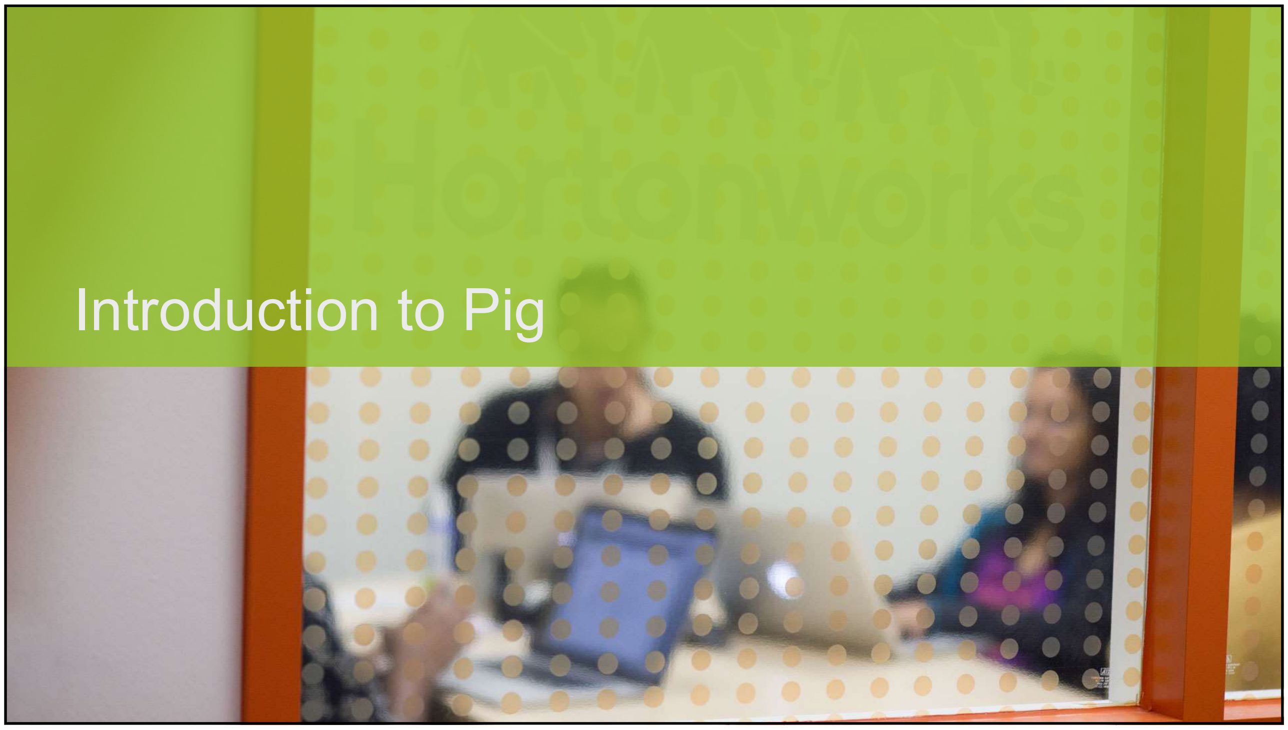
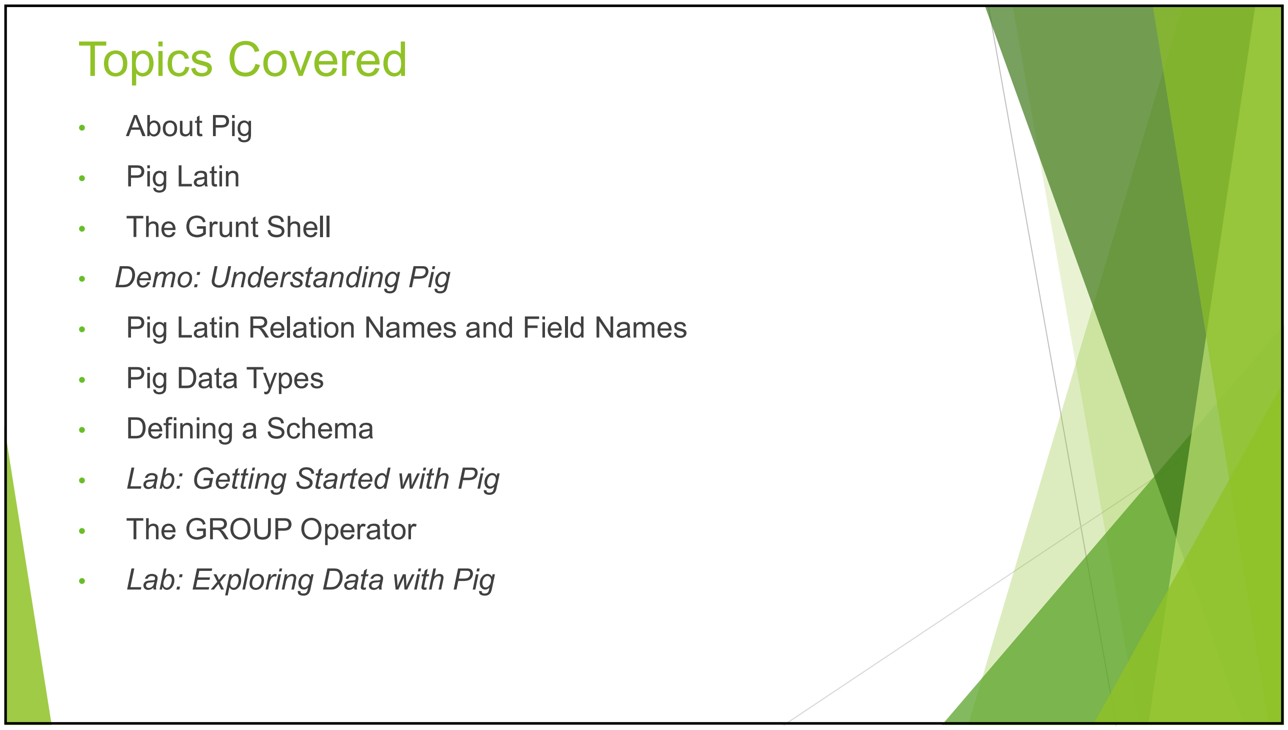
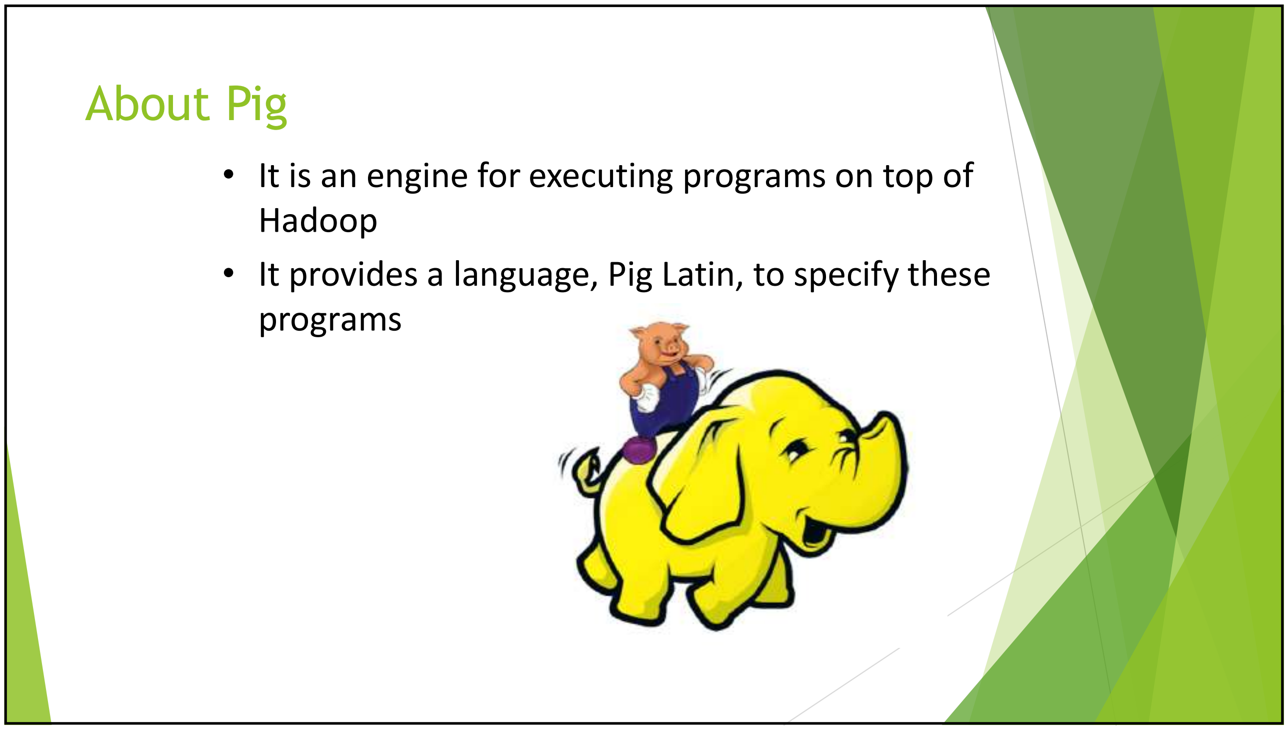


Introduction to Pig



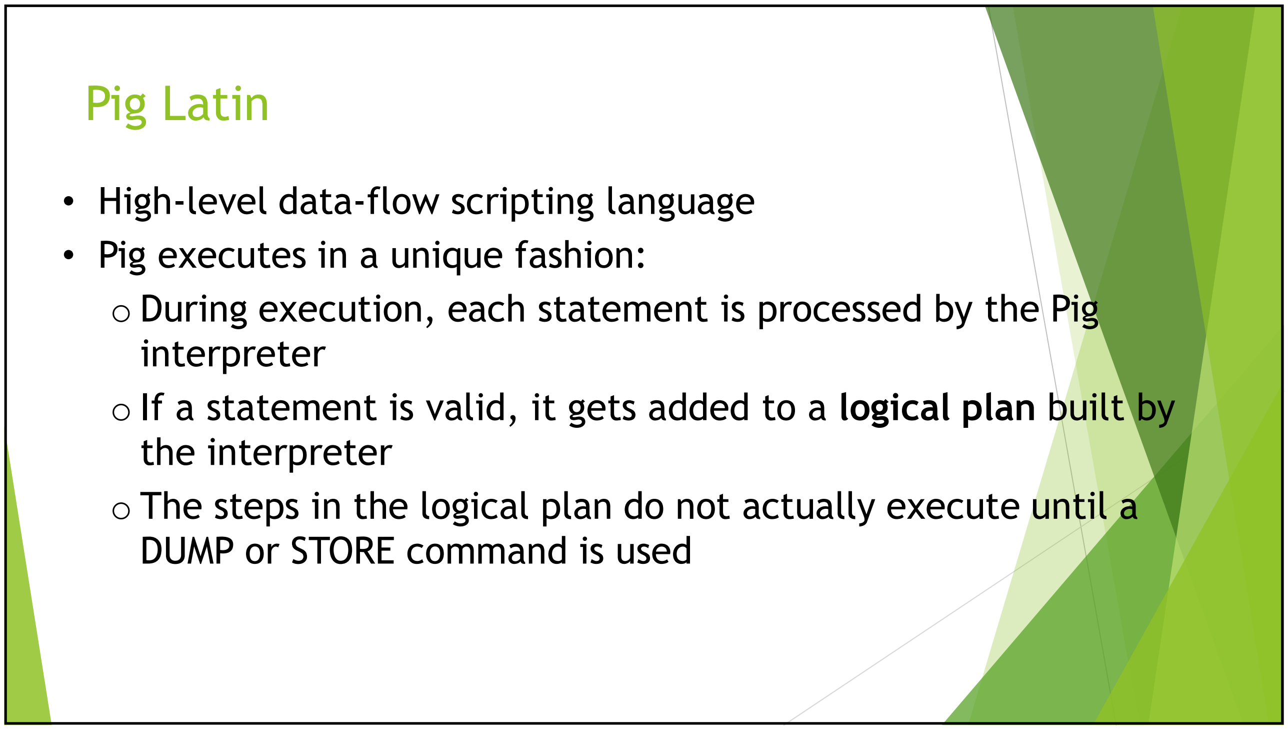
Topics Covered

- About Pig
- Pig Latin
- The Grunt Shell
- *Demo: Understanding Pig*
- Pig Latin Relation Names and Field Names
- Pig Data Types
- Defining a Schema
- *Lab: Getting Started with Pig*
- The GROUP Operator
- *Lab: Exploring Data with Pig*



About Pig

- It is an engine for executing programs on top of Hadoop
- It provides a language, Pig Latin, to specify these programs

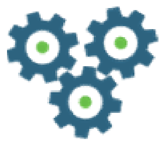


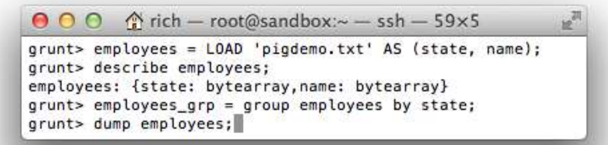
Pig Latin

- High-level data-flow scripting language
- Pig executes in a unique fashion:
  - During execution, each statement is processed by the Pig interpreter
  - If a statement is valid, it gets added to a **logical plan** built by the interpreter
  - The steps in the logical plan do not actually execute until a DUMP or STORE command is used

The Grunt Shell

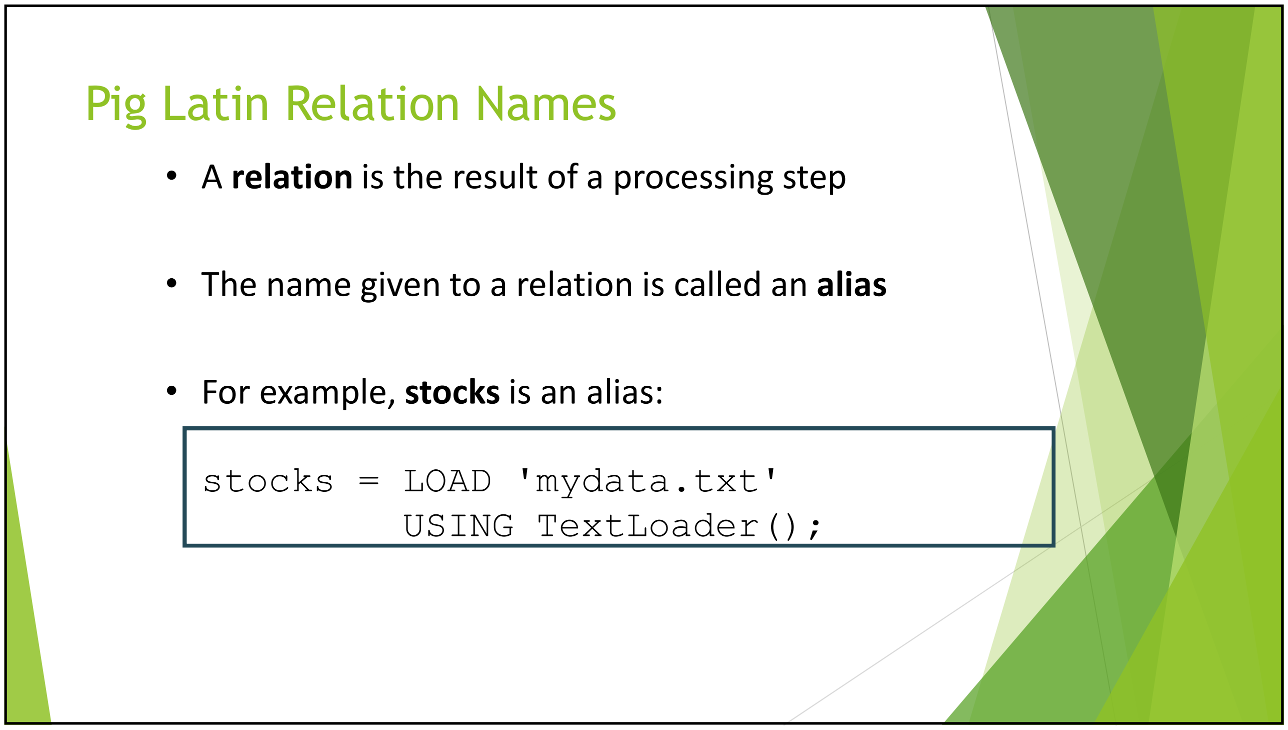
- Is an interactive shell for entering Pig Latin statements
- Is started by running the **pig** executable


Grunt shell



```
grunt> employees = LOAD 'pigdemo.txt' AS (state, name);
grunt> describe employees;
employees: {state: bytearray,name: bytearray}
grunt> employees_grp = group employees by state;
grunt> dump employees;
```

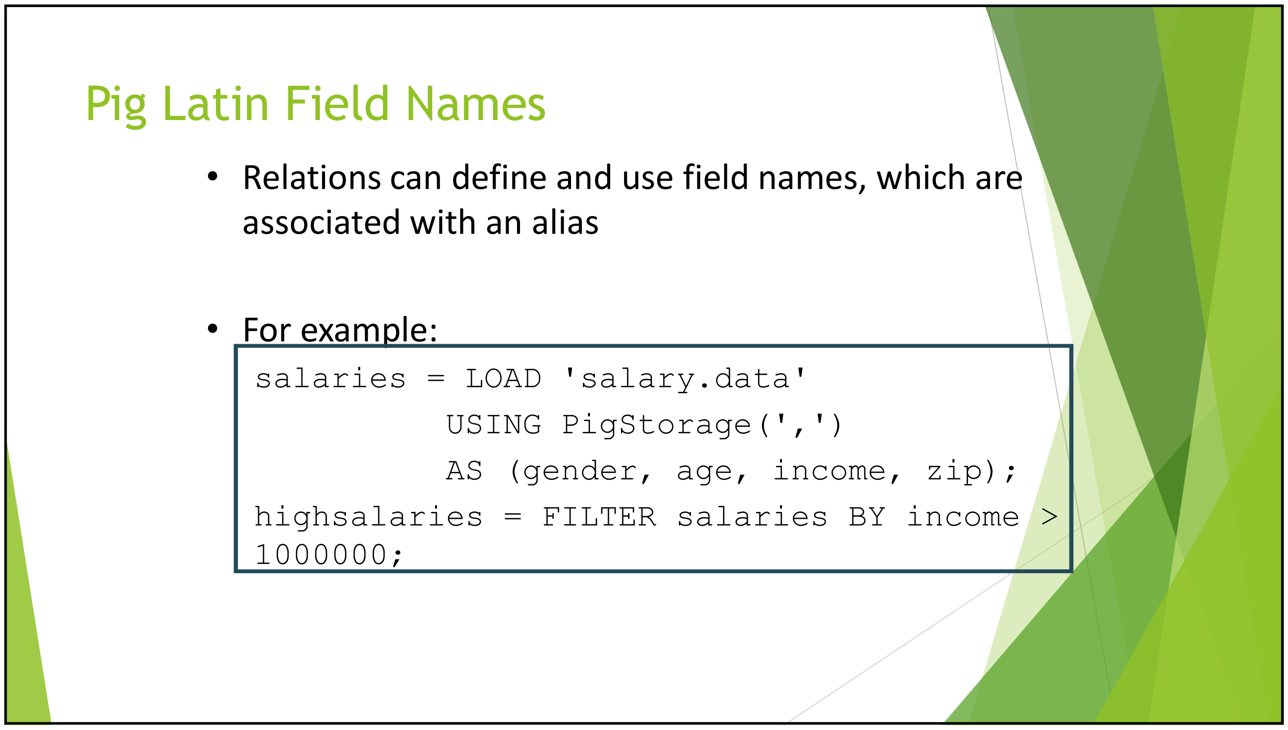
Demo: Understanding Pig



Pig Latin Relation Names

- A **relation** is the result of a processing step
- The name given to a relation is called an **alias**
- For example, **stocks** is an alias:

```
stocks = LOAD 'mydata.txt'
         USING TextLoader();
```

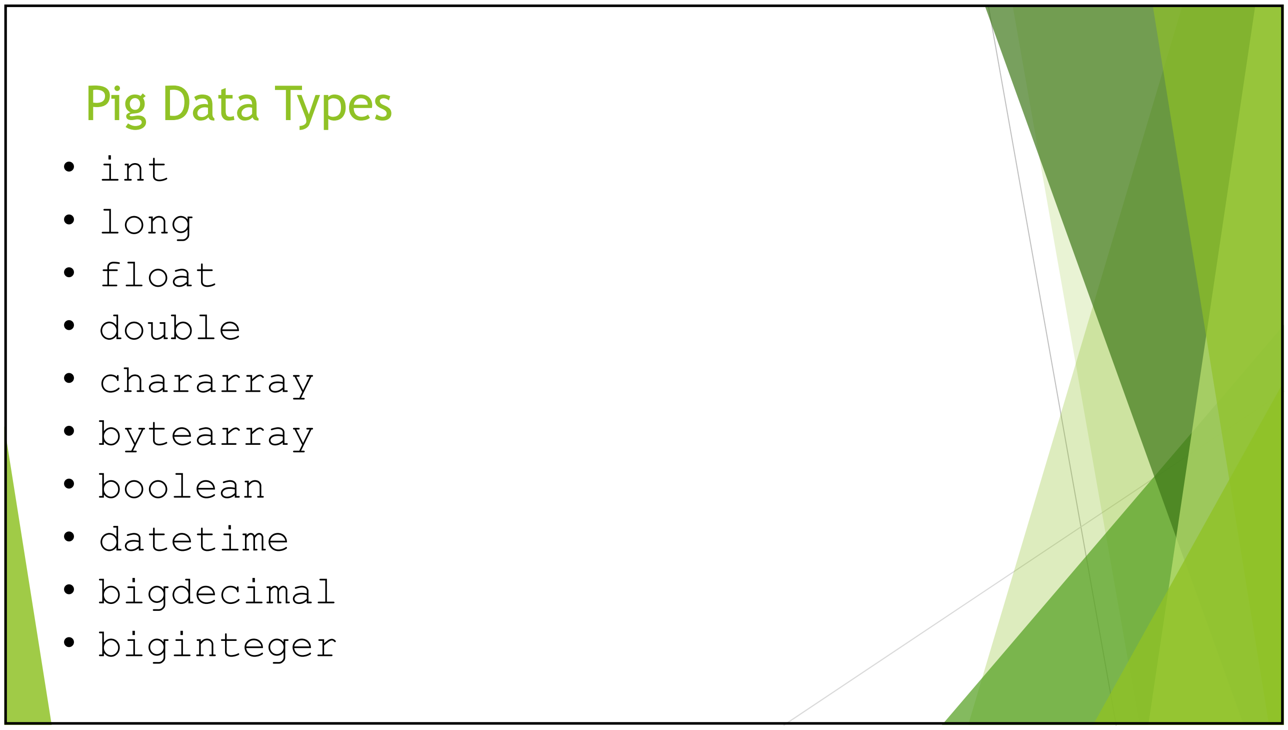


Pig Latin Field Names

- Relations can define and use field names, which are associated with an alias

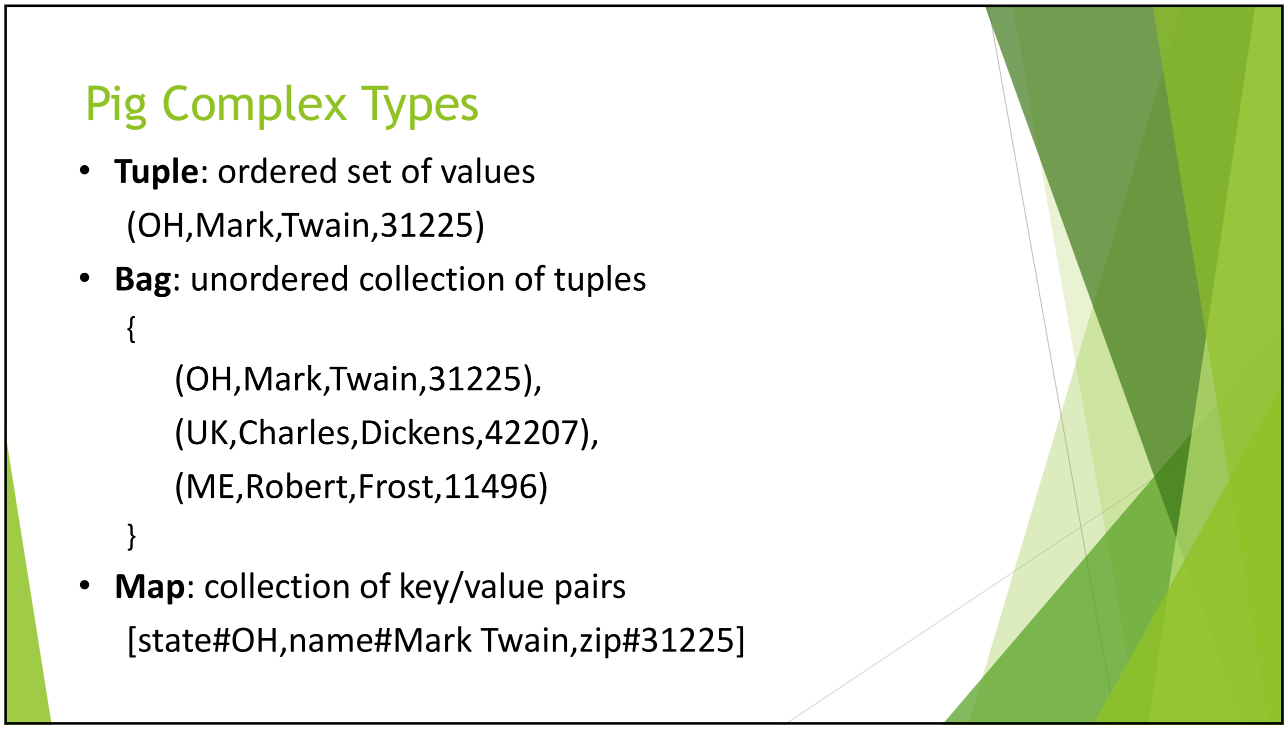
- For example:

```
salaries = LOAD 'salary.data'
          USING PigStorage(',')
          AS (gender, age, income, zip);
highsalaries = FILTER salaries BY income >
1000000;
```



Pig Data Types

- `int`
- `long`
- `float`
- `double`
- `chararray`
- `bytearray`
- `boolean`
- `datetime`
- `bigdecimal`
- `biginteger`



Pig Complex Types

- **Tuple**: ordered set of values

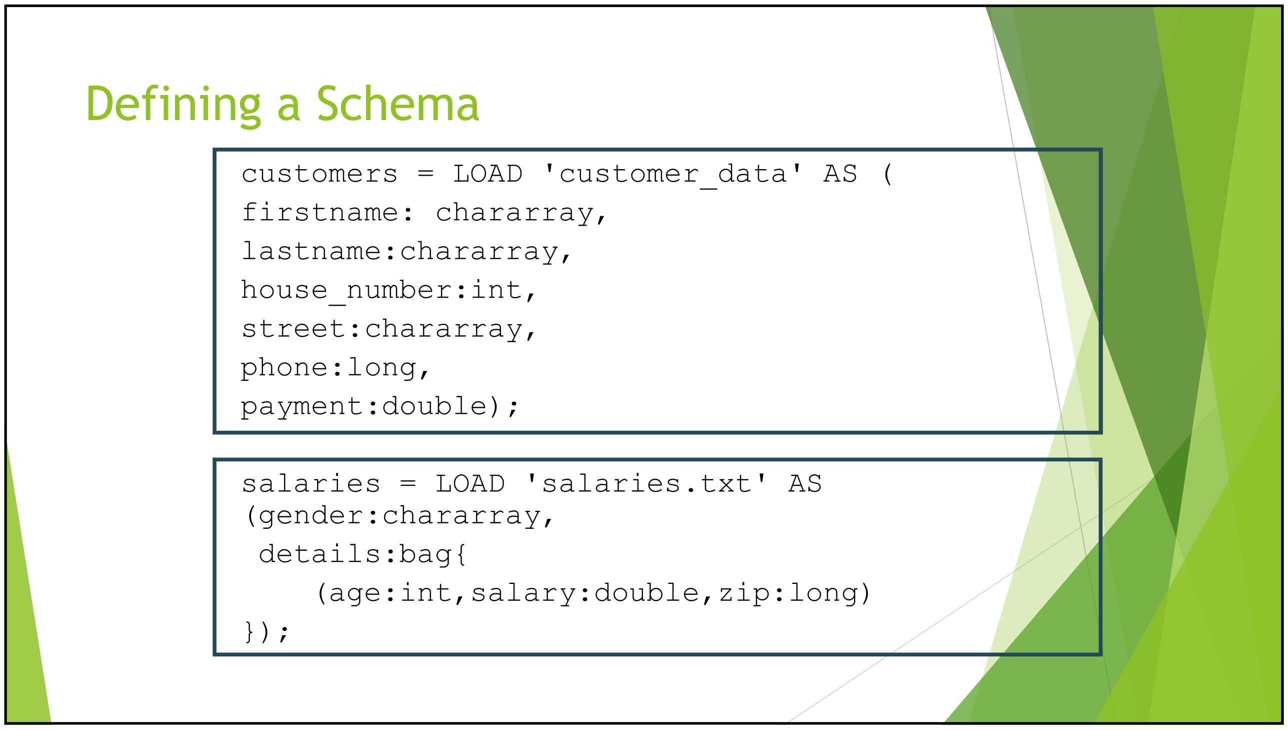
  (OH,Mark,Twain,31225)

- **Bag**: unordered collection of tuples

```
{
    (OH,Mark,Twain,31225),
    (UK,Charles,Dickens,42207),
    (ME,Robert,Frost,11496)
}
```

- **Map**: collection of key/value pairs

  [state#OH,name#Mark Twain,zip#31225]

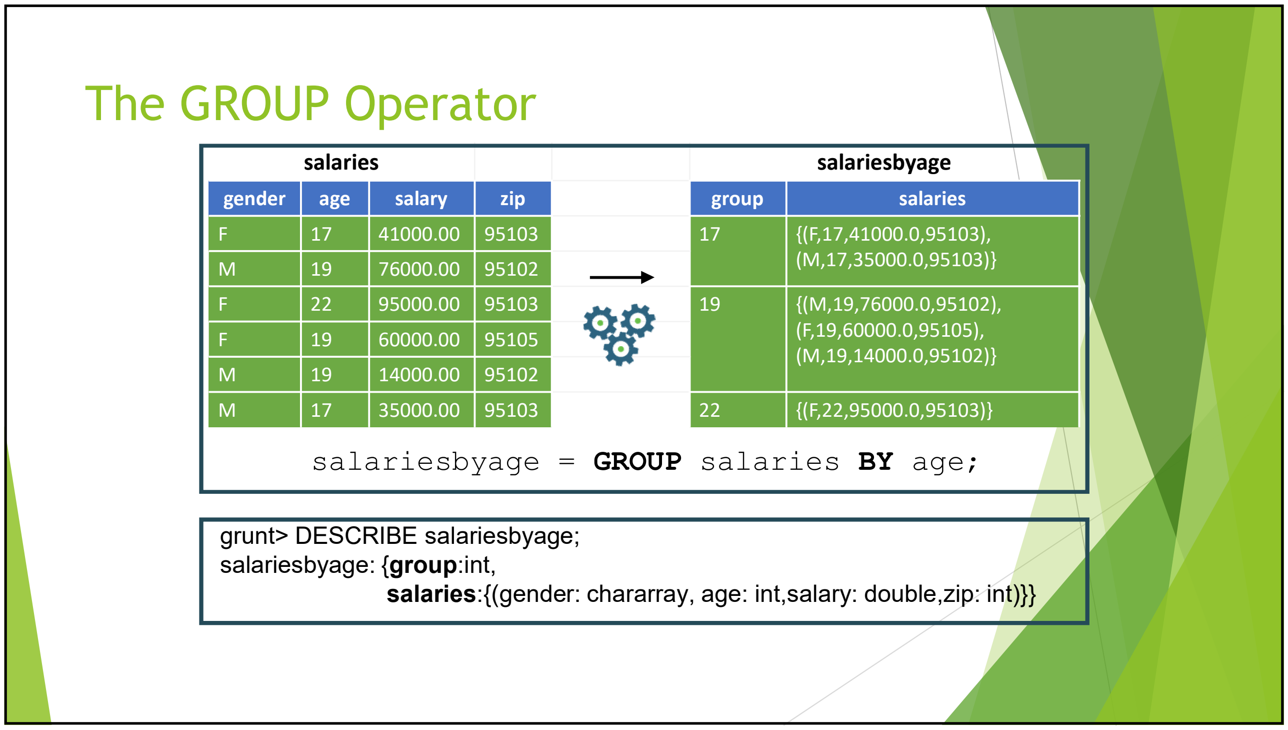


Defining a Schema

```
customers = LOAD 'customer_data' AS (
firstname: chararray,
lastname:chararray,
house_number:int,
street:chararray,
phone:long,
payment:double);
```

```
salaries = LOAD 'salaries.txt' AS
(gender:chararray,
 details:bag{
    (age:int,salary:double,zip:long)
});
```

Lab: Getting Started with Pig



The GROUP Operator

salaries

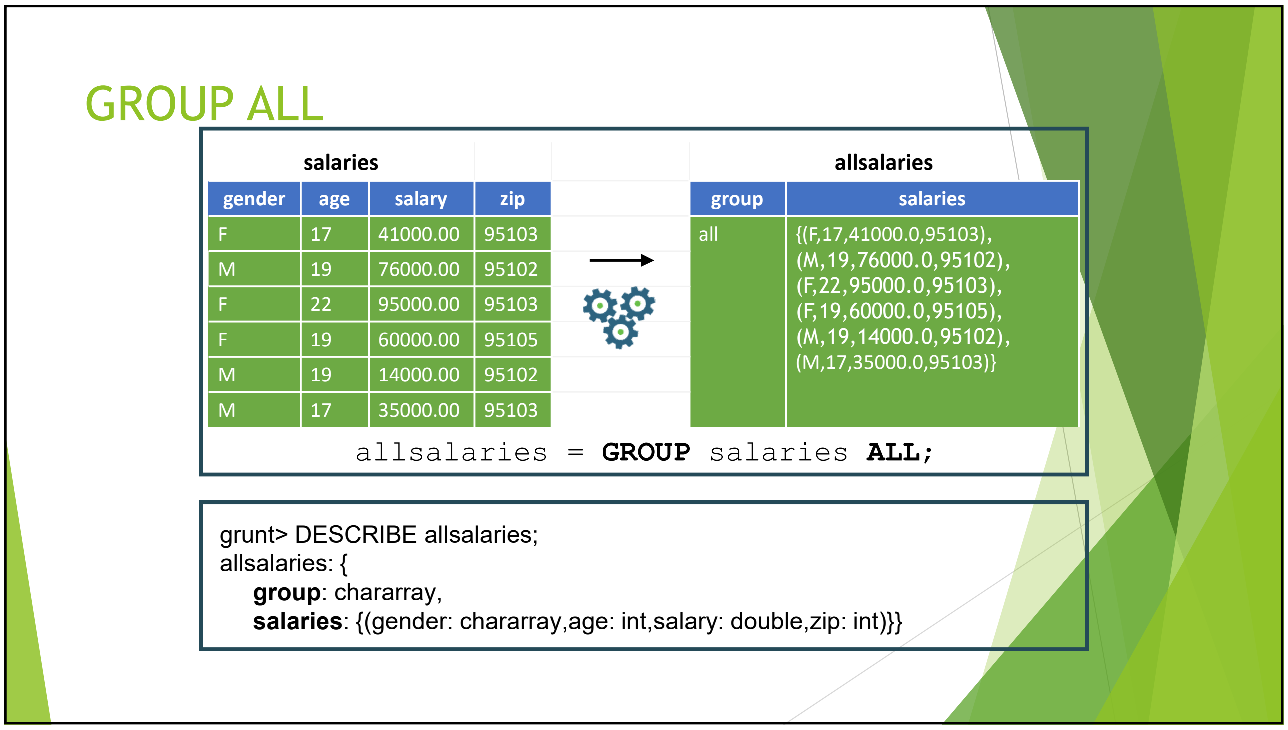
gender	age	salary	zip
F	17	41000.00	95103
M	19	76000.00	95102
F	22	95000.00	95103
F	19	60000.00	95105
M	19	14000.00	95102
M	17	35000.00	95103

salariesbyage

group	salaries
17	{(F,17,41000.0,95103), (M,17,35000.0,95103)}
19	{(M,19,76000.0,95102), (F,19,60000.0,95105), (M,19,14000.0,95102)}
22	{(F,22,95000.0,95103)}

```
salariesbyage = GROUP salaries BY age;
```

```
grunt> DESCRIBE salariesbyage;
salariesbyage: {group:int,
                salaries:{(gender: chararray, age: int,salary: double,zip: int)}}
```



GROUP ALL

salaries

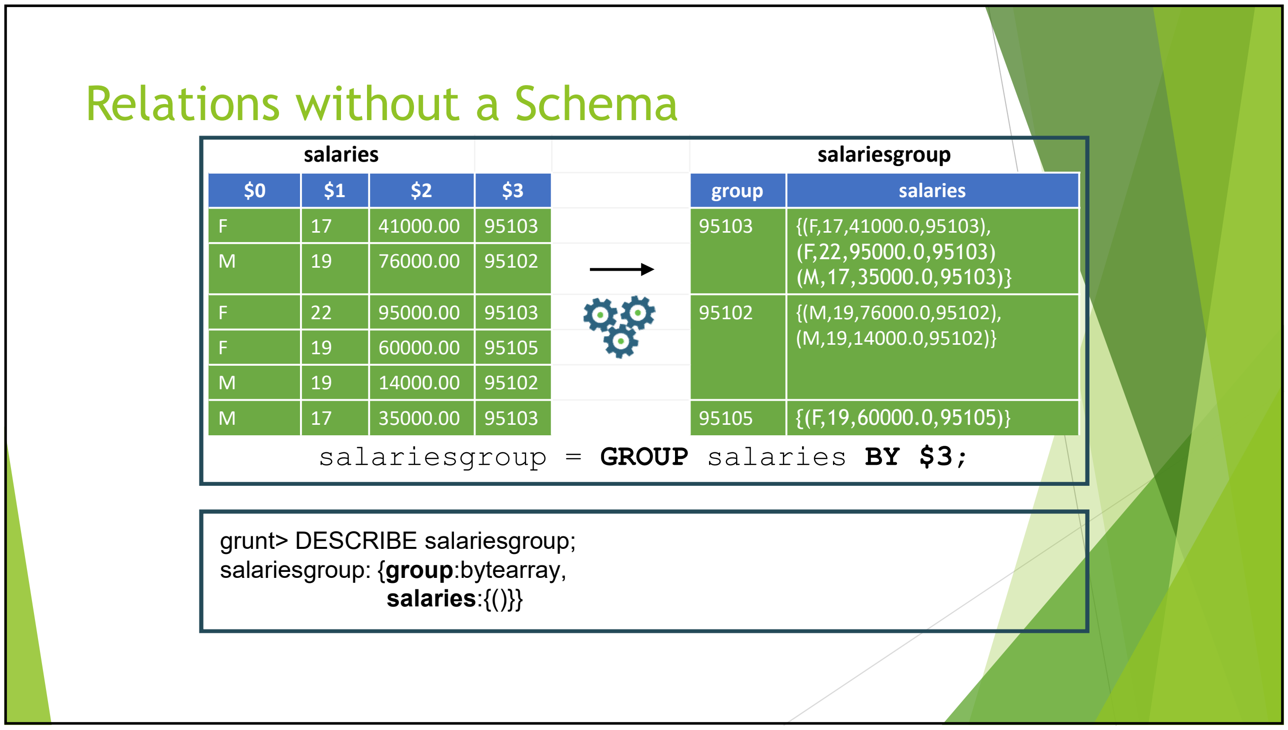
gender	age	salary	zip
F	17	41000.00	95103
M	19	76000.00	95102
F	22	95000.00	95103
F	19	60000.00	95105
M	19	14000.00	95102
M	17	35000.00	95103

allsalaries

group	salaries
all	{(F,17,41000.0,95103), (M,19,76000.0,95102), (F,22,95000.0,95103), (F,19,60000.0,95105), (M,19,14000.0,95102), (M,17,35000.0,95103)}

```
allsalaries = GROUP salaries ALL;
```

```
grunt> DESCRIBE allsalaries;
allsalaries: {
   group: chararray,
   salaries: {(gender: chararray,age: int,salary: double,zip: int)}}
```



Relations without a Schema

salaries

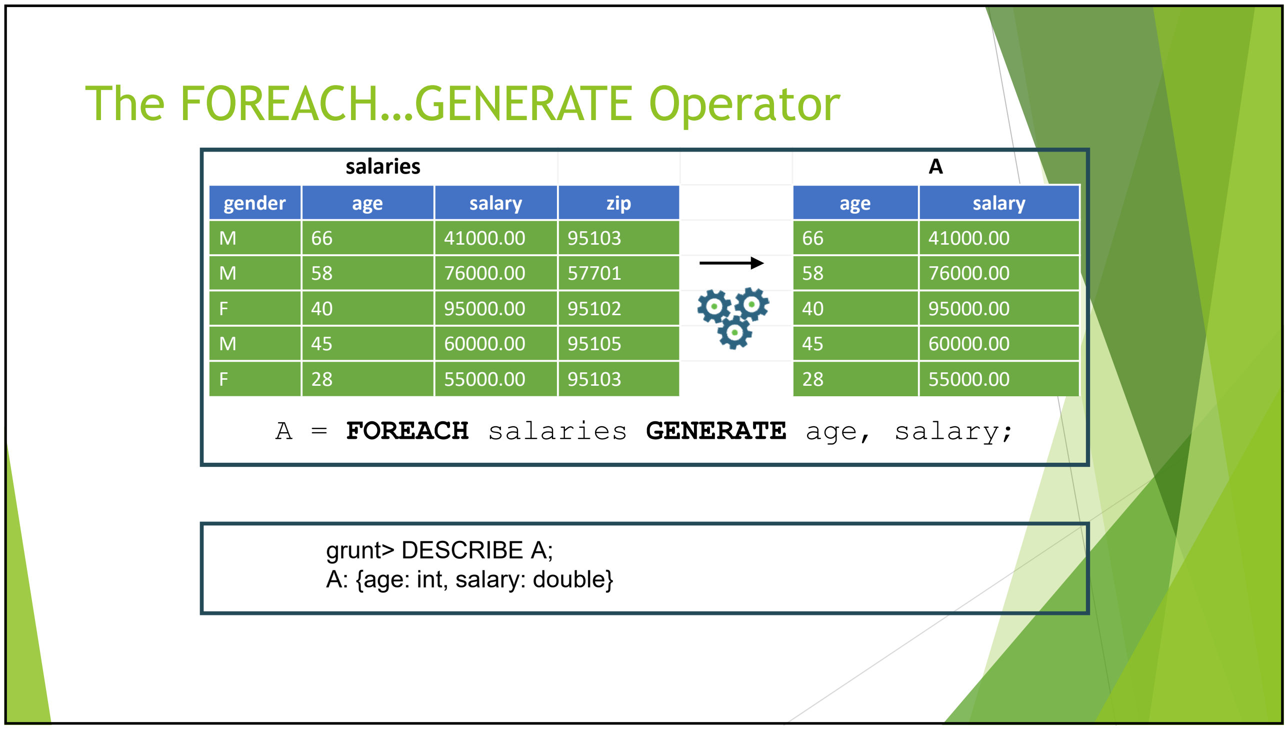
$0	$1	$2	$3
F	17	41000.00	95103
M	19	76000.00	95102
F	22	95000.00	95103
F	19	60000.00	95105
M	19	14000.00	95102
M	17	35000.00	95103

salariesgroup

group	salaries
95103	{(F,17,41000.0,95103), (F,22,95000.0,95103) (M,17,35000.0,95103)}
95102	{(M,19,76000.0,95102), (M,19,14000.0,95102)}
95105	{(F,19,60000.0,95105)}

```
salariesgroup = GROUP salaries BY $3;
```

```
grunt> DESCRIBE salariesgroup;
salariesgroup: {group:bytearray,
                salaries:{()}}
```



The FOREACH…GENERATE Operator

salaries

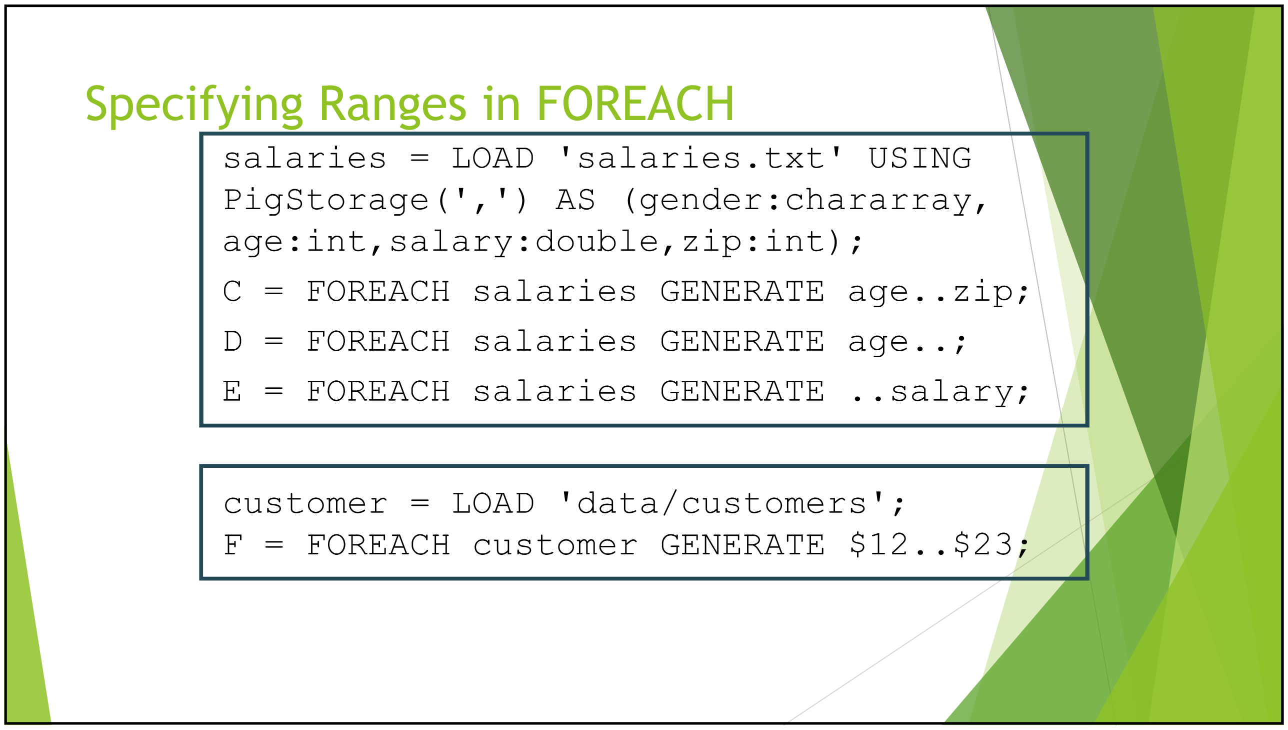
gender	age	salary	zip
M	66	41000.00	95103
M	58	76000.00	57701
F	40	95000.00	95102
M	45	60000.00	95105
F	28	55000.00	95103

A

age	salary
66	41000.00
58	76000.00
40	95000.00
45	60000.00
28	55000.00

```
A = FOREACH salaries GENERATE age, salary;
```

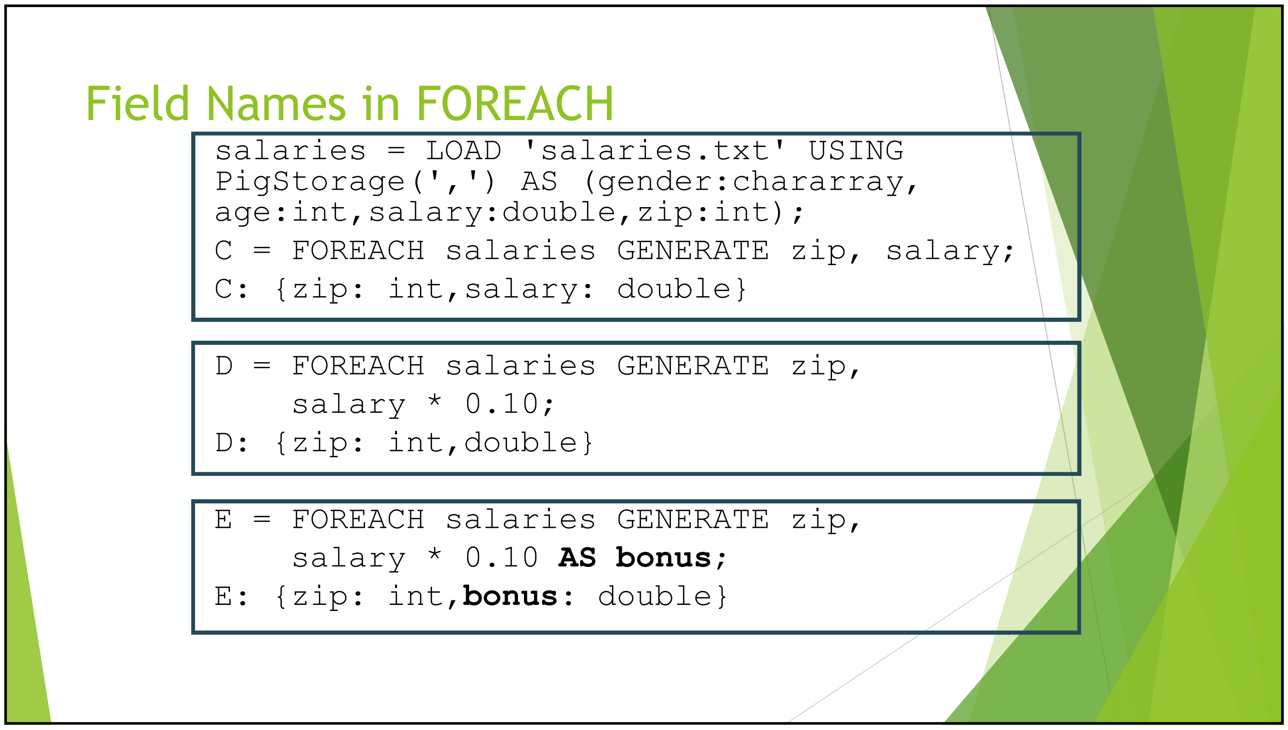
```
grunt> DESCRIBE A;
A: {age: int, salary: double}
```



Specifying Ranges in FOREACH

```
salaries = LOAD 'salaries.txt' USING
PigStorage(',') AS (gender:chararray,
age:int,salary:double,zip:int);
C = FOREACH salaries GENERATE age..zip;
D = FOREACH salaries GENERATE age..;
E = FOREACH salaries GENERATE ..salary;
```

```
customer = LOAD 'data/customers';
F = FOREACH customer GENERATE $12..$23;
```

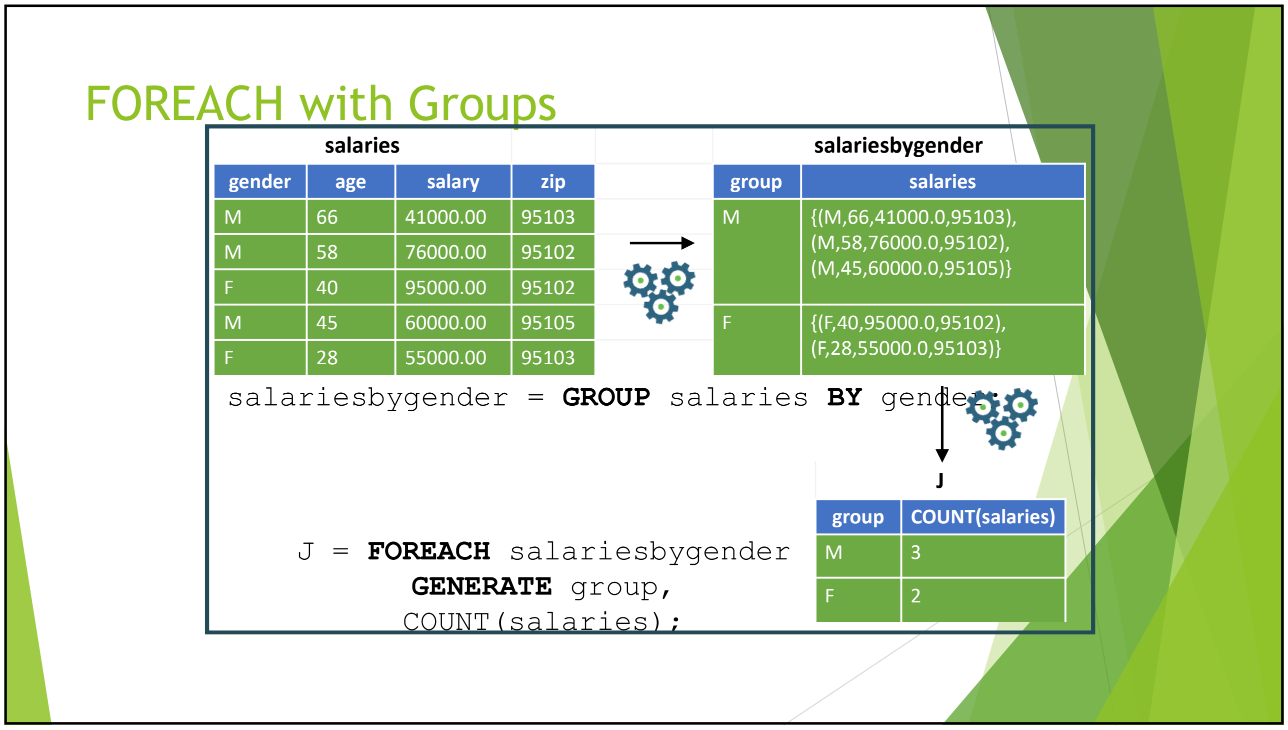


Field Names in FOREACH

```
salaries = LOAD 'salaries.txt' USING
PigStorage(',') AS (gender:chararray,
age:int,salary:double,zip:int);
C = FOREACH salaries GENERATE zip, salary;
C: {zip: int,salary: double}
```

```
D = FOREACH salaries GENERATE zip,
    salary * 0.10;
D: {zip: int,double}
```

```
E = FOREACH salaries GENERATE zip,
    salary * 0.10 AS bonus;
E: {zip: int,bonus: double}
```



FOREACH with Groups

salaries

gender	age	salary	zip
M	66	41000.00	95103
M	58	76000.00	95102
F	40	95000.00	95102
M	45	60000.00	95105
F	28	55000.00	95103

salariesbygender

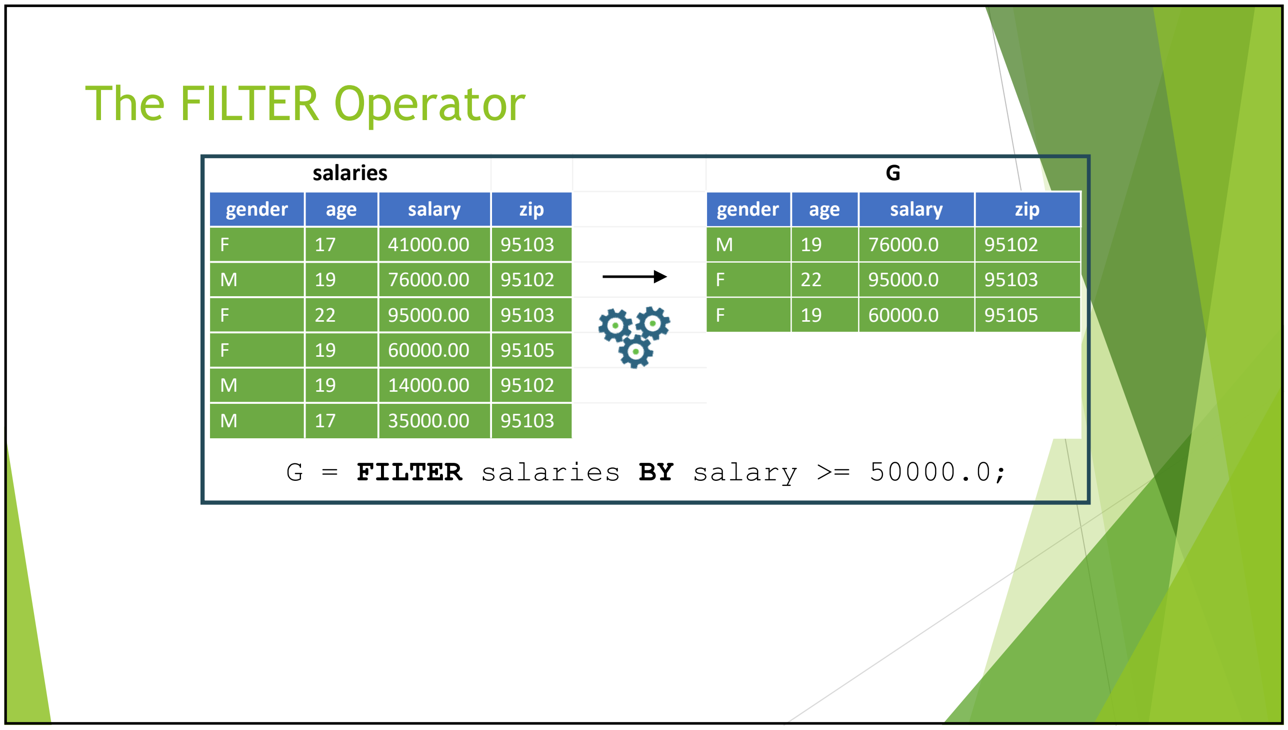
group	salaries
M	{(M,66,41000.0,95103), (M,58,76000.0,95102), (M,45,60000.0,95105)}
F	{(F,40,95000.0,95102), (F,28,55000.0,95103)}

```
salariesbygender = GROUP salaries BY gender;
```

J

group	COUNT(salaries)
M	3
F	2

```
J = FOREACH salariesbygender
    GENERATE group,
    COUNT(salaries);
```



The FILTER Operator

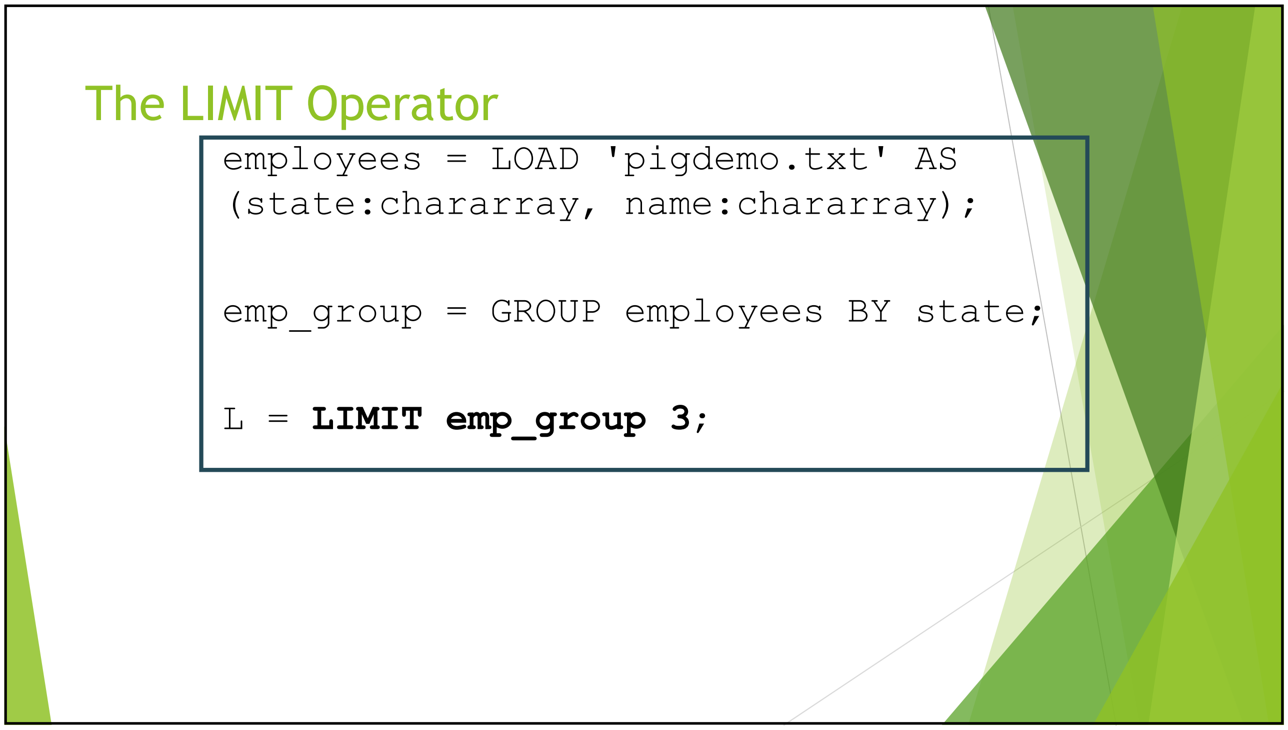
salaries

gender	age	salary	zip
F	17	41000.00	95103
M	19	76000.00	95102
F	22	95000.00	95103
F	19	60000.00	95105
M	19	14000.00	95102
M	17	35000.00	95103

G

gender	age	salary	zip
M	19	76000.0	95102
F	22	95000.0	95103
F	19	60000.0	95105

```
G = FILTER salaries BY salary >= 50000.0;
```

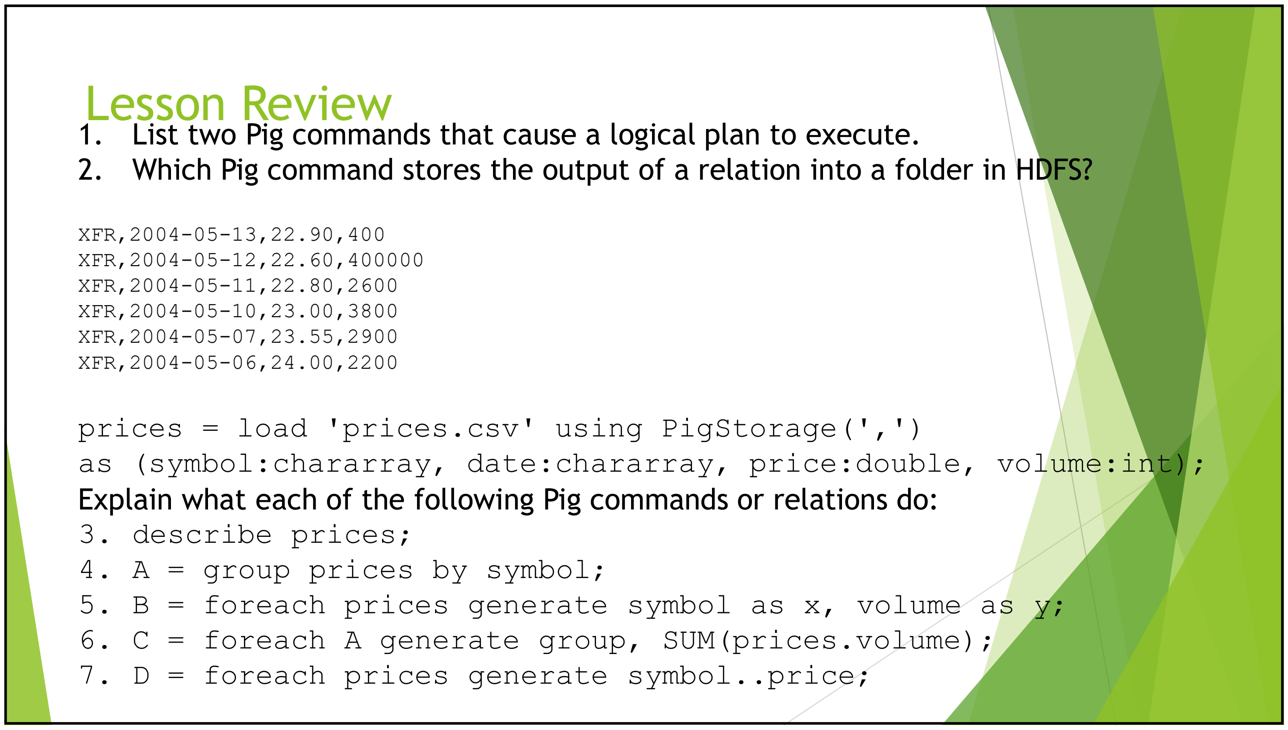


The LIMIT Operator

```
employees = LOAD 'pigdemo.txt' AS
(state:chararray, name:chararray);


emp_group = GROUP employees BY state;


L = LIMIT emp_group 3;
```



Lesson Review

1. List two Pig commands that cause a logical plan to execute.
2. Which Pig command stores the output of a relation into a folder in HDFS?

```
XFR,2004-05-13,22.90,400
XFR,2004-05-12,22.60,400000
XFR,2004-05-11,22.80,2600
XFR,2004-05-10,23.00,3800
XFR,2004-05-07,23.55,2900
XFR,2004-05-06,24.00,2200
```

```
prices = load 'prices.csv' using PigStorage(',')
as (symbol:chararray, date:chararray, price:double, volume:int);
```

Explain what each of the following Pig commands or relations do:

```
3. describe prices;
4. A = group prices by symbol;
5. B = foreach prices generate symbol as x, volume as y;
6. C = foreach A generate group, SUM(prices.volume);
7. D = foreach prices generate symbol..price;
```

Lab: Exploring Data with Pig