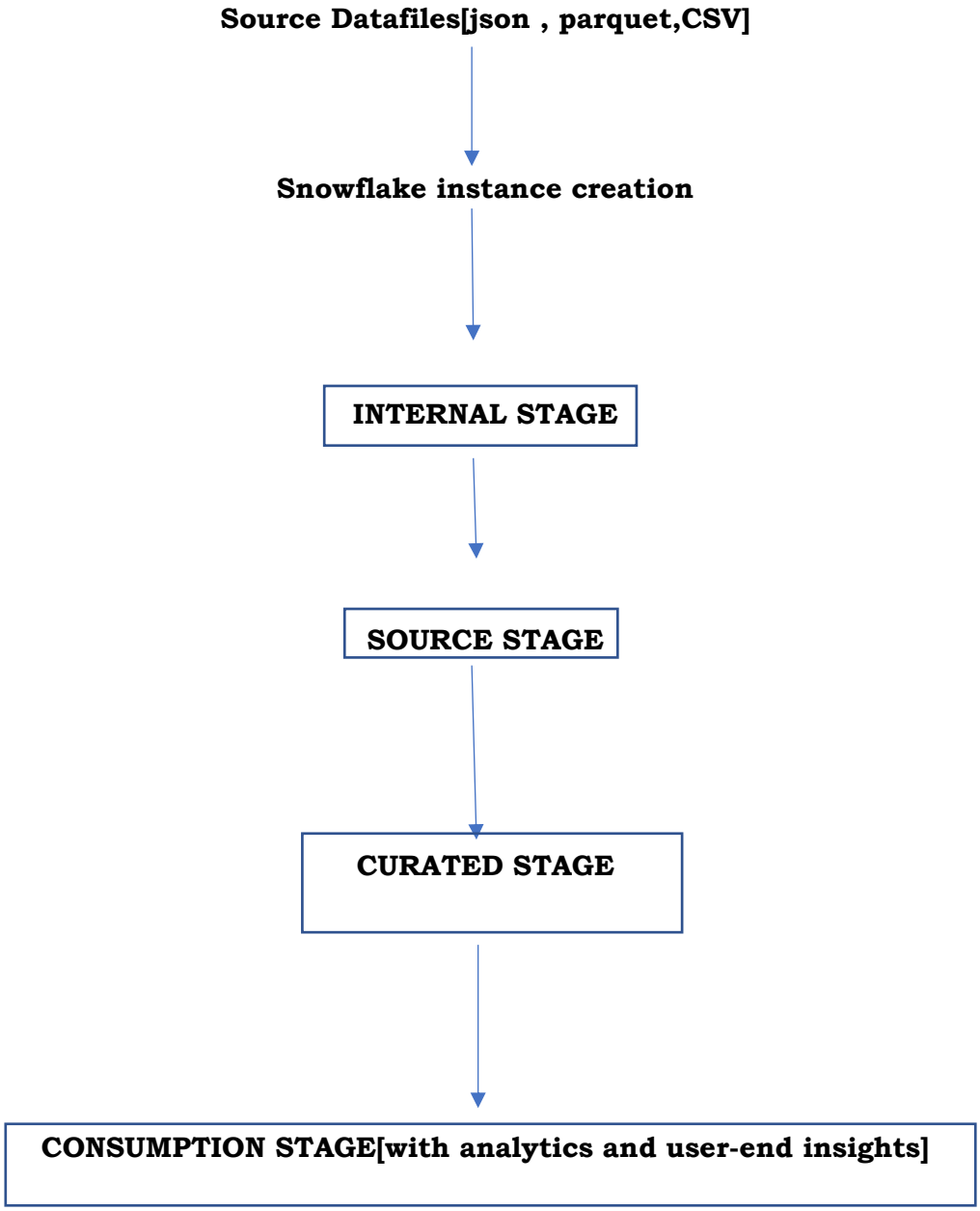
PROCEEDINGS OF THE PROJECT

This project foresees the ingestion of datatsets of different forms like parquet from US amazon site , csv from amazon IN and json from amazon France.

And I had tried to do this by exploring an end-to-end ETL (Extract, Transform, Load) data flow using the powerful combination of Snowpark and Snowflake.

The end2end ETL flow is something as follows:



Loading the datasets into out snowflake instance

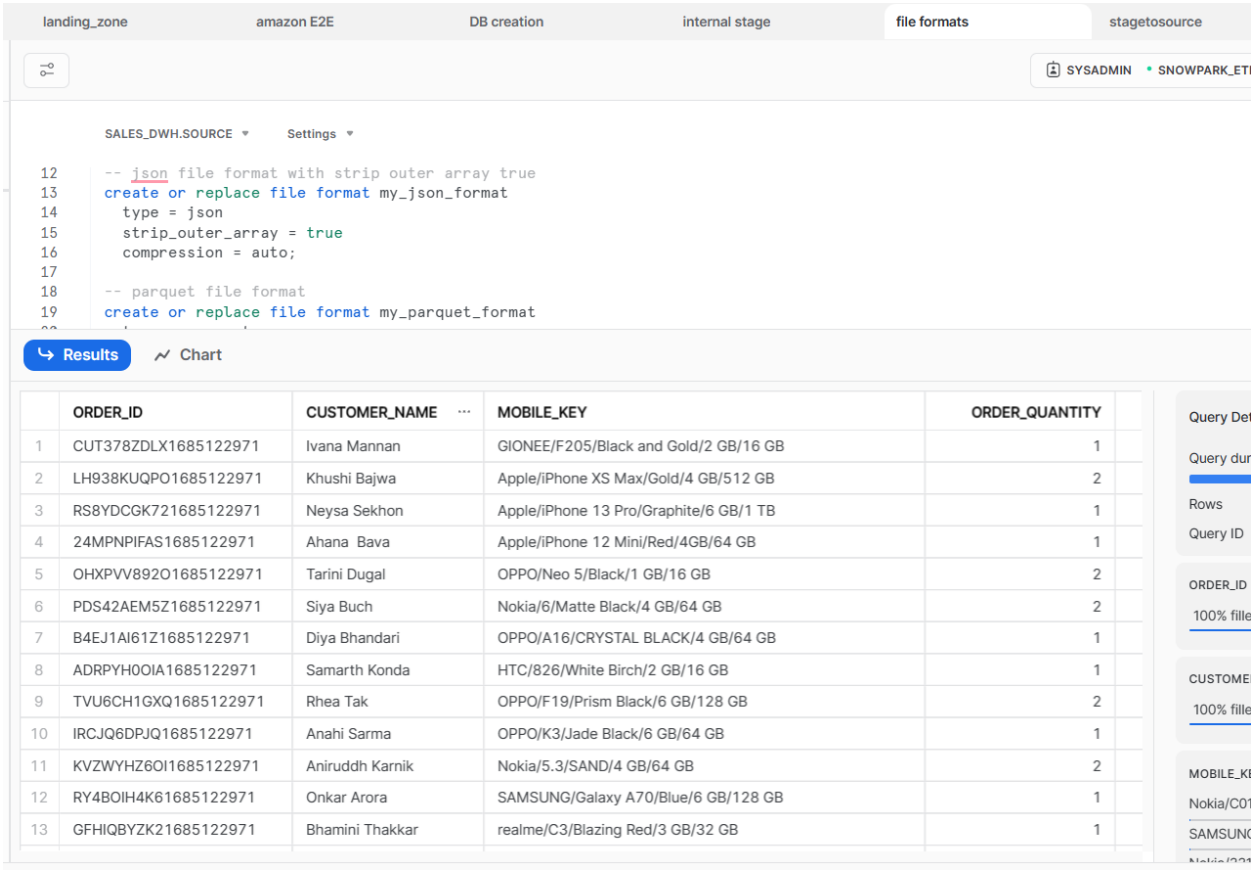
```
C:\Users\LENOVO>snowsql -a ek87911.ap-southeast-1 -u snowpark_user
Password:
* SnowSQL * v1.2.27
Type SQL statements or !help
snowpark_user#(no warehouse)@(no database).(no schema)>use role SYSADMIN;
+----------------------------------+
| status                           |
|----------------------------------|
| Statement executed successfully. |
+----------------------------------+
1 Row(s) produced. Time Elapsed: 0.701s
snowpark_user#(no warehouse)@(no database).(no schema)>use SNOWFLAKE_SAMPLE_DATA;
+----------------------------------+
| status                           |
|----------------------------------|
| Statement executed successfully. |
+----------------------------------+
1 Row(s) produced. Time Elapsed: 0.165s
snowpark_user#(no warehouse)@SNOWFLAKE_SAMPLE_DATA.(no schema)>use SCHEMA
snowpark_user#(no warehouse)@SNOWFLAKE_SAMPLE_DATA.(no schema)>use SCHEMA TPCH_SF1;
+----------------------------------+
| status                           |
|----------------------------------|
| Statement executed successfully. |
+----------------------------------+
1 Row(s) produced. Time Elapsed: 0.151s
snowpark_user#(no warehouse)@SNOWFLAKE_SAMPLE_DATA.TPCH_SF1>use WAREHOUSE SNOWPARK_ETL_WH;
+----------------------------------+
| status                           |
|----------------------------------|
| Statement executed successfully. |
+----------------------------------+
1 Row(s) produced. Time Elapsed: 0.141s
```

```
snowpark_user#SNOWPARK_ETL_WH@SNOWFLAKE_SAMPLE_DATA.TPCH_SF1>use database SALES_DWH;
+----------------------------------+
| status                           |
|----------------------------------|
| Statement executed successfully. |
+----------------------------------+
1 Row(s) produced. Time Elapsed: 0.163s
snowpark_user#SNOWPARK_ETL_WH@SALES_DWH.PUBLIC>use schema SOURCE;
+----------------------------------+
| status                           |
|----------------------------------|
| Statement executed successfully. |
+----------------------------------+
1 Row(s) produced. Time Elapsed: 0.129s
snowpark_user#SNOWPARK_ETL_WH@SALES_DWH.SOURCE>create or replace stage my_internal_stg;
+-----------------------------------------------+
| status                                        |
|-----------------------------------------------|
| Stage area MY_INTERNAL_STG successfully created. |
+-----------------------------------------------+
1 Row(s) produced. Time Elapsed: 1.013s
snowpark_user#SNOWPARK_ETL_WH@SALES_DWH.SOURCE>desc stage my_internal_stg;
+-------------------+-------------------------------+---------------+-----------------+-------------------+
| parent_property   | property                      | property_type | property_value  | property_default  |
|-------------------+-------------------------------+---------------+-----------------+-------------------|
| STAGE_FILE_FORMAT | TYPE                          | String        | CSV             | CSV               |
| STAGE_FILE_FORMAT | RECORD_DELIMITER              | String        | \n              | \n                |
| STAGE_FILE_FORMAT | FIELD_DELIMITER               | String        | ,               | ,                 |
| STAGE_FILE_FORMAT | FILE_EXTENSION                | String        |                 |                   |
| STAGE_FILE_FORMAT | SKIP_HEADER                   | Integer       | 0               | 0                 |
| STAGE_FILE_FORMAT | PARSE_HEADER                  | Boolean       | false           | false             |
| STAGE_FILE_FORMAT | DATE_FORMAT                   | String        | AUTO            | AUTO              |
| STAGE_FILE_FORMAT | TIME_FORMAT                   | String        | AUTO            | AUTO              |
| STAGE_FILE_FORMAT | TIMESTAMP_FORMAT              | String        | AUTO            | AUTO              |
```

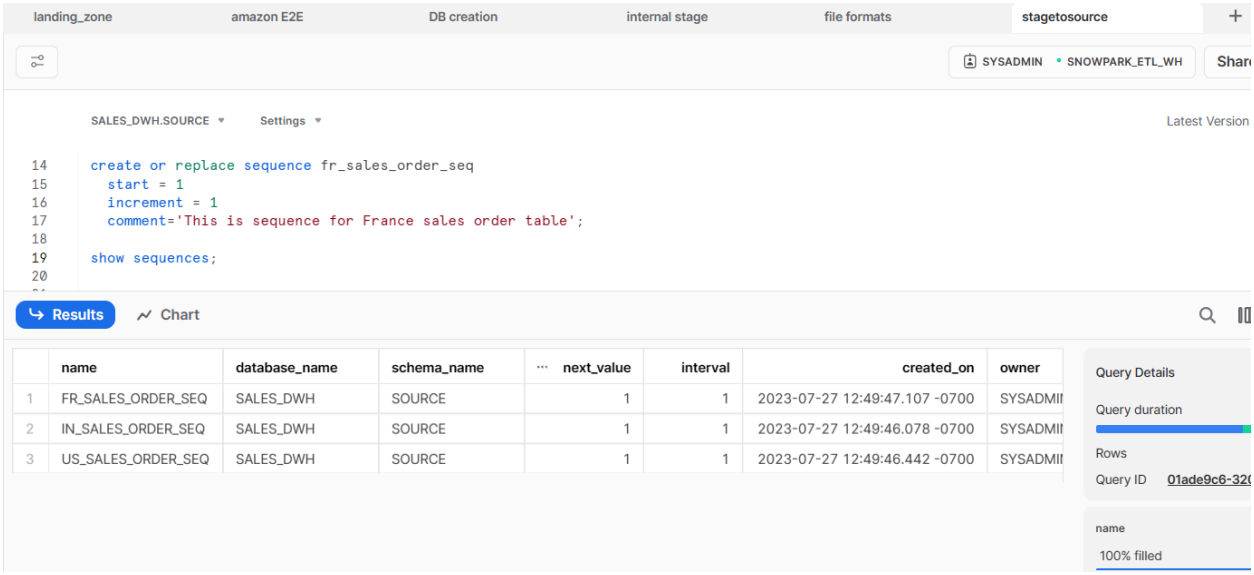
```
snowpark_user#SNOWPARK_ETL_WH@SALES_DWH.SOURCE>list @my_internal_stg;
+------+------+-----+---------------+
| name | size | md5 | last_modified |
|------+------+-----+---------------|
+------+------+-----+---------------+
0 Row(s) produced. Time Elapsed: 0.761s
snowpark_user#SNOWPARK_ETL_WH@SALES_DWH.SOURCE>put file://C:\Users\LENOVO\Downloads\snowparke2e\order-20200101.csv @SAL
                                               ES_DWH.SOURCE.MY_INTERNAL_STG;
+--------------------+-----------------------+-------------+-------------+--------------------+--------------------+----------+---------+
| source             | target                | source_size | target_size | source_compression | target_compression | status   | message |
|--------------------+-----------------------+-------------+-------------+--------------------+--------------------+----------+---------|
| order-20200101.csv | order-20200101.csv.gz |      172648 |       58880 | NONE               | GZIP               | UPLOADED |         |
+--------------------+-----------------------+-------------+-------------+--------------------+--------------------+----------+---------+
1 Row(s) produced. Time Elapsed: 3.234s
```

After onloading our data we shall take a preview

Example from one of the data from india



Now this is how we load data from internal stage to source tables



Finally data has been copied into the tables (please refer to python file provided)

Code written on my internal stage [after onloading dataset and creating suitable tables]

Creating internal stage that will host all the data setup available in our local machine.

SALES_DWH.SOURCE ▾ Settings ▾

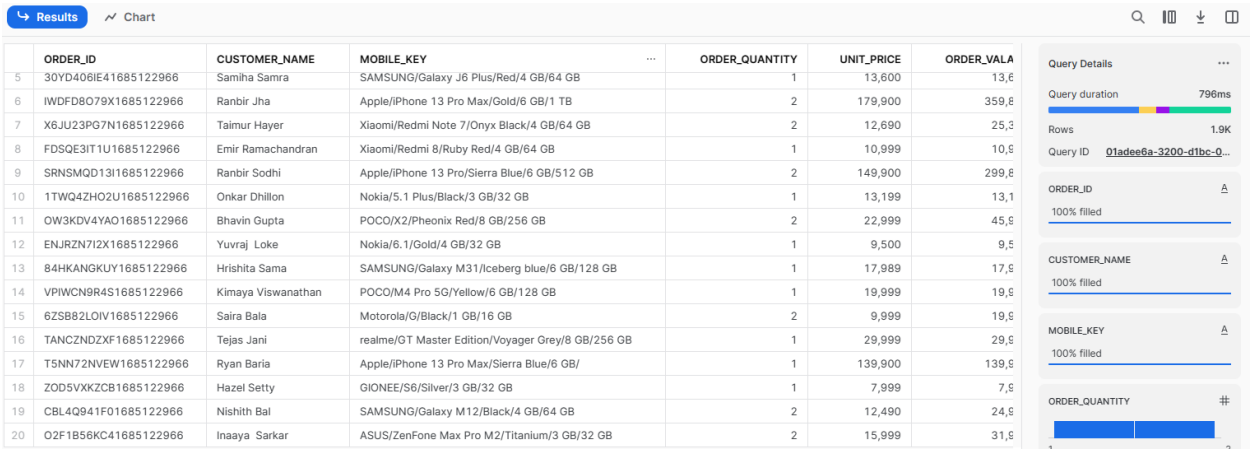
```
use schema source ;
create or replace stage my_internal_stg;

desc stage my_internal_stg;

list @my_internal_stg;

list @my_internal_stg/exchange/;
use schema common;
create or replace transient table exchange_rate(
    date date,
    usd2usd decimal(10,7),
    usd2eu decimal(10,7),
    usd2can decimal(10,7),
    usd2uk decimal(10,7),
    usd2inr decimal(10,7),
    usd2jp decimal(10,7)
);

copy into sales_dwh.common.exchange_rate
from
(
select
    t.$1::date as exchange_dt,
    to_decimal(t.$2) as usd2usd,
    to_decimal(t.$3,12,10) as usd2eu,
    to_decimal(t.$4,12,10) as usd2can,
    to_decimal(t.$4,12,10) as usd2uk,
    to_decimal(t.$4,12,10) as usd2inr,
    to_decimal(t.$4,12,10) as usd2jp
from
     @sales_dwh.source.my_internal_stg/exchange/exchange-rate.csv
     (file_format => 'sales_dwh.common.my_csv_format') t
);

select * from sales_dwh.source.IN_SALES_ORDER;


```

Below is the code for file formatting and other table creation

SALES_DWH.SOURCE ▾ Settings ▾

```
use schema common;
-- create file formats csv (India), json (France), Parquet (USA)
create or replace file format my_csv_format
  type = csv
  field_delimiter = ','
  skip_header = 1
  null_if = ('null', 'null')
  empty_field_as_null = true
  field_optionally_enclosed_by = '\042'
  compression = auto;

-- json file format with strip outer array true
create or replace file format my_json_format
  type = json
  strip_outer_array = true
  compression = auto;

-- parquet file format
create or replace file format my_parquet_format
  type = parquet
  compression = snappy;

use schema source;

select
    t.$1::text as order_id,
    t.$2::text as customer_name,
    t.$3::text as mobile_key,
    t.$4::number as order_quantity,
    t.$5::number as unit_price,
    t.$6::number as order_valaue,
    t.$7::text as promotion_code ,
    t.$8::number(10,2)  as final_order_amount,
    t.$9::number(10,2) as tax_amount,
    t.$10::date as order_dt,
    t.$11::text as payment_status,
    t.$12::text as shipping_status,
    t.$13::text as payment_method,
    t.$14::text as payment_provider,
    t.$15::text as mobile,
    t.$16::text as shipping_address
```

File format stage results



Results | Chart

	ORDER_ID	CUSTOMER_NAME	MOBILE_KEY	ORDER_QUANTITY	UNIT_PRICE	ORDER_VALA
5	30YD406IE41685122966	Samiha Samra	SAMSUNG/Galaxy J6 Plus/Red/4 GB/64 GB	1	13,600	13,6
6	IWDFD8O79X1685122966	Ranbir Jha	Apple/iPhone 13 Pro Max/Gold/6 GB/1 TB	2	179,900	359,8
7	X6JU23PG7N1685122966	Taimur Hayer	Xiaomi/Redmi Note 7/Onyx Black/4 GB/64 GB	2	12,690	25,3
8	FDSQE3IT1U1685122966	Emir Ramachandran	Xiaomi/Redmi 8/Ruby Red/4 GB/64 GB	1	10,999	10,9
9	SRNSMQD13I1685122966	Ranbir Sodhi	Apple/iPhone 13 Pro/Sierra Blue/6 GB/512 GB	2	149,900	299,8
10	1TWQ4ZHO2U1685122966	Onkar Dhillon	Nokia/5.1 Plus/Black/3 GB/32 GB	1	13,199	13,1
11	OW3KDV4YAO1685122966	Bhavin Gupta	POCO/X2/Pheonix Red/8 GB/256 GB	2	22,999	45,9
12	ENJRZN7I2X1685122966	Yuvraj Loke	Nokia/6.1/Gold/4 GB/32 GB	1	9,500	9,5
13	84HKANGKUY1685122966	Hrishita Sama	SAMSUNG/Galaxy M31/Iceberg blue/6 GB/128 GB	1	17,989	17,9
14	VPIWCN9R4S1685122966	Kimaya Viswanathan	POCO/M4 Pro 5G/Yellow/6 GB/128 GB	1	19,999	19,9
15	6ZSB82LOIV1685122966	Saira Bala	Motorola/G/Black/1 GB/16 GB	2	9,999	19,9
16	TANCZNDZXF1685122966	Tejas Jani	realme/GT Master Edition/Voyager Grey/8 GB/256 GB	1	29,999	29,9
17	T5NN72NVEW1685122966	Ryan Baria	Apple/iPhone 13 Pro Max/Sierra Blue/6 GB/	1	139,900	139,9
18	ZOD5VXKZCB1685122966	Hazel Setty	GIONEE/S6/Silver/3 GB/32 GB	1	7,999	7,9
19	CBL4Q941F01685122966	Nishith Bal	SAMSUNG/Galaxy M12/Black/4 GB/64 GB	2	12,490	24,9
20	O2F1B56KC41685122966	Inaaya Sarkar	ASUS/ZenFone Max Pro M2/Titanium/3 GB/32 GB	2	15,999	31,9

Query Details

Query duration 796ms

Rows 1.9K

Query ID 01adee6a-3200-d1bc-0...

ORDER_ID — 100% filled

CUSTOMER_NAME — 100% filled

MOBILE_KEY — 100% filled

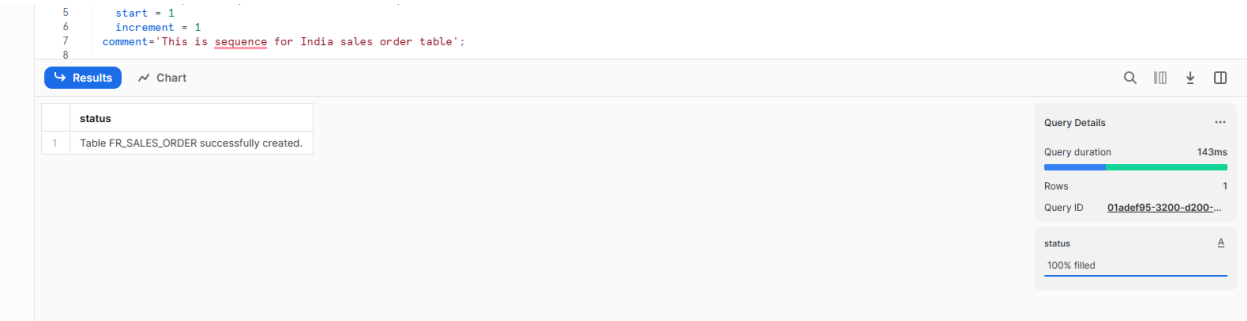
ORDER_QUANTITY

Source stage code

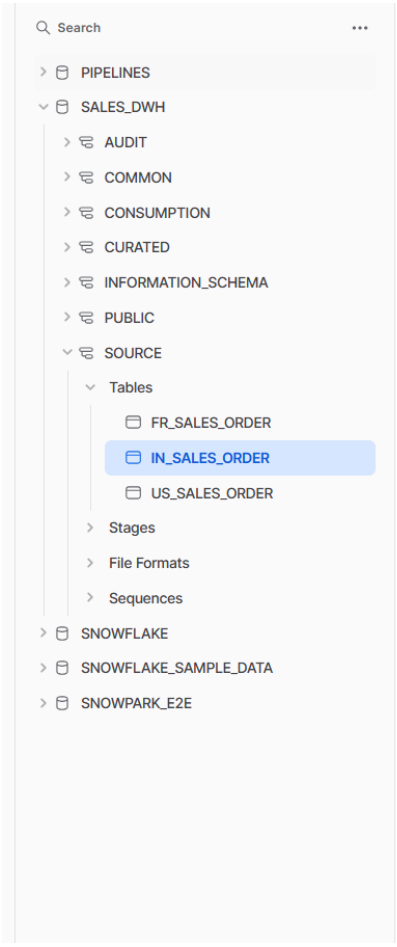
SALES_DWH.SOURCE | Settings

```
 customer_name varchar(),
 mobile_key varchar(),
 order_quantity number(38,0),
 unit_price number(38,0),
 order_valaue number(38,0),
 promotion_code varchar(),
 final_order_amount number(10,2),
 tax_amount number(10,2),
 order_dt date,
 payment_status varchar(),
 shipping_status varchar(),
 payment_method varchar(),
 payment_provider varchar(),
 mobile varchar(),
 shipping_address varchar(),
 _metadata_file_name varchar(),
 _metadata_row_numer number(38,0),
 _metadata_last_modified timestamp_ntz(9)
);

-- US Sales Table in Source Schema (Parquet File)
create or replace transient table us_sales_order (
 sales_order_key number(38,0),
 order_id varchar(),
 customer_name varchar(),
 mobile_key varchar(),
 order_quantity number(38,0),
 unit_price number(38,0),
 order_valaue number(38,0),
 promotion_code varchar(),
 final_order_amount number(10,2),
 tax_amount number(10,2),
 order_dt date,
 payment_status varchar(),
 shipping_status varchar(),
 payment_method varchar(),
 payment_provider varchar(),
 phone varchar(),
 shipping_address varchar(),
 _metadata_file_name varchar(),
 _metadata_row_numer number(38,0),
 _metadata_last_modified timestamp_ntz(9)
);

-- France Sales Table in Source Schema (JSON File)
create or replace transient table fr_sales_order (
 sales_order_key number(38,0),
 order_id varchar(),
 customer_name varchar(),
 mobile_key varchar(),
 order_quantity number(38,0),
 unit_price number(38,0),
 order_valaue number(38,0),
 promotion_code varchar(),
 final_order_amount number(10,2),
 tax_amount number(10,2),
 order_dt date,
 payment_status varchar(),
 shipping_status varchar(),
 payment_method varchar(),
 payment_provider varchar(),
 phone varchar(),
 shipping_address varchar(),
 _metadata_file_name varchar(),
 _metadata_row_numer number(38,0),
 _metadata_last_modified timestamp_ntz(9)
);

show
tables;
```

Source stage results



FINAL REPRESENTATION OF AMAZON INDIA MOBILE ORDER DATA



SALES_DWH / SOURCE / IN_SALES_ORDER

Load Data

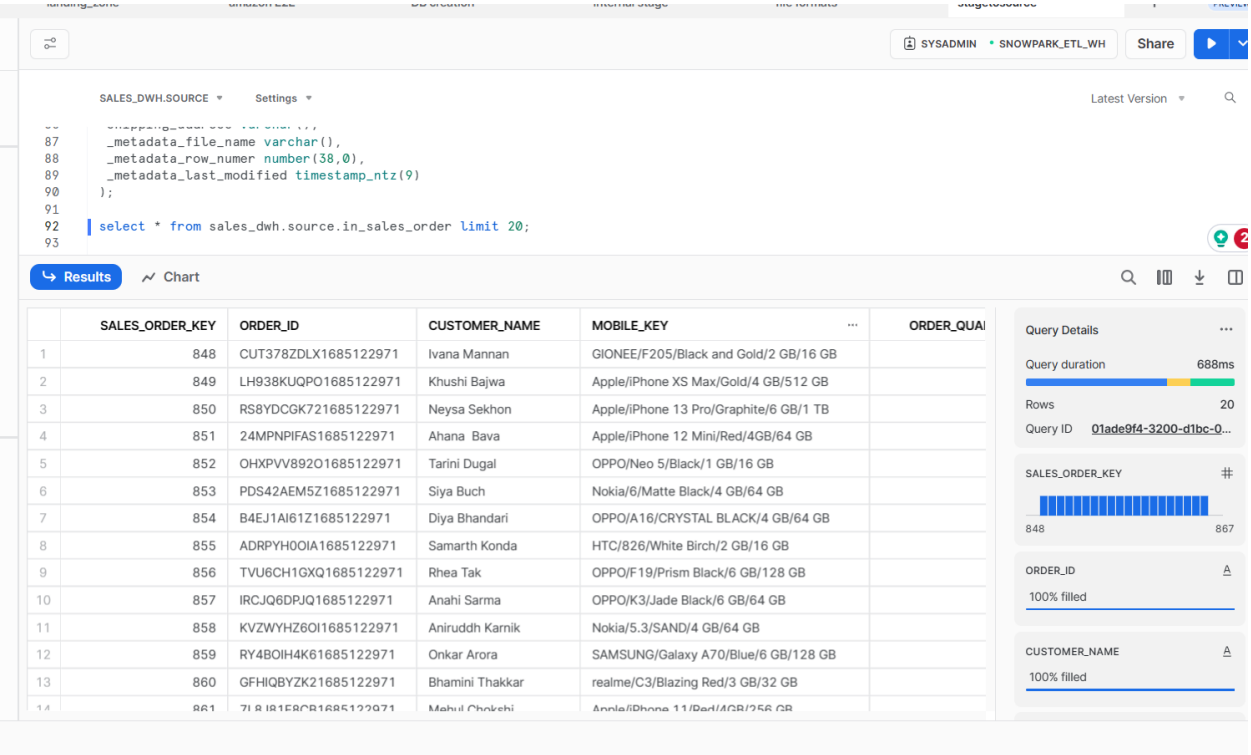
Table | SYSADMIN | 35 minutes ago | 1.9K | 170.5KB

Table Details | Columns | Data Preview | Copy History

COMPUTE_WH 100 of 1.9K Rows • Updated just now

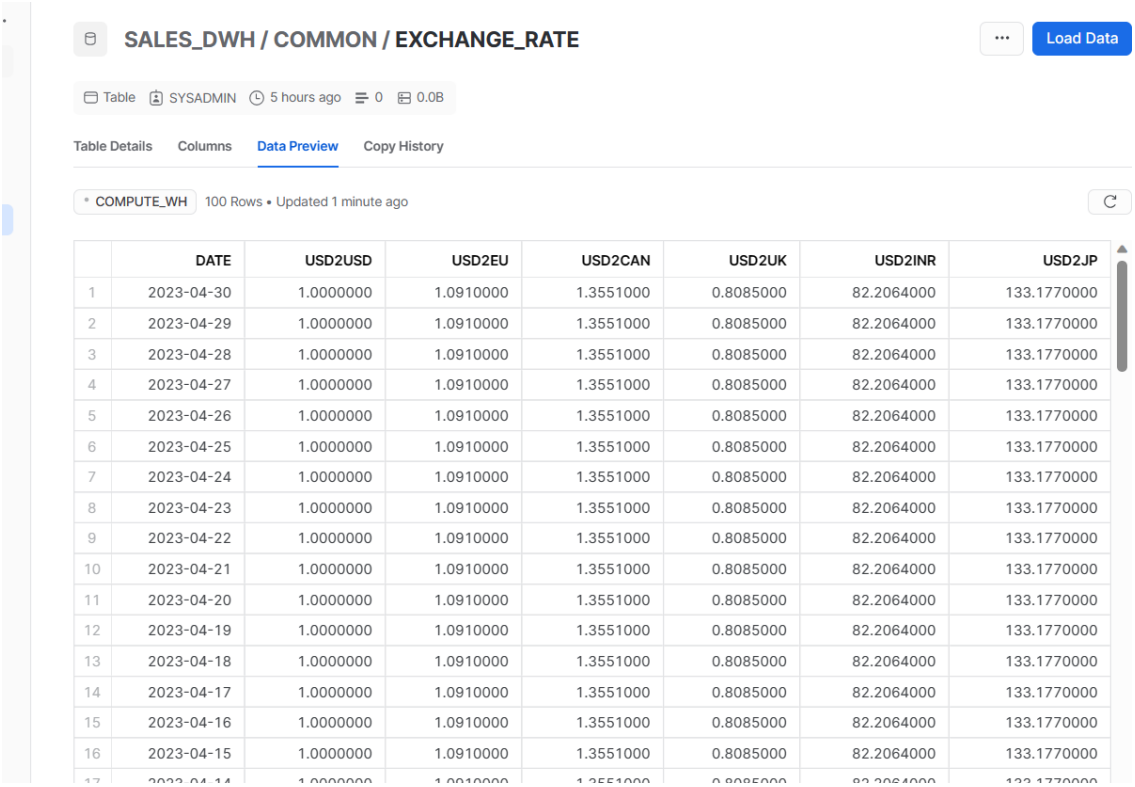
	SALES_ORDER_KEY	ORDER_ID	CUSTOMER_NAME	MOBILE_KEY
1	848	CUT378ZDLX1685122971	Ivana Mannan	GIONEE/F205/Black and Gold/2 GB/16 GB
2	849	LH938KUQPO1685122971	Khushi Bajwa	Apple/iPhone XS Max/Gold/4 GB/512 GB
3	850	RS8YDCGK721685122971	Neysa Sekhon	Apple/iPhone 13 Pro/Graphite/6 GB/1 TB
4	851	24MPNPIFAS1685122971	Ahana Bava	Apple/iPhone 12 Mini/Red/4GB/64 GB
5	852	OHXPVV892O1685122971	Tarini Dugal	OPPO/Neo 5/Black/1 GB/16 GB
6	853	PDS42AEM5Z1685122971	Siya Buch	Nokia/6/Matte Black/4 GB/64 GB
7	854	B4EJ1AI61Z1685122971	Diya Bhandari	OPPO/A16/CRYSTAL BLACK/4 GB/64 GB
8	855	ADRPYH0OIA1685122971	Samarth Konda	HTC/826/White Birch/2 GB/16 GB
9	856	TVU6CH1GXQ1685122971	Rhea Tak	OPPO/F19/Prism Black/6 GB/128 GB
10	857	IRCJQ6DPJQ1685122971	Anahi Sarma	OPPO/K3/Jade Black/6 GB/64 GB
11	858	KVZWYHZ6OI1685122971	Aniruddh Karnik	Nokia/5.3/SAND/4 GB/64 GB
12	859	RY4BOIH4K61685122971	Onkar Arora	SAMSUNG/Galaxy A70/Blue/6 GB/128 GB
13	860	GFHIQBYZK21685122971	Bhamini Thakkar	realme/C3/Blazing Red/3 GB/32 GB
14	861	7L8J81E8CB1685122971	Mehul Chokshi	Apple/iPhone 11/Red/4GB/256 GB
15	862	UA71X5EXBO1685122971	Lakshay Guha	realme/X2/Pearl White/8 GB/128 GB
16	863	2EWMS78FSJ1685122971	Kiaan Sanghvi	SAMSUNG/Galaxy J2 Ace/Gold/1.5 GB/8 GB
17	864	PK2295EOK11685122971	Nitya Brar	OPPO/A3s/Red/4 GB/64 GB
18	865	GD8PU4RPDL1685122971	Umang Bhardwaj	SAMSUNG/Galaxy A20s/Blue/4 GB/64 GB

We are also successfully able to select * from databse and call the table with swift results



Loaded data into forex table

In order to construct PKI in a single currency and compare performance, produce foreign exchange rate data to convert local currency data (such as INR or Euro) to US Dollar. This will allow us to create total sales at the global level.



Now after creating sales_in table under curated schema , from the original 1.9k rows It reduced to 300 as it only contains paid and delivered order with no null values .

FINALLY WE COME TO THE CURATED AND CONSUMPTION STAGE

BELOW IS THE CODE FOR THE CURATED STAGE WITH SCHEMAS CREATED FOR THE CURATED STAGE

landing_zone amazon E2E DB creation

SALES_DWH.CURATED Settings

```
use schema curated;
-- curated India sales order table
create or replace table in_sales_order (
 sales_order_key number(38,0),
 order_id varchar(),
 order_dt date,
 customer_name varchar(),
 mobile_key varchar(),
 country varchar(),
 region varchar(),
 order_quantity number(38,0),
 local_currency varchar(),
 local_unit_price number(38,0),
 promotion_code varchar(),
 local_total_order_amt number(10,2),
 local_tax_amt number(10,2),
 exhchange_rate number(15,7),
 us_total_order_amt number(23,8),
 usd_tax_amt number(23,8),
 payment_status varchar(),
 shipping_status varchar(),
 payment_method varchar(),
 payment_provider varchar(),
 conctact_no varchar(),
 shipping_address varchar()
);

select * from sales_dwh.curated.in_sales_order;
```

RESULTS

Results Chart

	SALES_ORDER_KEY	ORDER_ID	ORDER_DT	CUSTOMER_NAME	MOBILE_KEY	COUNTRY	REGION
1	1,932	CUT378ZDLX1685122971	2020-01-02	Ivana Mannan	GIONEE/F205/Black and Gold/2 GB/16 GB	IN	APAC
2	1,937	PDS42AEM5Z1685122971	2020-01-02	Siya Buch	Nokia/6/Matte Black/4 GB/64 GB	IN	APAC
3	1,940	TVU6CH1GXQ1685122971	2020-01-02	Rhea Tak	OPPO/F19/Prism Black/6 GB/128 GB	IN	APAC
4	1,942	KVZWYHZ6OI1685122971	2020-01-02	Aniruddh Karnik	Nokia/5.3/SAND/4 GB/64 GB	IN	APAC
5	1,951	X6MYAAZTIK1685122971	2020-01-02	Aarav Chawla	vivo/U12/Electric Blue/4 GB/64 GB	IN	APAC
6	1,953	9VPGGVFG4N1685122971	2020-01-02	Biju Barman	Infinix/Zero 8i/Silver Diamond/8 GB/128 GB	IN	APAC
7	1,960	FBFQW7I1EV1685122971	2020-01-02	Piya Yadav	SAMSUNG/Galaxy A8/Gold/2 GB/32 GB	IN	APAC
8	1,963	0MEHQN34KD1685122971	2020-01-02	Shaan Sabharwal	SAMSUNG/Galaxy A12/Black/6 GB/128 GB	IN	APAC
9	1,966	0QP3XTJP5K1685122971	2020-01-02	Ritvik Dua	OPPO/F9 Pro/Starry Purple/6 GB/64 GB	IN	APAC

Query Details

Query duration 765ms

Rows 320

Query ID 01adf1f1-3200-d237-0...

SALES_ORDER_KEY

1,932 3,861

ORDER_ID

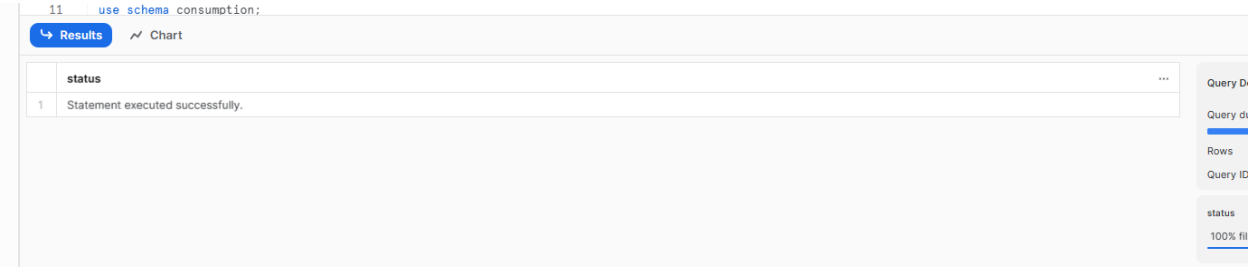
100% filled

Consumption stage code

```
    Region text,
    isActive text(1)
);

-- product dimension
use schema consumption;
create or replace sequence product_dim_seq start = 1 increment = 1;
create or replace transient table product_dim(
    product_id_pk number primary key,
    Mobile_key text,
    Brand text,
    Model text,
    Color text,
    Memory text,
    isActive text(1)
);

-- promo_code dimension
use schema consumption;
create or replace sequence promo_code_dim_seq start = 1 increment = 1;
create or replace transient table promo_code_dim(
    promo_code_id_pk number primary key,
    promo_code text,
    isActive text(1)
);

-- customer dimension
use schema consumption;
create or replace sequence customer_dim_seq start = 1 increment = 1;
create or replace transient table customer_dim(
    customer_id_pk number primary key,
    customer_name text,
    CONCTACT_NO text,
    SHIPPING_ADDRESS text,
    country text,
    region text,
    isActive text(1)


-- payment dimension
use schema consumption;
create or replace sequence payment_dim_seq start = 1 increment = 1;
create or replace transient table payment_dim(
    payment_id_pk number primary key,
    PAYMENT_METHOD text,
    PAYMENT_PROVIDER text,
    country text,
    region text,
    isActive text(1)
);
-- payment dimension
use schema consumption;
create or replace sequence payment_dim_seq start = 1 increment = 1;
create or replace transient table payment_dim(
    payment_id_pk number primary key,
    PAYMENT_METHOD text,
    PAYMENT_PROVIDER text,
    country text,
    region text,
    isActive text(1)
);

use schema consumption;
create or replace sequence date_dim_seq start = 1 increment = 1;
create or replace transient table date_dim(
    date_id_pk int primary key,
    order_dt date,
    order_year int,
    oder_month int,
    order_quater int,
    order_day int,
    order_dayofweek int,
    order_dayname text,
    order_dayofmonth int,
    order_weekday text
);

-- fact tables
create or replace table sales_fact (
 order_id_pk number(38,0),
 order_code varchar(),
 date_id_fk number(38,0),
 region_id_fk number(38,0),
 customer_id_fk number(38,0),
 payment_id_fk number(38,0),
 product_id_fk number(38,0),
 promo_code_id_fk number(38,0),
 order_quantity number(38,0),
 local_total_order_amt number(10,2),
 local_tax_amt number(10,2),
 exhchange_rate number(15,7),
 us_total_order_amt number(23,8),
 usd_tax_amt number(23,8)
);

-- Table Containts
alter table sales_fact add
    constraint fk_sales_region FOREIGN KEY (REGION_ID_FK) REFERENCES region_dim (REGION_ID_PK) NOT ENFORCED;

alter table sales_fact add
    constraint fk_sales_date FOREIGN KEY (DATE_ID_FK) REFERENCES date_dim (DATE_ID_PK) NOT ENFORCED;

alter table sales_fact add
    constraint fk_sales_customer FOREIGN KEY (CUSTOMER_ID_FK) REFERENCES customer_dim (CUSTOMER_ID_PK) NOT ENFORCED;
--
alter table sales_fact add
    constraint fk_sales_payment FOREIGN KEY (PAYMENT_ID_FK) REFERENCES payment_dim (PAYMENT_ID_PK) NOT ENFORCED;

alter table sales_fact add
    constraint fk_sales_product FOREIGN KEY (PRODUCT_ID_FK) REFERENCES product_dim (PRODUCT_ID_PK) NOT ENFORCED;

alter table sales_fact add
    constraint fk_sales_promot FOREIGN KEY (PROMO_CODE_ID_FK) REFERENCES promo_code_dim (PROMO_CODE_ID_PK) NOT ENFORCED;
```

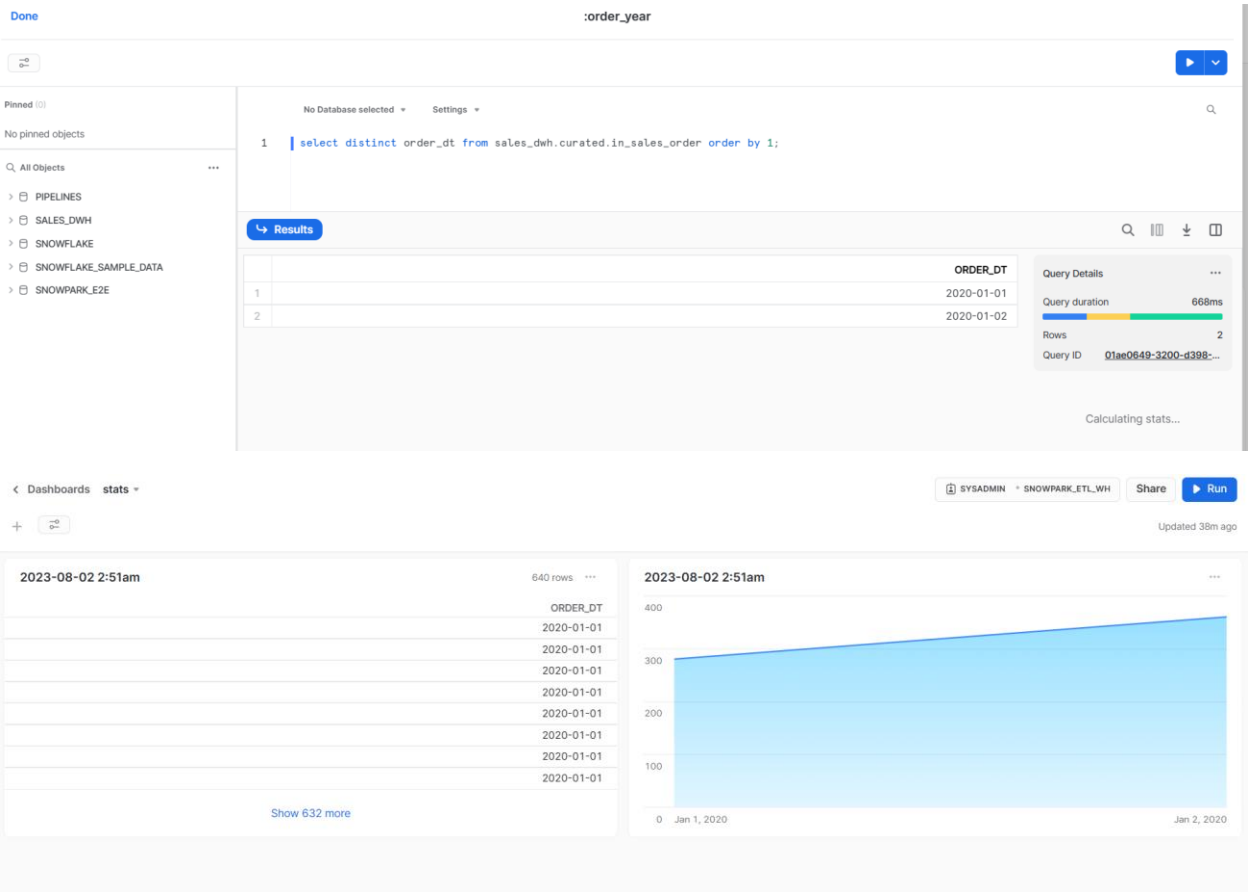
CONSUMPTION STAGE RESULTS

[IN THIS WE HAVE CREATED REGION , PROMO CODE , PRODUCT , CUSTOMER , PAYMENT AND DATE SEQUENCE OBJECTS]



With this we have finished with the creation of the ETL insights analytics pipeline and now we can easily take insights further on with the help of python code to create specialized joined tables.

Created filter order_year for defining the snowpark insight and many more to come as I am expanding my knowledge.



Please do find the python file provided to understand the background process other than the sql worksheets provided above.