

Course Notes

Links to Dataset

Course Notes and Project Brief:

Course Notes:

https://www.dropbox.com/s/sz86g4k41lt0qwt/Course_Notes.pdf?dl=0

Project Brief:

<https://www.dropbox.com/s/022qqqfcc98tmlb/Project%20Brief.pdf?dl=0>

Data:

orders.csv

<https://www.dropbox.com/s/7lqf6j7cp1gh1v7/orders.csv?dl=0>

product.csv

<https://www.dropbox.com/s/pdvp7qm5wze0brh/product.csv?dl=0>

customers.csv

<https://www.dropbox.com/s/ziqne9pkbp0l5lo/customers.csv?dl=0>

markdowncopy.txt

<https://www.dropbox.com/s/84yibkmmq8xxt09/markdowncopy.txt?dl=0>

Slides:

What is DBT

<https://www.dropbox.com/s/jgv6h4nezmx8ju/What%20is%20DBT.pdf?dl=0>

ELT vs ETL

<https://www.dropbox.com/s/fjahyd8z8hww47l/ELT%20vs%20ETL.pdf?dl=0>

How the Course is Structured:

<https://www.dropbox.com/s/1ieqxa134sqvym/How%20the%20Course%20is%20Structured%20%28DBT%29.pdf?dl=0>

Useful Links

Signing up:

<https://www.getdbt.com/signup/>

<https://github.com/join>

<https://signup.snowflake.com/>

hub.getdbt.com (packages)

Commands

runs all dbt files related to the project:

```
dbt run
```

runs only a specific model and its downstream file

```
dbt run -s +model_name
```

runs only a specific model and its upstream file

```
dbt run -s model_name+
```

runs test

```
dbt test
```

runs docs

```
dbt docs generate
```

runs seeds

```
dbt seed
```

runs packages

```
dbt deps
```

Modularity

Modularity

stg_orders

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Product ID	Order Cost Price	Order Selling Price	Order Profit
1	CA-2017-152156	2021/11/08	2021/11/11	Second Class	CG-12520	FUR-BO-10001798	472	786	314
2	CA-2017-152156	2021/11/08	2021/11/11	Second Class	CG-12520	FUR-CH-10000454	1 098	2 196	1 098
3	CA-2017-138688	2021/06/12	2021/06/16	Second Class	DV-13045	OFF-LA-10000240	66	73	7
4	US-2016-108966	2020/10/11	2020/10/18	Standard Class	SO-20335	FUR-TA-10000577	3 447	5 745	2 298
5	US-2016-108966	2020/10/11	2020/10/18	Standard Class	SO-20335	OFF-ST-10000760	81	134	54
6	CA-2015-115812	2019/06/09	2019/06/14	Standard Class	BH-11710	FUR-FU-10001487	195	391	195
7	CA-2015-115812	2019/06/09	2019/06/14	Standard Class	BH-11710	OFF-AR-10002833	41	58	17
8	CA-2015-115812	2019/06/09	2019/06/14	Standard Class	BH-11710	TEC-PH-10002275	4 354	7 257	2 903
9	CA-2015-115812	2019/06/09	2019/06/14	Standard Class	BH-11710	OFF-BI-10003910	133	148	15

Total Profit by Product

ProductID	Product Name	Category	Sub Category	
FUR-BO-10000112	Bush Birmingham Collecti..	Furniture	Bookcases	578
FUR-BO-10000330	Sauder Camden County B..	Furniture	Bookcases	1 979
FUR-BO-10000362	Sauder Inglewood Library ..	Furniture	Bookcases	3 618
FUR-BO-10000468	O'Sullivan 2-Shelf Heavy-..	Furniture	Bookcases	1 333
FUR-BO-10000711	Hon Metal Bookcases- Gray	Furniture	Bookcases	2 406
FUR-BO-10000780	O'Sullivan Plantations 2-..	Furniture	Bookcases	12 457
FUR-BO-10001337	O'Sullivan Living Dimensi..	Furniture	Bookcases	5 509
FUR-BO-10001519	O'Sullivan 3-Shelf Heavy-..	Furniture	Bookcases	1 961
FUR-BO-10001567	Bush Westfield Collection ..	Furniture	Bookcases	318

Total Profit by Customer

CustomerID	Segment	Customer Name	Country	
GT-14710	Consumer	Greg Tran	United States	54 456
KL-16645	Consumer	Ken Lonsdale	United States	52 598
AB-10105	Consumer	Adrian Barton	United States	50 448
CL-12565	Consumer	Clay Ludtke	United States	49 080
SC-20095	Consumer	Sanjit Chand	United States	41 169
AH-10690	Corporate	Anna Høberlin	United States	40 769
SE-20110	Consumer	Sanjit Engle	United States	38 369
MS-17365	Consumer	Maribeth Schnellling	United States	37 389
SM-20320	Home Office	Sean Miller	United States	36 347

Code Snippets

Creating Models

stg_orders.sql

```
select
--from raw_orders
orderid,
orderdate,
shipdate,
shipmode,
o.customerid,
o.productid,
ordersellingprice,
ordercostprice,
--from raw_customer
customername,
segment,
country,
--from raw_product
category,
productname,
subcategory,
ordersellingprice - ordercostprice as orderprofit
from {{ ref('raw_orders') }} as o
left join {{ ref('raw_customer') }} as c
on o.customerid = c.customerid
left join {{ ref('raw_product') }} as p
on o.productid = p.productid
```

report_profit_by_product.sql

```

select
  productid,
  productname,
  category,
  subcategory,
  sum(orderprofit) as profit
from {{ ref('stg_orders') }}
group by
  productid,
  productname,
  category,
  subcategory

```

report_profit_by_customer.sql

```

select
  customerid,
  segment,
  country
  sum(orderprofit) as profit
from {{ ref('stg_orders') }}
group by
  customerid,
  segment,
  country

```

Sources - Source Freshness Test Example

```

version: 2

sources:
  - name: globalmart
    database: raw
    schema: globalmart
    tables:
      - name: customer
      - name: orders
      - name: product
    columns:
      loaded_at_field: loadedat
    freshness:
      warn_after: {count:12, period:hour}
      error_after: {count:24, period:hour}

```

Testing

Go from reactive to proactive

DBT enables data teams to deploy with the same confidence of software engineers, with rapid, reliable testing. It's important to do testing so you can know what exactly goes wrong and how to fix it. DBT allows testing at scale which is so easy and it's run with the command `dbt test`.

SCHEMA TESTING

Validate essential data quality

Out of the box, dbt supports schema tests for uniqueness, null or accepted values, or referential integrity between tables.

These can be extended by writing your own custom schema tests.

DATA VALUE TESTING

Flag out-of-range values

Define data test failure conditions via plain SQL SELECT statements.

So when you write a model, you can also write a test for that model and they can run simultaneously. You can continue to run these tests every day etc and you have the confidence that the data has been tested.

Types of Tests

- singular - a specific test for a specific model
- generic tests - scalable tests where you write a few lines of yml code and then testing a model or a column
 - unique - every value in a column of a model is unique
 - not null - every value in a column of a model is not null
 - accepted values - every value in the column exists in a given lists
 - relationships - each value in a column exists in a column of another table

These can be added to your project quickly

Generic Tests

raw_globalmart.yml

```
version: 2

models:
  - name: raw_customer
    columns:
      - name: customerid
        tests:
          - unique
          - not_null
  - name: raw_orders
    columns:
      - name: orderid
        tests:
          - unique
      - name: shipmode
        tests:
          - accepted_values:
              values:
                - 'First Class'
                - 'Same Day'
                - 'Second Class'
                - 'Standard Class'
```

Singular Tests

test_raw_orders_selling_price_is_positive.sql

```
with

orders as (
  select * from {{ ref('raw_orders') }}
)
```

```
select orderid,
sum(ordersellingprice) as total_sp
from orders
group by orderid
having total_sp<0
```

Testing Sources

src_globalmart.yml

```
version: 2

sources:
  - name: globalmart
    database: raw
    schema: globalmart
    tables:
      - name: customer

      - name: orders
        columns:
          - name: orderid
            tests:
              - unique
              - not_null

      - name: product
```

Documentation in DBT

stg_globalmart.yml

```
version: 2

models:
  - name: stg_orders
    description: one unique order per row
    columns:
      - name: orderid
        description: the primary key for stg_orders
```

markdown to copy:

shipmode	definition
First Class	Orders are shipped via First Class with Courier
Second Class	Orders are shipped via Second Class with Courier
Standard Class	Orders are shipped via Standard Class with Courier
Same Day	Orders are personally shipped via Globalmart Team

globalmart.md

```
{% docs shipmode %}

One of the following values:
```

shipmode	definition
First Class	Orders are shipped via First Class with Courier
Second Class	Orders are shipped via Second Class with Courier
Standard Class	Orders are shipped via Standard Class with Courier
Same Day	Orders are personally shipped via Globalmart Team

{% enddocs %}

stg_globalmart.yml

```
version: 2

models:
  - name: stg_orders
    description: one unique order per row
    columns:
      - name: orderid
        description: the primary key for stg_orders
      - name: shipmode
        description: '{{doc("shipmode")}}'
```

Documentation for Source Files

src_globalmart

```
version: 2

sources:
  - name: globalmart
    description: a clone of the Snowflake Database
    database: raw
    schema: globalmart
    tables:
      - name: customer
      - name: orders
        description: raw orders table
        columns:
          - name: orderid
            description: primary key for orders
        tests:
          - unique
          - not_null
      - name: product
```

Jinja Code

Example 1

```
{%- set tabletype = "orderstable" -%}

select
orderid,
'{{tabletype}}' as tabletype
from {{ ref('stg_orders') }}
```

Example 2

```
{%- set category = "Furniture" -%}

select
orderid,
case when category = '{{category}}' then orderprofit end as {{category}}_orderprofit
from {{ ref('stg_orders') }}
```

Example 3

```
{%- set category = ["Furniture", "Office", "Technology"] -%}

select
orderid,
{% for category in category %}
sum(case when category = '{{category}}' then orderprofit end) as {{category}}_orderprofit
{% if not loop.last %},{% endif -%}
{% endfor %}
from {{ ref('stg_orders') }}
group by 1
```

Macros in DBT

markup.sql

```
{% macro markup() %}
(ordersellingprice-ordercostprice)/ordercostprice
{% endmacro %}
```

stg_orders with macro added

```
select
--from raw_orders
orderid,
orderdate,
shipdate,
shipmode,
o.customerid,
o.productid,
ordersellingprice,
ordercostprice,
--from raw_customer
customername,
segment,
country,
--from raw_product
category,
productname,
subcategory,
ordersellingprice - ordercostprice as orderprofit,
{{ markup() }} as markup
from {{ ref('raw_orders') }} as o
left join {{ ref('raw_customer') }} as c
on o.customerid = c.customerid
left join {{ ref('raw_product') }} as p
on o.productid = p.productid
```

markup.sql (with arguments)


```
{% macro markup(column1, column2) -%}
({{column1}}-{{column2}})/{{column2}}
{% endmacro %}
```

limit_data_in_dev example

```
{% macro limit_data_in_dev(column_name) %}
{% if target.name == 'development' %}
where {{column_name}} >= dateadd('day', -30, current_timestamp)
{% endif %}
{% endmacro %}
```

adding limit_data_in_dev to stg_orders

```
select
--from raw_orders
orderid,
orderdate,
shipdate,
shipmode,
o.customerid,
o.productid,
ordersellingprice,
ordercostprice,
--from raw_customer
customername,
segment,
country,
--from raw_product
category,
productname,
subcategory,
ordersellingprice - ordercostprice as orderprofit,
{{ markup('ordersellingprice', 'ordercostprice' ) }} as markup
from {{ ref('raw_orders') }} as o
left join {{ ref('raw_customer') }} as c
on o.customerid = c.customerid
left join {{ ref('raw_product') }} as p
on o.productid = p.productid

{{limit_data_in_dev('orderdate')}}
```

DBT Packages

hub.getdbt.com has macro packages, install dbtutils

Install Package

Install the surrogate key package

so copy the snippet

```
{{ dbt_utils.surrogate_key(['field_a', 'field_b'[ , ... ]]) }}
```

adding it to stg_orders

```

select
--from raw_orders
{{ dbt_utils.surrogate_key(['orderid', 'o.customerid', 'o.productid']) }} as sk_orders,
orderid,
orderdate,
shipdate,
shipmode,
o.customerid,
o.productid,
ordersellingprice,
ordercostprice,
--from raw_customer
customername,
segment,
country,
--from raw_product
category,
productname,
subcategory,
ordersellingprice - ordercostprice as orderprofit,
{{ markup('ordersellingprice', 'ordercostprice' ) }} as markup
from {{ ref('raw_orders') }} as o
left join {{ ref('raw_customer') }} as c
on o.customerid = c.customerid
left join {{ ref('raw_product') }} as p
on o.productid = p.productid

{{limit_data_in_dev('orderdate')}}

```

Seeds

delivery_team.csv

```

shipmode,delivery_team
First Class, RHL_Couriers
Second Class, RHL_Couriers
Standard Class, RHL_Couriers
Same Day, Globalmart

```