1. Selecting Specific fields from the relation

grunt> orders = load 'order_data' using PigStorage(',') as(Order_ID:chararray, Customer_ID:int,Order_Quantity:int,Order_Value:float,Order_Date:chararray,Product_ID:int, Product_Name:chararray);

grunt>orders_customers = foreach orders generateOrder_ID,Customer_ID;

grunt>dump orders_customers

IF WE HAVEN'T SPECIFIED A SCHEMAFOR THE ORDERS TABLE WE WOULD USETHE $ NOTATION TO SELECT COLUMNS

grunt>orders_customers = foreach ordersgenerate $0,$1;

THE TUPLE WAS LOADED USING:

order_customer = load 'order_customer' as(order_customer_id: tuple(Order_ID:chararray,Customer_ID:int));

FIELDS IN A TUPLE CAN BE ACCESSED USING THE NAME OF THE FIELD AND '.' OPERATOR

grunt>orders_only=foreachorder_customergenerate order_customer_id.Order_ID;

LET US ASSUME WE LOAD DATA FROM A FILE CALLED MAP_EXAMPLE.TXT

map_example = load 'map_example.txt' as(Customer_Fields:MAP[chararray]);

grunt>customer_name = foreachmap_example generate Customer_Fields#'Customer_Name';

order_bag = load 'order_bag' as
(order_customer_id: bag{tuple_name:(Order_ID: chararray,
Customer_ID: int,
Order_Quantity: int,
Order_Value: float,
Order_Date: chararray,
Product_ID: int,
Product_Name: chararray) });

```
grunt>orders_product_tuple = foreachorder_baggenerate
order_customer_id.(Order_ID,Product_Name);
```

2. APPLYING SPECIFIC FUNCTIONS TO A FIELD
FOREACH CAN APPLY EXPRESSIONS ORFORMULAS ALSO TO THE FIELDS
LET'S SAY WE WANT TO COMPUTE THE PRODUCT OFTWO FIELDS

```
grunt>quantity_value_product = foreach orders
generate(Order_Quantity * Order_Value)
asquantity_value_product,Order_ID,Customer_ID;
quantity_value_product = foreach ordersgenerate(Order_Quantity *
Order_Value) as
quantity_value_product,Order_ID,Customer_ID,ROUND(Order_Value) as
Trimmed_Value;

grunt>quantity_value_product = foreach orders
generate(Order_Quantity * Order_Value) as
quantity_value_product,Order_ID,Customer_ID,
SUBSTRING(Order_ID,2,3) as Order_Number;

grunt>order_customer_map = foreach orders
generate(TOMAP(Order_ID,Customer_ID));

grunt>order_bag = foreach orders generate
TOBAG(Order_ID,Customer_ID,Order_Quantity,Order_Value,Order_Date,odu
ct_ID,Product_Name) as Order_Customer_ID;

grunt>orders = load 'order_data.csv' using PigStorage(',') as
(Order_ID:chararray,
Customer_ID:int, Order_Quantity:int,Order_Value:float,Order_Date,
Product_ID:int,
Product_Name:chararray);

grunt>time_temp = foreach orders generate
Order_ID,Customer_ID,ToDate(Order_Date,'yyyy-mm-dd') as
Order_TimeStamp;

grunt>order_month_table = foreachtime_temp generate
Order_ID,Customer_ID,GetMonth(Order_TimeStamp);
grunt> dump order_month_table;
```

3. SELECTING A SPECIFIC NUMBER OF RECORDS, OR A SPECIFIC NUMBER OF DISTINCT VALUES

```
grunt>orders_temp = foreach orders generate Order_ID;
grunt>distinct_orders = distinct orders_temp;
grunt> dump distinct_orders;

grunt>orders_temp = limit orders 3;
grunt> dump orders_temp;
```

4. ORDERING RECORDS BASED ON SOME SPECIFIC COLUMN

```
grunt>orders_desc = order orders by Order_IDdesc;
grunt> dump orders_desc;
```

5. FILTERING RECORDS BASED ON A CONDITION

```
grunt>orders_quantity_filter = filter orders by
Order_Quantity>=2;
grunt> dump orders_quantity_filter;

grunt>orders_id_filter = filter orders by Order_ID matches ‘OD02';
grunt> dump orders_id_filter;
grunt>product_id_filter = filter orders by Product_Name matches
'Apple iPhone 4.*';
grunt> dump product_id_filter;
grunt>product_order_bag = filter order_bag by not
IsEmpty(order_customer_id);
grunt>map_example_filter = filter map_example by not
IsEmpty(Customer_Fields);

grunt>null_id_filter = filter orders by Order_ID IS NULL;

grunt>orders_id_quantity_filter = filter orders by Order_ID matches 'OD02'
and not (Order_Quantity == 1);
grunt> dump orders_id_quantity_filter;
```

6. GROUPING/AGGREGATING DATA BASED ON SPECIFIC COLUMNS

LET US SAY WE WANTTO CALCULATE TOTALORDER_QUANTITY PERORDER

```
grunt>groupd = group orders BY Order_ID;
```

```
grunt>aggr_by_order = foreachgroupd
generategroup,SUM(orders.Order_Quantity);
```

LET US SAY WE WANT TOSEE HOW MANY PRODUCTSOUR CUSTOMER BUY

```
grunt>groupd = group orders by Customer_ID;
grunt>aggr_by_cust = foreachgroupd generate
group,COUNT(orders.Product_ID) as Total_Products;
```

7. JOINING DATA OF ONE RELATION WITH ANOTHER RELATION

```
customers = load 'customer_data.csv' using PigStorage(',') as
(Customer_ID:int,First_Name:chararray,Second_Name:chararray,Contact
_No:Long,Created_Date:chararray,Email_ID:chararray);

orders = load 'order_data.csv' using PigStorage(',') as
(Order_ID:chararray,Customer_ID:int,Order_Quantity:int,Order_Value:
float,Order_Date:chararray, Product_ID:int,
Product_Name:chararray);

grunt> joined= join customers by Customer_ID,orders byCustomer_ID;

grunt> temp = foreach joined generate
customers::Customer_ID,customers::Contact_No,orders::Order_ID,orders::Prod
uct_ID;

grunt> temp = foreach joined generate
customers::Customer_ID,Contact_No,Order_ID,Product_ID;

grunt> joined= join customers by Customer_ID left outer,orders
byCustomer_ID;

grunt> joined= join customers by Customer_ID right outer,orders
byCustomer_ID;

grunt> joined= join customers by Customer_ID full outer,ordersby
Customer_ID;
grunt> orders1 = load 'order_data.csv' using PigStorage(',') as
(Order_ID:chararray,Customer_ID:int,Order_Quantity:int);
grunt> orders2 = load 'order_data.csv' using PigStorage(',') as
(Order_ID:chararray,Customer_ID:int,Order_Quantity:int);

grunt> joined= join orders1 by Customer_ID,orders2 byCustomer_ID;
```

grunt> crossed= cross customers,orders;

LET US SAY WE WANT TO FIND THE INVALIDORDERS IN ORDER DATAINVALID ORDERS ARE THOSE ORDERS WHERECUSTOMER'S ACCOUNT CREATED DATE ISGREATER THAN ORDER DATE

grunt> crossed= cross customers,orders;
grunt>invalid_orders = filter crossed by
customers::Created_Date>=orders::Order_Date;
grunt>invalid_orders_list = foreachinvalid_orders generate Order_ID;

UNION:
grunt> file1 = load '/user/navdeepsingh/order_data/order_data/
date-1.csv';
grunt> file2 = load '/user/navdeepsingh/order_data/order_data/
date-2.csv';
grunt>total_data = union file1,file2;

IN CASE NEW COLUMNS AREADDED -UNION ONSCHEMA

grunt> file1 = load '/user/navdeepsingh/sales/sales1' as
(Order_ID:Chararray,Customer_ID:int,Order_Quantity:Int,Order_Value:float);
grunt> file2 = load '/user/navdeepsingh/sales/sales2' as
(Order_ID:Chararray,Customer_ID:int,Order_Datetime:Chararray,Order_Value
:float);
grunt>total_data = union onschema file1,file2;
grunt> describe total_data;
total_data:
{Order_ID: bytearray,Customer_ID: int,Order_Quantity: int,Order_Value:
float,Order_Datetime: chararray}

8. TRANSFORMING NESTED DATA IN A COLUMN TO MULTIPLE ROWS

grunt>groupd = group orders by Customer_ID;
grunt>customer_product = foreach group generate group as
Customer_ID,orders.Product_Name as Products_Bought;

grunt>flattened_customer_product = foreachcustomer_product generate
Customer_ID, flatten(Products_Bought) as Products;
grunt> dump flattened_customer_product;

CO-GROUPIS A MORE GENERAL FORM OF GROUP

customers = load 'customer_data.csv' using PigStorage(',') as (Customer_ID:int,First_Name:chararray,Second_Name:chararray,Contact _No:Long,Created_Date:chararray,Email_ID:chararray);

orders = load 'order_data.csv' using PigStorage(',') as (Order_ID:chararray,Customer_ID:int,Order_Quantity:int,Order_Value: float,Order_Date:chararray, Product_ID:int, Product_Name:chararray);

cogrouped=cogroup customers by Customer_ID, orders by Customer_ID;

C0-GROUP HELPS EXECUTE SEMI-JOINS: A LEFT SEMI JOIN IS A JOIN THAT
RETURNS THE RECORDS ONLY FROM THELEFT-HAND TABLE.

LET US ASSUME CUSTOMERS IS THE LEFT TABLEAND ORDERS IS THE RIGHT TABLE. LEFT SEMI JOIN RETURNS THE CUSTOMERSDATA FOR ONLY THOSE CUSTOMERS WHOHAVE PLACED ANY ORDER.

grunt>cogroupd = cogroup customers by Customer_ID, orders by Customer_ID;
grunt>semijoin = filter cogroupd by not IsEmpty(orders);
grunt>semijoin_dump = foreachsemijoin generate flatten(customers);
grunt> dump semijoin_dump;

9. SAMPLING RECORDS

grunt> file1 = sample orders 0.27;

10. NESTED FOREACH

LET US ASSUME WE WANT TO CALCULATE TOTALNUMBER OF ITEMS BOUGHT BY CUSTOMERSAND FIND WHICH ITEMS WERE BOUGHT MOST OFTEN

grunt>groupd = group orders by (Customer_ID,Product_ID);

```
grunt> temp = foreachgroupd generate group.Customer_ID as Customer_ID,
group.Product_ID as Product_ID, SUM(orders.Order_Quantity) as
no_of_items;
grunt> groupd2 = group temp by Customer_ID;
grunt> result1 = foreach groupd2{
total_items = SUM(temp.no_of_items);
sorted_items = order temp by no_of_itemsdesc;
highest_item = limit sorted_items 1;
generate Flatten(highest_item),total_items;
};
```

ONE MORE EXAMPLE: COUNT DISTINCT NUMBER OF PRODUCTS PER PERSON

```
grunt>groupd = group orders by (Customer_ID,Product_ID);
grunt> temp = foreachgroupd generate group.Customer_ID as Customer_ID,
group.Product_ID as Product_ID, SUM(orders.Order_Quantity) as
no_of_items;
grunt> groupd2 = group temp by Customer_ID;
grunt>unique_order_customer = foreach groupd2{
total_products = temp.Product_ID;
unique_products = distinct total_products;
generate COUNT(unique_products) as unique_products,group;
};
```

‘**SPLIT**’ COMMAND:

IT IS USED TOEXPLICITLY SPLITDATA FLOWS

```
grunt> split customers into customer_group_1 if
Customer_ID<=25,
customer_group_2 if Customer_ID<=50 and
Customer_ID>25,
customer_group_3 if Customer_ID<=75 and
Customer_ID>50,
customer_group_4 if Customer_ID>75;
```

EXPLAIN: EXPLAINS THE PLANS OF A RELATION OR PIG SCRIPT

```
grunt> explain cogrouped;
```

ILLUSTRATE:ILLUSTRATE TAKES A SAMPLE OF THE DATA ANDRUNS IT THROUGH YOUR SCRIPT

```
grunt> illustrate result1;
```