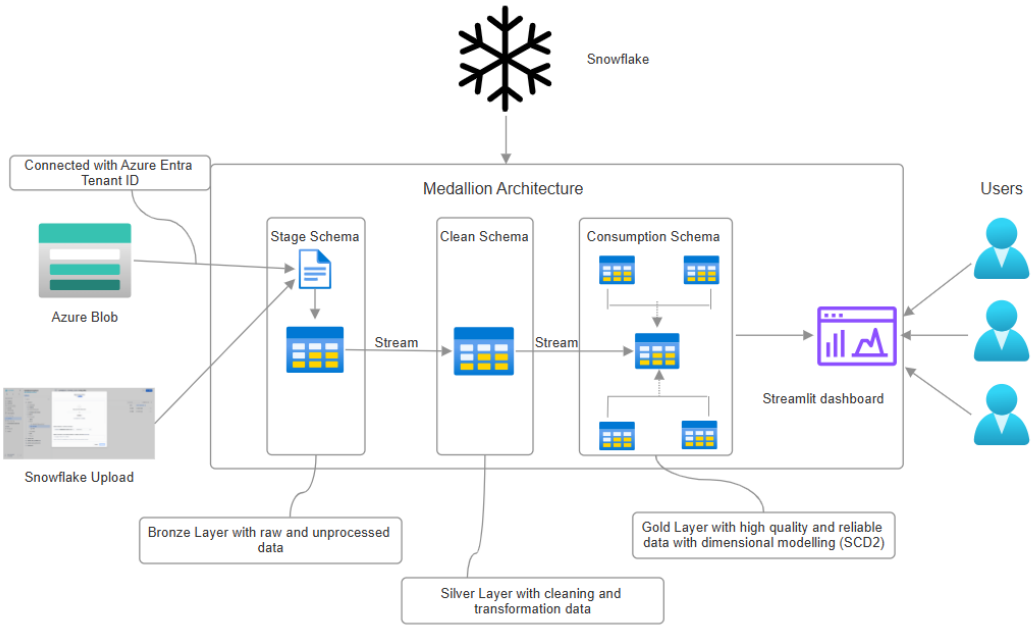
##Veer Karadia

Data Engineering

Snowflake end to end Data Engineering with Swiggy Data



End to End Data Flow

Aim

Ever wondered how food delivery and quick-commerce platforms like Swiggy or Zomato handle massive amounts of orders in real time? In this project, I will demonstrate an end-to-end data pipeline built on Azure and Snowflake for a real-life food aggregator, leveraging the Medallion architecture to ensure scalability, governance, and flexibility across ingestion, transformation, and consumption layers. We will learn how data moves from Azure Blob→ Stage schema→ Clean schema→ Consumption schema→ insights, with a step-by-step hands-on experience.

What I have Learned:

- Understand the food order process flow and design an OLTP source ER diagram.
- Build a 3-layer Snowflake data warehouse (Stage → Clean → Consumption), also known as Bronze → Silver → Gold layers.
- Learn how Azure Event Grid + Storage Queue trigger Snowpipe for automatic data loading.
- Implement Streams & Tasks to handle real-time delta changes.
- Apply Slowly Changing Dimension (SCD Type 2) in the dimensional model.
- Create an interactive Streamlit dashboard for KPIs and insights.

Design Highlights

- Stage Folder Structure & Snowflake Streams:

```
 1. Created database and schemas (Stage, Clean, Consumption), file formats, and external stages to connect Snowflake with Azure Blob Storage and internal stage with two folders: initial load and delta for incremental processing.
 2. Stage schema stores raw data for entities such as Customer, Location, Customer Address, Restaurant, Menu, Orders, Order Items, Delivery Agent, Delivery, and Date.
 3. Data validation: Initial and delta CSV files were manually processed across Stage → Clean → Consumption schemas to verify pipeline correctness before Snowpipe automation.
 4. Streams: Snowflake streams on Stage and Clean schemas capture incremental changes, enabling event-driven propagation to the next schema.
 5. Snowpipe: Implemented at the Stage schema level to automatically ingest data from Azure Blob Storage, using Event Grid and Storage Queue notifications to trigger the pipeline for multiple entities.
```

- Azure Cloud Setup & Integration:

```
1. Resource Creation: Provisioned Azure Storage Account and Blob containers with hierarchical folder structure..
2. Identity & Access Management: Secured Snowflake integration with Microsoft Entra ID (Azure AD), granting least-privilege access (Reader at container/folder level and Viewer role for monitoring).
3. Event-Driven Pipeline: Built an event-driven ingestion pipeline, where Event Grid detects new blobs and notifies Snowpipe via Storage Queue.
```

    - Automation Using Schedulers:
        1. Task 1: Processes new files from Azure Blob → Stage schema (raw tables).
        2. Task 2: Merges data from Stage → Clean → Consumption schemas, applying transformations, validations, and business rules.
        3. Fully automated, near real-time pipeline with minimal manual intervention.

- Data Transformation & Warehouse Design:

```
 1. Stage → Clean: Applied data cleansing, deduplication, type casting, and enrichment. Implemented business rules such as state normalization, city tier classification, and surrogate key generation.
 2. Clean → Consumption: Implemented dimensional modeling with SCD Type 2 to track historical changes for each entity. A custom hash key is generated for every record to detect changes efficiently and maintain historical versions.
```

- Analytics & Visualizationn:

```
 1. Built interactive dashboards using Streamlit with Python, leveraging curated Snowflake views created from Snowpipe-ingested data to perform near real-time analysis.
```

Key Concepts I was able to learn

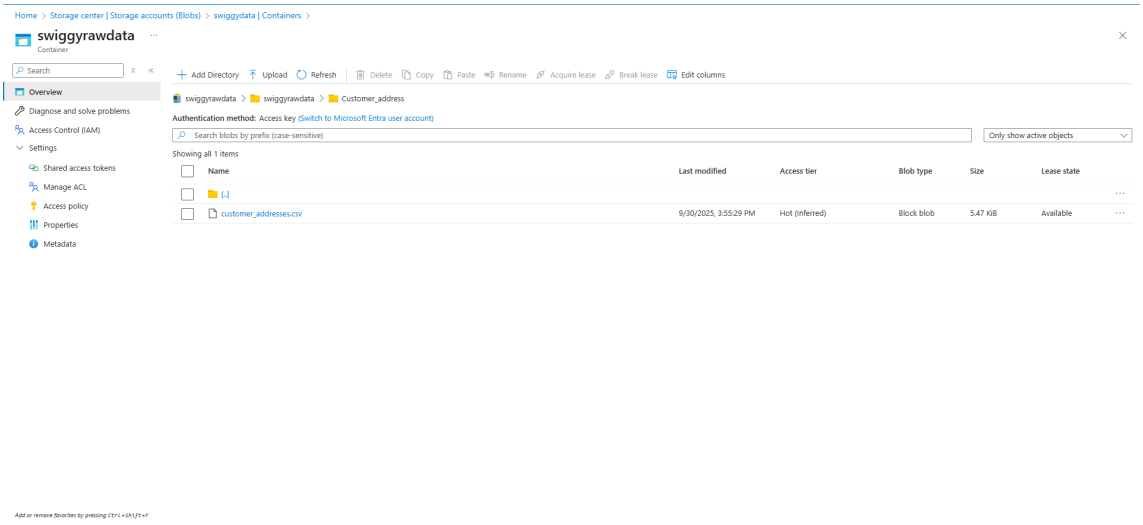
- How can I load data into Snowflake directly from Azure Blob Storage?

- How does Event Grid + Snowpipe enable near real-time ingestion?
- What’s the role of Streams in capturing incremental changes (delta loads)?
- How do I structure my Stage, Clean, and Consumption schemas for ELT pipelines?
- How to query files in the Stage using $ notation before ingestion?
- How do COPY INTO commands load CSV files into Snowflake tables?
- How can I implement SCD Type 2 for handling historical changes in dimension data?
- How do I secure my pipeline using Azure Entra ID + RBAC?
- How to visualize KPIs and insights with a Streamlit dashboard connected to Snowflake?
- How to set up an automated scheduler (Tasks) for data refresh?
- What best practices ensure data quality, governance, and scalability in the pipeline?

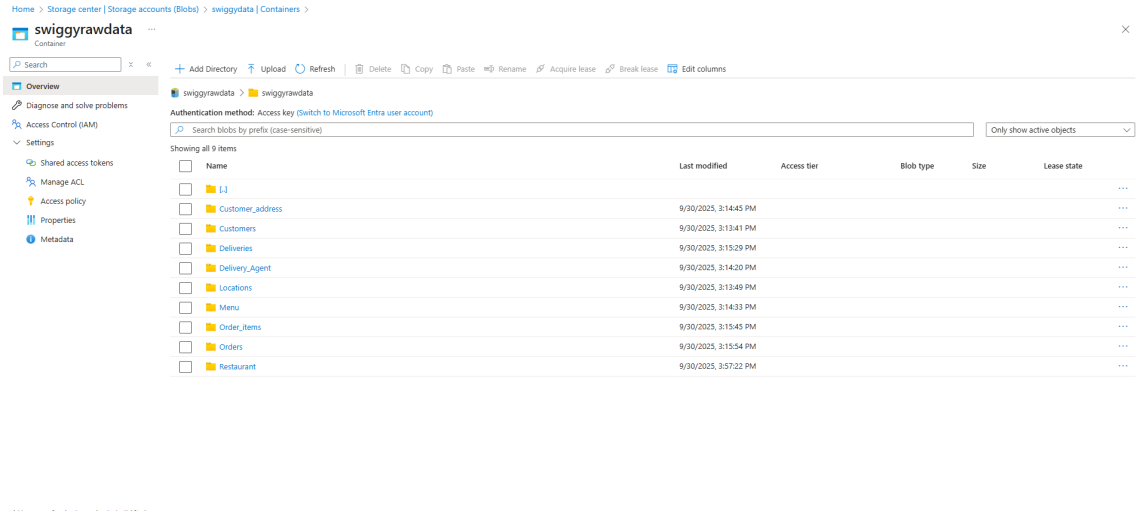
Azure Cloud

The images illustrate the Azure setup: the Storage Account hierarchy and folders, the Event Grid triggering when new records are added, latency and other metrics tracking, and the Storage Queue receiving notifications.

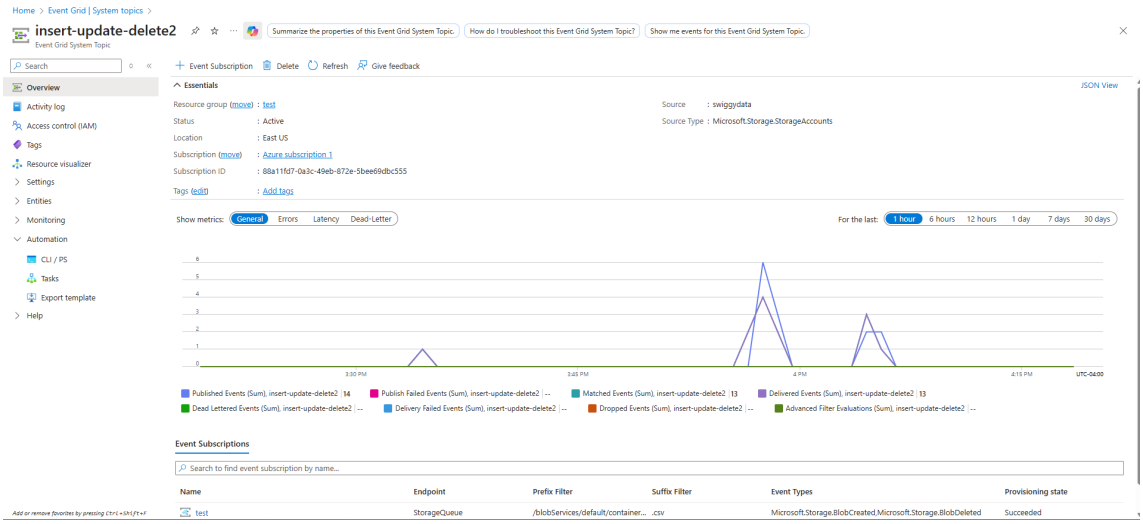
Customer folder with csv file:



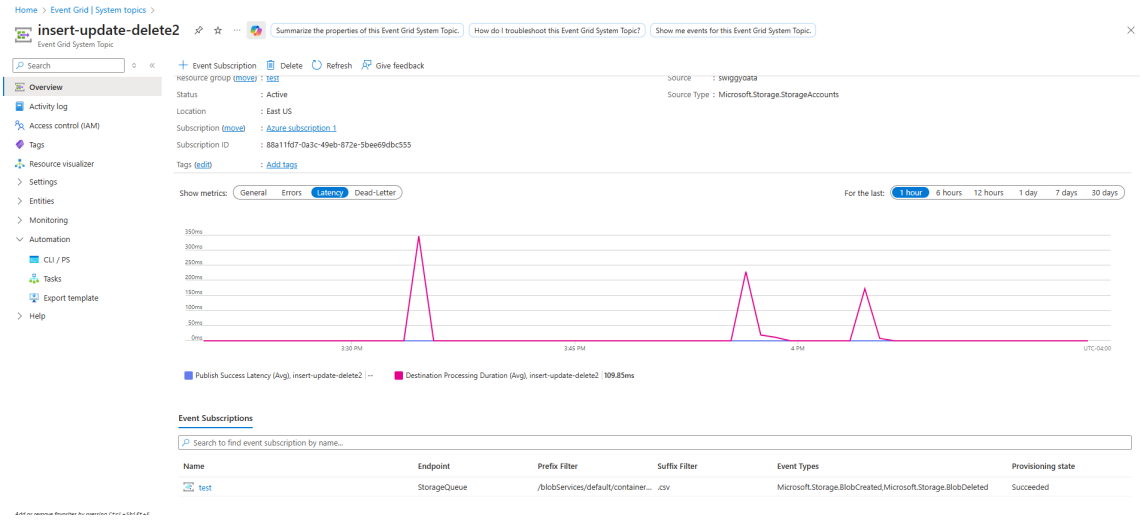
Storage Account hierarchy:



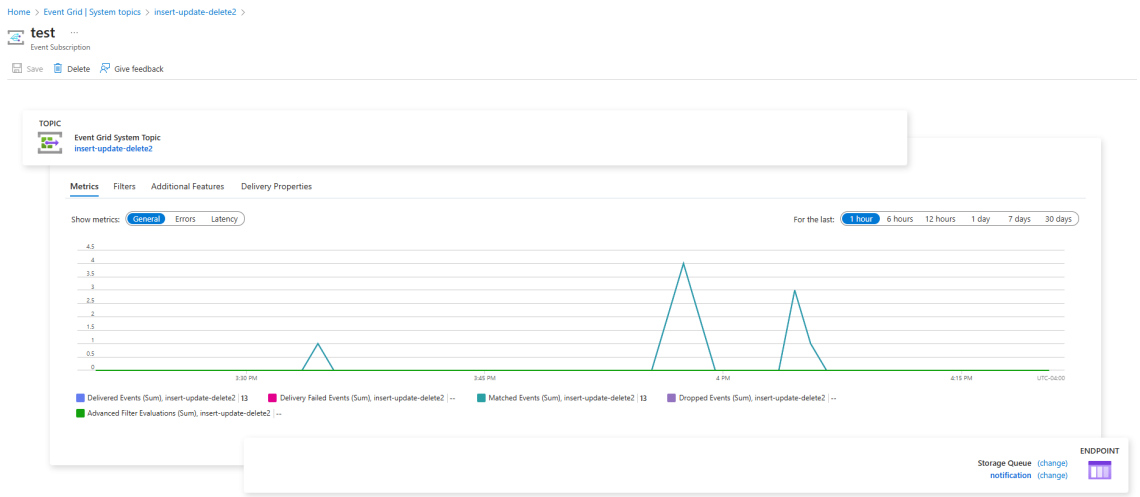
Event Grid Metrics once new file comes in the blob:



Latency Metrics:

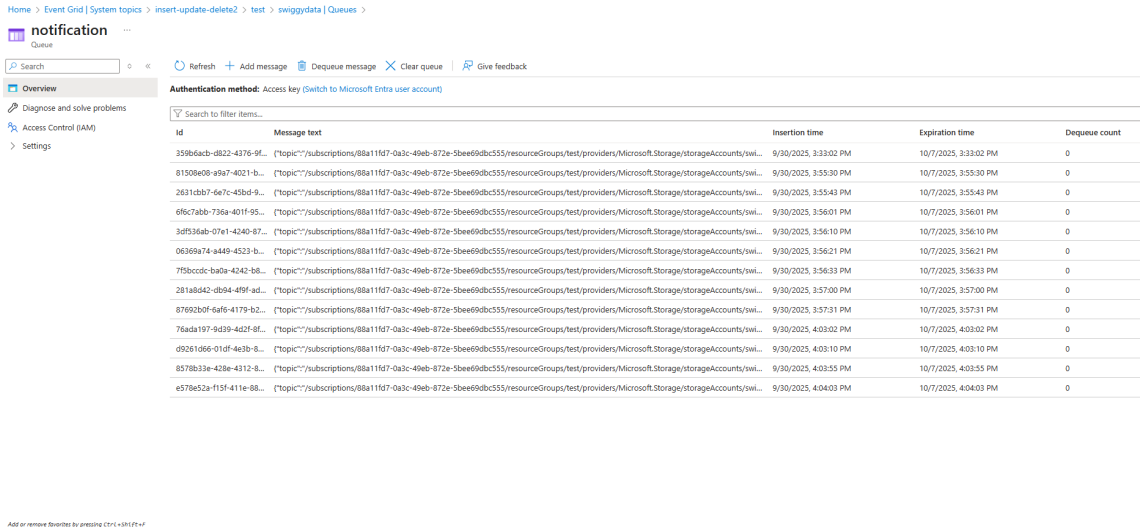


Event Subscription:

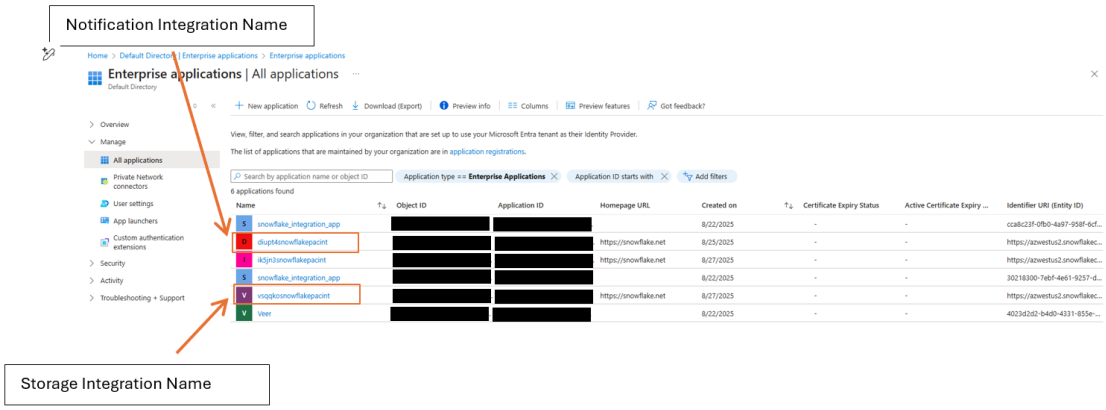


When new data is added to the Storage Account, Event Grid sends a topic to the Storage Queue, and Snowpipe reads the notification to automatically ingest the data: Note: Notifications are intentionally shown in the Storage Queue for demonstration purposes; once Snowpipe is integrated, they are processed immediately, and the notifications

page remains empty.



When both the Storage Account and notification integration are configured in Snowflake, their names appear in Azure Active Directory, indicating successful setup:



Development

Set Up Database & Schemas → Configure File Formats → Connect Snowflake with Azure → Apply Masking Policies for Sensitive Data:

```
-- First create a sandbox database to isolate development work. Inside it, then define schemas for different layers of our ELT pipeline:
create database if not exists sandbox;

use database sandbox;

create schema if not exists stage_sch;  -- Raw data from source systems
create schema if not exists clean_sch;  -- Transformed/validated data
create schema if not exists consumption_sch;  -- Final data for BI/Analytics
```

```
create schema if not exists common;   -- For shared policy objects


use schema stage_sch;
 -- Create file format to process the cvs files
create file format if not exists stage_sch.csv_file_format
        type='csv'
        compression='auto'
        field_delimiter =','
        record_delimiter= '\n'
        skip_header=1
        field_optionally_enclosed_by = '\042'
        null_if=('\\N')
        SKIP_BLANK_LINES = TRUE;

-- Manually uplaoding the the file in Stage_sch
create stage stage_sch.cvs_stg
directory =(enable=True)
COMMENT= "this is snowflake internal stage";

--Policy tag
create or replace tag
    common.pii_policy_tag
    allowed_values 'PII','PRICE','SENSITIVE', 'EMAIL'
    comment= 'This PII policy tag object';

create or replace masking policy
    common.pii_masking_policy as (pii_text string)
    returns string ->
    to_varchar('**PII**');

create or replace masking policy
    common.email_masking_policy as (email_text string)
    returns string ->
    to_varchar('** EMAIL**');

create or replace masking policy
    common.phone_masking_policy as (phone string)
    returns string ->
    to_varchar('**Phone**');

-- To check the loading data and validating the data use the root location. When
tables are not create and only files is loaded in the stage environment
list @stage_sch.cvs_stg/initial/location;
list @stage_sch.cvs_stg/delta;SANDBOX.STAGE_SCH.CVS_STG


-- External stage azure

-- Granting permissions for sysadmin to access the external table, integration and
notification integration.
GRANT USAGE ON stage stage_sch.azure_external_stage TO ROLE SYSADMIN;
```

```
--To connect Azure blob with snowflake I am using Azure integration and Notification
Integration with Azure Tenant ID. This is to make my data pull automatically from the
blob to snowflake.

CREATE OR REPLACE STAGE stage_sch.azure_external_stage
  URL = 'azure://swiggydata.blob.core.windows.net/swiggyrawdata/data/'
  STORAGE_INTEGRATION = AZURE_INTEGRATION;

--Integration and Notifcation Integration set up
CREATE STORAGE INTEGRATION AZURE_INTEGRATION
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = 'AZURE'
  ENABLED = TRUE
  AZURE_TENANT_ID = '6cf763d42c2-334b-2345-aa1f-3712a02f730d'  -- From Azure AD
  STORAGE_ALLOWED_LOCATIONS =
('azure://swiggydata.blob.core.windows.net/swiggyrawdata/data/');

desc  STORAGE INTEGRATION AZURE_INTEGRATION;

CREATE OR REPLACE  NOTIFICATION INTEGRATION Azure_Notification
  ENABLED = TRUE
  TYPE = QUEUE
  NOTIFICATION_PROVIDER = AZURE_STORAGE_QUEUE
  AZURE_STORAGE_QUEUE_PRIMARY_URI =
'https://swiggydata.queue.core.windows.net/notification'
  AZURE_TENANT_ID = '6c6162c2-356b-4825-aa1f-3782a01f730d'
  USE_PRIVATELINK_ENDPOINT = False
  COMMENT = 'notfication'

DESC  NOTIFICATION INTEGRATION Azure_Notification;

--Additional PII for Email, Phone.
create or replace tag
    common.pii_policy_tag
    allowed_values 'PII','PRICE','SENSITIVE', 'EMAIL'
    comment= 'This PII policy tag object';

create or replace masking policy
    common.pii_masking_policy as (pii_text string)
    returns string ->
    to_varchar('**PII**');

create or replace masking policy
    common.email_masking_policy as (email_text string)
    returns string ->
    to_varchar('** EMAIL**');

create or replace masking policy
    common.phone_masking_policy as (phone string)
```

```
    returns string ->
    to_varchar('**Phone**');
```

Location:

```
use role sysadmin;
use warehouse compute_wh;
use database sandbox;
use schema stage_sch;

-- Create table location stage table where the raw data will be stored
create table stage_sch.location (
    locationid text,
    city text,
    state text,
    zipcode text,
    activeflag text,
    createddate text,
    modifieddate text,
    -- audit columns for tracking and debugging
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment='This is the location stage/raw table where data will be copied from internal
stage using copy command. This is as-is data represetation from the source location.
All the columns are text data type except the audit columns that are added for
traceability.';
-- Stream to capture the changes
create or replace stream stage_sch.location_stm
on table stage_sch.location
append_only =true
comment = 'This is the append-only steam object on location table that gets delta data
based on chnages. It will track insert, update or delelte in the location data ';

-- Creating a snowpipe so i can automaticall pull data froma azure blob in stage_Schma
csv folder
CREATE OR REPLACE PIPE stage_sch.location_pipe
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
COPY INTO stage_sch.location (
    locationid, city, state, zipcode, activeflag, createddate,
    modifieddate, _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts
)
FROM (
    SELECT
        t.$1::text AS locationid,
        t.$2::text AS city,
        t.$3::text AS state,
        t.$4::text AS zipcode,
```

```
        t.$5::text AS activeflag,
        t.$6::text AS createddate,
        t.$7::text AS modifieddate,
        METADATA$FILENAME AS _stg_file_name,
        METADATA$FILE_LAST_MODIFIED AS _stg_file_load_ts,
        METADATA$FILE_CONTENT_KEY AS _stg_file_md5,
        CURRENT_TIMESTAMP AS _copy_data_ts
     from @azure_external_stage/location t

)
FILE_FORMAT = (FORMAT_NAME='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';

--level 2
use schema clean_sch;
-- Creating location table with in clean schema where all the transformation,
enriching data is stored
create or replace table clean_sch.restaurant_location(
        restaurant_location_sk number autoincrement primary key,
        location_id number  not null unique,
        city string(100) not null,
        state string(100) not null,
        state_code string(2) not null,
        is_union_territory boolean not null default false,
        capital_city_flag boolean not null default false,
        city_tier text(6),
        zip_code string(10) not null,
        active_flage string(10) not null,
        created_ts timestamp_tz not null,
        modified_ts timestamp_tz,
        --additional audit columns

        _stg_file_name string,
        _stg_file_load_ts timestamp_ntz,
        _stg_file_md5 string,
        _copy_data_ts timestamp_ntz default current_timestamp
)
comment ='Location entity under clean schema with appropritte data type under clean
schema layer, data is populated using merge statemnt from the the stage layer location
table. This table does not support SCD2';

-- Stream to capture the insert, update, delete and process data to the next level
create or replace stream clean_sch.restaurant_location_stm
on table clean_sch.restaurant_location
comment = ' this is standard stream object on the location table to track insert,
delete and update changes';

-- Creating a schedulle in snowflake to process data from one schema to anothe schema.
I impleneted the merge logic to get that from stage location stream do the necessary
transformation and store the data onto clean table.
```

```
CREATE OR REPLACE TASK clean_sch.location_merge_task2
WAREHOUSE = compute_wh
SCHEDULE = NULL  -- event-driven
WHEN SYSTEM$STREAM_HAS_DATA('stage_sch.location_stm')
AS
Merge into clean_sch.restaurant_location as target
using(
        select
        Cast(LocationID as Number) as Location_ID,
        Cast(City as String) as City,
        case when cast(State as String) ='Delhi' then 'New Delhi'
        else cast(State as String)
        end as State,
        -- state code mapping
        Case
                when State='Delhi' then 'DL'
                when State='Maharashtra' then 'MH'
                when State='Uttar Pradesh' then 'UP'
                when State='Gujarat' then 'GJ'
                when State='Rajasthan' then 'RJ'
                when State='Kerala' then 'KL'
                when State='Punjab' then 'PB'
                when State='Karnataka' then 'KA'
                when State='Madhya Pradesh' then 'MP'
                when State='Odisha' then 'OR'
                when State='Chandigarh' then 'CH'
                when State='West Bengal' then 'WB'
                when State='Sikkim' then 'SK'
                when State='Andhra Pradesh' then 'AP'
                when State='Assam' then 'AS'
                when State='Jammu and Kashmir' then 'JK'
                when State='Puducherry' then 'PY'
                when State='Uttarakhand' then 'UK'
                when State='Himachal Pradesh' then 'HP'
                when State='Tamil Nadu' then 'TN'
                when State='Goa' then 'GA'
                when State='Telangana' then 'TG'
                when State='Chhattisgarh' then 'CG'
                when State='Jharkhand' then 'JH'
                when State='Bihar' then 'BR'
                when STATE='Tripura' then 'TR'
                when STATE='Manipur' then 'MN'
                Else null
                end as state_code,
                case
                    when State in('Delhi', 'Chandigarh', 'Puducherry', 'Jammu and
Kashmir') then 'Y'
                    else 'N'
                    end as is_union_territory,
                case when (State='Delhi' or State='New Delhi' and City='Delhi') THEN
True
                when (State='Maharashtra' and City='Mumbai') Then True
```

```
                else False
                end as capital_city_flag,
                case when City in ('Mumbai','Delhi', 'Bengaluru', 'Hyderabad',
'Chennai','Noida', 'Kolkata', 'Pune', 'Ahmedabad') then 'Tier-1'
                when City in ('Jaipur', 'Lucknow', 'Kanpur',
'Nagpur','Varanasi','Indore', 'Bhopal', 'Patna','Vadodara', 'Coimbatore',
                'Ludhiana', 'Agra', 'Nashik', 'Ranchi',
'Meerut','Allahganj','Moradabad','Bareilly','Aligarh', 'Raipur','Ghazipur',
'Guwahati', 'Chandigarh') then 'Tier-1'
                else 'Tier-3'
                end as city_tier,
                Cast(ZipCode as String) as Zip_Code,
                Cast(ActiveFlag as String) as Active_Flag,
                to_char(to_timestamp(CreatedDate, 'MM/DD/YYYY HH24:MI'), 'YYYY-MM-DD
HH24:MI:SS') as created_ts,
                to_char(to_timestamp(ModifiedDate, 'MM/DD/YYYY HH24:MI'), 'YYYY-MM-DD
HH24:MI:SS') as modified_ts,
                _stg_file_name,
                _stg_file_load_ts,
                _stg_file_md5,
                _copy_data_ts
        from stage_sch.location_stm
        WHERE LocationID IS NOT NULL
) as source
on target.Location_ID=source.Location_ID
when matched and (
    target.City!=source.City or
    target.State!=source.State or
    target.state_code!=source.state_code or
    target.is_union_territory!=source.is_union_territory or
    target.capital_city_flag !=source.capital_city_flag or
    target.city_tier != source.city_tier  or
    target.Zip_Code != source.Zip_Code or
    target.Active_Flag != source.Active_Flag  or
    target.modified_ts !=source.modified_ts
) then
    update set
    target.City=source.City,
    target.State=source.State,
    target.state_code=source.state_code,
    target.is_union_territory=source.is_union_territory,
    target.capital_city_flag=source.capital_city_flag ,
    target.city_tier = source.city_tier  ,
    target.Zip_Code = source.Zip_Code ,
    target.Active_Flag = source.Active_Flag,
    target.modified_ts =source.modified_ts,
    target._stg_file_load_ts = source._stg_file_load_ts,
    target._stg_file_md5 = source._stg_file_md5,
    target._copy_data_ts = source._copy_data_ts
when not matched then
    insert(
        Location_ID,
```

```
        City,
        State,
        state_code,
        is_union_territory,
        capital_city_flag,
        city_tier,
        Zip_Code,
        Active_Flag,
        created_ts,
        modified_ts,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    )
    values(
        source.Location_ID,
        source.City,
        source.State,
        source.state_code,
        source.is_union_territory,
        source.capital_city_flag,
        source.city_tier,
        source.Zip_Code,
        source.Active_Flag,
        source.created_ts,
        source.modified_ts,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );

-- Making the shcema task resume after suspeneded
ALTER TASK clean_sch.location_merge_task2 RESUME;

/* list @stage_sch.cvs_stg/delta/location; */


--In next step I am trying to import new and updated record from delta folder and see
how my piepline words
/* copy into stage_sch.location (locationid, city, state, zipcode, activeflag,
createddate,
                            modifieddate, _stg_file_name, _stg_file_load_ts,
_stg_file_md5, _copy_data_ts)
from (
select
    t.$1::text as locationid,
    t.$2::text as city,
    t.$3::text as state,
    t.$4::text as zipcode,
    t.$5::text as activeflag,
```

```
    t.$6::text as createddate,
    t.$7::text as modifieddate,
    -- audit columns for tracking and debugging
    metadata$filename as _stg_file_name,
    metadata$file_last_modified as _stg_file_load_ts,
    metadata$file_content_key as _stg_file_md5,
    current_timestamp as _copy_data_ts

from @stage_sch.cvs_stg/delta/location/delta-day02-2rows-update.csv t
)
file_format = (format_name='stage_sch.csv_file_format')
on_error = abort_statement; */
```

Restaurant:

```
use warehouse compute_wh;
use database sandbox;
use schema stage_sch;

-- level 1
create or replace table stage_sch.restaurant (
    restaurantid text,
    name text ,                                          -- restaurant name, required
field
    cuisinetype text,                                     -- type of cuisine offered
    pricing_for_2 text,                                   -- pricing for two people as
text
    restaurant_phone text WITH TAG (common.pii_policy_tag = 'SENSITIVE'),
-- phone number as text
    operatinghours text,                                  -- restaurant operating hours
    locationid text ,                                     -- location id, default as
text
    activeflag text ,                                     -- active status
    openstatus text ,                                     -- open status
    locality text,                                        -- locality as text
    restaurant_address text,                              -- address as text
    latitude text,                                        -- latitude as text for
precision
    longitude text,                                       -- longitude as text for
precision
    createddate text,                                     -- record creation date
    modifieddate text,                                    -- last modified date
    -- audit columns for debugging
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment='This is the restaurant stage/raw table where data will be copied from
internal stage using copy command. This is as-is data represetation from the source
restaurant. All the columns are text data type except the audit columns that are added
for traceability.';
```

```
create or replace stream stage_sch.restaurant_stm
on table stage_sch.restaurant
append_only=True
comment='This is the append-only steam object on restaurant table that gets delta data
based on chnages. It will track insert, update or delelte in the restaurant data ';

-- Snowpipe
CREATE OR REPLACE PIPE stage_sch.restaurant_pipes
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
copy into stage_sch.restaurant (restaurantid, name, cuisinetype, pricing_for_2,
restaurant_phone,
                      operatinghours, locationid, activeflag, openstatus,
                      locality, restaurant_address, latitude, longitude,
                      createddate, modifieddate,
                      _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
        select
        t.$1::text as restaurantid,        -- restaurantid as the first column
        t.$2::text as name,
        t.$3::text as cuisinetype,
        t.$4::text as pricing_for_2,
        t.$5::text as restaurant_phone,
        t.$6::text as operatinghours,
        t.$7::text as locationid,
        t.$8::text as activeflag,
        t.$9::text as openstatus,
        t.$10::text as locality,
        t.$11::text as restaurant_address,
        t.$12::text as latitude,
        t.$13::text as longitude,
        t.$14::text as createddate,
        t.$15::text as modifieddate,
        -- audit columns for tracking & debugging
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp() as _copy_data_ts

        from @azure_external_stage/restaurants t
)
file_format=(format_name='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';

--using the next schema that is clean schema level 2
create or replace table clean_sch.restaurant (
    restaurant_sk number autoincrement primary key,              -- primary key with
auto-increment
```

```
    restaurant_id number unique,                                        -- restaurant
id without auto-increment
    name string(100) not null,                                      -- restaurant name,
required field
    cuisine_type string,                                            -- type of cuisine
offered
    pricing_for_two number(10, 2),                                  -- pricing for two
people, up to 10 digits with 2 decimal places
    restaurant_phone string(15) WITH TAG (common.pii_policy_tag = 'SENSITIVE'),
-- phone number, supports 10-digit or international format
    operating_hours string(100),                                     -- restaurant
operating hours
    location_id_fk number,                                          -- reference id for
location, defaulted to 1
    active_flag string(10),                                         -- indicates if the
restaurant is active
    open_status string(10),                                         -- indicates if the
restaurant is currently open
    locality string(100),                                           -- locality of the
restaurant
    restaurant_address string,                                      -- address of the
restaurant, supports longer text
    latitude number(9, 6),                                          -- latitude with 6
decimal places for precision
    longitude number(9, 6),                                         -- longitude with 6
decimal places for precision
    created_dt timestamp_tz,                                        -- record creation
date
    modified_dt timestamp_tz,                                       -- last modified
date, allows null if not modified

    -- additional audit columns
    _stg_file_name string,                                          -- file name for
audit
    _stg_file_load_ts timestamp_ntz,                                -- file load
timestamp for audit
    _stg_file_md5 string,                                           -- md5 hash for file
content for audit
    _copy_data_ts timestamp_ntz default current_timestamp        -- timestamp when
data is copied, defaults to current timestamp
)
comment = 'Restaurant entity under clean schema with appropriate data type under clean
schema layer, data is populated using merge statement from the stage layer location
table. This table does not support SCD2'

create or replace stream clean_sch.restaurant_stm
on table clean_sch.restaurant
comment='This is a standard stream object on the clean restaurant table to track
insert, update, and delete changes';
SHOW TASKS LIKE 'restauran_task2' IN SCHEMA clean_sch;
```

```
--level 2
CREATE OR REPLACE TASK clean_sch.restauran_task4
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
merge into clean_sch.restaurant as target
using (
            select
            try_cast(restaurantid  as number) as restaurant_id,
            try_cast(name as string) as name,
            try_cast(cuisinetype as string ) as cuisine_type,
            try_cast(pricing_for_2 as number(10,2)) as pricing_for_two,
            try_cast(restaurant_phone as string) as restaurant_phone,
            try_cast(operatinghours  as string) as operating_hours,
            try_cast(locationid  as number) as location_id_fk,
            try_cast(activeflag  as string) as active_flag,
            try_cast(openstatus as string) as open_status,
            try_cast(locality  as string) as locality,
            try_cast(restaurant_address as string) as restaurant_address,
            try_cast(latitude as number(9,6)) as latitude,
            try_cast(longitude as number(9,6)) as longitude,
            TO_CHAR(COALESCE(TRY_TO_TIMESTAMP(createddate, 'MM/DD/YYYY
HH24:MI'),TRY_TO_TIMESTAMP(createddate, 'MM/DD/YYYY')),'YYYY-MM-DD HH24:MI:SS.FF9') AS
created_dt,
            TO_CHAR(COALESCE(TRY_TO_TIMESTAMP(modifieddate, 'MM/DD/YYYY
HH24:MI'),TRY_TO_TIMESTAMP(createddate, 'MM/DD/YYYY')),'YYYY-MM-DD HH24:MI:SS.FF9') AS
modified_dt,
            _stg_file_name,
            _stg_file_load_ts,
            _stg_file_md5,
            _copy_data_ts
            from stage_sch.restaurant_stm
            where restaurant_id is not null
) as source
on target.restaurant_id=source.restaurant_id
when matched and  (
        target.name != source.name or
        target.cuisine_type != source.cuisine_type or
        target.pricing_for_two != source.pricing_for_two or
        target.restaurant_phone != source.restaurant_phone or
        target.operating_hours != source.operating_hours or
        target.location_id_fk != source.location_id_fk or
        target.active_flag != source.active_flag or
        target.open_status != source.open_status or
        target.locality != source.locality or
        target.restaurant_address != source.restaurant_address or
        target.latitude != source.latitude or
        target.longitude != source.longitude or
        target.created_dt != source.created_dt or
        target.modified_dt != source.modified_dt
) then
    update set
```

```
        target.name = source.name,
        target.cuisine_type = source.cuisine_type,
        target.pricing_for_two = source.pricing_for_two,
        target.restaurant_phone = source.restaurant_phone,
        target.operating_hours = source.operating_hours,
        target.location_id_fk = source.location_id_fk,
        target.active_flag = source.active_flag,
        target.open_status = source.open_status,
        target.locality = source.locality,
        target.restaurant_address = source.restaurant_address,
        target.latitude = source.latitude,
        target.longitude = source.longitude,
        target.created_dt = source.created_dt,
        target.modified_dt = source.modified_dt,
        target._stg_file_name = source._stg_file_name,
        target._stg_file_load_ts = source._stg_file_load_ts,
        target._stg_file_md5 = source._stg_file_md5,
        target._copy_data_ts = source._copy_data_ts
when not matched then
    insert(
        restaurant_id,
        name,
        cuisine_type,
        pricing_for_two,
        restaurant_phone,
        operating_hours,
        location_id_fk,
        active_flag,
        open_status,
        locality,
        restaurant_address,
        latitude,
        longitude,
        created_dt,
        modified_dt,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    )
    values(
        source.restaurant_id,
        source.name,
        source.cuisine_type,
        source.pricing_for_two,
        source.restaurant_phone,
        source.operating_hours,
        source.location_id_fk,
        source.active_flag,
        source.open_status,
        source.locality,
        source.restaurant_address,
```

```
        source.latitude,
        source.longitude,
        source.created_dt,
        source.modified_dt,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );

ALTER TASK clean_sch.restauran_task4 RESUME;

-- level 3
Create or replace table consumption_sch.restaurant_dim(
            restaurant_hk number primary key,
            restaurant_id number,
            name string(100),
            cuisine_type string,
            pricing_for_two number(10,2),
            restaurant_phone string(15) with tag (common.pii_policy_tag ='SENSITIVE'),
            operating_hours string(100),
            location_id_fk number,
            active_flag string(10),
            open_status string(10),
            locality string(100),
            restaurant_address string,
            latitude number(9, 6),
            longitude number(9, 6),
            Eff_start_date timestamp_tz,
            Eff_End_date timestamp_tz,
            current_flage boolean

)comment =' Restaurant dimensional table with scd enabled';


CREATE OR REPLACE TASK consumption_sch.restaurant_scd2_task4
WAREHOUSE = compute_wh
SCHEDULE  = 'USING CRON 3 21 * * * UTC'
AS
merge into
    consumption_sch.restaurant_dim as target
using
    clean_sch.restaurant_stm as source
on
    target.restaurant_id = source.restaurant_id and
    target.name = source.name and
    target.cuisine_type = source.cuisine_type and
    target.pricing_for_two = source.pricing_for_two and
    target.restaurant_phone = source.restaurant_phone and
    target.operating_hours = source.operating_hours and
    target.location_id_fk = source.location_id_fk and
    target.active_flag = source.active_flag and
```

```
    target.open_status = source.open_status and
    target.locality = source.locality and
    target.restaurant_address = source.restaurant_address and
    target.latitude = source.latitude and
    target.longitude = source.longitude
when matched
    and source.METADATA$ACTION = 'DELETE' and source.METADATA$ISUPDATE = 'TRUE' then
    -- Update the existing record to close its validity period
    UPDATE SET
        target.EFF_END_DATE = CURRENT_TIMESTAMP(),
        target.current_flage = FALSE
when not matched
    and source.METADATA$ACTION = 'INSERT' and source.METADATA$ISUPDATE = 'TRUE' then
    -- Insert new record with current data and new effective start date
    INSERT (
            restaurant_hk,
            restaurant_id ,
            name,
            cuisine_type ,
            pricing_for_two,
            restaurant_phone,
            operating_hours ,
            location_id_fk ,
            active_flag,
            open_status,
            locality,
            restaurant_address,
            latitude,
            longitude,
            Eff_start_date,
            Eff_End_date,
            current_flage
    )
    values (
        hash(SHA1_hex(CONCAT(source.RESTAURANT_ID, source.NAME, source.CUISINE_TYPE,
            source.PRICING_FOR_TWO, source.RESTAURANT_PHONE, source.OPERATING_HOURS,
            source.LOCATION_ID_FK, source.ACTIVE_FLAG, source.OPEN_STATUS,
source.LOCALITY,
            source.RESTAURANT_ADDRESS, source.LATITUDE, source.LONGITUDE))),
        source.RESTAURANT_ID,
        source.NAME,
        source.CUISINE_TYPE,
        source.PRICING_FOR_TWO,
        source.RESTAURANT_PHONE,
        source.OPERATING_HOURS,
        source.LOCATION_ID_FK,
        source.ACTIVE_FLAG,
        source.OPEN_STATUS,
        source.LOCALITY,
        source.RESTAURANT_ADDRESS,
        source.LATITUDE,
        source.LONGITUDE,
```

```
        CURRENT_TIMESTAMP(),
        NULL,
        TRUE
    )
when not matched
    and source.METADATA$ACTION = 'INSERT' and source.METADATA$ISUPDATE = 'FALSE' then
    -- Insert new record with current data and new effective start date
    INSERT (
        restaurant_hk,
            restaurant_id ,
            name,
            cuisine_type ,
            pricing_for_two,
            restaurant_phone,
            operating_hours ,
            location_id_fk ,
            active_flag,
            open_status,
            locality,
            restaurant_address,
            latitude,
            longitude,
            Eff_start_date,
            Eff_End_date,
            current_flage
    )
    values (
        hash(SHA1_hex(CONCAT(source.RESTAURANT_ID, source.NAME, source.CUISINE_TYPE,
            source.PRICING_FOR_TWO, source.RESTAURANT_PHONE, source.OPERATING_HOURS,
            source.LOCATION_ID_FK, source.ACTIVE_FLAG, source.OPEN_STATUS,
source.LOCALITY,
            source.RESTAURANT_ADDRESS, source.LATITUDE, source.LONGITUDE))),
        source.RESTAURANT_ID,
        source.NAME,
        source.CUISINE_TYPE,
        source.PRICING_FOR_TWO,
        source.RESTAURANT_PHONE,
        source.OPERATING_HOURS,
        source.LOCATION_ID_FK,
        source.ACTIVE_FLAG,
        source.OPEN_STATUS,
        source.LOCALITY,
        source.RESTAURANT_ADDRESS,
        source.LATITUDE,
        source.LONGITUDE,
        CURRENT_TIMESTAMP(),
        NULL,
        TRUE
    );

ALTER TASK consumption_sch.restaurant_scd2_task4 RESUME;
```

```
--In next step I am trying to import new and updated record from delta folder and see
how my piepline words

/* list @stage_sch.cvs_stg/delta/restaurant;*/

/* copy into stage_sch.restaurant (restaurantid, name, cuisinetype, pricing_for_2,
restaurant_phone,
                      operatinghours, locationid, activeflag, openstatus,
                      locality, restaurant_address, latitude, longitude,
                      createddate, modifieddate,
                      _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
        select
        t.$1::text as restaurantid,        -- restaurantid as the first column
        t.$2::text as name,
        t.$3::text as cuisinetype,
        t.$4::text as pricing_for_2,
        t.$5::text as restaurant_phone,
        t.$6::text as operatinghours,
        t.$7::text as locationid,
        t.$8::text as activeflag,
        t.$9::text as openstatus,
        t.$10::text as locality,
        t.$11::text as restaurant_address,
        t.$12::text as latitude,
        t.$13::text as longitude,
        t.$14::text as createddate,
        t.$15::text as modifieddate,
        -- audit columns for tracking & debugging
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp() as _copy_data_ts

        from @stage_sch.cvs_stg/delta/restaurant/day-02-upsert-restaurant-
delhi+NCR.csv t
)
file_format=(format_name='stage_sch.csv_file_format')
on_error = abort_statement;*/
```

Customer:

```
use role sysadmin;
use warehouse compute_wh;
use database sandbox;
use schema stage_sch;

 select
        t.$1::text as customerid,
        t.$2::text as name,
        t.$3::text as mobile,
        t.$4::text as email,
```

```
        t.$5::text as loginbyusing,
        t.$6::text as gender,
        t.$7::text as dob,
        t.$8::text as anniversary,
        t.$9::text as preferences,
        t.$10::text as createddate,
        t.$11::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
    from @stage_sch.cvs_stg/delta/customer/day-02-insert-update-Copy3.csv t

---level 1
create or replace table stage_sch.customer (
    customerid text,                    -- primary key as text
    name text,                          -- name as text
    mobile text WITH TAG (common.pii_policy_tag = 'PII'),                       --
mobile number as text
    email text WITH TAG (common.pii_policy_tag = 'EMAIL'),                        --
email as text
    loginbyusing text,                  -- login method as text
    gender text WITH TAG (common.pii_policy_tag = 'PII'),                       --
gender as text
    dob text WITH TAG (common.pii_policy_tag = 'PII'),                          --
date of birth as text
    anniversary text,                   -- anniversary as text
    preferences text,                   -- preferences as text
    createddate text,                   -- created date as text
    modifieddate text,                  -- modified date as text

    -- audit columns with appropriate data types
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment = 'This is the customer stage/raw table where data will be copied from
internal stage using copy command. This is as-is data represetation from the source
location. All the columns are text data type except the audit columns that are added
for traceability.';

create or replace stream stage_sch.customer_stm
on table stage_sch.customer
append_only = true
comment=' this is customer stream where we are just taking delta table details '

-- SNOWPIPE
CREATE OR REPLACE PIPE stage_sch.customer_pipes
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
```

```
copy into stage_sch.customer (customerid, name, mobile, email, loginbyusing, gender,
dob, anniversary,
                    preferences, createddate, modifieddate,
                    _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as customerid,
        t.$2::text as name,
        t.$3::text as mobile,
        t.$4::text as email,
        t.$5::text as loginbyusing,
        t.$6::text as gender,
        t.$7::text as dob,
        t.$8::text as anniversary,
        t.$9::text as preferences,
        t.$10::text as createddate,
        t.$11::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
        from @azure_external_stage/customers t
)
file_format=(format_name='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';

-- level 2 clean

CREATE OR REPLACE TABLE CLEAN_SCH.CUSTOMER (

    CUSTOMER_SK NUMBER AUTOINCREMENT PRIMARY KEY,                -- Auto-incremented
primary key
    CUSTOMER_ID STRING NOT NULL,                                 -- Customer ID
    First_Name STRING(100) NOT NULL,    -- Customer name
    Last_Name STRING(100) NOT NULL,      -- Customer name
    MOBILE STRING(15)  WITH TAG (common.pii_policy_tag = 'PII'),
-- Mobile number, accommodating international format
    EMAIL STRING(100) WITH TAG (common.pii_policy_tag = 'EMAIL'),
-- Email
    LOGIN_BY_USING STRING(50),                                   -- Method of login
(e.g., Social, Google, etc.)
    GENDER STRING(10)  WITH TAG (common.pii_policy_tag = 'PII'),
-- Gender
    DOB DATE WITH TAG (common.pii_policy_tag = 'PII'),
-- Date of birth in DATE format
    ANNIVERSARY DATE,                                            -- Anniversary in
DATE format
    PREFERENCES STRING,                                          -- Customer
preferences
    CUISINE_TYPES STRING,                                        -- Customer
CUISINE_Types
```

```
    CREATED_DT TIMESTAMP_TZ DEFAULT CURRENT_TIMESTAMP,           -- Record creation
timestamp
    MODIFIED_DT TIMESTAMP_TZ,                                    -- Record
modification timestamp, allows NULL if not modified

    -- Additional audit columns
    _STG_FILE_NAME STRING,                                       -- File name for
audit
    _STG_FILE_LOAD_TS TIMESTAMP_NTZ,                             -- File load
timestamp
    _STG_FILE_MD5 STRING,                                        -- MD5 hash for file
content
    _COPY_DATA_TS TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP        -- Copy data
timestamp
)
comment = 'Customer entity under clean schema with appropriate data type under clean
schema layer, data is populated using merge statement from the stage layer location
table. This table does not support SCD2';

create or replace stream CLEAN_SCH.CUSTOMER_Stm
on table CLEAN_SCH.CUSTOMER
comment='This is the stream object on customer entity to track insert, update, and
delete changes';


CREATE OR REPLACE TASK clean_sch.customer_task
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
merge into CLEAN_SCH.CUSTOMER AS target
USING (
    SELECT
        CUSTOMERID::STRING AS CUSTOMER_ID,
        SPLIT_PART(Name, ' ', 1)::STRING AS First_Name,
        SPLIT_PART(Name, ' ', 2)::STRING AS Last_Name,
        MOBILE::STRING AS MOBILE,
        EMAIL::STRING AS EMAIL,
        LOGINBYUSING::STRING AS LOGIN_BY_USING,
        GENDER::STRING AS GENDER,
        COALESCE(TRY_TO_DATE(DOB, 'YYYY-MM-DD'), TRY_TO_DATE(DOB, 'MM/DD/YYYY')) AS
DOB,
        COALESCE(TRY_TO_DATE(ANNIVERSARY, 'YYYY-MM-DD'), TRY_TO_DATE(ANNIVERSARY,
'MM/DD/YYYY')) AS ANNIVERSARY,

        -- Extract food preference and cuisine types from JSON
        PARSE_JSON(PREFERENCES):FoodPreference::STRING AS PREFERENCES, -- Customer
preferences
        ARRAY_TO_STRING(PARSE_JSON(PREFERENCES):CuisineTypes, ', ') AS CUISINE_TYPES,
--CuisineTypes
        TO_CHAR(COALESCE(TRY_TO_TIMESTAMP(CREATEDDATE, 'MM/DD/YYYY
HH24:MI'),TRY_TO_TIMESTAMP(CREATEDDATE, 'MM/DD/YYYY')),'YYYY-MM-DD"T"HH24:MI:SS.FF6')
AS CREATED_DT,
```

```
            TO_CHAR(COALESCE(TRY_TO_TIMESTAMP(MODIFIEDDATE, 'MM/DD/YYYY
HH24:MI'),TRY_TO_TIMESTAMP(MODIFIEDDATE, 'MM/DD/YYYY')),'YYYY-MM-DD"T"HH24:MI:SS.FF6')
AS MODIFIED_DT,
        _STG_FILE_NAME
        _STG_FILE_LOAD_TS,
        _STG_FILE_MD5,
        _COPY_DATA_TS
    FROM STAGE_SCH.CUSTOMER_STM
) AS source
ON target.CUSTOMER_ID = source.CUSTOMER_ID
WHEN MATCHED AND (
        target.FIRST_NAME != source.FIRST_NAME or
        target.LAST_NAME != source.LAST_NAME or
        target.MOBILE != source.MOBILE or
        target.EMAIL != source.EMAIL or
        target.LOGIN_BY_USING != source.LOGIN_BY_USING or
        target.GENDER != source.GENDER or
        target.DOB != source.DOB or
        target.ANNIVERSARY != source.ANNIVERSARY or
        target.PREFERENCES != source.PREFERENCES or
        target.CUISINE_TYPES != source.CUISINE_TYPES or
        target.CREATED_DT != source.CREATED_DT or
        target.MODIFIED_DT != source.MODIFIED_DT
) THEN
    UPDATE SET
        target.FIRST_NAME = source.FIRST_NAME,
        target.LAST_NAME = source.LAST_NAME,
        target.MOBILE = source.MOBILE,
        target.EMAIL = source.EMAIL,
        target.LOGIN_BY_USING = source.LOGIN_BY_USING,
        target.GENDER = source.GENDER,
        target.DOB = COALESCE(source.DOB, target.DOB),
        target.ANNIVERSARY = COALESCE(source.ANNIVERSARY, target.ANNIVERSARY),
        target.PREFERENCES = source.PREFERENCES,
        target.CUISINE_TYPES = source.CUISINE_TYPES,
        target.CREATED_DT = source.CREATED_DT,
        target.MODIFIED_DT = source.MODIFIED_DT,
        target._STG_FILE_NAME = source._STG_FILE_NAME,
        target._STG_FILE_LOAD_TS = source._STG_FILE_LOAD_TS,
        target._STG_FILE_MD5 = source._STG_FILE_MD5,
        target._COPY_DATA_TS = source._COPY_DATA_TS
WHEN NOT MATCHED THEN
    INSERT (
        CUSTOMER_ID,
        FIRST_NAME,
        LAST_NAME,
        MOBILE,
        EMAIL,
        LOGIN_BY_USING,
        GENDER,
        DOB,
        ANNIVERSARY,
```

```
        PREFERENCES,
        CUISINE_TYPES,
        CREATED_DT,
        MODIFIED_DT,
        _STG_FILE_NAME,
        _STG_FILE_LOAD_TS,
        _STG_FILE_MD5,
        _COPY_DATA_TS
    )
    VALUES (
        source.CUSTOMER_ID,
        source.FIRST_NAME,
        source.LAST_NAME,
        source.MOBILE,
        source.EMAIL,
        source.LOGIN_BY_USING,
        source.GENDER,
        source.DOB,
        source.ANNIVERSARY,
        source.PREFERENCES,
        source.CUISINE_TYPES,
        source.CREATED_DT,
        source.MODIFIED_DT,
        source._STG_FILE_NAME,
        source._STG_FILE_LOAD_TS,
        source._STG_FILE_MD5,
        source._COPY_DATA_TS
    );

ALTER TASK clean_sch.customer_task RESUME;

-- LEVEL 3
CREATE OR REPLACE TABLE CONSUMPTION_SCH.CUSTOMER_DIM (
    CUSTOMER_HK NUMBER PRIMARY KEY,               -- Surrogate key for the customer
    CUSTOMER_ID STRING NOT NULL,                                -- Natural key for
the customer
    FIRST_NAME STRING(100) NOT NULL,                                    -- Customer
FIRST name
    LAST_NAME STRING(100) NOT NULL,                            -- Customer LAST name
    MOBILE STRING(15) WITH TAG (common.pii_policy_tag = 'PII'),
-- Mobile number
    EMAIL STRING(100) WITH TAG (common.pii_policy_tag = 'EMAIL'),
-- Email
    LOGIN_BY_USING STRING(50),                                  -- Method of login
    GENDER STRING(10) WITH TAG (common.pii_policy_tag = 'PII'),
-- Gender
    DOB DATE WITH TAG (common.pii_policy_tag = 'PII'),
-- Date of birth
    ANNIVERSARY DATE,                                           -- Anniversary
    PREFERENCES STRING,                                         -- Preferences
    CUISINE_TYPES String,                                        --CUISINE_TYPES
    EFF_START_DATE TIMESTAMP_TZ,                                -- Effective start
```

```
date
    EFF_END_DATE TIMESTAMP_TZ,                                      -- Effective end date
(NULL if active)
    IS_CURRENT BOOLEAN                                              -- Flag to indicate
the current record
)
COMMENT = 'Customer Dimension table with SCD Type 2 handling for historical
tracking.';

-- Stream
CREATE OR REPLACE TASK consumption_sch.customer_scd2_task
WAREHOUSE = compute_wh
SCHEDULE  = 'USING CRON 3 21 * * * UTC'
AS
MERGE INTO
    CONSUMPTION_SCH.CUSTOMER_DIM AS target
USING
    CLEAN_SCH.CUSTOMER_STM AS source
ON
    target.CUSTOMER_ID = source.CUSTOMER_ID AND
    target.FIRST_NAME = source.FIRST_NAME AND
    target.LAST_NAME = source.LAST_NAME AND
    target.MOBILE = source.MOBILE AND
    target.EMAIL = source.EMAIL AND
    target.LOGIN_BY_USING = source.LOGIN_BY_USING AND
    target.GENDER = source.GENDER AND
    target.DOB = source.DOB AND
    target.ANNIVERSARY = source.ANNIVERSARY AND
    target.PREFERENCES = source.PREFERENCES AND
    target.CUISINE_TYPES = source.CUISINE_TYPES
WHEN MATCHED
    AND source.METADATA$ACTION = 'DELETE' AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Update the existing record to close its validity period
    UPDATE SET
        target.EFF_END_DATE = CURRENT_TIMESTAMP(),
        target.IS_CURRENT = FALSE
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT' AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        CUSTOMER_HK,
        CUSTOMER_ID,
        FIRST_NAME,
        LAST_NAME,
        MOBILE,
        EMAIL,
        LOGIN_BY_USING,
        GENDER,
        DOB,
        ANNIVERSARY,
        PREFERENCES,
        CUISINE_TYPES,
```

```
        EFF_START_DATE,
        EFF_END_DATE,
        IS_CURRENT
    )
    VALUES (
        hash(SHA1_hex(CONCAT(source.CUSTOMER_ID, source.FIRST_NAME,source.LAST_NAME,
source.MOBILE,
            source.EMAIL, source.LOGIN_BY_USING, source.GENDER, source.DOB,
            source.ANNIVERSARY, source.PREFERENCES, source.CUISINE_TYPES))),
        source.CUSTOMER_ID,
        source.FIRST_NAME,
        source.LAST_NAME,
        source.MOBILE,
        source.EMAIL,
        source.LOGIN_BY_USING,
        source.GENDER,
        source.DOB,
        source.ANNIVERSARY,
        source.PREFERENCES,
        source.CUISINE_TYPES,
        CURRENT_TIMESTAMP(),
        NULL,
        TRUE
    )
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT' AND source.METADATA$ISUPDATE = 'FALSE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        CUSTOMER_HK,
        CUSTOMER_ID,
        FIRST_NAME,
        LAST_NAME,
        MOBILE,
        EMAIL,
        LOGIN_BY_USING,
        GENDER,
        DOB,
        ANNIVERSARY,
        PREFERENCES,
        CUISINE_TYPES,
        EFF_START_DATE,
        EFF_END_DATE,
        IS_CURRENT
    )
    VALUES (
        hash(SHA1_hex(CONCAT(source.CUSTOMER_ID, source.FIRST_NAME, source.LAST_NAME,
source.MOBILE,
            source.EMAIL, source.LOGIN_BY_USING, source.GENDER, source.DOB,
            source.ANNIVERSARY, source.PREFERENCES, source.CUISINE_TYPES))),
        source.CUSTOMER_ID,
        source.FIRST_NAME,
        source.LAST_NAME,
```

```
        source.MOBILE,
        source.EMAIL,
        source.LOGIN_BY_USING,
        source.GENDER,
        source.DOB,
        source.ANNIVERSARY,
        source.PREFERENCES,
        source.CUISINE_TYPES,
        CURRENT_TIMESTAMP(),
        NULL,
        TRUE
    );

 ALTER TASK consumption_sch.customer_scd2_task RESUME;
-- In next step I am trying to import new and updated record from delta folder and see
how my piepline words

/* list @stage_sch.cvs_stg/delta/customer/; */

/* copy into stage_sch.customer (customerid, name, mobile, email, loginbyusing,
gender, dob, anniversary,
                    preferences, createddate, modifieddate,
                    _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as customerid,
        t.$2::text as name,
        t.$3::text as mobile,
        t.$4::text as email,
        t.$5::text as loginbyusing,
        t.$6::text as gender,
        t.$7::text as dob,
        t.$8::text as anniversary,
        t.$9::text as preferences,
        t.$10::text as createddate,
        t.$11::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
    from @stage_sch.cvs_stg/delta/customer/day-02-insert-update-Copy3.csv t
)
file_format = (format_name = 'stage_sch.csv_file_format')
on_error = abort_statement; */
```

Customer Address:

```
use warehouse compute_wh;
use database sandbox;
use schema stage_sch;

create or replace table stage_sch.customeraddress(
```

```
        addressid text,
        customerid text comment 'Customer FK (soruce data)',
        flatno text,
        houseno text,
        floor text,
        building text,
        landmark text,
        locality text,
        city text,
        state text,
        pincode text,
        coordinates text,
        primaryflag text,
        addresstype text,
        createddate text,
        modifiedate text,

        --audit columns with appropriate data types
        _stg_file_name text,
        _stg_file_load_ts timestamp,
        _stg_file_md5 text,
        _copy_data_ts timestamp default current_timestamp
)comment='This is customer address stage table without any scd2 applied '

create or replace stream stage_sch.customeraddress_stm
on table stage_sch.customeraddress
append_only= true
comment='Stream data for customeraddress where all the data will be inserted';

--- pipe
CREATE OR REPLACE PIPE stage_sch.customer_address_pipes
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
copy into stage_sch.customeraddress(addressid, customerid, flatno, houseno, floor,
building,
                                landmark, locality,city,pincode, state, coordinates,
primaryflag, addresstype,
                                createddate, modifiedate,
                                _stg_file_name, _stg_file_load_ts, _stg_file_md5,
_copy_data_ts)
from (
    select
        t.$1::text as addressid,
        t.$2::text as customerid,
        t.$3::text as flatno,
        t.$4::text as houseno,
        t.$5::text as floor,
        t.$6::text as building,
        t.$7::text as landmark,
        t.$8::text as locality,
        t.$9::text as city,
```

```
        t.$10::text as State,
        t.$11::text as Pincode,
        t.$12::text as coordinates,
        t.$13::text as primaryflag,
        t.$14::text as addresstype,
        t.$15::text as createddate,
        t.$16::text as modifiedate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
        from @azure_external_stage/customer_address t
)
file_format=(format_name='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';


--level 2
CREATE OR REPLACE TABLE CLEAN_SCH.CUSTOMER_ADDRESS (
    CUSTOMER_ADDRESS_SK NUMBER AUTOINCREMENT PRIMARY KEY comment 'Surrogate Key
(EWH)',                -- Auto-incremented primary key
    ADDRESS_ID INT comment 'Primary Key (Source Data)',                 -- Primary key
as string
    CUSTOMER_ID_FK INT comment 'Customer FK (Source Data)',                -- Foreign
key reference as string (no constraint in Snowflake)
    FLAT_NO STRING,                    -- Flat number as string
    HOUSE_NO STRING,                   -- House number as string
    FLOOR STRING,                      -- Floor as string
    BUILDING STRING,                   -- Building name as string
    LANDMARK STRING,                   -- Landmark as string
    locality STRING,                   -- locality as string
    CITY STRING,                       -- City as string
    STATE STRING,                      -- State as string
    PINCODE STRING,                    -- Pincode as string
    COORDINATES STRING,                -- Coordinates as string
    PRIMARY_FLAG STRING,               -- Primary flag as string
    ADDRESS_TYPE STRING,               -- Address type as string
    CREATED_DATE TIMESTAMP_TZ,         -- Created date as timestamp with time zone
    MODIFIED_DATE TIMESTAMP_TZ,        -- Modified date as timestamp with time zone

    -- Audit columns with appropriate data types
    _STG_FILE_NAME STRING,
    _STG_FILE_LOAD_TS TIMESTAMP,
    _STG_FILE_MD5 STRING,
    _COPY_DATA_TS TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
comment = 'Customer address entity under clean schema with appropriate data type under
clean schema layer, data is populated using merge statement from the stage layer
location table. This table does not support SCD2';

-- Stream object to capture the changes.
```

```
create or replace stream CLEAN_SCH.CUSTOMER_ADDRESS_STM
on table CLEAN_SCH.CUSTOMER_ADDRESS
comment = 'This is the stream object on customer address entity to track insert,
update, and delete changes';


CREATE OR REPLACE TASK clean_sch.customer_address_task
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
MERGE INTO clean_sch.customer_address AS target
USING (
    SELECT
        CAST(addressid AS INT) AS address_id,
        CAST(customerid AS INT) AS customer_id_fk,
        flatno AS flat_no,
        houseno AS house_no,
        floor,
        building,
        landmark,
        locality,
        city,
        state,
        pincode,
        coordinates,
        primaryflag AS primary_flag,
        addresstype AS address_type,
        TO_CHAR(COALESCE(TRY_TO_TIMESTAMP(createddate, 'MM/DD/YYYY
HH24:MI'),TRY_TO_TIMESTAMP(createddate, 'MM/DD/YYYY')),'YYYY-MM-DD"T"HH24:MI:SS') AS
created_date,
        TO_CHAR(COALESCE(TRY_TO_TIMESTAMP(modifiedate, 'MM/DD/YYYY
HH24:MI'),TRY_TO_TIMESTAMP(modifiedate, 'MM/DD/YYYY')),'YYYY-MM-DD"T"HH24:MI:SS') AS
modified_date,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    FROM stage_sch.customeraddress_stm
) AS source
ON target.address_id = source.address_id
WHEN MATCHED AND (
        target.flat_no != source.flat_no or
        target.house_no != source.house_no or
        target.floor != source.floor or
        target.building != source.building or
        target.landmark != source.landmark or
        target.locality != source.locality or
        target.city != source.city or
        target.state != source.state or
        target.pincode != source.pincode or
        target.coordinates != source.coordinates or
        target.primary_flag != source.primary_flag or
```

```
        target.address_type != source.address_type or
        target.created_date != source.created_date or
        target.modified_date != source.modified_date
) THEN
    UPDATE SET
        target.flat_no = source.flat_no,
        target.house_no = source.house_no,
        target.floor = source.floor,
        target.building = source.building,
        target.landmark = source.landmark,
        target.locality = source.locality,
        target.city = source.city,
        target.state = source.state,
        target.pincode = source.pincode,
        target.coordinates = source.coordinates,
        target.primary_flag = source.primary_flag,
        target.address_type = source.address_type,
        target.created_date = source.created_date,
        target.modified_date = source.modified_date,
        target._stg_file_name = source._stg_file_name,
        target._stg_file_load_ts = source._stg_file_load_ts,
        target._stg_file_md5 = source._stg_file_md5,
        target._copy_data_ts = source._copy_data_ts
WHEN NOT MATCHED THEN
    INSERT (
        address_id,
        customer_id_fk,
        flat_no,
        house_no,
        floor,
        building,
        landmark,
        locality,
        city,
        state,
        pincode,
        coordinates,
        primary_flag,
        address_type,
        created_date,
        modified_date,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    )
    VALUES (
        source.address_id,
        source.customer_id_fk,
        source.flat_no,
        source.house_no,
        source.floor,
```

```
        source.building,
        source.landmark,
        source.locality,
        source.city,
        source.state,
        source.pincode,
        source.coordinates,
        source.primary_flag,
        source.address_type,
        source.created_date,
        source.modified_date,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );
ALTER TASK clean_sch.customer_task RESUME;

--level3

    CREATE OR REPLACE TABLE CONSUMPTION_SCH.CUSTOMER_ADDRESS_DIM (
    CUSTOMER_ADDRESS_HK NUMBER PRIMARY KEY comment 'Customer Address HK (EDW)',
-- Surrogate key (hash key)
    ADDRESS_ID INT comment 'Primary Key (Source System)',
-- Original primary key
    CUSTOMER_ID_FK STRING comment 'Customer FK (Source System)',
-- Surrogate key from Customer Dimension (Foreign Key)
    FLAT_NO STRING,                                 -- Flat number
    HOUSE_NO STRING,                                -- House number
    FLOOR STRING,                                   -- Floor
    BUILDING STRING,                                -- Building name
    LANDMARK STRING,                                -- Landmark
    LOCALITY STRING,                                -- Locality
    CITY STRING,                                    -- City
    STATE STRING,                                   -- State
    PINCODE STRING,                                 -- Pincode
    COORDINATES STRING,                             -- Geo-coordinates
    PRIMARY_FLAG STRING,                            -- Whether it's the primary address
    ADDRESS_TYPE STRING,                            -- Type of address (e.g., Home,
Office)

    -- SCD2 Columns
    EFF_START_DATE TIMESTAMP_TZ,                                -- Effective start
date
    EFF_END_DATE TIMESTAMP_TZ,                                  -- Effective end date
(NULL if active)
    IS_CURRENT BOOLEAN                                          -- Flag to indicate
the current record
);


CREATE OR REPLACE TASK consumption_sch.customer_address_scd2_task
```

```
WAREHOUSE = compute_wh
SCHEDULE  = 'USING CRON 3 21 * * * UTC'
AS
MERGE INTO
    CONSUMPTION_SCH.CUSTOMER_ADDRESS_DIM AS target
USING
    CLEAN_SCH.CUSTOMER_ADDRESS_STM AS source
ON
    target.ADDRESS_ID = source.ADDRESS_ID AND
    target.CUSTOMER_ID_FK = source.CUSTOMER_ID_FK AND
    target.FLAT_NO = source.FLAT_NO AND
    target.HOUSE_NO = source.HOUSE_NO AND
    target.FLOOR = source.FLOOR AND
    target.BUILDING = source.BUILDING AND
    target.LANDMARK = source.LANDMARK AND
    target.LOCALITY = source.LOCALITY AND
    target.CITY = source.CITY AND
    target.STATE = source.STATE AND
    target.PINCODE = source.PINCODE AND
    target.COORDINATES = source.COORDINATES AND
    target.PRIMARY_FLAG = source.PRIMARY_FLAG AND
    target.ADDRESS_TYPE = source.ADDRESS_TYPE
WHEN MATCHED
    AND source.METADATA$ACTION = 'DELETE' AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Update the existing record to close its validity period
    UPDATE SET
        target.EFF_END_DATE = CURRENT_TIMESTAMP(),
        target.IS_CURRENT = FALSE
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT' AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        CUSTOMER_ADDRESS_HK,
        ADDRESS_ID,
        CUSTOMER_ID_FK,
        FLAT_NO,
        HOUSE_NO,
        FLOOR,
        BUILDING,
        LANDMARK,
        LOCALITY,
        CITY,
        STATE,
        PINCODE,
        COORDINATES,
        PRIMARY_FLAG,
        ADDRESS_TYPE,
        EFF_START_DATE,
        EFF_END_DATE,
        IS_CURRENT
    )
    VALUES (
```

```
        hash(SHA1_hex(CONCAT(source.ADDRESS_ID, source.CUSTOMER_ID_FK, source.FLAT_NO,
            source.HOUSE_NO, source.FLOOR, source.BUILDING, source.LANDMARK,
            source.LOCALITY, source.CITY, source.STATE, source.PINCODE,
            source.COORDINATES, source.PRIMARY_FLAG, source.ADDRESS_TYPE))),
        source.ADDRESS_ID,
        source.CUSTOMER_ID_FK,
        source.FLAT_NO,
        source.HOUSE_NO,
        source.FLOOR,
        source.BUILDING,
        source.LANDMARK,
        source.LOCALITY,
        source.CITY,
        source.STATE,
        source.PINCODE,
        source.COORDINATES,
        source.PRIMARY_FLAG,
        source.ADDRESS_TYPE,
        CURRENT_TIMESTAMP(),
        NULL,
        TRUE
    )
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT' AND source.METADATA$ISUPDATE = 'FALSE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        CUSTOMER_ADDRESS_HK,
        ADDRESS_ID,
        CUSTOMER_ID_FK,
        FLAT_NO,
        HOUSE_NO,
        FLOOR,
        BUILDING,
        LANDMARK,
        LOCALITY,
        CITY,
        STATE,
        PINCODE,
        COORDINATES,
        PRIMARY_FLAG,
        ADDRESS_TYPE,
        EFF_START_DATE,
        EFF_END_DATE,
        IS_CURRENT
    )
    VALUES (
        hash(SHA1_hex(CONCAT(source.ADDRESS_ID, source.CUSTOMER_ID_FK, source.FLAT_NO,
            source.HOUSE_NO, source.FLOOR, source.BUILDING, source.LANDMARK,
            source.LOCALITY, source.CITY, source.STATE, source.PINCODE,
            source.COORDINATES, source.PRIMARY_FLAG, source.ADDRESS_TYPE))),
        source.ADDRESS_ID,
        source.CUSTOMER_ID_FK,
```

```
        source.FLAT_NO,
        source.HOUSE_NO,
        source.FLOOR,
        source.BUILDING,
        source.LANDMARK,
        source.LOCALITY,
        source.CITY,
        source.STATE,
        source.PINCODE,
        source.COORDINATES,
        source.PRIMARY_FLAG,
        source.ADDRESS_TYPE,
        CURRENT_TIMESTAMP(),
        NULL,
        TRUE
    );

 ALTER TASK consumption_sch.customer_address_scd2_task RESUME;

--  In next step I am trying to import new and updated record from delta folder and
see how my piepline words
/*list @stage_sch.cvs_stg/delta/customer-address;*/

/*copy into stage_sch.customeraddress(addressid, customerid, flatno, houseno, floor,
building,
                                landmark, locality,city,pincode, state, coordinates,
primaryflag, addresstype,
                                createddate, modifiedate,
                                _stg_file_name, _stg_file_load_ts, _stg_file_md5,
_copy_data_ts)
from (
    select
        t.$1::text as addressid,
        t.$2::text as customerid,
        t.$3::text as flatno,
        t.$4::text as houseno,
        t.$5::text as floor,
        t.$6::text as building,
        t.$7::text as landmark,
        t.$8::text as locality,
        t.$9::text as city,
        t.$10::text as State,
        t.$11::text as Pincode,
        t.$12::text as coordinates,
        t.$13::text as primaryflag,
        t.$14::text as addresstype,
        t.$15::text as createddate,
        t.$16::text as modifiedate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
```

```
    from @stage_sch.cvs_stg/delta/customer-address/day-02-customer-address-book.csv t
)
file_format = (format_name = 'stage_sch.csv_file_format')
on_error = abort_statement;*/
```

Menu:

```
use role sysadmin;
use database sandbox;
use schema stage_sch;
use warehouse compute_wh;

create or replace table stage_sch.menu (
    menuid text comment 'Primary Key (Source System)',                    -- primary
key as text
    restaurantid text comment 'Restaurant FK(Source System)',              -- foreign
key reference as text (no constraint in snowflake)
    itemname text,                 -- item name as text
    description text,              -- description as text
    price text,                    -- price as text (no decimal constraint)
    category text,                 -- category as text
    availability text,             -- availability as text
    itemtype text,                 -- item type as text
    createddate text,              -- created date as text
    modifieddate text,             -- modified date as text

    -- audit columns with appropriate data types
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment = 'This is the menu stage/raw table where data will be copied from internal
stage using copy command. This is as-is data represetation from the source location.
All the columns are text data type except the audit columns that are added for
traceability.';

create or replace stream stage_sch.menu_stm
on table  stage_sch.menu
append_only=True
comment= 'Menu stage table to process data in the clean dim';

--- pipe
CREATE OR REPLACE PIPE stage_sch.menu_pipes
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
copy into stage_sch.menu(menuid, restaurantid, itemname, description, price, category,
                availability, itemtype, createddate, modifieddate,
                _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
```

```
        t.$1::text as menuid,
        t.$2::text as restaurantid,
        t.$3::text as itemname,
        t.$4::text as description,
        t.$5::text as price,
        t.$6::text as category,
        t.$7::text as availability,
        t.$8::text as itemtype,
        t.$9::text as createddate,
        t.$10::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
        from @azure_external_stage/menu t
)
file_format=(format_name='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';

--level2 clean

CREATE OR REPLACE TABLE clean_sch.menu (
    Menu_SK INT AUTOINCREMENT PRIMARY KEY comment 'Surrogate Key (EDW)',  -- Auto-
incrementing primary key for internal tracking
    Menu_ID INT NOT NULL UNIQUE comment 'Primary Key (Source System)' ,             --
Unique and non-null Menu_ID
    Restaurant_ID_FK INT comment 'Restaurant FK(Source System)' ,
-- Identifier for the restaurant
    Item_Name STRING not null,                        -- Name of the menu item
    Description STRING not null,                     -- Description of the menu item
    Price DECIMAL(10, 2) not null,                   -- Price as a numeric value with
2 decimal places
    Category STRING,                         -- Food category (e.g., North Indian)
    Availability BOOLEAN,                    -- Availability status (True/False)
    Item_Type STRING,                         -- Dietary classification (e.g., Vegan)
    Created_dt TIMESTAMP_NTZ,                -- Date when the record was created
    Modified_dt TIMESTAMP_NTZ,               -- Date when the record was last modified

    -- Audit columns for traceability
    _STG_FILE_NAME STRING,                   -- Source file name
    _STG_FILE_LOAD_TS TIMESTAMP_NTZ,         -- Timestamp when data was loaded from the
staging layer
    _STG_FILE_MD5 STRING,                    -- MD5 hash of the source file
    _COPY_DATA_TS TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP -- Timestamp when data was
copied to the clean layer
)
comment = 'Menu entity under clean schema with appropriate data type under clean
schema layer, data is populated using merge statement from the stage layer location
table. This table does not support SCD2';

create or replace stream CLEAN_SCH.menu_stm
```

```
on table CLEAN_SCH.menu
comment = 'This is the stream object on menu table table to track insert, update, and
delete changes';


CREATE OR REPLACE TASK clean_sch.menu_task
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
MERGE INTO clean_sch.menu AS target
USING (
    SELECT
        TRY_CAST(menuid AS INT) AS Menu_ID,
        TRY_CAST(restaurantid AS INT) AS Restaurant_ID_FK,
        TRIM(itemname) AS Item_Name,
        TRIM(description) AS Description,
        TRY_CAST(price AS DECIMAL(10, 2)) AS Price,
        TRIM(category) AS Category,
        CASE
            WHEN LOWER(availability) = 'true' THEN TRUE
            WHEN LOWER(availability) = 'false' THEN FALSE
            ELSE NULL
        END AS Availability,
        TRIM(itemtype) AS Item_Type,
        TO_VARCHAR(TRY_TO_TIMESTAMP_NTZ(createddate), 'YYYY-MM-DD HH24:MI:SS.FF6') AS
Created_dt,
        TO_VARCHAR(TRY_TO_TIMESTAMP_NTZ(modifieddate), 'YYYY-MM-DD HH24:MI:SS.FF6') AS
Modified_dt,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    FROM stage_sch.menu_stm
) AS source
ON target.Menu_ID = source.Menu_ID
WHEN MATCHED AND (
        target.Item_Name != source.Item_Name or
        target.Description != source.Description or
        target.Price != source.Price or
        target.Category != source.Category or
        target.Availability != source.Availability or
        target.Item_Type != source.Item_Type or
        target.Created_dt != source.Created_dt or
        target.Modified_dt != source.Modified_dt
) THEN
    UPDATE SET
        target.Item_Name = source.Item_Name,
        target.Description = source.Description,
        target.Price = source.Price,
        target.Category = source.Category,
        target.Availability = source.Availability,
        target.Item_Type = source.Item_Type,
```

```
        target.Created_dt = source.Created_dt,
        target.Modified_dt = source.Modified_dt,
        target._stg_file_name = source._stg_file_name,
        target._stg_file_load_ts = source._stg_file_load_ts,
        target._stg_file_md5 = source._stg_file_md5,
        target._copy_data_ts = source._copy_data_ts
WHEN NOT MATCHED THEN
    INSERT (
        Menu_ID,
        Restaurant_ID_FK,
        Item_Name,
        Description,
        Price,
        Category,
        Availability,
        Item_Type,
        Created_dt,
        Modified_dt,
        _STG_FILE_NAME,
        _STG_FILE_LOAD_TS,
        _STG_FILE_MD5,
        _COPY_DATA_TS
    )
    VALUES (
        source.Menu_ID,
        source.Restaurant_ID_FK,
        source.Item_Name,
        source.Description,
        source.Price,
        source.Category,
        source.Availability,
        source.Item_Type,
        source.Created_dt,
        source.Modified_dt,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );

ALTER TASK clean_sch.menu_task RESUME;

--level 3 Dimension
    CREATE OR REPLACE TABLE consumption_sch.menu_dim (
    Menu_Dim_HK NUMBER primary key comment 'Menu Dim HK (EDW)',
-- Hash key generated for Menu Dim table
    Menu_ID INT NOT NULL comment 'Primary Key (Source System)',
-- Unique and non-null Menu_ID
    Restaurant_ID_FK INT NOT NULL comment 'Restaurant FK (Source System)',
-- Identifier for the restaurant
    Item_Name STRING,                           -- Name of the menu item
    Description STRING,                        -- Description of the menu item
```

```
    Price DECIMAL(10, 2),                       -- Price as a numeric value with 2
decimal places
    Category STRING,                            -- Food category (e.g., North Indian)
    Availability BOOLEAN,                       -- Availability status (True/False)
    Item_Type STRING,                           -- Dietary classification (e.g.,
Vegan)
    EFF_START_DATE TIMESTAMP_NTZ,               -- Effective start date of the record
    EFF_END_DATE TIMESTAMP_NTZ,                 -- Effective end date of the record
    IS_CURRENT BOOLEAN                         -- Flag to indicate if the record is
current (True/False)
)
COMMENT = 'This table stores the dimension data for the menu items, tracking
historical changes using SCD Type 2. Each menu item has an effective start and end
date, with a flag indicating if it is the current record or historical. The hash key
(Menu_Dim_HK) is generated based on Menu_ID and Restaurant_ID.';

CREATE OR REPLACE TASK consumption_sch.menu_scd2_task
WAREHOUSE = compute_wh
SCHEDULE  = 'USING CRON 3 21 * * * UTC'
AS
MERGE INTO
    CONSUMPTION_SCH.menu_dim AS target
USING
    CLEAN_SCH.menu_stm AS source
ON
   target.Menu_ID = source.Menu_ID AND
    target.Restaurant_ID_FK = source.Restaurant_ID_FK AND
    target.Item_Name = source.Item_Name AND
    target.Description = source.Description AND
    target.Price = source.Price AND
    target.Category = source.Category AND
    target.Availability = source.Availability AND
    target.Item_Type = source.Item_Type
WHEN MATCHED
    AND source.METADATA$ACTION = 'DELETE' AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Update the existing record to close its validity period
    UPDATE SET
        target.EFF_END_DATE = CURRENT_TIMESTAMP(),
        target.IS_CURRENT = FALSE
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT' AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        Menu_Dim_HK,               -- Hash key
        Menu_ID,
        Restaurant_ID_FK,
        Item_Name,
        Description,
        Price,
        Category,
        Availability,
        Item_Type,
```

```
        EFF_START_DATE,
        EFF_END_DATE,
        IS_CURRENT
    )
    VALUES (
        hash(SHA1_hex(CONCAT(source.Menu_ID, source.Restaurant_ID_FK,
            source.Item_Name, source.Description, source.Price,
            source.Category, source.Availability, source.Item_Type))),  -- Hash key
        source.Menu_ID,
        source.Restaurant_ID_FK,
        source.Item_Name,
        source.Description,
        source.Price,
        source.Category,
        source.Availability,
        source.Item_Type,
        CURRENT_TIMESTAMP(),       -- Effective start date
        NULL,                      -- Effective end date (NULL for current record)
        TRUE                       -- IS_CURRENT = TRUE for new record
    )
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT' AND source.METADATA$ISUPDATE = 'FALSE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        Menu_Dim_HK,               -- Hash key
        Menu_ID,
        Restaurant_ID_FK,
        Item_Name,
        Description,
        Price,
        Category,
        Availability,
        Item_Type,
        EFF_START_DATE,
        EFF_END_DATE,
        IS_CURRENT
    )
    VALUES (
        hash(SHA1_hex(CONCAT(source.Menu_ID, source.Restaurant_ID_FK,
            source.Item_Name, source.Description, source.Price,
            source.Category, source.Availability, source.Item_Type))),  -- Hash key
        source.Menu_ID,
        source.Restaurant_ID_FK,
        source.Item_Name,
        source.Description,
        source.Price,
        source.Category,
        source.Availability,
        source.Item_Type,
        CURRENT_TIMESTAMP(),       -- Effective start date
        NULL,                      -- Effective end date (NULL for current record)
        TRUE                       -- IS_CURRENT = TRUE for new record
```

```
    );

ALTER TASK consumption_sch.menu_scd2_task RESUME;


-- In next step I am trying to import new and updated record from delta folder and see
how my piepline words
/*list @stage_sch.cvs_stg/delta/menu;*/


/* copy into stage_sch.menu (menuid, restaurantid, itemname, description, price,
category,
                availability, itemtype, createddate, modifieddate,
                _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as menuid,
        t.$2::text as restaurantid,
        t.$3::text as itemname,
        t.$4::text as description,
        t.$5::text as price,
        t.$6::text as category,
        t.$7::text as availability,
        t.$8::text as itemtype,
        t.$9::text as createddate,
        t.$10::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
    from @stage_sch.cvs_stg/delta/menu/day-02-menu-data.csv t
)
file_format = (format_name = 'stage_sch.csv_file_format')
on_error = abort_statement;*/
```

Delivery Agent:

```
use role sysadmin;
use database sandbox;
use schema stage_sch;
use warehouse compute_wh;


create or replace table stage_sch.deliveryagent (
    deliveryagentid text comment 'Primary Key (Source System)',         -- primary key
as text
    name text,           -- name as text, required field
    phone text,            -- phone as text, unique constraint indicated
    vehicletype text,             -- vehicle type as text
    locationid text,              -- foreign key reference as text (no constraint in
snowflake)
    status text,                  -- status as text
```

```
    gender text,                  -- status as text
    rating text,                  -- rating as text
    createddate text,             -- created date as text
    modifieddate text,            -- modified date as text

    -- audit columns with appropriate data types
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment = 'This is the delivery stage/raw table where data will be copied from
internal stage using copy command. This is as-is data represetation from the source
location. All the columns are text data type except the audit columns that are added
for traceability.';

create or replace stream stage_sch.deliveryagent_stm
on table stage_sch.deliveryagent
append_only = true
comment = 'This is the append-only stream object on delivery agent table that only
gets delta data';


--pipe
CREATE OR REPLACE PIPE stage_sch.delivery_agents_pipes
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
copy into stage_sch.deliveryagent (deliveryagentid, name, phone, vehicletype,
locationid,
                        status, gender, rating, createddate, modifieddate,
                        _stg_file_name, _stg_file_load_ts, _stg_file_md5,
_copy_data_ts)
from (
    select
        t.$1::text as deliveryagentid,
        t.$2::text as name,
        t.$3::text as phone,
        t.$4::text as vehicletype,
        t.$5::text as locationid,
        t.$6::text as status,
        t.$7::text as gender,
        t.$8::text as rating,
        t.$9::text as createddate,
        t.$10::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
        from @azure_external_stage/delivery_agents t
)
file_format=(format_name='stage_sch.csv_file_format')
```

```
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';

--level 2

CREATE OR REPLACE TABLE clean_sch.delivery_agent (
    delivery_agent_sk INT AUTOINCREMENT PRIMARY KEY comment 'Surrogate Key (EDW)', --
Primary key with auto-increment
    delivery_agent_id INT NOT NULL UNIQUE comment 'Primary Key (Source System)',
-- Delivery agent ID as integer
    name STRING NOT NULL,                -- Name as string, required field
    phone STRING NOT NULL,                 -- Phone as string, unique constraint
    vehicle_type STRING NOT NULL,                -- Vehicle type as string
    location_id_fk INT comment 'Location FK(Source System)',                    --
Location ID as integer
    status STRING,                       -- Status as string
    gender STRING,                       -- Gender as string
    rating number(4,2),                        -- Rating as float
    created_dt TIMESTAMP_NTZ,          -- Created date as timestamp without timezone
    modified_dt TIMESTAMP_NTZ,         -- Modified date as timestamp without timezone

    -- Audit columns with appropriate data types
    _stg_file_name STRING,               -- Staging file name as string
    _stg_file_load_ts TIMESTAMP,         -- Staging file load timestamp
    _stg_file_md5 STRING,                -- Staging file MD5 hash as string
    _copy_data_ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- Data copy timestamp with
default value
)
comment = 'Delivery entity under clean schema with appropriate data type under clean
schema layer, data is populated using merge statement from the stage layer location
table. This table does not support SCD2';

create or replace stream CLEAN_SCH.delivery_agent_stm
on table CLEAN_SCH.delivery_agent
comment = 'This is the stream object on delivery agent table table to track insert,
update, and delete changes';


CREATE OR REPLACE TASK clean_sch.delivery_agents_task
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
MERGE INTO clean_sch.delivery_agent AS target
USING (
    SELECT
        TRY_CAST(deliveryagentid AS INT) AS delivery_agent_id,
        TRY_CAST(name as string) as name,
        TRY_CAST(phone as STRING) as phone,
        TRY_CAST(vehicletype as STRING) as vehicle_type,
        TRY_CAST(locationid  as Int) as location_id_fk,
        TRY_CAST(status as string) as status,
        TRY_CAST(gender as string) as gender,
```

```
        TRY_TO_DECIMAL(rating, 4, 2) AS rating,
        TO_VARCHAR(TRY_TO_TIMESTAMP_NTZ(createddate), 'YYYY-MM-DD HH24:MI:SS.FF6') AS
Created_dt,
        TO_VARCHAR(TRY_TO_TIMESTAMP_NTZ(modifieddate), 'YYYY-MM-DD HH24:MI:SS.FF6') AS
Modified_dt,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    FROM stage_sch.deliveryagent_stm
) AS source
ON target.delivery_agent_id = source.delivery_agent_id
WHEN MATCHED AND (
        target.phone != source.phone or
        target.vehicle_type != source.vehicle_type or
        target.location_id_fk != source.location_id_fk or
        target.status != source.status or
        target.gender != source.gender or
        target.rating != source.rating or
        target.created_dt != source.created_dt or
        target.modified_dt != source.modified_dt
) THEN
    UPDATE SET
        target.phone = source.phone,
        target.vehicle_type = source.vehicle_type,
        target.location_id_fk = source.location_id_fk,
        target.status = source.status,
        target.gender = source.gender,
        target.rating = source.rating,
        target.created_dt = source.created_dt,
        target.modified_dt = source.modified_dt,
        target._stg_file_name = source._stg_file_name,
        target._stg_file_load_ts = source._stg_file_load_ts,
        target._stg_file_md5 = source._stg_file_md5,
        target._copy_data_ts = source._copy_data_ts
WHEN NOT MATCHED THEN
    INSERT (
        delivery_agent_id,
        name,
        phone,
        vehicle_type,
        location_id_fk,
        status,
        gender,
        rating,
        created_dt,
        modified_dt,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    )
```

```
    VALUES (
        source.delivery_agent_id,
        source.name,
        source.phone,
        source.vehicle_type,
        source.location_id_fk,
        source.status,
        source.gender,
        source.rating,
        source.created_dt,
        source.modified_dt,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );
ALTER TASK clean_sch.delivery_agents_task RESUME;

--level3

CREATE OR REPLACE TABLE consumption_sch.delivery_agent_dim (
    delivery_agent_hk number primary key comment 'Delivery Agend Dim HK (EDW)',
-- Hash key for unique identification
    delivery_agent_id NUMBER not null comment 'Primary Key (Source System)',
-- Business key
    name STRING NOT NULL,                   -- Delivery agent name
    phone STRING UNIQUE,                    -- Phone number, unique
    vehicle_type STRING,                    -- Type of vehicle
    location_id_fk NUMBER NOT NULL comment 'Location FK (Source System)',
-- Location ID
    status STRING,                          -- Current status of the delivery agent
    gender STRING,                          -- Gender
    rating NUMBER(4,2),                     -- Rating with one decimal precision
    eff_start_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Effective start date
    eff_end_date TIMESTAMP,                 -- Effective end date (NULL for active
record)
    is_current BOOLEAN DEFAULT TRUE
)
comment =  'Dim table for delivery agent entity with SCD2 support.';

CREATE OR REPLACE TASK consumption_sch.delivery_agent_scd2_task
WAREHOUSE = compute_wh
SCHEDULE  = 'USING CRON 3 21 * * * UTC'
AS
MERGE INTO consumption_sch.delivery_agent_dim AS target
USING CLEAN_SCH.delivery_agent_stm AS source
ON
    target.delivery_agent_id = source.delivery_agent_id AND
    target.name = source.name AND
    target.phone = source.phone AND
    target.vehicle_type = source.vehicle_type AND
    target.location_id_fk = source.location_id_fk AND
```

```
    target.status = source.status AND
    target.gender = source.gender AND
    target.rating = source.rating
WHEN MATCHED
    AND source.METADATA$ACTION = 'DELETE'
    AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Update the existing record to close its validity period
    UPDATE SET
        target.eff_end_date = CURRENT_TIMESTAMP,
        target.is_current = FALSE
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT'
    AND source.METADATA$ISUPDATE = 'TRUE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        delivery_agent_hk,        -- Hash key
        delivery_agent_id,
        name,
        phone,
        vehicle_type,
        location_id_fk,
        status,
        gender,
        rating,
        eff_start_date,
        eff_end_date,
        is_current
    )
    VALUES (
        hash(SHA1_HEX(CONCAT(source.delivery_agent_id, source.name, source.phone,
            source.vehicle_type, source.location_id_fk, source.status,
            source.gender, source.rating))), -- Hash key
        delivery_agent_id,
        source.name,
        source.phone,
        source.vehicle_type,
        location_id_fk,
        source.status,
        source.gender,
        source.rating,
        CURRENT_TIMESTAMP,       -- Effective start date
        NULL,                    -- Effective end date (NULL for current record)
        TRUE                    -- IS_CURRENT = TRUE for new record
    )
WHEN NOT MATCHED
    AND source.METADATA$ACTION = 'INSERT'
    AND source.METADATA$ISUPDATE = 'FALSE' THEN
    -- Insert new record with current data and new effective start date
    INSERT (
        delivery_agent_hk,        -- Hash key
        delivery_agent_id,
        name,
```

```
        phone,
        vehicle_type,
        location_id_fk,
        status,
        gender,
        rating,
        eff_start_date,
        eff_end_date,
        is_current
    )
    VALUES (
        hash(SHA1_HEX(CONCAT(source.delivery_agent_id, source.name, source.phone,
            source.vehicle_type, source.location_id_fk, source.status,
            source.gender, source.rating))), -- Hash key
        source.delivery_agent_id,
        source.name,
        source.phone,
        source.vehicle_type,
        source.location_id_fk,
        source.status,
        source.gender,
        source.rating,
        CURRENT_TIMESTAMP,       -- Effective start date
        NULL,                    -- Effective end date (NULL for current record)
        TRUE                   -- IS_CURRENT = TRUE for new record
    );

ALTER TASK consumption_sch.delivery_agent_scd2_task RESUME;

-- In next step I am trying to import new and updated record from delta folder and see
how my piepline words

/* list @stage_sch.cvs_stg/delta/delivery-agent;*/

/*  copy into deliveryagent (deliveryagentid, name, phone, vehicletype, locationid,
                         status, gender, rating, createddate, modifieddate,
                         _stg_file_name, _stg_file_load_ts, _stg_file_md5,
_copy_data_ts)
from (
    select
        t.$1::text as deliveryagentid,
        t.$2::text as name,
        t.$3::text as phone,
        t.$4::text as vehicletype,
        t.$5::text as locationid,
        t.$6::text as status,
        t.$7::text as gender,
        t.$8::text as rating,
        t.$9::text as createddate,
        t.$10::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
```

```
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
    from @stage_sch.cvs_stg/delta/delivery-agent/day-02-delivery-agent.csv t
)
file_format = (format_name = 'stage_sch.csv_file_format')
on_error = abort_statement;*/
```

Delivery: Transaction table so not populating the dimension table as they represent as fact table.

```
use role sysadmin;

use database sandbox;
use schema stage_sch;
use warehouse compute_wh;


-- this table may have additional information like picked time, accept time etc.
create or replace table stage_sch.delivery (
    deliveryid text comment 'Primary Key (Source System)',                          -
- foreign key reference as text (no constraint in snowflake)
    orderid text comment 'Order FK (Source System)',                          --
foreign key reference as text (no constraint in snowflake)
    deliveryagentid text comment 'Delivery Agent FK(Source System)',
-- foreign key reference as text (no constraint in snowflake)
    deliverystatus text,                    -- delivery status as text
    estimatedtime text,                     -- estimated time as text
    addressid text comment 'Customer Address FK(Source System)',
-- foreign key reference as text (no constraint in snowflake)
    deliverydate text,                      -- delivery date as text
    createddate text,                       -- created date as text
    modifieddate text,                      -- modified date as text

    -- audit columns with appropriate data types
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment = 'This is the delivery stage/raw table where data will be copied from
internal stage using copy command. This is as-is data represetation from the source
location. All the columns are text data type except the audit columns that are added
for traceability.';

create or replace stream stage_sch.delivery_stm
on table stage_sch.delivery
append_only = true
comment = 'this is the append-only stream object on delivery table that only gets
delta data';

--pipe
CREATE OR REPLACE PIPE stage_sch.deliveries_pipes
```

```
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
copy into stage_sch.delivery (deliveryid,orderid, deliveryagentid, deliverystatus,
                    estimatedtime, addressid, deliverydate, createddate,
                    modifieddate, _stg_file_name, _stg_file_load_ts,
                    _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as deliveryid,
        t.$2::text as orderid,
        t.$3::text as deliveryagentid,
        t.$4::text as deliverystatus,
        t.$5::text as estimatedtime,
        t.$6::text as addressid,
        t.$7::text as deliverydate,
        t.$8::text as createddate,
        t.$9::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
         from @azure_external_stage/deliveries t
)
file_format=(format_name='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';


--level 2 clean

CREATE OR REPLACE TABLE clean_sch.delivery (
    delivery_sk INT AUTOINCREMENT PRIMARY KEY comment 'Surrogate Key (EDW)', --
Primary key with auto-increment
    delivery_id INT NOT NULL comment 'Primary Key (Source System)',
    order_id_fk NUMBER NOT NULL comment 'Order FK (Source System)',
-- Foreign key reference, converted to numeric type
    delivery_agent_id_fk NUMBER NOT NULL comment 'Delivery Agent FK (Source System)',
-- Foreign key reference, converted to numeric type
    delivery_status STRING,                 -- Delivery status, stored as a string
    estimated_time STRING,                  -- Estimated time, stored as a string
    customer_address_id_fk NUMBER NOT NULL  comment 'Customer Address FK (Source
System)',                       -- Foreign key reference, converted to numeric type
    delivery_date TIMESTAMP,                -- Delivery date, converted to timestamp
    created_date TIMESTAMP,                 -- Created date, converted to timestamp
    modified_date TIMESTAMP,                -- Modified date, converted to timestamp

    -- Audit columns with appropriate data types
    _stg_file_name STRING,                  -- Source file name
    _stg_file_load_ts TIMESTAMP,            -- Source file load timestamp
    _stg_file_md5 STRING,                   -- MD5 checksum of the source file
    _copy_data_ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- Metadata timestamp
```

```
)
comment = 'Delivery entity under clean schema with appropriate data type under clean
schema layer, data is populated using merge statement from the stage layer location
table. This table does not support SCD2';

create or replace stream CLEAN_SCH.delivery_stm
on table CLEAN_SCH.delivery
comment = 'This is the stream object on delivery agent table table to track insert,
update, and delete changes';


CREATE OR REPLACE TASK clean_sch.delivery_task
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
MERGE INTO clean_sch.delivery AS target
USING (
    SELECT
        TRY_CAST(deliveryid AS INT) AS delivery_id,
        TRY_CAST(orderid as NUMBER) as order_id_fk,
        TRY_CAST(deliveryagentid AS NUMBER) as delivery_agent_id_fk,
        TRY_CAST(deliverystatus AS STRING) as delivery_status,
        TRY_CAST(estimatedtime  AS STRING) as estimated_time,
        TRY_CAST(addressid AS NUMBER) AS customer_address_id_fk,
        TRY_CAST(deliverydate AS TIMESTAMP_NTZ) as delivery_date,
        TO_VARCHAR(TRY_TO_TIMESTAMP_NTZ(createddate), 'YYYY-MM-DD HH24:MI:SS.FF6') AS
created_date,
        TO_VARCHAR(TRY_TO_TIMESTAMP_NTZ(modifieddate), 'YYYY-MM-DD HH24:MI:SS.FF6') AS
modified_date,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    FROM stage_sch.delivery_stm
    where deliveryid is not null and TRY_CAST(deliveryid AS NUMBER) IS NOT NULL
) AS source
ON target.delivery_id = source.delivery_id and
    target.order_id_fk = source.order_id_fk and
    target.delivery_agent_id_fk = source.delivery_agent_id_fk
WHEN MATCHED AND (
        target.delivery_status != source.delivery_status or
        target.estimated_time != source.estimated_time or
        target.customer_address_id_fk != source.customer_address_id_fk or
        target.delivery_date != source.delivery_date or
        target.created_date != source.created_date or
        target.modified_date != source.modified_date
) THEN
    UPDATE SET
        target.delivery_status = source.delivery_status,
        target.estimated_time = source.estimated_time,
        target.customer_address_id_fk = source.customer_address_id_fk,
        target.delivery_date = source.delivery_date,
```

```
        target.created_date = source.created_date,
        target.modified_date = source.modified_date,
        target._stg_file_name = source._stg_file_name,
        target._stg_file_load_ts = source._stg_file_load_ts,
        target._stg_file_md5 = source._stg_file_md5,
        target._copy_data_ts = source._copy_data_ts
WHEN NOT MATCHED THEN
    INSERT (
        delivery_id,
        order_id_fk,
        delivery_agent_id_fk,
        delivery_status,
        estimated_time,
        customer_address_id_fk,
        delivery_date,
        created_date,
        modified_date,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    )
    VALUES (
        source.delivery_id,
        source.order_id_fk,
        source.delivery_agent_id_fk,
        source.delivery_status,
        source.estimated_time,
        source.customer_address_id_fk,
        source.delivery_date,
        source.created_date ,
        source.modified_date,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );


ALTER TASK clean_sch.delivery_task RESUME;

--In next step I am trying to import new and updated record from delta folder and see
how my piepline words
/* list @stage_sch.cvs_stg/delta/delivery/;*/



/*  copy into stage_sch.delivery (deliveryid,orderid, deliveryagentid, deliverystatus,
                    estimatedtime, addressid, deliverydate, createddate,
                    modifieddate, _stg_file_name, _stg_file_load_ts,
                    _stg_file_md5, _copy_data_ts)
from (
```

```
    select
        t.$1::text as deliveryid,
        t.$2::text as orderid,
        t.$3::text as deliveryagentid,
        t.$4::text as deliverystatus,
        t.$5::text as estimatedtime,
        t.$6::text as addressid,
        t.$7::text as deliverydate,
        t.$8::text as createddate,
        t.$9::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
    from @stage_sch.cvs_stg/delta/delivery/day-02-delivery.csv t
)
file_format = (format_name = 'stage_sch.csv_file_format')
on_error = abort_statement;*/
```

Orders: Transaction table so not populating the dimension table as they represent as fact table.

```
use role sysadmin;
use database sandbox;
use schema stage_sch;
use warehouse compute_wh;

create or replace table stage_sch.orders (
    orderid text comment 'Primary Key (Source System)',                  -- primary
key as text
    customerid text comment 'Customer FK(Source System)',               -- foreign key
reference as text (no constraint in snowflake)
    restaurantid text comment 'Restaurant FK(Source System)',             -- foreign
key reference as text (no constraint in snowflake)
    orderdate text,                -- order date as text
    totalamount text,              -- total amount as text (no decimal constraint)
    status text,                   -- status as text
    paymentmethod text,            -- payment method as text
    createddate text,              -- created date as text
    modifieddate text,             -- modified date as text

    -- audit columns with appropriate data types
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment = 'This is the order stage/raw table where data will be copied from internal
stage using copy command. This is as-is data represetation from the source location.
All the columns are text data type except the audit columns that are added for
traceability.';
```

```
--Stream
create or replace stream stage_sch.orders_stm
on table stage_sch.orders
append_only = true
comment = 'This is the append-only stream object on orders entity that only gets delta
data';

--pipe
CREATE OR REPLACE PIPE stage_sch.orders_pipes
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
copy into stage_sch.orders (orderid, customerid, restaurantid, orderdate, totalamount,
                  status, paymentmethod, createddate, modifieddate,
                  _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as orderid,
        t.$2::text as customerid,
        t.$3::text as restaurantid,
        t.$4::text as orderdate,
        t.$5::text as totalamount,
        t.$6::text as status,
        t.$7::text as paymentmethod,
        t.$8::text as createddate,
        t.$9::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
         from @azure_external_stage/orders t
)
file_format=(format_name='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';

--level 2

CREATE OR REPLACE TABLE CLEAN_SCH.ORDERS (
    ORDER_SK NUMBER AUTOINCREMENT PRIMARY KEY comment 'Surrogate Key (EDW)',
-- Auto-incremented primary key
    ORDER_ID BIGINT UNIQUE comment 'Primary Key (Source System)',
-- Primary key inferred as BIGINT
    CUSTOMER_ID_FK BIGINT comment 'Customer FK(Source System)',                    --
Foreign key inferred as BIGINT
    RESTAURANT_ID_FK BIGINT comment 'Restaurant FK(Source System)',                  --
Foreign key inferred as BIGINT
    ORDER_DATE TIMESTAMP,                 -- Order date inferred as TIMESTAMP
    TOTAL_AMOUNT DECIMAL(10, 2),          -- Total amount inferred as DECIMAL with two
decimal places
    STATUS STRING,                        -- Status as STRING
    PAYMENT_METHOD STRING,                -- Payment method as STRING
```

```
    created_dt timestamp_tz,                                        -- record creation
date
    modified_dt timestamp_tz,                                       -- last modified
date, allows null if not modified

    -- additional audit columns
    _stg_file_name string,                                          -- file name for
audit
    _stg_file_load_ts timestamp_ntz,                                -- file load
timestamp for audit
    _stg_file_md5 string,                                           -- md5 hash for file
content for audit
    _copy_data_ts timestamp_ntz default current_timestamp           -- timestamp when
data is copied, defaults to current timestamp
)
comment = 'Order entity under clean schema with appropriate data type under clean
schema layer, data is populated using merge statement from the stage layer location
table. This table does not support SCD2';

-- Stream object to capture the changes.
create or replace stream CLEAN_SCH.ORDERS_stm
on table CLEAN_SCH.ORDERS
comment = 'This is the stream object on ORDERS table table to track insert, update,
and delete changes';


CREATE OR REPLACE TASK clean_sch.orders_task
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
MERGE INTO CLEAN_SCH.ORDERS AS target
USING (
    SELECT
        TRY_CAST(orderid AS INT) AS ORDER_ID,
        TRY_CAST(customerid as INT) as CUSTOMER_ID_FK,
        TRY_CAST(restaurantid AS INT) as RESTAURANT_ID_FK,
        TRY_CAST(orderdate  AS TIMESTAMP_NTZ) as ORDER_DATE,
        TRY_TO_DECIMAL(totalamount, 10, 2) AS TOTAL_AMOUNT,
        TRY_CAST(status AS STRING) as STATUS,
        Try_Cast(paymentmethod as STRING) AS PAYMENT_METHOD,
        TRY_CAST(createddate AS TIMESTAMP_NTZ) AS created_dt,  -- Renamed column
        TRY_CAST(modifieddate AS TIMESTAMP_NTZ) AS modified_dt, -- Renamed column
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    FROM stage_sch.orders
) AS source
ON target.ORDER_ID = source.ORDER_ID
WHEN MATCHED AND (
        target.TOTAL_AMOUNT != source.TOTAL_AMOUNT or
        target.STATUS != source.STATUS or
```

```
        target.PAYMENT_METHOD != source.PAYMENT_METHOD or
        target.created_dt != source.created_dt or
        target.modified_dt != source.modified_dt
) THEN
    UPDATE SET
        target.TOTAL_AMOUNT = source.TOTAL_AMOUNT,
        target.STATUS = source.STATUS,
        target.PAYMENT_METHOD = source.PAYMENT_METHOD,
        target.created_dt = source.created_dt,
        target.modified_dt = source.modified_dt,
        target._stg_file_name = source._stg_file_name,
        target._stg_file_load_ts = source._stg_file_load_ts,
        target._stg_file_md5 = source._stg_file_md5,
        target._copy_data_ts = source._copy_data_ts
WHEN NOT MATCHED THEN
    INSERT (
        ORDER_ID,
        CUSTOMER_ID_FK,
        RESTAURANT_ID_FK,
        ORDER_DATE,
        TOTAL_AMOUNT,
        STATUS,
        PAYMENT_METHOD,
        created_dt,
        modified_dt,
        _STG_FILE_NAME,
        _STG_FILE_LOAD_TS,
        _STG_FILE_MD5,
        _COPY_DATA_TS
    )
    VALUES (
        source.ORDER_ID,
        source.CUSTOMER_ID_FK,
        source.RESTAURANT_ID_FK,
        source.ORDER_DATE,
        source.TOTAL_AMOUNT,
        source.STATUS,
        source.PAYMENT_METHOD,
        source.created_dt ,
        source.modified_dt,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );
ALTER TASK clean_sch.orders_task RESUME;

-- In next step I am trying to import new and updated record from delta folder and see
how my piepline words
 /* list @stage_sch.cvs_stg/delta/orders/; */
/* copy into stage_sch.orders (orderid, customerid, restaurantid, orderdate,
totalamount,
```

```
                  status, paymentmethod, createddate, modifieddate,
                  _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as orderid,
        t.$2::text as customerid,
        t.$3::text as restaurantid,
        t.$4::text as orderdate,
        t.$5::text as totalamount,
        t.$6::text as status,
        t.$7::text as paymentmethod,
        t.$8::text as createddate,
        t.$9::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
    from @stage_sch.cvs_stg/delta/orders/day-02-orders.csv t
)
file_format = (format_name = 'stage_sch.csv_file_format')
on_error = abort_statement; */
```

Order Item: Transaction table so not populating the dimension table as they represent as fact table.

```
use role sysadmin;
use database sandbox;
use schema stage_sch;
use warehouse compute_wh;


create or replace table stage_sch.orderitem (
    orderitemid text comment 'Primary Key (Source System)',              -- primary
key as text
    orderid text comment 'Order FK(Source System)',                  -- foreign key
reference as text (no constraint in snowflake)
    menuid text comment 'Menu FK(Source System)',                   -- foreign key
reference as text (no constraint in snowflake)
    quantity text,                -- quantity as text
    price text,                   -- price as text (no decimal constraint)
    subtotal text,                -- subtotal as text (no decimal constraint)
    createddate text,             -- created date as text
    modifieddate text,            -- modified date as text

    -- audit columns with appropriate data types
    _stg_file_name text,
    _stg_file_load_ts timestamp,
    _stg_file_md5 text,
    _copy_data_ts timestamp default current_timestamp
)
comment = 'This is the order item stage/raw table where data will be copied from
internal stage using copy command. This is as-is data represetation from the source
```

```
location. All the columns are text data type except the audit columns that are added
for traceability.';

-- Stream
create or replace stream stage_sch.orderitem_stm
on table stage_sch.orderitem
append_only = true
comment = 'This is the append-only stream object on order item table that only gets
delta data';

-- Pipe
CREATE OR REPLACE PIPE stage_sch.order_items_pipes
AUTO_INGEST = TRUE
INTEGRATION = Azure_Notification
AS
copy into stage_sch.orderitem (orderitemid, orderid, menuid, quantity, price,
                    subtotal, createddate, modifieddate,
                    _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as orderitemid,
        t.$2::text as orderid,
        t.$3::text as menuid,
        t.$4::text as quantity,
        t.$5::text as price,
        t.$6::text as subtotal,
        t.$7::text as createddate,
        t.$8::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
         from @azure_external_stage/order_items t
)
file_format=(format_name='stage_sch.csv_file_format')
ON_ERROR = 'CONTINUE'
PATTERN = '.*\.csv';


-- level2
CREATE OR REPLACE TABLE clean_sch.order_item (
    order_item_sk NUMBER AUTOINCREMENT primary key comment 'Surrogate Key (EDW)',    -
- Auto-incremented unique identifier for each order item
    order_item_id NUMBER  NOT NULL UNIQUE comment 'Primary Key (Source System)',
    order_id_fk NUMBER  NOT NULL comment 'Order FK(Source System)',                  -
- Foreign key reference for Order ID
    menu_id_fk NUMBER  NOT NULL comment 'Menu FK(Source System)',                    --
Foreign key reference for Menu ID
    quantity NUMBER(10, 2),                -- Quantity as a decimal number
    price NUMBER(10, 2),                   -- Price as a decimal number
    subtotal NUMBER(10, 2),                -- Subtotal as a decimal number
    created_dt TIMESTAMP,                -- Created date of the order item
```

```
    modified_dt TIMESTAMP,                -- Modified date of the order item

    -- Audit columns
    _stg_file_name VARCHAR(255),            -- File name of the staging file
    _stg_file_load_ts TIMESTAMP,            -- Timestamp when the file was loaded
    _stg_file_md5 VARCHAR(255),             -- MD5 hash of the file for integrity
check
    _copy_data_ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- Timestamp when data is copied
into the clean layer
)
comment = 'Order item entity under clean schema with appropriate data type under clean
schema layer, data is populated using merge statement from the stage layer location
table. This table does not support SCD2';

create or replace stream CLEAN_SCH.order_item_stm
on table CLEAN_SCH.order_item
comment = 'This is the stream object on order_item table table to track insert,
update, and delete changes';


CREATE OR REPLACE TASK clean_sch.order_items
WAREHOUSE = compute_wh
SCHEDULE = 'USING CRON 30 20 * * * UTC' -- event-driven
AS
MERGE INTO clean_sch.order_item AS target
USING (
    SELECT
        TRY_CAST(orderitemid AS Int) AS order_item_id,
        TRY_CAST(orderid as NUMBER) as order_id_fk,
        TRY_CAST(menuid  AS NUMBER) as menu_id_fk,
        TRY_CAST(quantity AS NUMBER(10, 2)) AS quantity,
        TRY_CAST(price AS NUMBER(10, 2)) as price,
        Try_Cast(subtotal as NUMBER(10, 2)) AS subtotal,
        TRY_CAST(createddate AS TIMESTAMP_NTZ) AS created_dt,  -- Renamed column
        TRY_CAST(modifieddate AS TIMESTAMP_NTZ) AS modified_dt, -- Renamed column
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    FROM stage_sch.orderitem_stm
    where orderitemid >=40
) AS source
ON target.order_item_id = source.order_item_id and
target.order_id_fk = source.order_id_fk and
target.menu_id_fk = source.menu_id_fk
WHEN MATCHED AND (
        target.quantity != source.quantity or
        target.price != source.price or
        target.subtotal != source.subtotal or
        target.created_dt != source.created_dt or
        target.modified_dt != source.modified_dt
) THEN
```

```
    UPDATE SET
        target.quantity = source.quantity,
        target.price = source.price,
        target.subtotal = source.subtotal,
        target.created_dt = source.created_dt,
        target.modified_dt = source.modified_dt,
        target._stg_file_name = source._stg_file_name,
        target._stg_file_load_ts = source._stg_file_load_ts,
        target._stg_file_md5 = source._stg_file_md5,
        target._copy_data_ts = source._copy_data_ts
WHEN NOT MATCHED THEN
    INSERT (
       order_item_id,
        order_id_fk,
        menu_id_fk,
        quantity,
        price,
        subtotal,
        created_dt,
        modified_dt,
        _stg_file_name,
        _stg_file_load_ts,
        _stg_file_md5,
        _copy_data_ts
    )
    VALUES (
       source.order_item_id,
        source.order_id_fk,
        source.menu_id_fk,
        source.quantity,
        source.price,
        source.subtotal,
        source.created_dt ,
        source.modified_dt,
        source._stg_file_name,
        source._stg_file_load_ts,
        source._stg_file_md5,
        source._copy_data_ts
    );

ALTER TASK clean_sch.order_items RESUME;

-- -- In next step I am trying to import new and updated record from delta folder and
see how my piepline words
/* list @stage_sch.cvs_stg/delta/order-item/; */

/* copy into stage_sch.orderitem (orderitemid, orderid, menuid, quantity, price,
                    subtotal, createddate, modifieddate,
                    _stg_file_name, _stg_file_load_ts, _stg_file_md5, _copy_data_ts)
from (
    select
        t.$1::text as orderitemid,
```

```
        t.$2::text as orderid,
        t.$3::text as menuid,
        t.$4::text as quantity,
        t.$5::text as price,
        t.$6::text as subtotal,
        t.$7::text as createddate,
        t.$8::text as modifieddate,
        metadata$filename as _stg_file_name,
        metadata$file_last_modified as _stg_file_load_ts,
        metadata$file_content_key as _stg_file_md5,
        current_timestamp as _copy_data_ts
    from @stage_sch.cvs_stg/delta/order-item/day-02-order-item.csv t
)
file_format = (format_name = 'stage_sch.csv_file_format')
on_error = abort_statement; */
```

Date: Using Common table experection to create Date Dimentation, using the minimum order_date from order table

```
use role sysadmin;
use warehouse compute_wh;
use database sandbox;
use schema CONSUMPTION_SCH;


CREATE OR REPLACE TABLE CONSUMPTION_SCH.DATE_DIM (
    DATE_DIM_HK NUMBER PRIMARY KEY comment 'Menu Dim HK (EDW)',   -- Surrogate key for
date dimension
    CALENDAR_DATE DATE UNIQUE,                    -- The actual calendar date
    YEAR NUMBER,                                  -- Year
    QUARTER NUMBER,                               -- Quarter (1-4)
    MONTH NUMBER,                                 -- Month (1-12)
    WEEK NUMBER,                                  -- Week of the year
    DAY_OF_YEAR NUMBER,                           -- Day of the year (1-365/366)
    DAY_OF_WEEK NUMBER,                           -- Day of the week (1-7)
    DAY_OF_THE_MONTH NUMBER,                      -- Day of the month (1-31)
    DAY_NAME STRING                               -- Name of the day (e.g., Monday)
)
comment = 'Date dimension table created using min of order data.';

insert into CONSUMPTION_SCH.DATE_DIM
with recursive my_date_dim_cte as
(
    -- anchor clause
    select
        current_date() as today,
        year(today) as year,
        quarter(today) as quarter,
        month(today) as month,
        week(today) as week,
        dayofyear(today) as day_of_year,
        dayofweek(today) as day_of_week,
```

```
        day(today) as day_of_the_month,
        dayname(today) as day_name

    union all

     -- recursive clause
    select
        dateadd('day', -1, today) as today_r,
        year(today_r) as year,
        quarter(today_r) as quarter,
        month(today_r) as month,
        week(today_r) as week,
        dayofyear(today_r) as day_of_year,
        dayofweek(today_r) as day_of_week,
        day(today_r) as day_of_the_month,
        dayname(today_r) as day_name
    from
        my_date_dim_cte
    where
        today_r > (select date(min(order_date)) from clean_sch.orders)
)
select
    hash(SHA1_hex(today)) as DATE_DIM_HK,
    today ,                     -- The actual calendar date
    YEAR,                               -- Year
    QUARTER,                            -- Quarter (1-4)
    MONTH,                              -- Month (1-12)
    WEEK,                               -- Week of the year
    DAY_OF_YEAR,                        -- Day of the year (1-365/366)
    DAY_OF_WEEK,                        -- Day of the week (1-7)
    DAY_OF_THE_MONTH,                   -- Day of the month (1-31)
    DAY_NAME
from my_date_dim_cte;
```

Order Fact: Using Common table experection to create Date Dimentation, using the minimum order_date from order table

```
use role sysadmin;
use warehouse compute_wh;
use database sandbox;
use schema CONSUMPTION_SCH;

-- Order Fact table
Create or replace table CONSUMPTION_SCH.order_item_fact (
    order_item_fact_hk number autoincrement comment 'Hash key for order item',
    order_item_id  number not null comment 'Order item FK',
    order_id number not null comment 'order FK',
    customer_dim_key  number not null comment 'Customer dim hash key',
    customer_address_dim_key  not null number,
    restaurant_dim_key  not null  number,
    restaurant_location_dim_key  not null  number,
    menu_dim_key  not null  number,
```

```
    delivery_agent_dim_key  not null number,
    order_date_dim_key  not null number,
    quantity  number,
    price  number(10,2),
    subtotal  number(10,2),
    delivery_status  varchar,
    estimated_time  varchar

)
comment='The item order fact table that has item level price, quantity, and other
details'

MERGE INTO CONSUMPTION_SCH.order_item_fact AS target
USING (
    SELECT
        oi.Order_Item_ID AS order_item_id,
        oi.Order_ID_fk AS order_id,
        c.CUSTOMER_HK AS customer_dim_key,
        ca.CUSTOMER_ADDRESS_HK AS customer_address_dim_key,
        r.RESTAURANT_HK AS restaurant_dim_key,
        rl.restaurant_location_hk as restaurant_location_dim_key,
        m.Menu_Dim_HK AS menu_dim_key,
        da.DELIVERY_AGENT_HK AS delivery_agent_dim_key,
        dd.DATE_DIM_HK AS order_date_dim_key,
        oi.Quantity::number(2) AS quantity,
        oi.Price AS price,
        oi.Subtotal AS subtotal,
        o.PAYMENT_METHOD,
        d.delivery_status AS delivery_status,
        d.estimated_time AS estimated_time,
    FROM
        clean_sch.order_item_stm oi
    JOIN
        clean_sch.orders_stm o ON oi.Order_ID_fk = o.Order_ID
    JOIN
        clean_sch.delivery_stm d ON o.Order_ID = d.Order_ID_fk
    JOIN
        consumption_sch.CUSTOMER_DIM c on o.Customer_ID_fk = c.customer_id
    JOIN
        consumption_sch.CUSTOMER_ADDRESS_DIM ca on c.Customer_ID = ca.CUSTOMER_ID_fk
    JOIN
        consumption_sch.restaurant_dim r on o.Restaurant_ID_fk = r.restaurant_id
    JOIN
        consumption_sch.menu_dim m ON oi.MENU_ID_fk = m.menu_id
    JOIN
        consumption_sch.delivery_agent_dim da ON d.Delivery_Agent_ID_fk =
da.delivery_agent_id
    JOIN
        consumption_sch.restaurant_location_dim rl on r.LOCATION_ID_FK =
rl.location_id
    JOIN
        CONSUMPTION_SCH.DATE_DIM dd on dd.calendar_date = date(o.order_date)
```

```
) AS source_stm
ON
    target.order_item_id = source_stm.order_item_id and
    target.order_id = source_stm.order_id
WHEN MATCHED THEN
    -- Update existing fact record
    UPDATE SET
        target.customer_dim_key = source_stm.customer_dim_key,
        target.customer_address_dim_key = source_stm.customer_address_dim_key,
        target.restaurant_dim_key = source_stm.restaurant_dim_key,
        target.restaurant_location_dim_key = source_stm.restaurant_location_dim_key,
        target.menu_dim_key = source_stm.menu_dim_key,
        target.delivery_agent_dim_key = source_stm.delivery_agent_dim_key,
        target.order_date_dim_key = source_stm.order_date_dim_key,
        target.quantity = source_stm.quantity,
        target.price = source_stm.price,
        target.subtotal = source_stm.subtotal,
        target.delivery_status = source_stm.delivery_status,
        target.estimated_time = source_stm.estimated_time
WHEN NOT MATCHED THEN
    -- Insert new fact record
    INSERT (
        order_item_id,
        order_id,
        customer_dim_key,
        customer_address_dim_key,
        restaurant_dim_key,
        restaurant_location_dim_key,
        menu_dim_key,
        delivery_agent_dim_key,
        order_date_dim_key,
        quantity,
        price,
        subtotal,
        delivery_status,
        estimated_time
    )
    VALUES (
        source_stm.order_item_id,
        source_stm.order_id,
        source_stm.customer_dim_key,
        source_stm.customer_address_dim_key,
        source_stm.restaurant_dim_key,
        source_stm.restaurant_location_dim_key,
        source_stm.menu_dim_key,
        source_stm.delivery_agent_dim_key,
        source_stm.order_date_dim_key,
        source_stm.quantity,
        source_stm.price,
        source_stm.subtotal,
        source_stm.delivery_status,
        source_stm.estimated_time
```

```
    );

-- Another way to write the merge statment
/*merge into CONSUMPTION_SCH.order_item_fact as target
using(
    select
TRY_CAST(o.order_item_id as Number) AS order_item_id,
TRY_CAST(o.order_id_fk AS NUMBER) AS order_id,
TRY_CAST(c.CUSTOMER_HK AS NUMBER) AS customer_dim_key,
TRY_CAST(c1.CUSTOMER_ADDRESS_HK AS NUMBER) AS customer_address_dim_key,
TRY_CAST(r.RESTAURANT_HK AS NUMBER) AS restaurant_dim_key,
TRY_CAST(r1.restaurant_location_hk AS NUMBER) AS restaurant_location_dim_key,
TRY_CAST(m.Menu_Dim_HK AS NUMBER) AS menu_dim_key,
TRY_CAST(da.DELIVERY_AGENT_HK AS NUMBER) AS delivery_agent_dim_key,
TRY_CAST(d3.Date_Dim_HK AS NUMBER) AS order_date_dim_key,
TRY_CAST(o.quantity AS NUMBER(10, 2)) AS quantity,
TRY_CAST(o.price AS NUMBER(10, 2)) as price,
Try_Cast(o.subtotal as NUMBER(10, 2)) AS subtotal,
TRY_CAST(o1.Payment_method AS VARCHAR) AS payment_method,
TRY_CAST(d.delivery_status AS VARCHAR) AS delivery_status,
TRY_CAST(d.estimated_time AS VARCHAR) AS estimated_time

    from clean_sch.order_item_stm as o

     join clean_sch.orders_stm as o1 on o1.Order_ID=o.order_id_fk
     join clean_sch.delivery_stm as d on d.order_id_fk=O1.Order_ID
     join consumption_sch.customer_dim as c on o1.CUSTOMER_ID_FK= c.CUSTOMER_ID
     join consumption_sch.customer_address_dim as c1 on
c1.CUSTOMER_ID_FK=c.CUSTOMER_ID
     join consumption_sch.restaurant_dim as r on o1.RESTAURANT_ID_FK=r.RESTAURANT_ID
     join consumption_sch.menu_dim as m on m.menu_id = o.menu_id_fk
     join consumption_sch.delivery_agent_dim as da on
da.delivery_agent_id=d.delivery_agent_id_fk
     join consumption_sch.restaurant_location_dim r1 on
r.location_id_fk=r1.LOCATION_ID
     join consumption_sch.date_dim as d3 on d3.calendar_date= date(o1.order_date)) as
source
on target.order_item_id=source.order_item_id and
target.order_id = source.order_id
WHEN MATCHED AND (
        target.customer_dim_key != source.customer_dim_key or
        target.customer_address_dim_key != source.customer_address_dim_key or
        target.restaurant_dim_key != source.restaurant_dim_key or
        target.restaurant_location_dim_key != source.restaurant_location_dim_key or
        target.menu_dim_key != source.menu_dim_key or
        target.delivery_agent_dim_key != source.delivery_agent_dim_key or
        target.order_date_dim_key != source.order_date_dim_key or
        target.quantity != source.quantity or
        target.price != source.price or
        target.subtotal != source.subtotal or
        target.delivery_status != source.delivery_status or
        target.estimated_time != source.estimated_time
```

```
) THEN
    UPDATE SET
         target.customer_dim_key = source.customer_dim_key,
        target.customer_address_dim_key = source.customer_address_dim_key,
        target.restaurant_dim_key = source.restaurant_dim_key,
        target.restaurant_location_dim_key = source.restaurant_location_dim_key,
        target.menu_dim_key = source.menu_dim_key,
        target.delivery_agent_dim_key = source.delivery_agent_dim_key,
        target.order_date_dim_key = source.order_date_dim_key,
        target.quantity = source.quantity,
        target.price = source.price,
        target.subtotal = source.subtotal,
        target.delivery_status = source.delivery_status,
        target.estimated_time = source.estimated_time
WHEN NOT MATCHED THEN
    -- Insert new fact record
    INSERT (
    order_item_id,
        order_id,
        customer_dim_key,
        customer_address_dim_key,
        restaurant_dim_key,
        restaurant_location_dim_key,
        menu_dim_key,
        delivery_agent_dim_key,
        order_date_dim_key,
        quantity,
        price,
        subtotal,
        delivery_status,
        estimated_time
    )
    VALUES (
        source.order_item_id,
        source.order_id,
        source.customer_dim_key,
        source.customer_address_dim_key,
        source.restaurant_dim_key,
        source.restaurant_location_dim_key,
        source.menu_dim_key,
        source.delivery_agent_dim_key,
        source.order_date_dim_key,
        source.quantity,
        source.price,
        source.subtotal,
        source.delivery_status,
        source.estimated_time

);*/
```

Connections: Helping use to create relationship between Entity and Fact

```
-- start with
alter table consumption_sch.order_item_fact
    add constraint fk_order_item_fact_customer_dim
    foreign key (customer_dim_key)
    references consumption_sch.customer_dim (customer_hk);

alter table consumption_sch.order_item_fact
    add constraint fk_order_item_fact_customer_address_dim
    foreign key (customer_address_dim_key)
    references consumption_sch.customer_address_dim (CUSTOMER_ADDRESS_HK);

alter table consumption_sch.order_item_fact
    add constraint fk_order_item_fact_restaurant_dim
    foreign key (restaurant_dim_key)
    references consumption_sch.restaurant_dim (restaurant_hk);

alter table consumption_sch.order_item_fact
    add constraint fk_order_item_fact_restaurant_location_dim
    foreign key (restaurant_location_dim_key)
    references consumption_sch.restaurant_location_dim (restaurant_location_hk);

alter table consumption_sch.order_item_fact
    add constraint fk_order_item_fact_menu_dim
    foreign key (menu_dim_key)
    references consumption_sch.menu_dim (menu_dim_hk);

alter table consumption_sch.order_item_fact
    add constraint fk_order_item_fact_delivery_agent_dim
    foreign key (delivery_agent_dim_key)
    references consumption_sch.delivery_agent_dim (delivery_agent_hk);

alter table consumption_sch.order_item_fact
    add constraint fk_order_item_fact_delivery_date_dim
    foreign key (order_date_dim_key)
    references consumption_sch.date_dim (date_dim_hk);
```

Views:

```
use role sysadmin;
use warehouse adhoc_wh;
use database sandbox;
use schema consumption_sch;

select * from consumption_sch.order_item_fact
where DELIVERY_STATUS='Delivered';

create or replace view consumption_sch.vw_yearly_revenue_kpis as
select
    d.year as year, -- fetch year from date_dim
    sum(fact.subtotal) as total_revenue,
    count(distinct fact.order_id) as total_orders,
    round(sum(fact.subtotal) / count(distinct fact.order_id), 2) as
```

```
avg_revenue_per_order,
    round(sum(fact.subtotal) / count(fact.order_item_id), 2) as avg_revenue_per_item,
    max(fact.subtotal) as max_order_value
from
    consumption_sch.order_item_fact fact
join
    consumption_sch.date_dim d
on
    fact.order_date_dim_key = d.date_dim_hk -- join fact table with date_dim table
where DELIVERY_STATUS = 'Delivered'
group by
    d.year
order by
    d.year;

select * from  consumption_sch.vw_yearly_revenue_kpis;

-- scd2 safe
CREATE OR REPLACE VIEW consumption_sch.vw_yearly_revenue_kpis2 AS
WITH dedup_order_items AS (
    SELECT
        order_item_id,
        order_id,
        order_date_dim_key,
        MAX(subtotal) AS item_subtotal  -- pick one version per item
    FROM consumption_sch.order_item_fact
    WHERE DELIVERY_STATUS = 'Delivered'
    GROUP BY order_item_id, order_id, order_date_dim_key
)
SELECT
    d.year AS year,
    SUM(o.item_subtotal) AS total_revenue,
    COUNT(DISTINCT o.order_id) AS total_orders,
    ROUND(SUM(o.item_subtotal) / COUNT(DISTINCT o.order_id), 2) AS
avg_revenue_per_order,
    ROUND(SUM(o.item_subtotal) / COUNT(*), 2) AS avg_revenue_per_item,
    MAX(o.item_subtotal) AS max_order_value
FROM dedup_order_items o
JOIN consumption_sch.date_dim d
    ON o.order_date_dim_key = d.date_dim_hk
GROUP BY d.year
ORDER BY d.year;

/*WITH dedup_order_items AS (
    SELECT
        order_item_id,
        order_id,
        order_date_dim_key,
        MAX(subtotal) AS item_subtotal  -- pick one version per item
    FROM consumption_sch.order_item_fact
    WHERE DELIVERY_STATUS = 'Delivered'
    GROUP BY order_item_id, order_id, order_date_dim_key
```

```
)
SELECT
    d.year AS year,
    SUM(o.item_subtotal) AS total_revenue
FROM dedup_order_items o
JOIN consumption_sch.date_dim d
    ON o.order_date_dim_key = d.date_dim_hk
GROUP BY d.year
ORDER BY d.year;*/

/*SELECT
    d.year AS year,
    SUM(fact.subtotal) AS total_revenue
FROM consumption_sch.order_item_fact fact
JOIN consumption_sch.date_dim d
    ON fact.order_date_dim_key = d.date_dim_hk
WHERE DELIVERY_STATUS = 'Delivered'
GROUP BY d.year
ORDER BY d.year;*/

select * from consumption_sch.vw_yearly_revenue_kpis2;

CREATE OR REPLACE VIEW consumption_sch.vw_monthly_revenue_kpis AS
SELECT
    d.YEAR AS year,                      -- Fetch year from DATE_DIM
    d.MONTH AS month,                    -- Fetch month from DATE_DIM
    SUM(fact.subtotal) AS total_revenue,
    COUNT(DISTINCT fact.order_id) AS total_orders,
    ROUND(SUM(fact.subtotal) / COUNT(DISTINCT fact.order_id), 2) AS
avg_revenue_per_order,
    ROUND(SUM(fact.subtotal) / COUNT(fact.order_item_id), 2) AS avg_revenue_per_item,
    MAX(fact.subtotal) AS max_order_value
FROM
    consumption_sch.order_item_fact fact
JOIN
    consumption_sch.DATE_DIM d
ON
    fact.order_date_dim_key = d.DATE_DIM_HK -- Join fact table with DATE_DIM table
where DELIVERY_STATUS = 'Delivered'
GROUP BY
    d.YEAR, d.MONTH
ORDER BY
    d.YEAR, d.MONTH;

/*WITH dedup_order_items AS (
    SELECT
        order_item_id,
        order_id,
        order_date_dim_key,
        MAX(subtotal) AS item_subtotal
    FROM consumption_sch.order_item_fact
    WHERE DELIVERY_STATUS = 'Delivered' AND order_id = 14
```

```
    GROUP BY order_item_id, order_id, order_date_dim_key
)
SELECT
    d.year AS year,
    SUM(o.item_subtotal) AS total_revenue,
    COUNT(DISTINCT o.order_id) AS total_orders,
    ROUND(SUM(o.item_subtotal) / COUNT(DISTINCT o.order_id), 2) AS
avg_revenue_per_order,
    ROUND(SUM(o.item_subtotal) / COUNT(*), 2) AS avg_revenue_per_item,
    MAX(o.item_subtotal) AS max_order_value
FROM dedup_order_items o
JOIN consumption_sch.date_dim d
    ON o.order_date_dim_key = d.date_dim_hk
GROUP BY d.year
ORDER BY d.year;*/

select * from consumption_sch.vw_monthly_revenue_kpis;

CREATE OR REPLACE VIEW consumption_sch.vw_daily_revenue_kpis AS
SELECT
    d.YEAR AS year,                       -- Fetch year from DATE_DIM
    d.MONTH AS month,                     -- Fetch month from DATE_DIM
    d.DAY_OF_THE_MONTH AS day,            -- Fetch day from DATE_DIM
    SUM(fact.subtotal) AS total_revenue,
    COUNT(DISTINCT fact.order_id) AS total_orders,
    ROUND(SUM(fact.subtotal) / COUNT(DISTINCT fact.order_id), 2) AS
avg_revenue_per_order,
    ROUND(SUM(fact.subtotal) / COUNT(fact.order_item_id), 2) AS avg_revenue_per_item,
    MAX(fact.subtotal) AS max_order_value
FROM
    consumption_sch.order_item_fact fact
JOIN
    consumption_sch.DATE_DIM d
ON
    fact.order_date_dim_key = d.DATE_DIM_HK -- Join fact table with DATE_DIM table
    where DELIVERY_STATUS = 'Delivered'
GROUP BY
    d.YEAR, d.MONTH, d.DAY_OF_THE_MONTH     -- Group by year, month, and day
ORDER BY
    d.YEAR, d.MONTH, d.DAY_OF_THE_MONTH;    -- Order by year, month, and day

select * from consumption_sch.vw_daily_revenue_kpis;

CREATE OR REPLACE VIEW consumption_sch.vw_day_revenue_kpis AS
SELECT
    d.YEAR AS year,                       -- Fetch year from DATE_DIM
    d.MONTH AS month,                     -- Fetch month from DATE_DIM
    d.DAY_NAME AS DAY_NAME,                -- Fetch day from DATE_DIM-DAY_NAME
    SUM(fact.subtotal) AS total_revenue,
    COUNT(DISTINCT fact.order_id) AS total_orders,
    ROUND(SUM(fact.subtotal) / COUNT(DISTINCT fact.order_id), 2) AS
avg_revenue_per_order,
```

```
    ROUND(SUM(fact.subtotal) / COUNT(fact.order_item_id), 2) AS avg_revenue_per_item,
    MAX(fact.subtotal) AS max_order_value
FROM
    consumption_sch.order_item_fact fact
JOIN
    consumption_sch.DATE_DIM d
ON
    fact.order_date_dim_key = d.DATE_DIM_HK -- Join fact table with DATE_DIM table
GROUP BY
    d.YEAR, d.MONTH, d.DAY_NAME     -- Group by year, month, and day
ORDER BY
    d.YEAR, d.MONTH, d.DAY_NAME;    -- Order by year, month, and day

select * from consumption_sch.vw_day_revenue_kpis;

CREATE OR REPLACE VIEW consumption_sch.vw_monthly_revenue_by_restaurant AS
SELECT
    d.YEAR AS year,                       -- Fetch year from DATE_DIM
    d.MONTH AS month,                     -- Fetch month from DATE_DIM
    fact.DELIVERY_STATUS,
    r.name as restaurant_name,
    SUM(fact.subtotal) AS total_revenue,
    COUNT(DISTINCT fact.order_id) AS total_orders,
    ROUND(SUM(fact.subtotal) / COUNT(DISTINCT fact.order_id), 2) AS
avg_revenue_per_order,
    ROUND(SUM(fact.subtotal) / COUNT(fact.order_item_id), 2) AS avg_revenue_per_item,
    MAX(fact.subtotal) AS max_order_value
FROM
    consumption_sch.order_item_fact fact
JOIN
    consumption_sch.DATE_DIM d
ON
    fact.order_date_dim_key = d.DATE_DIM_HK
JOIN
    consumption_sch.restaurant_dim r
ON
    fact.restaurant_dim_key = r.RESTAURANT_HK
GROUP BY
    d.YEAR, d.MONTH,fact.DELIVERY_STATUS,restaurant_name
ORDER BY
    d.YEAR, d.MONTH;

select * from consumption_sch.vw_monthly_revenue_by_restaurant;
select * from consumption_sch.customer_address_dim;

CREATE OR REPLACE VIEW consumption_sch.vv_revenue_by_state as
select cd.CITY as city,
        d.YEAR as  year,-- fetch year from date_dim
    sum(fact.subtotal) as total_revenue,
    count(distinct fact.order_id) as total_orders,
    round(sum(fact.subtotal) / count(distinct fact.order_id), 2) as
avg_revenue_per_order,
```

```
    round(sum(fact.subtotal) / count(fact.order_item_id), 2) as avg_revenue_per_item,
    max(fact.subtotal) as max_order_value
FROM CONSUMPTION_SCH.order_item_fact fact
join consumption_sch.date_dim d
on fact.order_date_dim_key = d.DATE_DIM_HK
join consumption_sch.customer_address_dim cd
on fact.customer_address_dim_key=cd.CUSTOMER_ADDRESS_HK
GROUP BY cd.CITY, d.YEAR
ORDER BY d.YEAR, cd.CITY;


select * from consumption_sch.order_item_fact;

/*select m.category, count(distinct o.order_id) as order_count, SUM(o.subtotal) AS
total_revenue  from consumption_sch.menu_dim as m
join consumption_sch.order_item_fact as o
on m.menu_dim_hk=o.menu_dim_key
where o.DELIVERY_STATUS = 'Delivered'
group by m.category
order by order_count desc*/

/*WITH order_totals AS (
    SELECT order_id, menu_dim_key, SUM(subtotal) AS order_total
    FROM consumption_sch.order_item_fact
    WHERE DELIVERY_STATUS = 'Delivered'
    GROUP BY order_id, menu_dim_key
)
SELECT m.category,
       COUNT(DISTINCT o.order_id) AS order_count,
       SUM(o.order_total) AS total_revenue
FROM consumption_sch.menu_dim m
JOIN order_totals o
    ON m.menu_dim_hk = o.menu_dim_key
GROUP BY m.category
ORDER BY order_count DESC;*/
```

Stream Lit: Python code in snowflake to display dashboard

```
import streamlit as st
import pandas as pd
import altair as alt
from snowflake.snowpark.context import get_active_session

# App Title
st.title("Revenue Dashboard")

# Get the current credentials
session = get_active_session()


def format_revenue(revenue):
    #return f"{revenue / 1_000_000:.1f}M"
```

```
    return f"{revenue:.1f}"

# Function to alternate row colors
def highlight_rows(row):
    color = '#f2f2f2' if row.name % 2 == 0 else 'white'  # Alternate rows
    return ['background-color: {}'.format(color)] * len(row)

# Function to fetch KPI data from Snowflake
def fetch_kpi_data():
    query = """
    SELECT
        year,
        total_revenue,
        total_orders,
        avg_revenue_per_order,
        avg_revenue_per_item,
        max_order_value
    FROM sandbox.consumption_sch.vw_yearly_revenue_kpis
    ORDER BY year;
    """
    return session.sql(query).collect()

#TO_CHAR(TO_DATE(month::text, 'MM'), 'Mon') AS month_abbr,  -- Converts month number
to abbreviated month name
def fetch_monthly_kpi_data(year):
    query = f"""
    SELECT
        month::number(2) as month,
        total_revenue::NUMBER(10) AS TOTAL_REVENUE
    FROM
    sandbox.consumption_sch.vw_monthly_revenue_kpis
    WHERE year = {year}
    ORDER BY month;
    """
    return session.sql(query).collect()


def fetch_unique_months(year):
    query = f"""
    SELECT
        DISTINCT MONTH FROM
    sandbox.consumption_sch.vw_monthly_revenue_by_restaurant
    WHERE YEAR = {year}
    ORDER BY MONTH;
    """
    return session.sql(query).collect()

def fetch_top_restaurants(year, month):
    query = f"""
    SELECT
        restaurant_name,
        total_revenue,
```

```
        total_orders,
        avg_revenue_per_order,
        avg_revenue_per_item,
        max_order_value
    FROM
        sandbox.consumption_sch.vw_monthly_revenue_by_restaurant
    WHERE
        YEAR = {year}
        AND MONTH = {month}
    ORDER BY
        total_revenue DESC
    LIMIT 10;
    """
    return session.sql(query).collect()

# Function to convert Snowpark DataFrame to Pandas DataFrame
def snowpark_to_pandas(snowpark_df):
    return pd.DataFrame(
        snowpark_df,
        columns=[
            'Restaurant Name',
            'Total Revenue ( )',
            'Total Orders',
            'Avg Revenue per Order ( )',
            'Avg Revenue per Item ( )',
            'Max Order Value ( )'
        ]
    )
# Fetch data
sf_df = fetch_kpi_data()
df = pd.DataFrame(
    sf_df,
    columns=
['YEAR','TOTAL_REVENUE','TOTAL_ORDERS','AVG_REVENUE_PER_ORDER','AVG_REVENUE_PER_ITEM','M

)

# Aggregate Metrics for All Years
#st.subheader("Aggregate KPIs: Overall Performance")
col1, col2, col3 = st.columns(3)

with col1:
    st.metric("Total Revenue (All Years)", format_revenue(df['TOTAL_REVENUE'].sum()))
with col2:
    st.metric("Total Orders (All Years)", f"{df['TOTAL_ORDERS'].sum():,}")
with col3:
    st.metric("Max Order Value (Overall)", f" {df['MAX_ORDER_VALUE'].max():,.0f}")

st.divider()

# Year Selection Box
years = df["YEAR"].unique()
```

```
default_year = max(years)  # Select the most recent year by default
selected_year = st.selectbox("Select Year", sorted(years),
index=list(years).index(default_year))

# Filter data for selected year
year_data = df[df["YEAR"] == selected_year]
total_revenue = year_data["TOTAL_REVENUE"].iloc[0]
total_orders = year_data["TOTAL_ORDERS"].iloc[0]
avg_revenue_per_order = year_data["AVG_REVENUE_PER_ORDER"].iloc[0]
avg_revenue_per_item = year_data["AVG_REVENUE_PER_ITEM"].iloc[0]
max_order_value = year_data["MAX_ORDER_VALUE"].iloc[0]

# Get previous year data
previous_year = selected_year - 1
previous_year_data = df[df["YEAR"] == previous_year]

# If previous year data exists, calculate differences
if not previous_year_data.empty:
    prev_total_revenue = previous_year_data["TOTAL_REVENUE"].iloc[0]
    prev_total_orders = previous_year_data["TOTAL_ORDERS"].iloc[0]
    prev_avg_revenue_per_order = previous_year_data["AVG_REVENUE_PER_ORDER"].iloc[0]
    prev_avg_revenue_per_item = previous_year_data["AVG_REVENUE_PER_ITEM"].iloc[0]
    prev_max_order_value = previous_year_data["MAX_ORDER_VALUE"].iloc[0]

    # Calculate differences
    revenue_diff = total_revenue - prev_total_revenue
    orders_diff = total_orders - prev_total_orders
    avg_rev_order_diff = avg_revenue_per_order - prev_avg_revenue_per_order
    avg_rev_item_diff = avg_revenue_per_item - prev_avg_revenue_per_item
    max_order_diff = max_order_value - prev_max_order_value
else:
    # If previous year data is not found, set differences to None or zero
    revenue_diff = orders_diff = avg_rev_order_diff = avg_rev_item_diff =
max_order_diff = None


# Display Metrics for Selected Year with Comparison to Previous Year
# st.subheader(f"KPI Scorecard for {selected_year}")
col1, col2, col3 = st.columns(3)

with col1:
    st.metric(
        "Total Revenue",
        format_revenue(total_revenue),
        delta=f"{revenue_diff:.1f}" if revenue_diff is not None else None
    )
    st.metric("Total Orders", f"{total_orders:,}", delta=f"{orders_diff:,}" if
orders_diff is not None else None)

    #st.metric("Total Revenue", f"{total_revenue:,.0f}", delta=f"
{revenue_diff:,.0f}" if revenue_diff is not None else None)
    #st.metric("Total Orders", f"{total_orders:,}", delta=f"{orders_diff:,}" if
```

```
orders_diff is not None else None)

with col2:
    st.metric("Avg Revenue per Order", f"₹{avg_revenue_per_order:,.0f}", delta=f"₹
{avg_rev_order_diff:,.0f}" if avg_rev_order_diff is not None else None)
    st.metric("Avg Revenue per Item", f"₹{avg_revenue_per_item:,.0f}", delta=f"₹
{avg_rev_item_diff:,.0f}" if avg_rev_item_diff is not None else None)

with col3:
    st.metric("Max Order Value", f"₹{max_order_value:,.0f}", delta=f"₹
{max_order_diff:,.0f}" if max_order_diff is not None else None)



st.divider()
# ----------------------------------------


# Fetch and prepare data
month_sf_df = fetch_monthly_kpi_data(selected_year)
month_df = pd.DataFrame(
    month_sf_df,
    columns=['Month', 'Total Monthly Revenue']
)

# Map numeric months to abbreviated month names
month_mapping = {
    1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr',
    5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug',
    9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'
}
month_df['Month'] = month_df['Month'].map(month_mapping)

# Ensure months are in the correct chronological order
month_df['Month'] = pd.Categorical(
    month_df['Month'],
    categories=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',
'Nov', 'Dec'],
    ordered=True
)
month_df = month_df.sort_values('Month')  # Sort by chronological month order

# Convert revenue to millions
month_df['Total Monthly Revenue'] = month_df['Total Monthly Revenue']

# Plot Monthly Revenue Trend using Bar Chart
st.subheader(f"{selected_year} - Monthly Revenue Trend")
# Create the Altair Bar Chart with Custom Color
bar_chart = alt.Chart(month_df).mark_bar(color="#ff5200").encode(
    x=alt.X('Month', sort='ascending', title='Month'),
    y=alt.Y('Total Monthly Revenue', title='Revenue (₹)')
).properties(
```

```
    width=700,
    height=400
)

# Display the chart in Streamlit
st.altair_chart(bar_chart, use_container_width=True)

# Add a Trending Chart using Altair
st.subheader(f"{selected_year} - Monthly Revenue Trend")

trend_chart = alt.Chart(month_df).mark_line(color="#ff5200",
point=alt.OverlayMarkDef(color="#ff5200")).encode(
    x=alt.X('Month', sort='ascending', title='Month'),
    y=alt.Y('Total Monthly Revenue', title='Revenue ( )', scale=alt.Scale(domain=[0,
month_df['Total Monthly Revenue'].max()])),
    tooltip=[
        alt.Tooltip('Month', title='Month'),
        alt.Tooltip('Total Monthly Revenue', title='Revenue ( M)', format='.2f')  #
Format to 2 decimal places
    ]
).properties(
    width=700,
    height=400
).configure_point(
    size=60
)

st.altair_chart(trend_chart, use_container_width=True)

# Month Selection based on the selected year
if selected_year:

    #get the unique months
    month_sf_df = fetch_unique_months(selected_year)
    print(month_sf_df)
    #convert into df
    month_df = pd.DataFrame(
        month_sf_df,
        columns=['MONTH']
    )
    print(month_df)

    # Year Selection Box
    months = month_df["MONTH"].unique()
    default_month = max(months)  # Select the most recent year by default
    selected_month = st.selectbox(f"Select Month For {selected_year}", sorted(months),
index=list(months).index(default_month))

    # Fetch and Display Data
    if selected_month:
        st.subheader(f"Top 10 Restaurants for {selected_month}/{selected_year}")
        top_restaurants = fetch_top_restaurants(selected_year, selected_month)
```

```
        if top_restaurants:
            top_restaurants_df = snowpark_to_pandas(top_restaurants)
            # Remove index from DataFrame by resetting it and dropping the index
column
            #top_restaurants_df_reset = top_restaurants_df.reset_index(drop=True)

            # Display the DataFrame without index
            #st.dataframe(top_restaurants_df_reset)
            #st.dataframe(top_restaurants_df)

            # Apply the alternate color style
            styled_df = top_restaurants_df.style.apply(highlight_rows, axis=1)

            # Display the styled DataFrame
            st.dataframe(styled_df, hide_index= True)
        else:
            st.warning("No data found for the selected year and month.")
```