

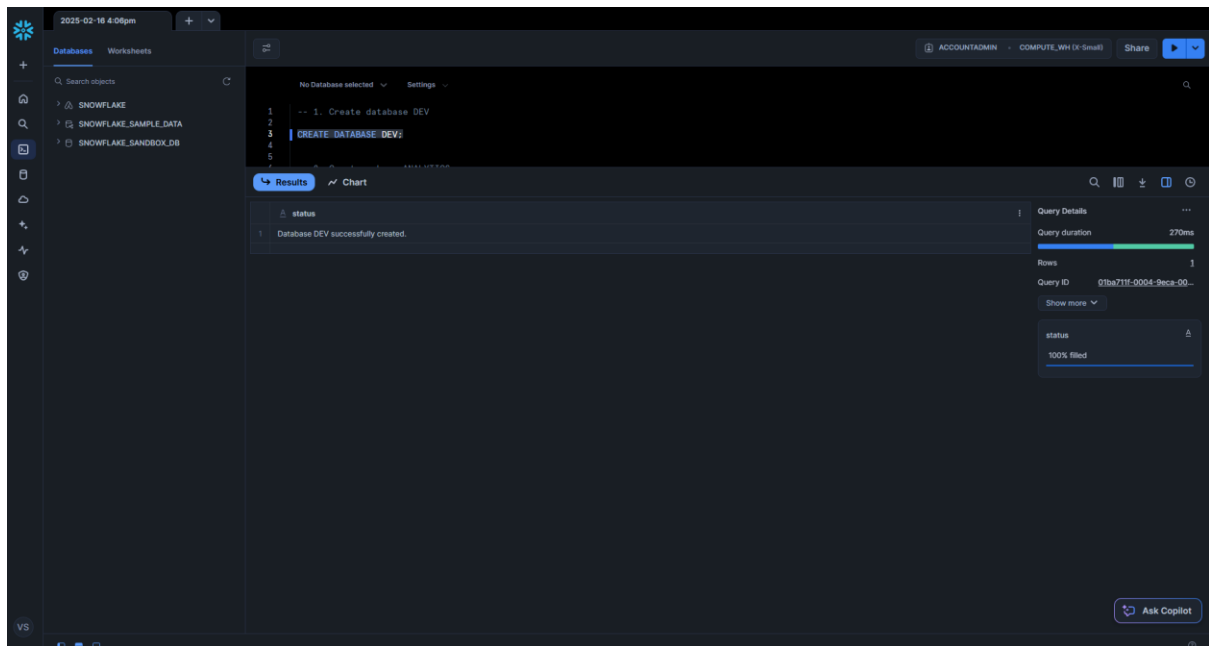
# DATA 226- DATAWAREHOUSE

## Homework 3

Name : Vimalanandhan Sivanandham

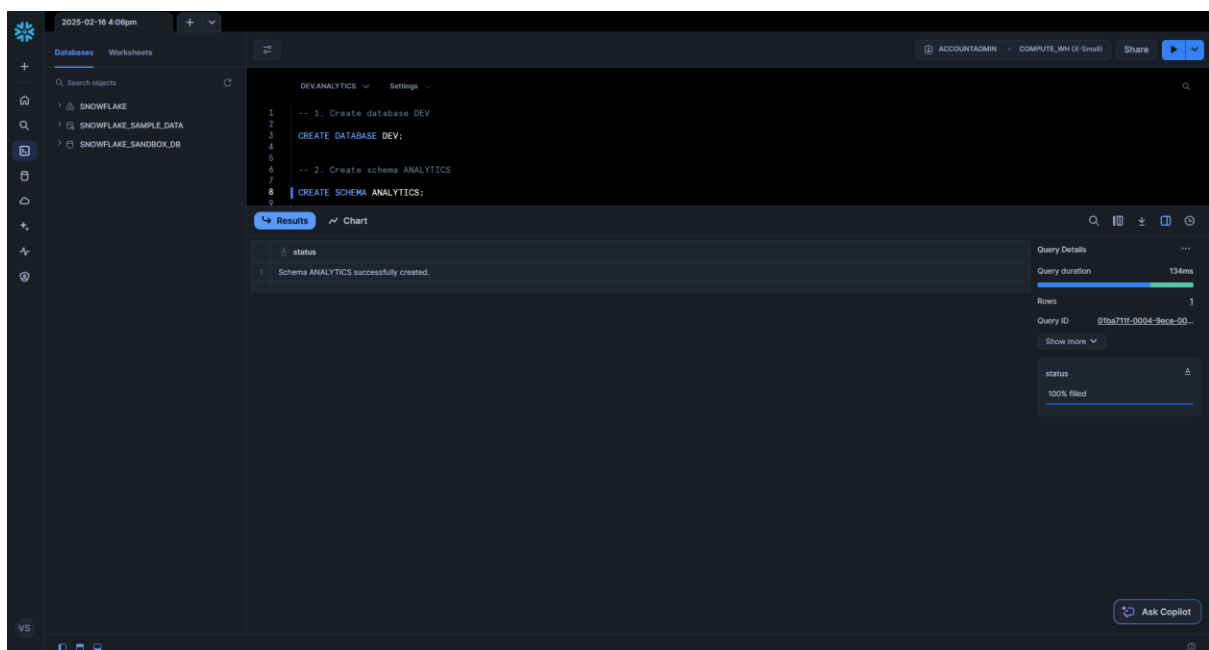
SJSU ID: 017596436

### 1. (+1) Create database DEV and schema ANALYTICS



-- 1. Create database DEV

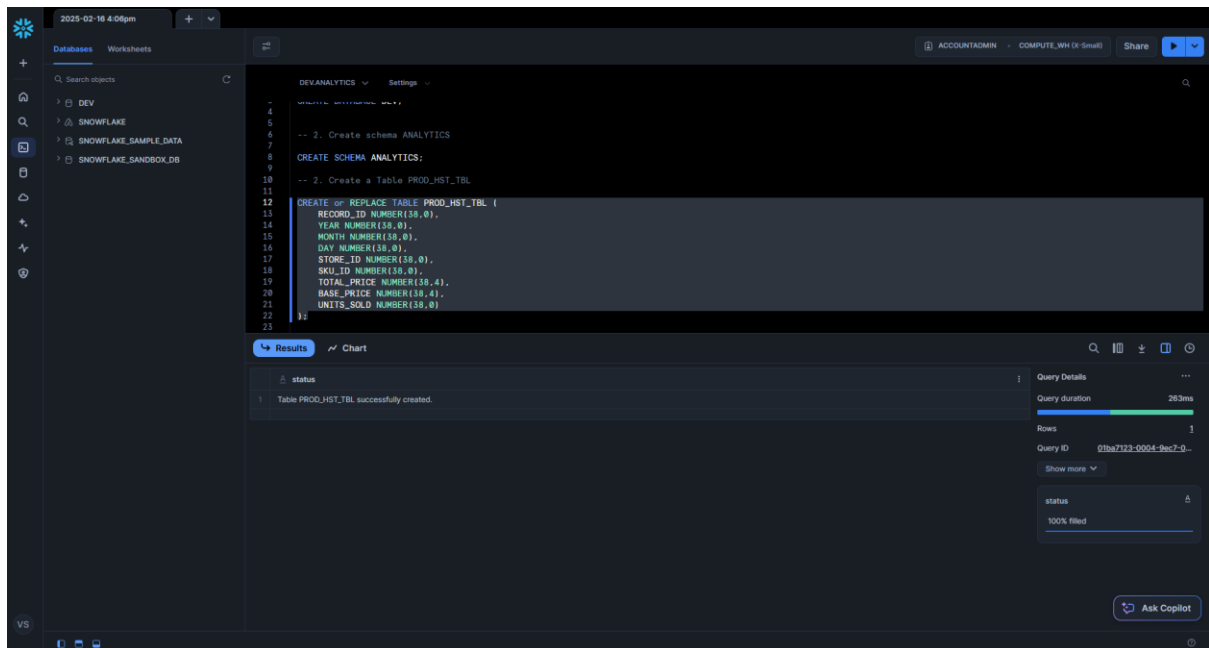
CREATE DATABASE DEV;



## -- 2. Create schema ANALYTICS

CREATE SCHEMA ANALYTICS;

## 2. (+2) Create a Table PROD\_HST\_TBL



## -- 2. Create a Table PROD\_HST\_TBL

CREATE or REPLACE TABLE PROD\_HST\_TBL (

RECORD\_ID NUMBER(38,0),

YEAR NUMBER(38,0),

MONTH NUMBER(38,0),

DAY NUMBER(38,0),

STORE\_ID NUMBER(38,0),

SKU\_ID NUMBER(38,0),

TOTAL\_PRICE NUMBER(38,4),

BASE\_PRICE NUMBER(38,4),

UNITS\_SOLD NUMBER(38,0)

);

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure: DEV > ANALYTICS > Tables > PROD\_HST\_TBL. The main editor contains the following SQL code:

```

1  -- 1. Create database DEV
2  CREATE DATABASE DEV;
3
4  -- 2. Create schema ANALYTICS
5  CREATE SCHEMA ANALYTICS;
6
7  -- 2. Create a Table PROD_HST_TBL
8  CREATE or REPLACE TABLE PROD_HST_TBL (
9      RECORD_ID NUMBER(38,0),
10     YEAR NUMBER(38,0),
11     MONTH NUMBER(38,0),
12     DAY NUMBER(38,0),
13     STORE_ID NUMBER(38,0),
14     SKU_ID NUMBER(38,0),
15     TOTAL_PRICE NUMBER(38,4),
16     BASE_PRICE NUMBER(38,4),
17     UNITS_SOLD NUMBER(38,0)
18 );
19
20 SELECT * FROM PROD_HST_TBL;

```

Below the editor, the 'Results' tab shows a preview of the data. The table has 10 columns: RECORD\_ID, YEAR, MONTH, DAY, STORE\_ID, SKU\_ID, TOTAL\_PRICE, BASE\_PRICE, and UNITS\_SOLD. The data is sorted by RECORD\_ID. The 'Query Details' panel on the right shows a query duration of 825ms and 15.1K rows.

**SELECT \* FROM PROD\_HST\_TBL;**

**3. (+3) Create a view to forecast SKU '219029' of STORE '9490'**

The screenshot shows the Snowflake Data Preview interface for the table DEV / ANALYTICS / PROD\_HST\_TBL. The table has 10 columns: RECORD\_ID, YEAR, MONTH, DAY, STORE\_ID, SKU\_ID, TOTAL\_PRICE, BASE\_PRICE, and UNITS\_SOLD. The data is sorted by RECORD\_ID. The 'Table Details' panel on the right shows the table has 15.1K rows and 132.0KB of data. The 'Query Details' panel on the right shows a query duration of 825ms and 15.1K rows.

RECORD_ID	YEAR	MONTH	DAY	STORE_ID	SKU_ID	TOTAL_PRICE	BASE_PRICE	UNITS_SOLD	
1	40955	2023	7	11	8063	219009	222.3000	222.3000	68
2	51074	2023	8	22	8562	219009	187.3875	187.3875	87
3	140379	2024	9	4	9845	219029	314.9250	314.9250	97
4	29745	2023	5	23	8422	219009	203.0625	203.0625	59
5	59816	2023	9	26	9490	219009	219.4500	219.4500	42
6	21230	2023	4	11	9984	219009	223.7250	223.7250	99
7	161025	2024	12	4	9279	219029	295.6875	295.6875	13
8	23680	2023	4	25	9371	219029	327.0375	327.0375	17
9	114566	2024	5	22	8222	219029	319.9125	319.9125	71
10	13469	2023	3	14	8869	219009	194.5125	225.8625	66
11	30763	2023	5	23	9823	219029	319.2000	319.2000	120
12	157087	2024	11	20	8063	219009	183.8250	183.8250	57
13	101186	2024	3	20	9876	219029	295.6875	295.6875	8
14	56919	2023	9	12	9837	219029	310.6500	310.6500	13
15	41416	2023	7	11	9112	219009	224.4375	224.4375	143
16	22019	2023	4	18	9328	219029	312.7875	312.7875	15
17	140403	2024	9	4	9872	219009	138.2250	175.2750	239
18	141027	2024	9	11	8869	219029	320.6250	320.6250	39
19	111671	2024	5	8	9112	219009	235.1250	235.1250	113
20	131840	2024	7	31	9578	219009	225.8625	225.8625	49
21	140182	2024	9	4	9713	219029	325.6125	325.6125	26
22	17825	2023	3	28	9840	219009	214.4625	214.4625	53
23	31744	2023	5	30	9250	219009	203.7750	203.7750	91
24	8152	2023	2	14	9961	219009	117.5625	227.2875	1197
25	116721	2024	5	29	9221	219029	312.7875	312.7875	20
26	18651	2023	4	4	9250	219009	218.0250	218.0250	67

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database structure with 'DEV' selected. The main editor contains the following SQL code:

```

21  UNITS_SOLD NUMBER(38,0)
22  );
23
24  SELECT * FROM PROD_HST_TBL;
25
26
27  CREATE OR REPLACE VIEW BOOKS_ST_VW
28  AS
29  WITH feature_engineering AS (
30    SELECT
31      TO_TIMESTAMP_NTZ(TO_DATE(YEAR || '-' || MONTH || '-' || DAY)) AS SALE_TS,
32      UNITS_SOLD
33    FROM PROD_HST_TBL
34    WHERE SKU_ID = 219029 AND STORE_ID = 9490
35  )
36
37  SELECT SALE_TS, UNITS_SOLD FROM feature_engineering
38  GROUP BY SALE_TS, UNITS_SOLD
39  ORDER BY SALE_TS;
40

```

The 'Results' tab shows a single message: 'View BOOKS\_ST\_VW successfully created.' The 'Query Details' panel on the right indicates a query duration of 262ms and 1 row returned.

The screenshot shows the Snowflake SQL Editor with the same SQL code as above. The 'Results' tab now displays the data from the view:

SALE_TS	UNITS_SOLD
2023-01-17 00:00:00.000	9
2023-01-24 00:00:00.000	4
2023-01-31 00:00:00.000	7
2023-02-07 00:00:00.000	16
2023-02-14 00:00:00.000	107
2023-02-21 00:00:00.000	66
2023-02-28 00:00:00.000	12
2023-03-07 00:00:00.000	23
2023-03-14 00:00:00.000	23
2023-03-21 00:00:00.000	18
2023-03-28 00:00:00.000	16
2023-04-04 00:00:00.000	22
2023-04-11 00:00:00.000	25
2023-04-18 00:00:00.000	44
2023-04-25 00:00:00.000	21

The 'Query Details' panel on the right shows a query duration of 597ms and 109 rows returned. A small bar chart for 'UNITS\_SOLD' is also visible.

-- 3. Create a view to forecast SKU '219029' of STORE '9490'

CREATE OR REPLACE VIEW BOOKS\_ST\_VW

AS

WITH feature\_engineering AS (

SELECT

TO\_TIMESTAMP\_NTZ(TO\_DATE(YEAR || '-' || MONTH || '-' || DAY)) AS SALE\_TS,

UNITS\_SOLD

FROM PROD\_HST\_TBL

WHERE SKU\_ID = 219029 AND STORE\_ID = 9490

)

SELECT SALE\_TS, UNITS\_SOLD FROM feature\_engineering

GROUP BY SALE\_TS, UNITS\_SOLD

ORDER BY SALE\_TS;

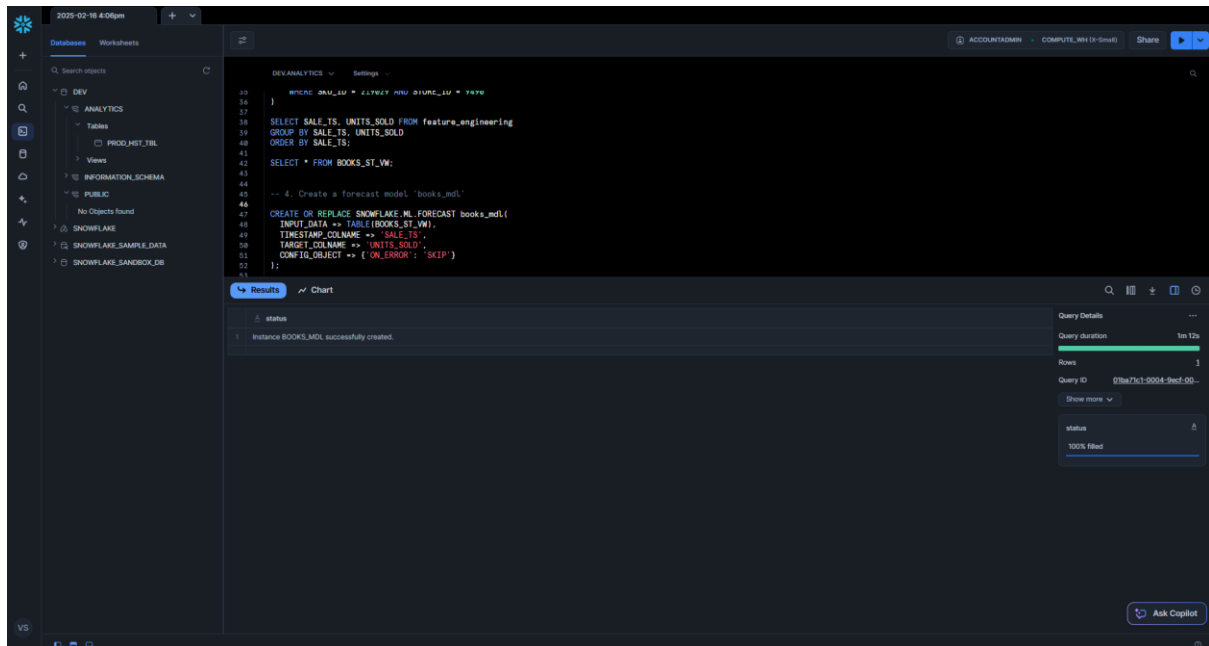
The screenshot displays the Snowflake SQL IDE interface. The top bar shows the date and time (2025-02-16 4:06pm) and the user (ACCOUNTADMIN). The left sidebar contains a navigation pane with a search bar and a tree view showing the database structure: DEV, ANALYTICS, Tables (PROD\_HST\_TBL), INFORMATION\_SCHEMA, Views, PUBLIC, No Objects found, SNOWFLAKE, SNOWFLAKE\_SAMPLE\_DATA, and SNOWFLAKE\_SANDBOX\_DB. The main editor area shows a SQL query with line numbers 20 to 42. The query defines a view BOOKS\_ST\_VW, creates a CTE feature\_engineering, and then selects data from it. The bottom section shows the query results in a table with two columns: SALE\_TS and UNITS\_SOLD. The results are sorted by SALE\_TS. On the right, a 'Query Details' panel shows the query duration (726ms), the number of rows (109), and the query ID (01ba7180-0004-8ec0-0-...). There are also visualizations for SALE\_TS and UNITS\_SOLD, and an 'Ask Copilot' button.

```
20  BASE_PRICE NUMBER(38,4),
21  UNITS_SOLD NUMBER(38,0)
22  );
23
24  SELECT * FROM PROD_HST_TBL;
25
26
27  CREATE OR REPLACE VIEW BOOKS_ST_VW
28  AS
29  WITH feature_engineering AS (
30  SELECT
31    TO_TIMESTAMP_NTZ(TO_DATE(YEAR || '-' || MONTH || '-' || DAY)) AS SALE_TS,
32    UNITS_SOLD
33  FROM PROD_HST_TBL
34  WHERE SKU_ID = 219029 AND STORE_ID = 9490
35  )
36
37  SELECT SALE_TS, UNITS_SOLD FROM feature_engineering
38  GROUP BY SALE_TS, UNITS_SOLD
39  ORDER BY SALE_TS;
40
41  SELECT * FROM BOOKS_ST_VW;
42
```

	SALE_TS	UNITS_SOLD
1	2023-01-17 00:00:00.000	
2	2023-01-24 00:00:00.000	
3	2023-01-31 00:00:00.000	
4	2023-02-07 00:00:00.000	
5	2023-02-14 00:00:00.000	
6	2023-02-21 00:00:00.000	
7	2023-02-28 00:00:00.000	
8	2023-03-07 00:00:00.000	
9	2023-03-14 00:00:00.000	
10	2023-03-21 00:00:00.000	
11	2023-03-28 00:00:00.000	

SELECT \* FROM BOOKS\_ST\_VW;

#### 4. (+3) Create a forecast model 'books\_mdl'



-- 4. Create a forecast model 'books\_mdl'

CREATE OR REPLACE SNOWFLAKE.ML.FORECAST books\_mdl(

INPUT\_DATA => TABLE(BOOKS\_ST\_VW),

TIMESTAMP\_COLNAME => 'SALE\_TS',

TARGET\_COLNAME => 'UNITS\_SOLD',

CONFIG\_OBJECT => {'ON\_ERROR': 'SKIP'}

);

## 5. (+1) Display the Results to predict next 4 weeks.

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database schema with the following structure:

- DEV
  - ANALYTICS
    - Tables
      - PROD\_HST\_TBL
    - Views
      - BOOKS\_ST\_VW
  - INFORMATION\_SCHEMA
  - PUBLIC
  - SNOWFLAKE
    - SNOWFLAKE\_SAMPLE\_DATA
    - SNOWFLAKE\_SANDBOX\_DB

The main editor contains the following SQL code:

```
14 -- 1. Create a view of units sold
15 FROM PROD_HST_TBL
16 WHERE SKU_ID = 219029 AND STORE_ID = 9498
17
18
19 SELECT SALE_TS, UNITS_SOLD FROM feature_engineering
20 GROUP BY SALE_TS, UNITS_SOLD
21 ORDER BY SALE_TS;
22
23 -- 2. Create a forecast model 'books_mdl'
24
25 -- 3. Create a forecast model 'books_mdl'
26 CREATE OR REPLACE SNOWFLAKE.ML.FORECAST books_mdl(
27   INPUT_DATA => TABLE(BOOKS_ST_VW),
28   TIMESTAMP_COLUMN => 'SALE_TS',
29   TARGET_COLUMN => 'UNITS_SOLD',
30   CONFIG_OBJECT => {'ON_ERROR': 'SKIP'}
31 );
32
33 -- 4. Display the results to predict the next 4 weeks
34
35 CALL test_model!FORECAST (FORECASTING_PERIODS => 4);
```

The Results tab shows the following data:

SERIES	TS	FORECAST	LOWER_BOUND	UPPER_BOUND
1 null	2025-02-18 00:00:00.000	20.13104669	-3.076256761	43.34134961
2 null	2025-02-25 00:00:00.000	20.111590442	-6.546268479	46.769467889
3 null	2025-03-04 00:00:00.000	18.102697567	-10.442921676	48.84831819
4 null	2025-03-11 00:00:00.000	18.185680643	-13.267078132	51.630450548

Query Details: Query duration 19s, Rows 4, Query ID 01ba71a-0004-Sec2-9...

-- 5. Display the results to predict the next 4 weeks

CALL test\_model!FORECAST (FORECASTING\_PERIODS => 4);

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays the database schema with the following structure:

- DEV
  - ANALYTICS
    - Tables
      - PROD\_HST\_TBL
    - Views
      - BOOKS\_ST\_VW
  - INFORMATION\_SCHEMA
  - PUBLIC
  - SNOWFLAKE
    - SNOWFLAKE\_SAMPLE\_DATA
    - SNOWFLAKE\_SANDBOX\_DB

The main editor contains the following SQL code:

```
14 -- 1. Create a view of units sold
15 FROM PROD_HST_TBL
16 WHERE SKU_ID = 219029 AND STORE_ID = 9498
17
18
19 SELECT SALE_TS, UNITS_SOLD FROM feature_engineering
20 GROUP BY SALE_TS, UNITS_SOLD
21 ORDER BY SALE_TS;
22
23 -- 2. Create a forecast model 'books_mdl'
24
25 -- 3. Create a forecast model 'books_mdl'
26 CREATE OR REPLACE SNOWFLAKE.ML.FORECAST books_mdl(
27   INPUT_DATA => TABLE(BOOKS_ST_VW),
28   TIMESTAMP_COLUMN => 'SALE_TS',
29   TARGET_COLUMN => 'UNITS_SOLD',
30   CONFIG_OBJECT => {'ON_ERROR': 'SKIP'}
31 );
32
33 -- 4. Display the results to predict the next 4 weeks
34
35 CALL test_model!FORECAST (FORECASTING_PERIODS => 4);
36
37 SELECT
38   TS,
39   ROUND(FORECAST) AS UNITS_SOLD_ROUNDED
40 FROM TABLE(test_model!FORECAST (FORECASTING_PERIODS => 4));
```

The Results tab shows the following data:

TS	UNITS_SOLD_ROUNDED
1 2025-02-18 00:00:00.000	20
2 2025-02-25 00:00:00.000	20
3 2025-03-04 00:00:00.000	19
4 2025-03-11 00:00:00.000	19

Query Details: Query duration 21s, Rows 4, Query ID 01ba71a-0004-Sec2-9...

```
SELECT  
  
    TS,  
  
    ROUND(FORECAST) AS UNITS_SOLD_ROUNDED  
  
FROM TABLE(test_model!!FORECAST(FORECASTING_PERIODS => 4));
```

#### **6. (+3) Explain your understanding about the Forecasting Process.**

This assignment's forecasting procedure entails building a structured pipeline in Snowflake to project future sales for a certain SKU and retailer. In order to store historical sales data, the PROD\_HST\_TBL table must be created after the DEV database and ANALYTICS schema have been set up. Data is then explicitly filtered for SKU '219029' at shop '9490' using a view that has been developed. Building a forecasting model (books\_md1) with Snowflake ML is the main step in the forecasting process. This model uses historical sales data (BOOKS\_ST\_VW), finds trends, and forecasts future sales based on patterns in the SALE\_TS (timestamp) and UNITS\_SOLD (goal variable) columns. Presenting forecasts for the upcoming four weeks, which include information on anticipated sales, is the last phase.

In order to ensure real-time updates, the pipeline diagram illustrates how data is imported and converted through the RAW, CURATION, and ANALYTICS layers, and then saved in a dynamic table. The forecasts are more accurate and useful since this methodical methodology guarantees that only pertinent and clean data is used.

#### **(+1) For following the format**

Followed the process.