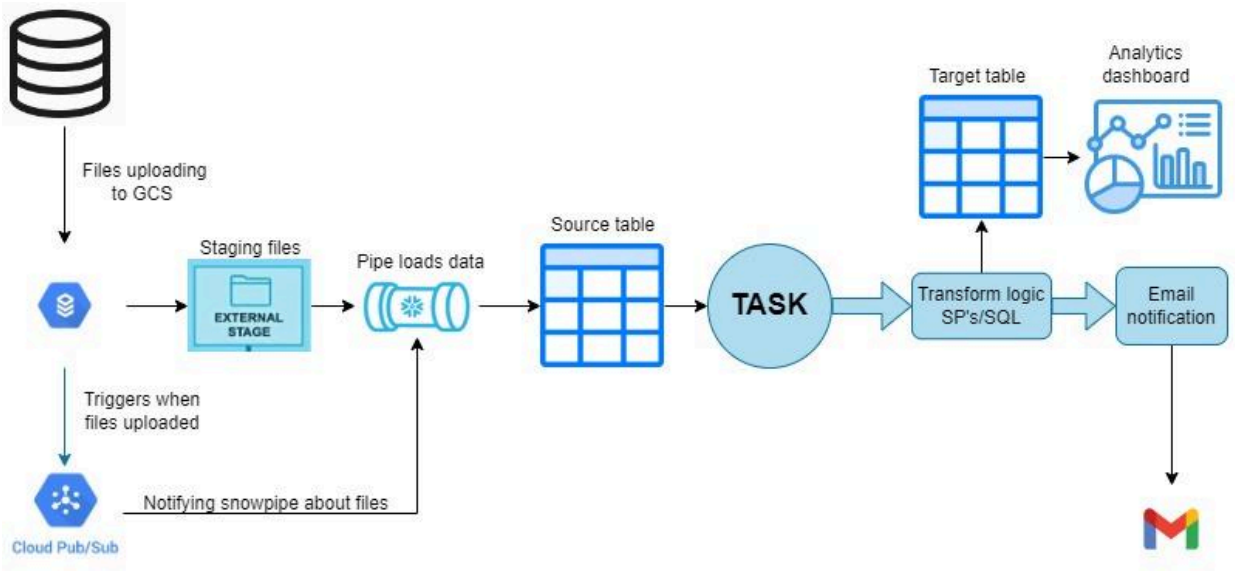
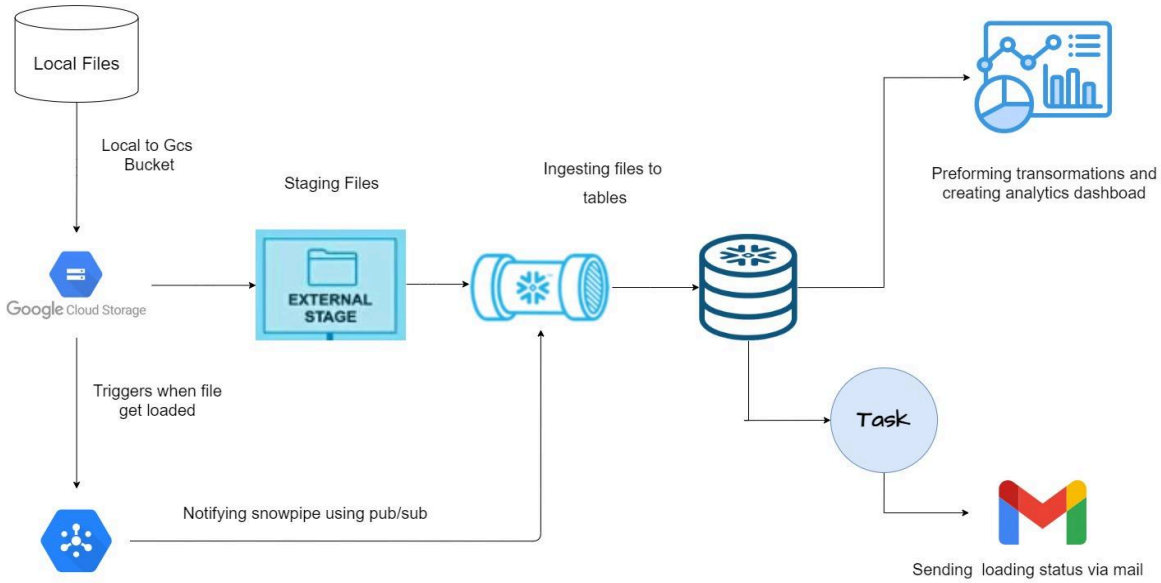
SNOWFLAKE ELT NOTIFICATION USE CASE





Initial Setup Requirements

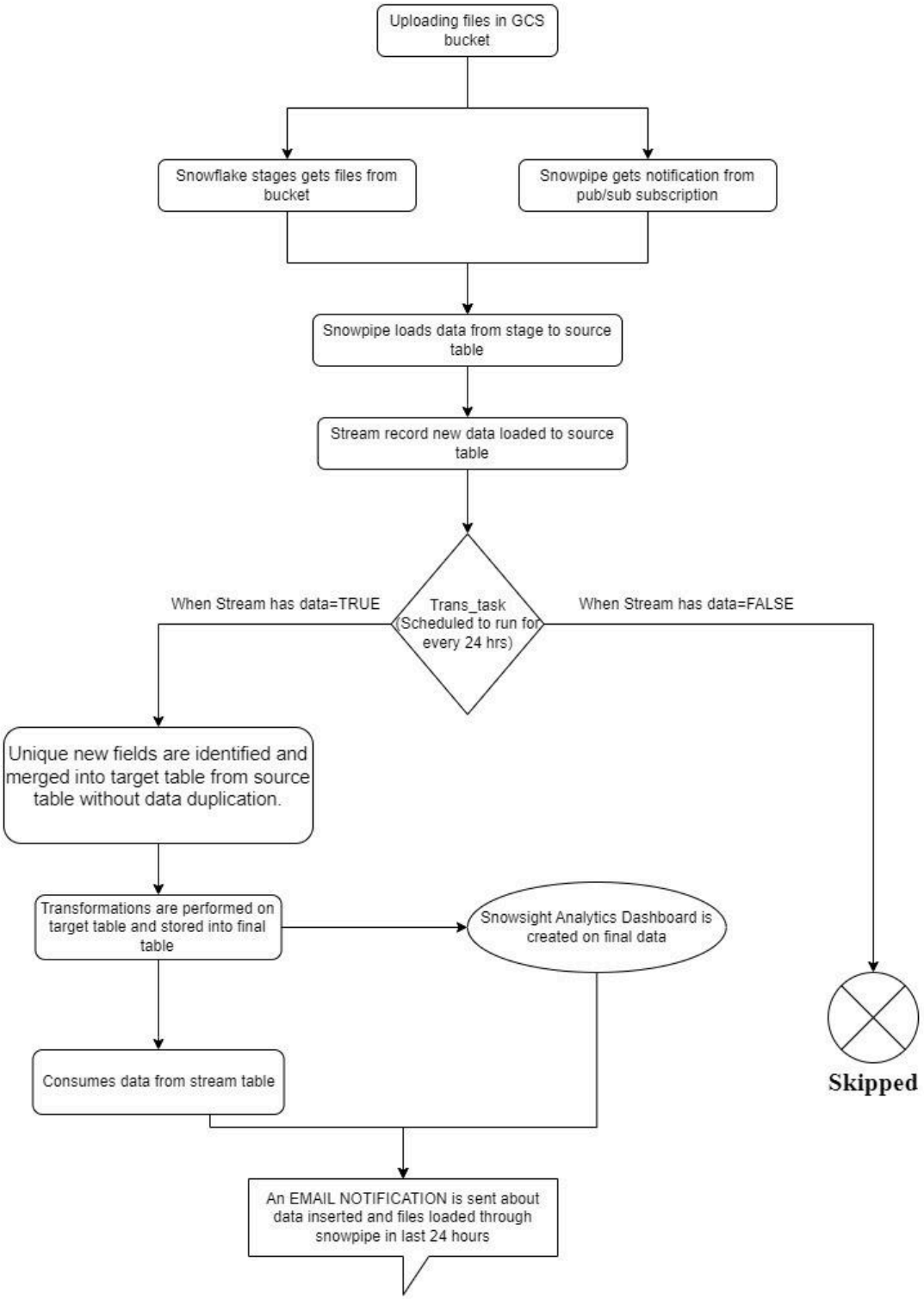
- **Source Table: liquor_sales_src**
  Create a source table which is used for loading data from a GCS bucket.

- **Target Table: liquor_sales_target**
  Create a target table to transform data from the source table .

- **Stream : trans_stream**
  A Stream is created on a table for change data capture(CDC).Streams records information about the DML changes happening on the table.
  Create a stream on the source table which records the newly added data of the source table.

```
CREATE OR REPLACE STREAM_NAME ON TABLE TABLE_NAME
```

Functional FlowChart



Snowpipe

Snowpipe enables loading data from files as soon as they're available in a stage making it available to users within minutes rather than manually executing.It executes COPY statements on a schedule to load larger batches.

A pipe is a named, first-class Snowflake object that contains a COPY statement used by Snowpipe. The COPY statement identifies the source location of the data files (i.e., a stage) and a target table. All data types are supported, including semi-structured data types such as JSON and Avro.
Here we are using **Automating Snowpipe using Cloud Messaging** for detecting the staged files.

https://docs.snowflake.com/en/user-guide/data-load-snowpipe-auto-gcs

Automating Snowpipe for Google Cloud Storage

This topic provides instructions for triggering Snowpipe data loads automatically using google cloud pub/sub messages for Google Cloud Storage (GCS) events.

In this use case, we want to load data from CSV files in the existing buckets in google cloud storage to a source table in snowflake.

Cloud Storage Integration in Snowflake

- The Storage Integration includes information about your cloud storage provider, such as the provider type (e.g., S3, Azure, GCS), the location of the storage, and access credentials.

```
CREATE or replace STORAGE INTEGRATION gcs_snowpipe
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = GCS
  ENABLED = TRUE
  STORAGE_ALLOWED_LOCATIONS = ('gcs://gcs_snowpipe')
```

- **Retrieve the Cloud Storage Service Account for your Snowflake Account**: Execute the DESCRIBE INTEGRATION command to retrieve the ID for the Cloud Storage service account that was created automatically for your Snowflake account.

```
DESC STORAGE INTEGRATION <integration_name>;
```

```
+-----------------------------+---------------+------------------------------------------------
| property                    | property_type | property_value
+-----------------------------+---------------+------------------------------------------------
| ENABLED                     | Boolean       | true
| STORAGE_ALLOWED_LOCATIONS   | List          | gcs://mybucket1/path1/,gcs://mybucket2/path2/
| STORAGE_BLOCKED_LOCATIONS   | List          | gcs://mybucket1/path1/sensitivedata/,gcs://myb
| STORAGE_GCP_SERVICE_ACCOUNT | String        | service-account-id@project1-123456.iam.gservic
+-----------------------------+---------------+------------------------------------------------
```

You need to copy the property_value of property “STORAGE_GCP_SERVICE_ACCOUNT”

- **Grant the Service Account Permissions to Access Bucket Objects**: Log into the Google Cloud Platform Console as a project editor and create a custom role with necessary permissions. Here we have created a **custom role** with data loading permissions mentioned below.

Data loading only

- `storage.buckets.get`
- `storage.objects.get`
- `storage.objects.list`

Configuring Access to GCS Buckets

In Google Cloud Platform Console,choose **Cloud Storage** and Select a bucket to configure for access.

- Click **SHOW INFO PANEL** and Click the **ADD PRINCIPAL** button.
- In the **New principals** field, Paste the service account which you copied in the property_value .
- From the **Select a role** dropdown, select the custom role which you created in the above process.
- Click the **Save** button. The service account name is added to the **Storage Object Viewer** role dropdown in the information panel

Creating the Pub/Sub Topic in GCP

Create the topic and enable it to listen for activity in the specified GCS bucket.
Execute the below command using CLOUD SHELL

```
$ gsutil notification create -t <topic> -f json gs://<bucket-name> -e OBJECT_FINALIZE
```

Where:

- `<topic>` is the name for the topic.
- `<bucket-name>` is the name of your GCS bucket.

Creating the Pub/Sub Subscription:

Create a subscription with pull delivery to the Pub/Sub topic using the Cloud Console.

- In Google Cloud Platform Console, search **PUB/SUB** and select the topic which you created in the above process.
- For that topic create a subscription with default pull delivery.

Retrieving the Pub/Sub Subscription ID:

- Log into the Google Cloud Platform Console as a project editor.
- From the home dashboard, choose Big Data » Pub/Sub » Subscriptions.
- Copy the ID in the Subscription ID column for the topic subscription.

Create a Notification Integration in Snowflake

The notification integration references your Pub/Sub subscription. Snowflake associates the notification integration with a GCS service account created for your account.

```
CREATE NOTIFICATION INTEGRATION <integration_name>
  TYPE = QUEUE
  NOTIFICATION_PROVIDER = GCP_PUBSUB
  ENABLED = true
  GCP_PUBSUB_SUBSCRIPTION_NAME = '<subscription_id>';
```

Where:

- `<integration_name>` is the name of the new integration.
- `<subscription_id>` is the subscription name you recorded in Retrieving the Pub/Sub Subscription ID.

After creating , execute the following command to retrieve the Snowflake service account ID

```
DESC NOTIFICATION INTEGRATION <integration_name>;
```

Record the service account name in the GCP_PUBSUB_SERVICE_ACCOUNT column, which has the following format:
“ **<service_account>@<project_id>.iam.gserviceaccount.com**”

Grant Snowflake Access to the Pub/Sub Subscription

- Log into the Google Cloud Platform Console as a project editor.
- From the home dashboard, choose **Big Data » Pub/Sub » Subscriptions**.
- Select the **subscription** to configure for access.
- Click **SHOW INFO PANEL** in the upper-right corner. The information panel for the subscription slides out.
- Click the **ADD PRINCIPAL** button.
- In the **New principals** field, search for the service account name you recorded.
- From the Select a **role** dropdown, select **Pub/Sub** Subscriber.
- Click the **Save** button. The service account name is added to the **Pub/Sub Subscriber** role dropdown in the information panel.

Create a File format and Stage

- A file format is a configuration object that defines how data files are formatted and structured when loaded into the Snowflake data warehouse.

```
CREATE OR REPLACE FILE FORMAT my_csv_file_format
TYPE = CSV
FIELD_DELIMITER = ','
ERROR_ON_COLUMN_COUNT_MISMATCH = FALSE
SKIP_HEADER = 1;
```

- Create an external stage that references your GCS bucket using the CREATE STAGE command. Snowflake reads your staged data files into the external table metadata.

```
CREATE or replace STAGE mystage
URL='gcs://gcs_snowpipe/'
STORAGE_INTEGRATION = gcs_snowpipe
file_format = my_file_format;
```

URL : Refers to GCS bucket
STORAGE_INTEGRATION : Storage integration which we have created earlier.

Create a Pipe with Auto-Ingest Enabled

Create a pipe using the **CREATE PIPE** command.The pipe defines the **COPY INTO** statement used by Snowpipe to load data from the ingestion queue into the target table.

Here, we created a pipe which loads data from files staged in an external stage named **mystage** into a destination table named **liquor_sales_src** using a notification integration.

```
CREATE or replace PIPE liquorsales_pipe
AUTO_INGEST = true
INTEGRATION = snowpipe_notification
AS
COPY INTO liquor_sales_src
from @mystage
on_error=CONTINUE;
```

Stored Procedures

A stored procedure contains logic you write so you can call it from SQL. A stored procedure's logic typically performs database operations by executing SQL statements.

Snowflake supports different languages for writing stored procedures like Java, Javascript, Python,Scala and SQL . Snowpark API library is used for some of the languages to be supported.

In this use case we have used Python and Sql majorly for creating stored procedures.There are 4 stored procedures we have created for performing different operations.

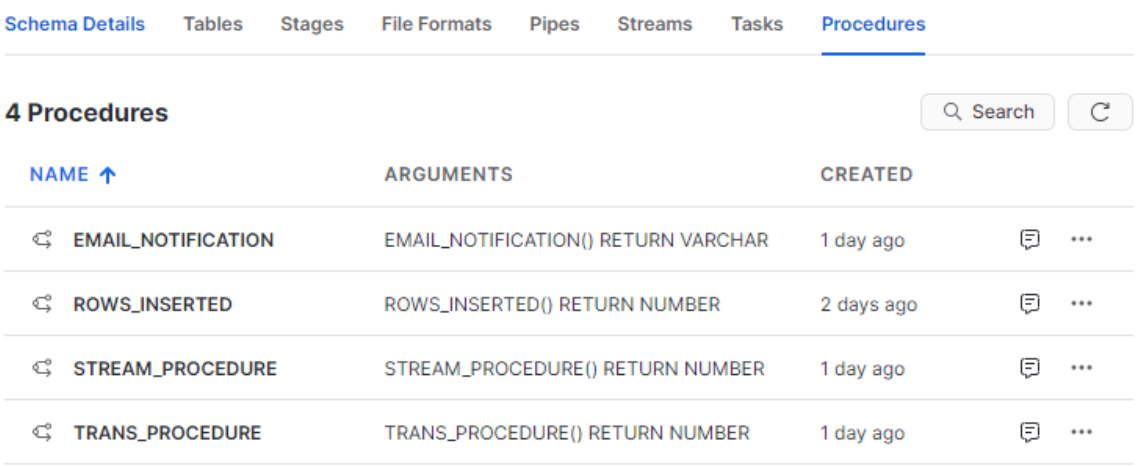
1. **rows_inserted( )** : This procedure is written in SQL language. It performs merge sql operations for merging the newly loaded records from source table to target table without any data duplications and returns the row count of newly inserted records.

2. **trans_procedure( ) :**The main function of this procedure is to perform transformations on the target table data and store the results in the final table which is further used for creating snowsight analytics dashboards. SQL language is used for writing this procedure.

3. **stream_procedure( ) :** This procedure is used for consuming the data from stream and store them in liquor_stream_consume table for reference.It is written in SQL language

4. **email_notification( ) :** This is the core stored procedure which is written in python language. All the packages required are imported using SNOWPARK API library. It makes a call to all the above 3 procedures and collects the results. Load history of files from the last 24 hours are also collected. All the results are stored as a pandas dataframe and converted into html format.

   Using the snowflake inbuilt **system$send_email** feature a mail is sent to recipients whenever the procedure is called showing load status.



Creating a task for performing transformations and sending status mail

A Snowflake task in simple terms is a scheduler that can help you to schedule a single SQL or a stored procedure. A task can be very useful when combined with streams to make an end-to-end data pipeline. When you create the task, it always gets created in **suspended** mode. So that means you have to explicitly resume the task to get things moving.

In this use case, we have created a task which runs for every 24 hours, checks for the condition whether the stream has any new data present and makes a call to a stored procedure when the condition is satisfied.

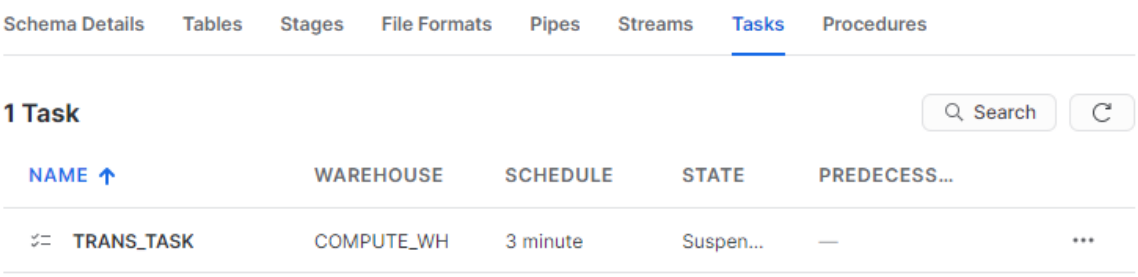
The task in this use case is named **trans_task** which checks for data in stream named **trans_stream** and calls the stored procedure named **email_notification( ).**

The trans_task is responsible for all the procedure calls and sending mail notifications.In this,the checks for the condition **system$stream_has_data('trans_stream')** where the function checking for stream having data is a snowflake inbuilt function.

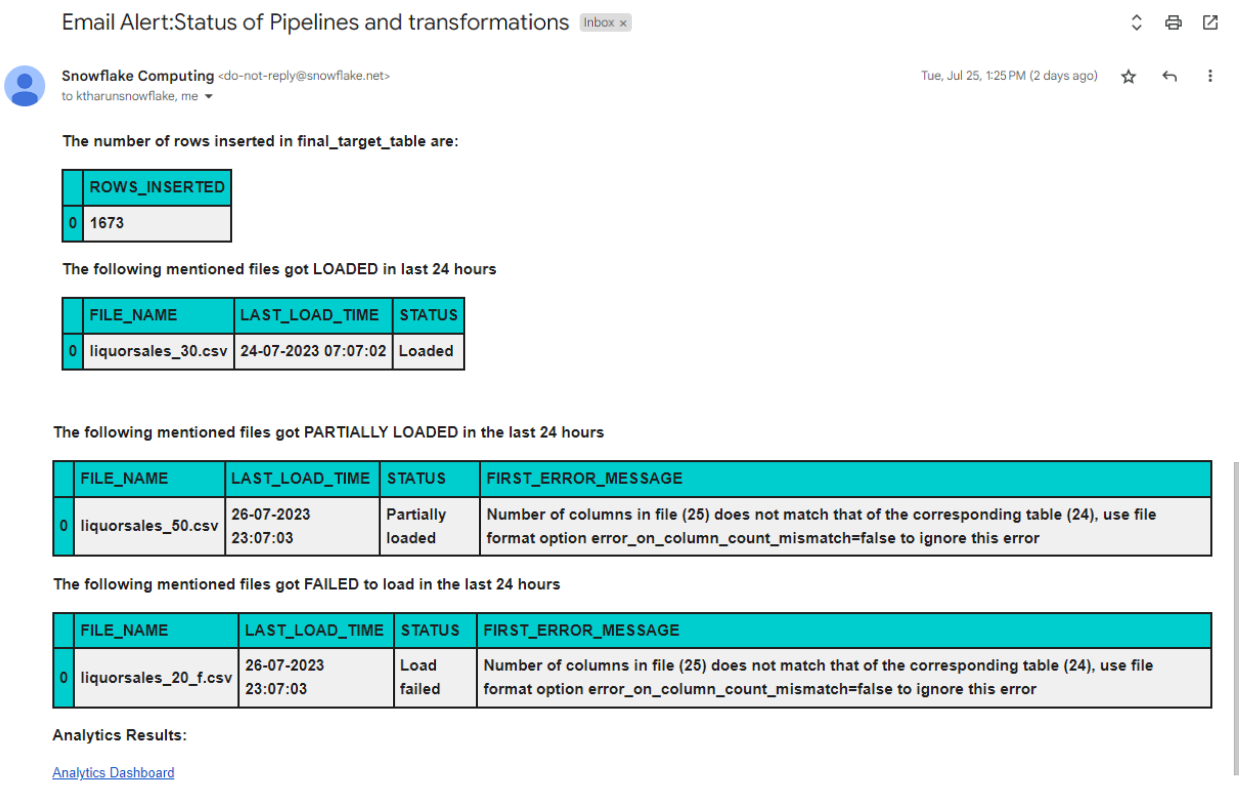
These are the steps performed whenever a task condition is true:

- Unique new fields are identified and merged into the target table from the source table without data duplication.
- Transformations are performed on the target table and stored into the final table.
- Consumes data from stream table.
- An email notification is sent about data inserted and files loaded through snowpipe in the last 24 hours.

The Task runs as per the schedule, if there is no data in stream the task is **skipped.**



- The email notification is sent in this format whenever the procedure is called on the schedule basis.



Email Alert:Status of Pipelines and transformations Inbox ×

Snowflake Computing <do-not-reply@snowflake.net>
to ktharunsnowflake, me

Tue, Jul 25, 1:25 PM (2 days ago)

The number of rows inserted in final_target_table are:

	ROWS_INSERTED
0	1673

The following mentioned files got LOADED in last 24 hours

	FILE_NAME	LAST_LOAD_TIME	STATUS
0	liquorsales_30.csv	24-07-2023 07:07:02	Loaded

The following mentioned files got PARTIALLY LOADED in the last 24 hours

	FILE_NAME	LAST_LOAD_TIME	STATUS	FIRST_ERROR_MESSAGE
0	liquorsales_50.csv	26-07-2023 23:07:03	Partially loaded	Number of columns in file (25) does not match that of the corresponding table (24), use file format option error_on_column_count_mismatch=false to ignore this error

The following mentioned files got FAILED to load in the last 24 hours

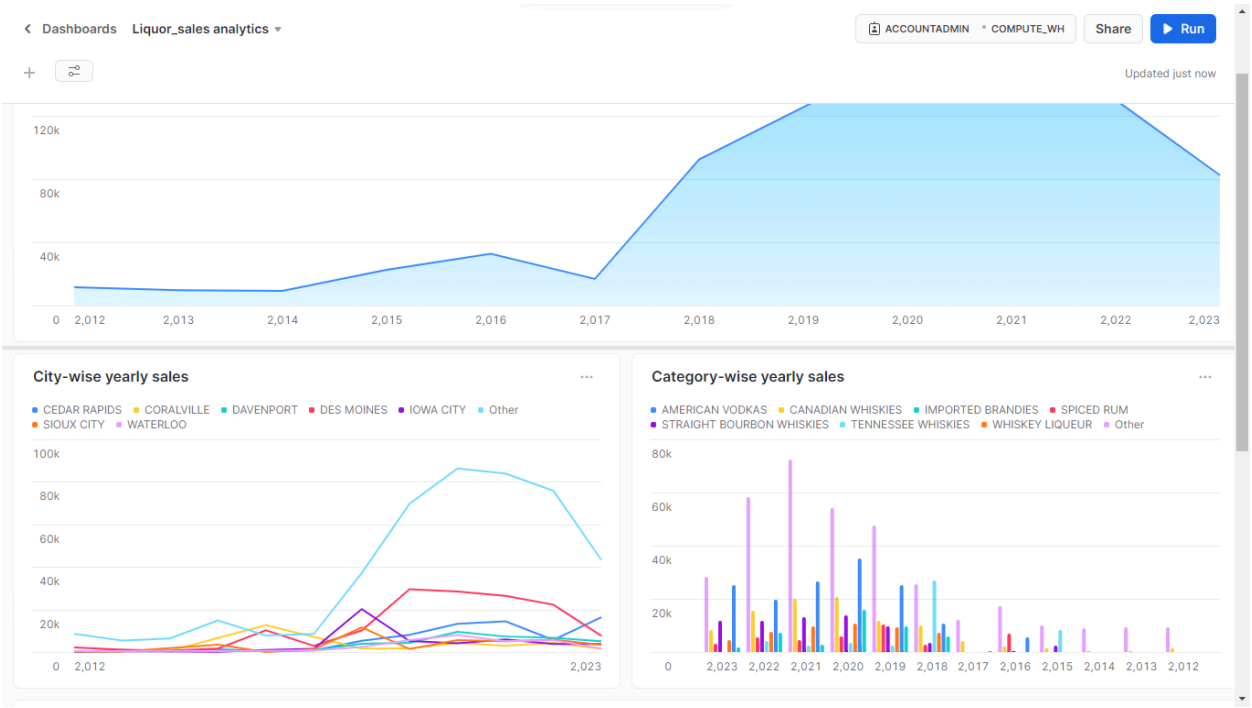
	FILE_NAME	LAST_LOAD_TIME	STATUS	FIRST_ERROR_MESSAGE
0	liquorsales_20_f.csv	26-07-2023 23:07:03	Load failed	Number of columns in file (25) does not match that of the corresponding table (24), use file format option error_on_column_count_mismatch=false to ignore this error

Analytics Results:

Analytics Dashboard

Snowsight Analytics Dashboard

Dashboards are flexible collections of charts arranged as tiles. The charts are generated by query results and can be customized. In this use case, we have created a dashboard on the final transformed table. This dashboard showcases the analytics from the **sales_final** table.



References:

https://docs.snowflake.com/en/
https://docs.snowflake.com/en/sql-reference/sql/create-pipe
https://docs.snowflake.com/en/user-guide/data-load-snowpipe-auto-gcs
https://docs.snowflake.com/en/user-guide/tasks-intro
https://docs.snowflake.com/en/developer-guide/snowpark/index
https://docs.snowflake.com/en/developer-guide/snowpark/python/creating-sprocs