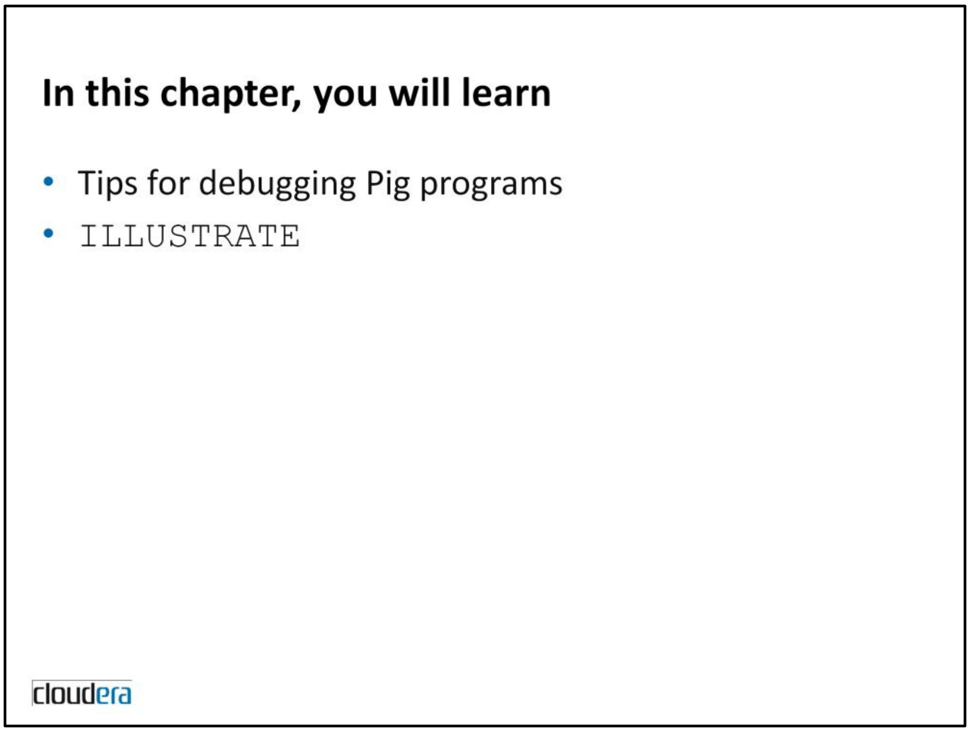
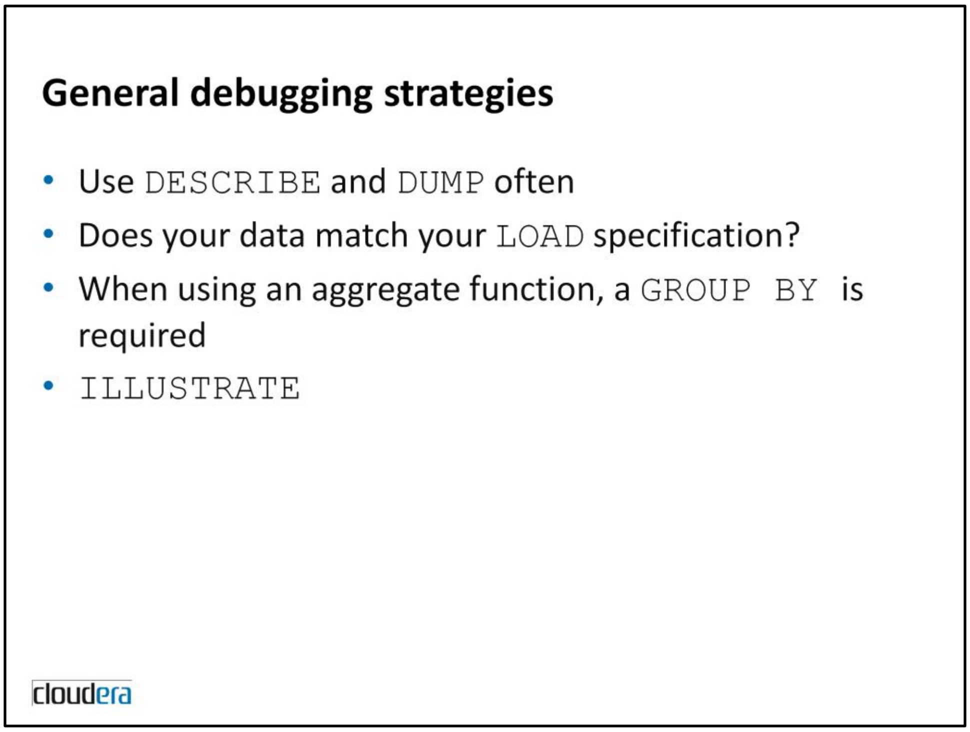
Debugging



In this chapter, you will learn

- Tips for debugging Pig programs
- `ILLUSTRATE`


General debugging strategies

- Use DESCRIBE and DUMP often
- Does your data match your LOAD specification?
- When using an aggregate function, a GROUP BY is required
- ILLUSTRATE

cloudera

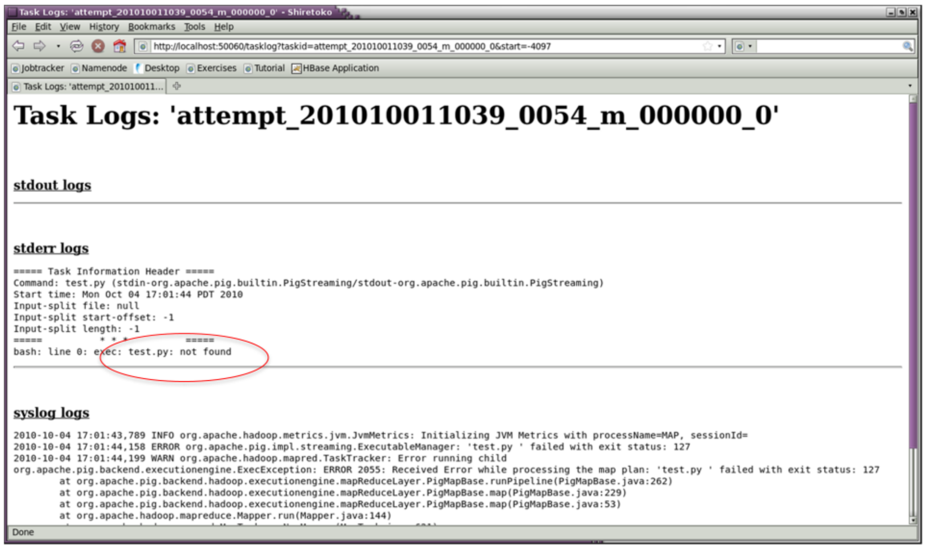
DESCRIBE and DUMP can be very useful for debugging Pig programs. Remember that you can use these on any relation (i.e., alias). This is perfectly legal:

```
a = LOAD 'data';
DESCRIBE a;
b = FILTER a BY $0 == 42;
DUMP b;
c = GROUP b BY $1;
DESCRIBE c;
```

Another common cause of issues is the input data. Does it have the data format you expect? Are there missing or extra fields?

A useful command for testing a Pig program is ILLUSTRATE. We'll see this in detail later.

JobTracker - http://localhost:50030/jobtracker.jsp


Task Logs: 'attempt_201010011039_0054_m_000000_0' - Shiretoko
File Edit View History Bookmarks Tools Help
http://localhost:50060/tasklog?taskid=attempt_201010011039_0054_m_000000_0&start=-4097
Jobtracker Namenode Desktop Exercises Tutorial HBase Application
Task Logs: 'attempt_201010011...
Done

Task Logs: 'attempt_201010011039_0054_m_000000_0'

stdout logs

stderr logs

```
===== Task Information Header =====
Command: test.py (stdin-org.apache.pig.builtin.PigStreaming/stdout-org.apache.pig.builtin.PigStreaming)
Start time: Mon Oct 04 17:01:44 PDT 2010
Input-split file: null
Input-split start-offset: -1
Input-split length: -1
=====          * * *          =====
bash: line 0: exec: test.py: not found
```

syslog logs

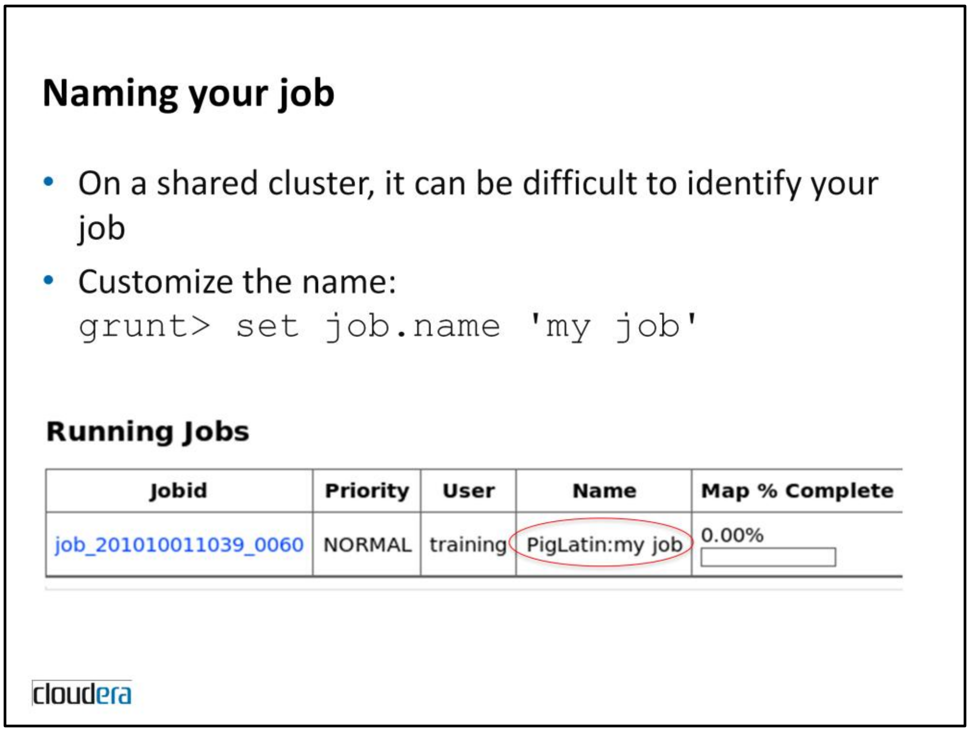
```
2010-10-04 17:01:43,789 INFO org.apache.hadoop.metrics.jvm.JvmMetrics: Initializing JVM Metrics with processName=MAP, sessionId=
2010-10-04 17:01:44,158 ERROR org.apache.pig.impl.streaming.ExecutableManager: 'test.py ' failed with exit status: 127
2010-10-04 17:01:44,199 WARN org.apache.hadoop.mapred.TaskTracker: Error running child
org.apache.pig.backend.executionengine.ExecException: ERROR 2055: Received Error while processing the map plan: 'test.py ' failed with exit status: 127
        at org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapBase.runPipeline(PigMapBase.java:262)
        at org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapBase.map(PigMapBase.java:229)
        at org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapBase.map(PigMapBase.java:53)
        at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:144)
```

cloudera

As with any Hadoop program, the JobTracker's Web interface can be a useful source of information, especially if a job is failing.

1. Go to the main JobTracker page: http://localhost:50030/jobtracker.jsp
2. Find the failed job
3. Look at the last 4KB of the log file

In the above screen capture, the job was failing because it referenced a script in a `STREAM` operation, but the script file did not exist.



Naming your job

- On a shared cluster, it can be difficult to identify your job
- Customize the name:
  ```
  grunt> set job.name 'my job'
  ```

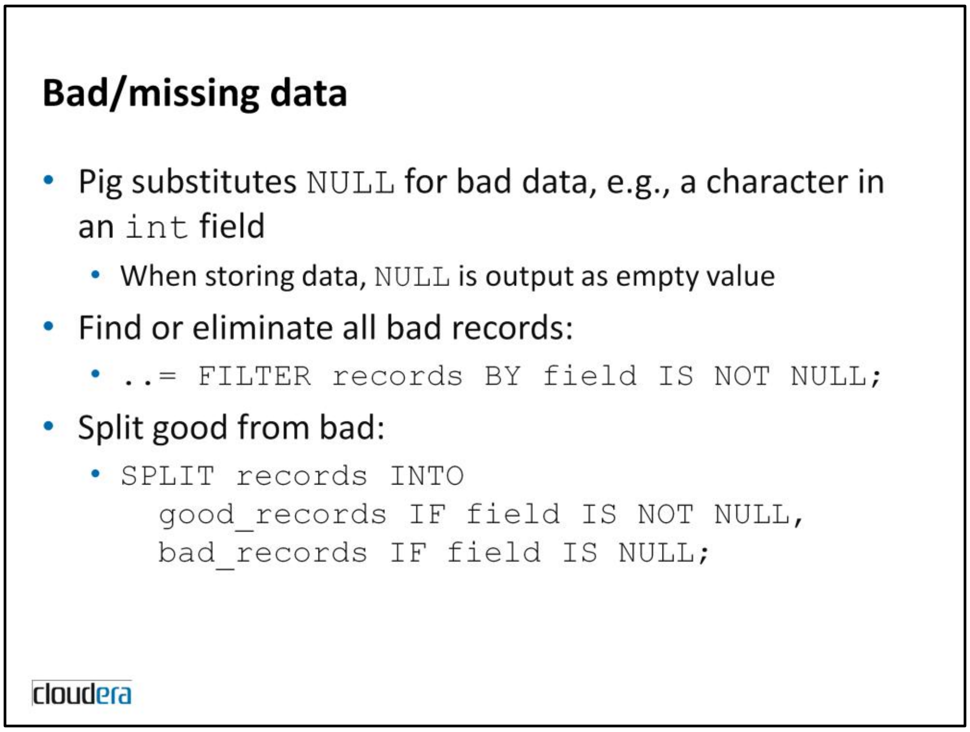
Running Jobs

Jobid	Priority	User	Name	Map % Complete
job_201010011039_0060	NORMAL	training	PigLatin:my job	0.00%

By default the Pig jobs are assigned a default job name, such as "Job380678878696361864.jar". On a shared Hadoop cluster, finding your job can be difficult. You can assign a meaningful job name using the property `job.name`:

```
grunt> set job.name 'my job'
```

When the job is executed (e.g., a `STORE` operation is submitted), this custom job name will be visible in the JobTracker Web interface.



Bad/missing data

- Pig substitutes NULL for bad data, e.g., a character in an int field
  - When storing data, NULL is output as empty value
- Find or eliminate all bad records:
  - `..= FILTER records BY field IS NOT NULL;`
- Split good from bad:

  -
    ```
    SPLIT records INTO
      good_records IF field IS NOT NULL,
      bad_records IF field IS NULL;
    ```

cloudera

If there is missing data or data that cannot be cast into the specified data type, Pig will return NULL. When storing data, NULL is output as an empty value.

It is possible to test for NULL (not, do not use '== NULL'). There are operators for testing against NULL.

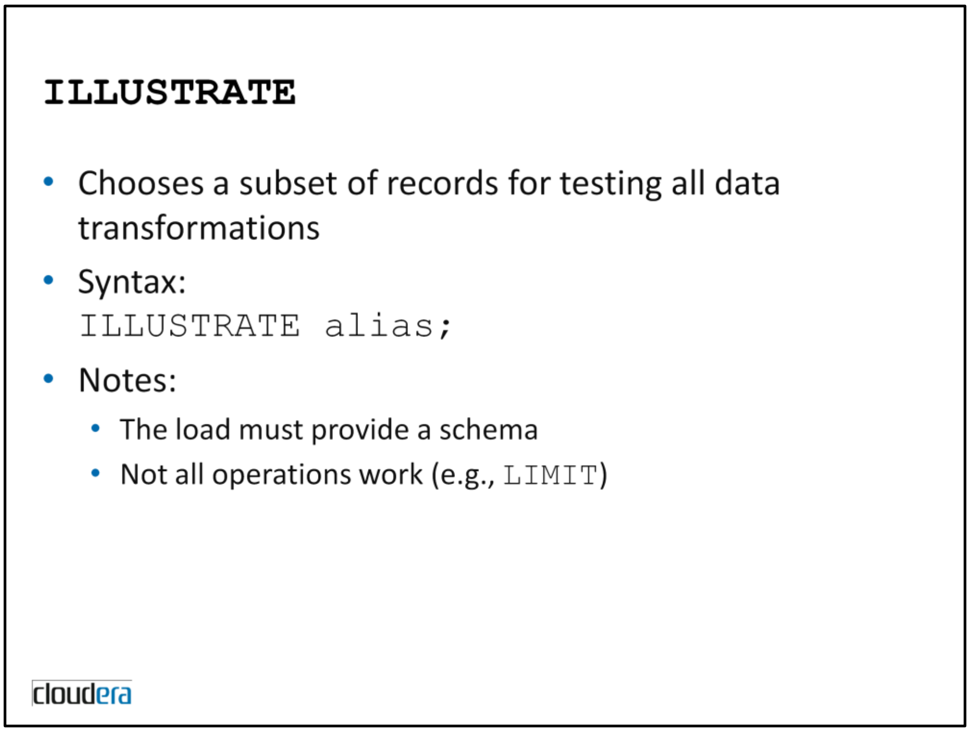
Finding all records that do not have bad values:

```
alias = FILTER records BY field IS NOT NULL;
```

Another useful trick is splitting a relation into the set of "good" records and "bad" records:

```
SPLIT records INTO
  good_records IF field IS NOT NULL,
  bad_records IF field IS NULL;
```

Note, if a schema was not specified during the LOAD, then bad values may not be detected since no casting is performed.

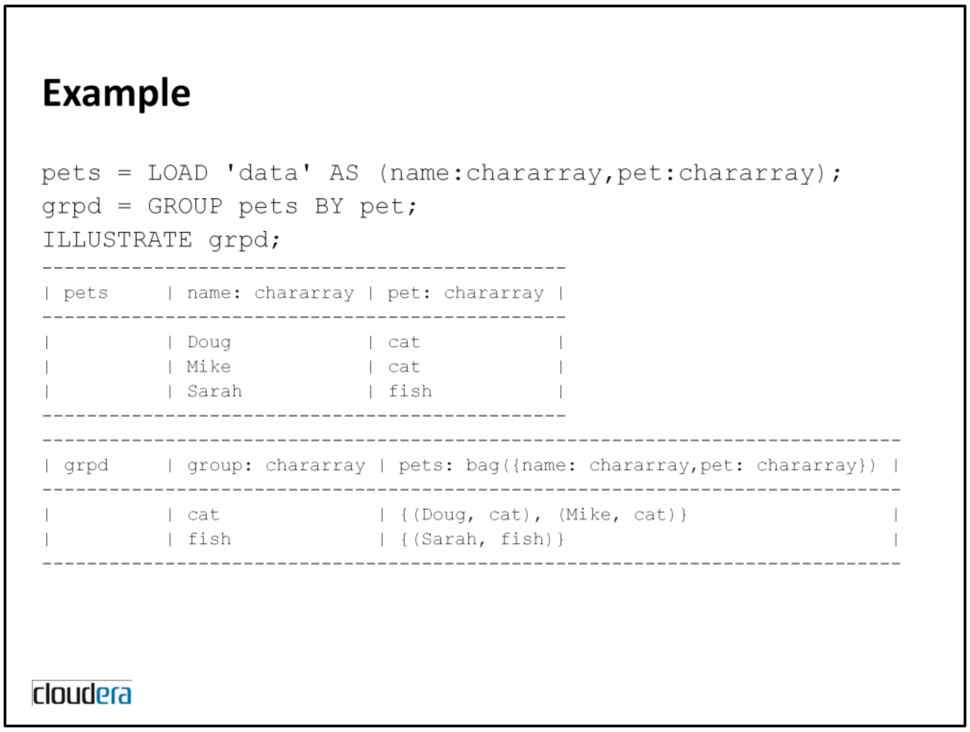


ILLUSTRATE

- Chooses a subset of records for testing all data transformations
- Syntax:
  `ILLUSTRATE alias;`
- Notes:
  - The load must provide a schema
  - Not all operations work (e.g., `LIMIT`)

`ILLUSTRATE` may be used to test a set of Pig Latin commands without using the entire data set. It is able to choose a representative set of rows that would test each part of the script. For example, if the FILTER operator is used to look for all users with occupation "scientist", then the chosen data set will include at least 1 record that is a scientist and at least 1 that is not.

There are a few limitations to `ILLUSTRATE`, specifically:
• The data load statement must provide a schema
• Not all operations are currently supported, such as `LIMIT`



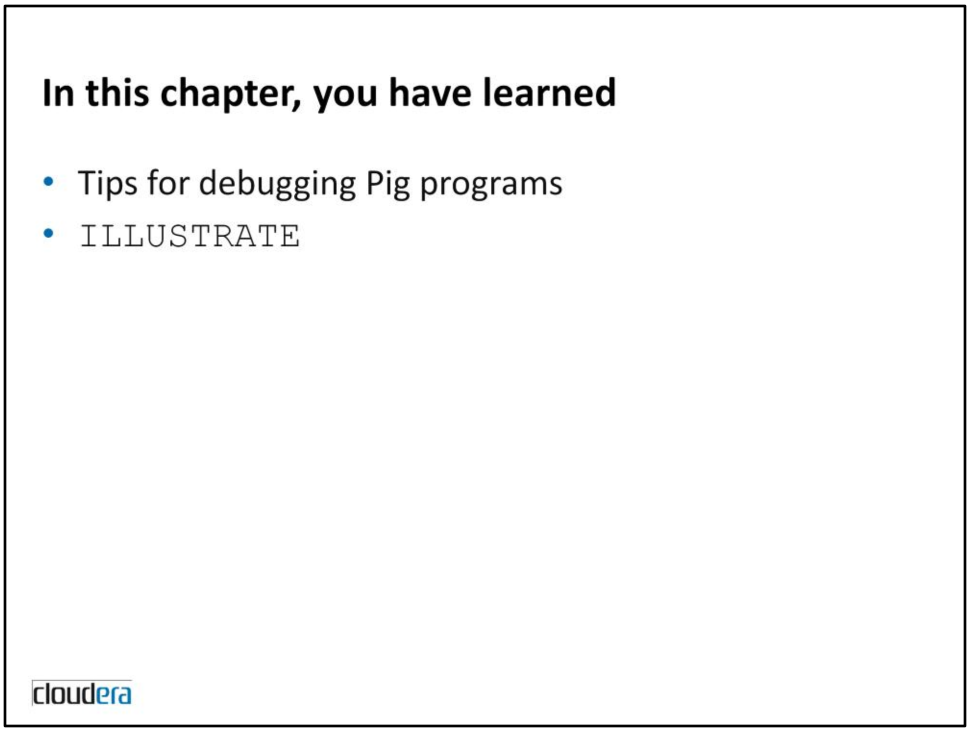
Example

```
pets = LOAD 'data' AS (name:chararray,pet:chararray);
grpd = GROUP pets BY pet;
ILLUSTRATE grpd;
---------------------------------------------
| pets     | name: chararray | pet: chararray |
---------------------------------------------
|          | Doug            | cat            |
|          | Mike            | cat            |
|          | Sarah           | fish           |
---------------------------------------------
------------------------------------------------------------------------------
| grpd     | group: chararray | pets: bag({name: chararray,pet: chararray}) |
------------------------------------------------------------------------------
|          | cat              | {(Doug, cat), (Mike, cat)}                  |
|          | fish             | {(Sarah, fish)}                             |
------------------------------------------------------------------------------
```

cloudera

Example:

```
pets = LOAD 'pets.txt' AS (name:chararray,pet:chararray);
grpd = GROUP pets BY pet;
counted = FOREACH grpd GENERATE group, COUNT(pets) AS num:long;
ILLUSTRATE counted;
-----------------------------------------------
| pets     | name: bytearray | pet: bytearray |
-----------------------------------------------
|          | Doug            | cat            |
|          | Mike            | cat            |
|          | Sarah           | fish           |
-----------------------------------------------
-----------------------------------------------
| pets     | name: chararray | pet: chararray |
-----------------------------------------------
|          | Doug            | cat            |
|          | Mike            | cat            |
|          | Sarah           | fish           |
-----------------------------------------------
------------------------------------------------------------------------------
| grpd     | group: chararray | pets: bag({name: chararray,pet: chararray}) |
------------------------------------------------------------------------------
|          | cat              | {(Doug, cat), (Mike, cat)}                  |
|          | fish             | {(Sarah, fish)}                             |
------------------------------------------------------------------------------
----------------------------------------------
| counted     | group: chararray | num: long |
----------------------------------------------
|             | cat              | 2         |
|             | fish             | 1         |
----------------------------------------------
```



In this chapter, you have learned

- Tips for debugging Pig programs
- `ILLUSTRATE`

cloudera