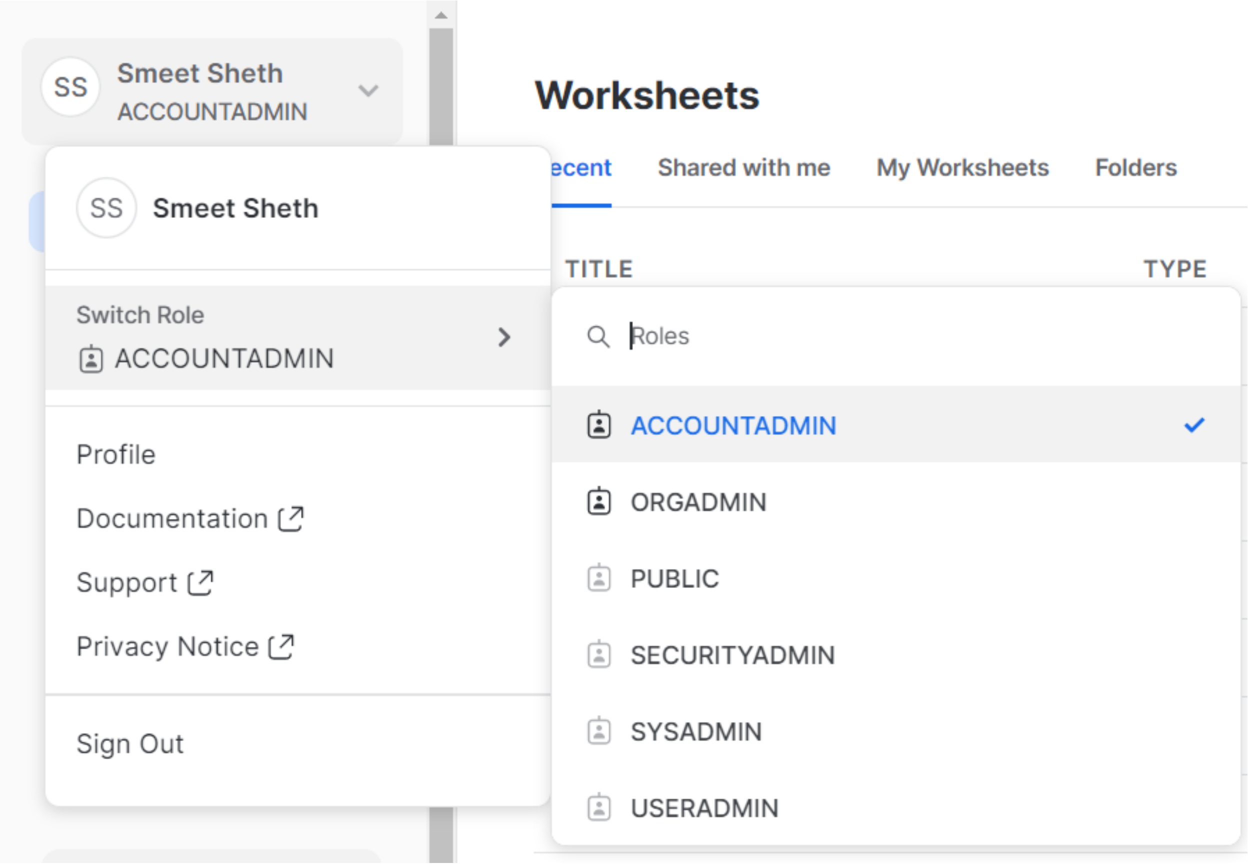
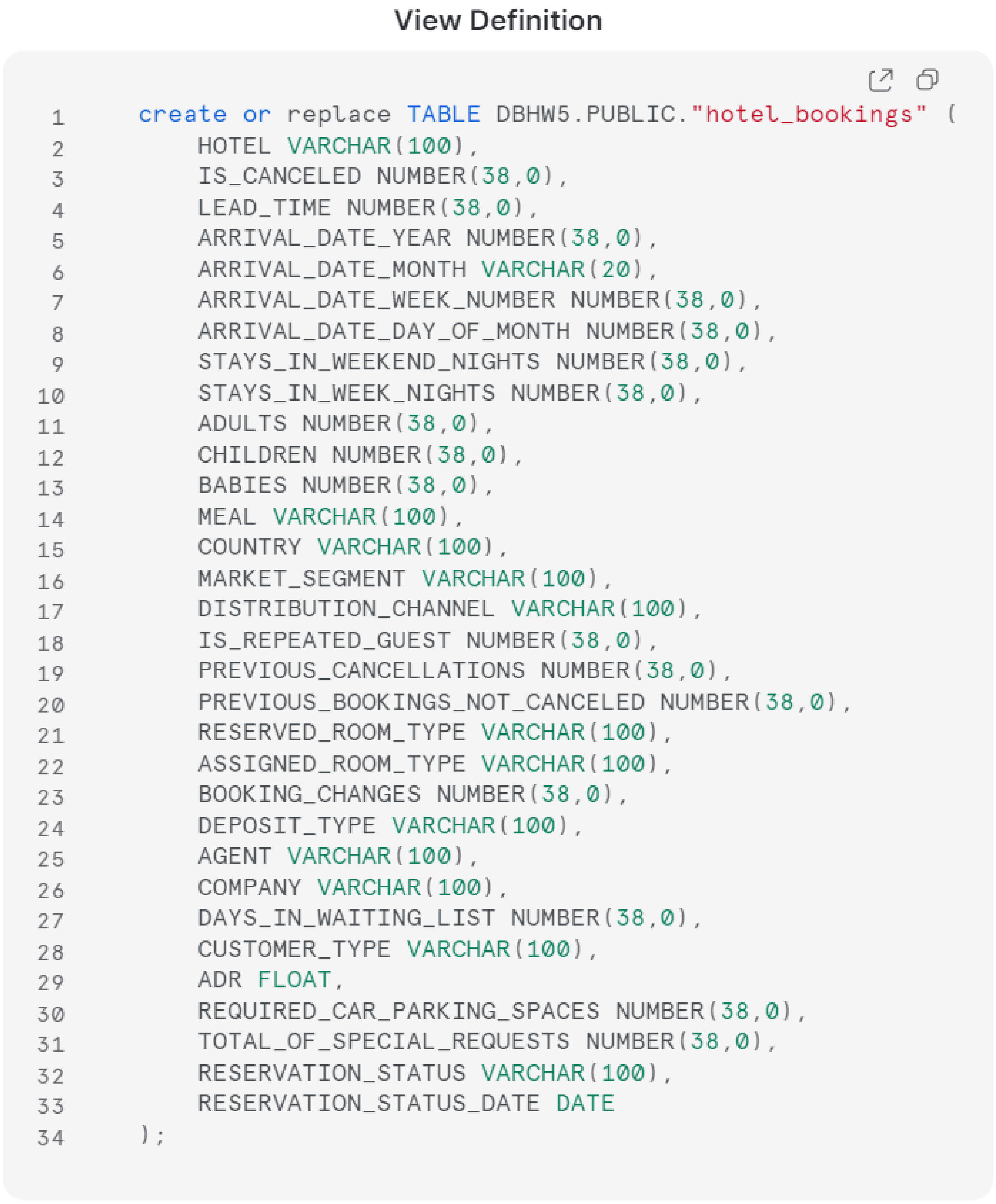
Smeet Sheth
Data 225 HW5
016786133

Getting started with Snowflake:



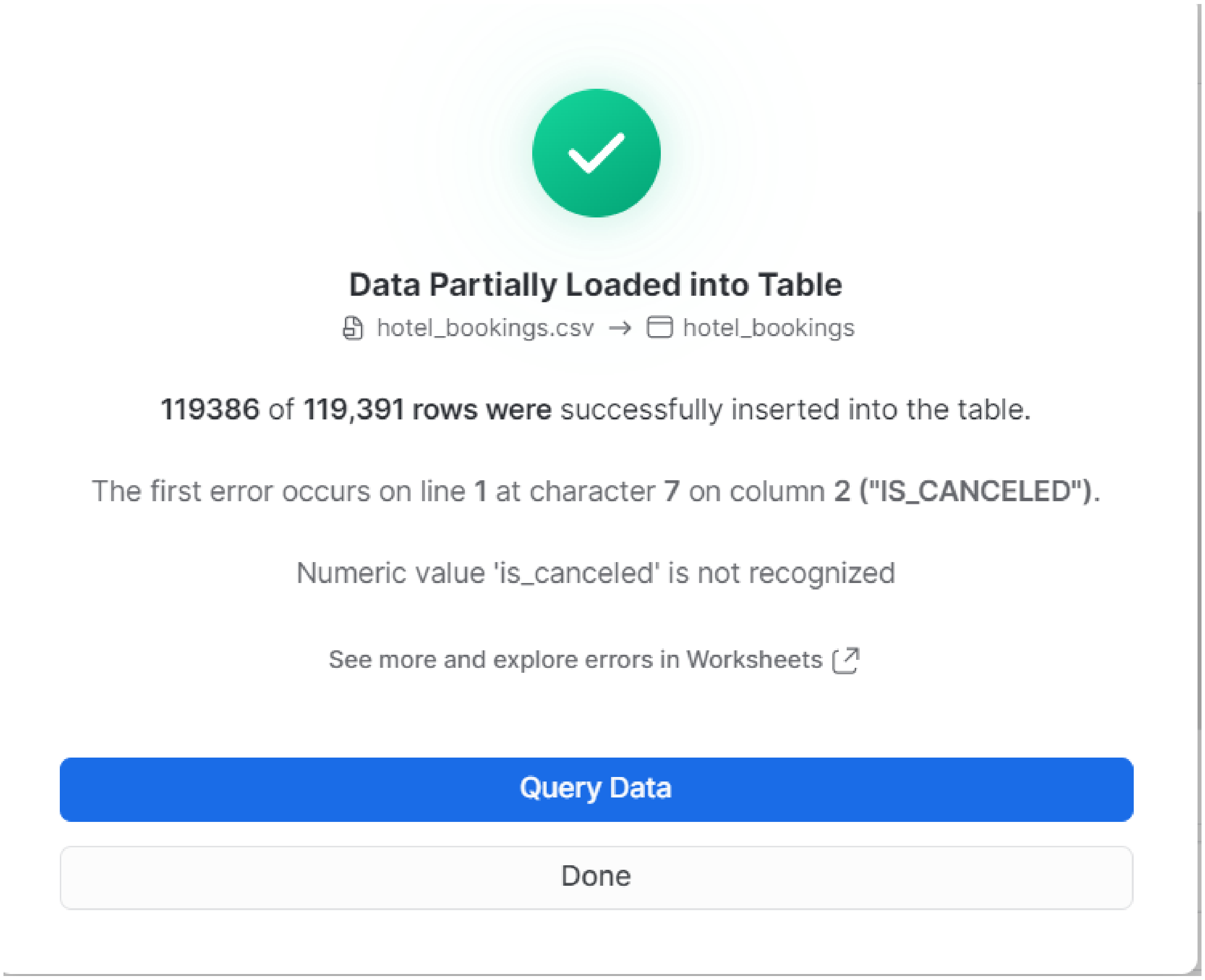
Query for creating the hotel_bookings table:



View Definition

```
create or replace TABLE DBHW5.PUBLIC."hotel_bookings" (
    HOTEL VARCHAR(100),
    IS_CANCELED NUMBER(38,0),
    LEAD_TIME NUMBER(38,0),
    ARRIVAL_DATE_YEAR NUMBER(38,0),
    ARRIVAL_DATE_MONTH VARCHAR(20),
    ARRIVAL_DATE_WEEK_NUMBER NUMBER(38,0),
    ARRIVAL_DATE_DAY_OF_MONTH NUMBER(38,0),
    STAYS_IN_WEEKEND_NIGHTS NUMBER(38,0),
    STAYS_IN_WEEK_NIGHTS NUMBER(38,0),
    ADULTS NUMBER(38,0),
    CHILDREN NUMBER(38,0),
    BABIES NUMBER(38,0),
    MEAL VARCHAR(100),
    COUNTRY VARCHAR(100),
    MARKET_SEGMENT VARCHAR(100),
    DISTRIBUTION_CHANNEL VARCHAR(100),
    IS_REPEATED_GUEST NUMBER(38,0),
    PREVIOUS_CANCELLATIONS NUMBER(38,0),
    PREVIOUS_BOOKINGS_NOT_CANCELED NUMBER(38,0),
    RESERVED_ROOM_TYPE VARCHAR(100),
    ASSIGNED_ROOM_TYPE VARCHAR(100),
    BOOKING_CHANGES NUMBER(38,0),
    DEPOSIT_TYPE VARCHAR(100),
    AGENT VARCHAR(100),
    COMPANY VARCHAR(100),
    DAYS_IN_WAITING_LIST NUMBER(38,0),
    CUSTOMER_TYPE VARCHAR(100),
    ADR FLOAT,
    REQUIRED_CAR_PARKING_SPACES NUMBER(38,0),
    TOTAL_OF_SPECIAL_REQUESTS NUMBER(38,0),
    RESERVATION_STATUS VARCHAR(100),
    RESERVATION_STATUS_DATE DATE
);
```

Imported the data into our table:



Data Partially Loaded into Table

hotel_bookings.csv → hotel_bookings

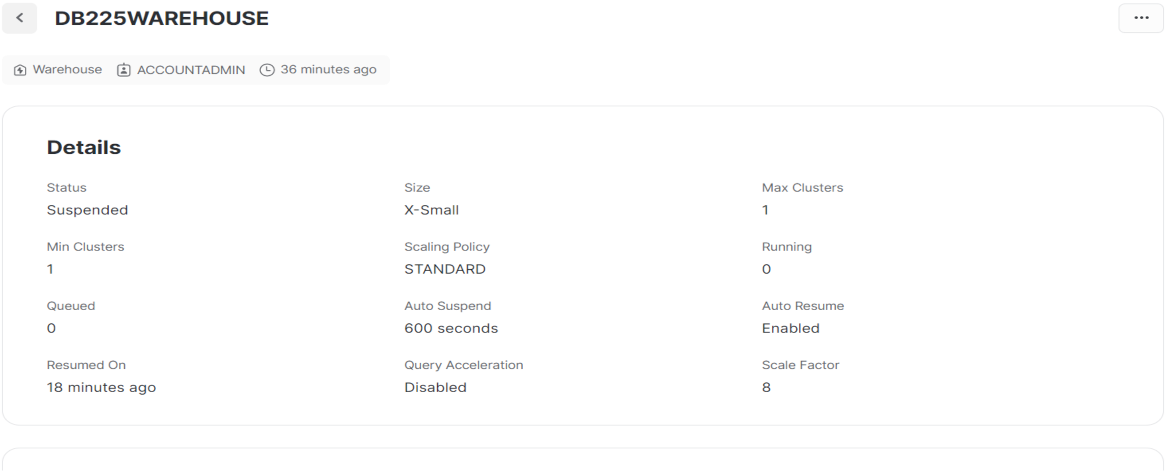
119386 of 119,391 rows were successfully inserted into the table.

The first error occurs on line 1 at character 7 on column 2 ("IS_CANCELED").

Numeric value 'is_canceled' is not recognized

See more and explore errors in Worksheets

Created a Warehouse for the project:

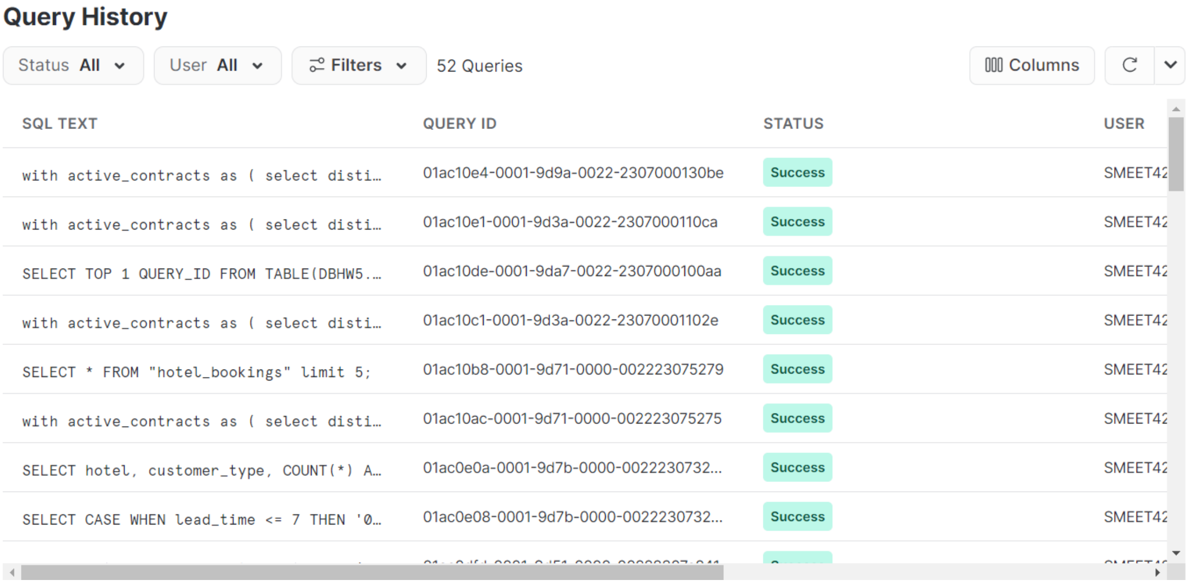


DB225WAREHOUSE

Warehouse ACCOUNTADMIN 36 minutes ago

Details

Status Suspended	Size X-Small	Max Clusters 1
Min Clusters 1	Scaling Policy STANDARD	Running 0
Queued 0	Auto Suspend 600 seconds	Auto Resume Enabled
Resumed On 18 minutes ago	Query Acceleration Disabled	Scale Factor 8

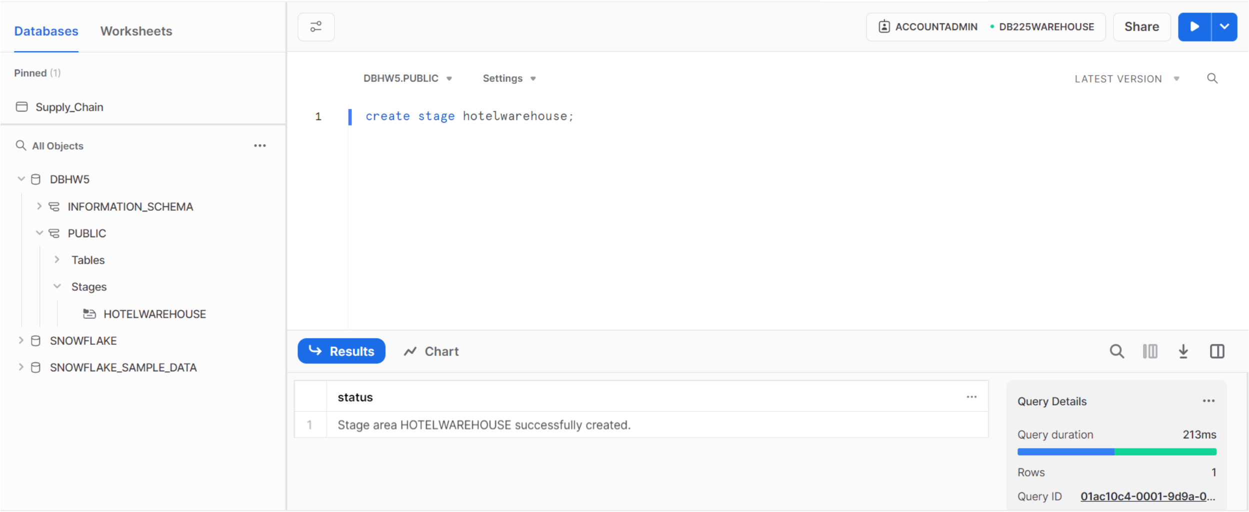


Query History

52 Queries

SQL TEXT	QUERY ID	STATUS	USER
with active_contracts as (select disti…	01ac10e4-0001-9d9a-0022-2307000130be	Success	SMEET42
with active_contracts as (select disti…	01ac10e1-0001-9d3a-0022-2307000110ca	Success	SMEET42
SELECT TOP 1 QUERY_ID FROM TABLE(DBHW5.…	01ac10de-0001-9da7-0022-2307000100aa	Success	SMEET42
with active_contracts as (select disti…	01ac10c1-0001-9d3a-0022-23070001102e	Success	SMEET42
SELECT * FROM "hotel_bookings" limit 5;	01ac10b8-0001-9d71-0000-002223075279	Success	SMEET42
with active_contracts as (select disti…	01ac10ac-0001-9d71-0000-002223075275	Success	SMEET42
SELECT hotel, customer_type, COUNT(*) A…	01ac0e0a-0001-9d7b-0000-0022230732…	Success	SMEET42
SELECT CASE WHEN lead_time <= 7 THEN '0…	01ac0e08-0001-9d7b-0000-0022230732…	Success	SMEET42

Created Internal stage to extract data from the database:



```
create stage hotelwarehouse;
```

	status
1	Stage area HOTELWAREHOUSE successfully created.

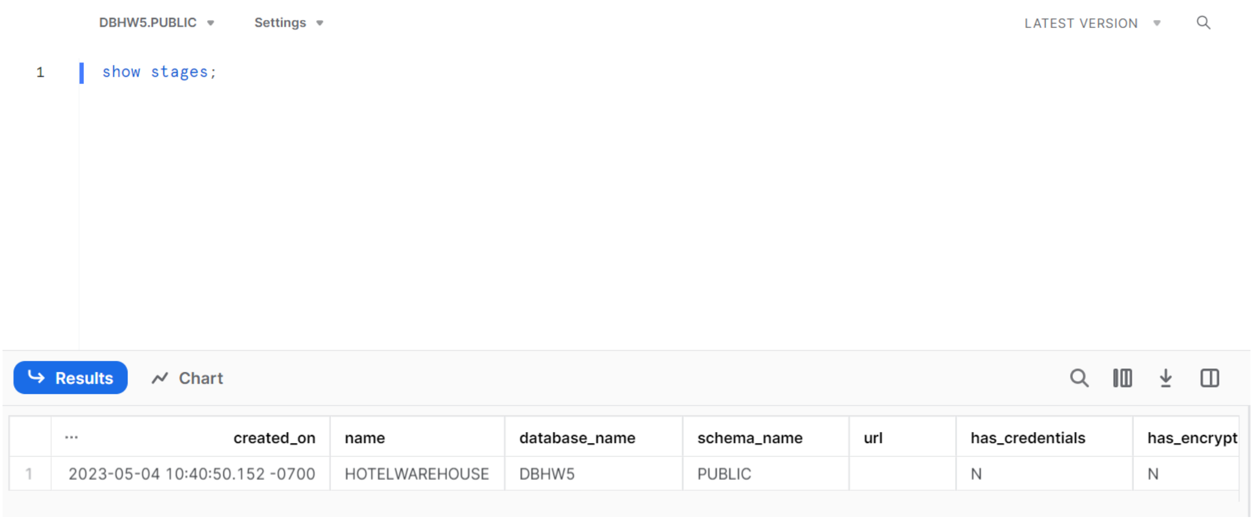
Query Details

Query duration 213ms

Rows 1

Query ID 01ac10c4-0001-9d9a-0…

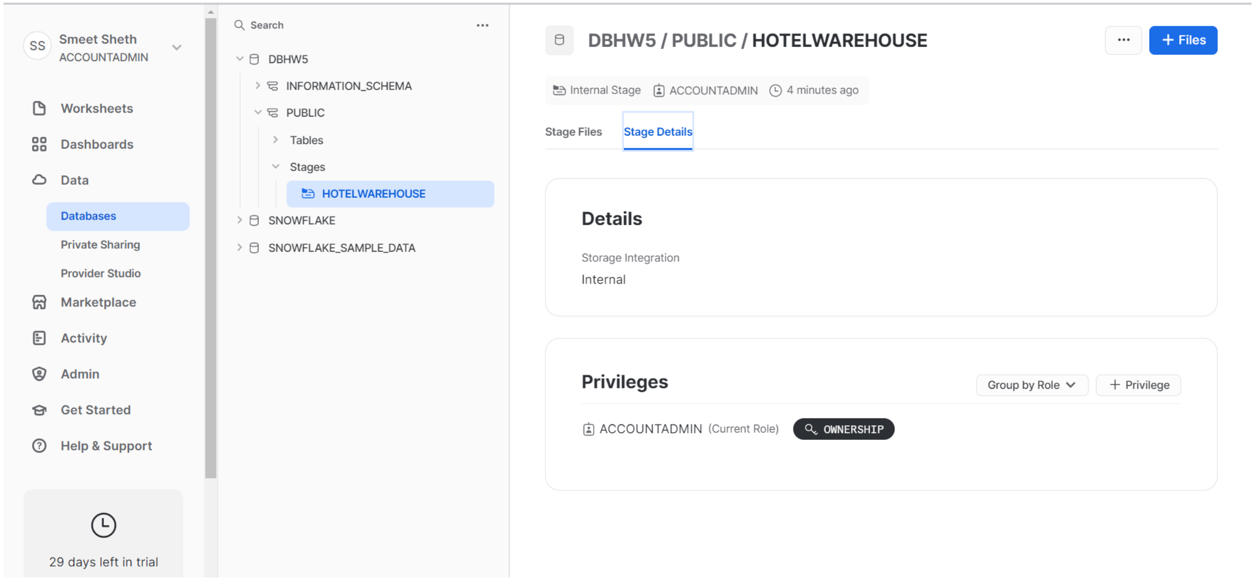
Displaying the stage we created and its details:



```
show stages;
```

	created_on	name	database_name	schema_name	url	has_credentials	has_encrypt
1	2023-05-04 10:40:50.152 -0700	HOTELWAREHOUSE	DBHW5	PUBLIC		N	N

Stage details:



DBHW5 / PUBLIC / HOTELWAREHOUSE

Internal Stage ACCOUNTADMIN 4 minutes ago

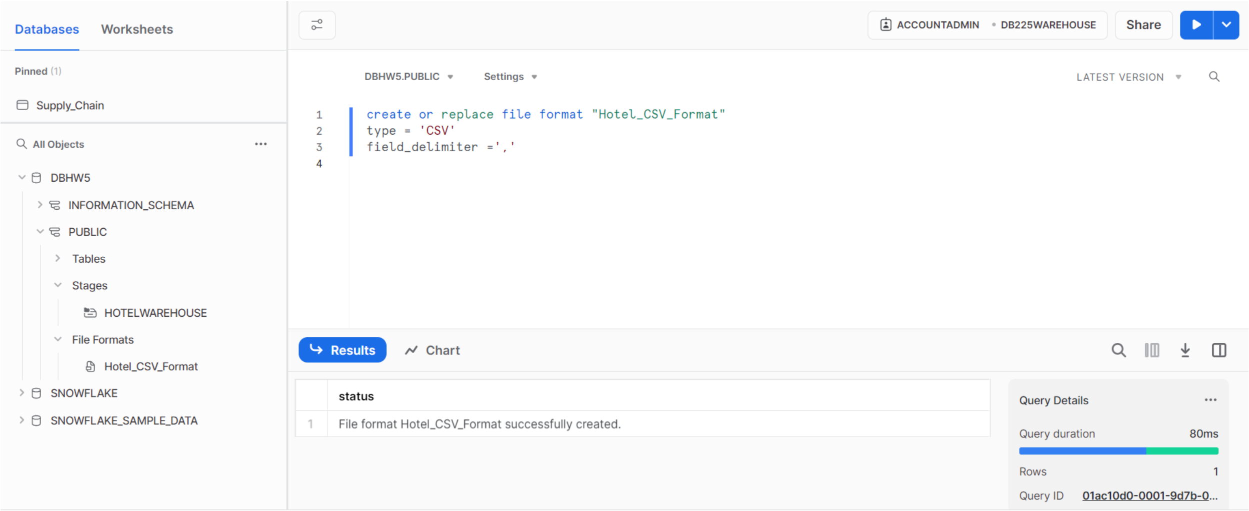
Details

Storage Integration
Internal

Privileges

ACCOUNTADMIN (Current Role) OWNERSHIP

Creating file format that will be uploaded into the warehouse:



```
create or replace file format "Hotel_CSV_Format"
type = 'CSV'
field_delimiter =','

```

	status
1	File format Hotel_CSV_Format successfully created.

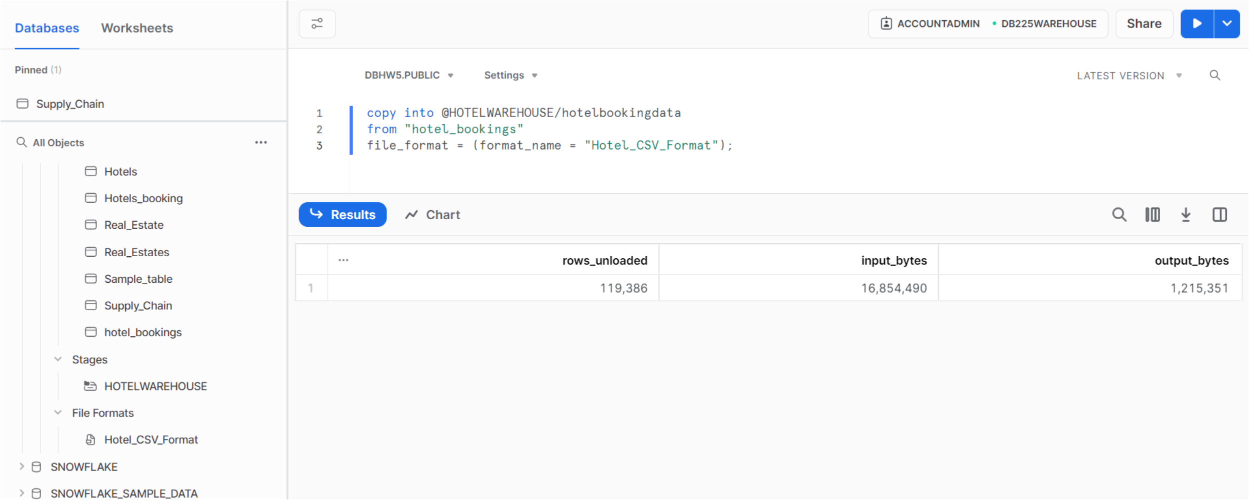
Query Details

Query duration 80ms

Rows 1

Query ID 01ac10d0-0001-9d7b-0...

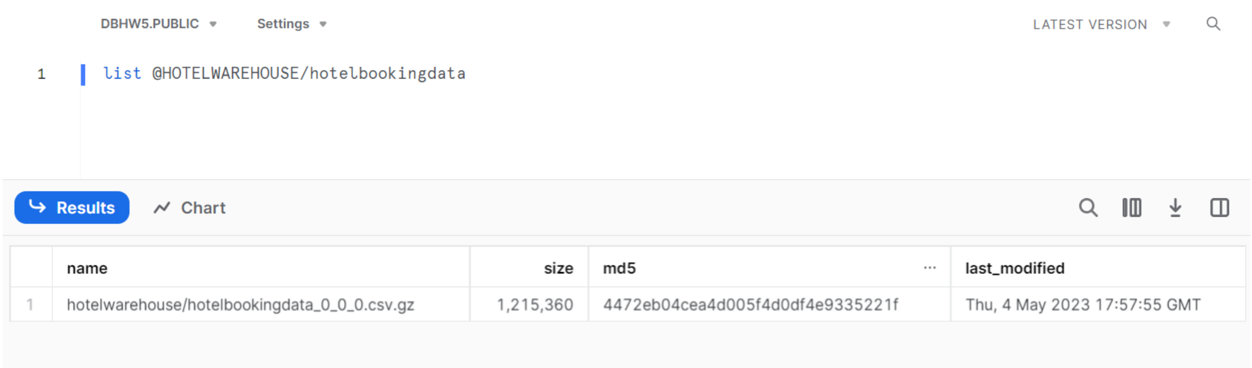
Creating table ‘hotelbookingdata’ inside stage:



```
copy into @HOTELWAREHOUSE/hotelbookingdata
from "hotel_bookings"
file_format = (format_name = "Hotel_CSV_Format");
```

...	rows_unloaded	input_bytes	output_bytes
1	119,386	16,854,490	1,215,351

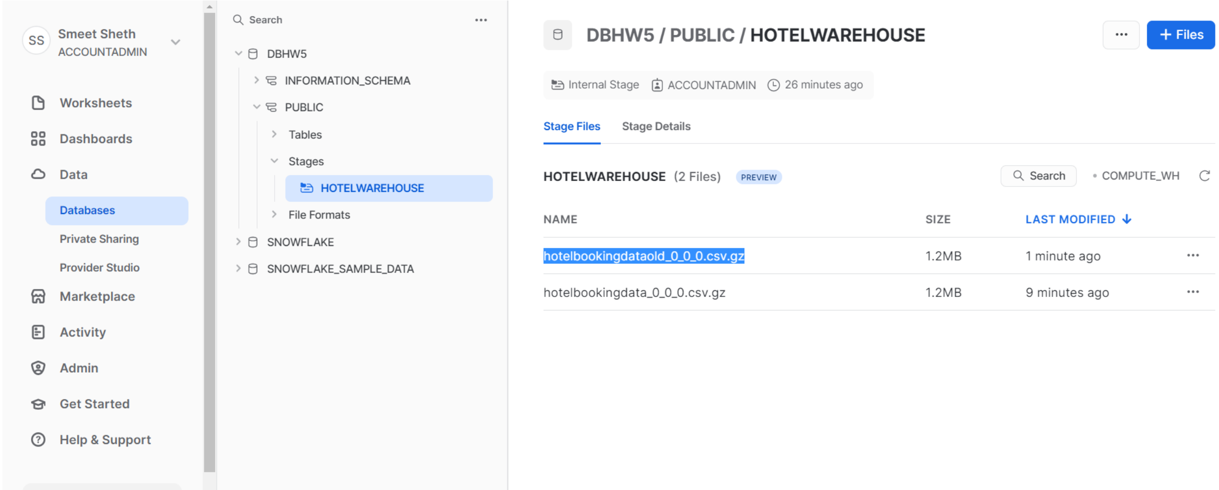
Table created successfully:



```
list @HOTELWAREHOUSE/hotelbookingdata
```

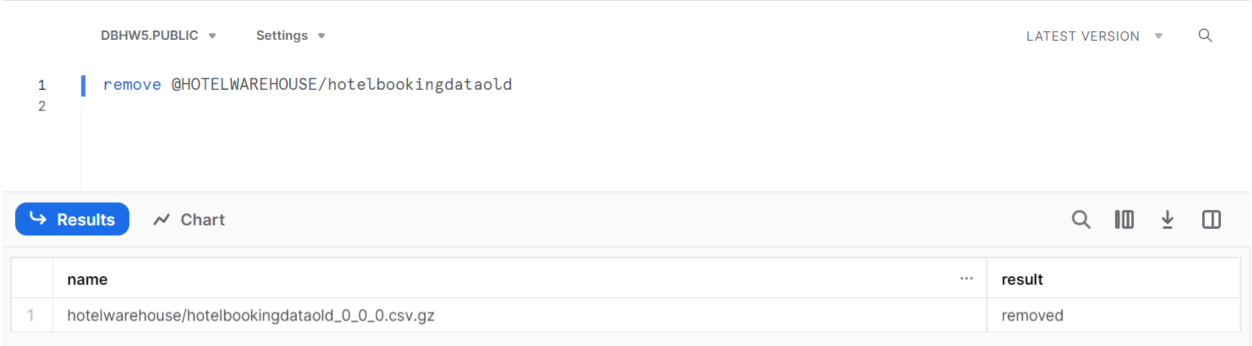
	name	size	md5	last_modified
1	hotelwarehouse/hotelbookingdata_0_0_0.csv.gz	1,215,360	4472eb04cea4d005f4d0df4e9335221f	Thu, 4 May 2023 17:57:55 GMT

New table created inside stage:



NAME	SIZE	LAST MODIFIED
hotelbookingdataold_0_0_0.csv.gz	1.2MB	1 minute ago
hotelbookingdata_0_0_0.csv.gz	1.2MB	9 minutes ago

Removing the new table:



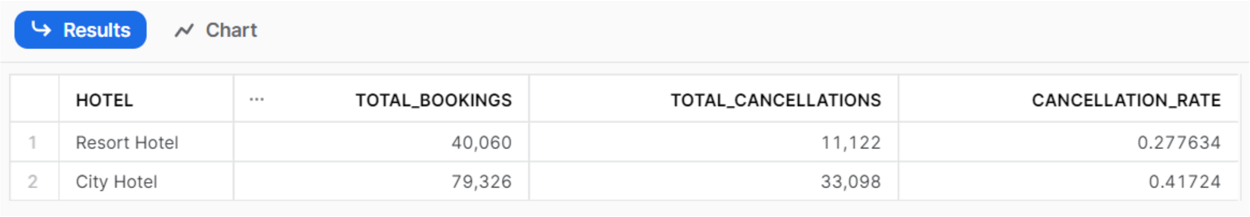
```
remove @HOTELWAREHOUSE/hotelbookingdataold
```

	name	result
1	hotelwarehouse/hotelbookingdataold_0_0_0.csv.gz	removed

Queries for analysis:

Query 1.

```
SELECT
  hotel,
  COUNT(*) AS total_bookings,
  SUM(is_canceled) AS total_cancellations,
  SUM(is_canceled) / COUNT(*) AS cancellation_rate
FROM
  "hotel_bookings"
GROUP BY
  hotel;
```



	HOTEL	TOTAL_BOOKINGS	TOTAL_CANCELLATIONS	CANCELLATION_RATE
1	Resort Hotel	40,060	11,122	0.277634
2	City Hotel	79,326	33,098	0.41724

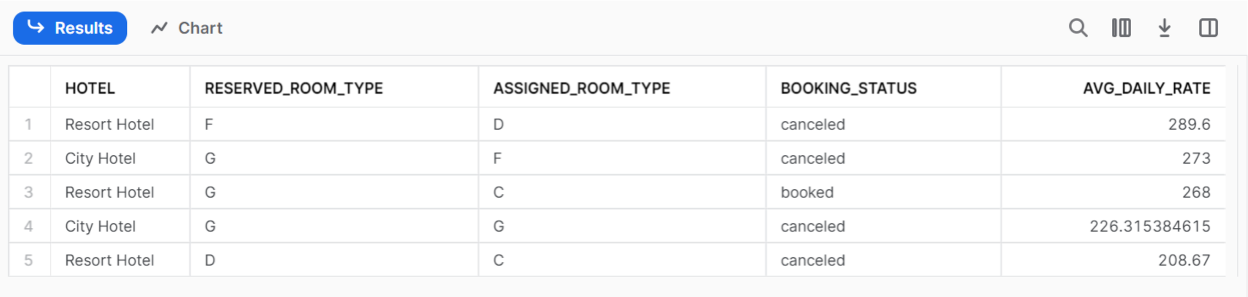
This query calculates the cancellation rate of two different categories of hotels.

Query 2.

```
SELECT
  sub.hotel,
  sub.reserved_room_type,
  sub.assigned_room_type,
  sub.booking_status,
  sub.avg_daily_rate
FROM
(
  SELECT
    hotel,
    reserved_room_type,
    assigned_room_type,
    CASE
      WHEN is_canceled = 0 THEN 'booked'
      ELSE 'canceled'
    END AS booking_status,
    AVG(adr) AS avg_daily_rate
```

```
 FROM
  "hotel_bookings"
 GROUP BY
  hotel,
  reserved_room_type,
  assigned_room_type,
  booking_status
) sub
WHERE
 sub.hotel IN (
  SELECT
   hotel
  FROM
   "hotel_bookings"
  WHERE
   is_canceled = 0
  GROUP BY
   hotel
  ORDER BY
   AVG(adr) DESC
  LIMIT 10
 )
ORDER BY
 sub.avg_daily_rate DESC

limit 5;
```

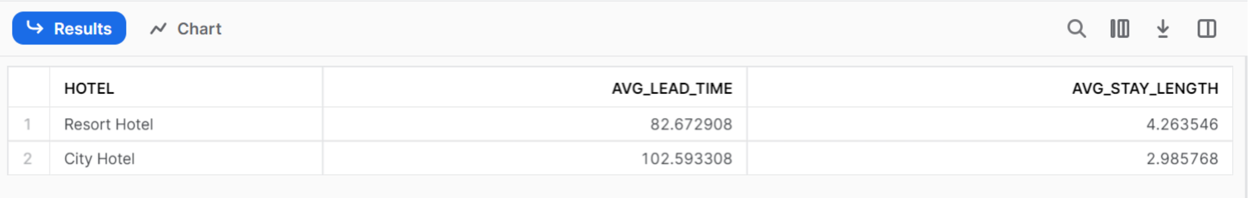


	HOTEL	RESERVED_ROOM_TYPE	ASSIGNED_ROOM_TYPE	BOOKING_STATUS	AVG_DAILY_RATE
1	Resort Hotel	F	D	canceled	289.6
2	City Hotel	G	F	canceled	273
3	Resort Hotel	G	C	booked	268
4	City Hotel	G	G	canceled	226.315384615
5	Resort Hotel	D	C	canceled	208.67

The above query calculates the average daily rate for all the hotels and displays the top 5 hotels with highest ADR along with their booking status. We can use the query to understand the functioning of these hotels and apply them to the hotels with comparatively low ADR

Query 3.

```
SELECT
  h.hotel,
  AVG(h.lead_time) AS avg_lead_time,
  AVG(h.stays_in_week_nights + h.stays_in_weekend_nights) AS avg_stay_length
FROM
  "hotel_bookings" h
  INNER JOIN (
    SELECT
      hotel,
      arrival_date_month,
      AVG(adr) AS avg_monthly_adr
    FROM
      "hotel_bookings"
    WHERE
      is_canceled = 0
    GROUP BY
      hotel,
      arrival_date_month
  ) m ON h.hotel = m.hotel AND h.arrival_date_month = m.arrival_date_month
WHERE
  h.is_canceled = 0
  AND h.assigned_room_type = 'A'
  AND h.arrival_date_year = 2017
GROUP BY
  h.hotel;
```

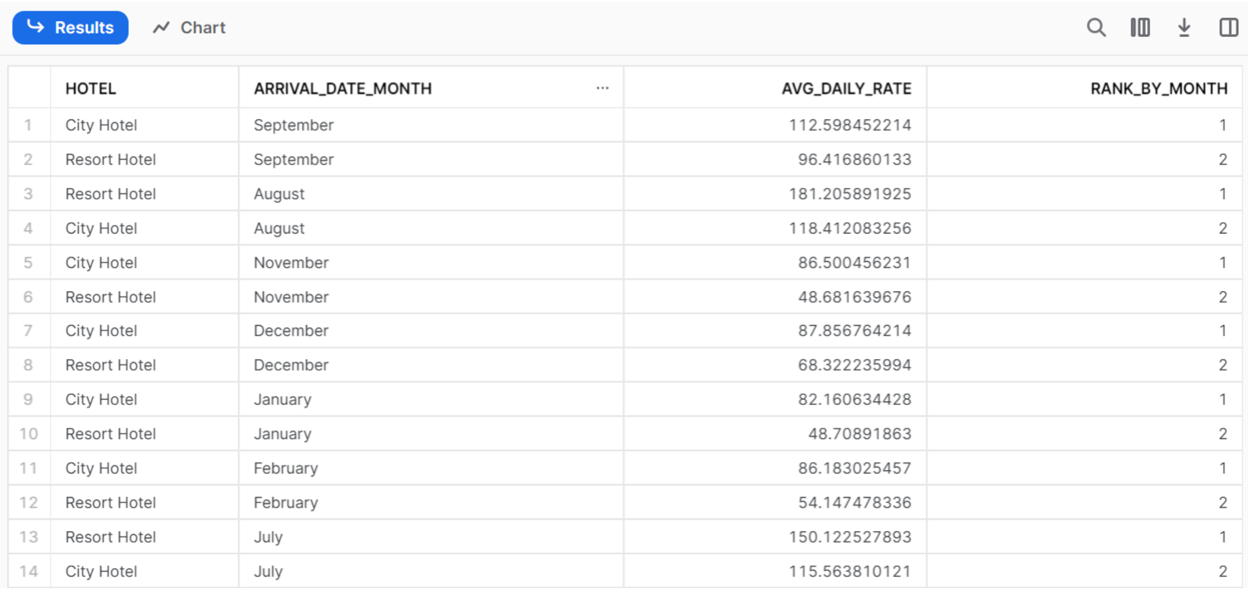


	HOTEL	AVG_LEAD_TIME	AVG_STAY_LENGTH
1	Resort Hotel	82.672908	4.263546
2	City Hotel	102.593308	2.985768

The above query calculates the Average stay of the customers for both the type of hotels in the year 2017 with room type ‘A’. This can help understand the customer pattern for that particular hotel and can be analyzed to provide services accordingly.

Query 4.

```
SELECT
  hotel,
  arrival_date_month,
  AVG(adr) AS avg_daily_rate,
    RANK() OVER (PARTITION BY arrival_date_month ORDER BY AVG(adr) DESC) AS rank_by_month
FROM
  "hotel_bookings"
WHERE
  is_canceled = 0
GROUP BY
  hotel,
  arrival_date_month;
```

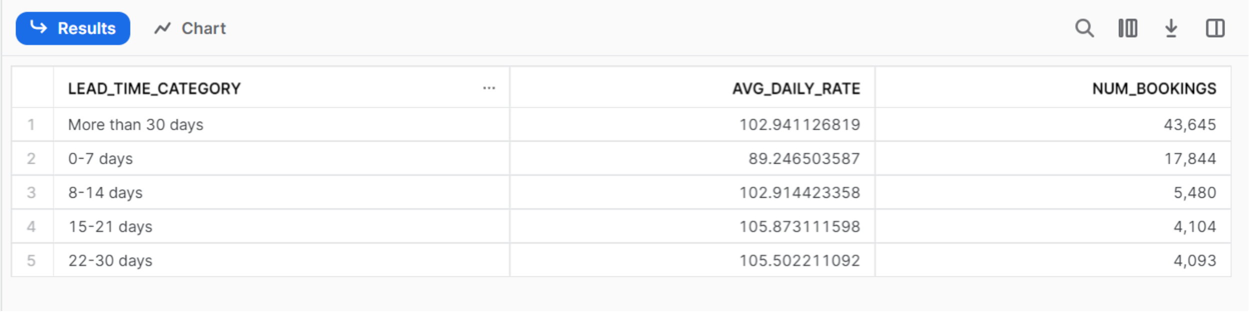


	HOTEL	ARRIVAL_DATE_MONTH	AVG_DAILY_RATE	RANK_BY_MONTH
1	City Hotel	September	112.598452214	1
2	Resort Hotel	September	96.416860133	2
3	Resort Hotel	August	181.205891925	1
4	City Hotel	August	118.412083256	2
5	City Hotel	November	86.500456231	1
6	Resort Hotel	November	48.681639676	2
7	City Hotel	December	87.856764214	1
8	Resort Hotel	December	68.322235994	2
9	City Hotel	January	82.160634428	1
10	Resort Hotel	January	48.70891863	2
11	City Hotel	February	86.183025457	1
12	Resort Hotel	February	54.147478336	2
13	Resort Hotel	July	150.122527893	1
14	City Hotel	July	115.563810121	2

The query uses a window function to compare the ADR for every month and rank them accordingly. Further analysis can be done to understand the pattern of the customers which are changing based on the change in seasons.

Query 5.

```
SELECT
  CASE
    WHEN lead_time <= 7 THEN '0-7 days'
    WHEN lead_time <= 14 THEN '8-14 days'
    WHEN lead_time <= 21 THEN '15-21 days'
    WHEN lead_time <= 30 THEN '22-30 days'
    ELSE 'More than 30 days'
  END AS lead_time_category,
  AVG(adr) AS avg_daily_rate,
  COUNT(*) AS num_bookings
FROM
  "hotel_bookings"
WHERE
  is_canceled = 0
GROUP BY
  lead_time_category;
```



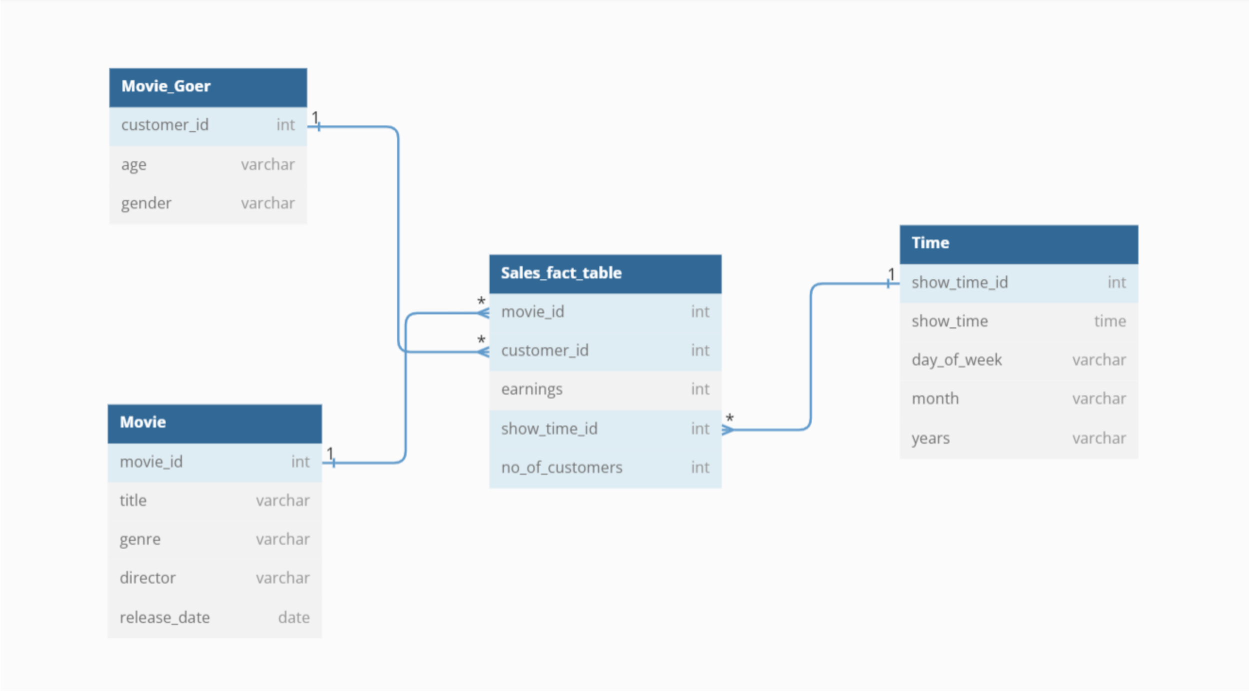
	LEAD_TIME_CATEGORY	AVG_DAILY_RATE	NUM_BOOKINGS
1	More than 30 days	102.941126819	43,645
2	0-7 days	89.246503587	17,844
3	8-14 days	102.914423358	5,480
4	15-21 days	105.873111598	4,104
5	22-30 days	105.502211092	4,093

This query categorizes the ADR for different lengths of stay of the customers and also compare the total number of bookings that fall under each category.

Q1 B:

Blog link:
https://medium.com/@smeetsheth2001/data-warehousing-using-snowflakeuser-experience-fbf5b3a06a31

Q2 A:



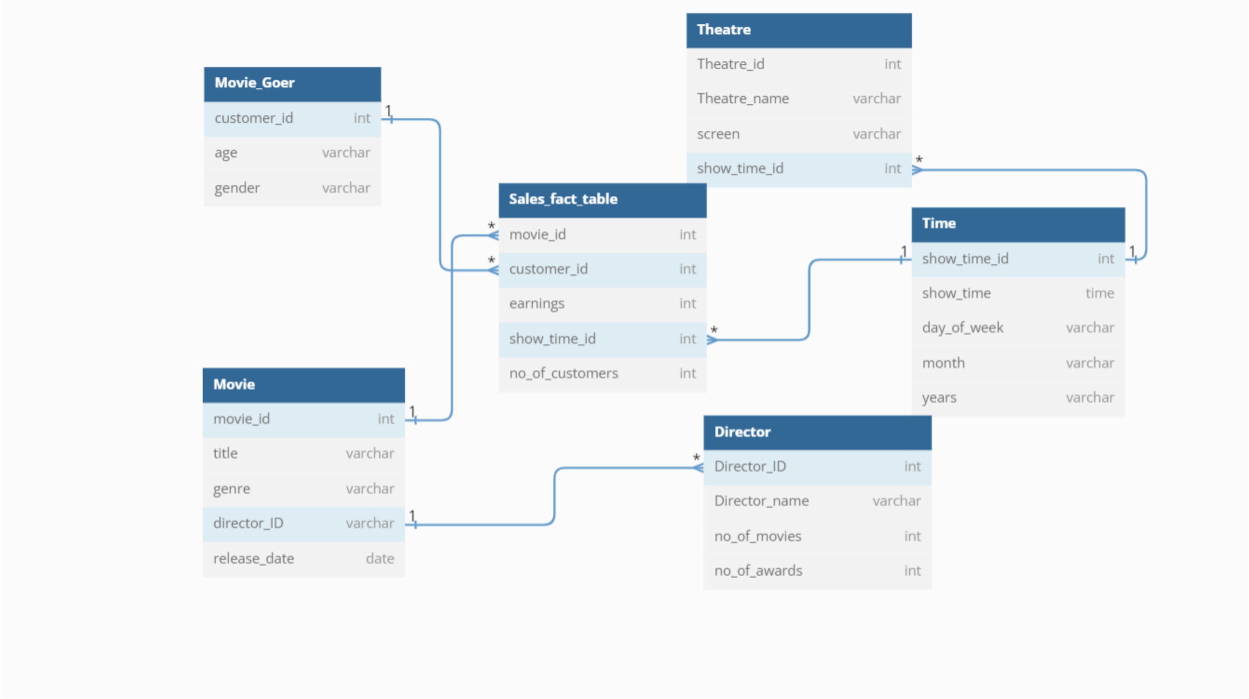
Q2 B:

```
SELECT m.title,sum(sf.revenue) as “Total Revenue”
from Sales_fact_table sf, time t, movie m
WHERE sf.show_time_id=t.show_time_id
and m.movie_id=sf.movie_id
and t.years = 2020
group by m.title;
```

Q2 C:

```
SELECT m.title, SUM(ts.ticket_price) AS “Total Revenue”
FROM ticket_sales ts
JOIN Movie m ON ts.movie = m.title
WHERE ts.year = 2020
GROUP BY m.title
```

Q2 D:



Query 1:

SELECT Theatre_name, screen, SUM(Sales_fact_table.earnings) AS total_earnings
FROM Theatre
JOIN Sales_fact_table ON Time.show_time_id = Sales_fact_table.show_time_id
JOIN Theatre ON time.show_time_id = Theatre.show_time_id
GROUP BY Theatre_name, screen;

Query 2:

SELECT Director_name, no_of_movies
FROM Director
WHERE no_of_movies =
(SELECT MAX(no_of_movies)
FROM Director

where Director_name = “Christopher Nolan”);

Q3 A:

1. Columnar Databases can easily handle data warehousing applications that involve the analysis of very large datasets. For example any retail industry produces a large volume of data such as sales done, customers information, transactions, inventory management, purchase orders etc. Columnar databases can be used to analyze all this data and identify hidden patterns and trends in customer behavior which could be beneficial for the business owners.
2. Instead of iterating over each row, the columnar database can read and process only specific columns needed for the query. This results in faster execution of the queries.
3. Columnar Database can store different type data such as structured, semi structured and unstructured data. For example social media sites generate a vast amount of unstructured data as well as semi structured data. The columnar database can combine all types of data and then perform the analysis to get the sentiment analysis on the post to understand the opinions of the users.
4. It provides various options to select the data source from such as different databases, file systems etc. It can be useful when the company having large data uses a particular type of data storage cloud. It gives them the liberty to use the data source accordingly.
5. Columnar database is a very good option for real time data processing and analysis as it can handle high speed transaction processing quite efficiently.

Q3 B:

1. IBM Cognos: It is BI tool providing different OLAP capabilities used for reporting, analytics and monitoring of events or metrics. It has different modules such as Cognos Planning, Cognos Controller and Cognos Analytics which can be used to retrieve information in different cases.
2. Oracle OLAP is an OLAP which provides various services for the Oracle Database. Custom functions can be made which comprises many small analytical functions which can be used to solve complex analytical problems.
3. Tableau : It is a data visualization tool handling large datasets easily. It is very user friendly and has simple drag and drop functions. It provides multidimensional as well as tabular models which can handle multiple dimensions and measures.
4. Mondrian: It is a java written OLAP engine. It uses MDX language for executing the queries. It also provides multidimensional analysis.
5. Microsoft SSAS: Microsoft SQL Server Analytic Services is a tool used for analysis and data mining to get information from multiple data sources. It also provides time-series analysis, regression analysis and clustering which can be useful while performing ad-hoc queries.