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/jgv6h4nezmhx8ju/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/1ieqxa134sqvyms/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
```

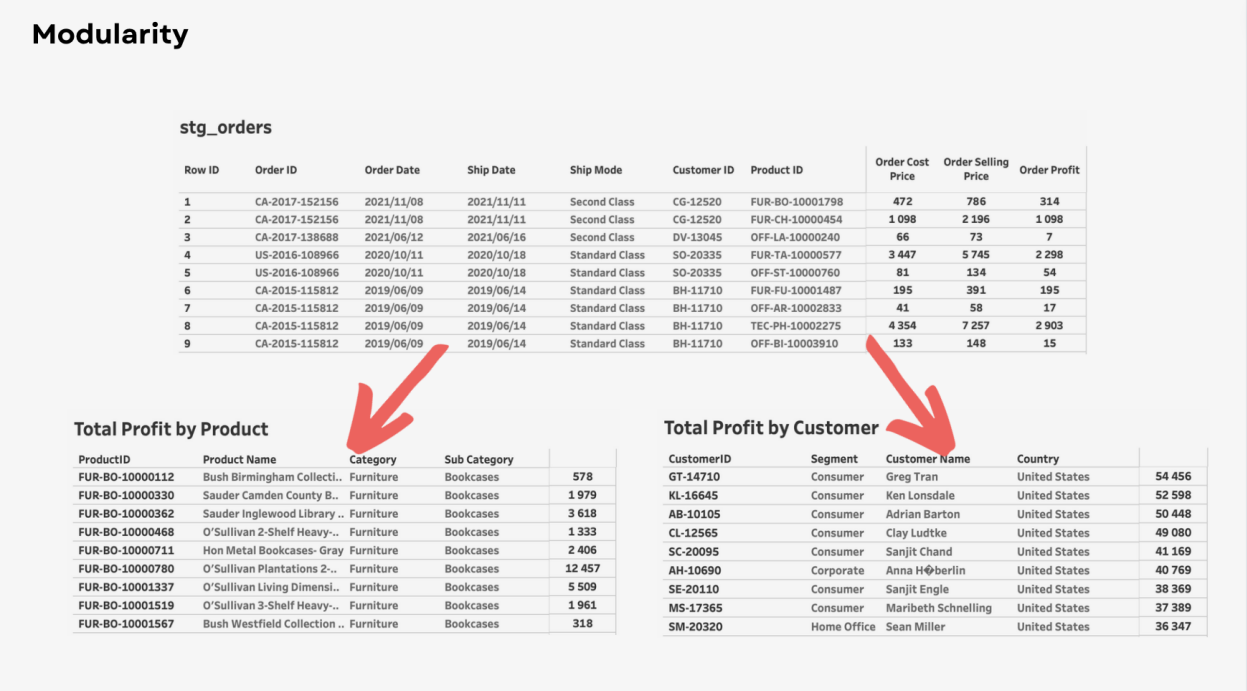
runs seeds

```
dbt seed
```

runs packages

```
dbt deps
```

Modularity



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
```