**Data warehouse notes**

**0 Introduction:**

**Data warehouse:**

* central repository which has data from disparate sources, consistent and comprehensive data analysis
* current and historical data in one place which means we keep historical data for analyzing by time for different use cases
* separation from operations: data in data warehouse separate from operation databases because it helps improve performance, ensure data consistency, and support better decision-making across the business.
* separate compute and storage computational resources (like processing power and memory) are kept independent from the storage resources (where data is kept). This separation provides greater flexibility, scalability, and cost-efficiency
* scalable: horizontally, vertically
* support bi, analysis, reporting

**Data warehouse lifecycle:**

**01 Course structure: stack technology**

**02 Setup snowflake snowsql:**

Path for config file in snowflake: /Users/rajneet/.snowsql/config/

Account name

ev60976.eu-north-1.aws

Terminal/Git commands:

|  |  |
| --- | --- |
| cd | Go back to home directory |
| ls | List files and direcxtories in the current directory |
| pwd | Print current directory |
| cd .. | Go up one directory level |
| cd ~ | Go to the home directory |
| cmd + k | clear |
| cd <DIRECTORY> | Change directory |
| . | Current directory |
| .. | Home directory |
| git clone | create a copy of a specific repository from a remote source, such as GitHub |
| mkdir | make directory |
| rmdir | Remove directory |
| ls -a | Get hidden files also |
| .git | Git can Version control |
| .gitignore | Because we choose python in version control |
| cat | Is used for read .gitignore file |
| code . | Open visual studio |
| ls -al | Authentication(read, write or read/write) |
| git status | used to display the current state of a Git repository. |
| git add . | used to stage changes in your working directory for the next commit in a Git repository. |
| git push | this command will push your changes to the corresponding branch on the default remote repository |
| git commit -m”message” | command is used to save your changes to the local repository in Git |
| git pull |  |
| Git branch | All branches in local repository |
| Git checkout -b <branch name> | Create new branch and switched to that branch |
| Git checkout<branch\_name> |  |
| Git push origin <Branch\_name> |  |

**03 Snowsight navigation:**

**04 What is snowflake: (data warehouse)**

**We need scalability when there is increase workload, So need to buy more hardware which is not flexible, hard to manage and large upfront cost.**

**Snowflake three- tier architecture:**

1. authentication and access control 1) virtual warehouses.
2. infrastructure
3. optimizer
4. metadata
5. security

**Pricing:**

* Storage cost: capacity storage (upfront cost), on demand storage, storage fees per month, time-travel & fail-safe cost
* Compute cost: based on credit consumption, 1 credit per node/hour, warehouse active, billed per second minimum 1 min.
* Cloud services: computer storage, data transfer, indirect cost related to cloud providers
* Billing: credits transformed to currency based on snowflake edition or cloud provider & region chosen

**Securable objects:**

* Organization: organization have several accounts
* Account: on account can have several databases
* Databases: in database we have information schema and other schema
* Schema: organize database objects such as tables and view

**Virtual warehouse:**

Scaling up: good for handling large but few workloads, to scale up choose t shirt size L

* Warehouse size: XS (1 node), S ( 2 nodes), M (3 nodes), L (4 nodes)

Scaling out: good for handling many workloads in parallel

**Virtual warehouse commands:**

show warehouses;

create warehouse demo\_warehouse

*with*

warehouse\_size = 'X-small'

auto\_suspend = 300

auto\_resume = TRUE

initially\_suspended = true

*comment* = 'demo warehouse through worksheet';

alter warehouse compute\_wh

set auto\_suspend = 120;

drop warehouse demo\_warehouse

database , table, schema commands:

show databases;

create database if not exists demo\_db;

use database demo\_db;

show schemas in demo\_db;

create schema if not exists demo\_db\_staging;

create table if not exists customer(

customer\_id int *primary key*,

age integer,

email varchar(50));

show tables;

select \* from customer;

insert into customer(customer\_id, age, email)

values

(1,32,'abc@gmail.com'),

(2,23,'def@gmail.com');

**Revise sql :**

show databases;

use database alesco\_consumer\_database\_sample;

show schemas;

use schema public;

show tables;

show views;

select \* from consumer\_test\_data\_view limit 10;

DESC view consumer\_test\_data\_view;

select count(\*) as number\_of\_rows from consumer\_test\_data\_view;

select distinct prefix from consumer\_test\_data\_view;

select count(distinct first) as number\_unique\_name from consumer\_test\_data\_view;

select count(distinct city) as number\_unique\_city from consumer\_test\_data\_view;

select count(\*) as number\_unique\_city, city

from consumer\_test\_data\_view

group by city

order by number\_unique\_city desc;

select count(\*) as number\_unique\_city, city

from consumer\_test\_data\_view

group by city

having city in('NEW YORK', 'DAYTON', 'APOPKA')

order by number\_unique\_city desc;

select count(\*) as number\_unique\_city, city, state

from consumer\_test\_data\_view

group by city, state

having city in('NEW YORK', 'DAYTON', 'APOPKA')

order by number\_unique\_city desc;

**Glossary:**

|  |  |
| --- | --- |
| Data Warehouse | digital library where all the information a company collects is organized and stored |
| Upstream | The steps involved in collecting, generating, or processing data before it is stored in the data warehouse |
| Downstream | The steps that happen after the data is stored—accessing, analyzing, reporting, distributing, and using the data to make decisions. |
| OLTP | Online transactional processing |
| OLAP | Online analytical processing |
| Cloud computing | It is a specific type of computing that leverages remote, internet-based resources managed by a cloud provider, offering greater flexibility, scalability, and reduced management overhead. |
| Computing | It is the broader term, referring to using computers and other devices to process data and perform tasks, usually on physical hardware you manage yourself. |
| Storage | the technology and methods used to store digital data |
| Scalable | the ability of a system, process, or technology to handle increased workload or demand without compromising performance or efficiency. |
| Vertical scalability (Scaling Up) | This involves adding more power to an existing system, such as upgrading the CPU, adding more RAM, or increasing storage capacity in a single server. |
| Horizontal scalability (Scaling out) | This involves adding more machines or nodes to a system, such as adding more servers to a network or cloud environment |
| dlt | Data load |
| dbt | Data build tool |
| IDE | Integrated development environment |
| LLM | Large language model |
| CLI | Command line interface |
| Meta data | This describes the structure of the database, including tables, columns, data types, relationships, and constraints. |
| Table | a fundamental structure in a database that stores data in a structured format, consisting of rows and columns |
| View | a virtual table that is derived from one or more tables (or other views). It does not store data itself but provides a way to present and query data from underlying tables in a specific manner. |
| CapEx | Capital expenditure |
| Virtual warehouse | **virtual warehouse** is a cloud-based resource in data warehousing that provides compute power for processing queries and managing data. Unlike a traditional physical warehouse, a virtual warehouse doesn't physically store data; instead, it accesses and processes data stored in a centralized data repository or cloud storage |
| Modern data stack | The **modern data stack** is a collection of cloud-native tools and technologies designed to manage, process, and analyze data in a highly efficient, scalable, and integrated way |
| DML | Data Manipulation language |
| DDL | Data Definition Language |
| DQL | Data Query Language |
| DCL | Data Control Language |
| snowflake credit | Snowflake credits are the fundamental unit of consumption in the Snowflake data platform, representing the cost associated with various operations such as data processing, storage, and services within a Snowflake account. |
| securable object | in Snowflake, a **securable object** refers to any resource within the platform that can have access control permissions (or privileges) applied to it. These objects are fundamental to managing security and access within Snowflake, allowing administrators to control who can view, modify, or manage different parts of the data infrastructure |
| snowflake object | In Snowflake, **objects** are the fundamental components used to store and manage data within the platform. Like database, schema, tables |
| schema | A **schema** in Snowflake is a logical container within a database that organizes and categorizes database objects, such as tables, views, and other related structures. |
| permanent table | Time travel, fail safe and long term storage |
| transient table | No time travel, no fail safe , short term storage |
| temporary table | Automatically dropped after session |
| fail-safe | Fail-Safe is a feature in Snowflake that provides an additional safety net for your data. It allows you to recover data even after the Time Travel period has ended. |
| time-travel | To get back data if we loose it, we can travel back. Time Travel is a feature in Snowflake that lets you look at your data as it was at a previous time. This means you can access and recover data that was changed or deleted. |
| external stage | Snowflake is a connection to an outside storage service, like Amazon S3, Google Cloud Storage, or Azure Blob Storage. It allows you to easily load data into Snowflake or export data from Snowflake without having to store everything directly in Snowflake. |
| data consumer | A **data consumer** is an individual or system that accesses and uses data for various purposes such as analysis, reporting, decision-making, or operational processes. |
| idempotent | **Idempotent** refers to an operation or action that can be performed multiple times without changing the result beyond the initial application. In other words, no matter how many times you execute it, the outcome remains the same after the first execution. |

**05\_users and roles in snowflake:**

Privileges then role then user

**06\_ setup dlt:**

Pip install uv

Uv venv

source .venv/bin/activate

uv pip install "dlt[snowflake]" ipykernel pandas

dlt –version

uv pip list

dlt init load\_snowflake snowflake(initate project)

go to dlt change configration in secrets.toml file in order to setup snowflake connection

uv pip freeze > requirements.txt

uv pip install -r requirements.txt

open ~/.dbt/profiles.yml

**07\_setup api:**

dlt init parkering snowflake

import dlt

import pandas as pd

from pathlib import Path

import os

@dlt.resource(*write\_disposition*="append")

def load\_snowflake\_resource(*file\_path*: str, \*\**kwargs*):

df = pd.read\_csv(file\_path, \*\*kwargs)

yield df.to\_dict(*orient* = "records")

if \_\_name\_\_ == "\_\_main\_\_":

working\_directory = Path(\_\_file\_\_).parent

os.chdir(working\_directory)

# specify the pipeline name, destination and dataset name when configuring pipeline,

# otherwise the defaults will be used that are derived from the current script name

pipeline = dlt.pipeline(

*pipeline\_name*='load\_snowflake',

*destination*='snowflake',

*dataset\_name*='staging', #schema

)

print(working\_directory)

data = list(load\_snowflake\_resource(working\_directory /"data"/"NetflixOriginals.csv", *encoding* = 'latin1'),)

# print the data yielded from resource

print(data)

# run the pipeline with your parameters

load\_info = pipeline.run(data, *table\_name*="netflix")

# pretty print the information on data that was loaded

print(load\_info)

**09\_ dimensional modelling:**

**Why?**

* Clear and visible business model
* Data more accessible in the organization
* Mental model of what to build
* Reduce cost and risks
* Query uability and performance

**What is dimensional modelling:** Dimensional modeling is a way to design databases that makes it easier to analyze data. It’s especially useful for data warehouses, which are large databases that store information from different sources and are used for reporting and analysis.

Facts: measurements you want to analyze. fact table represents each row is transaction

Dimension: Dimensions provide context to the facts. Fact table enrich by dimensions.

**4 steps of dimensional modelling:**

1. Find the bussiness process: achieve specific goals
2. Define the grain: more granular, the better(atomic grain, transcational grain). How should one row look like
3. Identify the dimension
4. Identify facts

uv pip install dbt-snowflake

dbt init dbt\_code

.dbt/profiles.yml

.dbt/profiles.yml

dbt debug

**Glossary**

|  |  |
| --- | --- |
| DAC | Discretionary access control |
| RBAC | Role based access control |
| ORGADMIN | Create accounts and manage operations at organization level |
| ACCOUNTADMIN | Top level role, grant to few users |
| SECURITYADMIN | Manage objects grant globally |
| SYSADMIN | Create warehouse, create databases, create other objects, create all custom roles |
| USERADMIN | User and role managment |
| PUBLIC | Objects owned by public is available to everyone, pseudo-role granted to everyone |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |