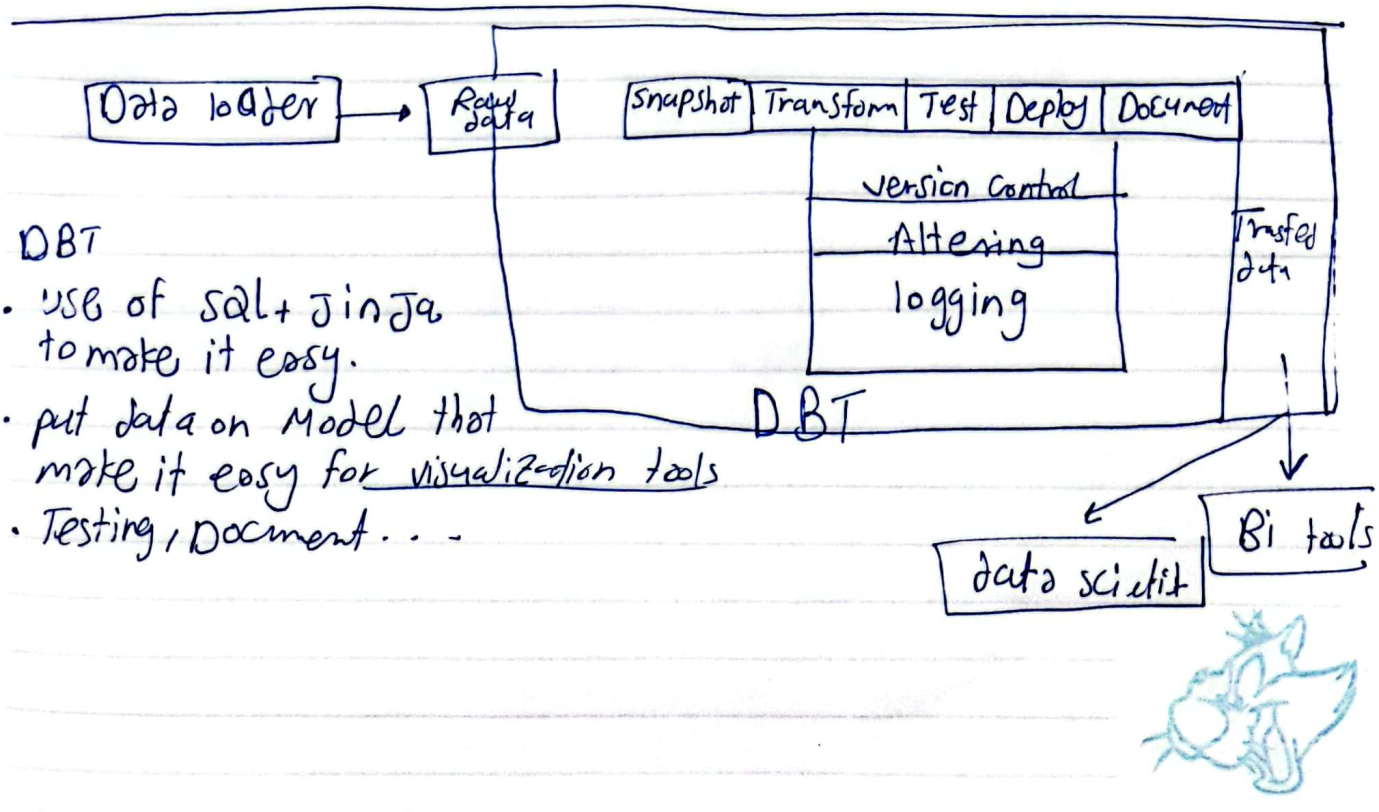
Week 4: Analytics Engineering

"Transform data in DW [Data warehouse] to Analytical views"

Content

- Introduction to analytics Eng
- What's dbt ?
- start dbt project
  - using BQ + dbt cloud (Alternative A)
  - using PostGress + dbt core (Alternative B)
- Development of dbt models.
- Testing & document dbt models.
- Deploying dbt project.
- Visualize transformed data.



DBT

- use of sql + jinja to make it easy.
- put data on Model that make it easy for visualization tools
- Testing, Document...

what's Analytics Engineering.

[presentation] Data Analysis
Data Engineer [physical work]

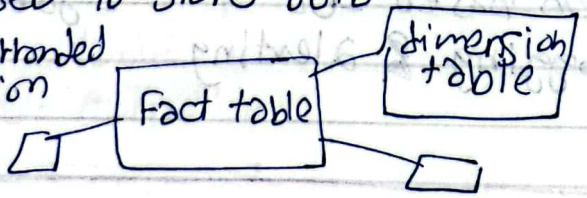
It's role that fill the gap between with a good software practice
- Data loading (stitch)
- Data storing (Data warehouse)
- Data Modeling (dbt, dataform)
- Data presentation (BI tools, Google studio, Tableau..)

- Now there is difference between [Analytics Engineer & Data Analysit]

Dimensional Modeling

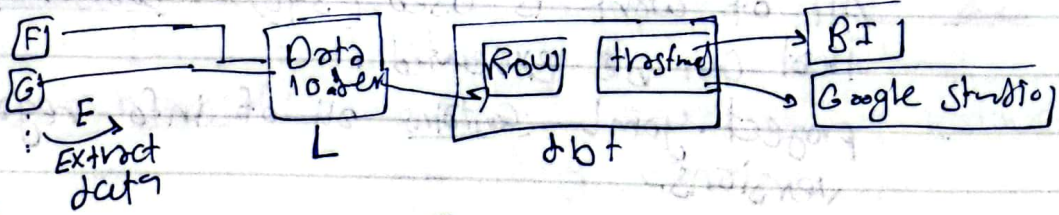
- Method that's used to store the data, store data in a certin structure to make it fast for Data analyst
- star schema

schema used to store data
Fact table surrended with dimession table.



Introduction to dbt [data build tool]

- we use dbt for transformation of raw data



ELT → Transformation means to trasfer raw data to dimensional model for fast analytics

- dbt
  transform process raw data in our Data warehouse to transformed data which can be later used by business intelligence.

- dbt allows:
  - Develope model.
  - Test and Document model.
  - Deploy model with version control.

- dbt works by define modeling layer which turn tables into models

- dbt have options -dbt (Core offline with postgres)
  -dbt (Cloud with bq or else)

- dbt cloud support:
  - web based IDE
  - Job schedule
  - logging & alerting

---

dbt cloud
- cloud give you folders like github with git ignore contains files or folders names to ignore.
- All of work is used (SQL) work.sql that can be excuted.
- project.yaml contains all of info regarding versions.

dbt core

- dbt_project.yml

```
name: 'tax_rides_ny'
version: '1.0.0'
config-version: 2

profile: 'pg-dbt-workshop'
```

↳ which profile dbt use local, cloud

Build first dbt Models

dbt Model my-model.sql

```
{{
  config(materialized = 'table')
}}
select *
from staging.source_table
where record_state = 'ACTIVE'
```

this {{ for Jinja code

Compiled code

```
sql code
```

(dbt →)

- dbt models [is essentially select query]
- materialized strategy for presenting dbt like
  1. table : Model will be built each time.
  2. incremental: add records incrementaly rather than rebuilding each run

From clauses
- to define source of data.
- we can define source freshness to get fresh records only.

Macros
pieces of code in jinja that can be reused
e.x config(), source()...

- Macros allows to add features that aren't otherwise available.
  - IF statment, loops.
  - use environmental variables.
  - operate on result of one query to make decision to another.

- Macros stored in .sql Files which stored in macros directory

- {% ..... %} statment (macros add(,))
- {# ..... #} Comment
- {{ ..... }} Expressions

e.x

```
{# Macro to return description of payment #}
{% macro get_pay_description(paymat_numb) %}
  Case {{ payment_numb }}
      when 1 then 'credit'
      when 2 then 'cash'
      when 3 then 'unknown'
    end
{% end macro %}
```

in the query

```
select
{{ get_pay_description(payment_numb) }} as payment_typedes
```

↳ fetch on data for payment_numb and replace each number with crossponding value and then save it as payment_type_description

---

- Packages
  - similar to other languages we can save macros to reuse them, and also we can download packages from internet.
    packages.yml

- Variables
  - variables can be used and defined across all of the project.
  - can be defined in different way:
    - vars:
      payment_numb : [1, 2, 3, 4 ...]
    - dbt build -m model.sql --var "is_test_run: false"
      ↳ when we build to check for var then make decision using macros.
    - {% if var('is_test_run', default=true) %}
      limit 100
      {% endif %}

* Refrencing older model to new one

- ref() is used for:
  - reference tables and views
  - referee seeds.

Testing and documenting

- Not required step, but expected in any professional setting.

- dbt provide few predefined test, but also you can use your test [custom].
  test writed in .yml

e.x

```
columns:
  - name: tripid
    tests:
      - unique:
          severity: warn   <- Action to do
      - not null:
          severity: warn
```

↳ that would print warnning in cli.

e.x custom

```
name: payment_type-description
tests:
  - accepted_values:
      values: [1, 2, 3, 4, 5]
      severity: warn
```

↳ warn when value of payment_type-descriptn not same.

Documenting

models:

- name: stg_yellow_tripdata
  description: >
    Trips made by New York City's iconic yellow taxi.
  Columns:

description used to describe the work or model and also we write it in .yml.

dbt provide a way to generate document and render it to a website.

it contains:

- information about project [model, code, dependencies sources, Descriptions from yml]
- information about Data warehouse

dbt can be rendered to website locally or in cloud.

> dbt docs generate
> dbt docs server

Deployment Basics

- running models on development environment
- Deployment workflow:
  1. Develop user branch.
  2. open PR to merge into branch.
  3. Merge user branch into main.
  4. Run new models in production environment using main branch.
  5. schedule models.

proged is set of jobs :
- Job is set of commands such build, test
- Job may be triggered manually or scheduled.
- other tools like airflow, cron can also work.
- If dbt source freshness command run you will view results at the end with it's freshness

---

Continuous Integration (CI)

CI job will do things such as build, test ---etc when a certin circumstances happen.
"triggerend on certin conditions"

e.x
- GitHub/GitLab pull requests that trigger CI using webhook
- AirFlow, cron job, Clould scheduler ....

Deployment using dbt clould

- dbt enforce you to creat PRS for CI pupose
- we will creat new environment
- we will add this jobs, and check settings like ☑ generate Docs

to run model sql ← 1. dbt seed 2. dbt run 3. dbt test

- After job finish you can see
  - .logs
  - .Document contain all info abont project.

Deployment local

- dbt could be deployed local, but with limited features.
- settings will be eddited from profiles.yml

Data visualization

- Google Data studio , - tableau .
- Meta base "offline"

"we used dbt with BQ to structure data into a good way to visualize it usig Data studio"

- GDS
  - online tool to convert data into reports and dashboard
  - once you select Data or tables you want you will see item type with description for all of them
  - you can press creat report wich after that can handel data types and plots.
  - GDS have different widgets.
  - GDS can connect to Bigquery and even more....

- GDS support diffrent types of charts [bar, lines, graphs, PI chart, Pivot .....] all what you thinks of python.

- gives you interactive dashboard like drop box

- drop box to Choose interval date.
- You Can add new column :
  - add field
  - Name it pickup_month → we have only daily records in our data
  - writ formula MONTH (pickup_datatime)
  - save it.

- Same if we want by year
  YEAR (pickup_datatime)

Tricks

- You Can share dashboard with others.
- schedule report in mail.
- Set freshness of dashboard.
- This Can help alot for Trade software with your own python code.
- live screen for moniter data.

Meta base

- Can do most of GDS work.
- All of that offline.
- Can work with postgress.
- Have flexiable dashboard ui.