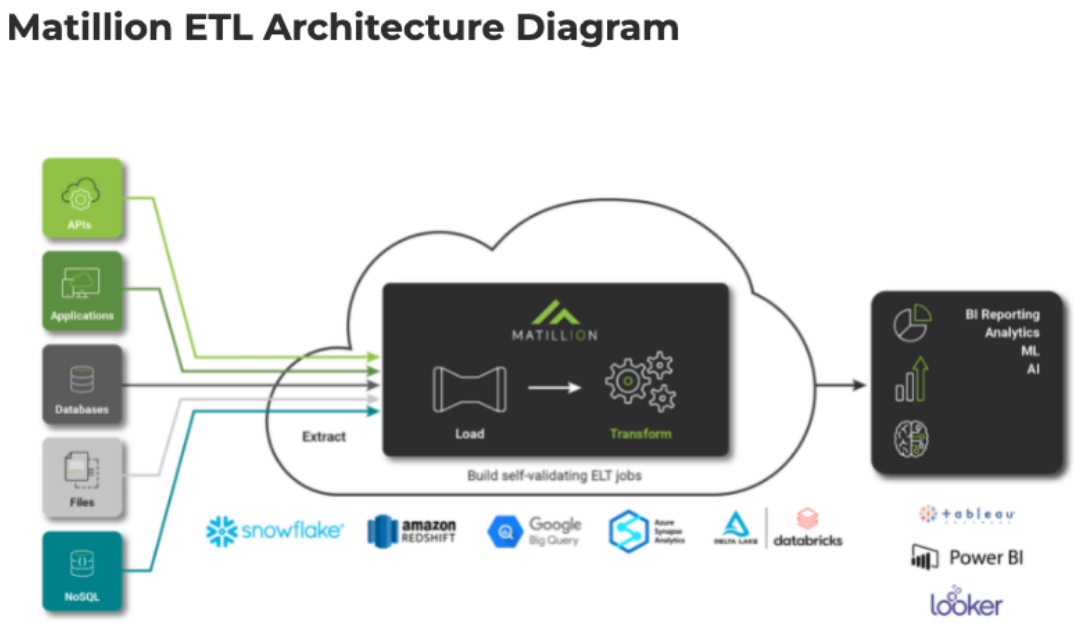
Sign up for the exam: https://www.webassessor.com/matillion
Online preparation course:
https://academy.matillion.com/certifications/0baba768-b088-11ea-b5e9-063e41907789
Read the documentation for specifics: https://documentation.matillion.com/docs

Loading and Transforming Data

- Using Matillion ETL, can load that any data into your cloud data warehouse and then transform that data, creating a full mock data warehouse for that input data
- Data Warehouse: system used for reporting and data analysis
    - It is considered a core component of business intelligence
    - They are central reporisoties of integrated data from one or more disparate sources
    - They store current and historical data into one single place thare are used for creating analytical reports for workers throughout the enterprise
    - A cloud data warehouse stores data in the cloud, which is represented as servers that can be accessed over the internet and your data is stored on machines spread across the world
- Dimension Table: Structure that categorizes facts and measures in order to enable users to answer business questions
    - Primary functions of dimensions are to provide filtering, grouping and labeling
- Fact Table: a fact is a value or measurement which represents a fact about the managed entity (stored in dimension tables)



- Matillion can be used for the loading of data into a data warehouse using Connector components from Redshift, BigQuery, Snowflake or Microsoft Synapse

- ETL represents the movement of data from one or many sources to a final repository, with transformation and manipulation of that data to meet business use cases and requirements applied to the data in transit
    - Matillion leverages the power of the underlying cloud data warehouse to optimize your workload efficiently, hence its speed and scalability
    - The transform step is not performed on a separate server, but on the cloud data warehouse to deliver performance and scalability
    - A staging area is used but it is generally just to store raw or partially formatted data while it is loaded into the data warehouse so this becomes a storage solution, rather than a server capable of significant computational work
- A historical or initial load extracts data from a predefined point in time and is usually synonymous with the initial load for a use case
    - Data can be loaded from a predefined point in time as the initial load for a use case
- Incremental load is where a target data repository (data warehouse) is refreshed with the most recent data from the source
    - Depends on the ability to identify changes in your data sources
- Data loading in matilion is performed through Connector components that reside inside orchestration jobs, which are capable of building complicated workflows that bring data from multiple sources into the target data warehouse along with managing objects on those data warehouse
- Database objects are objects in a database used to store or reference data
    - Matillion interacts with database object as part of the etl process
    - Tables: most common objects in relational databases, made up of rows and columns
    - Views: a virtual table based on the results of a sql statement that are saved in the database as named queries
        - The virtual table itself can be queried by the user
        - static views contain data that should be defined in the SELECT and Where clause
        - Dynamic views can be updated dynamically and automatically includes all columns of a chosen table or tables
        - Views are used in security to present a user from accessing specific columns and rows and are used to create data on demand by aggregating results from different tables
    - Indexes are used to find data quickly without having to search for each row
        - Improves database performances by reducing the number of disk accesses needed during a query
        - Created using a few columns: a search key that contains primary keys, and the data reference column which points to where a particular key is found
    - Clusters: two or more servers used to connect to one database / storage system
        - May refer to replicated servers and are made up of thousands of nodes (like a small computer)
        - Individual nodes can be added and replaced as computers grow, where data is spread across multiple nodes so if one fails, other clusters will continue to work
    - Sequences provide users with the ability to create unique values

        - The result of a sequence cannot be used as a primary key
- A schema defines the structure of something: a database schema is how data is organized and constructed (like how a relational database divided into tables)
    - It describes the relationship between its entities
    - In a relational database, the schema defines elements ;ikea tables, fields, relationships, functions, views, indexes, types and other elements
    - When you create your first project in matillion, an environment is created for you (which includes a default schema)
    - Typically stored in a data dictionary (a repository of data information) and is written in text as a set of formulas known as integrity constraints that refer to a graphical illustration of the database structure
    - A physical database schema relates to data storage on a storage system, and its form, like files and indexes
    - A logical database schema defines the local constraints that pertain to the stored data, like views, tables, integrity constraints

Project Menu

Import Export

- Jobs, environments and variables can be exported from on matillion ETL instance to be imported into another
- Right click on Project in top left, click Import or Export
    - Browse your local computer for files to be imported -> select the job
    - This job will be listed in the project menu once imported
- Exported information is saved in JSON format in a file on the machine that the browser window is running on, in whichever default download location is used by your browser and operating system

Manage versions

- Most commonly used to capture and freeze your development at a point in time to designate as live or production versions and to then continue working on default version
- Versioning system creates a copy of all jobs in your current project at that point in time
- All projects have a default version, which is the main working version

API Profiles

- Helps you decide which API connector to use in your matillion ETL instance to provide advice on how to manage your API profiles
- Project -> manage api profiles -> manage query profiles -> plus sign to add a new profile with a name -> configure the newly created profile -> new endpoint -> given endpoint a name -> give it an endpoint URI -> auth table and enable basic authentication -> api user for user parameter and pass in a password -< click send and see if the connection to api was successful
- Now can view metadata that was pulled from the api endpoint

- Tes tot generate a new table -> name of table to explore data within

Password Manager

- Project -> manage passwords -> plus sign to add a new password for components
    - Name, password, encoded encryption type,

Schedules

- Projects -> Manage Schedules -> plus sign to add a new schedule
    - Give it a name, the time zone where the schedule should be coordinated to, type the hours and the minutes, what days of week or month it should run, default version project
    - Select the job the schedule should align with and development environment
    - Click test to check if all parameters have been set up correctly
- When it starts running, can click on the Schedule Task in the task history to explore all if its steps

Search tab: Perform partial search on any term and it will spa jobs, notes, component properties within a project and return matches

Task History

- On the right hand side of the bottom, click on the arrow of the task section to see details of the job while it runs. Click to expand steps of the job.

Task Monitor

- Expand to explore task and see reason for possible failures
- Row count: how many rows have been inserted

ETL Basics

ETL: Extract data from its original source, transform it and load into target database

Environment Variables

- Project -> Management Env Vars -> click the plus side to add one
    - Type the name of the env var as env_
    - Type: numeric, text, datetime
    - Behavior: copied or shared
- Environment section of left hand side: expand development environment
    - Db_source [Default] schema -> Tables -> here new tables will be created and displayed
    - Right click db_dwh to refresh the schema list -> click to open Tables folder
        - Here it can be ensured that tables are listed as part of data warehouse schema
- Views folder -> check the views within the project

Jobs

- Can import jobs from JSON files that are used later on Transformation Jobs
    - When a job is imported into project menu, double click to open it and it will open the canvas with the job -> right click to revalidate the job -> revalidate job (dev)
    - Manage job variables -> delete variable values with the database you want to point it to (data warehouse schema, source schema)
- Right click on canvas to run any imported jobs, right click the job name to delete it
- Job Variables: right click on imported job -> manage job variables and change the name of the variables for the job run
- Jobs can be laid out and run within the canvas, there are two kinds of jobs in matillion

Orchestration Jobs

- An Orchestration job is to complete extracts and logs: where it extracts data from a source and loads it to the cloud data warehouse
    - Loading data from an RDS database running MySQL within AWS
- Right click on default -> add orchestration job -> give a job name -> switch to new job
- Can extract a component to a new job: right click canvas -> extract to new job
- Primarily concerned with DDL statements, loading data from external sources

Shared jobs

- Simply packaged jobs created by users that can be orchestration and/or transformation, used like a component

Transformation jobs

- A Transformation job assumes data is already available in the cloud data warehouse to extract tables and complete the transformations
- Right click default version of project -> add transformation job
- Used for transforming data that already exists within tables

UI: Right click on the canvas to run the job -> run job (development)

Orchestration

Adding the Transformation job into the Orchestration job

And Component

- Waits for all of its inputs to complete before continuing the job: the order the input components run in is irrelevant, since they must all complete before the job continues beyond this component

API Query component

- Lets you query a JSON or XML based API, and loads the resulting data into a table - deals with APIs that contain nested data structures and requires authentication and paging
- Read data from JSON and XML based APIs, where an API is described in a profile definition (which is a collection of XML files describing the API and mapping it to tables, rows, columns)
- Properties:
    - Profile: select the api profile
    - Connection options:
    - Data source: select the table created with api profile
    - Data selection: select the table columns
    - Limit: change the total number of rows coming from the api call
    - Target table:
    - S3 staging area: bucket where staging should happen
    - Authentication method: add username and password again bc the one who created the api profile might be different than the one using the component
    - username/ password: add in your authentication

CloudWatch Publish Component

- Drag component into canvas, then project -> manage env vars to create or use an env var that will be used by the publish component (needs to have a default value)
- Properties:
    - Namespace: must be available on aws account (create if don't have)
    - Metrics: need to map the metrics and select an env var that contains result that should be sent to cloudhealth

Create Table component

- To create or replace a new table of data, created from S3 load generator component
- Properties:
    - Name: to change the name of the component
    - Create/Replace: replace to recreate the table everytime the job is run
    - New Table Name: give the new table a name
- Sometimes the system doesn't guess the correct type for all of the columns so might need to change the size of the column values and the column type
- Right click only the component to run only that component -> run component (dev)

Data Staging components

- Components that can retrieve data from other services and load that data into a table: these components can be called data staging components
- Each data stagers retrieves data via API calls using SQL queries that can filter the data before it reaches the cloud data warehouse, the data is loaded into a table of your choosing and by default, it will destroy and recreate that table before loading the data

Data Transfer component

- Enables users to transfer tiles from a chosen source to a chosen target, can use a number of common network protocols to transfer data to a variety of sources
    - It copies (not moves) the target file
- Properties:
    - Source type: like s3
    - Source url: the location of s3 source bucket (pre created)
    - Target type: select the target dest type
    - Target object name: what object should the data be transferred to
    - Target url: location of s3 test bucket

End Failure component

- Used to mark the overall status of a job as a failure, even if it would otherwise have been successful, and serves to document the expected endpoint of a job
- Connect the false output from one component to the end failure component
- Use to get a failure message if transactions finish with a rollback

End Success component

- Used to mark the overall state of the job as a success, even if certain parts were not successful
    - Serves to document the expected end point of a job
- Connect true output from one component to the end success component

Excel Query component

- Load data stored in an excel sheet into a table by staging the data, so the table is reloaded each time and then you can use transformations to enrich / manage the data in permanent tabes
    - Should not be used to load excel files greater than 100 MB
- Properties:
    - Storage Type: amazon s3
    - Storage URL: s3 bucket where excel is stored, open the file with the bucket
    - Data source: the data source within the excel file (a tab)
    - Data selection:
    - Target table:
    - S3 staging area: select the bucket you use as the staging area

File Iterator

- Loops over matching files in a remote file system by searching for files and running the attached component once for each file found
- Properties:
    - Input data type: the remote file system to search
    - Input data url: the url, including the full path and the file name, that will point to the files to download to the selected staging area

    - Domain: connection domain

Fixed Iterator

- Loops over values of a simple sequence: implements a simple loop over rows of fixed data values and enables you to run an attached component multiple times, each time with different values set on any number of variables
    - Those variables can be reference form the attached component
- Drop it over S3 load component to connect it to the top of the component
- Properties:
    - Variables to Iterate: select the environment variable (the existing variable to iterate, can only be numeric)
    - Iteration Values: type the values the env var

Grid Iterator

- Enables users to loop over values of a simple sequence by looping over the rows of a grid variable (stores tabular data and can hold multiple values of different types) and can run an attached component multiple times (each time with different values set on any number of variables)
- Right click canvas, manage grid variables
- Give it a name, type as test mode to insert columns and their values
    - This creates a new grid variable that maps to different values

If component

- Evaluate an expression involving variables and direct the flow of an orchestration job depending upon whether the expression evaluates to true or false
    - Need to declare variables involved in the expression
- Type if in the component search box and drop next to component, like RDS query
- Properties:
    - Condition: add the environment variable , leave qualifier as is but change comparator to greater than or equal to with a default value
- Connect true output from if component to And component

Loop iterator

- Lets users loop over values of a simple sequence, t is a simple for loop that enables you to run an attached component multiple times and each time with a unique value of an iteration variable
    - That variable can be referenced from the attached component
- A loop from x to y, where x and y depend on the component and its configuration
    - User can iterate between two numeric values that contain a function that relies on the current value to do something (no need to create individual jobs)

Or component

- Waits for any of its inputs to complete before continuing the job
- Once any of the inputs complete, the tasks defined after the OR comp are added to the job

Python Script component

- Run a python script that is executed in process by an interpreter of the users choice
- Properties:
    - Script: add or write a python script

Run Orchestration component

- Run a job to completion so that after the orchestration job completes, other orchestration components can be lined to this one depending upon success or failure
- Use this to better organize complicated orch job flows by breaking the into simpler pieces

RDS Query component

- RDS Query Component: Use to create a new table in source schema with data stored in an RDS Database. Type rds into component search bar, select RDS Query and drop it on canvas close to Start Component
- Properties:
    - Name: give the component a name
    - Database Type: change the database type to mysql
    - RDS Endpoint: public DB link
    - Database Name: give database name (source DB already created)
    - Username: user name to the database
    - Password: click on option to store in component
        - Can also use password manager and select the password label that was stored in the password
    - JDBC Options: need to add if latest version
        - useSSL: false
    - SQL Query: query to get the data from the RDS database
        - Delete default syntax, write query, click sample to explore the results of this query
    - Target Table: name for destination table
    - S3 Staging Area: bucket used by matillion to copy data before loading it into the data warehouse
- Export Tab -> edit -> plus sign to add a mapping from the components internal variable to environment variables
    - Row count from source menu and the name of the env var

SQL Component

- Write own complete SQL select statement to run queries on any tables in the project
    - Can link one or more inputs into the SQL component to indicate which data sources are used within the query
- Properties:

    - Query: add a sql query and click run to test it

SQS Message component

- Post a message to an SQS
- Properties:
    - Queue name: the name of the queue to write to (must be defined)\
    - Message: the name of the message that should be sent to the queue

Start component

- Where the job will begin when it is run, should connect components from here

S3 Load Generator component

- Load delimited data from public objects in an S3 bucket
- Takes the form of a wizard that lets you load and view files on the ly, altering load component properties and observing their effects without the need for separate Transformation job
    - Can guess the schema of a table
- Type s3 to find the component -> drag close to the start component
- Provide s3 bucket path from an aws account that will contain desired data”
    - Select which file within the bucket you want, select the compression mode if it has
    - Get Sample to explore the data contained in the file
    - Guess Schema to get matillion to infer the data types needed for your destination using the first 50 rows
    - Select Empty Field as Null
- This creates two components: Create Table and S3 Load
    - Connect Start component to create table component

S3 Load component

- Load data into an existing table from objects stored in s3 and requires working aws credentials, with read access to the bucket containing the source data files (through IAM role)
- Generated from S3 Load Generator component
- Uses an empty table with schema from Create Table component and load data in there from S3
- Properties:
    - Pattern: use this if you want to concatenate the file name to each of the fixed env vars: training_s3_flights_${env_flights_year}.gz
    - Target Table: select the table created from the Create Table component

Table Iterator

- Loop over rows of data within a table or view and run another component for each row
- Implements a loop over rows of data in an existing table, enables you to run attache component multiple times (each time with different va;ues set on any number of variables taken from columns of that table)

Transactions

- Use to control the output of orchestration job
- Components -> Transactions -> Begin, Commit, Rollback
- Begin Component should be connected to the start component: it indicates the point where the transaction starts
    - Drag orchestration job close to the begin component to connect them
    - Starts a new transaction within the database, can help make multiple changes to a database as a single, logical unit of work
- Commit Component: ends a transaction within the database and makes all changes made since the latest begin component visible to other database users
    - Help make multiple changes to a database as a single, logical unit of work
- Rollback Component: failure output from orchestration job should be connected to rollback to destroy / rollback all changes made by orch job
    - Ends a transaction within the database and undoes any changes made since the latest begin component
    - Changes are never visible to other database users

Transformation

Aggregate Component

- Groups together multiple input rows into a single output row: input columns can be added to the groupings, or have an aggregation applied to them
    - Default output names are chosen by combining the source column name and the aggregation type
- Type agg to find the component, use to group data
- Properties:
    - Name: change the name of the aggregate component
    - Groupings: click the columns to group by
    - Aggregations: how should the column data be aggregated

Calculator Component

- Adds new columns by performing calculations: each input rows produces one output row and any pre existing columns that share a name with a created column will be overwritten
    - By default, all input columns are mapped into output columns of the same name
    - Can use the expression editor to add new columns with a user defined valculation
- Properties:
    - Name: give the calc a name
    - Calculations: click the plus sign to add a new expression
        - round(case when ‘distince_type’=’km’ then ‘distince’ else ‘distance’*’factor/ end)
        - round(case when ‘distancE_type’=’miles’ then ‘distance’ else ‘distance’*’factor’ end)

        - cast(date_part(year, ‘flight_date’), as integer)

Convert Type component

- Convert the data types of the input flow (always better to change the source data so that is already has the correct types)
- Metadata tab: see the current data types for the columns
- Properties:
    - Conversions: click the column you want to convert and the data type you want to convert it to

Detect Changes component

- Users can scan two separate but similar tables, and insert a new column detailing if data has been inserted, deleted, changes or even if the data is unchanged
    - Any rows with key columns that contain null values will be ignored and null comparison values are considered equal
- Identify differences between new and existing data
- Properties:
    - Master table: select the main table to detect : select a master table from the two inputs (this table is the one treated as default in the comparison with the second table)
    - Master key: change the property of master key, where maser key is the common column between the inputs of the detect changes component (select the key columns to join the two tables on, both columns must appear in the tables)
    - Compare columns: select the columns that will be checked for changes, both columns must appear in both tables
    - Output column mapping: each col will have master and compare version
        - Select input columns to map to output ames

Filter component

- Filter rows from the input to [pass a subset of rows to the next component based on a set of conditions
- Properties:
    - Name: give the component a name
    - Filter Condition: find the input column to use, qualifier as not or is, comparator as null
        - This will filter out null values
- Sample: row count icon to see the total number of rows after the filter

Fixed Flow component

- Allows you to generate lines of fixed input or input from from variables, useful for simple static mappings
- Use to create static data related to the measures
- Properties:
    - Name: change the name of the component

    - Columns: add a new column with name, type, give the factor a size and give it decimal places
    - Values: add a value name, target type and factor that will help do te conversion

Generate Sequence component

- Generates a sequence of rows from a starting value to an end value with a user defined increment: can be used to generate empty rows of data from nothing, for use in a job
- Find in component, drag to canvas
- Properties:
    - Name: change the name of the component
    - Starting value: give it a starting value
    - End value: give is an end value, can be a job or env variable ${job or env}
- Sample:
    - Click data to see the sequence that was generated

Join component

- Joins two or more input flows into a single output - defined using expressions which can be calculated from the input columns
- Type join to find the Join component: connect input tables to join component from Table Input component or the rank component
- Properties:
    - Main Table: select the main table you want to join
    - Main Table Alias: add alias like main
    - Joins: specify the kind of join you want to do
        - Type alias of the second joining table and click the left join if you want to keep all the rows from the main table, even if they don't match with the second table
    - Join Expressions: click on arrow next to the column that you want to join on
        - Column 1 in main table = column 1 in second table
    - Output Columns: Add all to include all columns from both tables are part of the output table, can delete a column if don't want duplicates

Map Values component

- Replaces one value with another in the dataset based on a specified condition: as the data is passed through the component, rules are applied to specific columns to replace one value with another (used to introduce mappings into the data)
- Properties:
    - Name: give it a name
    - Value map: select the mapping using the grid variable
    - Other: specify which value will be used when none of the values from the value map property match

Multi table input

- Read chosen columns from an input table into the job: reads data from many input tables based on filtering all available input tables matching a pattern
- Type multi to find the component in the search box
- Properties:
    - Name: change the name of the component
    - Pattern: use to find all the tables that follow the wild card component like view_flights_20%
    - Columns: select which cols you want
- Sample:
    - Click row count to see how many rows are obtained from the unite
- Results are the same as the unite component

Rank component

- Able to determine a rank of a value within a group of values
- Properties:
    - Name: change the name of the component
    - Partition data: columns for grouping the data
    - Ordering within partitions: find the actual column that you don't want partitions of
    - Functions: add a new function like Dense_rank and the output column
- Sample: click data to get a sample
    - This will show that a new column has been created for ranking based on airtime

Rename component

- change field names within the data flow
- By default, all input columns are mapped into output columns of the same name
- Any field can be dropped from the list or their names can be updated
- Properties:
    - Column mapping:

Rewrite Table component

- Uses input connection and writes the results out to a new external table
- Properties:
    - Name: give the component a name
    - Schema: change schema since don’t want a dimension being created in default schema (select an environment default schema)
        - Select use variable nd find env_ to find env vrs, select the env var create to replace the schema with the default value contained in the variable
    - Target Table: name of the resulting table

Table Input component

- Read chosen columns from an input table or view into the job
- Type input to find Table Input component

- Properties:
    - Target Table: select the table you want to input to the transformation job
    - Column Names: select the columns of the input table
- Drag tables from Environments -> db_source[Default] tables to the canvas

Table Update component

- Update a target table with a set of input rows: the rows to update are based on matching keys so it is important that the keys uniquely identify the rows, and that the keys are not null
- Properties:
    - Schema: use variable to find and use an existing env var
    - Target table: the table the update will go to\
    - Join expression: specify the expression
    - When matched: specify which kind of operation should be made as part of the update
    - Update mapping:

Table Output component

- Empowers users to write an input data flow out to an existing output table where successful validation of this component ensures: target table exists and target columns have been found
- Data is only written to the table when the transformation job containing table output component is actually run
- Type out to find table output component
- Properties:
    - Schema: select the schema from environment
    - Target table: select the summary fact table
    - Column mapping: system will try to map input to output columns to do a fuzzy mapping but might need to change the mapping
    - Truncate: select truncate since want an empty table each time the job is run
    -

Transpose Columns component

- Map sets of input columns into new output columns to perform an unpivot on the data
    - Re shapes data by outputting multiple rows for each individual input row
    - Each set of input columns are mapped to an output column, the output rows are labeled to determine which column the value originated from
- Properties:
    - Ordinary columns:
    - Row label name: add a label for the rows
    - Output columns: what name should the output column name
    - Column to row mapping: select text mode to include provided mapping

Unite component

- Combine all the rows from two or more input flows into a single output flow

    - The input flows should be very similar for this component to be useful, although slight differences in the input schema can be accommodate by using the Cast Types property
- Type unite to find the component and use mouse to connect the view or tables to unite to concatenate them
- Sample tab:
    - Row_count and refresh to see the total number of rows after unite

Windows Calculation component

- Allows the user to set up a calculator using a window function, which allow users to query tables for a partition of a dataset (termed a window)
- Properties:
    - Name: change the name of component
    - Partition data: chose the data for how to partition the data
    - Ordering within partitions:
    - Lower bound: what kind of preceding
    - Upper bound: modify the property
    - Functions: what kind of window function, input column and output column