LOAD 'data'…
b = FILTER a …
c = GROUP b BY f1;
d = GROUP b BY f2;
```

cloudera

Relations in Pig can be re-used for different types of analysis. This means the data can be streamed a single time but processed multiple different ways. This is especially useful in a script (not interactive mode) where you can issue multiple `STORE` operations in the sample Pig program.

Example:

```
a = LOAD…
b = FILTER a…
c = GROUP b BY f1;
results1 = FOREACH c GENERATE…
STORE results1 INTO 'results1';
d = GROUP b BY f2;
results2 = FOREACH d GENERATE…
STORE results2 INTO 'results2';
```



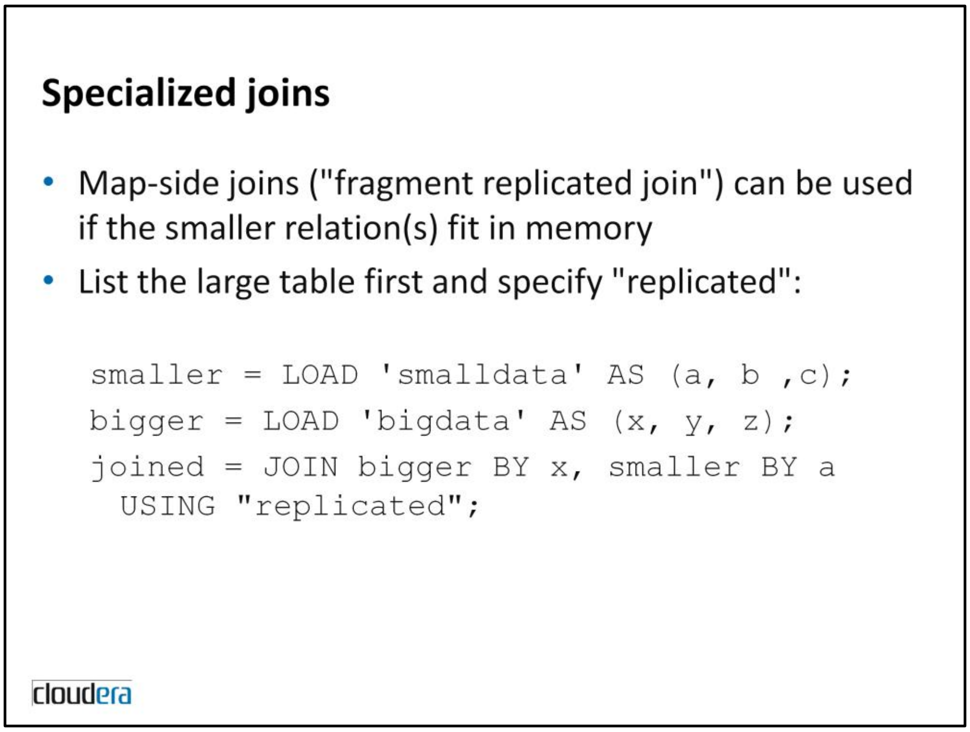
Use join optimizations

- Regular joins are reduce-side joins
  - List the biggest relation **last**:

```
smaller = LOAD 'smalldata' AS (a, b ,c);
bigger = LOAD 'bigdata' AS (x, y, z);
joined = JOIN smaller BY a, bigger by x;
```

A regular join operation in Pig is executed as a reduce-side join. Pig has an optimization where the last relation in the join is guaranteed to be streamed and not loaded into memory. Therefore, for better performance list the larger relation last:

```
smaller = LOAD 'smalldata' AS (a, b ,c);
bigger = LOAD 'bigdata' AS (x, y, z);
joined = JOIN smaller BY t, bigger by x;
```



Specialized joins

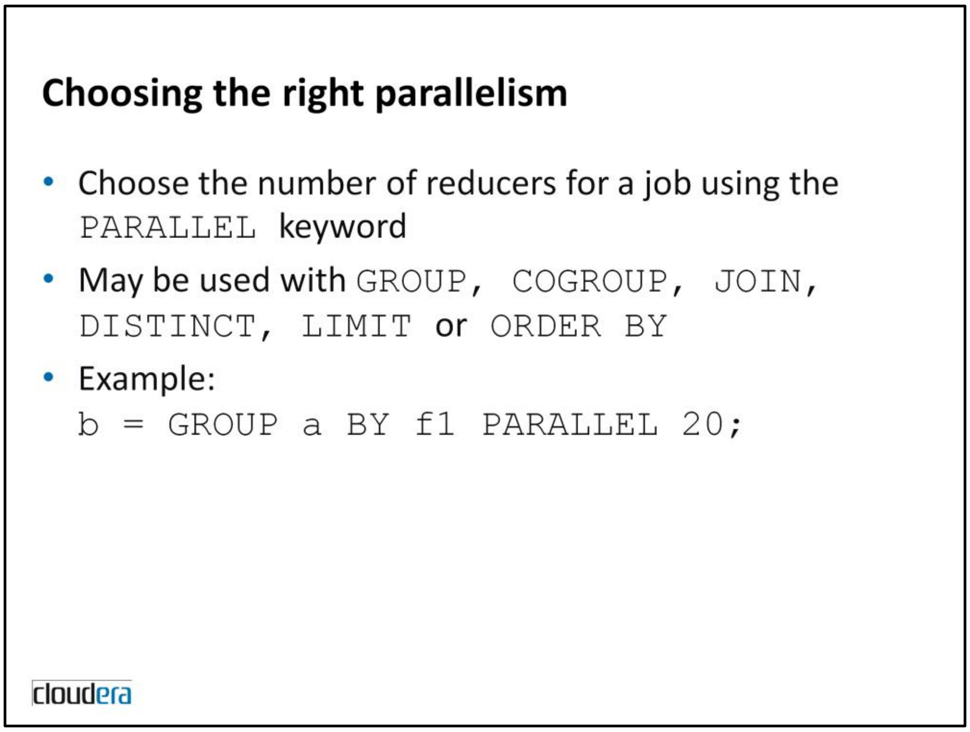
- Map-side joins ("fragment replicated join") can be used if the smaller relation(s) fit in memory
- List the large table first and specify "replicated":

```
smaller = LOAD 'smalldata' AS (a, b ,c);
bigger = LOAD 'bigdata' AS (x, y, z);
joined = JOIN bigger BY x, smaller BY a
  USING "replicated";
```

Pig supports a "fragment replicated" join which performs the join work in the map side instead of the reduce side. This can be much more performant, however, it requires that the smaller relations fit in memory.

Pig will not perform a map-side join unless you include the clause `USING "replicated"` and list the smaller relation(s) last:

```
smaller = LOAD 'smalldata' AS (a, b ,c);
bigger = LOAD 'bigdata' AS (x, y, z);
joined = JOIN bigger BY x, smaller BY a USING
"replicated";
```


Choosing the right parallelism

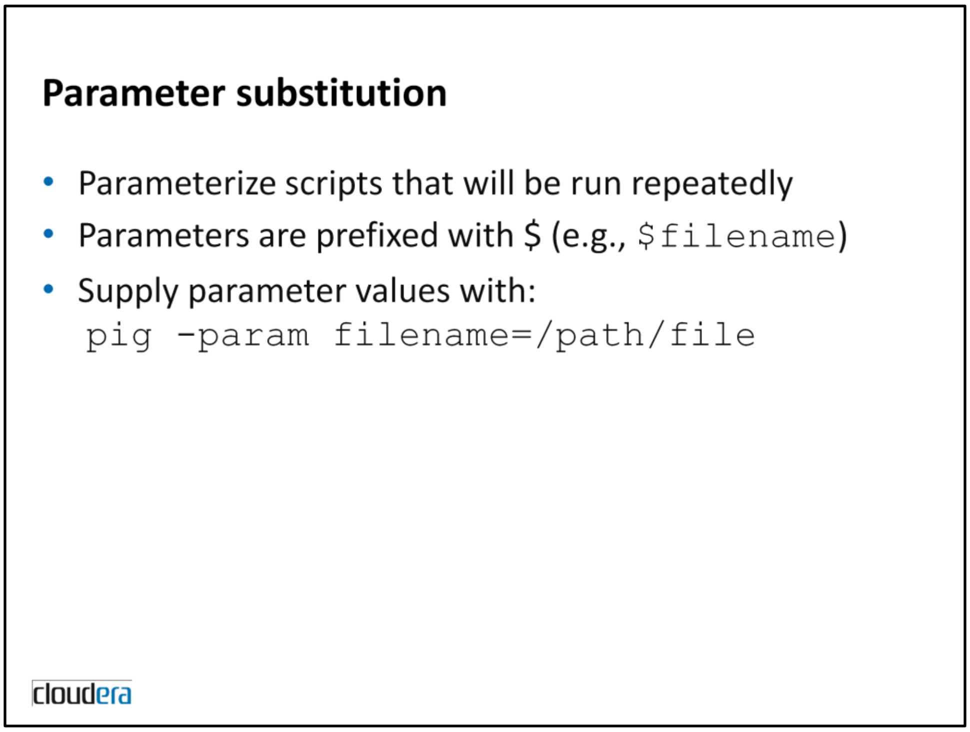
- Choose the number of reducers for a job using the `PARALLEL` keyword
- May be used with `GROUP, COGROUP, JOIN, DISTINCT, LIMIT` or `ORDER BY`
- Example:

```
b = GROUP a BY f1 PARALLEL 20;
```

cloudera

The `PARALLEL` keyword can be used to specify the number of reducers that are used for any operation that uses the reduce phase. This includes `GROUP, COGROUP, JOIN, DISTINCT, LIMIT` and `ORDER BY`.

If PARALLEL is not specified, then Pig will take Hadoop's default (mapred.reduce.tasks). On a small cluster, it may be suitable to allow the Hadoop administrator to choose a reasonable default for this setting. However, in large clusters, it is common to need a different number of reducers based on the type of job.


Parameter substitution

- Parameterize scripts that will be run repeatedly
- Parameters are prefixed with $ (e.g., `$filename`)
- Supply parameter values with:
  `pig -param filename=/path/file`

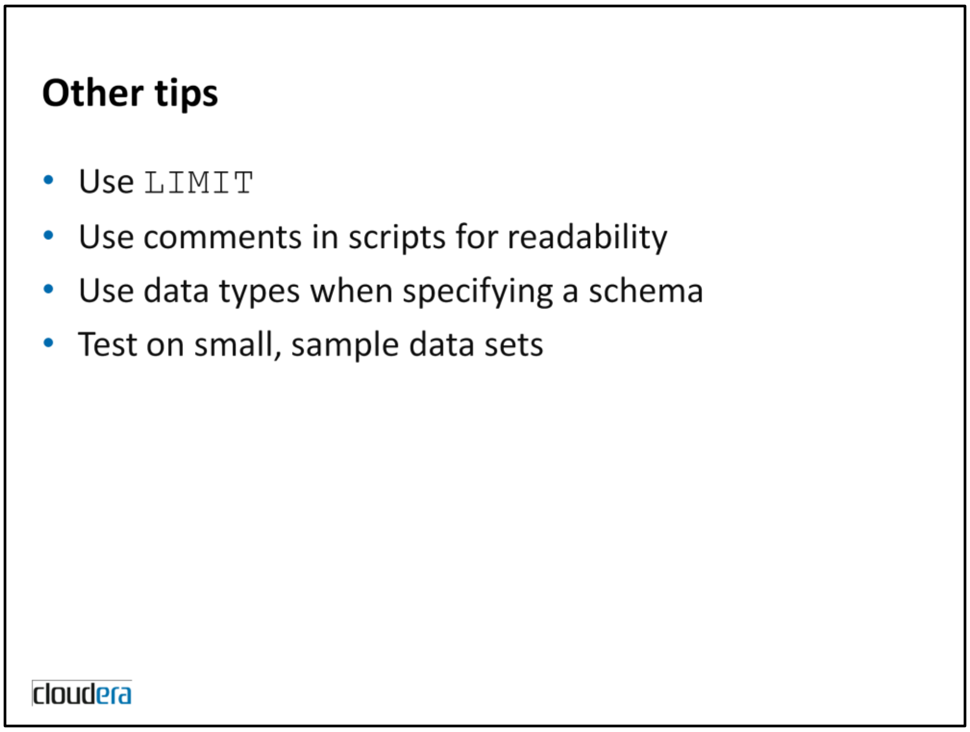
Pig scripts that will run repeatedly may need dynamic values passed in each time they execute. Pig scripts can take parameters for this purpose. In the script, prefix parameters with $. At runtime, provide the -param option. Example:

```
-- pig script
a = LOAD '$input' AS (a:int, b:chararray);
…
STORE result INTO '$output';


$ pig -param input=/user/training/logs -param
output=/user/training/processed
```

Default values can also be supplied in the Pig script, which will be used if a parameter value is not passed in:

```
-- pig script
%default input /user/training/logs
```



Other tips

- Use `LIMIT`
- Use comments in scripts for readability
- Use data types when specifying a schema
- Test on small, sample data sets

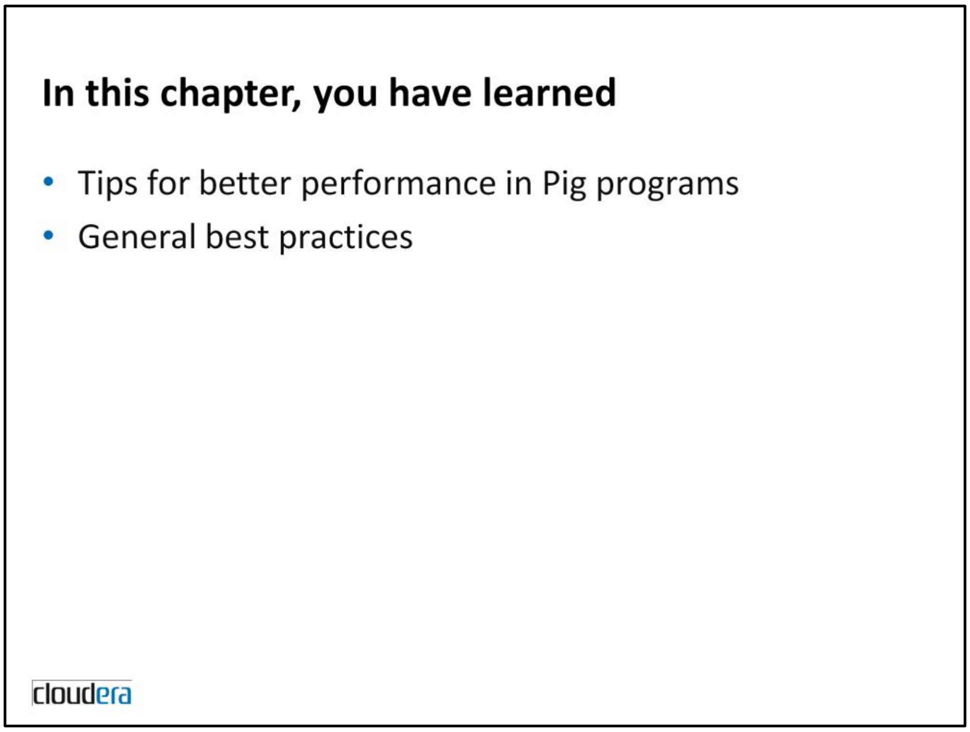
cloudera

Other tips:

• Use **`LIMIT`**: If you don't need to return all the records (e.g., looking for top-n results), then use `LIMIT`. Pig can push the `LIMIT` execution as early as possible to avoid doing extra work.

• Adding **comments** to Pig scripts adds readability

• Use data types when specifying a schema whenever possible. This will ensure that bad/missing values can be detected. It also leads to faster arithmetic computation if Pig does not have to implicitly cast bytearrays to numbers.



In this chapter, you have learned

- Tips for better performance in Pig programs
- General best practices