Project Overview

Objective:
Create an end-to-end analytics pipeline for an ecommerce platform to track sales, customers, products, and performance metrics.

Scope:

- Generate synthetic data for orders, products, customers, payments, shipping, and categories (>1 million records).
- Store data in Amazon S3.
- Extract data into Snowflake using Airbyte.
- Transform and model data using dbt.
- Perform analytics in Snowflake.
- Connect Snowflake marts to Power BI for professional dashboards.

Tools & Technologies:

Layer	Tool/Technology
Data Generation	Python
Data Storage	Amazon S3
Data Integration	Airbyte
Data Warehouse	Snowflake
Data Modeling & Transform	dbt
BI & Visualization	Power BI
Programming / Scripting	Python / SQL / DAX

1- Data Generation

```python

# Initialize S3 client
s3 = boto3.client(
    "s3",
    aws_access_key_id=AWS_ACCESS_KEY_ID,
    aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
    region_name=AWS_REGION
)

# ----------------------------
# DATASET SIZES
# ----------------------------
n_customers = 100_000
n_orders = 200_000
n_products = 5_000
n_regions = 20
n_categories = 50


def upload_df_to_s3(df, filename):
    """
    Convert DataFrame → Parquet → Upload to S3 (in-memory)
    """
    table = pa.Table.from_pandas(df)

    buffer = BytesIO()
    pq.write_table(table, buffer)
    buffer.seek(0)

    s3_path = f"{S3_PREFIX}{filename}.parquet"

    s3.upload_fileobj(buffer, BUCKET_NAME, s3_path)

    print(f"✓ Uploaded {filename}.parquet to s3://{BUCKET_NAME}/{s3_path}")

```



```python
# GENERATE SYNTHETIC TABLES
# ----------------------------
regions = pd.DataFrame({
    "region_id": range(1, n_regions + 1),
    "region_name": [f"Region_{i}" for i in range(1, n_regions + 1)]
})

categories = pd.DataFrame({
    "category_id": range(1, n_categories + 1),
    "category_name": [f"Category_{i}" for i in range(1, n_categories + 1)]
})

products = pd.DataFrame({
    "product_id": range(1, n_products + 1),
    "product_name": [f"Product_{i}" for i in range(1, n_products + 1)],
    "category_id": np.random.randint(1, n_categories + 1, n_products),
    "price": np.round(np.random.uniform(5, 2000, n_products), 2)
})

customers = pd.DataFrame({
    "customer_id": range(1, n_customers + 1),
    "first_name": [f"Name{i}" for i in range(1, n_customers + 1)],
    "last_name": [f"Last{i}" for i in range(1, n_customers + 1)],
    "email": [f"user{i}@example.com" for i in range(1, n_customers + 1)],
    "region_id": np.random.randint(1, n_regions + 1, n_customers),
    "created_at": pd.to_datetime("2023-01-01") +
                  pd.to_timedelta(np.random.randint(0, 400, n_customers), "D")
})

orders = pd.DataFrame({
    "order_id": range(1, n_orders + 1),
    "customer_id": np.random.randint(1, n_customers + 1, n_orders),
    "order_date": pd.to_datetime("2023-01-01") +
                  pd.to_timedelta(np.random.randint(0, 400, n_orders), "D"),
    "status": np.random.choice(["Pending", "Shipped", "Delivered"], n_orders),
    "total_amount": np.round(np.random.uniform(10, 3000, n_orders), 2)
})
```

Sales Snowflake > datagen.py > ...

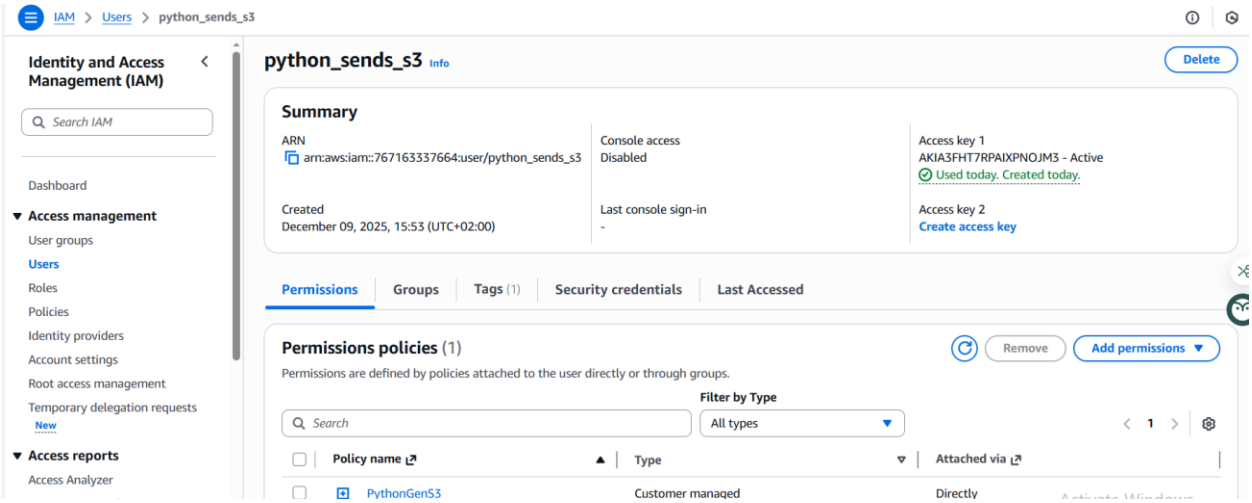
```
            pd.to_timedelta(np.random.randint(0, 400, n_customers), "D")
})

orders = pd.DataFrame({
    "order_id": range(1, n_orders + 1),
    "customer_id": np.random.randint(1, n_customers + 1, n_orders),
    "order_date": pd.to_datetime("2023-01-01") +
            pd.to_timedelta(np.random.randint(0, 400, n_orders), "D"),
    "status": np.random.choice(["Pending", "Shipped", "Delivered"], n_orders),
    "total_amount": np.round(np.random.uniform(10, 3000, n_orders), 2)
})

order_items = pd.DataFrame({
    "order_item_id": range(1, n_orders * 2 + 1),
    "order_id": np.random.randint(1, n_orders + 1, n_orders * 2),
    "product_id": np.random.randint(1, n_products + 1, n_orders * 2),
    "quantity": np.random.randint(1, 5, n_orders * 2)
})

payments = pd.DataFrame({
    "payment_id": range(1, n_orders + 1),
    "order_id": range(1, n_orders + 1),
    "payment_method": np.random.choice(["Card", "PayPal", "Cash"], n_orders),
    "amount": np.round(np.random.uniform(10, 3000, n_orders), 2),
    "payment_date": pd.to_datetime("2023-01-01") +
            pd.to_timedelta(np.random.randint(0, 400, n_orders), "D")
})

shipping = pd.DataFrame({
    "shipping_id": range(1, n_orders + 1),
    "order_id": range(1, n_orders + 1),
    "shipped_date": pd.to_datetime("2023-01-01") +
            pd.to_timedelta(np.random.randint(0, 400, n_orders), "D"),
    "delivery_date": pd.to_datetime("2023-01-01") +
            pd.to_timedelta(np.random.randint(0, 400, n_orders), "D"),
    "shipping_status": np.random.choice(["Pending", "Shipped", "Delivered"], n_orders)
})
```

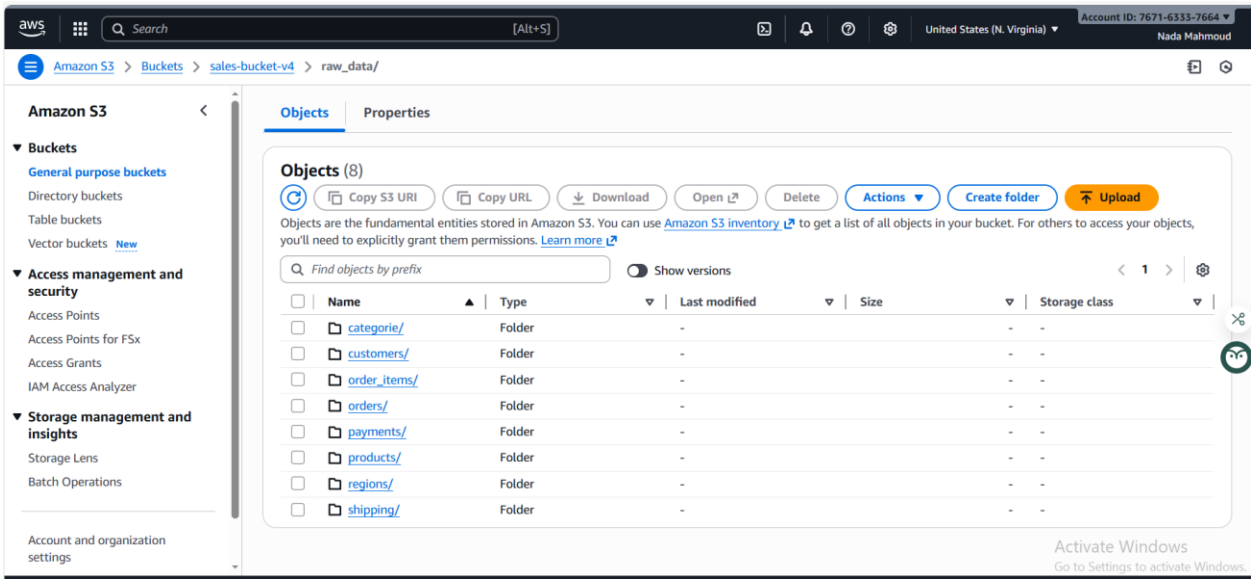
```
# UPLOAD TABLES TO S3
# ----------------------------
upload_df_to_s3(regions, "raw_regions")
upload_df_to_s3(categories, "raw_categories")
upload_df_to_s3(products, "raw_products")
upload_df_to_s3(customers, "raw_customers")
upload_df_to_s3(orders, "raw_orders")
upload_df_to_s3(order_items, "raw_order_items")
upload_df_to_s3(payments, "raw_payments")
upload_df_to_s3(shipping, "raw_shipping")

print("\n🎉 All Parquet files uploaded to S3 successfully!")
```

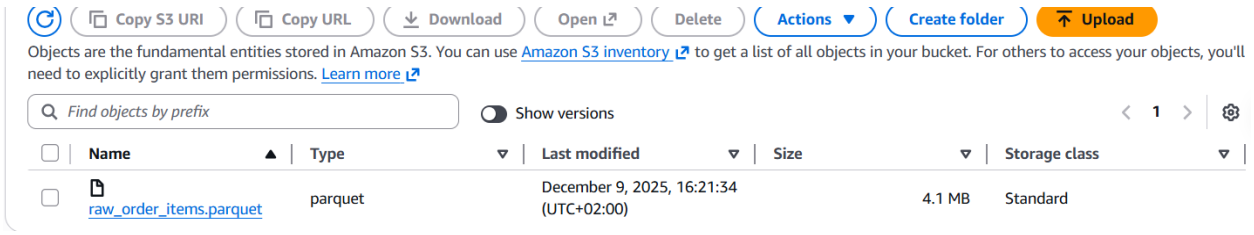
Tables Generated:

- **Orders** (order_id, customer_id, order_date, total_amount, status)
- **Order Items** (order_item_id, order_id, product_id, quantity, price)
- **Products** (product_id, product_name, category_id, price)
- **Customers** (customer_id, first_name, last_name, email, region_id)
- **Categories** (category_id, category_name)
- **Payments** (payment_id, order_id, amount, payment_date, payment_method)
- **Shipping** (shipping_id, order_id, shipped_date, delivery_date, shipping_status)
- **Regions** (region_id, region_name)

2- Data Storage







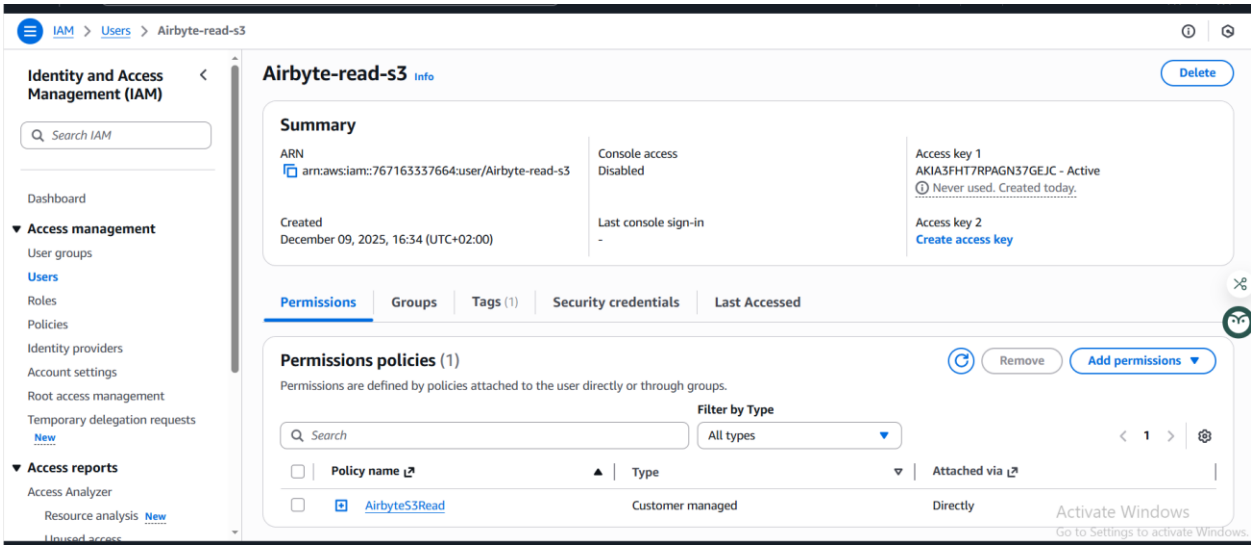
s3://ecommerce-data/

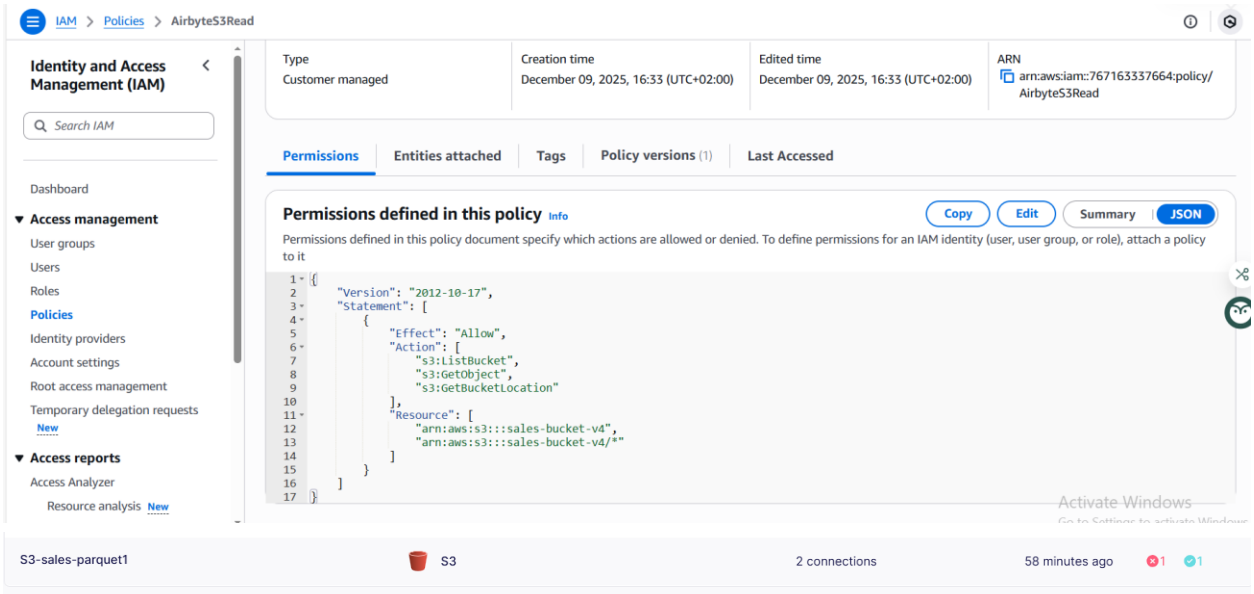
- ├─ orders/
- ├─ order_items/
- ├─ customers/
- ├─ products/
- ├─ categories/
- ├─ payments/
- ├─ shipping/
- ├─ regions/

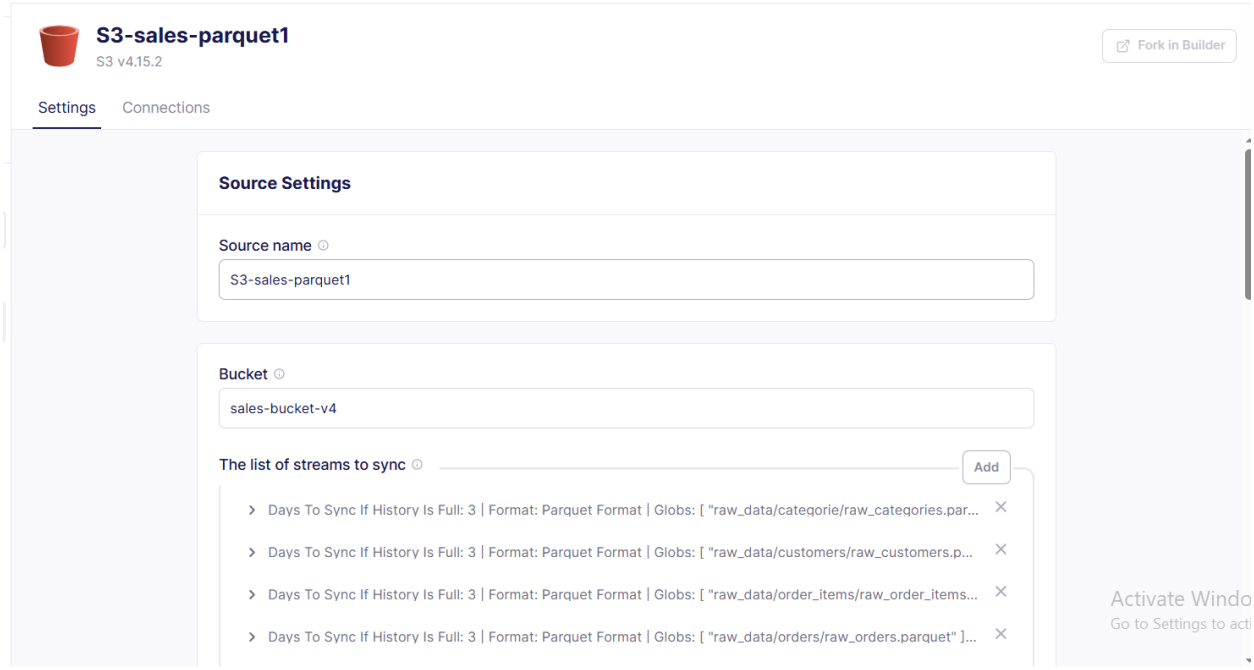
Format: Parquet

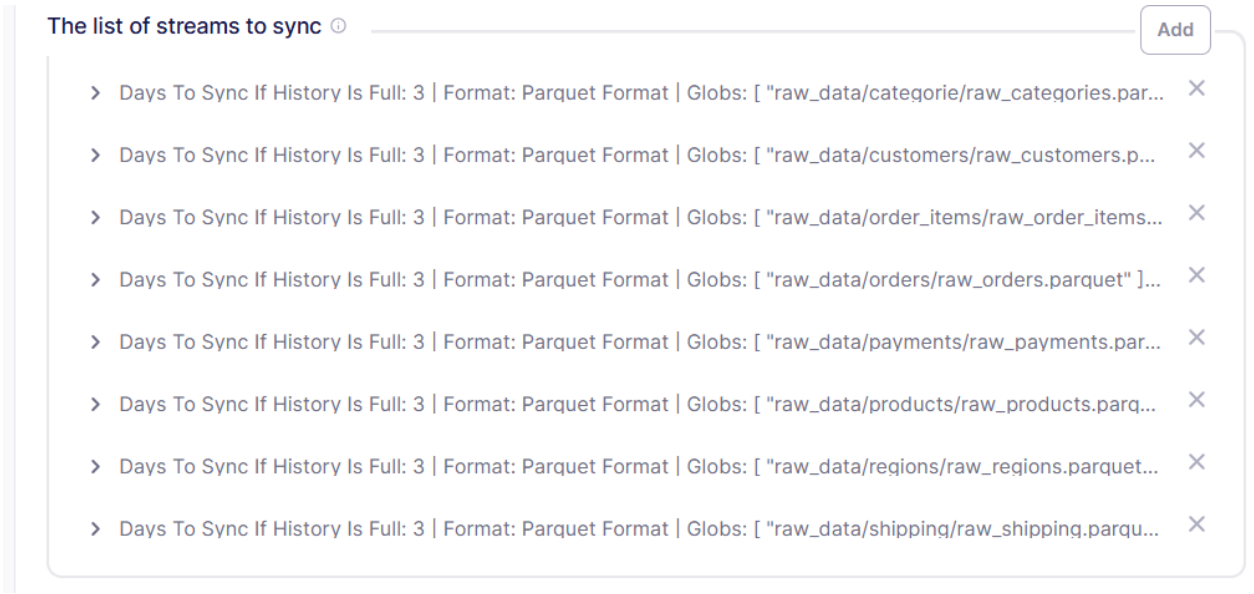
3- Data Extraction & Loading

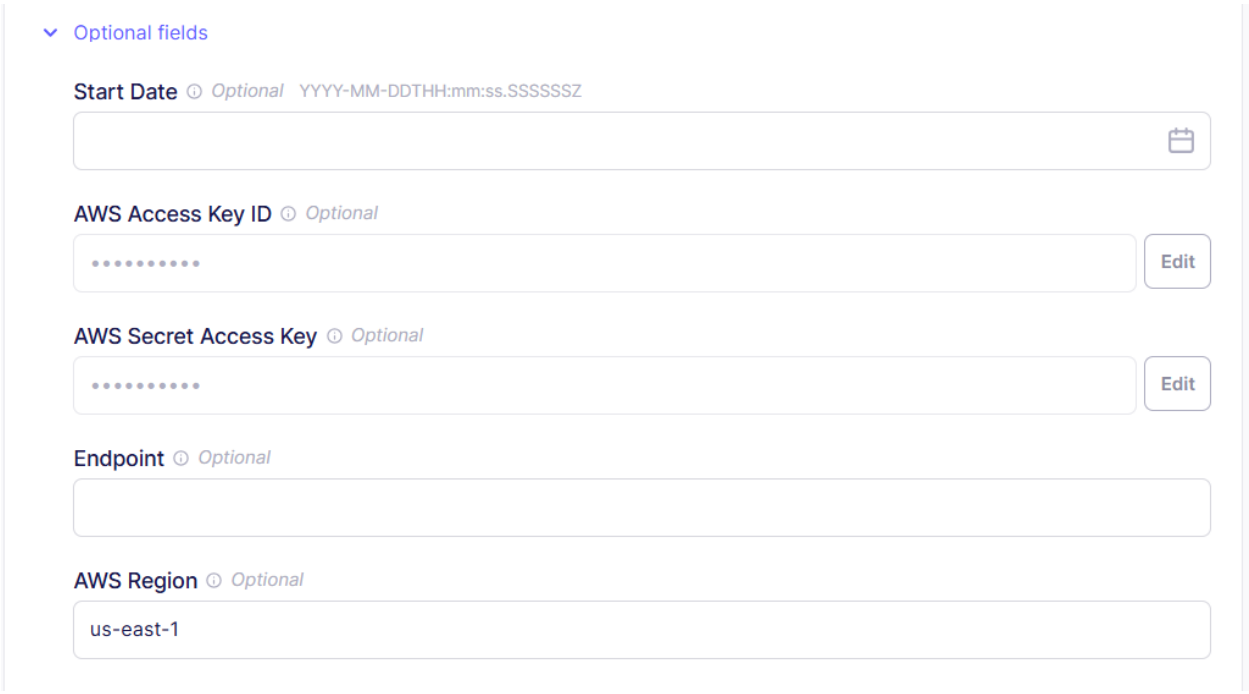
Tool: Airbyte



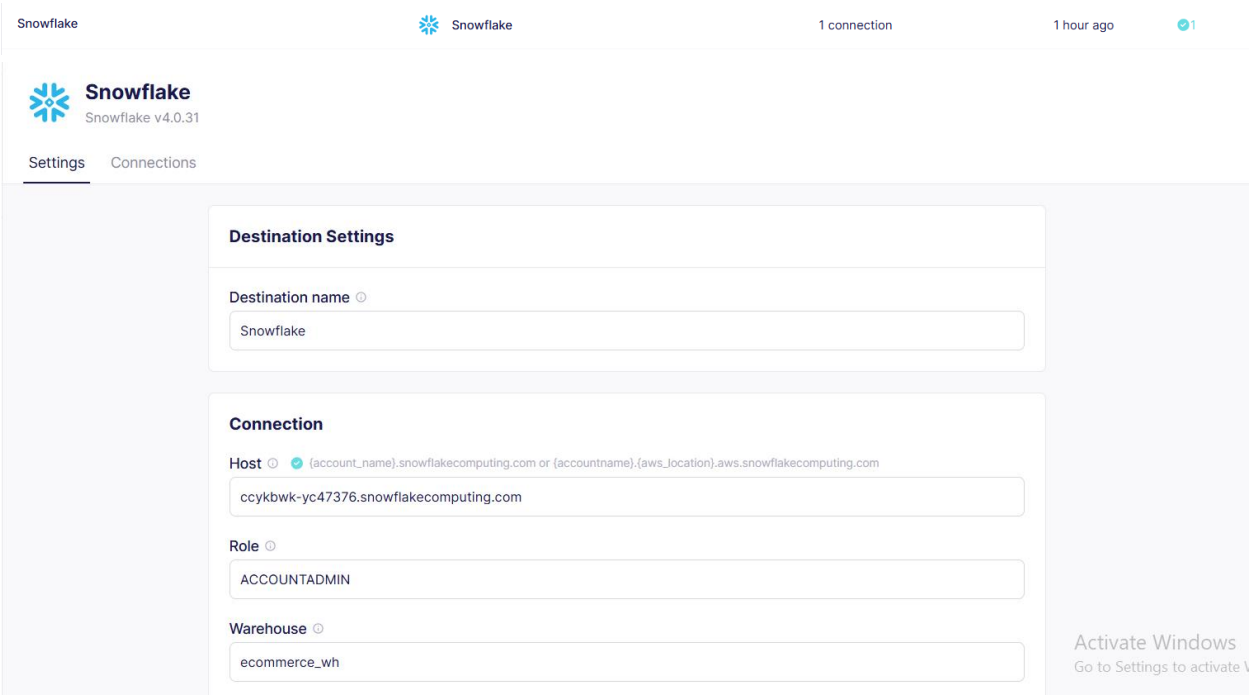



S3-sales-parquet1
S3 v4.15.2
Fork in Builder
Settings
Connections
Source Settings
Source name
S3-sales-parquet1
Bucket
sales-bucket-v4
The list of streams to sync
Add
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/categorie/raw_categories.par...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/customers/raw_customers.p...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/order_items/raw_order_items...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/orders/raw_orders.parquet"]...


The list of streams to sync
Add
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/categorie/raw_categories.par...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/customers/raw_customers.p...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/order_items/raw_order_items...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/orders/raw_orders.parquet"]...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/payments/raw_payments.par...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/products/raw_products.parq...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/regions/raw_regions.parquet...
Days To Sync If History Is Full: 3 | Format: Parquet Format | Globs: ["raw_data/shipping/raw_shipping.parqu...



From s3 load to snowflake



Warehouse

ecommerce_wh

Database

ecommerce_db

Default Schema

raw

Username

dbt_user

Authorization Method — Username and Password

Password

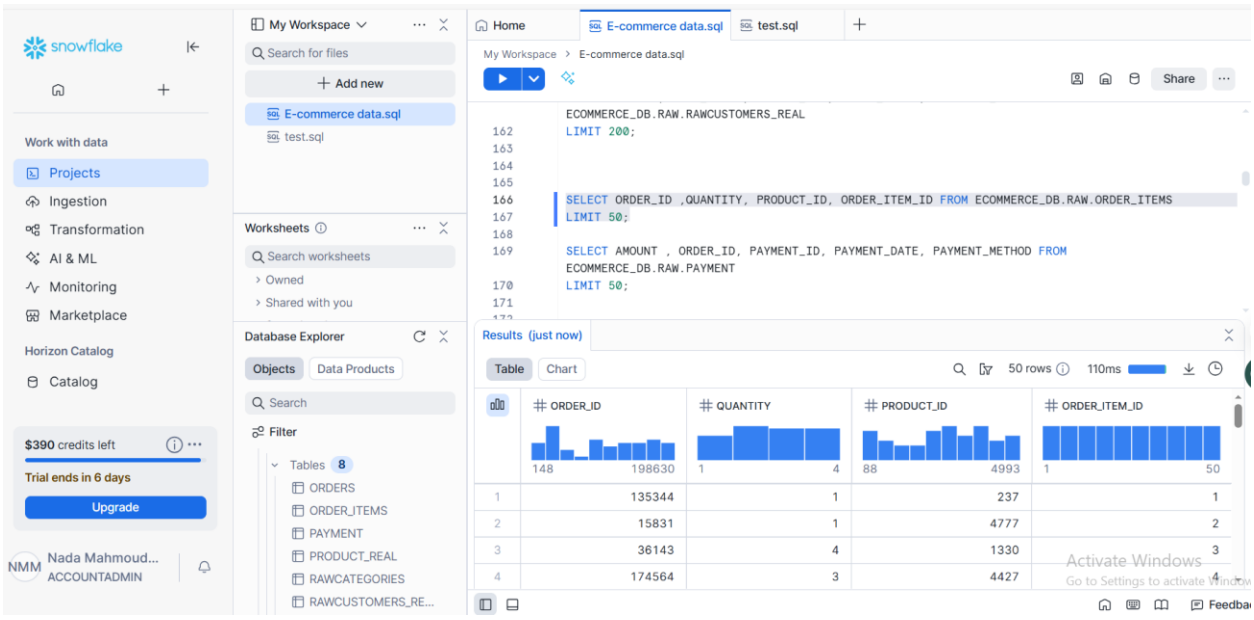
••••••••••

Edit

Active Streams

Search

Status	Stream name	Latest sync in records	Data fresh as of
Synced	categorie	50 loaded	27 minutes ago
Synced	Customers	100,000 loaded	27 minutes ago
Synced	order_items	400,000 loaded	27 minutes ago
Synced	orders	200,000 loaded	27 minutes ago
Synced	payment	200,000 loaded	27 minutes ago
Synced	products	5,000 loaded	27 minutes ago
Synced	regions	20 loaded	27 minutes ago



4- Data Transformation & Modeling

Tool: dbt (Data Build Tool)

Purpose: Transform RAW data into **analytics-ready marts**.

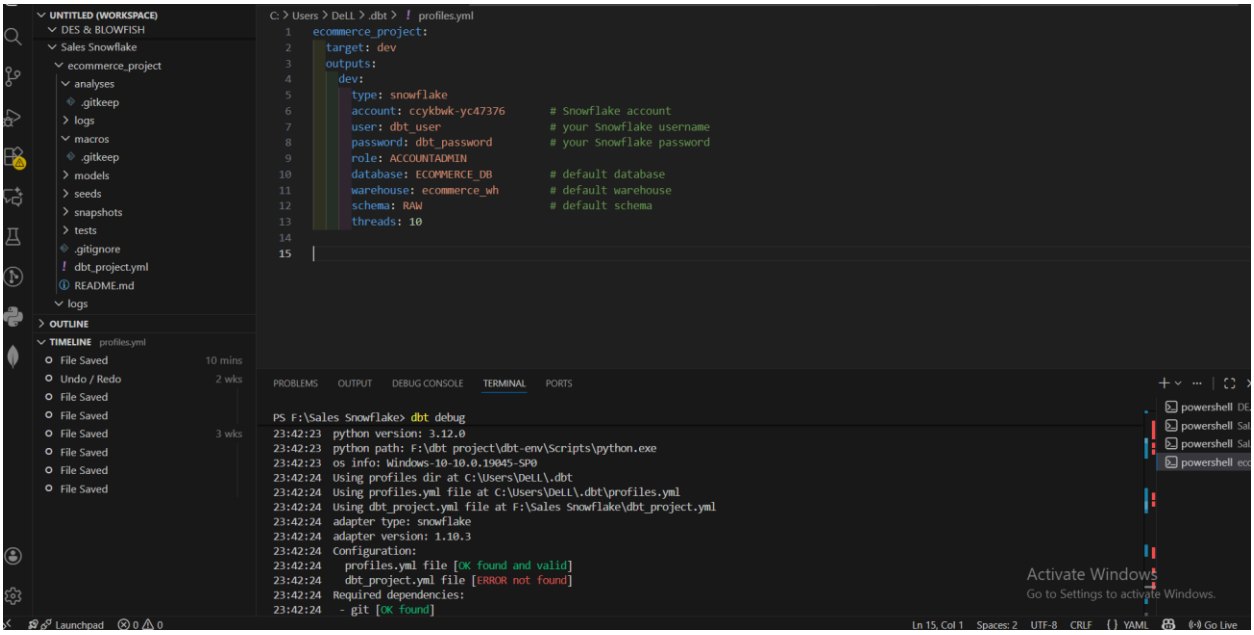
```
PS F:\Sales Snowflake> dbt init ecommerce_project
22:45:38  Running with dbt=1.10.13
22:45:38
Your new dbt project "ecommerce_project" was created!

For more information on how to configure the profiles.yml file,
please consult the dbt documentation here:

  https://docs.getdbt.com/docs/configure-your-profile

One more thing:

Need help? Don't hesitate to reach out to us via GitHub issues or on Slack:
```

```
PS F:\Sales Snowflake> dbt init ecommerce_project
22:45:38  Setting up your profile.
The profile ecommerce_project already exists in C:\Users\DeLL\.dbt\profiles.yml. Continue and overwrite it? [y/N]: y
Which database would you like to use?
[1] postgres
[2] snowflake

(Don't see the one you want? https://docs.getdbt.com/docs/available-adapters)

Enter a number: 2
account (https://<this_value>.snowflakecomputing.com): ccykbwk-yc47376
[1] password
[2] keypair
[3] sso
```

```
[3] sso
Desired authentication type option (enter a number): 1
password (dev password):
role (dev role): ACCOUNTADMIN
warehouse (warehouse name): ecommerce_wh
database (default database that dbt will build objects in): ECOMMERCE_DB
schema (default schema that dbt will build objects in): RAW
threads (1 or more) [1]: 10
22:48:32  Profile ecommerce_project written to C:\Users\DeLL\.dbt\profiles.yml using target's profile_template.yml and your supplied values. Run
PS F:\Sales Snowflake> dbt debug --profiles-dir C:\Users\DeLL\.dbt
22:49:25  Running with dbt=1.10.13
```


Sales Snowflake > ecommerce_project > ! dbt_project.yml

```yaml
name: 'ecommerce_project'
version: '1.0.0'

profile: 'ecommerce_project'

model-paths: ["models"]
analysis-paths: ["analyses"]
test-paths: ["tests"]
seed-paths: ["seeds"]
macro-paths: ["macros"]
snapshot-paths: ["snapshots"]

clean-targets:
  - "target"
  - "dbt_packages"

models:
  ecommerce_project:
    +materialized: view
```




```
23:43:12  Registered adapter: snowflake=1.10.3
23:43:14    Connection test: [OK connection ok]

23:43:14  All checks passed!
```

Transform data

Sales Snowflake > ecommerce_project > models > models > staging > stg_categories.sql

```sql
with raw as (
    select
        category_id,
        category_name
    from {{ source('ecommerce_raw', 'RAWCATEGORIES') }}
)
select * from raw
```

Sales Snowflake > ecommerce_project > models > models > staging > stg_customers.sql

```sql
with raw as (
    select
        customer_id,
        first_name,
        last_name,
        email,
        region_id
    from {{ source('ecommerce_raw', 'RAWCUSTOMERS_REAL') }}
)
select * from raw
```

Sales Snowflake > ecommerce_project > models > models > staging > stg_order_items.sql

```sql
with raw as (
    select
        order_item_id,
        order_id,
        product_id,
        quantity::integer as quantity
    from {{ source('ecommerce_raw', 'ORDER_ITEMS') }}
)
select * from raw
```

Sales Snowflake > ecommerce_project > models > models > staging > stg_orders.sql

```sql
-- models/staging/stg_orders.sql
with raw as (
    select
        order_id,
        customer_id,
        total_amount::decimal as total_amount,
        order_date::timestamp as order_date,
        status
    from {{ source('ecommerce_raw', 'ORDERS') }}
)
select * from raw
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Sales Snowflake > ecommerce_project > models > models > staging > stg_payment.sql

```sql
with raw as (
    select
        payment_id,
        order_id,
        amount::decimal as amount,
        payment_date::timestamp as payment_date,
        payment_method
    from {{ source('ecommerce_raw', 'PAYMENT') }}
)
select * from raw
```

Sales Snowflake > ecommerce_project > models > models > staging > stg_products.sql

```sql
with raw as (
    select
        product_id,
        product_name,
        category_id,
        price::decimal as price
    from {{ source('ecommerce_raw', 'PRODUCT_REAL') }}
)
select * from raw
```

Sales Snowflake > ecommerce_project > models > models > staging > stg_regions.sql

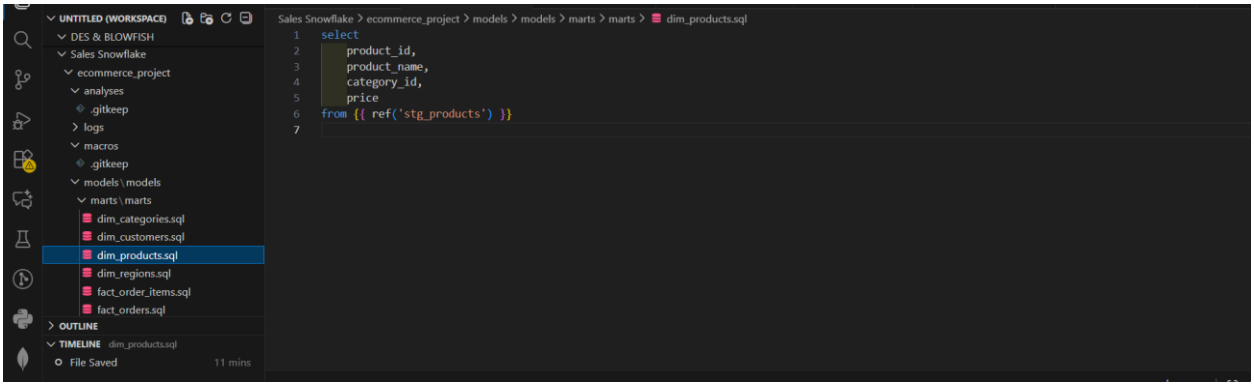
```sql
with raw as (
    select
        region_id,
        region_name
    from {{ source('ecommerce_raw', 'REGION_REAL') }}
)
select * from raw
```

Sales Snowflake > ecommerce_project > models > models > staging > stg_shipping.sql

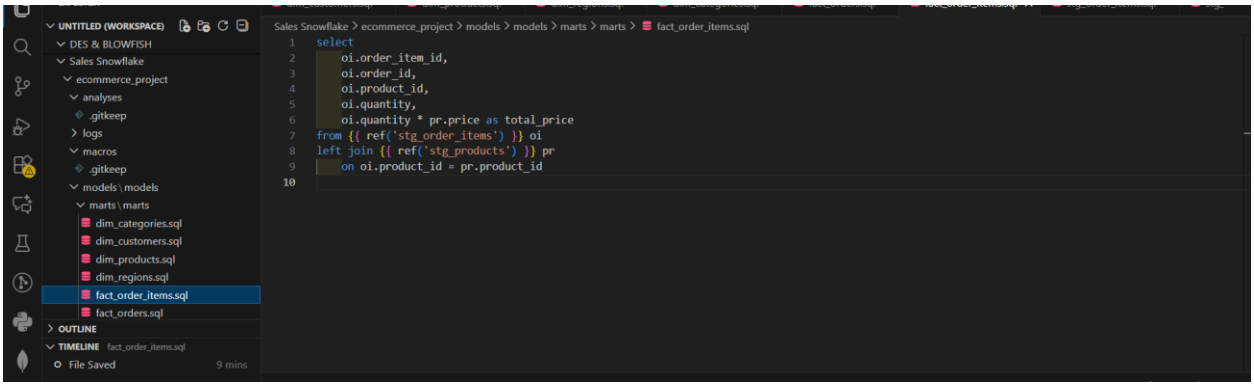
```sql
with raw as (
    select
        shipping_id,
        order_id,
        shipped_date::timestamp as shipped_date,
        delivery_date::timestamp as delivery_date,
        shipping_status
    from {{ source('ecommerce_raw', 'SHIPPING') }}
)
select * from raw
```

Creating marts facts and dimensions

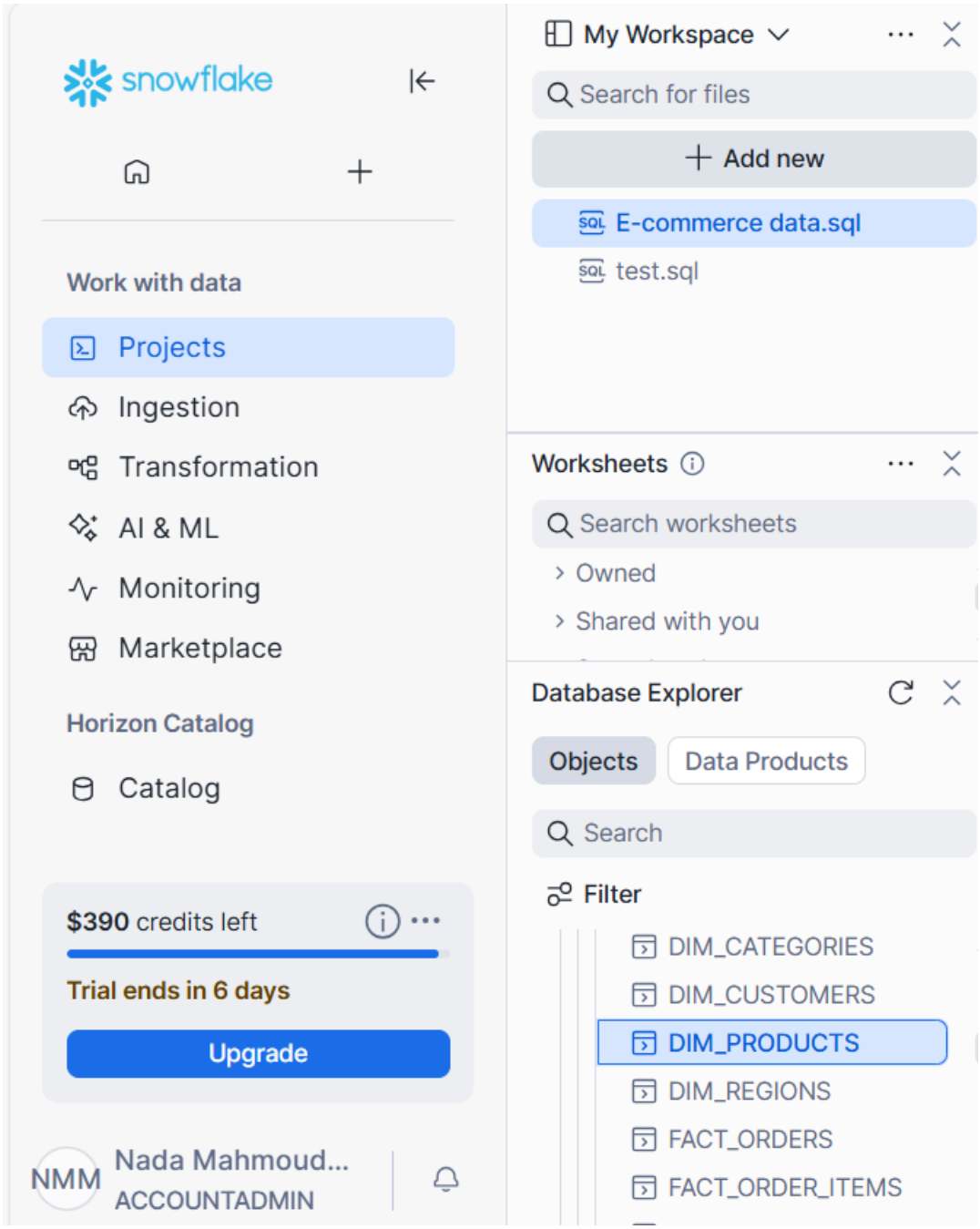

UNTITLED (WORKSPACE)
DES & BLOWFISH
Sales Snowflake
ecommerce_project
analyses
.gitkeep
logs
macros
.gitkeep
models \ models
marts \ marts
dim_categories.sql
dim_customers.sql
dim_products.sql
dim_regions.sql
fact_order_items.sql
fact_orders.sql
OUTLINE
TIMELINE dim_customers.sql
File Saved 11 mins
Sales Snowflake > ecommerce_project > models > models > marts > marts > dim_customers.sql
select
customer_id,
email,
region_id,
first_name || ' ' || last_name as customer_name
from {{ ref('stg_customers') }}


UNTITLED (WORKSPACE)
DES & BLOWFISH
Sales Snowflake
ecommerce_project
analyses
.gitkeep
logs
macros
.gitkeep
models \ models
marts \ marts
dim_categories.sql
dim_customers.sql
dim_products.sql
dim_regions.sql
fact_order_items.sql
fact_orders.sql
OUTLINE
TIMELINE dim_products.sql
File Saved 11 mins
Sales Snowflake > ecommerce_project > models > models > marts > marts > dim_products.sql
select
product_id,
product_name,
category_id,
price
from {{ ref('stg_products') }}

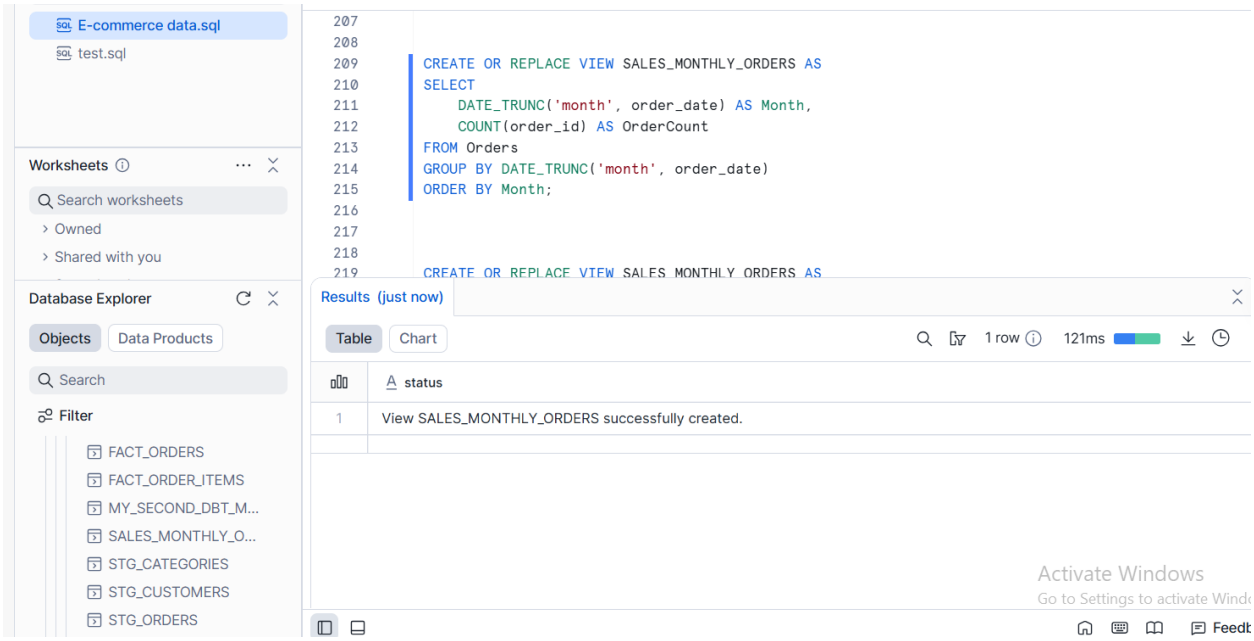

UNTITLED (WORKSPACE)
DES & BLOWFISH
Sales Snowflake
ecommerce_project
analyses
.gitkeep
logs
macros
.gitkeep
models \ models
marts \ marts
dim_categories.sql
dim_customers.sql
dim_products.sql
dim_regions.sql
fact_order_items.sql
fact_orders.sql
OUTLINE
TIMELINE dim_regions.sql
File Saved 11 mins
Sales Snowflake > ecommerce_project > models > models > marts > marts > dim_regions.sql
select
region_id,
region_name
from {{ ref('stg_regions') }}

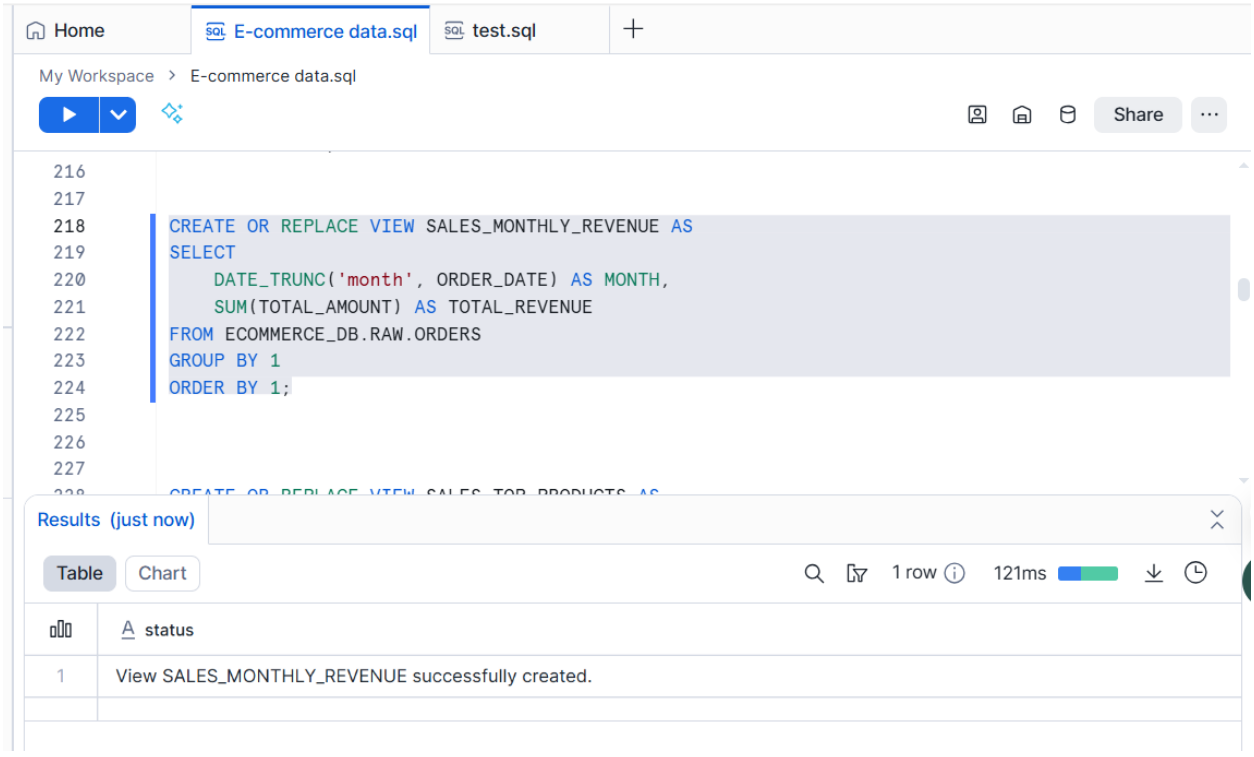

UNTITLED (WORKSPACE)
DES & BLOWFISH
Sales Snowflake
ecommerce_project
analyses
.gitkeep
logs
macros
.gitkeep
models\models
marts\marts
dim_categories.sql
dim_customers.sql
dim_products.sql
dim_regions.sql
fact_order_items.sql
fact_orders.sql
OUTLINE
TIMELINE fact_order_items.sql
File Saved 9 mins
Sales Snowflake > ecommerce_project > models > models > marts > marts > fact_order_items.sql
select
oi.order_item_id,
oi.order_id,
oi.product_id,
oi.quantity,
oi.quantity * pr.price as total_price
from {{ ref('stg_order_items') }} oi
left join {{ ref('stg_products') }} pr
on oi.product_id = pr.product_id


UNTITLED (WORKSPACE)
Sales Snowflake
ecommerce_project
macros
models\models
marts\marts
dim_categories.sql
dim_customers.sql
dim_products.sql
dim_regions.sql
fact_order_items.sql
fact_orders.sql
staging
my_first_dbt_model.sql
my_second_dbt_model.sql
schema.yml
OUTLINE
TIMELINE fact_orders.sql
File Saved 10 mins
Sales Snowflake > ecommerce_project > models > models > marts > marts > fact_orders.sql
select
o.order_id,
o.customer_id,
o.order_date,
o.total_amount,
o.status,
p.amount as payment_amount,
p.payment_method,
s.shipped_date,
s.delivery_date,
s.shipping_status
from {{ ref('stg_orders') }} o
left join {{ ref('stg_payment') }} p
on o.order_id = p.order_id
left join {{ ref('stg_shipping') }} s
on o.order_id = s.order_id


snowflake
Work with data
Projects
Ingestion
Transformation
AI & ML
Monitoring
Marketplace
Horizon Catalog
Catalog
$390 credits left
Trial ends in 6 days
Upgrade
NMM
Nada Mahmoud...
ACCOUNTADMIN
My Workspace
Search for files
Add new
E-commerce data.sql
test.sql
Worksheets
Search worksheets
Owned
Shared with you
Database Explorer
Objects
Data Products
Search
Filter
DIM_CATEGORIES
DIM_CUSTOMERS
DIM_PRODUCTS
DIM_REGIONS
FACT_ORDERS
FACT_ORDER_ITEMS

5- Analytics in Snowflake





Home | E-commerce data.sql | test.sql

My Workspace > E-commerce data.sql

Share

```
227
228 CREATE OR REPLACE VIEW SALES_TOP_PRODUCTS AS
229 SELECT
230     p.PRODUCT_NAME AS PRODUCT_NAME,
231     SUM(oi.TOTAL_PRICE) AS REVENUE
232 FROM ECOMMERCE_DB.RAW.FACT_ORDER_ITEMS oi
233 JOIN ECOMMERCE_DB.RAW.DIM_PRODUCTS p
234     ON oi.PRODUCT_ID = p.PRODUCT_ID
235 GROUP BY p.PRODUCT_NAME
236 ORDER BY REVENUE DESC
237 LIMIT 5;
238 | Ctrl+I to generate
```

Results (just now)

Table | Chart | 1 row | 227ms

	status
1	View SALES_TOP_PRODUCTS successfully created.

Home | E-commerce data.sql | test.sql

My Workspace > E-commerce data.sql

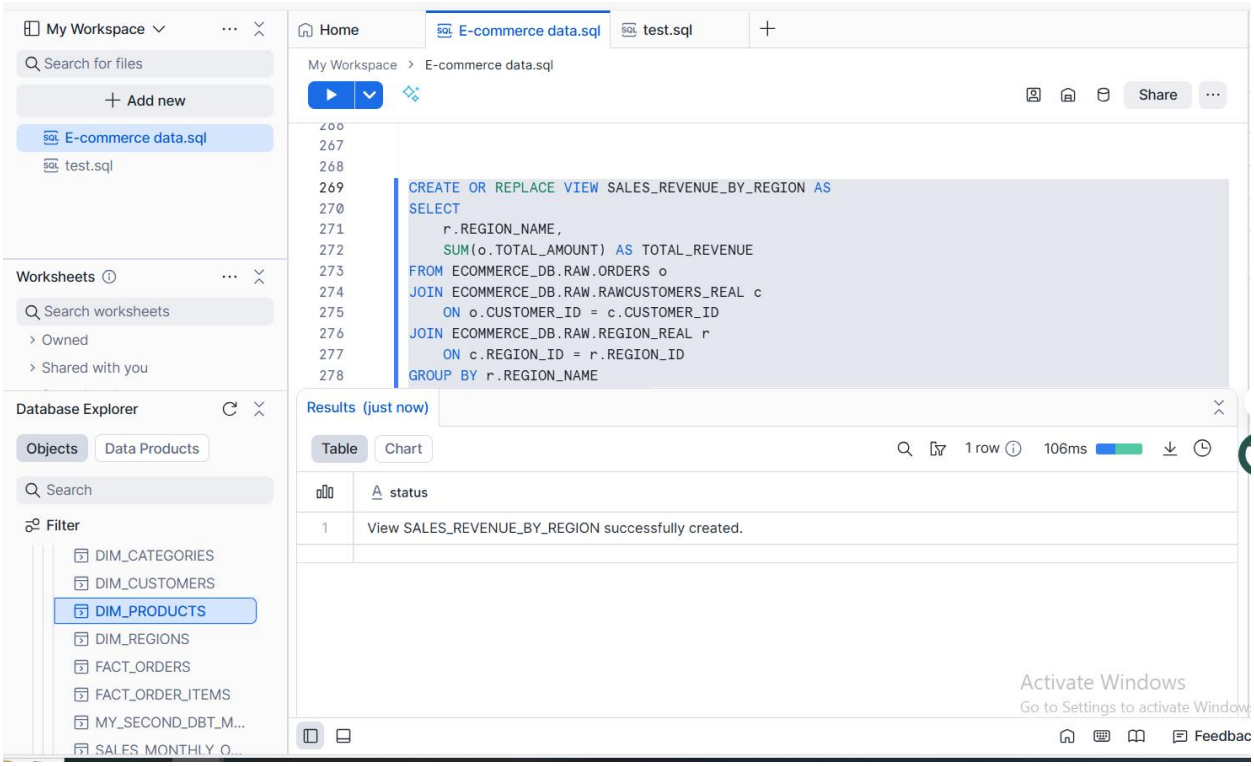
Share

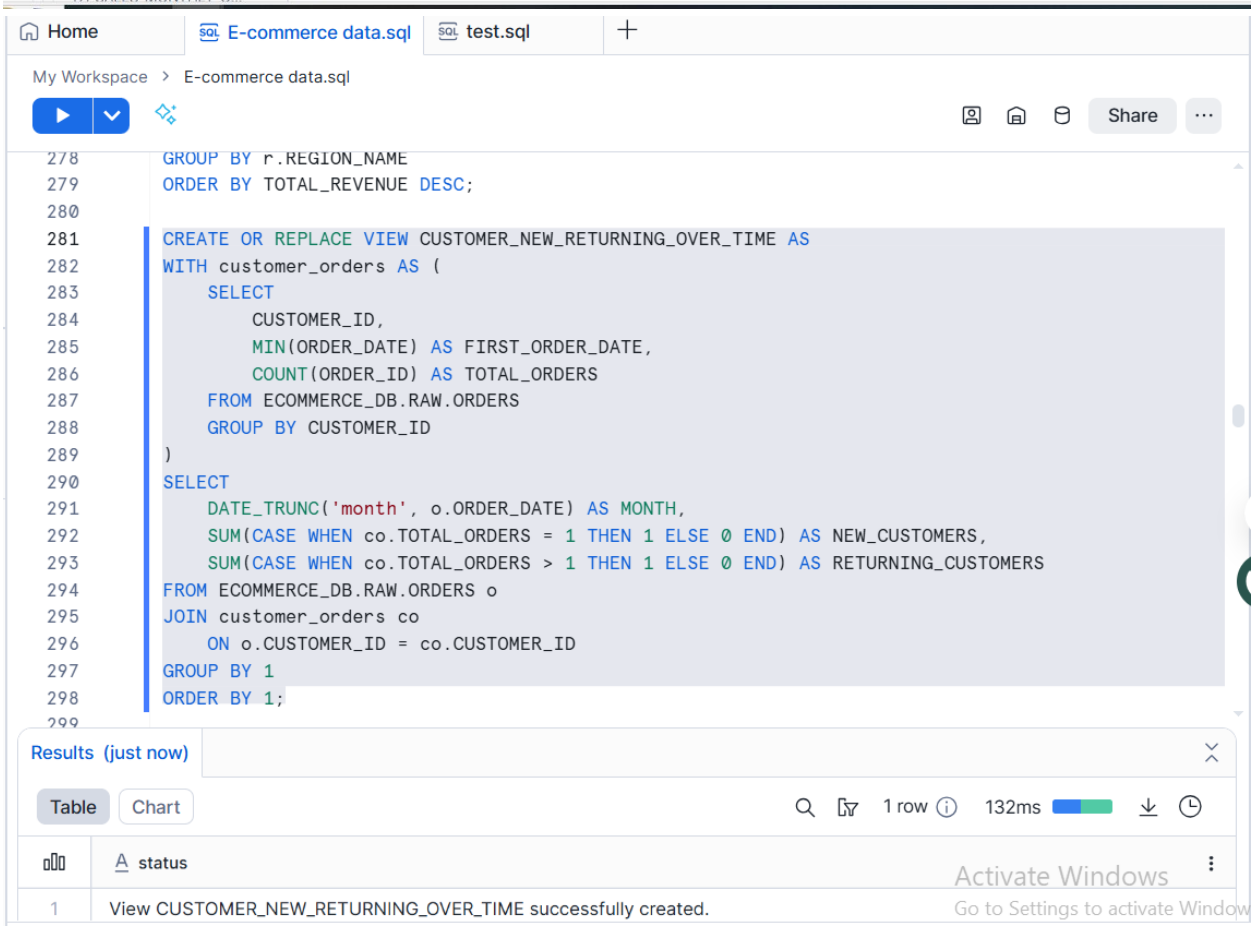
```
254
255 CREATE OR REPLACE VIEW SALES_REVENUE_BY_CATEGORY AS
256 SELECT
257     c.CATEGORY_NAME,
258     SUM(oi.QUANTITY * p.PRICE) AS REVENUE
259 FROM ECOMMERCE_DB.RAW.ORDER_ITEMS oi
260 JOIN ECOMMERCE_DB.RAW.PRODUCT_REAL p
261     ON oi.PRODUCT_ID = p.PRODUCT_ID
262 JOIN ECOMMERCE_DB.RAW.RAWCATEGORIES c
263     ON p.CATEGORY_ID = c.CATEGORY_ID
264 GROUP BY c.CATEGORY_NAME
265 ORDER BY REVENUE DESC;
266
```

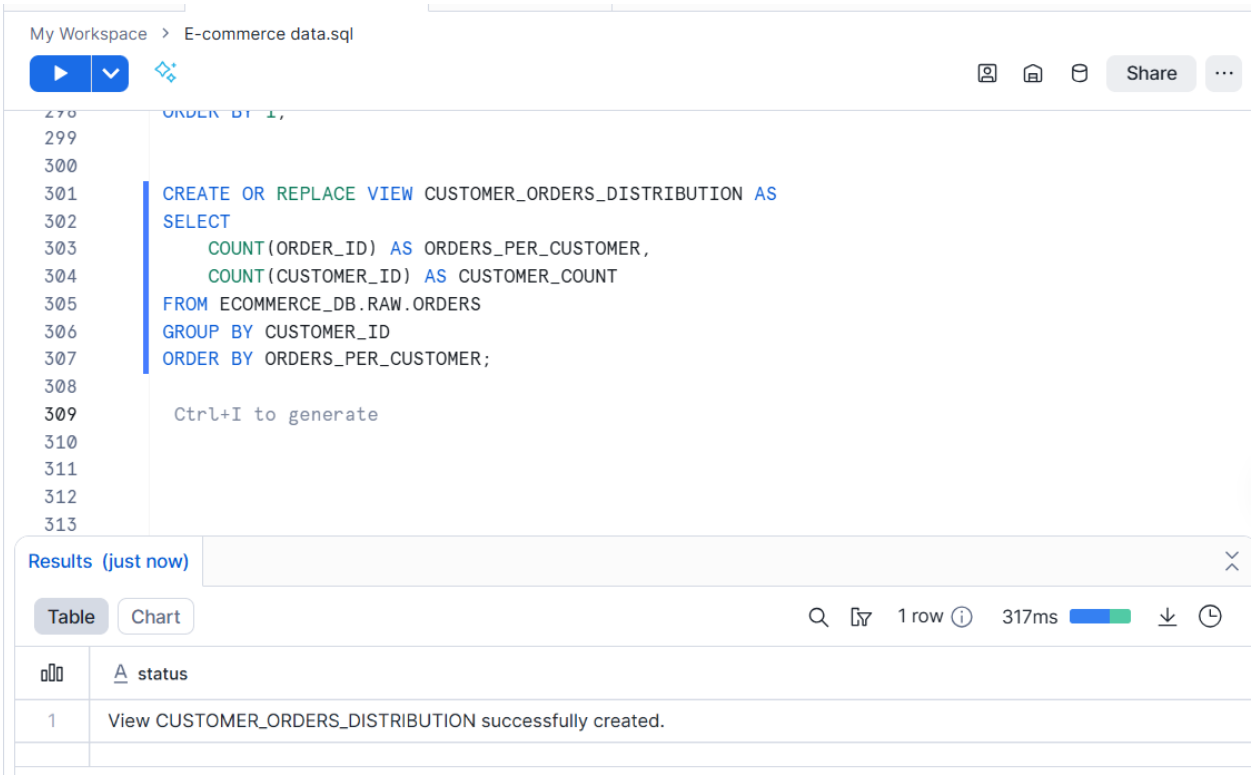
Results (just now)

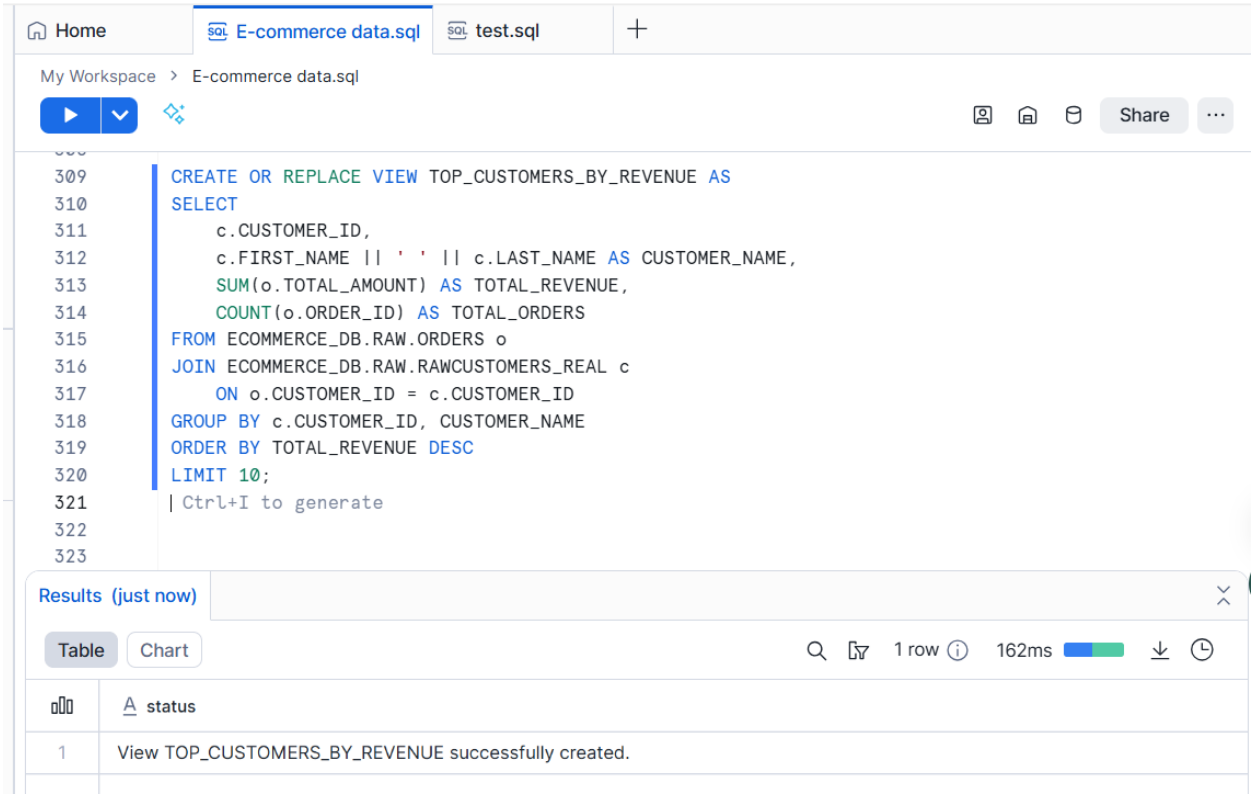
Table | Chart | 1 row | 123ms

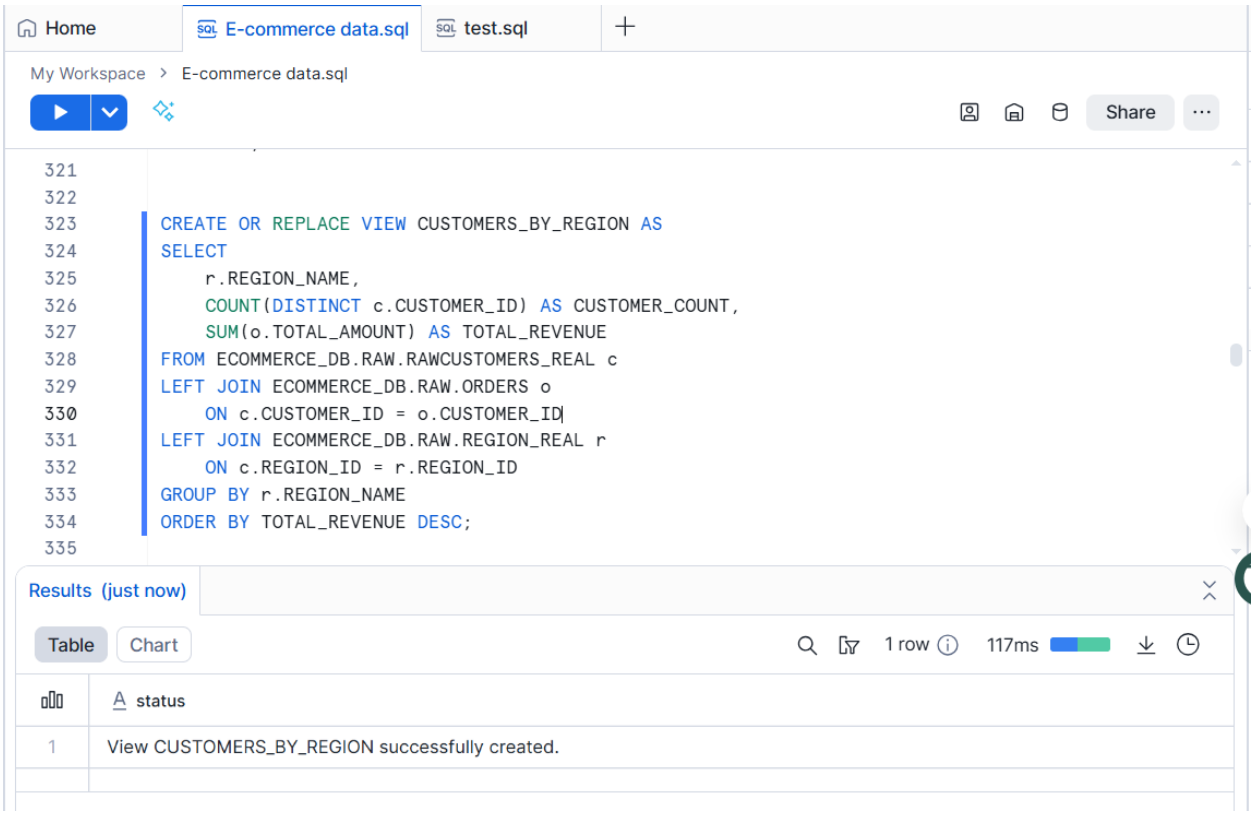
	status
1	View SALES_REVENUE_BY_CATEGORY successfully created.

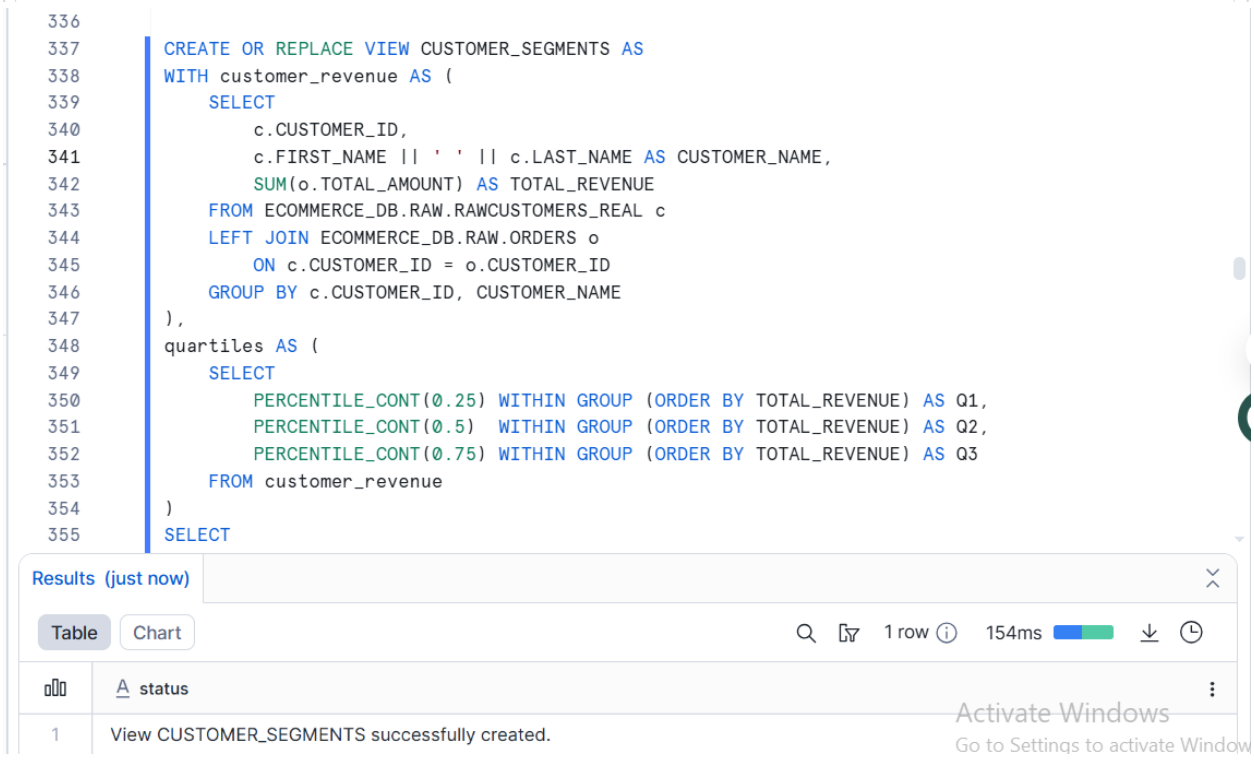

My Workspace
Search for files
Add new
E-commerce data.sql
test.sql
Worksheets
Search worksheets
Owned
Shared with you
Database Explorer
Objects
Data Products
Search
Filter
DIM_CATEGORIES
DIM_CUSTOMERS
DIM_PRODUCTS
DIM_REGIONS
FACT_ORDERS
FACT_ORDER_ITEMS
MY_SECOND_DBT_M...
Home
E-commerce data.sql
test.sql
My Workspace > E-commerce data.sql
Share
268
269 CREATE OR REPLACE VIEW SALES_REVENUE_BY_REGION AS
270 SELECT
271 r.REGION_NAME,
272 SUM(o.TOTAL_AMOUNT) AS TOTAL_REVENUE
273 FROM ECOMMERCE_DB.RAW.ORDERS o
274 JOIN ECOMMERCE_DB.RAW.RAWCUSTOMERS_REAL c
275 ON o.CUSTOMER_ID = c.CUSTOMER_ID
276 JOIN ECOMMERCE_DB.RAW.REGION_REAL r
277 ON c.REGION_ID = r.REGION_ID
278 GROUP BY r.REGION_NAME
Results (just now)
Table
Chart
1 row
106ms
status
1
View SALES_REVENUE_BY_REGION successfully created.
Activate Windows
Go to Settings to activate Windows
Feedbac


Home
E-commerce data.sql
test.sql
My Workspace > E-commerce data.sql
Share
278 GROUP BY r.REGION_NAME
279 ORDER BY TOTAL_REVENUE DESC;
280
281 CREATE OR REPLACE VIEW CUSTOMER_NEW_RETURNING_OVER_TIME AS
282 WITH customer_orders AS (
283 SELECT
284 CUSTOMER_ID,
285 MIN(ORDER_DATE) AS FIRST_ORDER_DATE,
286 COUNT(ORDER_ID) AS TOTAL_ORDERS
287 FROM ECOMMERCE_DB.RAW.ORDERS
288 GROUP BY CUSTOMER_ID
289)
290 SELECT
291 DATE_TRUNC('month', o.ORDER_DATE) AS MONTH,
292 SUM(CASE WHEN co.TOTAL_ORDERS = 1 THEN 1 ELSE 0 END) AS NEW_CUSTOMERS,
293 SUM(CASE WHEN co.TOTAL_ORDERS > 1 THEN 1 ELSE 0 END) AS RETURNING_CUSTOMERS
294 FROM ECOMMERCE_DB.RAW.ORDERS o
295 JOIN customer_orders co
296 ON o.CUSTOMER_ID = co.CUSTOMER_ID
297 GROUP BY 1
298 ORDER BY 1;
299
Results (just now)
Table
Chart
1 row
132ms
status
1
View CUSTOMER_NEW_RETURNING_OVER_TIME successfully created.
Activate Windows
Go to Settings to activate Windows


My Workspace > E-commerce data.sql
Share
ORDER BY 1;
CREATE OR REPLACE VIEW CUSTOMER_ORDERS_DISTRIBUTION AS
SELECT
COUNT(ORDER_ID) AS ORDERS_PER_CUSTOMER,
COUNT(CUSTOMER_ID) AS CUSTOMER_COUNT
FROM ECOMMERCE_DB.RAW.ORDERS
GROUP BY CUSTOMER_ID
ORDER BY ORDERS_PER_CUSTOMER;
Ctrl+I to generate
Results (just now)
Table
Chart
1 row
317ms
status
1
View CUSTOMER_ORDERS_DISTRIBUTION successfully created.

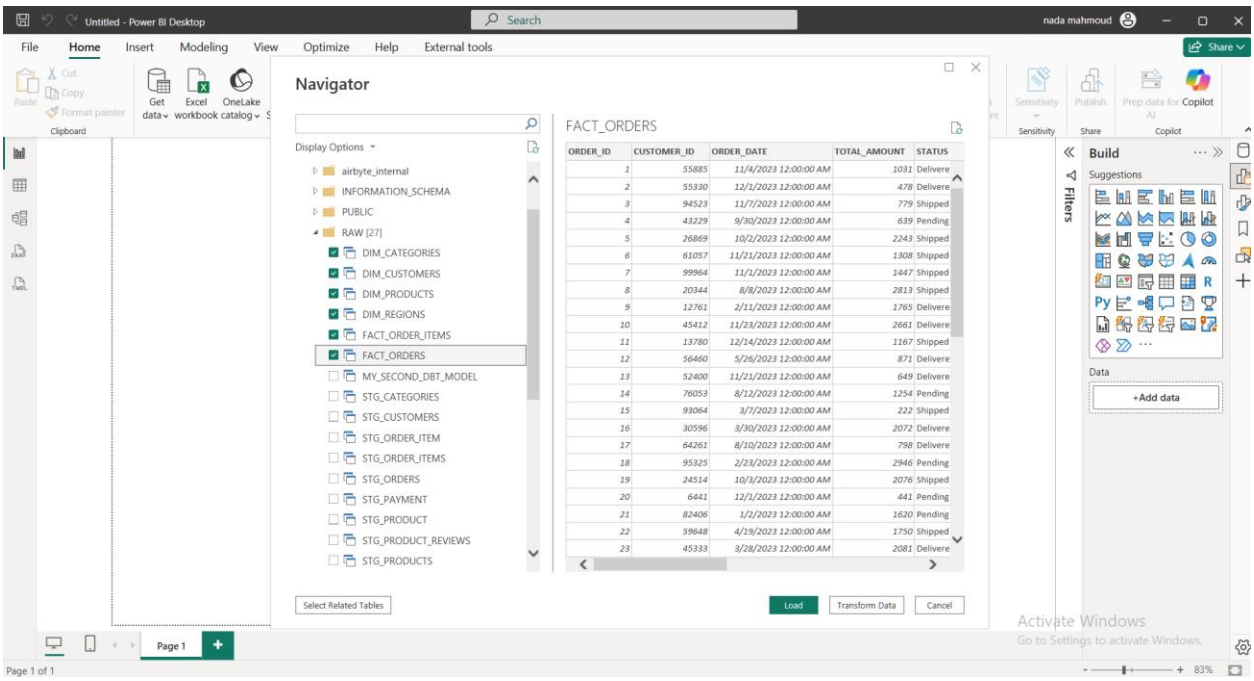

Home
E-commerce data.sql
test.sql
My Workspace > E-commerce data.sql
Share
CREATE OR REPLACE VIEW TOP_CUSTOMERS_BY_REVENUE AS
SELECT
c.CUSTOMER_ID,
c.FIRST_NAME || ' ' || c.LAST_NAME AS CUSTOMER_NAME,
SUM(o.TOTAL_AMOUNT) AS TOTAL_REVENUE,
COUNT(o.ORDER_ID) AS TOTAL_ORDERS
FROM ECOMMERCE_DB.RAW.ORDERS o
JOIN ECOMMERCE_DB.RAW.RAWCUSTOMERS_REAL c
ON o.CUSTOMER_ID = c.CUSTOMER_ID
GROUP BY c.CUSTOMER_ID, CUSTOMER_NAME
ORDER BY TOTAL_REVENUE DESC
LIMIT 10;
Ctrl+I to generate
Results (just now)
Table
Chart
1 row
162ms
status
1
View TOP_CUSTOMERS_BY_REVENUE successfully created.

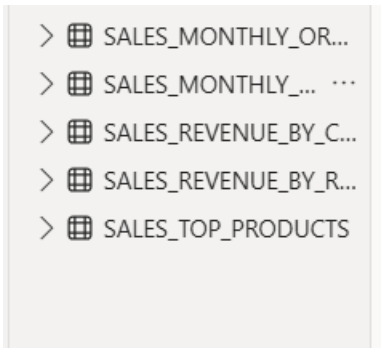




```sql
    SELECT
        PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY TOTAL_REVENUE) AS Q1,
        PERCENTILE_CONT(0.5)  WITHIN GROUP (ORDER BY TOTAL_REVENUE) AS Q2,
        PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY TOTAL_REVENUE) AS Q3
    FROM customer_revenue
)
SELECT
    cr.CUSTOMER_ID,
    cr.CUSTOMER_NAME,
    cr.TOTAL_REVENUE,
    CASE
        WHEN cr.TOTAL_REVENUE >= q.Q3 THEN 'High'
        WHEN cr.TOTAL_REVENUE >= q.Q2 THEN 'Medium'
        ELSE 'Low'
    END AS REVENUE_SEGMENT
FROM customer_revenue cr
CROSS JOIN quartiles q
ORDER BY cr.TOTAL_REVENUE DESC;
```

6- Connect to Power BI




SALES_MONTHLY_OR...
SALES_MONTHLY_...
SALES_REVENUE_BY_C...
SALES_REVENUE_BY_R...
SALES_TOP_PRODUCTS


CUSTOMER_NEW_RETURNING_OVER_TIME
CUSTOMER_ORDERS_DISTRIBUTION
CUSTOMER_SEGMENTS
CUSTOMERS_BY_REGION