Mini project using Snowflake.

Title: Library Management System Analytic

Objective: To analyze and visualize data from a library management system using Snowflake Data Platform, incorporating multiple fact and dimension tables for comprehensive insights.

Schema: Star schema with multiple fact and dimension tables:
Fact Tables: This table stores quantitative data like book checkouts and monetary transactions related to fines associated with checkouts.
checkoutsfact (checkoutid, bookid, borrowerid, checkoutdate, returndate)
finepaymentsfact (paymentid, checkoutid, paymentdate, amount)

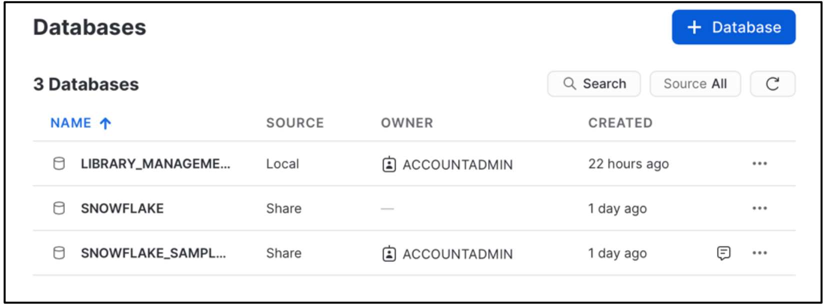
Dimension Tables: Contain descriptive attributes.
bookdim (bookid, title, author, genre, publishyear)
borrowerdim (bor_index, borrowerid, name, address)
librarybranchdim (branchid, branchname, location)
datedim (dateid, day, month, year, weekday)

The checkoutsfact table has relationships with the bookdim, borrowerdim, librarybranchdim, and datedim tables. It links to bookdim through bookid, to borrowerdim through borrowerid, to librarybranchdim through branchid, and to datedim through checkoutdate.

The finepaymentsfact table has a relationship with the checkoutsfact table through checkoutid, and it indirectly links to other dimension tables through its relationship with checkoutsfact.
Each dimension table (bookdim, borrowerdim, librarybranchdim, datedim) serves as a hub for details related to books, borrowers, library branches, and dates, respectively.

Process:

1. Logged into snowflake and created new database. Used below command to use the new database,
   USE DATABASE library_management;



Databases

3 Databases

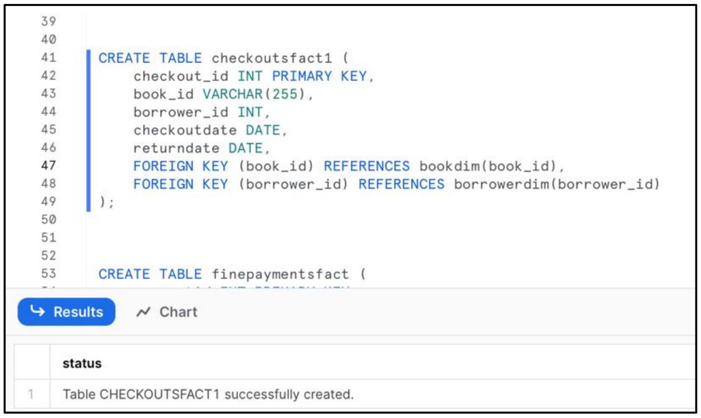
NAME ↑	SOURCE	OWNER	CREATED
LIBRARY_MANAGEME...	Local	ACCOUNTADMIN	22 hours ago
SNOWFLAKE	Share	—	1 day ago
SNOWFLAKE_SAMPL...	Share	ACCOUNTADMIN	1 day ago

2. Executed CREATE statement in Snowflake SQL worksheet:

```
CREATE TABLE bookdim (
   book_id VARCHAR(255) PRIMARY KEY,
   book_title VARCHAR(255),
    book_author VARCHAR(255),
    year_of_publication INT,
    publisher VARCHAR(225)
);


CREATE TABLE borrowerdim1 (
    bor_index INT PRIMARY KEY,
    borrower_name VARCHAR(255),
    borrower_area VARCHAR(255),
    borrower_id INT
);

CREATE TABLE librarybranchdim (
    branch_id INT PRIMARY KEY,
    branchname VARCHAR(255),
    branch_address VARCHAR(255),
    branch_city VARCHAR(255),
    branch_phone VARCHAR(255),
    branch_zip INT
);

CREATE TABLE datedim (
    dateid INT PRIMARY KEY,
    day INT,
    month INT,
    year INT,
    weekday VARCHAR(20)
);
```

Snowflake stores DATE data more efficiently than VARCHAR, providing better query performance. Hence, I have used DATE datatype with format (ISO YYYY-MM-DD)

```
CREATE TABLE checkoutsfact (
    checkout_id INT PRIMARY KEY,
    book_id VARCHAR(255),
    borrower_id INT,
    checkoutdate DATE,
    returndate DATE,
    FOREIGN KEY (book_id) REFERENCES bookdim(book_id),
    FOREIGN KEY (borrower_id) REFERENCES borrowerdim(borrower_id)
);


CREATE TABLE finepaymentsfact (
    paymentid INT PRIMARY KEY,
    checkoutid INT,
    paymentdate DATE,
    amount DECIMAL(10, 2),
    FOREIGN KEY (checkoutid) REFERENCES checkoutsfact(checkoutid)
);
```

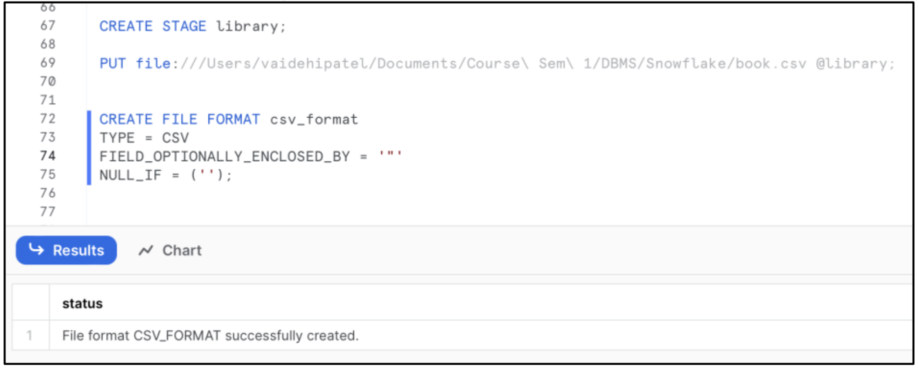


```
CREATE TABLE checkoutsfact1 (
    checkout_id INT PRIMARY KEY,
    book_id VARCHAR(255),
    borrower_id INT,
    checkoutdate DATE,
    returndate DATE,
    FOREIGN KEY (book_id) REFERENCES bookdim(book_id),
    FOREIGN KEY (borrower_id) REFERENCES borrowerdim(borrower_id)
);


CREATE TABLE finepaymentsfact (
```

	status
1	Table CHECKOUTSFACT1 successfully created.

3. Loaded data from csv to tables in snowflake.

- Created a stage using below command,
  CREATE STAGE library;
- Files are first copied ("staged") to an internal (Snowflake) stage, then loaded into a table.
- Uploaded CSV file from local machine to snowflake using worksheet query-
  ***PUT** file:///Users/vaidehipatel/Documents/Course\ Sem\ 1/DBMS/Snowflake/books.csv @library;*
- Executed COPY query to load data from csv to tables in snowflake.



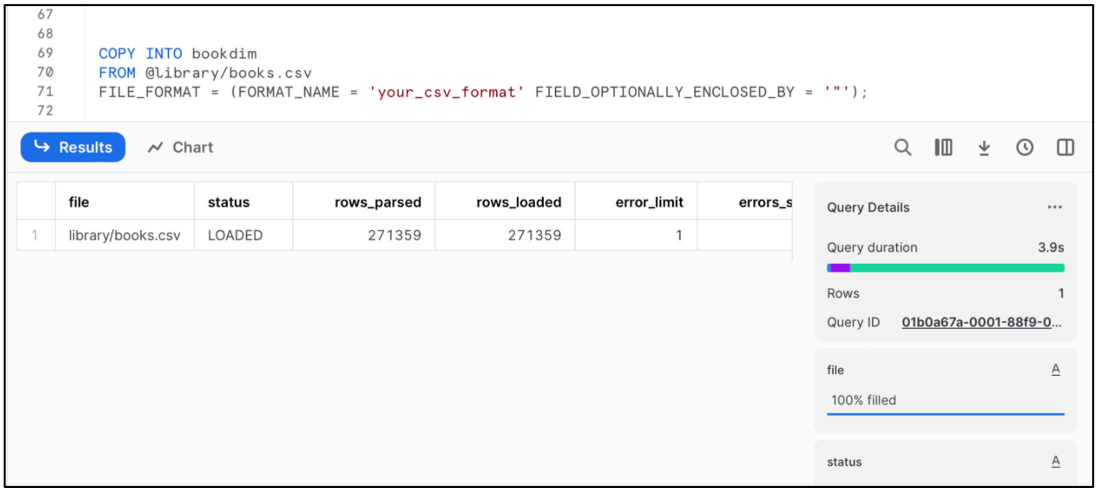
```
CREATE STAGE library;

PUT file:///Users/vaidehipatel/Documents/Course\ Sem\ 1/DBMS/Snowflake/book.csv @library;


CREATE FILE FORMAT csv_format
TYPE = CSV
FIELD_OPTIONALLY_ENCLOSED_BY = '"'
NULL_IF = ('');
```

	status
1	File format CSV_FORMAT successfully created.

Leveraged Snowflake's COPY command for efficient bulk loading.

```
CREATE FILE FORMAT csv_format
TYPE = CSV
FIELD_OPTIONALLY_ENCLOSED_BY = '"'
NULL_IF = (");
```

```
COPY INTO bookdim
FROM @library/books.csv
FILE_FORMAT = (FORMAT_NAME = 'csv_format'
FIELD_OPTIONALLY_ENCLOSED_BY = '"');

COPY INTO borrowerdim
FROM @library/borrower_data.csv
FILE_FORMAT = (FORMAT_NAME = 'csv_format'
FIELD_OPTIONALLY_ENCLOSED_BY = '"');

COPY INTO librarybranchdim
FROM @library/queens_library_branches.csv
FILE_FORMAT = (FORMAT_NAME = 'csv_format'
FIELD_OPTIONALLY_ENCLOSED_BY = '"');

COPY INTO checkoutsfact
FROM @library/checkouts_data.csv
FILE_FORMAT = (FORMAT_NAME = 'csv_format' DATE_FORMAT = 'YYYY-MM-DD');
```



```
COPY INTO bookdim
FROM @library/books.csv
FILE_FORMAT = (FORMAT_NAME = 'your_csv_format' FIELD_OPTIONALLY_ENCLOSED_BY = '"');
```

	file	status	rows_parsed	rows_loaded	error_limit	errors_s
1	library/books.csv	LOADED	271359	271359	1	

Query Details

Query duration 3.9s

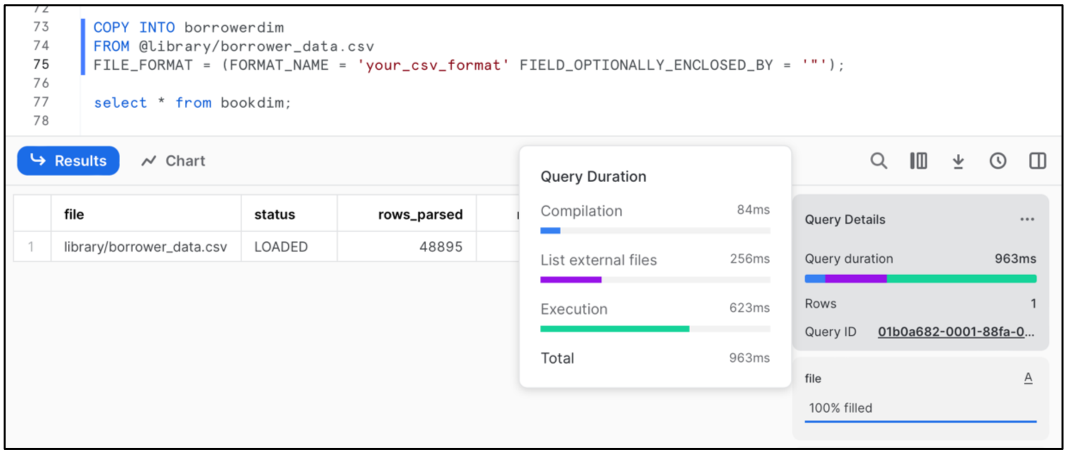
Rows 1

Query ID 01b0a67a-0001-88f9-0...

file

100% filled

status



```
COPY INTO borrowerdim
FROM @library/borrower_data.csv
FILE_FORMAT = (FORMAT_NAME = 'your_csv_format' FIELD_OPTIONALLY_ENCLOSED_BY = '"');

select * from bookdim;
```

	file	status	rows_parsed
1	library/borrower_data.csv	LOADED	48895

Query Duration

Compilation 84ms

List external files 256ms

Execution 623ms

Total 963ms

Query Details

Query duration 963ms

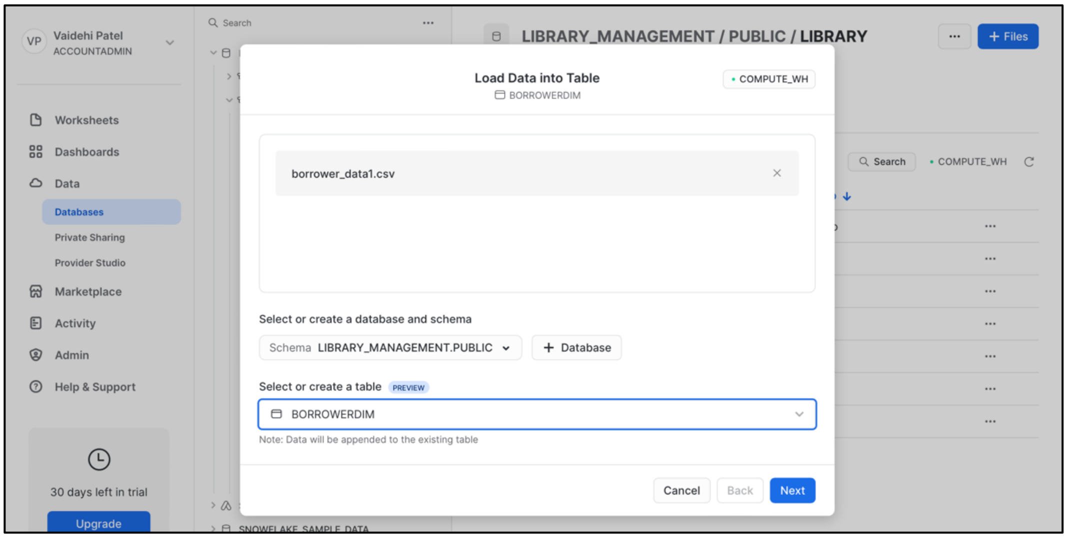
Rows 1

Query ID 01b0a682-0001-88fa-0...

file

100% filled

Loaded other tables using ‘Load a file’ button from Stage into an existing table using Snowsight.

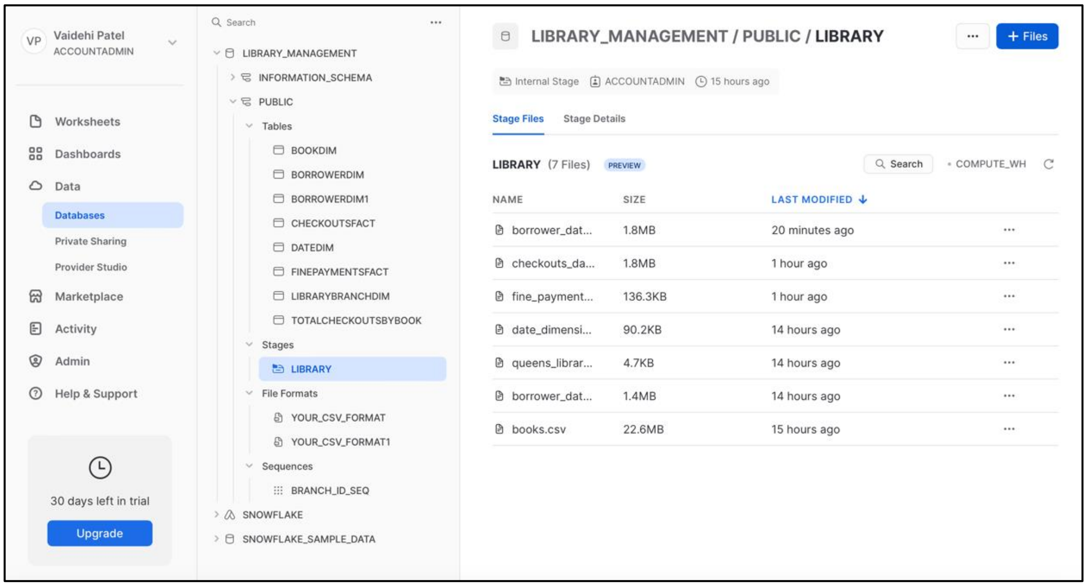


Load Data into Table
BORROWERDIM
COMPUTE_WH

borrower_data1.csv

Select or create a database and schema
Schema LIBRARY_MANAGEMENT.PUBLIC
+ Database

Select or create a table PREVIEW
BORROWERDIM
Note: Data will be appended to the existing table



LIBRARY_MANAGEMENT / PUBLIC / LIBRARY

Internal Stage ACCOUNTADMIN 15 hours ago

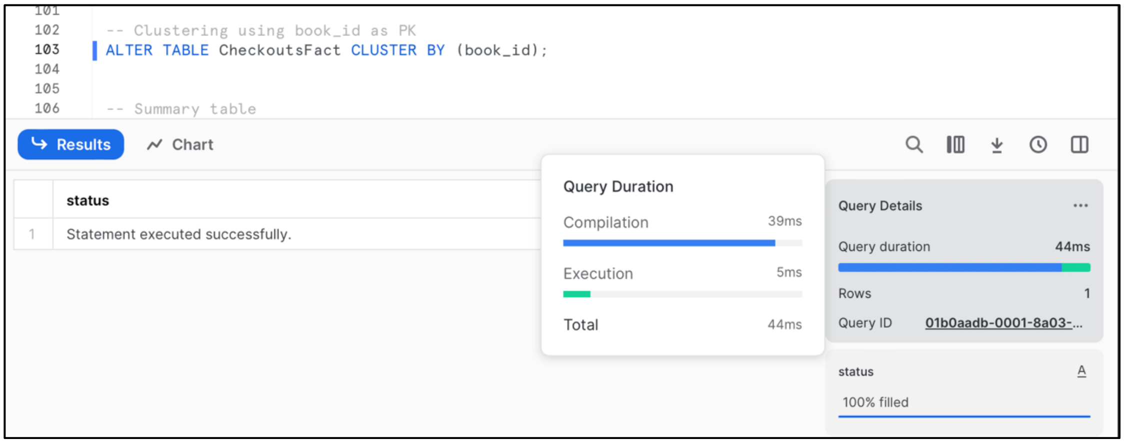
LIBRARY (7 Files) PREVIEW

NAME	SIZE	LAST MODIFIED
borrower_dat...	1.8MB	20 minutes ago
checkouts_da...	1.8MB	1 hour ago
fine_payment...	136.3KB	1 hour ago
date_dimensi...	90.2KB	14 hours ago
queens_librar...	4.7KB	14 hours ago
borrower_dat...	1.4MB	14 hours ago
books.csv	22.6MB	15 hours ago

4. Clustering

Leveraged Snowflake's automatic clustering to organize the data based on book_id key. This improves query performance by physically storing related data together.

```
ALTER TABLE CheckoutsFact CLUSTER BY (book_id);
```



```
-- Clustering using book_id as PK
ALTER TABLE CheckoutsFact CLUSTER BY (book_id);


-- Summary table
```

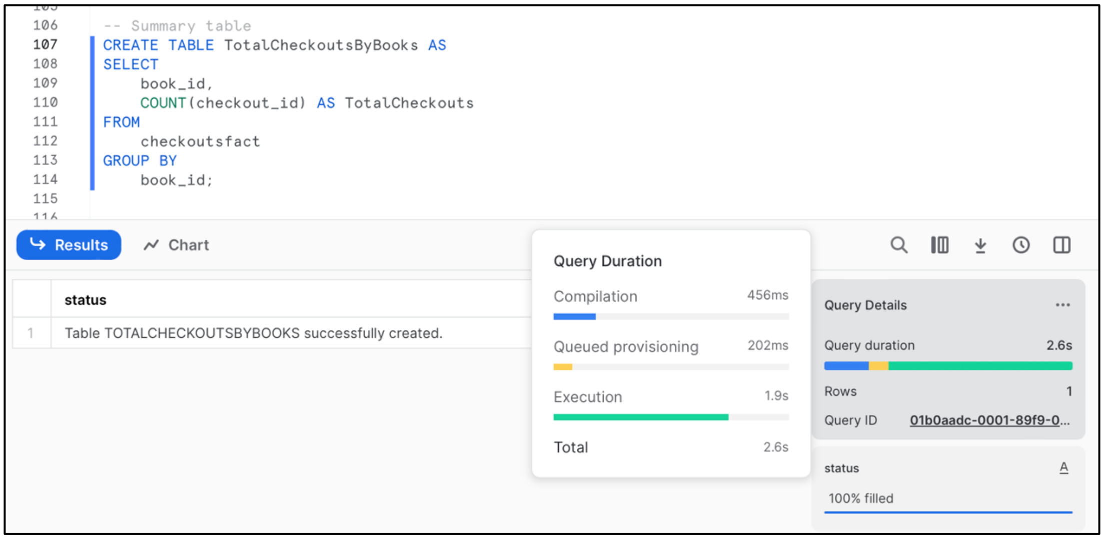
	status
1	Statement executed successfully.

```
Query Duration
Compilation          39ms
Execution             5ms
Total                44ms

Query Details
Query duration       44ms
Rows                    1
Query ID   01b0aadb-0001-8a03-...

status
100% filled
```

5. Summary table

Used to store pre-aggregated results.

```
CREATE TABLE TotalCheckoutsByBook AS
SELECT
   book_id,
   COUNT(checkout_id) AS TotalCheckouts
FROM
   checkoutsfact
GROUP BY
   book_id;
```



```
-- Summary table
CREATE TABLE TotalCheckoutsByBooks AS
SELECT
    book_id,
    COUNT(checkout_id) AS TotalCheckouts
FROM
    checkoutsfact
GROUP BY
    book_id;
```

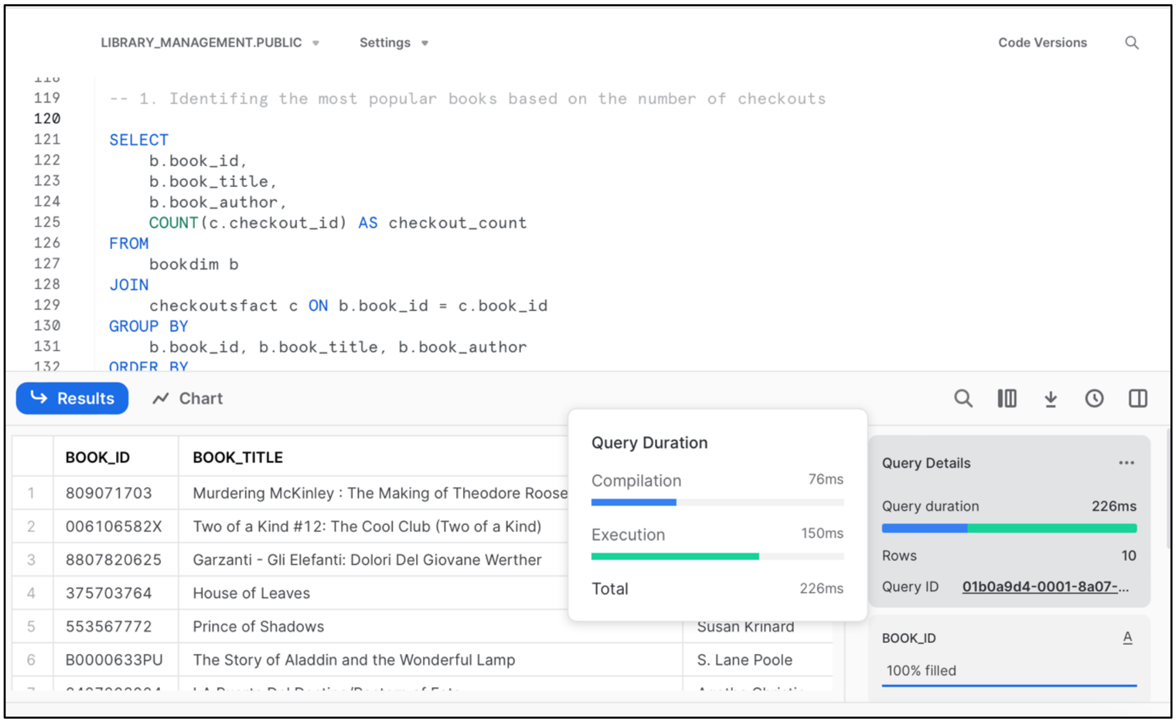
	status
1	Table TOTALCHECKOUTSBYBOOKS successfully created.

```
Query Duration
Compilation            456ms
Queued provisioning    202ms
Execution               1.9s
Total                   2.6s

Query Details
Query duration          2.6s
Rows                       1
Query ID   01b0aadc-0001-89f9-0...

status
100% filled
```

6. Analysis

a. Identifying the most popular books based on the number of checkouts.

```
SELECT
   b.book_id,
   b.book_title,
    b.book_author,
    COUNT(c.checkout_id) AS checkout_count
FROM
    bookdim b
JOIN
    checkoutsfact c ON b.book_id = c.book_id
GROUP BY
    b.book_id, b.book_title, b.book_author
ORDER BY
    checkout_count DESC
LIMIT 10;
```



```
-- 1. Identifing the most popular books based on the number of checkouts

SELECT
    b.book_id,
    b.book_title,
    b.book_author,
    COUNT(c.checkout_id) AS checkout_count
FROM
    bookdim b
JOIN
    checkoutsfact c ON b.book_id = c.book_id
GROUP BY
    b.book_id, b.book_title, b.book_author
ORDER BY
```

	BOOK_ID	BOOK_TITLE	
1	809071703	Murdering McKinley : The Making of Theodore Roose	
2	006106582X	Two of a Kind #12: The Cool Club (Two of a Kind)	
3	8807820625	Garzanti - Gli Elefanti: Dolori Del Giovane Werther	
4	375703764	House of Leaves	
5	553567772	Prince of Shadows	Susan Krinard
6	B0000633PU	The Story of Aladdin and the Wonderful Lamp	S. Lane Poole

Query Duration

Compilation 76ms

Execution 150ms

Total 226ms

Query Details

Query duration 226ms

Rows 10

Query ID 01b0a9d4-0001-8a07-...

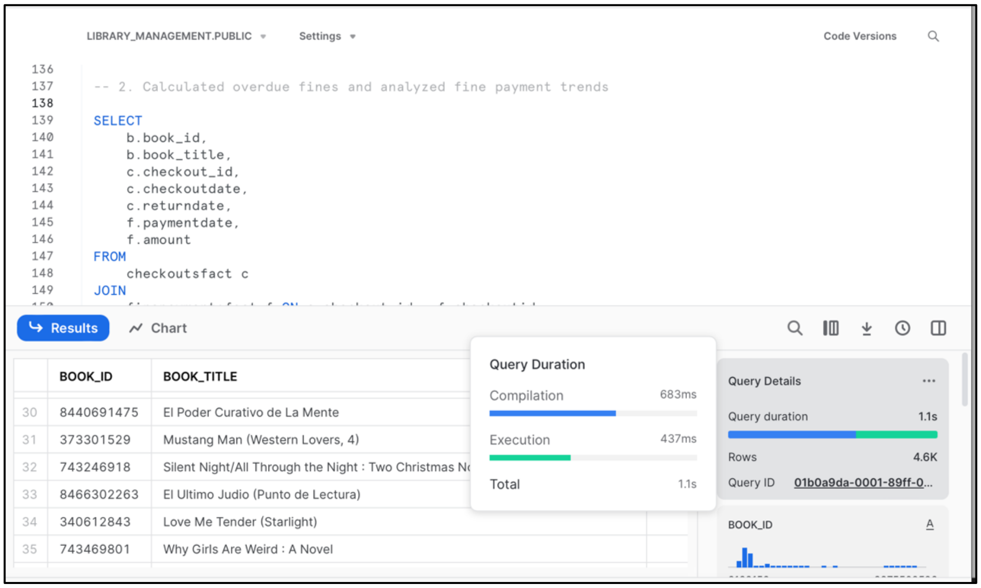
BOOK_ID

100% filled

b. Calculated overdue fines and analyzed fine payment trends.

```
SELECT
    b.book_id,
    b.book_title,
    c.checkout_id,
    c.checkoutdate,
    c.returndate,
    f.paymentdate,
    f.amount
FROM
    checkoutsfact c
JOIN
    finepaymentsfact f ON c.checkout_id = f.checkoutid
```

```
JOIN
    bookdim b ON c.book_id = b.book_id
WHERE
    f.paymentdate > c.returndate;
```



```
-- 2. Calculated overdue fines and analyzed fine payment trends

SELECT
    b.book_id,
    b.book_title,
    c.checkout_id,
    c.checkoutdate,
    c.returndate,
    f.paymentdate,
    f.amount
FROM
    checkoutsfact c
JOIN
```

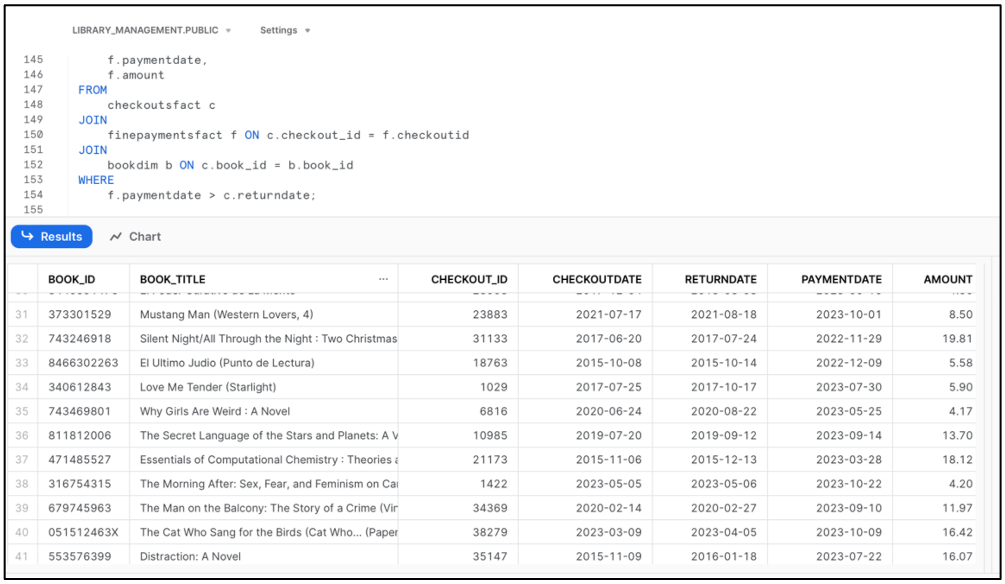
	BOOK_ID	BOOK_TITLE
30	8440691475	El Poder Curativo de La Mente
31	373301529	Mustang Man (Western Lovers, 4)
32	743246918	Silent Night/All Through the Night : Two Christmas N
33	8466302263	El Ultimo Judio (Punto de Lectura)
34	340612843	Love Me Tender (Starlight)
35	743469801	Why Girls Are Weird : A Novel

Query Duration

Compilation	683ms
Execution	437ms
Total	1.1s

Query Details

Query duration	1.1s
Rows	4.6K
Query ID	01b0a9da-0001-89ff-0...



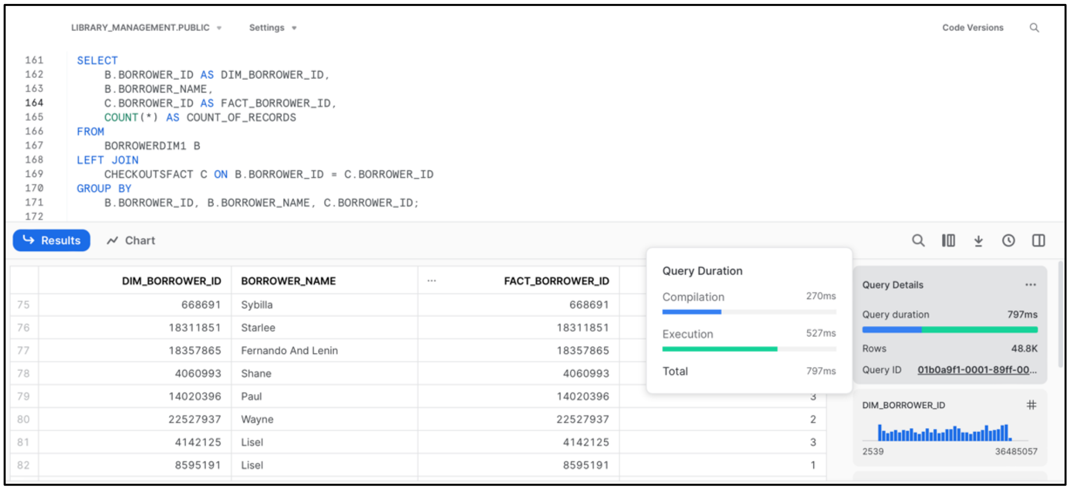
```
    f.paymentdate,
    f.amount
FROM
    checkoutsfact c
JOIN
    finepaymentsfact f ON c.checkout_id = f.checkoutid
JOIN
    bookdim b ON c.book_id = b.book_id
WHERE
    f.paymentdate > c.returndate;
```

	BOOK_ID	BOOK_TITLE	CHECKOUT_ID	CHECKOUTDATE	RETURNDATE	PAYMENTDATE	AMOUNT
31	373301529	Mustang Man (Western Lovers, 4)	23883	2021-07-17	2021-08-18	2023-10-01	8.50
32	743246918	Silent Night/All Through the Night : Two Christmas	31133	2017-06-20	2017-07-24	2022-11-29	19.81
33	8466302263	El Ultimo Judio (Punto de Lectura)	18763	2015-10-08	2015-10-14	2022-12-09	5.58
34	340612843	Love Me Tender (Starlight)	1029	2017-07-25	2017-10-17	2023-07-30	5.90
35	743469801	Why Girls Are Weird : A Novel	6816	2020-06-24	2020-08-22	2023-05-25	4.17
36	811812006	The Secret Language of the Stars and Planets: A V	10985	2019-07-20	2019-09-12	2023-09-14	13.70
37	471485527	Essentials of Computational Chemistry : Theories a	21173	2015-11-06	2015-12-13	2023-03-28	18.12
38	316754315	The Morning After: Sex, Fear, and Feminism on Ca	1422	2023-05-05	2023-05-06	2023-10-22	4.20
39	679745963	The Man on the Balcony: The Story of a Crime (Vir	34369	2020-02-14	2020-02-27	2023-09-10	11.97
40	051512463X	The Cat Who Sang for the Birds (Cat Who... (Paper	38279	2023-03-09	2023-04-05	2023-10-09	16.42
41	553576399	Distraction: A Novel	35147	2015-11-09	2016-01-18	2023-07-22	16.07

c. Number of times a User has borrowed books.
This query will provide a list of borrower IDs, borrower names, fact borrower IDs, and the count of records where there is a match between the BORROWER_ID values in the BORROWERDIM1 dimension table and the CHECKOUTSFACT fact table. It will be ordered by the borrower names.

```
SELECT
```

```
    B.BORROWER_ID AS DIM_BORROWER_ID,
    B.BORROWER_NAME,
    C.BORROWER_ID AS FACT_BORROWER_ID,
    COUNT(*) AS COUNT_OF_RECORDS
FROM
    BORROWERDIM1 B
LEFT JOIN
    CHECKOUTSFACT C ON B.BORROWER_ID = C.BORROWER_ID
GROUP BY
    B.BORROWER_ID, B.BORROWER_NAME, C.BORROWER_ID;
```



```
SELECT
    B.BORROWER_ID AS DIM_BORROWER_ID,
    B.BORROWER_NAME,
    C.BORROWER_ID AS FACT_BORROWER_ID,
    COUNT(*) AS COUNT_OF_RECORDS
FROM
    BORROWERDIM1 B
LEFT JOIN
    CHECKOUTSFACT C ON B.BORROWER_ID = C.BORROWER_ID
GROUP BY
    B.BORROWER_ID, B.BORROWER_NAME, C.BORROWER_ID;
```

	DIM_BORROWER_ID	BORROWER_NAME	... FACT_BORROWER_ID	
75	668691	Sybilla	668691	
76	18311851	Starlee	18311851	
77	18357865	Fernando And Lenin	18357865	
78	4060993	Shane	4060993	
79	14020396	Paul	14020396	3
80	22527937	Wayne	22527937	2
81	4142125	Lisel	4142125	3
82	8595191	Lisel	8595191	1

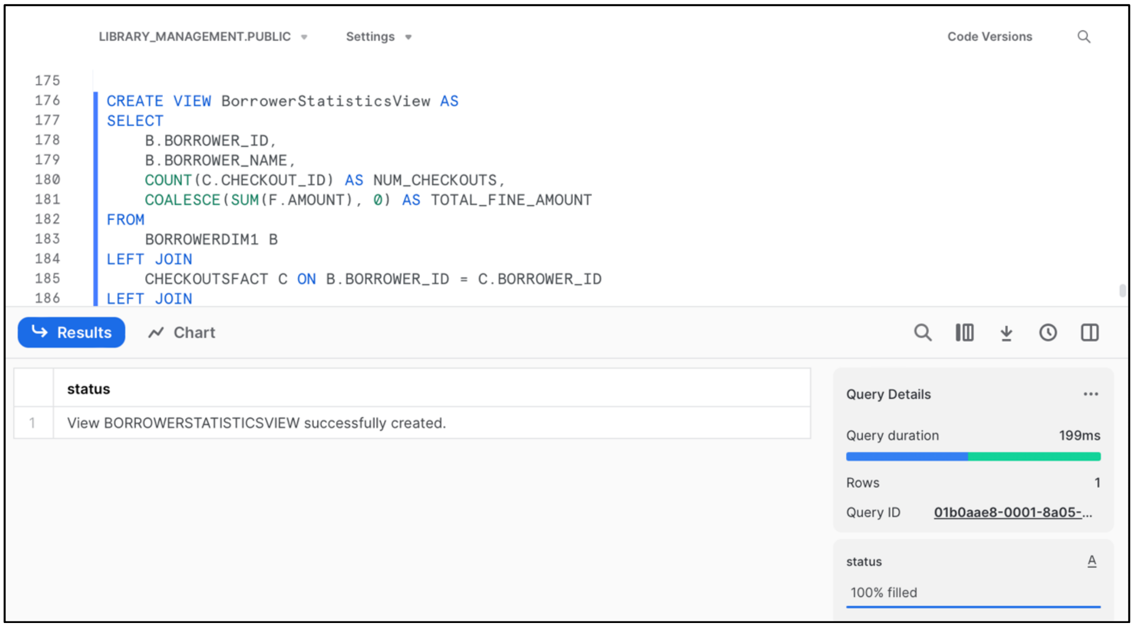
Query Duration
Compilation 270ms
Execution 527ms
Total 797ms

Query Details
Query duration 797ms
Rows 48.8K
Query ID 01b0a9f1-0001-89ff-00...

d. Views

This view aggregates data from the BORROWERDIM1, CHECKOUTSFACT, and FINEPAYMENTSFACT tables to provide borrower statistics, calculating the number of checkouts and total fine amounts for each borrower.

```
CREATE VIEW BorrowerStatisticsView AS
SELECT
    B.BORROWER_ID,
    B.BORROWER_NAME,
    COUNT(C.CHECKOUT_ID) AS NUM_CHECKOUTS,
    COALESCE(SUM(F.AMOUNT), 0) AS TOTAL_FINE_AMOUNT
FROM
    BORROWERDIM1 B
LEFT JOIN
    CHECKOUTSFACT C ON B.BORROWER_ID = C.BORROWER_ID
LEFT JOIN
    FINEPAYMENTSFACT F ON C.CHECKOUT_ID = F.CHECKOUTID
GROUP BY
    B.BORROWER_ID, B.BORROWER_NAME;
```



```sql
CREATE VIEW BorrowerStatisticsView AS
SELECT
    B.BORROWER_ID,
    B.BORROWER_NAME,
    COUNT(C.CHECKOUT_ID) AS NUM_CHECKOUTS,
    COALESCE(SUM(F.AMOUNT), 0) AS TOTAL_FINE_AMOUNT
FROM
    BORROWERDIM1 B
LEFT JOIN
    CHECKOUTSFACT C ON B.BORROWER_ID = C.BORROWER_ID
LEFT JOIN
```

	status
1	View BORROWERSTATISTICSVIEW successfully created.

Query Details

Query duration 199ms

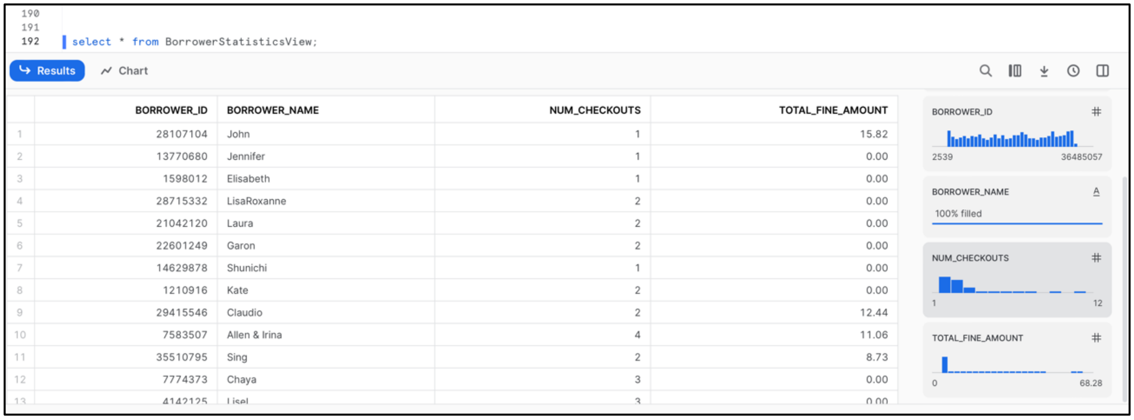
Rows 1

Query ID 01b0aae8-0001-8a05-...

status

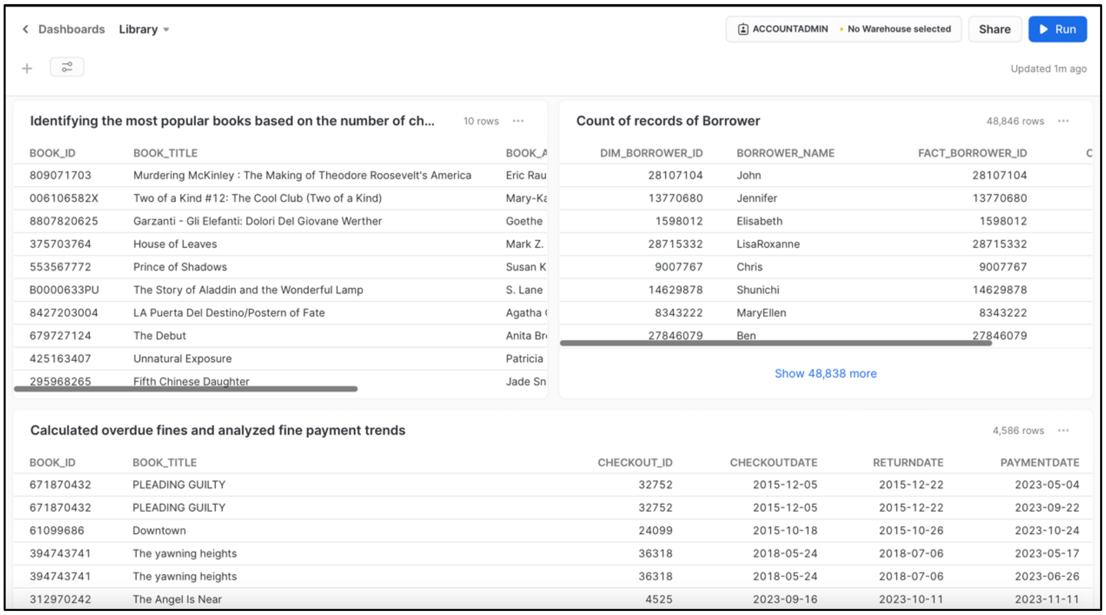
100% filled

SELECT * from BorrowerStatisticsView;


```sql
select * from BorrowerStatisticsView;
```

	BORROWER_ID	BORROWER_NAME	NUM_CHECKOUTS	TOTAL_FINE_AMOUNT
1	28107104	John	1	15.82
2	13770680	Jennifer	1	0.00
3	1598012	Elisabeth	1	0.00
4	28715332	LisaRoxanne	2	0.00
5	21042120	Laura	2	0.00
6	22601249	Garon	2	0.00
7	14629878	Shunichi	1	0.00
8	1210916	Kate	2	0.00
9	29415546	Claudio	2	12.44
10	7583507	Allen & Irina	4	11.06
11	35510795	Sing	2	8.73
12	7774373	Chaya	3	0.00
13	4142125	Lisel	3	0.00

<u>Dashboard</u>: Created a Dashboard from Existing Worksheets



Identifying the most popular books based on the number of ch... 10 rows

BOOK_ID	BOOK_TITLE	BOOK_A
809071703	Murdering McKinley : The Making of Theodore Roosevelt's America	Eric Rau
006106582X	Two of a Kind #12: The Cool Club (Two of a Kind)	Mary-Ka
8807820625	Garzanti - Gli Elefanti: Dolori Del Giovane Werther	Goethe
375703764	House of Leaves	Mark Z.
553567772	Prince of Shadows	Susan K
B0000633PU	The Story of Aladdin and the Wonderful Lamp	S. Lane
8427203004	LA Puerta Del Destino/Postern of Fate	Agatha
679727124	The Debut	Anita Br
425163407	Unnatural Exposure	Patricia
295968265	Fifth Chinese Daughter	Jade Sn

Count of records of Borrower 48,846 rows

DIM_BORROWER_ID	BORROWER_NAME	FACT_BORROWER_ID	C
28107104	John	28107104	
13770680	Jennifer	13770680	
1598012	Elisabeth	1598012	
28715332	LisaRoxanne	28715332	
9007767	Chris	9007767	
14629878	Shunichi	14629878	
8343222	MaryEllen	8343222	
27846079	Ben	27846079	

Show 48,838 more

Calculated overdue fines and analyzed fine payment trends 4,586 rows

BOOK_ID	BOOK_TITLE	CHECKOUT_ID	CHECKOUTDATE	RETURNDATE	PAYMENTDATE
671870432	PLEADING GUILTY	32752	2015-12-05	2015-12-22	2023-05-04
671870432	PLEADING GUILTY	32752	2015-12-05	2015-12-22	2023-09-22
61099686	Downtown	24099	2015-10-18	2015-10-26	2023-10-24
394743741	The yawning heights	36318	2018-05-24	2018-07-06	2023-05-17
394743741	The yawning heights	36318	2018-05-24	2018-07-06	2023-06-26
312970242	The Angel Is Near	4525	2023-09-16	2023-10-11	2023-11-11

Snowflake's automatic scaling capability dynamically adapted resources in response to varying workload demands, guaranteeing peak performance and efficiency. This led to accelerated query execution and a more streamlined data processing experience.

The robust performance, security measures, and collaborative features of the platform established a strong basis for thorough data analysis and visualization, effectively accomplishing the project's goals.

Takeaways from this mini- project:

1. **Continuous Loading Using Snowpipe**
   It loads small volumes of data and periodically make them available for analysis. The process swiftly loads data within minutes of adding files to a designated stage, guaranteeing users access to the most recent results as soon as the raw data becomes accessible.

2. **Materialized views** are crafted to enhance query efficiency for workloads characterized by recurring and typical query patterns.

3. **Interacting with Secure Views**
   The view definition of a secure view is restricted to authorized users who have been granted the corresponding role ownership. Secure views safeguard users from potential exposure to filtered table rows, yet careful construction is crucial to avoid inadvertent data exposure.

4. Snowflake functions on a **multi-cloud framework**, enabling users to utilize cloud services such as AWS, Azure, and Google Cloud.

5. The system **segregates storage and computing resources**, offering flexibility, scalability, and cost efficiency in the administration of data and analytics workloads.

6. Snowflake supports **unstructured data** as well as natively supports **semi-structured data** formats like JSON.

7. **Multi-cluster Warehouse** enable you to scale compute resources to manage your user and query concurrency needs as they change, such as during peak and off hours.

8. Snowflake supports both **scaling up** by resizing a warehouse and **scaling out** by adding clusters to the warehouse. Snowflake also provides **automatic scaling**, adjusting resources dynamically according to workload requirements to guarantee optimal performance and efficiency.

9. Snowflake supports **temporary** (for storing non-permanent data) and **transient tables** until explicitly dropped.

10. **Search access path** monitors the potential presence of table column values within individual micro-partitions, allowing for the possibility of skipping certain micro-partitions during the table scanning process.

Blog link:
https://medium.com/@vaidehi.patel_164/library-management-system-using-snowflake-data-platform-e73b94fe390b