DATA 225: Db for Analytics

Homework 5

3. (a) Explore the use of columnar databases for implementing data warehouse solutions in the industry and how it helps with analytic functions. Giving specific examples from the industry where possible, describe 5 salient observations from your study.

Analytical queries typically access only a few columns at a time, and columnar databases can retrieve and process only the required columns, reducing disk I/O and memory usage.

Columnar databases have gained popularity in recent years as a data warehousing solution for their ability to handle large volumes of data and optimize analytic queries. Here are five salient observations on the use of columnar databases for implementing data warehouse solutions in the industry:

Interesting Use case of Macy

Macy's, one of the largest department store chains in the United States, implemented a data warehouse solution using Amazon Redshift as the underlying columnar database to support its analytics and reporting needs. The data warehousing capabilities of columnar databases like Amazon Redshift helped Macy's in several ways:

One example of Macy's data warehouse implementation is their use of Redshift to analyze customer behavior and optimize marketing campaigns. Macy's had a large amount of customer data from various sources such as transactional systems, online sales systems, and social media platforms. By implementing a data warehouse solution with Amazon Redshift, Macy's was able to consolidate this data and analyze it in real-time to gain insights into customer behavior.

Macy's used Redshift's scalability and elasticity to handle sudden spikes in data volumes during peak sales seasons, such as the holiday season. Redshift's built-in support for analytical functions, such as aggregation and windowing functions, allowed Macy's to perform complex analytics on the data warehouse.

With the new data warehouse solution, Macy's was able to optimize its marketing campaigns by providing personalized recommendations based on customer behavior. For example, Macy's used Redshift to identify customers who frequently purchased high-end cosmetics and sent them targeted promotions for luxury cosmetics. This approach helped Macy's to increase sales and improve customer loyalty.

Analytic functions from different columnar databases are really important to store aggregations or to run complex analytical queries on very large datasets. I have tried to describe some of the analytic functions from different columnar databases, and their usage in real time with different companies

Columnar Database	**Analytic Function**	**Company**	**Real time use case**
Amazon Redshift	RANK()	Lyft	Used to rank drivers on different performance metrics
Google BigQuery	ARRAY_AGG()	Spotify	aggregate user listening data by playlists and artists
SAP HANA	PREDICT()	Coca-cola	Used to predict product demand & optimize supply chain
Vertica	REGEX_REPLACE()	Bank of America	to clean and standardize customer data for fraud detection
Cassandra	Basic SUM, COUNT, MAX	eBay	counting the number of items in each category or tracking inventory levels.

5 of the most important features of Columnar databases and suitable examples below;

1. **Improved query performance:** Store data as columns for faster retrieval. Amazon Redshift is a popular columnar database used by companies such as Yelp and Airbnb for their data warehousing needs.

2. **Data compression:** Compression algorithms for optimized data storage and faster query performance. For example, Vertica is a columnar database used by companies such as Comcast and Etsy that uses compression techniques such as run-length encoding and delta encoding to reduce storage requirements.

3. **Support for analytic functions**: Columnar databases support a wide range of analytic functions such as window functions, aggregations, and statistical functions, which enable advanced data analysis and reporting. Google bigquery for The New York Times to help with aggregations and ML algorithms.

4. **Integration with big data frameworks**: Columnar databases can integrate with big data frameworks such as Hadoop and Spark, enabling companies to process and analyze large

volumes of data in real-time. For example, Apache Cassandra is a columnar database used by companies such as Apple and Netflix that can be integrated with Hadoop and Spark for large-scale data processing.

5. **Scalability:** Columnar databases can scale horizontally by adding more nodes to a cluster, enabling companies to handle increasing volumes of data and queries. For example, Snowflake is a cloud-based columnar database used by companies such as Adobe and Capital One that can scale automatically to handle any amount of data and users.

3 (b) Study the various OLAP tools used in the industry such as IBM Cognos, Microsoft SSAS, Oracle OLAP, etc. Giving specific examples from the industry where possible, describe 5 salient observations from your study.

OLAP (Online Analytical Processing) tools provide powerful capabilities for managing and analyzing data, including multidimensional data analysis, aggregation and drill-down capabilities, data visualization, and advanced analytics.

They enable organizations to gain insights into their business operations, make data-driven decisions, and improve their overall performance. OLAP tools also provide the scope of extensive and exhaustive reporting capabilities which also include real time analytics to help the business to take data driven decisions and drive their business decisions accordingly.

Below are some of the salient features which I have summarized from studying different OLAP tools:

1. **Multidimensional Data Analysis:** OLAP tools allow users to analyze data across multiple dimensions, providing a more intuitive way of understanding complex data relationships. For example, Coca-Cola Enterprises uses IBM Cognos OLAP to analyze sales data by product, market, and time period to gain insights into customer behavior and preferences.

2. **Aggregation and Drill-Down Capabilities**: OLAP tools allow users to aggregate data at different levels of granularity and drill down to view data at a more detailed level. For example, Salesforce uses Microsoft SQL Server Analysis Services (SSAS) to analyze customer data by region, product, and sales rep to identify trends and patterns in customer behavior.

3. **Data Visualization**: OLAP tools often include powerful data visualization capabilities, such as charts, graphs, and dashboards, which enable users to explore and communicate their findings more effectively. For example, Airbnb uses Tableau OLAP to create data visualizations that help them make data-driven decisions about pricing and inventory management.

4. **Advanced Analytics**: OLAP tools often include advanced analytics capabilities, such as data mining, predictive modeling, and statistical analysis. For example, eBay uses Oracle OLAP to analyze customer behavior and preferences, and to make personalized product recommendations based on their browsing and purchase history.

5. **Integration with Other Tools:** OLAP tools can integrate with other business intelligence and analytics tools, such as data warehouses, ETL (Extract, Transform, Load) tools, and reporting tools, to provide a more comprehensive solution for managing and analyzing data. For example, Adobe uses SAP BusinessObjects OLAP, which integrates with SAP's data warehouse and reporting tools, to provide a complete business intelligence solution.

6. **Scalability:** OLAP tools can handle large volumes of data, making them suitable for use in enterprise-level applications. For example, Walmart uses Oracle OLAP to analyze sales data from over 11,000 stores across the world, and to make real-time decisions about inventory management and pricing.

7. **Real-Time Analytics**: OLAP tools can provide real-time analytics capabilities, allowing companies to analyze data as it is generated. For example, PayPal uses Microsoft SSAS to analyze transactional data in real-time to detect fraud and prevent unauthorized transactions.

8. **Security and Access Control:** OLAP tools often include security features that enable administrators to control access to data and ensure that sensitive information is only accessible to authorized users. For example, Wells Fargo uses IBM Cognos OLAP to analyze customer data while ensuring that customer information is kept secure and confidential.

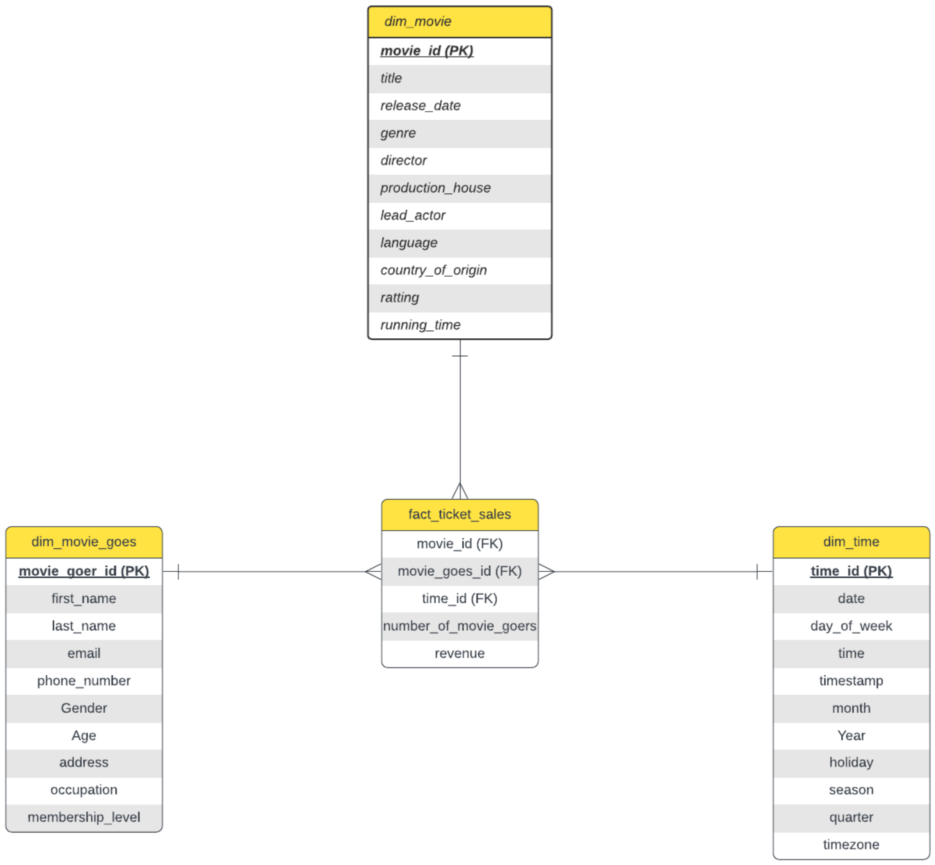
2. The Data warehouse of ABC Theaters has 3 dimensions: movie, movie_goer, and time and two summary metrics: number_of_movie_goers and revenue.

(a) The three dimensional tables with the schemas below,

1. **dim_movie:** movie_id (PK), title, release_date, genre, director, production_house, lead_actor, language, country_of_origin, rating, running_time.
2. **dim_time:** time_id (PK), date, day_of_week, timestamp, time, month, year, holiday, season, quarter, timezone
3. **dim_movie_goer:** movie_goer_id (PK), first_name,last_name, email, phone_number, gender, age, address, occupation, membership_level

The only fact table in the below star schema below;

1. **fact_ticket_sales:** movie_id (FK), time_id (FK), movie_goer_id (FK), number_of_movie_goers, revenue



The above star schema allows us to retrieve the all the below information;

- What is the membership_level of the majority of the movie_goers?
- What is the revenue contribution across different months for any movie?
- Which day of the week produces the highest revenue?
- What is the average revenue generated per movie_goer across different age groups?
- What is the average revenue generated per movie_goer across different occupations?
- Which locality does the highest revenue contributors belong from?
- Which combination of director and production house produced the highest revenue for what movie?
- Which genre of movies has the lowest average revenue per movie_goer?

(b) Assuming a relational model, write the SQL join query on the dimension and fact tables to output the total revenue collected through ticket sales by each movie in the pandemic year, 2020.

```
with main as ( select title as movie, sum(revenue) as total_revenue
from fact_ticket_sales ft
left join dim_movie dm on ft.movie_id=dm.movie_id
left join dim_time dt on ft.time_id=dt.time_id
where dt.year=2020
Group by title)

Select * from main
```

The above query is going to give you the output, each movie wise total sales generated for the pandemic year 2020.

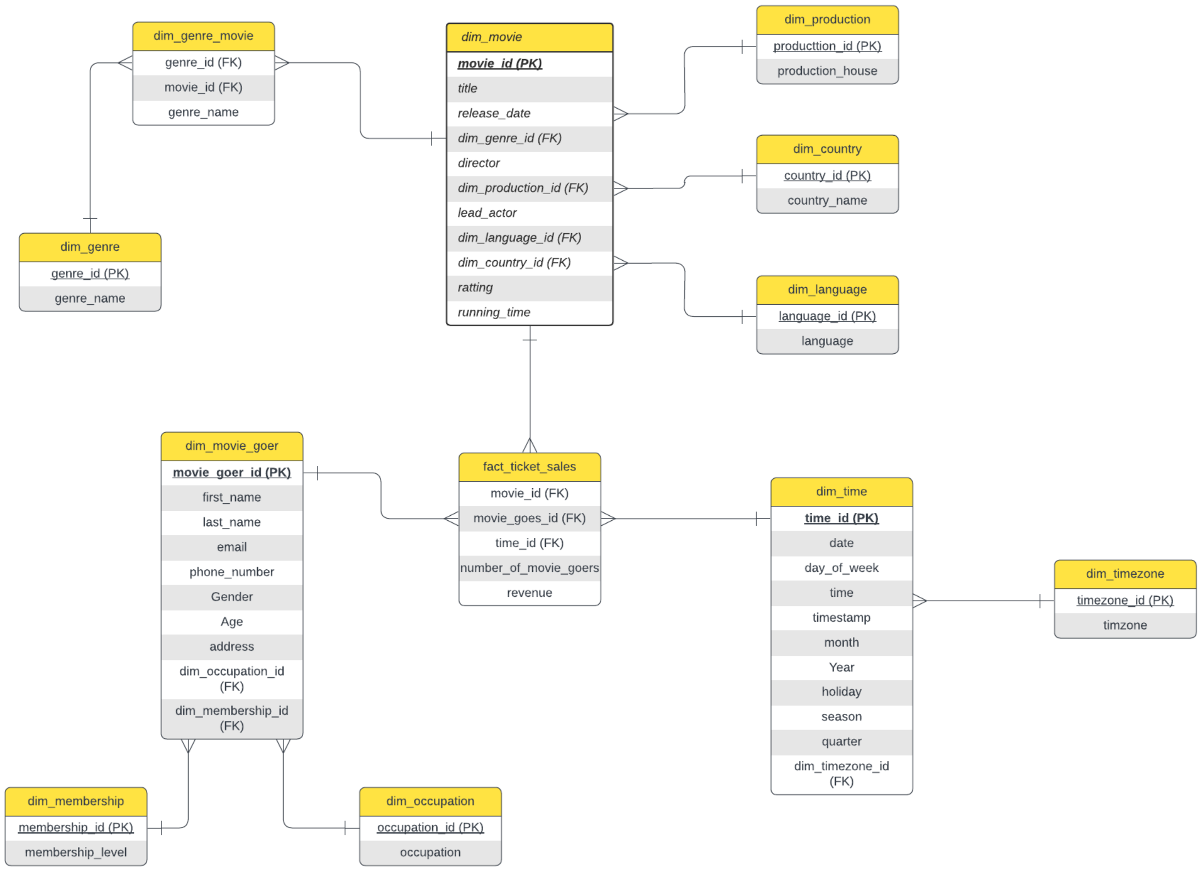
(c) Suppose the transactional data was stored in an RDBMS table, ticket_sales with 7 columns: theater, day, month, year, movie, movie_goer, ticket_price. Write the SQL query to generate the same output as [b] from this table.

```
with main as ( select movie, sum(ticket_price) as total_revenue
from ticket_sales ts
where year=2020
Group by movie)

Select * from main
```

The above query is going to give you the output, each movie wise total sales generated for the pandemic year 2020, just like [b].

(d) Expand the above data warehouse to justify the possibility of a Snowflake schema. What additional dimensional tables will you add and what additional summary information can you generate from it? Draw the snowflake schema and list one or more join queries you can perform to output the additional summary information.

Below is the snowflake schema diagram for the above star schema, after I added some more dimensional tables, thereby adding some foreign keys, and primary keys into existing and new tables respectively.



From the above snowflake schema, below are the attribute and tables description;

1. **dim_movie:** movie_id (PK), title, release_date, dim_genre_id (FK), director, dim_production_id (FK), lead_actor, dim_language_id (FK), dim_country_id (FK), rating, running_time
2. **dim_genre_movie**: dim_genre_id (FK), dim_movie_id (FK), genre_name
3. **dim_genre**: genre_id (PK), genre_name
4. **dim_production**: production_id (PK), production_house
5. **dim_country**: country_id (PK), country_name
6. **dim_language**: language_id (PK), language_name
7. **dim_movie_goer:** movie_goer_id (PK), first_name, last_name, email, phone_number, gender, age, address, dim_occupation_id (FK), dim_membership_id (FK)
8. **dim_membership**: membership_id (PK), membership_level
9. **dim_occupation**: occupation_id (PK), occupation
10. **dim_time:** time_id (PK), date, day_of_week, time, timestamp, month, holiday, year,season,quarter, dim_timezone_id (FK)
11. **dim_timezone**: timezone_id (PK), timezone

12. **fact_ticket_sales**: dim_movie_id (FK), dim_movie_goer_id(FK), dim_time_id (FK), number_of_movie_goers, revenue

Below are the use cases we can build from the above snowflake schema;

1. What is the total revenue and total movie goers of different membership levels across all the years?

```
with main as (select membership_level, year,sum(revenue) as total_revenue,
sum(number_of_movie_goers) as total_movie_goers
from fact_ticket_sales ft
left join dim_movie_goer dm on ft.dim_movie_goer_id = dm.movie_goer_id
left join dim_membership dms on dm.dim_membership_id=dms.membership_id
left join dim_time dt on ft.dim_time_id=dt.time_id
group by membership_level, year
Order by year,membership_level DESC)

Select * from main;
```

The above query will give us year wise, how different membership levels contribute towards the total revenue and number of moviegoers in the descending order, last year first. This will help the customer retention team of ABC Theaters to understand how well their loyalty program has been functioning across different years.

2. Top 10 movies with respect to each genre all time? Criteria for ranking the movies is - (*w.r.t 0.6*no_of_movie_goers+0.4*revenue***)**

```
with main as (select genre_name,title as movie, sum(revenue) as total_revenue,
sum(number_of_movie_goers) as movie_goers
from fact_ticket_sales fts
Left join dim_movie dm on fts.dim_movie_id=dm.movie_id
Left join dim_genre_movie dgm on dm.movie_id=dgm.dim_movie_id and
dm.dim_genre_id=dgm.dim_genre_id
Left join dim_genre dg on dgm.dim_genre_id=dg.genre_id
Group by genre_name,title),

rank_metric as ( select genre_name, movie, total_revenue, movie_goers,
(0.6*movie_goers+0.4*total_revenue) as rank_metric
From main),

Rank_movie as (select genre_name,movie,total_revenue,movie_goers,
DENSE_RANK() OVER(PARTITION BY genre_name ORDER BY rank_metric DESC) as movie_rank
From rank_metric)
Select * from rank_movie where movie_rank < 11
```

The above query will give us the genre wise top 10 films in that genre. The top 10 movies selection criteria will be those who had a movie_rank < 11 for that particular genre. This kind of information will give the customers the choice of watching a movie, based on a genre, and the popularity of that film. This information will also help the stakeholders of those theaters in further understanding their customer-movie preferences, based on the past revenue collections.

3. What is the majority occupation of the movie_goers who watched at least 5 movies in the year 2021?

```
with main as (select movie_goer_id, movie_id,year,dim_occupation_id
from fact_ticket_sales fts
Inner join dim_movie_goer dmg on fts.dim_movie_goer_id=dmg.movie_goer_id
Inner join dim_time dt on fts.dim_time_id=dt.time_id
where year=2021),

Main2 as (Select movie_goer_id,occupation,count(distinct movie_id) as distinct_movies
From main
Left join dim_occupation do on main.dim_occupation_id=do.occupation_id
Group by movie_goer_id,occupation)

Select occupation,count(distinct movie_goer_id) as movie_goers
from main2
Where distinct_movies > 5
Group by occupation
Order by count(distinct movie_goer_id) DESC
```

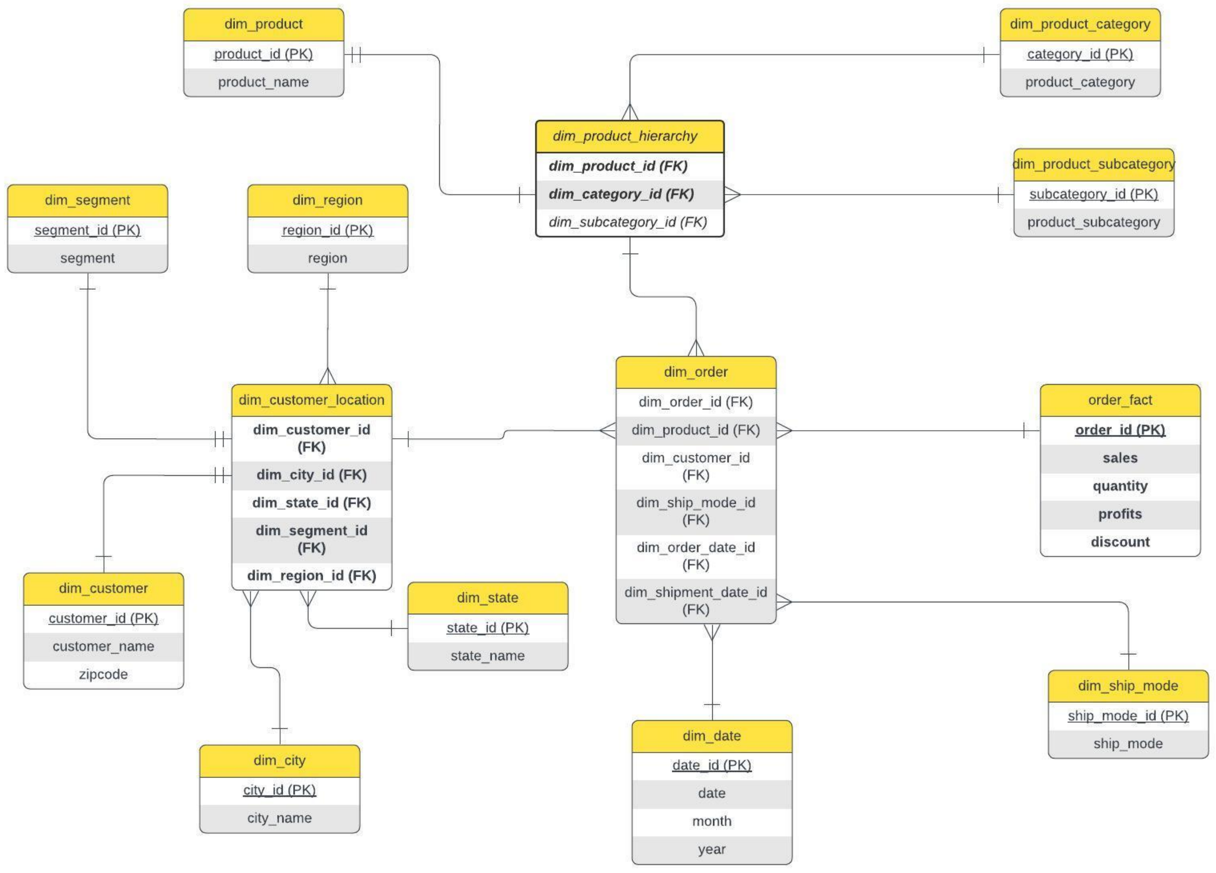
The above query will give us the list of occupations, the movie goers who watched more than 5 movies in 2021 belong to, in the descending order w.r.t count of movie goers. This data will help the marketing team to advertise to a set of similar people who have the same occupation. This will also help the customers to choose a similar movie which their co-worked might have liked.

Other queries which we can answer through the above snowflake schema are:

1. Average box office collections on any day of the week (Saturday, Friday, etc…)
2. The highest number of films came in which country and in what language?
3. The language that people usually enjoy movies in different genres?
4. Combination of Production_house, and languages? How did they fare in terms of number of moviegoers?
5. Monthly contribution to the yearly revenue of these theaters?
6. Occupation wise number of moviegoers on different days of the week? So that we can target those sets of customers for their most likely visit day.

1. Similar to what you did for HW4, implement a small project of your choice, but this time, to exercise the data warehousing and analytical capabilities of the Snowflake Data Platform. It is ok to use online datasets to populate your databases, but your project should be different from the solutions available online. Provide screenshots and as many details to illustrate your work.

I have modeled Superstore (Tableau dataset) as a snowflake schema below, and have implemented some queries on the same.



Schema detailed below:

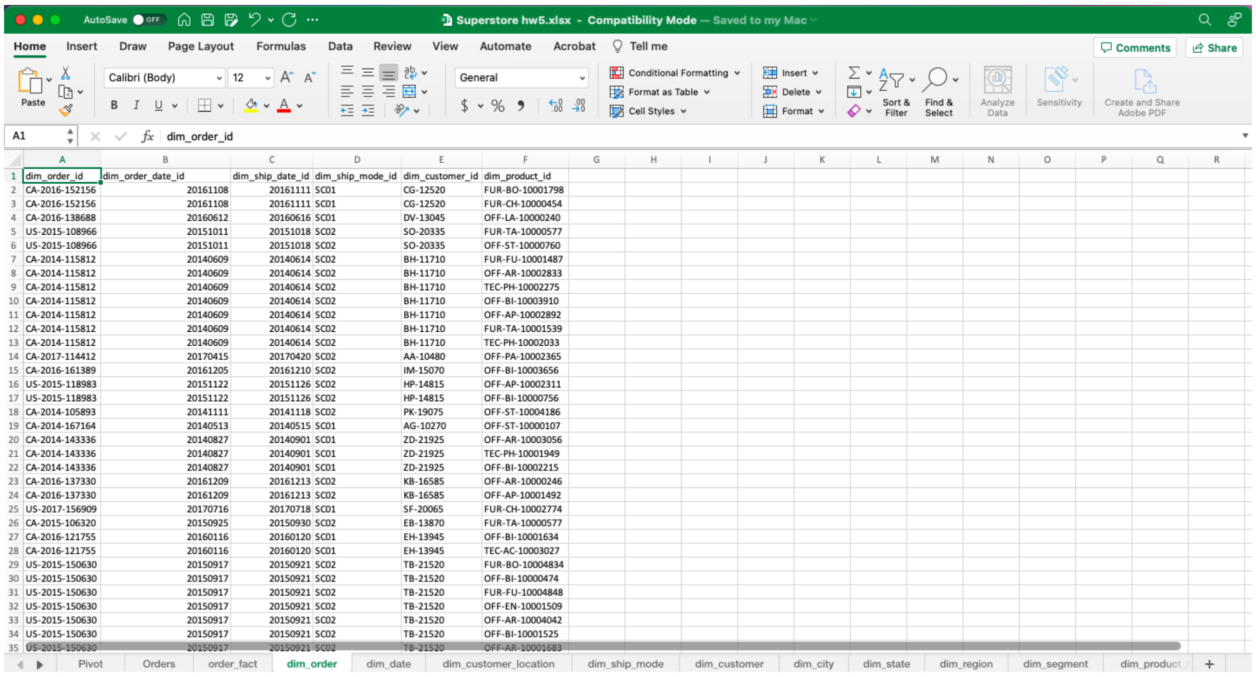
- **Order_fact:** order_id (PK), sales, quantity, profits, discount.
- **dim_order:** dim_order_id (FK), dim_product_id (FK), dim_customer_id (FK), dim_ship_mode_id (FK), dim_order_date_id (FK), dim_shipment_date_id (FK)
- **dim_date:** date_id (PK), date, month, year
- **dim_customer_location**: dim_customer_id (FK), dim_city_id (FK), dim_state_id (FK), dim_segment_id (FK), dim_region_id (FK).
- **dim_customer**: customer_id (PK), customer_name

- **dim_city:** city_id (PK), city_name
- **dim_state:** state_id (PK), state_name
- **dim_region:** region_id (PK), region
- **dim_segment**: segment_id (PK), segment
- **dim_product_hierarchy**: dim_product_id (FK), dim_category_id (FK), dim_subcategory_id (FK)
- **dim_product**: product_id (PK), product_name
- **dim_category:** category_id (PK), product_category
- **dim_subcategory:** subcategory_id (PK), product_subcategory
- **dim_ship_mode**: ship_mode_id (PK), ship_mode

We have modeled the above schema in such a way that the highest level of normalization is achieved, because of which there are way too many dimension tables, and only 1 fact table.

We will first create these tables in excel using in-built functions like VLOOKUP()

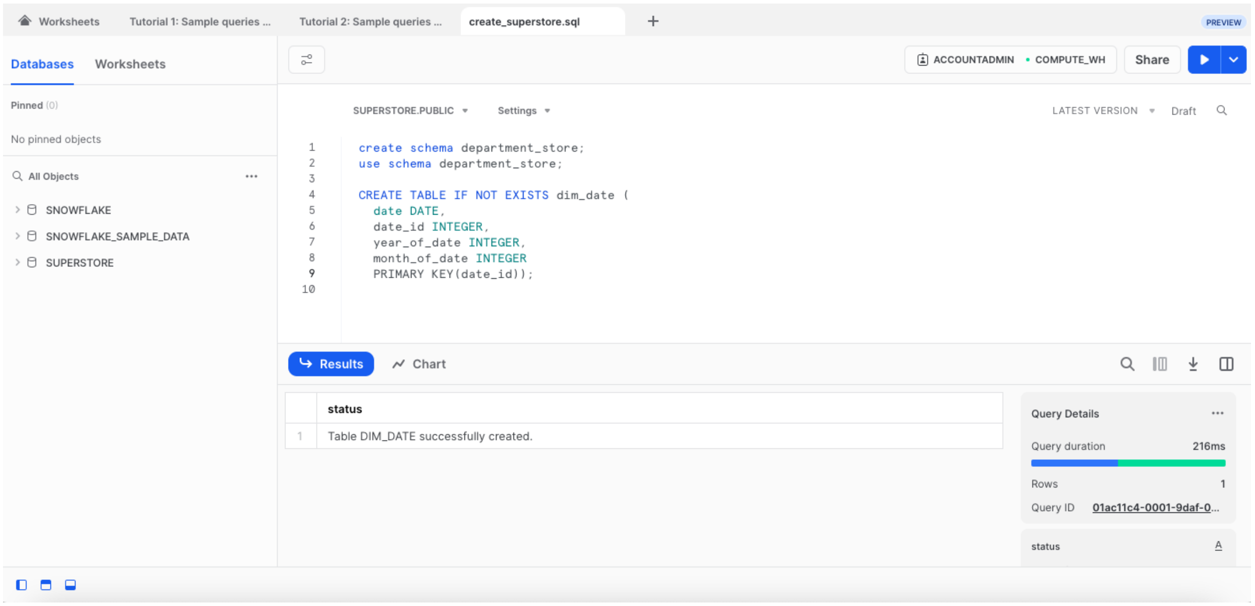
=VLOOKUP($O2,dim_category!A:B,2,0)


Superstore hw5.xlsx - Compatibility Mode — Saved to my Mac
Home Insert Draw Page Layout Formulas Data Review View Automate Acrobat Tell me
Comments Share
Pivot Orders order_fact dim_order dim_date dim_customer_location dim_ship_mode dim_customer dim_city dim_state dim_region dim_segment dim_product

A1 = dim_order_id

	A	B	C	D	E	F
1	dim_order_id	dim_order_date_id	dim_ship_date_id	dim_ship_mode_id	dim_customer_id	dim_product_id
2	CA-2016-152156	20161108	20161111	SC01	CG-12520	FUR-BO-10001798
3	CA-2016-152156	20161108	20161111	SC01	CG-12520	FUR-CH-10000454
4	CA-2016-138688	20160612	20160616	SC01	DV-13045	OFF-LA-10000240
5	US-2015-108966	20151011	20151018	SC02	SO-20335	FUR-TA-10000577
6	US-2015-108966	20151011	20151018	SC02	SO-20335	OFF-ST-10000760
7	CA-2014-115812	20140609	20140614	SC02	BH-11710	FUR-FU-10001487
8	CA-2014-115812	20140609	20140614	SC02	BH-11710	OFF-AR-10002833
9	CA-2014-115812	20140609	20140614	SC02	BH-11710	TEC-PH-10002275
10	CA-2014-115812	20140609	20140614	SC02	BH-11710	OFF-BI-10003910
11	CA-2014-115812	20140609	20140614	SC02	BH-11710	OFF-AP-10002892
12	CA-2014-115812	20140609	20140614	SC02	BH-11710	FUR-TA-10001539
13	CA-2014-115812	20140609	20140614	SC02	BH-11710	TEC-PH-10002033
14	CA-2017-114412	20170415	20170420	SC02	AA-10480	OFF-PA-10002365
15	CA-2016-161389	20161205	20161210	SC02	IM-15070	OFF-BI-10003656
16	US-2015-118983	20151122	20151126	SC02	HP-14815	OFF-AP-10002311
17	US-2015-118983	20151122	20151126	SC02	HP-14815	OFF-BI-10000756
18	CA-2014-105893	20141111	20141118	SC02	PK-19075	OFF-ST-10004186
19	CA-2014-167164	20140513	20140515	SC01	AG-10270	OFF-ST-10000107
20	CA-2014-143336	20140827	20140901	SC01	ZD-21925	OFF-AR-10003056
21	CA-2014-143336	20140827	20140901	SC01	ZD-21925	TEC-PH-10001949
22	CA-2014-143336	20140827	20140901	SC01	ZD-21925	OFF-BI-10002215
23	CA-2016-137330	20161209	20161213	SC02	KB-16585	OFF-AR-10000246
24	CA-2016-137330	20161209	20161213	SC02	KB-16585	OFF-AP-10001492
25	US-2017-156909	20170716	20170718	SC01	SF-20065	FUR-CH-10002774
26	CA-2015-106320	20150925	20150930	SC02	EB-13870	FUR-TA-10000577
27	CA-2016-121755	20160116	20160120	SC01	EH-13945	OFF-BI-10001634
28	CA-2016-121755	20160116	20160120	SC01	EH-13945	TEC-AC-10003027
29	US-2015-150630	20150917	20150921	SC02	TB-21520	FUR-BO-10004834
30	US-2015-150630	20150917	20150921	SC02	TB-21520	OFF-BI-10000474
31	US-2015-150630	20150917	20150921	SC02	TB-21520	FUR-FU-10004848
32	US-2015-150630	20150917	20150921	SC02	TB-21520	OFF-EN-10001509
33	US-2015-150630	20150917	20150921	SC02	TB-21520	OFF-AR-10004042
34	US-2015-150630	20150917	20150921	SC02	TB-21520	OFF-BI-10001525
35	US-2015-150630	20150917	20150921	SC02	TB-21520	OFF-AR-10001683

Once we have the separate csv files ready for each of these tables, we will start creating them in the snowflake schema, below is a screenshot of such an example.



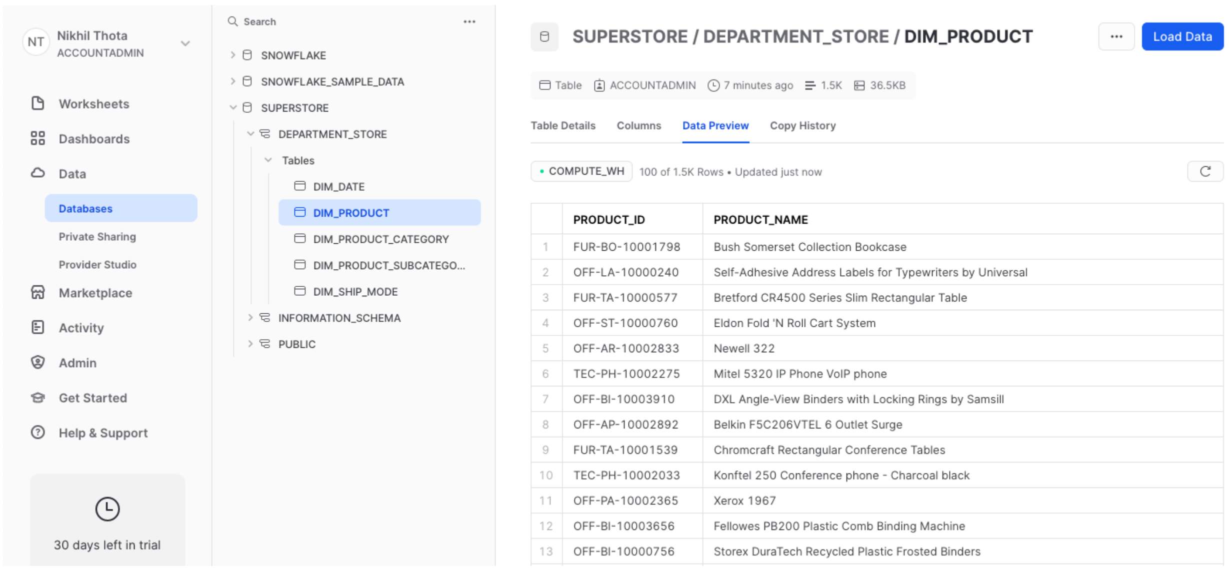
```
create schema department_store;
use schema department_store;

CREATE TABLE IF NOT EXISTS dim_date (
  date DATE,
  date_id INTEGER,
  year_of_date INTEGER,
  month_of_date INTEGER
  PRIMARY KEY(date_id));
```

	status
1	Table DIM_DATE successfully created.

Query Details
Query duration 216ms
Rows 1
Query ID 01ac11c4-0001-9daf-0...
status

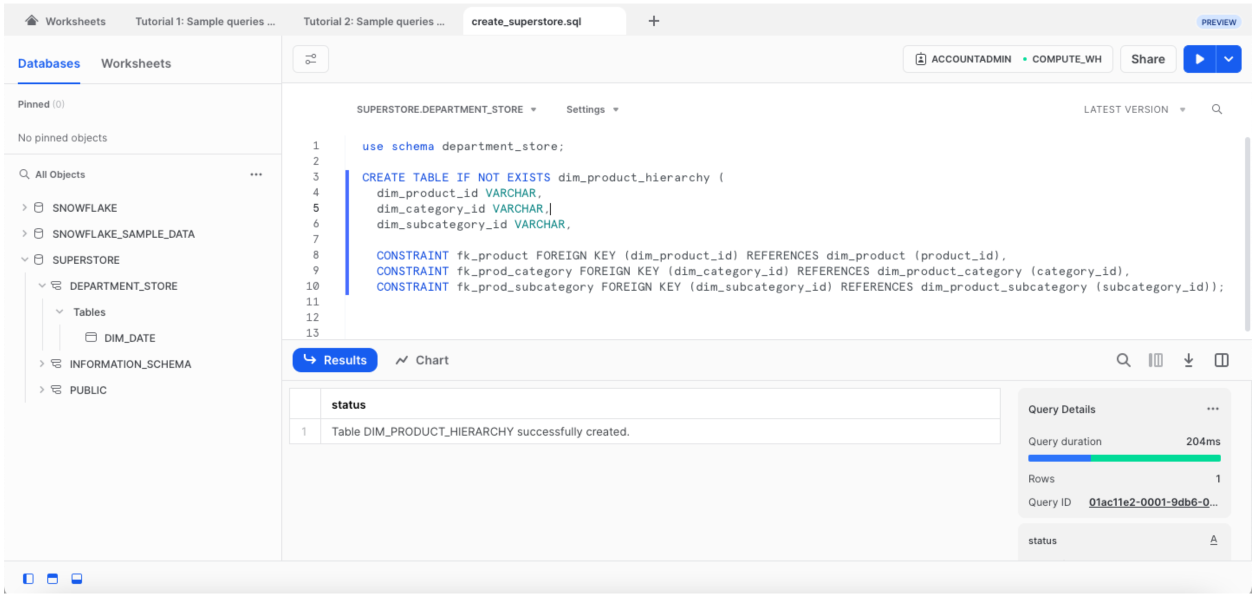
We will implement all the create SQL queries for all the tables one by one; [Scripts here]

While creating the data, we are also going to insert into these tables using the data import option that snowflake has, below is one such table which we imported using the data import option of snowflake UI



	PRODUCT_ID	PRODUCT_NAME
1	FUR-BO-10001798	Bush Somerset Collection Bookcase
2	OFF-LA-10000240	Self-Adhesive Address Labels for Typewriters by Universal
3	FUR-TA-10000577	Bretford CR4500 Series Slim Rectangular Table
4	OFF-ST-10000760	Eldon Fold 'N Roll Cart System
5	OFF-AR-10002833	Newell 322
6	TEC-PH-10002275	Mitel 5320 IP Phone VoIP phone
7	OFF-BI-10003910	DXL Angle-View Binders with Locking Rings by Samsill
8	OFF-AP-10002892	Belkin F5C206VTEL 6 Outlet Surge
9	FUR-TA-10001539	Chromcraft Rectangular Conference Tables
10	TEC-PH-10002033	Konftel 250 Conference phone - Charcoal black
11	OFF-PA-10002365	Xerox 1967
12	OFF-BI-10003656	Fellowes PB200 Plastic Comb Binding Machine
13	OFF-BI-10000756	Storex DuraTech Recycled Plastic Frosted Binders

We also enforced constraints on Foreign Key Relationships, below is the example of one such screenshot;

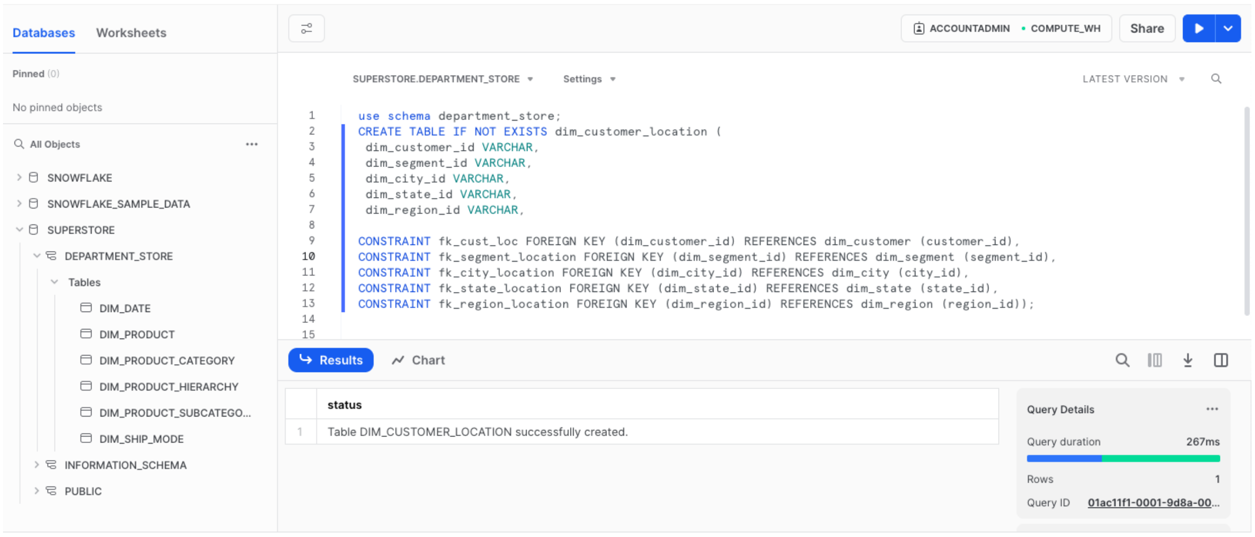


```sql
use schema department_store;

CREATE TABLE IF NOT EXISTS dim_product_hierarchy (
  dim_product_id VARCHAR,
  dim_category_id VARCHAR,
  dim_subcategory_id VARCHAR,

  CONSTRAINT fk_product FOREIGN KEY (dim_product_id) REFERENCES dim_product (product_id),
  CONSTRAINT fk_prod_category FOREIGN KEY (dim_category_id) REFERENCES dim_product_category (category_id),
  CONSTRAINT fk_prod_subcategory FOREIGN KEY (dim_subcategory_id) REFERENCES dim_product_subcategory (subcategory_id));
```

	status
1	Table DIM_PRODUCT_HIERARCHY successfully created.

Query Details
Query duration 204ms
Rows 1
Query ID 01ac11e2-0001-9db6-0...
status

Some of the dimensional tables are just a combination of dimensions, we have grouped them like this for the sake of Snowflake modeling, below is the creation code for one such table



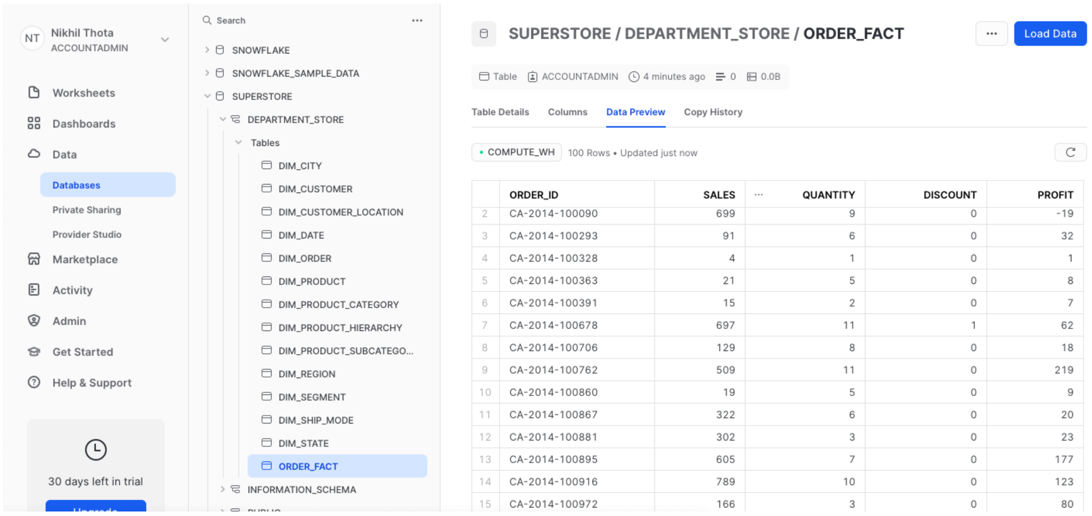
```sql
use schema department_store;
CREATE TABLE IF NOT EXISTS dim_customer_location (
 dim_customer_id VARCHAR,
 dim_segment_id VARCHAR,
 dim_city_id VARCHAR,
 dim_state_id VARCHAR,
 dim_region_id VARCHAR,

CONSTRAINT fk_cust_loc FOREIGN KEY (dim_customer_id) REFERENCES dim_customer (customer_id),
CONSTRAINT fk_segment_location FOREIGN KEY (dim_segment_id) REFERENCES dim_segment (segment_id),
CONSTRAINT fk_city_location FOREIGN KEY (dim_city_id) REFERENCES dim_city (city_id),
CONSTRAINT fk_state_location FOREIGN KEY (dim_state_id) REFERENCES dim_state (state_id),
CONSTRAINT fk_region_location FOREIGN KEY (dim_region_id) REFERENCES dim_region (region_id));
```

	status
1	Table DIM_CUSTOMER_LOCATION successfully created.

Query Details
Query duration 267ms
Rows 1
Query ID 01ac11f1-0001-9d8a-00...

Once all the dimension and fact tables are created, I have inserted the data into each of them from the individual csv files which I collated using VLOOKUP() and excel transformations from the superstore (single sheet) dataset.

Below is the screenshot of all the dimension and fact tables created and ready in Snowflake;



SUPERSTORE / DEPARTMENT_STORE / ORDER_FACT

Table · ACCOUNTADMIN · 4 minutes ago · 0 · 0.0B

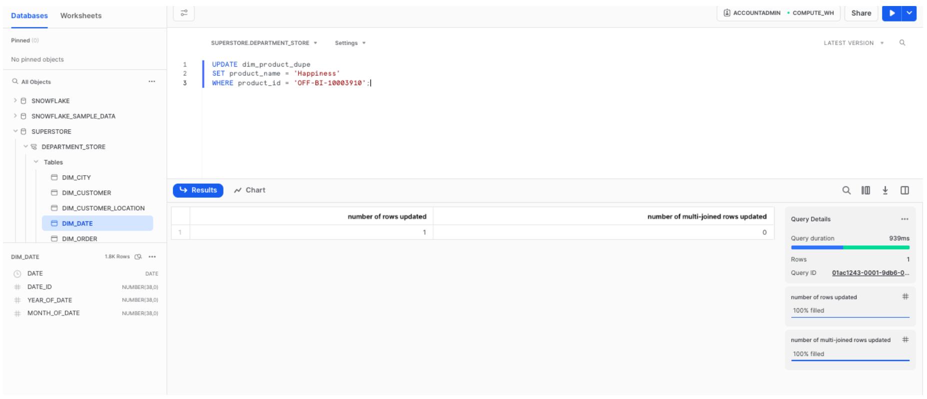
COMPUTE_WH · 100 Rows • Updated just now

	ORDER_ID	SALES	QUANTITY	DISCOUNT	PROFIT
2	CA-2014-100090	699	9	0	-19
3	CA-2014-100293	91	6	0	32
4	CA-2014-100328	4	1	0	1
5	CA-2014-100363	21	5	0	8
6	CA-2014-100391	15	2	0	7
7	CA-2014-100678	697	11	1	62
8	CA-2014-100706	129	8	0	18
9	CA-2014-100762	509	11	0	219
10	CA-2014-100860	19	5	0	9
11	CA-2014-100867	322	6	0	20
12	CA-2014-100881	302	3	0	23
13	CA-2014-100895	605	7	0	177
14	CA-2014-100916	789	10	0	123
15	CA-2014-100972	166	3	0	80

We will now write some SQL queries which can provide useful information from the created snowflake schema to the respective stakeholders.

Examples of CRUD operations with snowflake schema, and the above dataset;

Update operation;

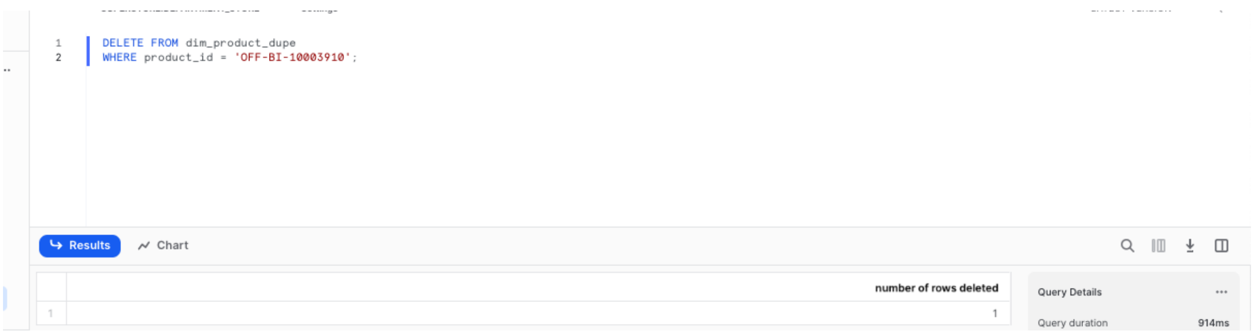


```
UPDATE dim_product_dupe
SET product_name = 'Happiness'
WHERE product_id = 'OFF-BI-10003910';
```

	number of rows updated	number of multi-joined rows updated
1	1	0

Query Details
Query duration 939ms
Rows 1
Query ID 01ac1243-0001-9db6-0...

Delete Operation;



```
DELETE FROM dim_product_dupe
WHERE product_id = 'OFF-BI-10003910';
```

	number of rows deleted
1	1

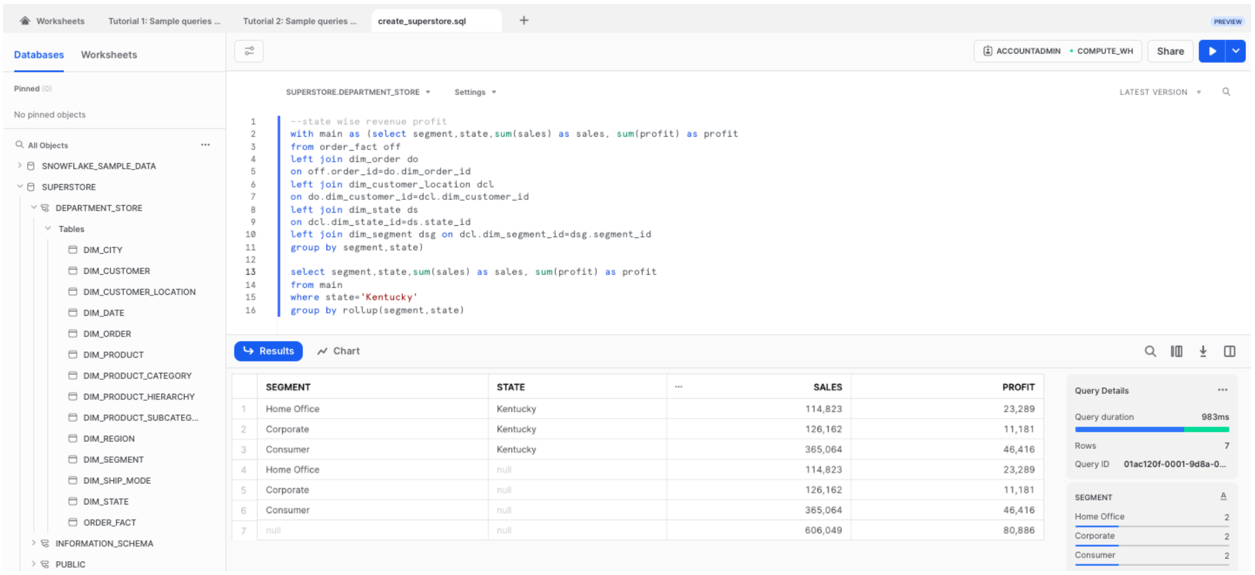
Query Details
Query duration 914ms

1. **Query to understand the profits of State wise consumer segment level, and see which segment is performing good at which geo?**

with main as (select segment,state,sum(sales) as sales, sum(profit) as profit
from order_fact off
left join dim_order do
on off.order_id=do.dim_order_id
left join dim_customer_location dcl
on do.dim_customer_id=dcl.dim_customer_id
left join dim_state ds
on dcl.dim_state_id=ds.state_id
left join dim_segment dsg on dcl.dim_segment_id=dsg.segment_id
group by segment,state)

select segment,state,sum(sales) as sales, sum(profit) as profit
from main
group by rollup(segment,state)

* I have added a filter on state, just to see how rollup works with dimensions, and totalling



```sql
--state wise revenue profit
with main as (select segment,state,sum(sales) as sales, sum(profit) as profit
from order_fact off
left join dim_order do
on off.order_id=do.dim_order_id
left join dim_customer_location dcl
on do.dim_customer_id=dcl.dim_customer_id
left join dim_state ds
on dcl.dim_state_id=ds.state_id
left join dim_segment dsg on dcl.dim_segment_id=dsg.segment_id
group by segment,state)

select segment,state,sum(sales) as sales, sum(profit) as profit
from main
where state='Kentucky'
group by rollup(segment,state)
```

	SEGMENT	STATE	SALES	PROFIT
1	Home Office	Kentucky	114,823	23,289
2	Corporate	Kentucky	126,162	11,181
3	Consumer	Kentucky	365,064	46,416
4	Home Office	null	114,823	23,289
5	Corporate	null	126,162	11,181
6	Consumer	null	365,064	46,416
7	null	null	606,049	80,886

Query Details
Query duration 983ms
Rows 7
Query ID 01ac120f-0001-9d8a-0...

The above rollup gives us the subtotals of state & segment level, segment level, and Grand total. Some of the insight from the above query;

- Consumer segment of California state generates the highest revenue, and is the most profitable segment & state combination.
- Consumer, Corporate from California state are the top 2 most profitable segments from all the possible combinations of segments and state.
- Useful to marketing, sales teams to drive their sales and advertising according to the geo.

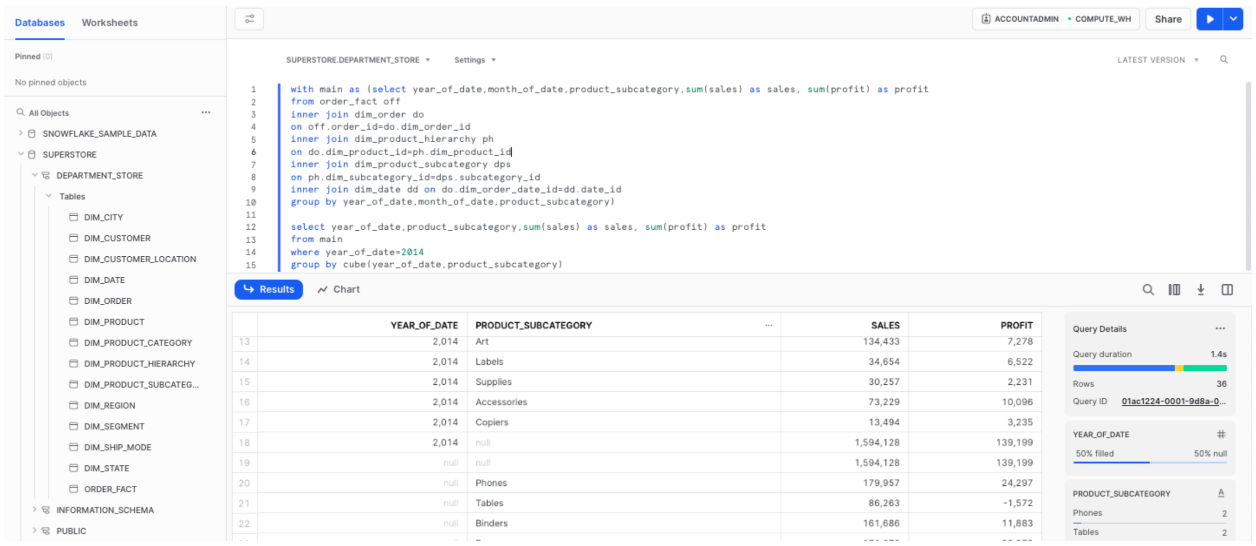
- The Home Office segment of Colorado state is the most loss making segment out of all the states, this puts that particular segment stakeholders on high alert, to do something about it.
- This data can also help with the optimization of cost, by re-sizing the workforce for the respective segments according to the trend from this data.

2. Subcategory level sales, and Profit report for Catalog Operations team, so that they can plan the demand and procurement accordingly.

with main as (select year,month,subcategory,sum(sales) as sales, sum(profit) as profit
from order_fact off
inner join dim_order do
on off.order_id=do.dim_order_id
inner join dim_product_hierarchy ph
on do.dim_product_id=ph.dim_product_id
inner join dim_product_subcategory dps
on ph.dim_subcategory_id=dps.subcategory_id
inner join dim_date dd on do.dim_order_date_id=dd.date_id
group by year,month,subcategory)

select year,subcategory,sum(sales) as sales, sum(profit) as profit
from main
group by cube(year,subcategory)

* I have given a filter on the year in the below screenshot to understand how cube() works, it looks like it gives us all the possible combinations of the subtotals.



SUPERSTORE.DEPARTMENT_STORE

```
with main as (select year_of_date,month_of_date,product_subcategory,sum(sales) as sales, sum(profit) as profit
from order_fact off
inner join dim_order do
on off.order_id=do.dim_order_id
inner join dim_product_hierarchy ph
on do.dim_product_id=ph.dim_product_id
inner join dim_product_subcategory dps
on ph.dim_subcategory_id=dps.subcategory_id
inner join dim_date dd on do.dim_order_date_id=dd.date_id
group by year_of_date,month_of_date,product_subcategory)

select year_of_date,product_subcategory,sum(sales) as sales, sum(profit) as profit
from main
where year_of_date=2014
group by cube(year_of_date,product_subcategory)
```

	YEAR_OF_DATE	PRODUCT_SUBCATEGORY	SALES	PROFIT
13	2,014	Art	134,433	7,278
14	2,014	Labels	34,654	6,522
15	2,014	Supplies	30,257	2,231
16	2,014	Accessories	73,229	10,096
17	2,014	Copiers	13,494	3,235
18	2,014	null	1,594,128	139,199
19	null	null	1,594,128	139,199
20	null	Phones	179,957	24,297
21	null	Tables	86,263	-1,572
22	null	Binders	161,686	11,883

Query Details
Query duration 1.4s
Rows 36
Query ID 01ac1224-0001-9d8a-0...
YEAR_OF_DATE 50% filled 50% null
PRODUCT_SUBCATEGORY
Phones 2
Tables 2

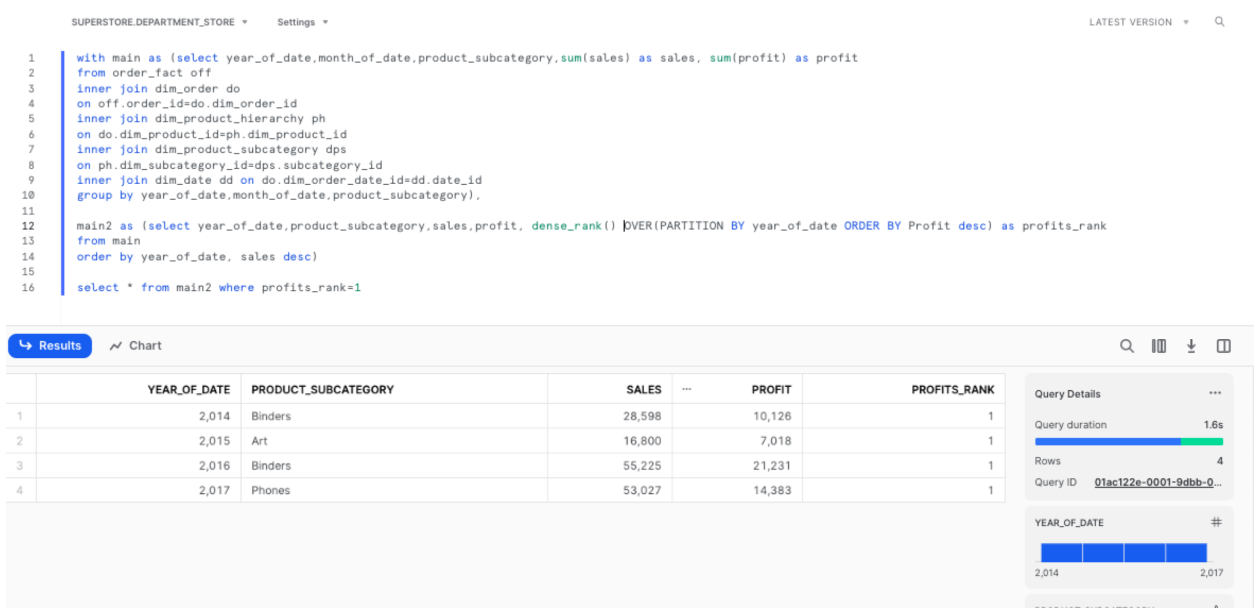
- From the above data, we can track year on year growth of a particular product subcategory, based on the revenue and profit trends we can decide what actions can be taken for that sub category.

3. Most Profitable sub category across years, to bring in new customers and drive new leads marketing.

with main as (select year_of_date,month_of_date,product_subcategory,sum(sales) as sales, sum(profit) as profit
from order_fact off
inner join dim_order do
on off.order_id=do.dim_order_id
inner join dim_product_hierarchy ph
on do.dim_product_id=ph.dim_product_id
inner join dim_product_subcategory dps
on ph.dim_subcategory_id=dps.subcategory_id
inner join dim_date dd on do.dim_order_date_id=dd.date_id
group by year_of_date,month_of_date,product_subcategory),

main2 as (select year_of_date,product_subcategory,sales,profit, dense_rank() OVER(PARTITION BY year_of_date ORDER BY Profit desc) as profits_rank
from main
order by year_of_date, sales desc)

select * from main2 where profits_rank=1



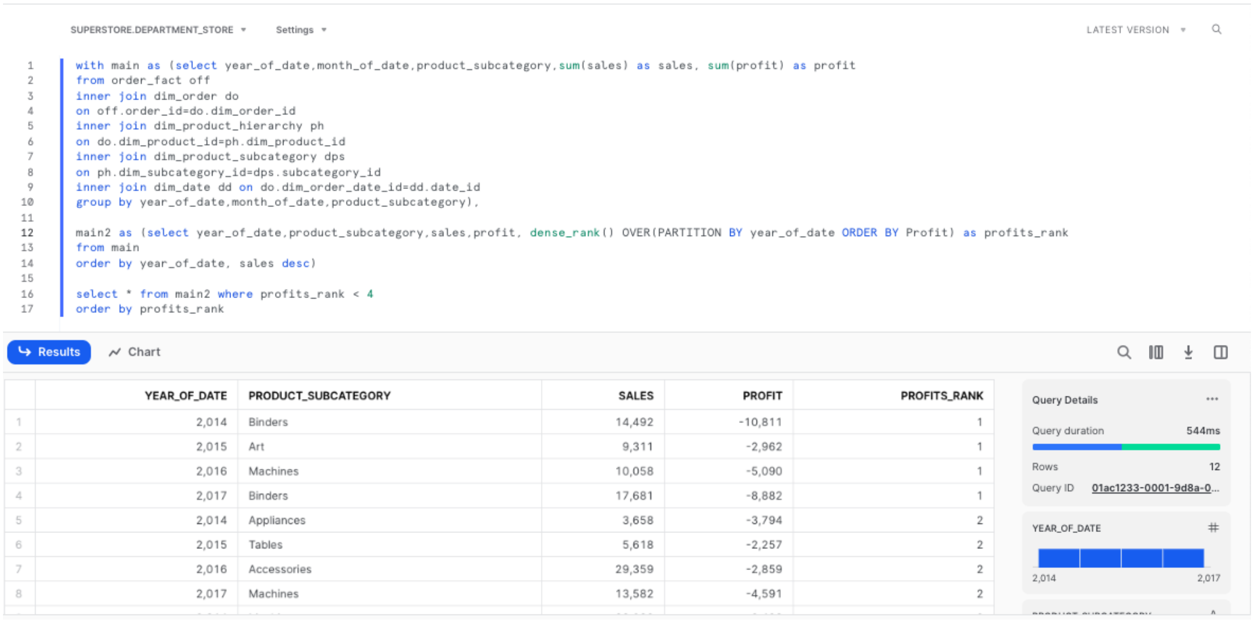
```
with main as (select year_of_date,month_of_date,product_subcategory,sum(sales) as sales, sum(profit) as profit
from order_fact off
inner join dim_order do
on off.order_id=do.dim_order_id
inner join dim_product_hierarchy ph
on do.dim_product_id=ph.dim_product_id
inner join dim_product_subcategory dps
on ph.dim_subcategory_id=dps.subcategory_id
inner join dim_date dd on do.dim_order_date_id=dd.date_id
group by year_of_date,month_of_date,product_subcategory),

main2 as (select year_of_date,product_subcategory,sales,profit, dense_rank() OVER(PARTITION BY year_of_date ORDER BY Profit desc) as profits_rank
from main
order by year_of_date, sales desc)

select * from main2 where profits_rank=1
```

	YEAR_OF_DATE	PRODUCT_SUBCATEGORY	SALES	PROFIT	PROFITS_RANK
1	2,014	Binders	28,598	10,126	1
2	2,015	Art	16,800	7,018	1
3	2,016	Binders	55,225	21,231	1
4	2,017	Phones	53,027	14,383	1

Query Details
Query duration 1.6s
Rows 4
Query ID 01ac122e-0001-9dbb-0...

The above information can turn out to be the information which most of the executives need in top level management for various brand boosting activities.

4. Understanding the worst 3 loss making subcategories across years?



```sql
with main as (select year_of_date,month_of_date,product_subcategory,sum(sales) as sales, sum(profit) as profit
from order_fact off
inner join dim_order do
on off.order_id=do.dim_order_id
inner join dim_product_hierarchy ph
on do.dim_product_id=ph.dim_product_id
inner join dim_product_subcategory dps
on ph.dim_subcategory_id=dps.subcategory_id
inner join dim_date dd on do.dim_order_date_id=dd.date_id
group by year_of_date,month_of_date,product_subcategory),

main2 as (select year_of_date,product_subcategory,sales,profit, dense_rank() OVER(PARTITION BY year_of_date ORDER BY Profit) as profits_rank
from main
order by year_of_date, sales desc)

select * from main2 where profits_rank < 4
order by profits_rank
```

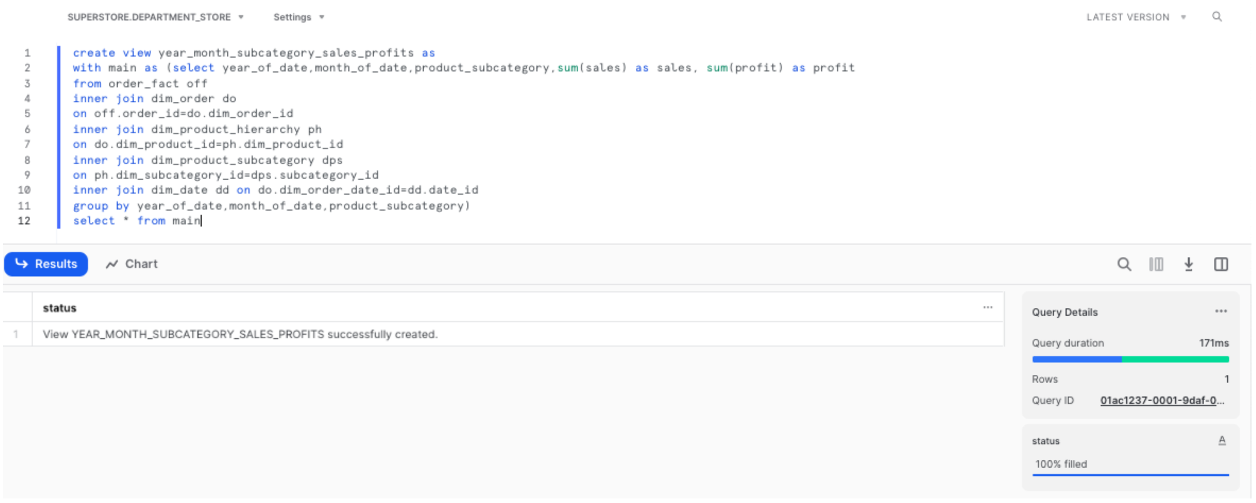
	YEAR_OF_DATE	PRODUCT_SUBCATEGORY	SALES	PROFIT	PROFITS_RANK
1	2,014	Binders	14,492	-10,811	1
2	2,015	Art	9,311	-2,962	1
3	2,016	Machines	10,058	-5,090	1
4	2,017	Binders	17,681	-8,882	1
5	2,014	Appliances	3,658	-3,794	2
6	2,015	Tables	5,618	-2,257	2
7	2,016	Accessories	29,359	-2,859	2
8	2,017	Machines	13,582	-4,591	2

Query Details

Query duration	544ms
Rows	12
Query ID	01ac1233-0001-9d8a-0...

From the above query, we can simply point out the worst loss making subcategories across years, and how this trend has changed, we can see the sudden huge loss in 2017 for Binders subcategory.

I have also tried to materialize a view here, because of all the dimension tables we have. I shortlisted the above used big subquery as the one which I need to materialize, and did it using the create view statement;



```sql
create view year_month_subcategory_sales_profits as
with main as (select year_of_date,month_of_date,product_subcategory,sum(sales) as sales, sum(profit) as profit
from order_fact off
inner join dim_order do
on off.order_id=do.dim_order_id
inner join dim_product_hierarchy ph
on do.dim_product_id=ph.dim_product_id
inner join dim_product_subcategory dps
on ph.dim_subcategory_id=dps.subcategory_id
inner join dim_date dd on do.dim_order_date_id=dd.date_id
group by year_of_date,month_of_date,product_subcategory)
select * from main
```

	status
1	View YEAR_MONTH_SUBCATEGORY_SALES_PROFITS successfully created.

Query Details

Query duration	171ms
Rows	1
Query ID	01ac1237-0001-9daf-0...

```
create view year_month_subcategory_sales_profits as
with main as (select year_of_date,month_of_date,product_subcategory,sum(sales) as sales,
sum(profit) as profit
from order_fact off
inner join dim_order do
on off.order_id=do.dim_order_id
inner join dim_product_hierarchy ph
on do.dim_product_id=ph.dim_product_id
inner join dim_product_subcategory dps
on ph.dim_subcategory_id=dps.subcategory_id
inner join dim_date dd on do.dim_order_date_id=dd.date_id
group by year_of_date,month_of_date,product_subcategory)
select * from main
```

We can similarly create a lot of highly used views to make the data retrieval a little complex in Snowflake, as the dimensions keep increasing the complexity (number of joins) will also increase, which will make it more confusing and complex.

1 (b) Blog to write down the important findings from the above mini-project which I implemented in this homework with the help of Sample Store dataset.

Link:

https://medium.com/@nikhil.thota_81762/exploration-of-snowflake-schema-by-implementing-samplestore-dataset-analytic-functions-dbc0106273a7