

# **E-commerce - Automate ETL to sync data for batch processing**



# Table of Contents

01



**Project Background &  
Objectives**

02



**Data Warehouse Design**

03



**Auto ETL Process**

04



**Data Visualization**

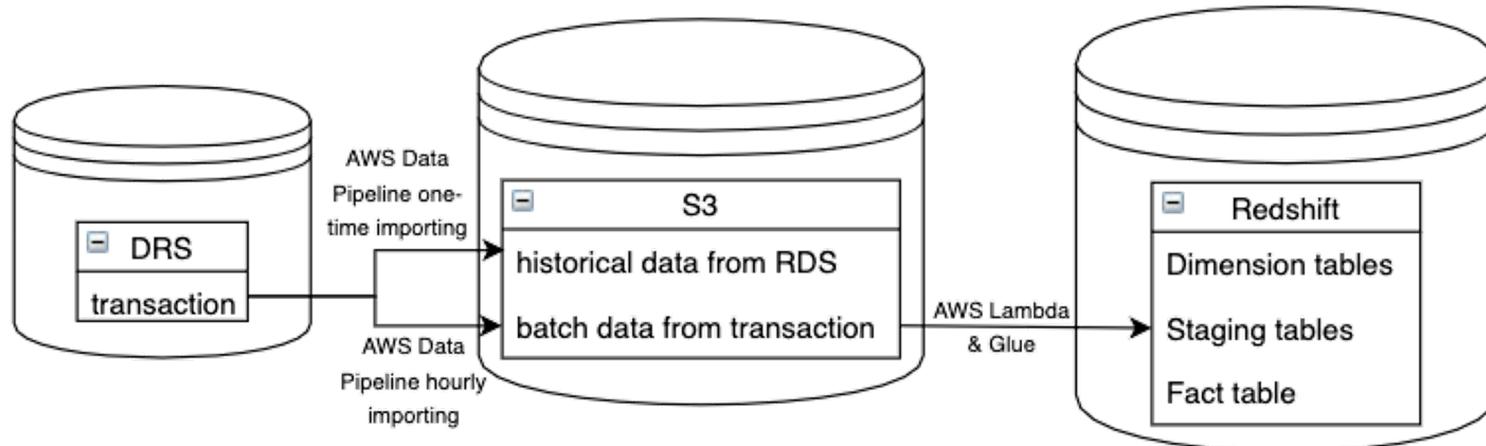
# 01 Project Background & Subjective

- Data comes from [Kaggle](#)
- E-commerce company X wants to migrate their historical data from traditional database RDS to Data Warehouse
- And then build automate data pipeline to do batch processing hourly to sync data for last 30 days (assuming orders' status will not change after 30 days)



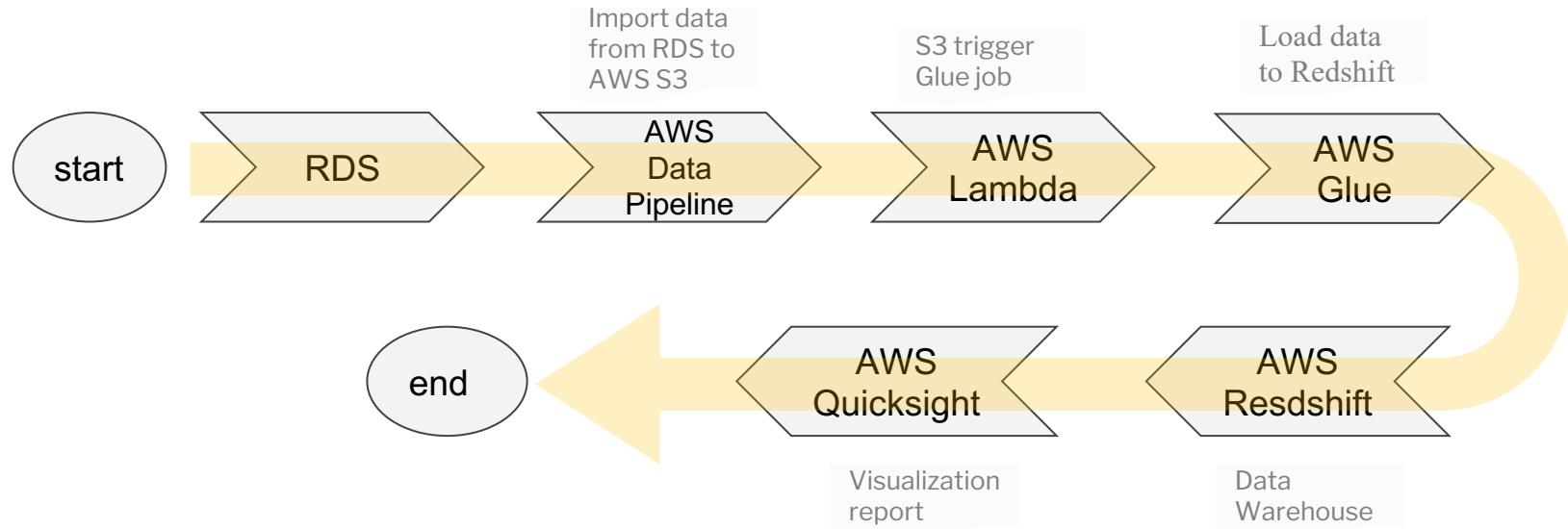
E-commerce  
company X

# 02 Data Warehouse Design

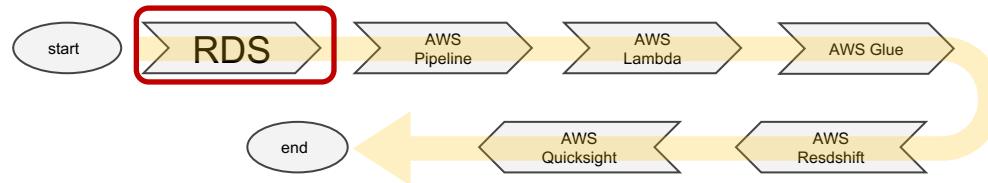


# 03 Auto ETL Process

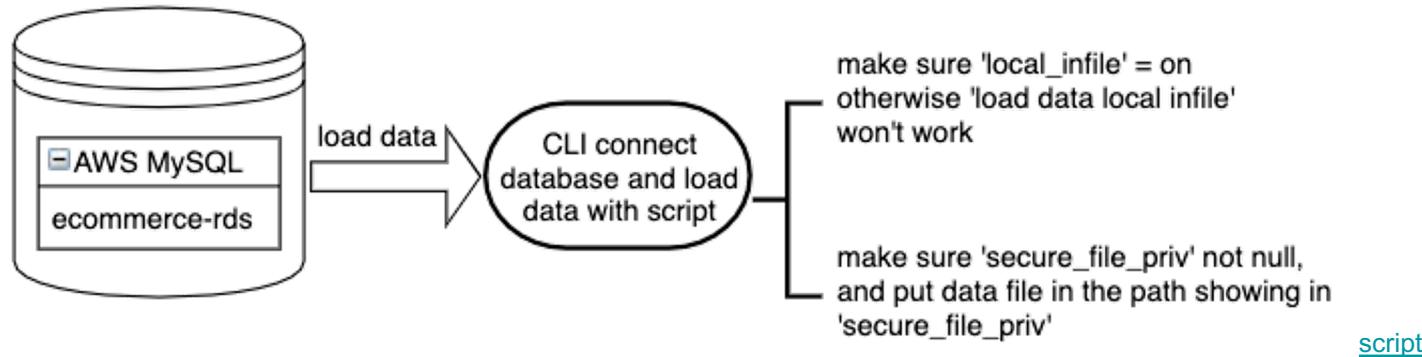
Project Process Flow Chart



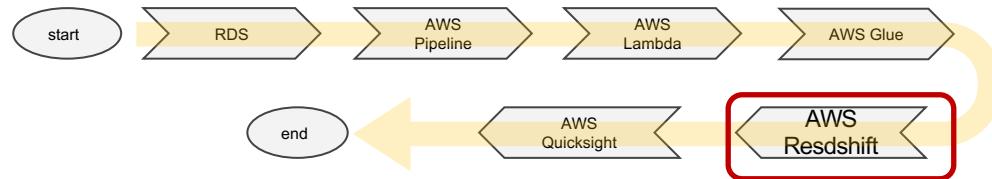
# 03 Auto ETL Process



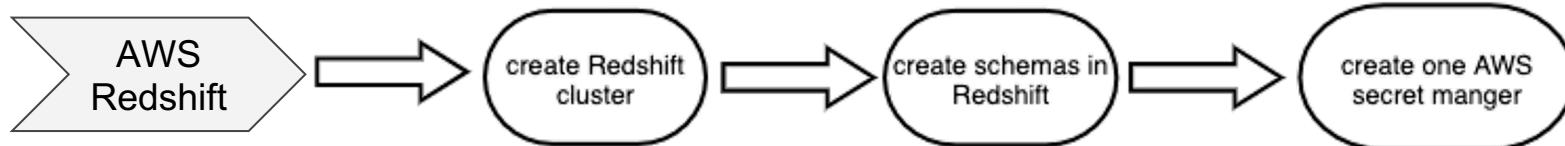
Since we are using Kaggle data to mimic real world job, we need to prepare data in Relational Database first.

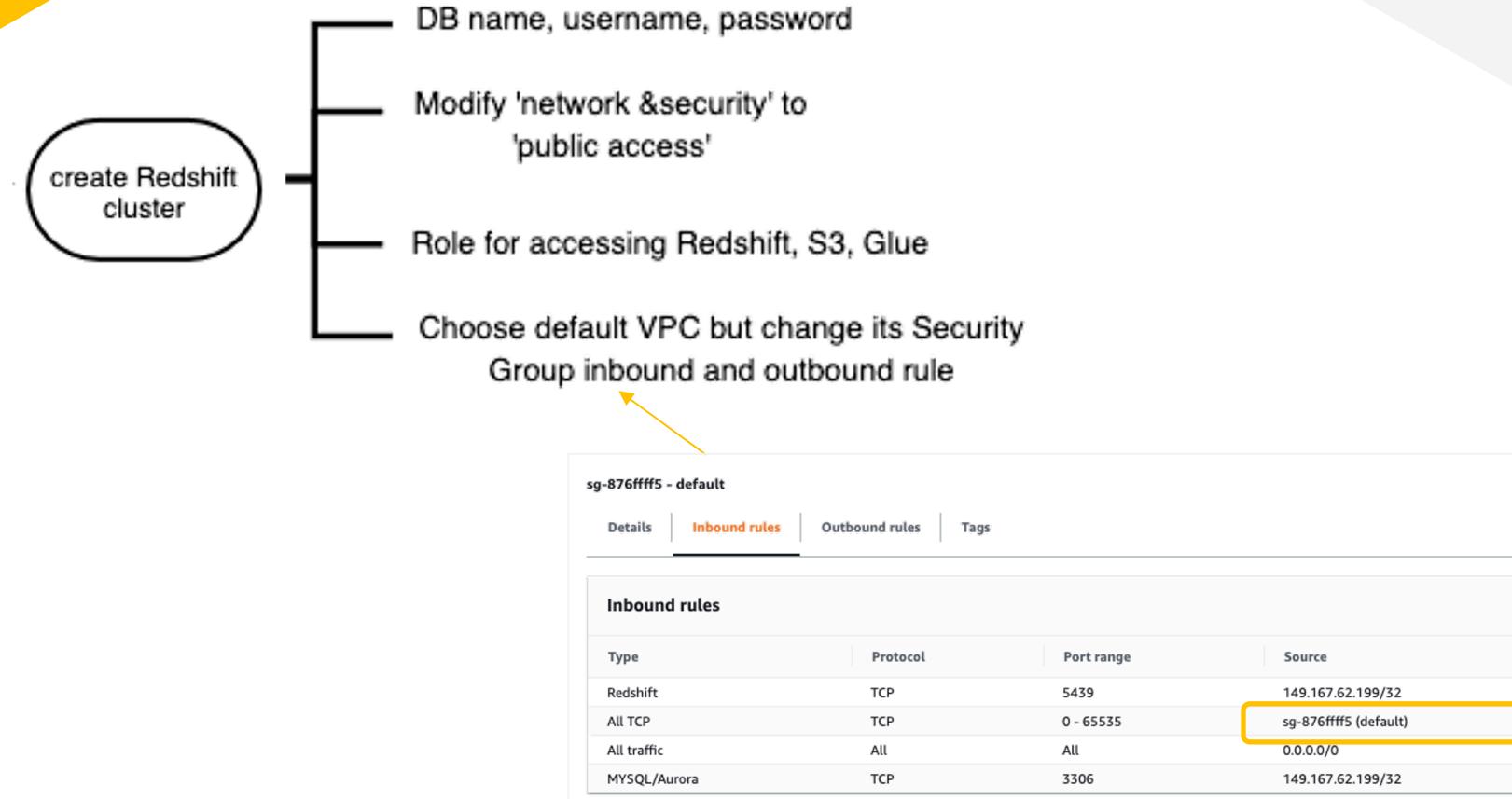


# 03 Auto ETL Process



Firstly, schemas were prepared in Data Warehouse Redshift

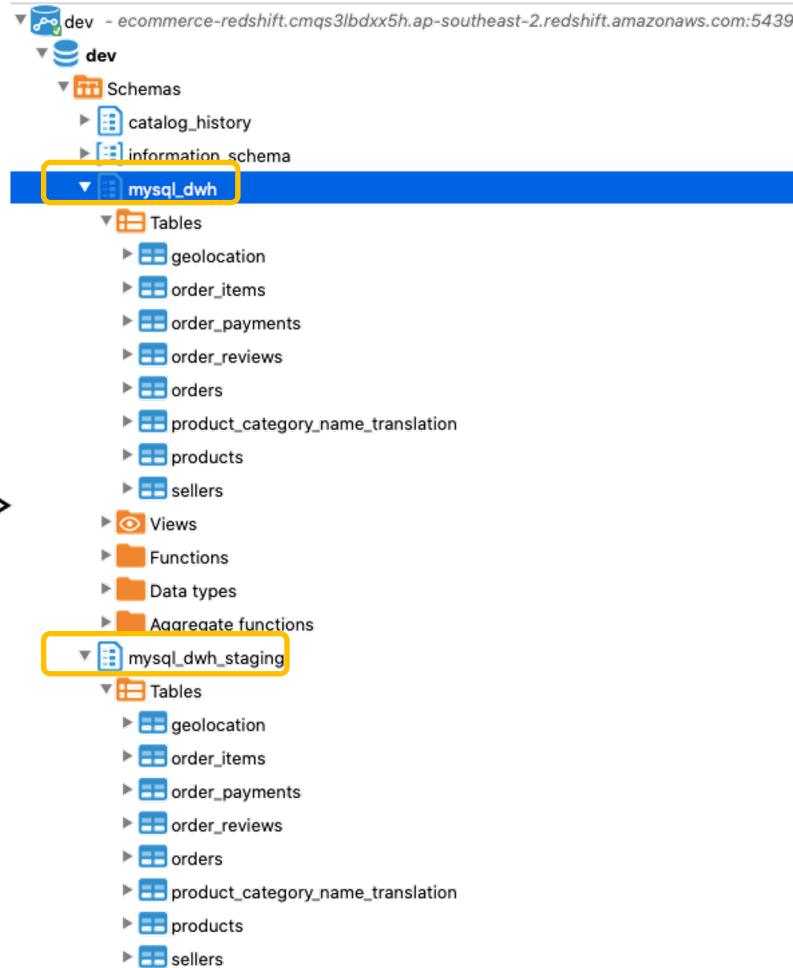
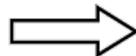




create schemas in Redshift

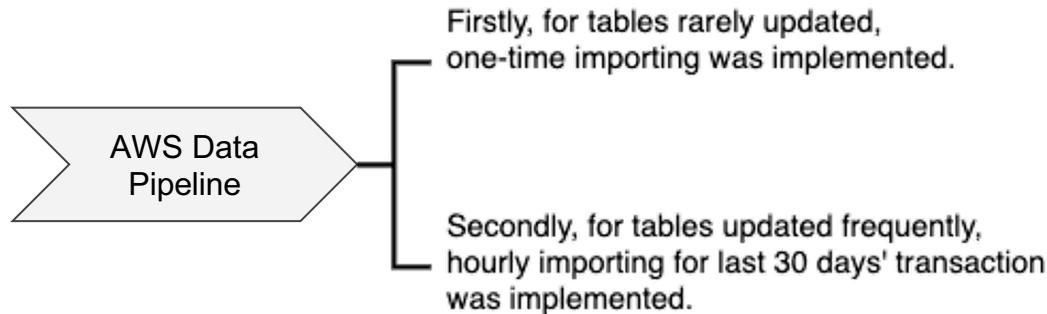
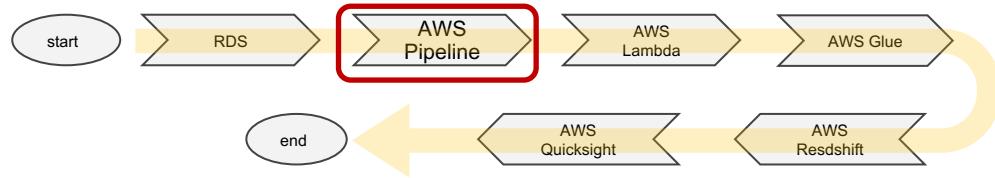


- Connect to cluster using command line
- source script to create schema



script

# 03 Auto ETL Process



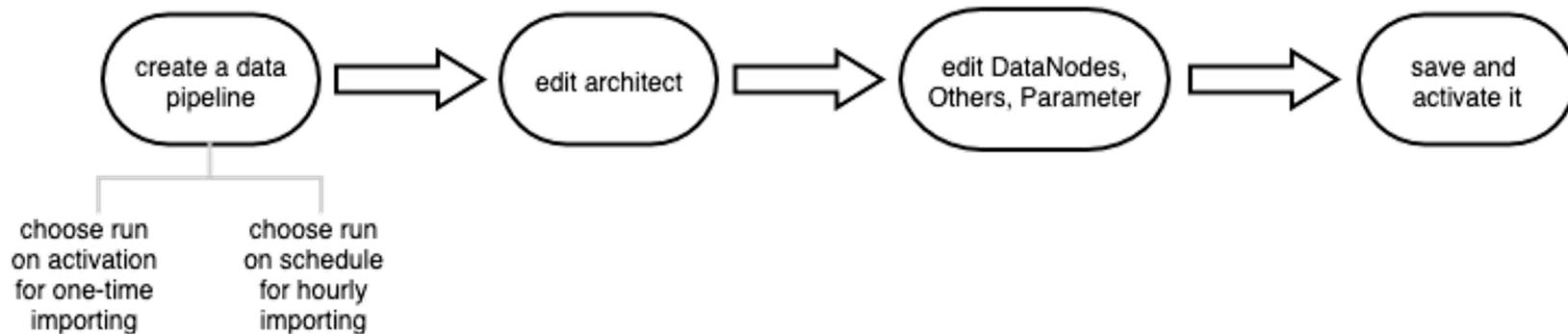
Note:

- One-time importing: click 'activate' to run data-pipeline whenever needed
- Hourly importing: auto-run data-pipeline every hour for last 30 days transaction

Script for  
DataNodes

# 03 Auto ETL Process

AWS Data Pipeline



create a data pipeline

## Create Pipeline

*(i)* You can create pipeline using a template or build one using the Architect page.

Name

Description (optional)

Source  Build using a template

Full copy of RDS MySQL table to S3

- Import a definition  
 Build using Architect

### Parameters

RDS MySQL password

Output S3 folder  

RDS MySQL username

RDS MySQL table name

Ec2 Security group(s) (optional)  

EC2 instance type

RDS Instance ID

### Schedule

*(i)* You can run your pipeline once or specify a schedule. [More](#)

Run  on pipeline activation

For one-time importing

on a schedule

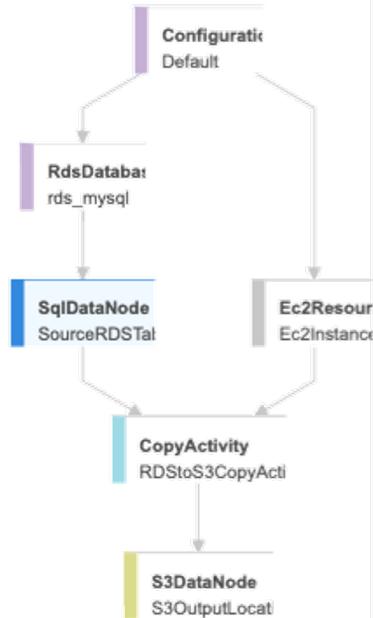
For hourly importing

### Pipeline Configuration

Logging  Enabled  
 Disabled

Copy execution logs to S3. [More](#)

edit DataNodes,  
Others, Parameter



Activities

DataNodes

select  
a.order\_id,  
a.order\_status,  
a.customer\_id,  
a.order\_approved\_at,  
a.order\_delivered\_carrier\_date,  
a.order\_delivered\_customer\_date,  
a.order\_estimated\_delivery\_date,  
a.order\_purchase\_timestamp,  
b.customer\_city,  
b.customer\_state,  
b.customer\_zip\_code\_prefix  
from  
ecommerce\_db.orders a  
join  
ecommerce\_db.customers b  
on  
a.customer\_id = b.customer\_id  
where  
date(a.order\_purchase\_timestamp) <= '2018-09-30'

Write query logic here

Select Query:

Add an optional field... ▾

S3OutputLocation

Name: S3OutputLocation

Type: S3DataNode

File Path: #{myOutputS3Loc}/orders.csv

Add your own file path

Schedules

Resources

Preconditions

Others

This panel shows the configuration for a **DataNodes** activity. It contains a SQL query for selecting data from the **ecommerce\_db** tables **orders** and **customers**. The query filters orders made on or before September 30, 2018. A yellow box highlights the **Select Query:** field, with an arrow pointing to the text "Write query logic here". Another yellow box highlights the **File Path:** field, containing the placeholder `#{myOutputS3Loc}/orders.csv`, with an arrow pointing to the text "Add your own file path". The panel also lists sections for **Schedules**, **Resources**, **Preconditions**, and **Others**.

edit DataNodes,  
Others, Parameter

▼ Others

edit DataNodes, Others, Parameter

rds\_mysql

Name: rds\_mysql

Type: RdsDatabase

Database Name: ecommerce\_db

\*Password: #{"myRDSPassword"}

Jdbc Properties: allowMultiQueries=true

Jdbc Properties: zeroDateTimeBehavior=convertT

Jdbc Driver Jar Url: s3://ecommerce-rds-dwh/driver/mysql-connector-java-5.1.48-bin.jar

Rds Instance Id: #{myRDSInstanceId}

Username: #{myRDSUsername}

Add an optional field...

ecommerce\_db Add your own DB name

zeroDateTimeBehavior=convertT avoid '0000-00-00 00:00:00' can not be represented as java.sql.Timestamp

s3://ecommerce-rds-dwh/driver/mysql-connector-java-5.1.48-bin.jar https://www.web3us.com/how-guides/aws-data-pipeline-rds-mysql-s3-unable

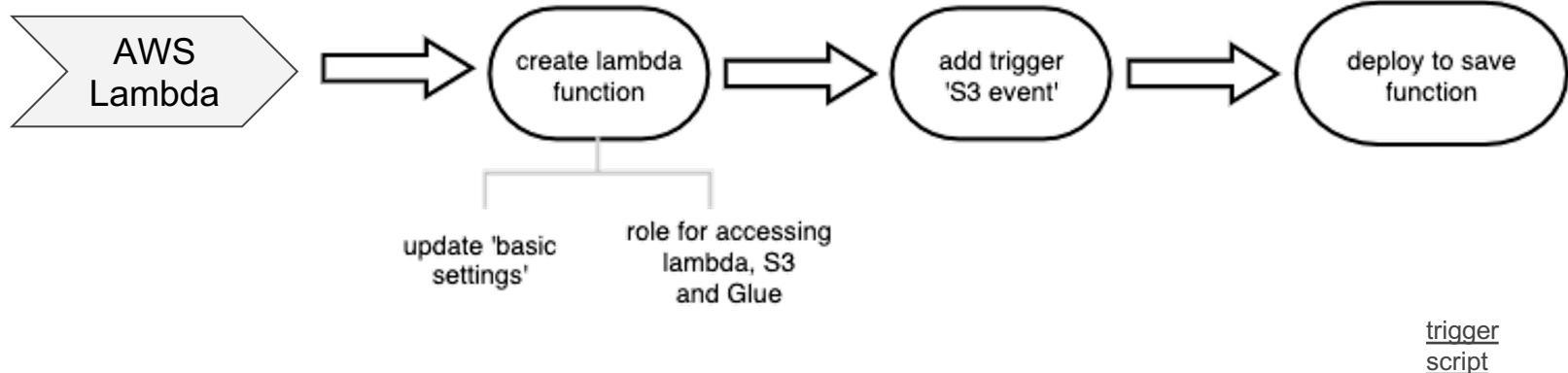
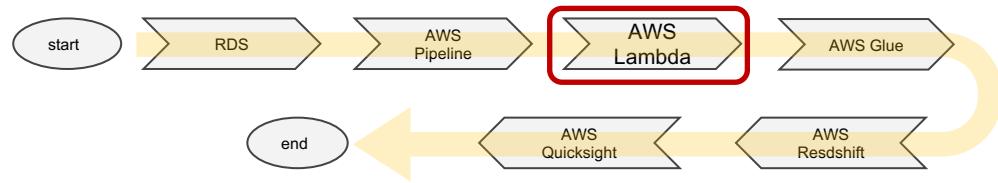
edit DataNodes.  
Others Parameter

Others

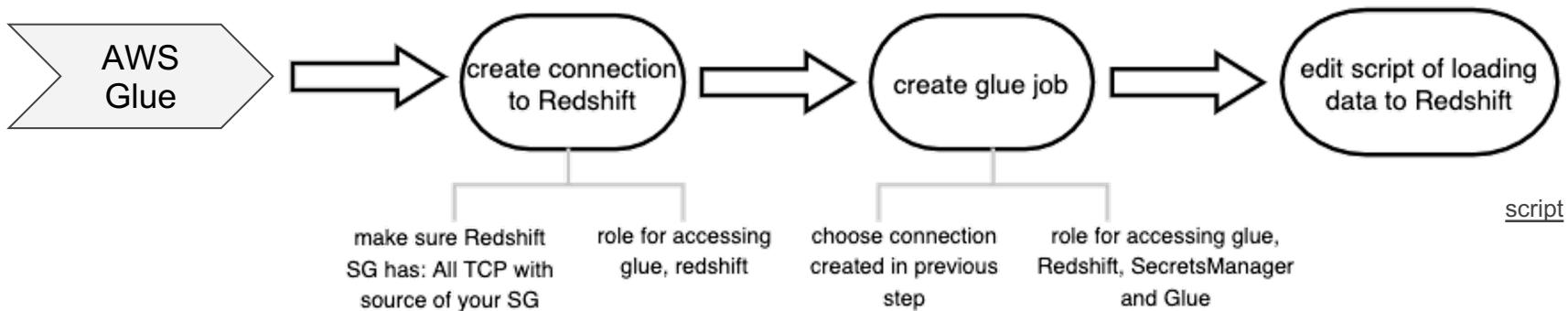
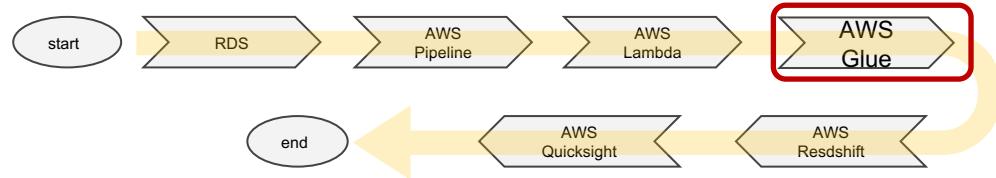
Parameters

*myRDSPassword	.....	your DB password
myOutputS3Loc	s3://ecommerce-rds-dwh/orders/historical	
myRDSUsername	etl_admin	your DB username
myRDSTableName	orders	
myEc2RdsSecurityGrps (optional)	security group name	
myEC2InstanceType	t1.micro	
myRDSInstanceId	etl-rds	your DB identifier

# 03 Auto ETL Process



# 03 Auto ETL Process



## create connection to Redshift



### Set up your connection's properties.

For more information, see [Working with Connections](#).

#### Connection name

test

#### Connection type

Amazon Redshift

Require SSL connection

Fail if unable to connect over SSL

#### Description (optional)

Enter description...

Next

For more information, see [Working with Connections](#).

#### JDBC URL

jdbc:redshift://ecommerce-redshift.cmqs3lbdxx5h.ap-southeast-2.redshift.amazonaws.com:5439/dev

JDBC syntax for most database engines is jdbc:protocol://host:port/databasename.

SQL Server syntax is jdbc:sqlserver://host:port;databaseName=db\_name. Oracle syntax is jdbc:oracle:thin://host:port/service\_name. For more variations, see [Working with Connections](#).

#### Username

awsuser

#### Password

\*\*\*\*\*

#### VPC

Choose the VPC name that contains your data store.

vpc-eaf7f78d | redshift

#### Subnet

Choose the subnet within your VPC.

subnet-4fea7017

#### Security groups

Choose one or more security groups that allow access to the data store in your VPC. AWS Glue associates these security groups to the ENI attached to your subnet. To allow AWS Glue components to communicate and also prevent access from other networks, at least one chosen security group must specify a self-referencing inbound rule for all TCP ports.

Group ID

sg-0000000000000000

sg-0b661c05227877f8c

sg-0c9d906189314c58d

sg-0ff1b723724e8268

sg-876ffff5

Add your own redshift cluster's information



create glue job

Configure the job properties

Name: import\_orders\_hourly

IAM role: ecomGlueRedshift

Type: Python shell

Python version: Python 3 (Glue Version 1.0)

This job runs:

An existing script that you provide

A new script to be authored by you

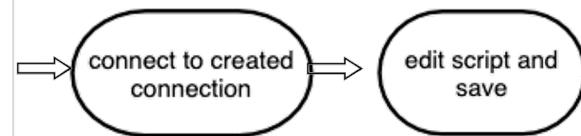
Script file name: import\_orders\_hourly

S3 path where the script is stored: s3://aws-glue-scripts-318140223133-ap-southeast-2/sl|x|

Tags (optional)

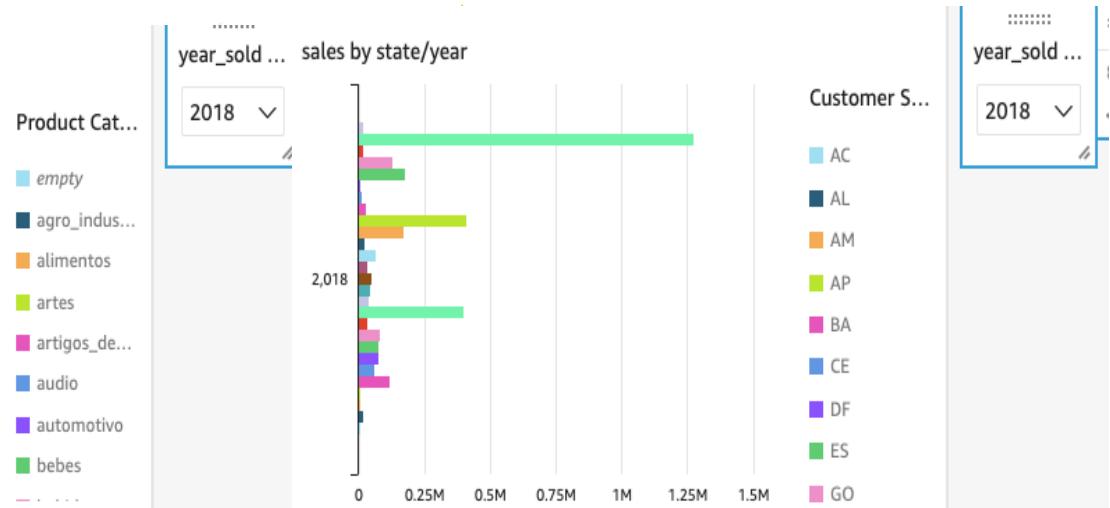
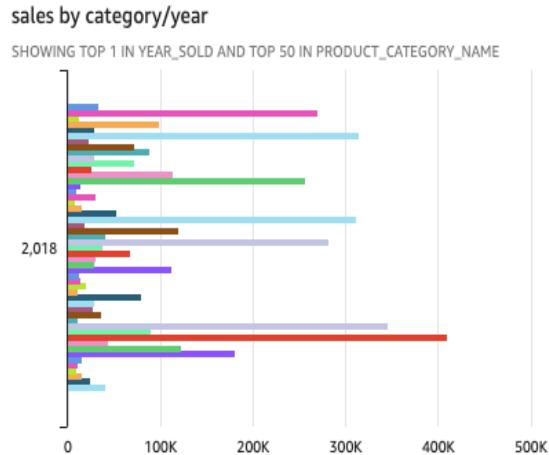
Security configuration, script libraries, and job parameters (optional)

Catalog options (optional)

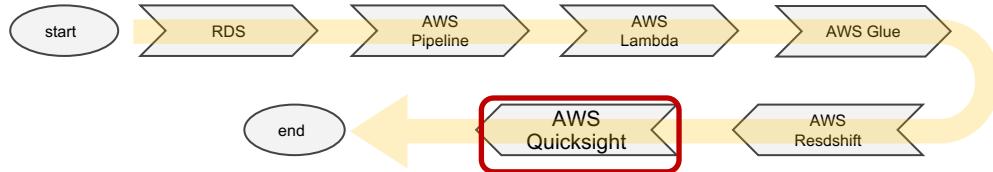


# 04 Data Visualization

- Use QuickSight, users can easily create reports,  
eg. report of year sales by category or state below



# 04 Data Visualization



create 'quick-sight'  
schema and views

- To manage everything easily, a separate schema for QuickSight was created to store all related views
- QuickSight can also do join too

The screenshot shows a database schema browser interface. On the left, the 'public' schema is expanded, revealing a folder named 'quicksight\_analysis'. This folder contains two items: 'Tables' and 'Views'. The 'Views' item is selected, highlighting 'quicksight\_sales\_by\_area' and 'quicksight\_sales\_by\_category'. On the right, the detailed SQL code for these views is shown.

```
create schema quicksight_analysis;
create or replace view quicksight_analysis.quicksight_sales_by_category as
select
p.product_category_name,
EXTRACT (year from date(op.order_purchase_timestamp)) as year_sold,
sum(op.payment_value) as value
FROM mysql_dwh.order_payments op
join mysql_dwh.order_items oi
on op.order_id = oi.order_id
join mysql_dwh.products p
on p.product_id = oi.product_id
group by 1,2;

create or replace view quicksight_analysis.quicksight_sales_by_area as
select
c.customer_city,
EXTRACT (year from date(op.order_purchase_timestamp)) as year_sold,
sum(op.payment_value) as value
FROM mysql_dwh.orders o
join mysql_dwh.customers c
on o.customer_id = c.customer_id
join mysql_dwh.order_payments op
on op.order_id = o.order_id
group by 1,2;
```

# To Add On...

- Scenario: combine user\_behavior data from the third party to do some analysis
- Solution: a) upload file to S3  
b) Glue crawler to glue data to data catalog  
c) check Athena to make sure table's importing  
d) use Redshift spectrum get external data from catalog, then do join with other tables if needed