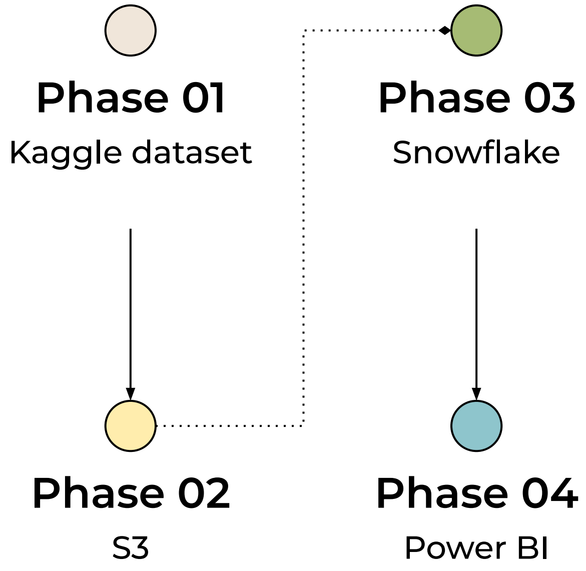
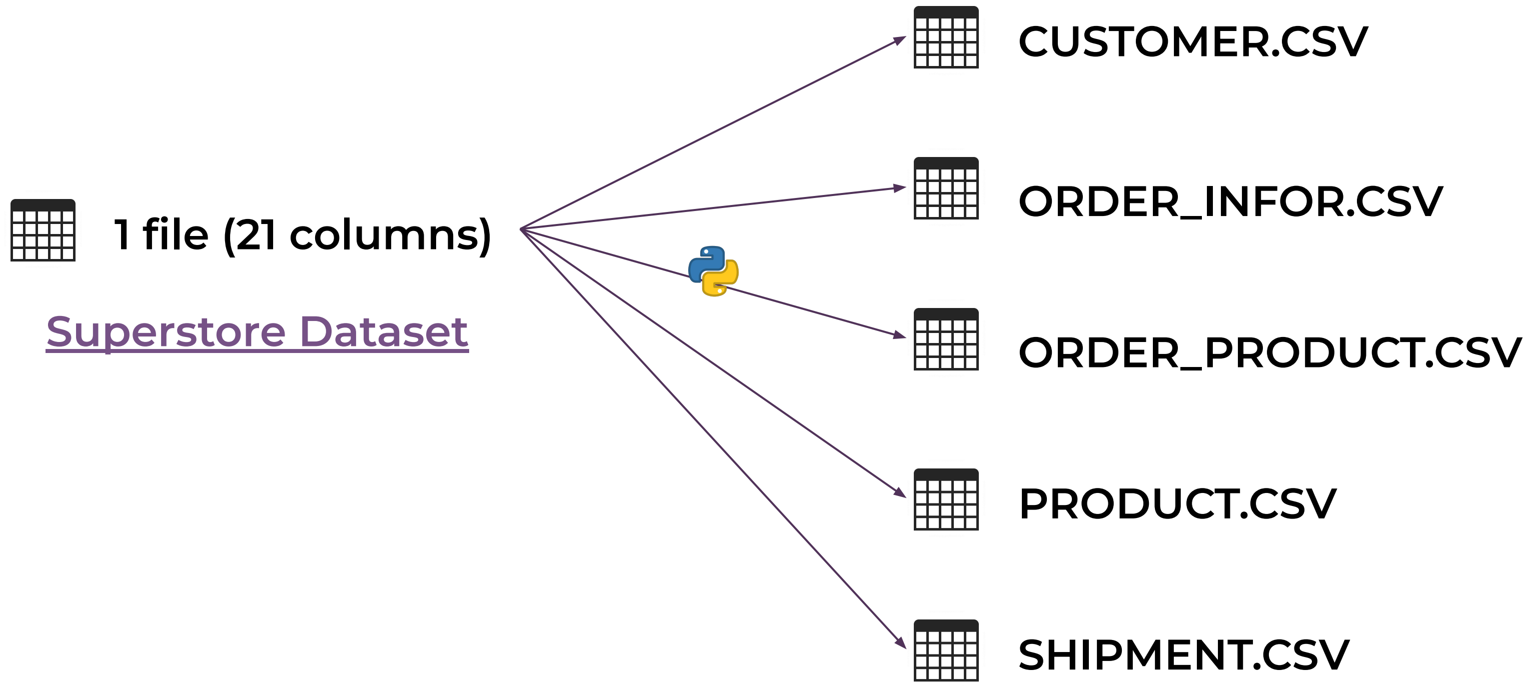
Small Project

Using Snowflake follow team’s ảchitecture

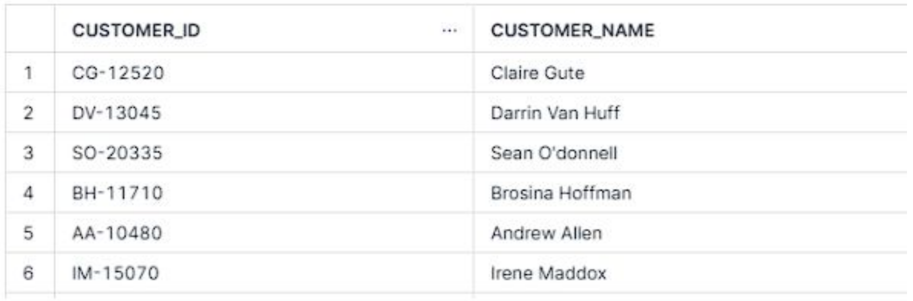


Datasets


1 file (21 columns)
Superstore Dataset
CUSTOMER.CSV
ORDER_INFOR.CSV
ORDER_PRODUCT.CSV
PRODUCT.CSV
SHIPMENT.CSV

Datasets

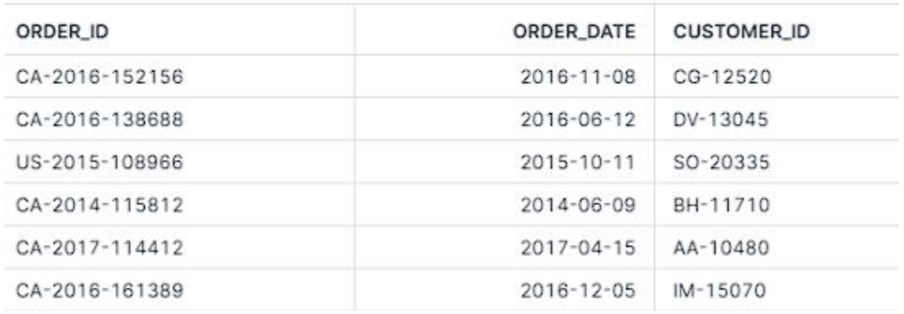
CUSTOMER.CSV (793 rows)



	CUSTOMER_ID	CUSTOMER_NAME
1	CG-12520	Claire Gute
2	DV-13045	Darrin Van Huff
3	SO-20335	Sean O'donnell
4	BH-11710	Brosina Hoffman
5	AA-10480	Andrew Allen
6	IM-15070	Irene Maddox

Datasets

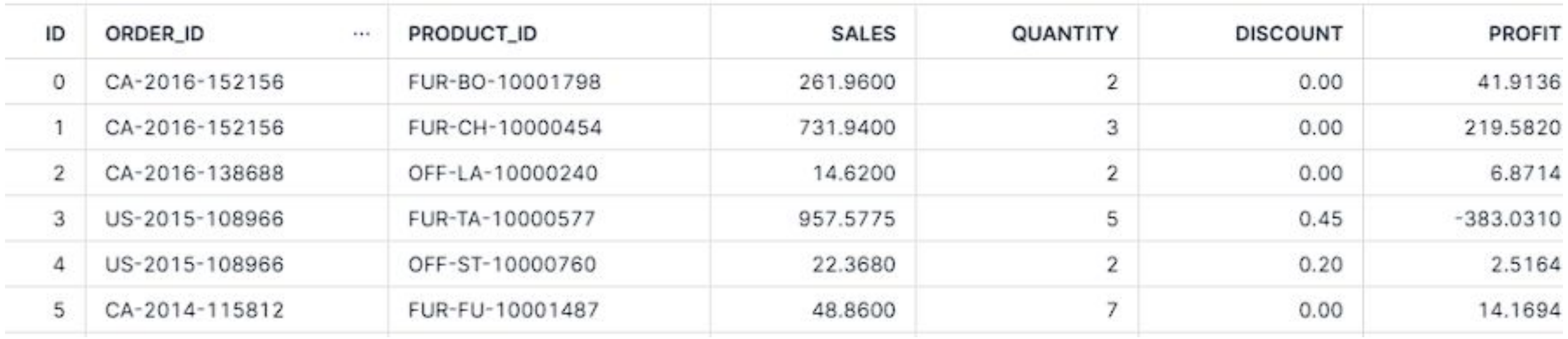
ORDER_INFOR.CSV (5009 rows)



ORDER_ID	ORDER_DATE	CUSTOMER_ID
CA-2016-152156	2016-11-08	CG-12520
CA-2016-138688	2016-06-12	DV-13045
US-2015-108966	2015-10-11	SO-20335
CA-2014-115812	2014-06-09	BH-11710
CA-2017-114412	2017-04-15	AA-10480
CA-2016-161389	2016-12-05	IM-15070

Datasets

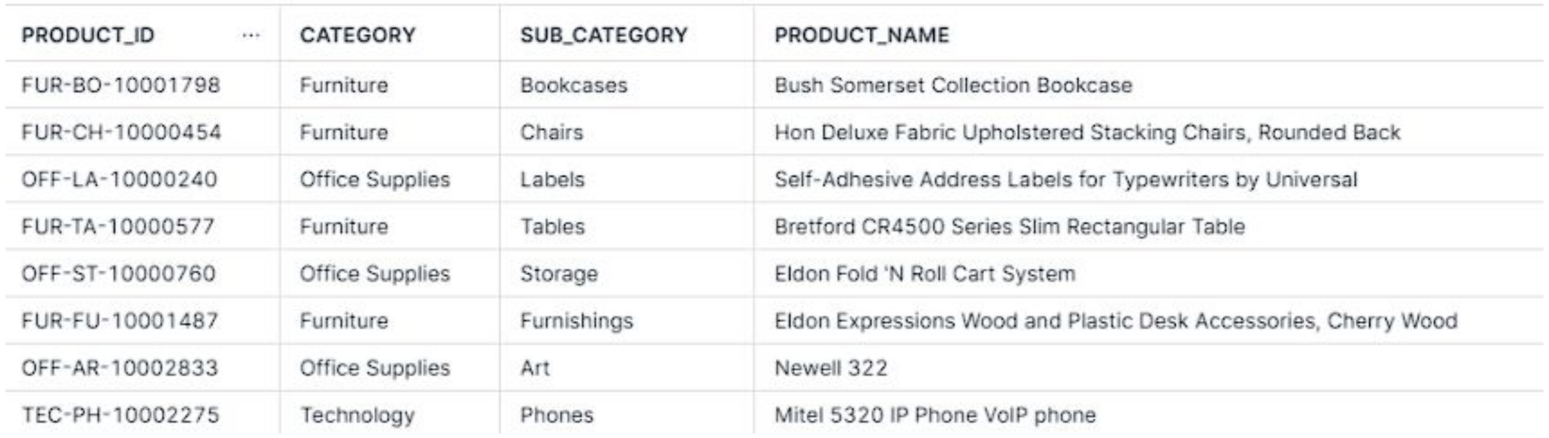
ORDER_PRODUCT.CSV (9993 rows)



ID	ORDER_ID	PRODUCT_ID	SALES	QUANTITY	DISCOUNT	PROFIT
0	CA-2016-152156	FUR-BO-10001798	261.9600	2	0.00	41.9136
1	CA-2016-152156	FUR-CH-10000454	731.9400	3	0.00	219.5820
2	CA-2016-138688	OFF-LA-10000240	14.6200	2	0.00	6.8714
3	US-2015-108966	FUR-TA-10000577	957.5775	5	0.45	-383.0310
4	US-2015-108966	OFF-ST-10000760	22.3680	2	0.20	2.5164
5	CA-2014-115812	FUR-FU-10001487	48.8600	7	0.00	14.1694

Datasets

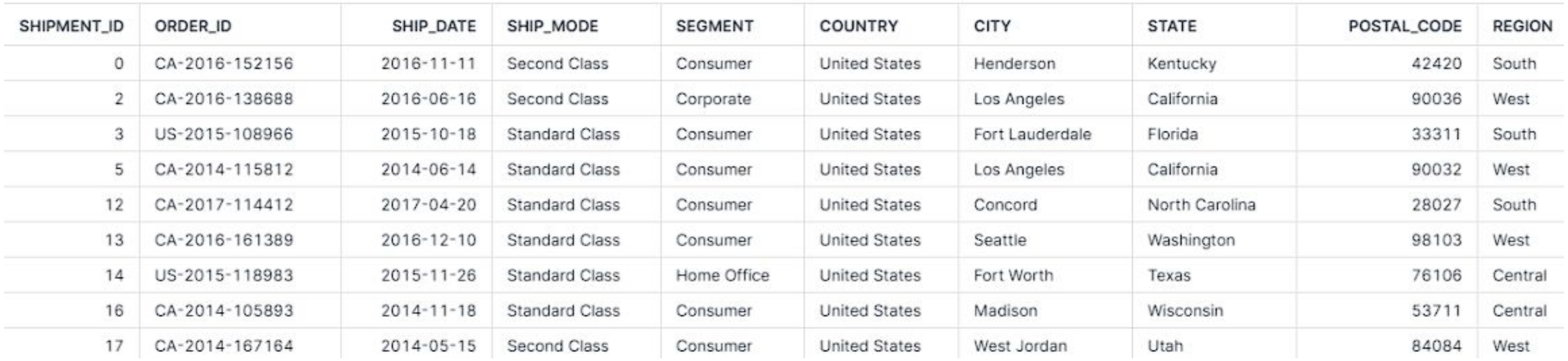
PRODUCT.CSV (1894 rows)



PRODUCT_ID	CATEGORY	SUB_CATEGORY	PRODUCT_NAME
FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase
FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back
OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters by Universal
FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table
OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System
FUR-FU-10001487	Furniture	Furnishings	Eldon Expressions Wood and Plastic Desk Accessories, Cherry Wood
OFF-AR-10002833	Office Supplies	Art	Newell 322
TEC-PH-10002275	Technology	Phones	Mitel 5320 IP Phone VoIP phone

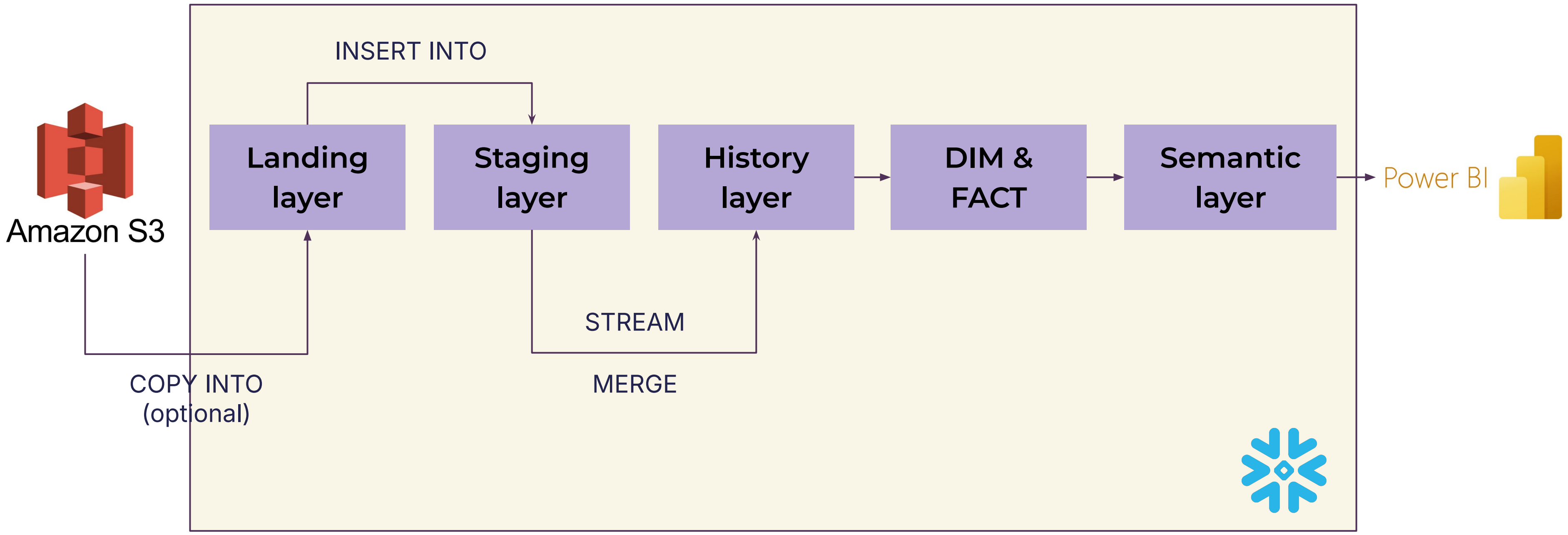
Datasets

SHIPMENT.CSV (5009 rows)

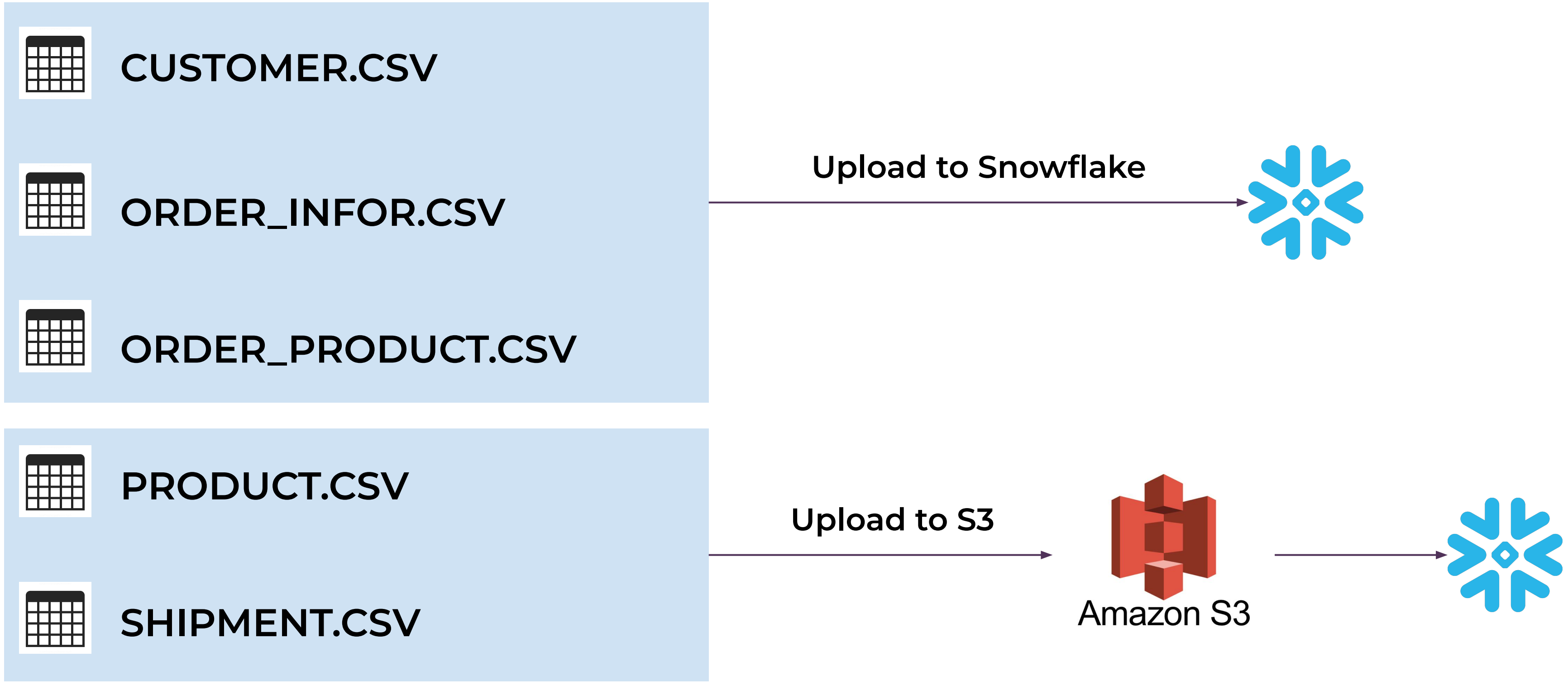


SHIPMENT_ID	ORDER_ID	SHIP_DATE	SHIP_MODE	SEGMENT	COUNTRY	CITY	STATE	POSTAL_CODE	REGION
0	CA-2016-152156	2016-11-11	Second Class	Consumer	United States	Henderson	Kentucky	42420	South
2	CA-2016-138688	2016-06-16	Second Class	Corporate	United States	Los Angeles	California	90036	West
3	US-2015-108966	2015-10-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South
5	CA-2014-115812	2014-06-14	Standard Class	Consumer	United States	Los Angeles	California	90032	West
12	CA-2017-114412	2017-04-20	Standard Class	Consumer	United States	Concord	North Carolina	28027	South
13	CA-2016-161389	2016-12-10	Standard Class	Consumer	United States	Seattle	Washington	98103	West
14	US-2015-118983	2015-11-26	Standard Class	Home Office	United States	Fort Worth	Texas	76106	Central
16	CA-2014-105893	2014-11-18	Standard Class	Consumer	United States	Madison	Wisconsin	53711	Central
17	CA-2014-167164	2014-05-15	Second Class	Consumer	United States	West Jordan	Utah	84084	West

Requirements



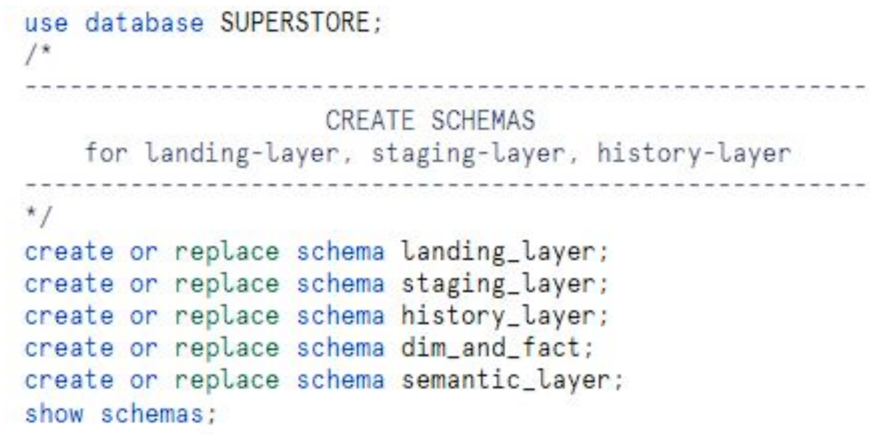
Project


CUSTOMER.CSV
ORDER_INFOR.CSV
ORDER_PRODUCT.CSV
Upload to Snowflake
PRODUCT.CSV
SHIPMENT.CSV
Upload to S3
Amazon S3

Project

Create database (SUPERSTORE)

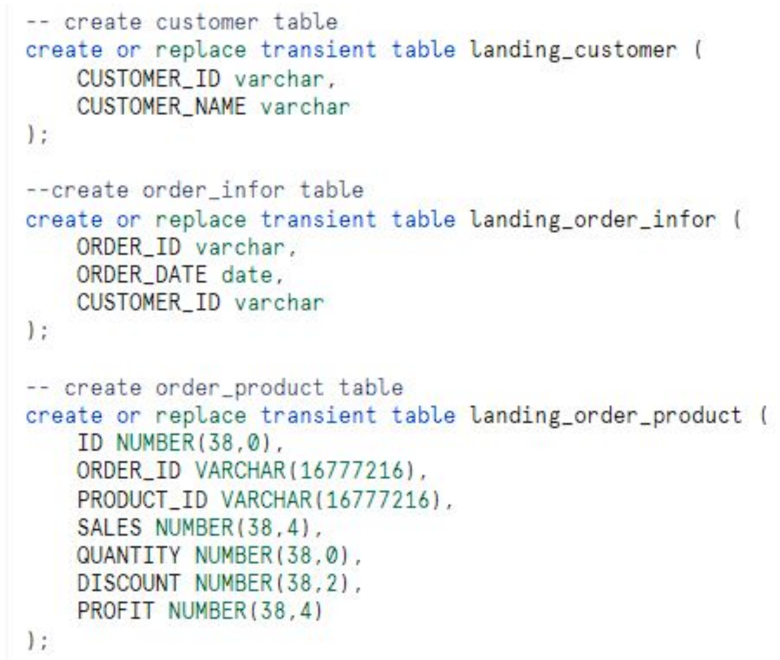
Create 5 schemas



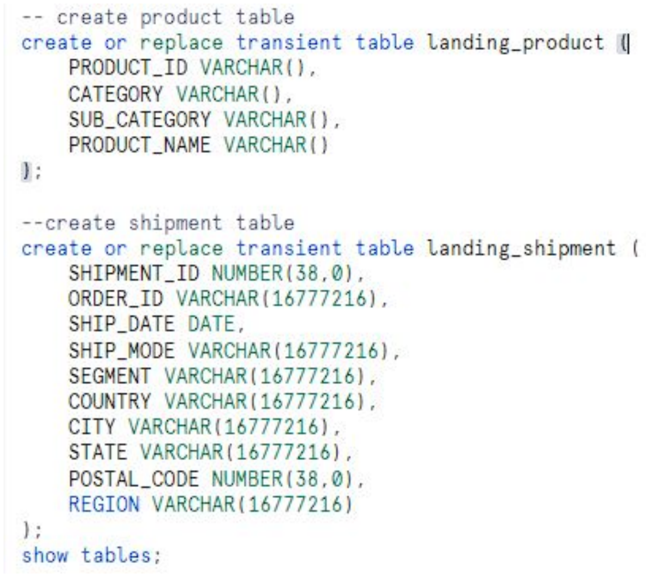
```
use database SUPERSTORE;
/*
--------------------------------------------------------
                    CREATE SCHEMAS
    for landing-layer, staging-layer, history-layer
--------------------------------------------------------
*/
create or replace schema landing_layer;
create or replace schema staging_layer;
create or replace schema history_layer;
create or replace schema dim_and_fact;
create or replace schema semantic_layer;
show schemas;
```

Project - LANDING LAYER

Create 5 transient tables



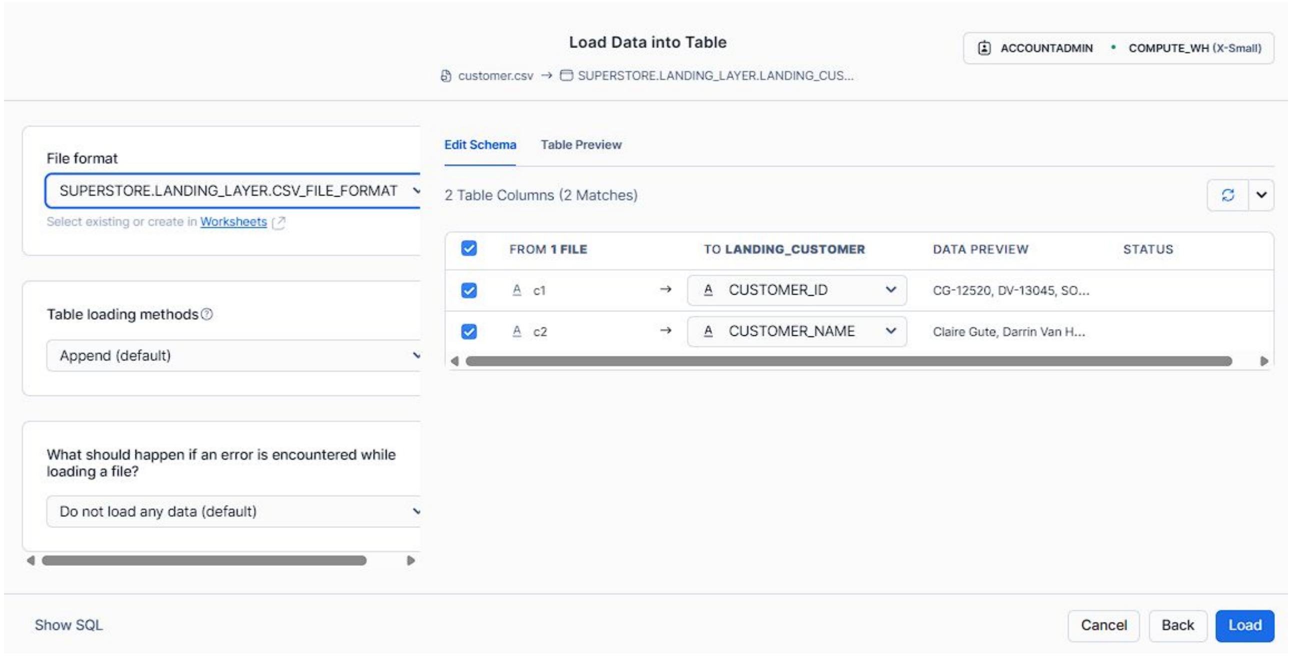
```sql
-- create customer table
create or replace transient table landing_customer (
    CUSTOMER_ID varchar,
    CUSTOMER_NAME varchar
);

--create order_infor table
create or replace transient table landing_order_infor (
    ORDER_ID varchar,
    ORDER_DATE date,
    CUSTOMER_ID varchar
);

-- create order_product table
create or replace transient table landing_order_product (
    ID NUMBER(38,0),
    ORDER_ID VARCHAR(16777216),
    PRODUCT_ID VARCHAR(16777216),
    SALES NUMBER(38,4),
    QUANTITY NUMBER(38,0),
    DISCOUNT NUMBER(38,2),
    PROFIT NUMBER(38,4)
);
```



```sql
-- create product table
create or replace transient table landing_product (
    PRODUCT_ID VARCHAR(),
    CATEGORY VARCHAR(),
    SUB_CATEGORY VARCHAR(),
    PRODUCT_NAME VARCHAR()
);

--create shipment table
create or replace transient table landing_shipment (
    SHIPMENT_ID NUMBER(38,0),
    ORDER_ID VARCHAR(16777216),
    SHIP_DATE DATE,
    SHIP_MODE VARCHAR(16777216),
    SEGMENT VARCHAR(16777216),
    COUNTRY VARCHAR(16777216),
    CITY VARCHAR(16777216),
    STATE VARCHAR(16777216),
    POSTAL_CODE NUMBER(38,0),
    REGION VARCHAR(16777216)
);
show tables;
```

Project - LANDING LAYER

Upload files



Load Data into Table

customer.csv → SUPERSTORE.LANDING_LAYER.LANDING_CUS...

ACCOUNTADMIN • COMPUTE_WH (X-Small)

File format: SUPERSTORE.LANDING_LAYER.CSV_FILE_FORMAT

Select existing or create in Worksheets

Table loading methods: Append (default)

What should happen if an error is encountered while loading a file?: Do not load any data (default)

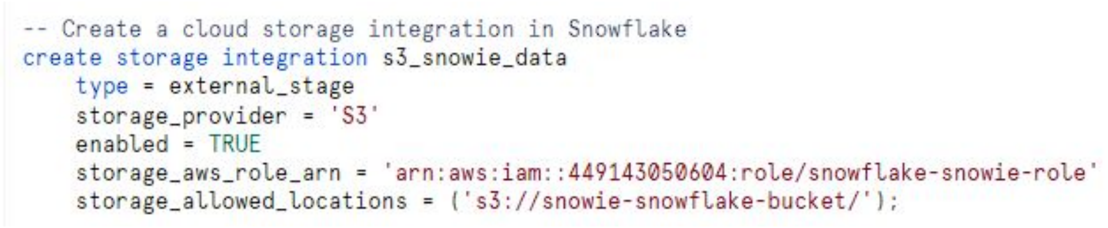
Edit Schema | Table Preview

2 Table Columns (2 Matches)

✓	FROM 1 FILE		TO LANDING_CUSTOMER	DATA PREVIEW	STATUS
✓	c1	→	CUSTOMER_ID	CG-12520, DV-13045, SO...	
✓	c2	→	CUSTOMER_NAME	Claire Gute, Darrin Van H...	

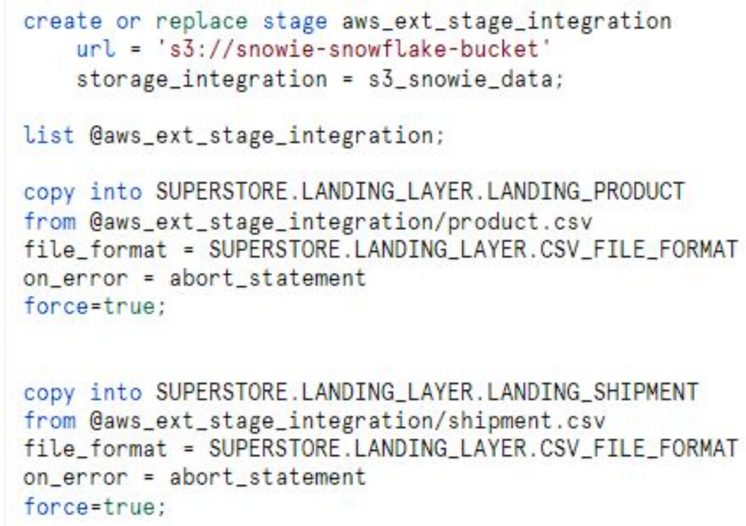
Project - LANDING LAYER

Create a storage integration



```
-- Create a cloud storage integration in Snowflake
create storage integration s3_snowie_data
    type = external_stage
    storage_provider = 'S3'
    enabled = TRUE
    storage_aws_role_arn = 'arn:aws:iam::449143050604:role/snowflake-snowie-role'
    storage_allowed_locations = ('s3://snowie-snowflake-bucket/');
```

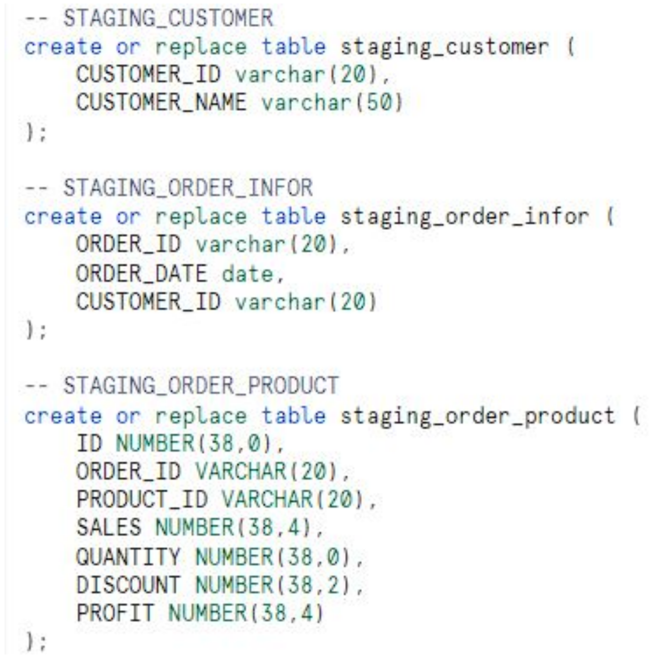
Create a stage + COPY INTO



```
create or replace stage aws_ext_stage_integration
    url = 's3://snowie-snowflake-bucket'
    storage_integration = s3_snowie_data;

list @aws_ext_stage_integration;

copy into SUPERSTORE.LANDING_LAYER.LANDING_PRODUCT
from @aws_ext_stage_integration/product.csv
file_format = SUPERSTORE.LANDING_LAYER.CSV_FILE_FORMAT
on_error = abort_statement
force=true;


copy into SUPERSTORE.LANDING_LAYER.LANDING_SHIPMENT
from @aws_ext_stage_integration/shipment.csv
file_format = SUPERSTORE.LANDING_LAYER.CSV_FILE_FORMAT
on_error = abort_statement
force=true;
```

Project - STAGING LAYER

Create 5 permanent tables

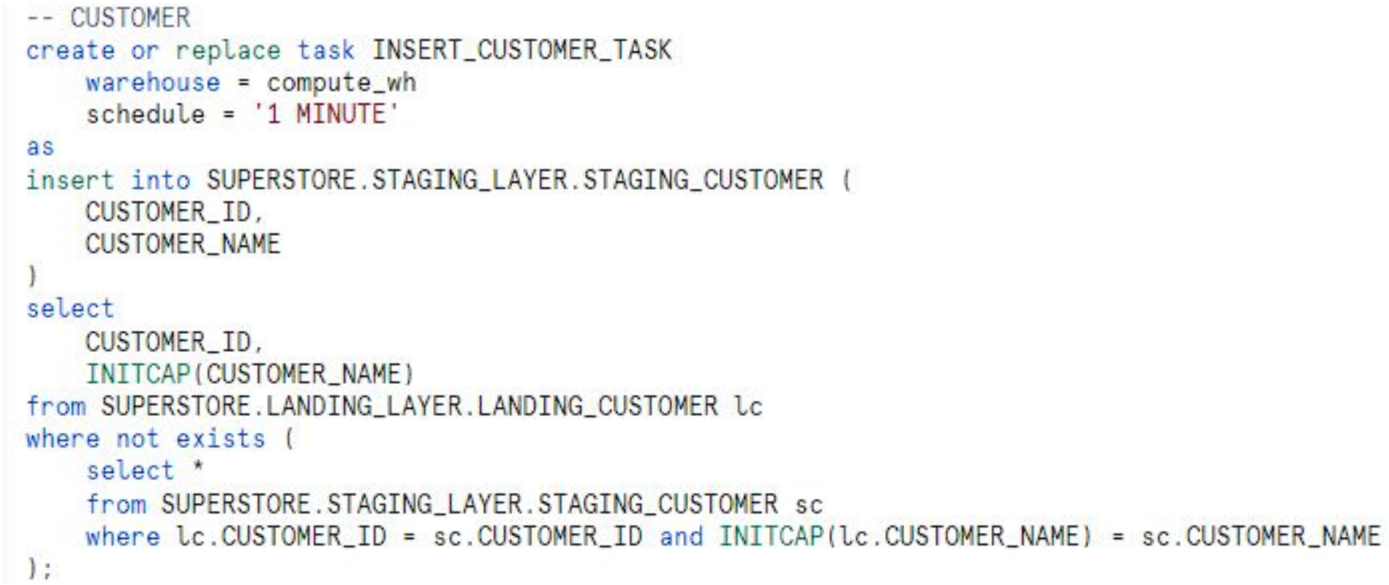


```
-- STAGING_CUSTOMER
create or replace table staging_customer (
    CUSTOMER_ID varchar(20),
    CUSTOMER_NAME varchar(50)
);

-- STAGING_ORDER_INFOR
create or replace table staging_order_infor (
    ORDER_ID varchar(20),
    ORDER_DATE date,
    CUSTOMER_ID varchar(20)
);

-- STAGING_ORDER_PRODUCT
create or replace table staging_order_product (
    ID NUMBER(38,0),
    ORDER_ID VARCHAR(20),
    PRODUCT_ID VARCHAR(20),
    SALES NUMBER(38,4),
    QUANTITY NUMBER(38,0),
    DISCOUNT NUMBER(38,2),
    PROFIT NUMBER(38,4)
);
```

+ Do the same for the others

Project - STAGING LAYER

CREATING 5 TASKs TO SEND NEW DATA TO STAGING-LAYER + Transform

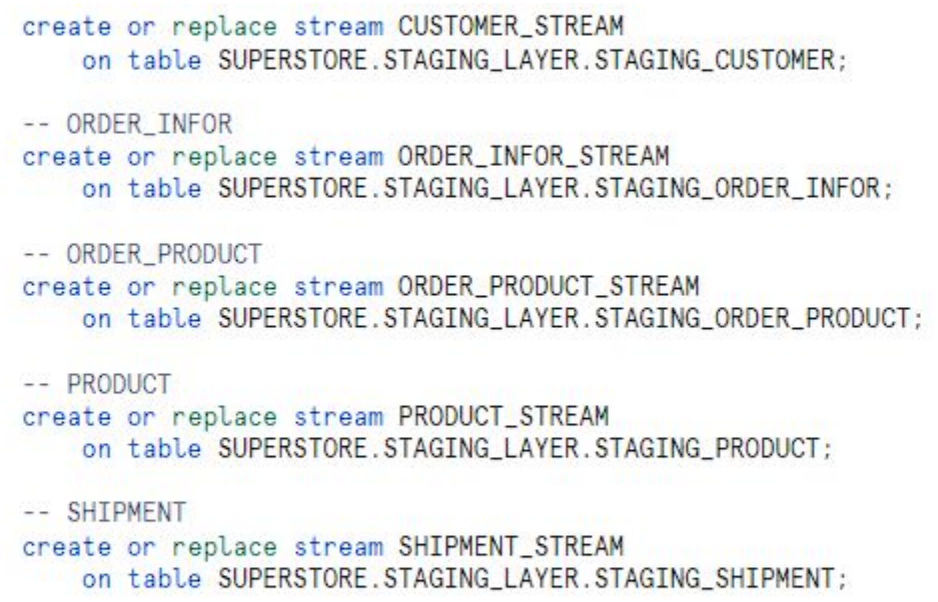


```
-- CUSTOMER
create or replace task INSERT_CUSTOMER_TASK
    warehouse = compute_wh
    schedule = '1 MINUTE'
as
insert into SUPERSTORE.STAGING_LAYER.STAGING_CUSTOMER (
    CUSTOMER_ID,
    CUSTOMER_NAME
)
select
    CUSTOMER_ID,
    INITCAP(CUSTOMER_NAME)
from SUPERSTORE.LANDING_LAYER.LANDING_CUSTOMER lc
where not exists (
    select *
    from SUPERSTORE.STAGING_LAYER.STAGING_CUSTOMER sc
    where lc.CUSTOMER_ID = sc.CUSTOMER_ID and INITCAP(lc.CUSTOMER_NAME) = sc.CUSTOMER_NAME
);
```

+ Do the same for the others

Project - STAGING LAYER

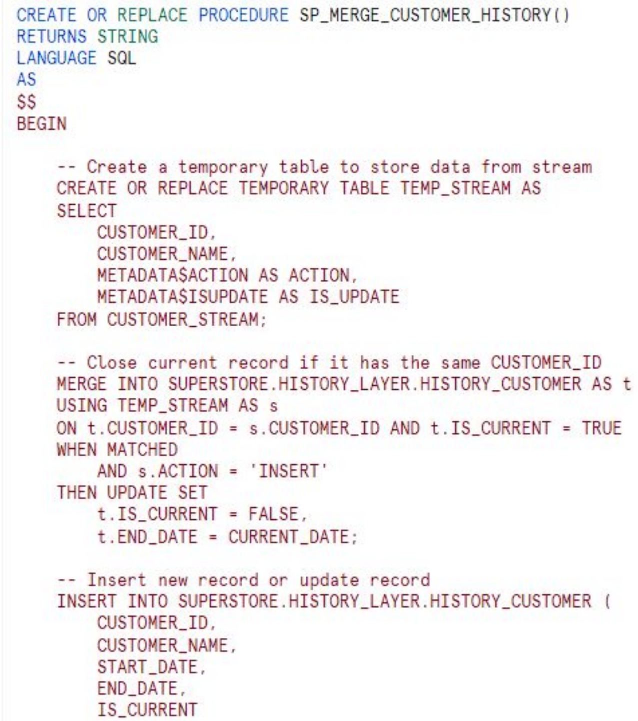
Create 5 streams to capture new data arrival



```
create or replace stream CUSTOMER_STREAM
    on table SUPERSTORE.STAGING_LAYER.STAGING_CUSTOMER;

-- ORDER_INFOR
create or replace stream ORDER_INFOR_STREAM
    on table SUPERSTORE.STAGING_LAYER.STAGING_ORDER_INFOR;

-- ORDER_PRODUCT
create or replace stream ORDER_PRODUCT_STREAM
    on table SUPERSTORE.STAGING_LAYER.STAGING_ORDER_PRODUCT;

-- PRODUCT
create or replace stream PRODUCT_STREAM
    on table SUPERSTORE.STAGING_LAYER.STAGING_PRODUCT;

-- SHIPMENT
create or replace stream SHIPMENT_STREAM
    on table SUPERSTORE.STAGING_LAYER.STAGING_SHIPMENT;
```

Project - STAGING LAYER

Create 5 stored procedure to merge data into HISTORY LAYER

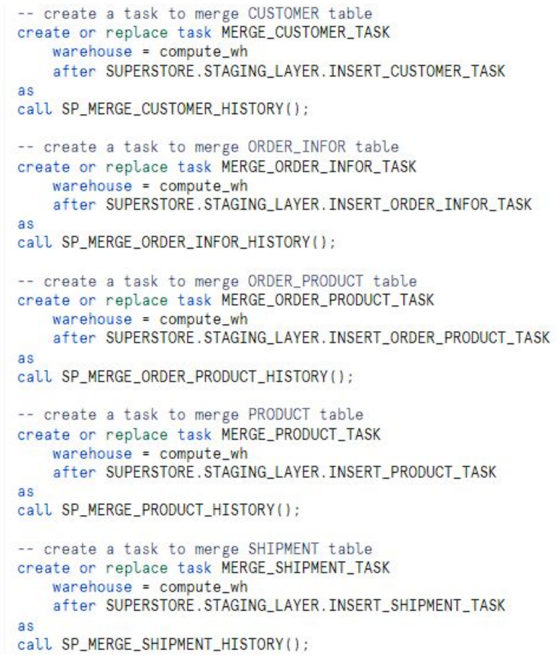


```
CREATE OR REPLACE PROCEDURE SP_MERGE_CUSTOMER_HISTORY()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN

    -- Create a temporary table to store data from stream
    CREATE OR REPLACE TEMPORARY TABLE TEMP_STREAM AS
    SELECT
        CUSTOMER_ID,
        CUSTOMER_NAME,
        METADATA$ACTION AS ACTION,
        METADATA$ISUPDATE AS IS_UPDATE
    FROM CUSTOMER_STREAM;

    -- Close current record if it has the same CUSTOMER_ID
    MERGE INTO SUPERSTORE.HISTORY_LAYER.HISTORY_CUSTOMER AS t
    USING TEMP_STREAM AS s
    ON t.CUSTOMER_ID = s.CUSTOMER_ID AND t.IS_CURRENT = TRUE
    WHEN MATCHED
        AND s.ACTION = 'INSERT'
    THEN UPDATE SET
        t.IS_CURRENT = FALSE,
        t.END_DATE = CURRENT_DATE;

    -- Insert new record or update record
    INSERT INTO SUPERSTORE.HISTORY_LAYER.HISTORY_CUSTOMER (
        CUSTOMER_ID,
        CUSTOMER_NAME,
        START_DATE,
        END_DATE,
        IS_CURRENT
```

+ Do the same for the others

Project - STAGING LAYER

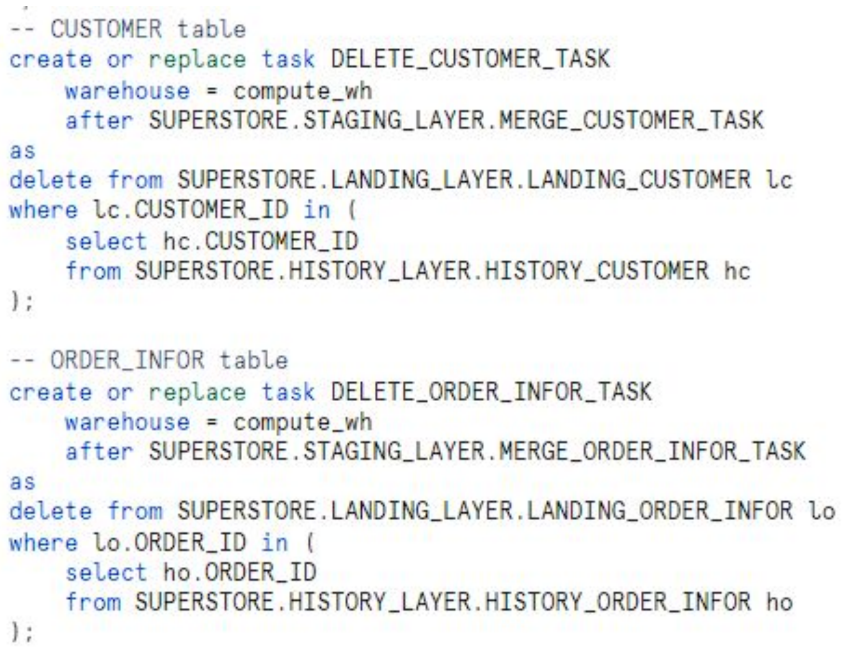
Create 5 tasks which call stored procedure to merge data into HISTORY LAYER



```sql
-- create a task to merge CUSTOMER table
create or replace task MERGE_CUSTOMER_TASK
    warehouse = compute_wh
    after SUPERSTORE.STAGING_LAYER.INSERT_CUSTOMER_TASK
as
call SP_MERGE_CUSTOMER_HISTORY();

-- create a task to merge ORDER_INFOR table
create or replace task MERGE_ORDER_INFOR_TASK
    warehouse = compute_wh
    after SUPERSTORE.STAGING_LAYER.INSERT_ORDER_INFOR_TASK
as
call SP_MERGE_ORDER_INFOR_HISTORY();

-- create a task to merge ORDER_PRODUCT table
create or replace task MERGE_ORDER_PRODUCT_TASK
    warehouse = compute_wh
    after SUPERSTORE.STAGING_LAYER.INSERT_ORDER_PRODUCT_TASK
as
call SP_MERGE_ORDER_PRODUCT_HISTORY();

-- create a task to merge PRODUCT table
create or replace task MERGE_PRODUCT_TASK
    warehouse = compute_wh
    after SUPERSTORE.STAGING_LAYER.INSERT_PRODUCT_TASK
as
call SP_MERGE_PRODUCT_HISTORY();

-- create a task to merge SHIPMENT table
create or replace task MERGE_SHIPMENT_TASK
    warehouse = compute_wh
    after SUPERSTORE.STAGING_LAYER.INSERT_SHIPMENT_TASK
as
call SP_MERGE_SHIPMENT_HISTORY();
```

Project - STAGING LAYER

Create 5 tasks to delete data in LANDING-LAYER after merging into HISTORY LAYER

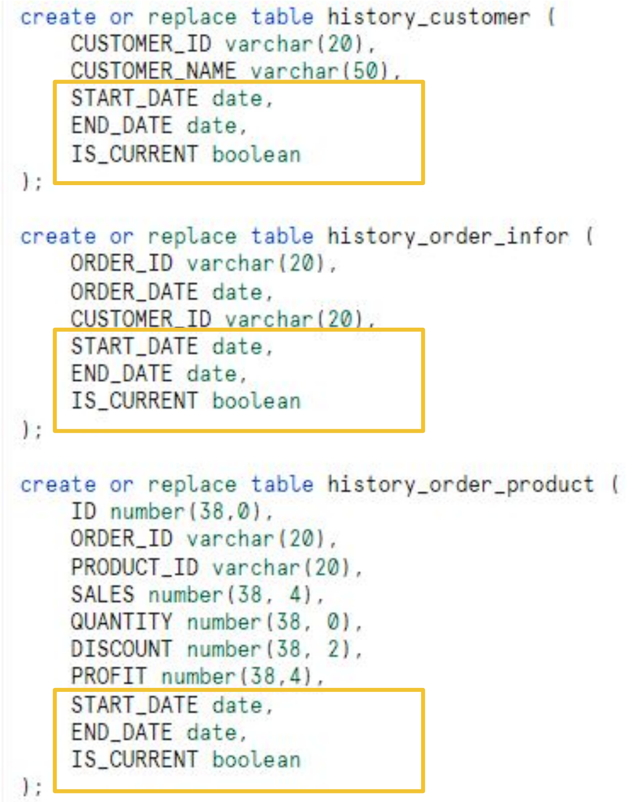


```
-- CUSTOMER table
create or replace task DELETE_CUSTOMER_TASK
    warehouse = compute_wh
    after SUPERSTORE.STAGING_LAYER.MERGE_CUSTOMER_TASK
as
delete from SUPERSTORE.LANDING_LAYER.LANDING_CUSTOMER lc
where lc.CUSTOMER_ID in (
    select hc.CUSTOMER_ID
    from SUPERSTORE.HISTORY_LAYER.HISTORY_CUSTOMER hc
);

-- ORDER_INFOR table
create or replace task DELETE_ORDER_INFOR_TASK
    warehouse = compute_wh
    after SUPERSTORE.STAGING_LAYER.MERGE_ORDER_INFOR_TASK
as
delete from SUPERSTORE.LANDING_LAYER.LANDING_ORDER_INFOR lo
where lo.ORDER_ID in (
    select ho.ORDER_ID
    from SUPERSTORE.HISTORY_LAYER.HISTORY_ORDER_INFOR ho
);
```

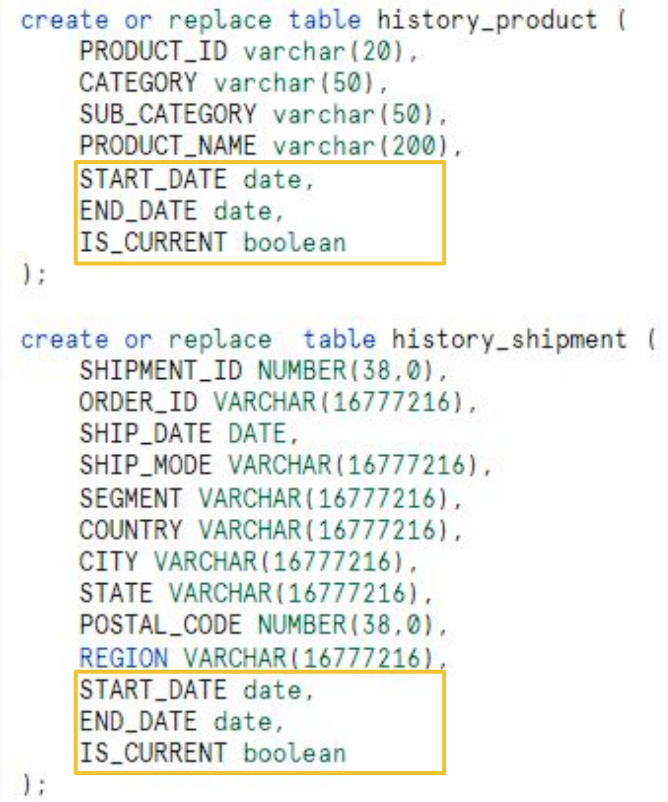
+ Do the same for the others

Project - HISTORY LAYER

Create 5 tables which has 3 more columns to store historical data



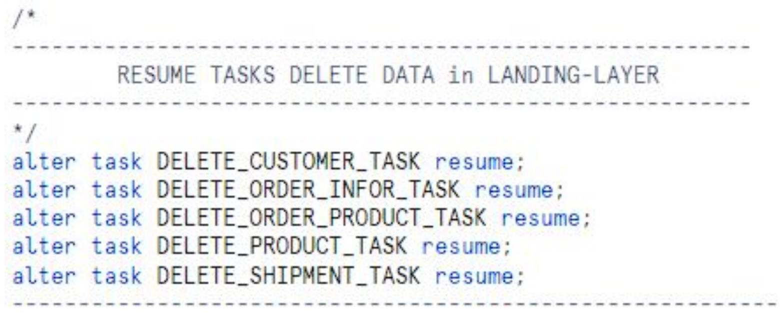
```
create or replace table history_customer (
    CUSTOMER_ID varchar(20),
    CUSTOMER_NAME varchar(50),
    START_DATE date,
    END_DATE date,
    IS_CURRENT boolean
);

create or replace table history_order_infor (
    ORDER_ID varchar(20),
    ORDER_DATE date,
    CUSTOMER_ID varchar(20),
    START_DATE date,
    END_DATE date,
    IS_CURRENT boolean
);

create or replace table history_order_product (
    ID number(38,0),
    ORDER_ID varchar(20),
    PRODUCT_ID varchar(20),
    SALES number(38, 4),
    QUANTITY number(38, 0),
    DISCOUNT number(38, 2),
    PROFIT number(38,4),
    START_DATE date,
    END_DATE date,
    IS_CURRENT boolean
);
```



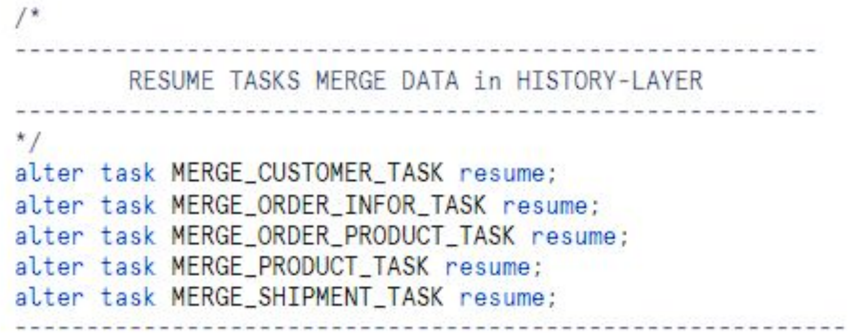
```
create or replace table history_product (
    PRODUCT_ID varchar(20),
    CATEGORY varchar(50),
    SUB_CATEGORY varchar(50),
    PRODUCT_NAME varchar(200),
    START_DATE date,
    END_DATE date,
    IS_CURRENT boolean
);

create or replace  table history_shipment (
    SHIPMENT_ID NUMBER(38,0),
    ORDER_ID VARCHAR(16777216),
    SHIP_DATE DATE,
    SHIP_MODE VARCHAR(16777216),
    SEGMENT VARCHAR(16777216),
    COUNTRY VARCHAR(16777216),
    CITY VARCHAR(16777216),
    STATE VARCHAR(16777216),
    POSTAL_CODE NUMBER(38,0),
    REGION VARCHAR(16777216),
    START_DATE date,
    END_DATE date,
    IS_CURRENT boolean
);
```

Project - RUN PIPELINE

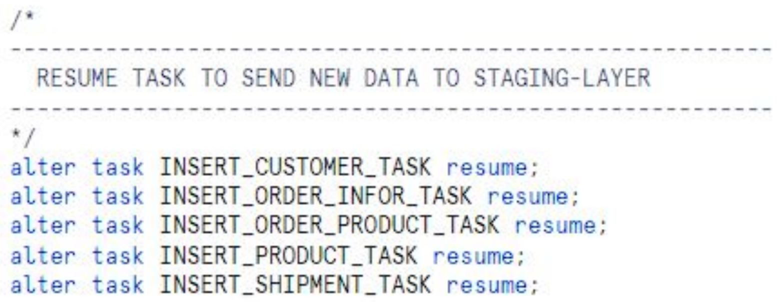
Resume all tasks (resume child task first)



```
/*
--------------------------------------------------------
        RESUME TASKS DELETE DATA in LANDING-LAYER
--------------------------------------------------------
*/
alter task DELETE_CUSTOMER_TASK resume;
alter task DELETE_ORDER_INFOR_TASK resume;
alter task DELETE_ORDER_PRODUCT_TASK resume;
alter task DELETE_PRODUCT_TASK resume;
alter task DELETE_SHIPMENT_TASK resume;
----------------------------------------------------------
```



```
/*
--------------------------------------------------------
        RESUME TASKS MERGE DATA in HISTORY-LAYER
--------------------------------------------------------
*/
alter task MERGE_CUSTOMER_TASK resume;
alter task MERGE_ORDER_INFOR_TASK resume;
alter task MERGE_ORDER_PRODUCT_TASK resume;
alter task MERGE_PRODUCT_TASK resume;
alter task MERGE_SHIPMENT_TASK resume;
----------------------------------------------------------
```

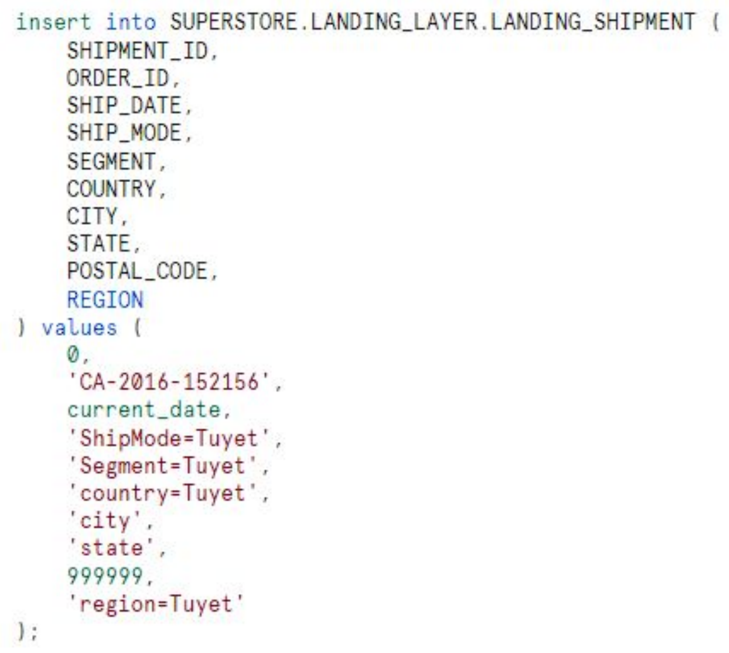


```
/*
--------------------------------------------------------
  RESUME TASK TO SEND NEW DATA TO STAGING-LAYER
--------------------------------------------------------
*/
alter task INSERT_CUSTOMER_TASK resume;
alter task INSERT_ORDER_INFOR_TASK resume;
alter task INSERT_ORDER_PRODUCT_TASK resume;
alter task INSERT_PRODUCT_TASK resume;
alter task INSERT_SHIPMENT_TASK resume;
```

Then check data

Project - RUN PIPELINE

Try to add one new row to LANDING-LAYER SHIPMENT table

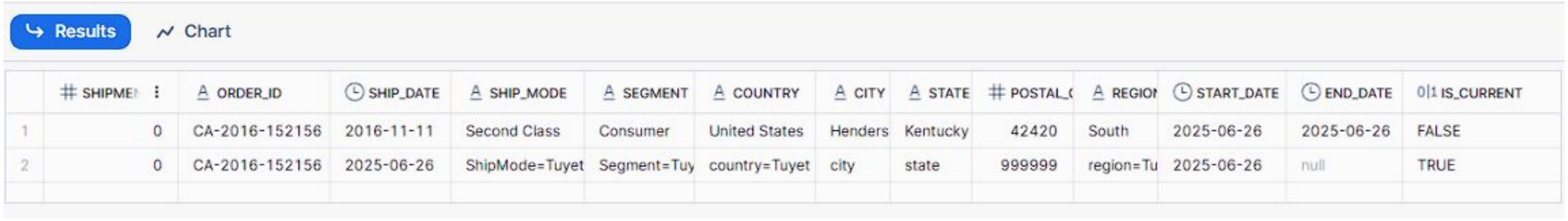


```
insert into SUPERSTORE.LANDING_LAYER.LANDING_SHIPMENT (
    SHIPMENT_ID,
    ORDER_ID,
    SHIP_DATE,
    SHIP_MODE,
    SEGMENT,
    COUNTRY,
    CITY,
    STATE,
    POSTAL_CODE,
    REGION
) values (
    0,
    'CA-2016-152156',
    current_date,
    'ShipMode=Tuyet',
    'Segment=Tuyet',
    'country=Tuyet',
    'city',
    'state',
    999999,
    'region=Tuyet'
);
```

Project - RUN PIPELINE

Result - there is a new row added in HISTORY-LAYER

- Old Shipment
  - End date is today
  - Is_curent becomes FALSE
- New Shipment
  - Start date is today
  - Is_curent becomes TRUE



	SHIPME	ORDER_ID	SHIP_DATE	SHIP_MODE	SEGMENT	COUNTRY	CITY	STATE	POSTAL_	REGIO	START_DATE	END_DATE	IS_CURRENT
1	0	CA-2016-152156	2016-11-11	Second Class	Consumer	United States	Henders	Kentucky	42420	South	2025-06-26	2025-06-26	FALSE
2	0	CA-2016-152156	2025-06-26	ShipMode=Tuyet	Segment=Tuy	country=Tuyet	city	state	999999	region=Tu	2025-06-26	null	TRUE

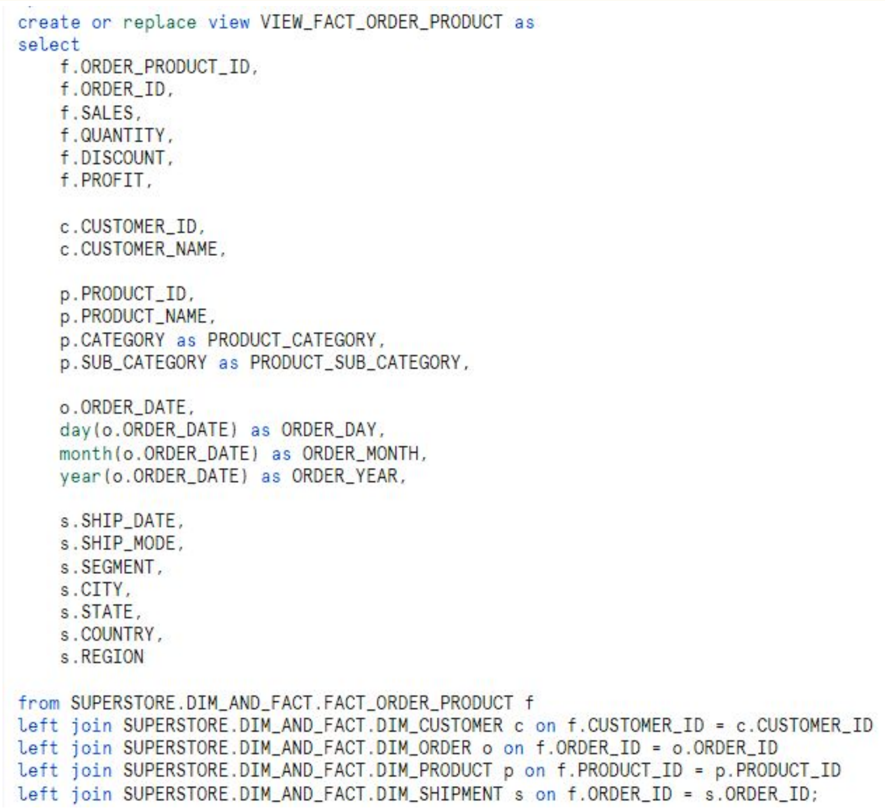
Project - DIM and FACT LAYER

Create DIM tables from HISTORY tables

Create FACT tables from DIM tables

Project - SEMANTIC LAYER

Create a view of each order product



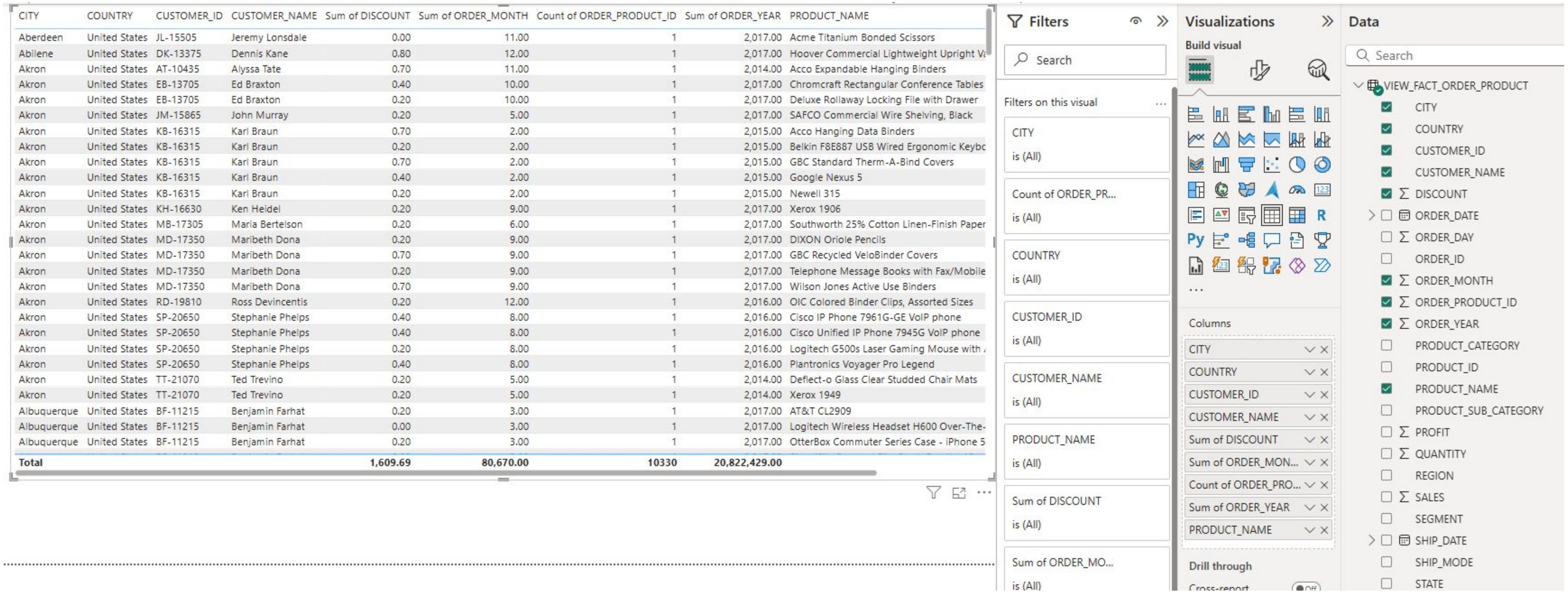
```sql
create or replace view VIEW_FACT_ORDER_PRODUCT as
select
    f.ORDER_PRODUCT_ID,
    f.ORDER_ID,
    f.SALES,
    f.QUANTITY,
    f.DISCOUNT,
    f.PROFIT,

    c.CUSTOMER_ID,
    c.CUSTOMER_NAME,

    p.PRODUCT_ID,
    p.PRODUCT_NAME,
    p.CATEGORY as PRODUCT_CATEGORY,
    p.SUB_CATEGORY as PRODUCT_SUB_CATEGORY,

    o.ORDER_DATE,
    day(o.ORDER_DATE) as ORDER_DAY,
    month(o.ORDER_DATE) as ORDER_MONTH,
    year(o.ORDER_DATE) as ORDER_YEAR,

    s.SHIP_DATE,
    s.SHIP_MODE,
    s.SEGMENT,
    s.CITY,
    s.STATE,
    s.COUNTRY,
    s.REGION

from SUPERSTORE.DIM_AND_FACT.FACT_ORDER_PRODUCT f
left join SUPERSTORE.DIM_AND_FACT.DIM_CUSTOMER c on f.CUSTOMER_ID = c.CUSTOMER_ID
left join SUPERSTORE.DIM_AND_FACT.DIM_ORDER o on f.ORDER_ID = o.ORDER_ID
left join SUPERSTORE.DIM_AND_FACT.DIM_PRODUCT p on f.PRODUCT_ID = p.PRODUCT_ID
left join SUPERSTORE.DIM_AND_FACT.DIM_SHIPMENT s on f.ORDER_ID = s.ORDER_ID;
```

Project - Power BI

Get latest data:

- Refresh data
- Select DirectQuery instead import when choosing table

Project - Power BI



CITY	COUNTRY	CUSTOMER_ID	CUSTOMER_NAME	Sum of DISCOUNT	Sum of ORDER_MONTH	Count of ORDER_PRODUCT_ID	Sum of ORDER_YEAR	PRODUCT_NAME
Aberdeen	United States	JL-15505	Jeremy Lonsdale	0.00	11.00	1	2,017.00	Acme Titanium Bonded Scissors
Abilene	United States	DK-13375	Dennis Kane	0.80	12.00	1	2,017.00	Hoover Commercial Lightweight Upright Va
Akron	United States	AT-10435	Alyssa Tate	0.70	11.00	1	2,014.00	Acco Expandable Hanging Binders
Akron	United States	EB-13705	Ed Braxton	0.40	10.00	1	2,017.00	Chromcraft Rectangular Conference Tables
Akron	United States	EB-13705	Ed Braxton	0.20	10.00	1	2,017.00	Deluxe Rollaway Locking File with Drawer
Akron	United States	JM-15865	John Murray	0.20	5.00	1	2,017.00	SAFCO Commercial Wire Shelving, Black
Akron	United States	KB-16315	Karl Braun	0.70	2.00	1	2,015.00	Acco Hanging Data Binders
Akron	United States	KB-16315	Karl Braun	0.20	2.00	1	2,015.00	Belkin F8E887 USB Wired Ergonomic Keybc
Akron	United States	KB-16315	Karl Braun	0.70	2.00	1	2,015.00	GBC Standard Therm-A-Bind Covers
Akron	United States	KB-16315	Karl Braun	0.40	2.00	1	2,015.00	Google Nexus 5
Akron	United States	KB-16315	Karl Braun	0.20	2.00	1	2,015.00	Newell 315
Akron	United States	KH-16630	Ken Heidel	0.20	9.00	1	2,017.00	Xerox 1906
Akron	United States	MB-17305	Maria Bertelson	0.20	6.00	1	2,017.00	Southworth 25% Cotton Linen-Finish Paper
Akron	United States	MD-17350	Maribeth Dona	0.20	9.00	1	2,017.00	DIXON Oriole Pencils
Akron	United States	MD-17350	Maribeth Dona	0.70	9.00	1	2,017.00	GBC Recycled VeloBinder Covers
Akron	United States	MD-17350	Maribeth Dona	0.20	9.00	1	2,017.00	Telephone Message Books with Fax/Mobile
Akron	United States	MD-17350	Maribeth Dona	0.70	9.00	1	2,017.00	Wilson Jones Active Use Binders
Akron	United States	RD-19810	Ross Devincentis	0.20	12.00	1	2,016.00	OIC Colored Binder Clips, Assorted Sizes
Akron	United States	SP-20650	Stephanie Phelps	0.40	8.00	1	2,016.00	Cisco IP Phone 7961G-GE VoIP phone
Akron	United States	SP-20650	Stephanie Phelps	0.40	8.00	1	2,016.00	Cisco Unified IP Phone 7945G VoIP phone
Akron	United States	SP-20650	Stephanie Phelps	0.20	8.00	1	2,016.00	Logitech G500s Laser Gaming Mouse with
Akron	United States	SP-20650	Stephanie Phelps	0.40	8.00	1	2,016.00	Plantronics Voyager Pro Legend
Akron	United States	TT-21070	Ted Trevino	0.20	5.00	1	2,014.00	Deflect-o Glass Clear Studded Chair Mats
Akron	United States	TT-21070	Ted Trevino	0.20	5.00	1	2,014.00	Xerox 1949
Albuquerque	United States	BF-11215	Benjamin Farhat	0.20	3.00	1	2,017.00	AT&T CL2909
Albuquerque	United States	BF-11215	Benjamin Farhat	0.00	3.00	1	2,017.00	Logitech Wireless Headset H600 Over-The-
Albuquerque	United States	BF-11215	Benjamin Farhat	0.20	3.00	1	2,017.00	OtterBox Commuter Series Case - iPhone 5
Total				**1,609.69**	**80,670.00**	**10330**	**20,822,429.00**	

Filters

Search

Filters on this visual

CITY is (All)

Count of ORDER_PR... is (All)

COUNTRY is (All)

CUSTOMER_ID is (All)

CUSTOMER_NAME is (All)

PRODUCT_NAME is (All)

Sum of DISCOUNT is (All)

Sum of ORDER_MO... is (All)

Visualizations

Build visual

Columns

- CITY
- COUNTRY
- CUSTOMER_ID
- CUSTOMER_NAME
- Sum of DISCOUNT
- Sum of ORDER_MON...
- Count of ORDER_PRO...
- Sum of ORDER_YEAR
- PRODUCT_NAME

Drill through

Cross-report Off

Data

Search

VIEW_FACT_ORDER_PRODUCT

- ✓ CITY
- ✓ COUNTRY
- ✓ CUSTOMER_ID
- ✓ CUSTOMER_NAME
- ✓ Σ DISCOUNT
- ☐ ORDER_DATE
- ☐ Σ ORDER_DAY
- ☐ ORDER_ID
- ✓ Σ ORDER_MONTH
- ✓ Σ ORDER_PRODUCT_ID
- ✓ Σ ORDER_YEAR
- ☐ PRODUCT_CATEGORY
- ☐ PRODUCT_ID
- ✓ PRODUCT_NAME
- ☐ PRODUCT_SUB_CATEGORY
- ☐ Σ PROFIT
- ☐ Σ QUANTITY
- ☐ REGION
- ☐ Σ SALES
- ☐ SEGMENT
- ☐ SHIP_DATE
- ☐ SHIP_MODE
- ☐ STATE