FA23: DATA-225 Sec 11 - Db Systems for Analytics

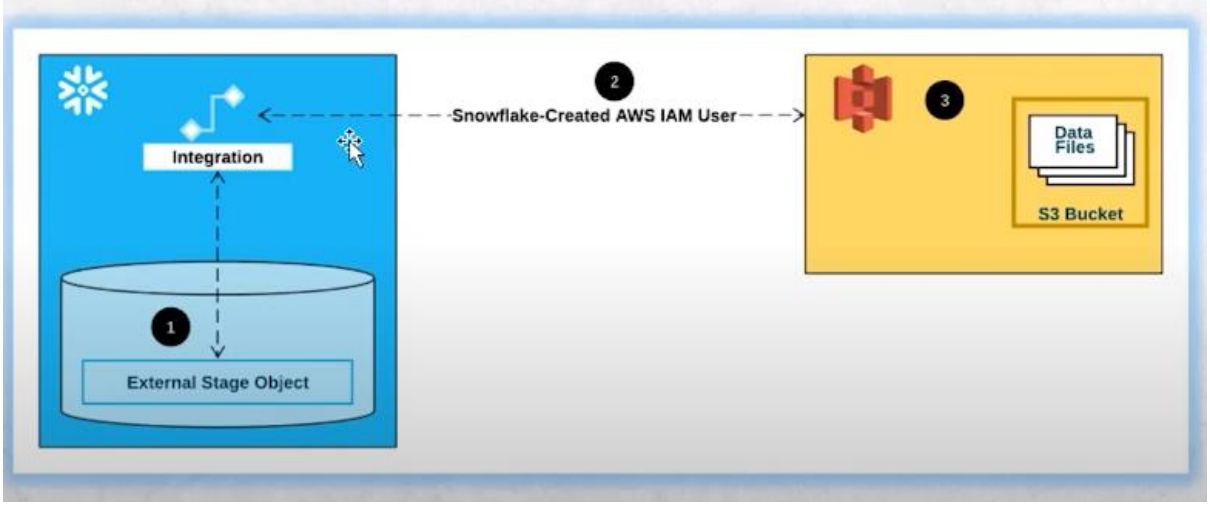
Homework - - 4

Name:- Prayag Nikul Purani

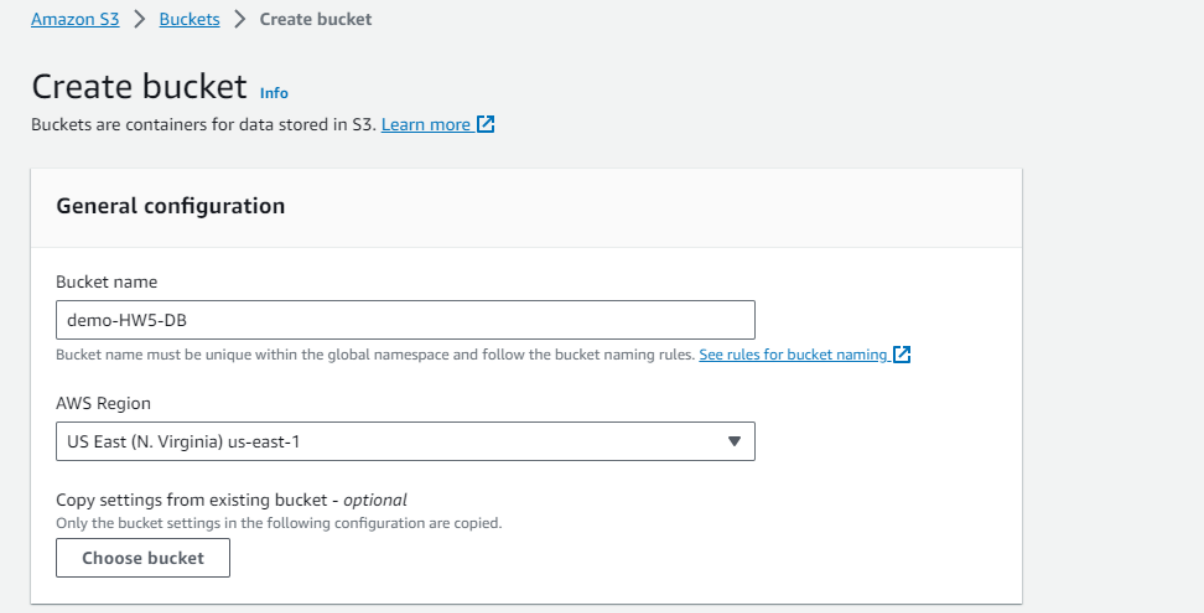
SJSU Id:- 017416737

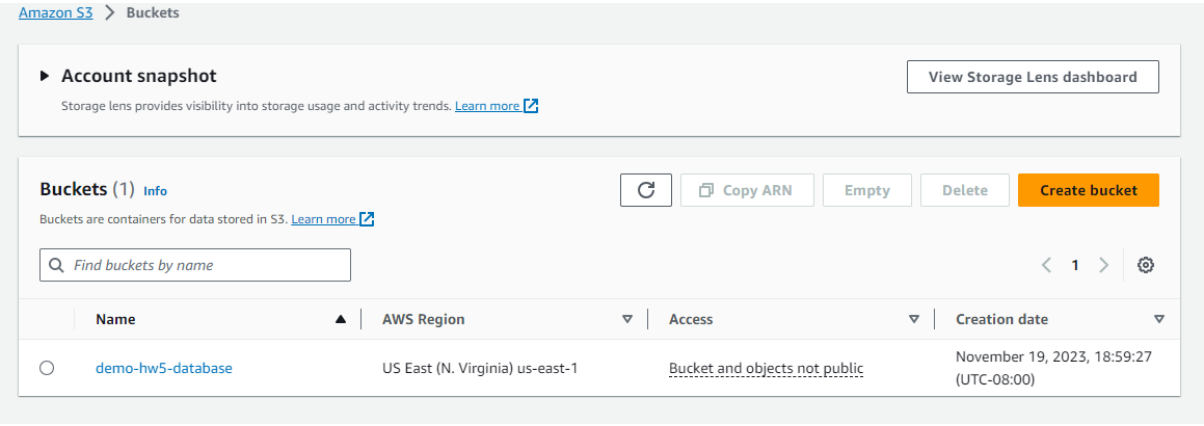
Question 1 a: Snowflakes data-ware house

The data-set is taken from the https://brightdata.com/cp/datasets/browse?id=hl_1876193f which real-estate data set of from Poland

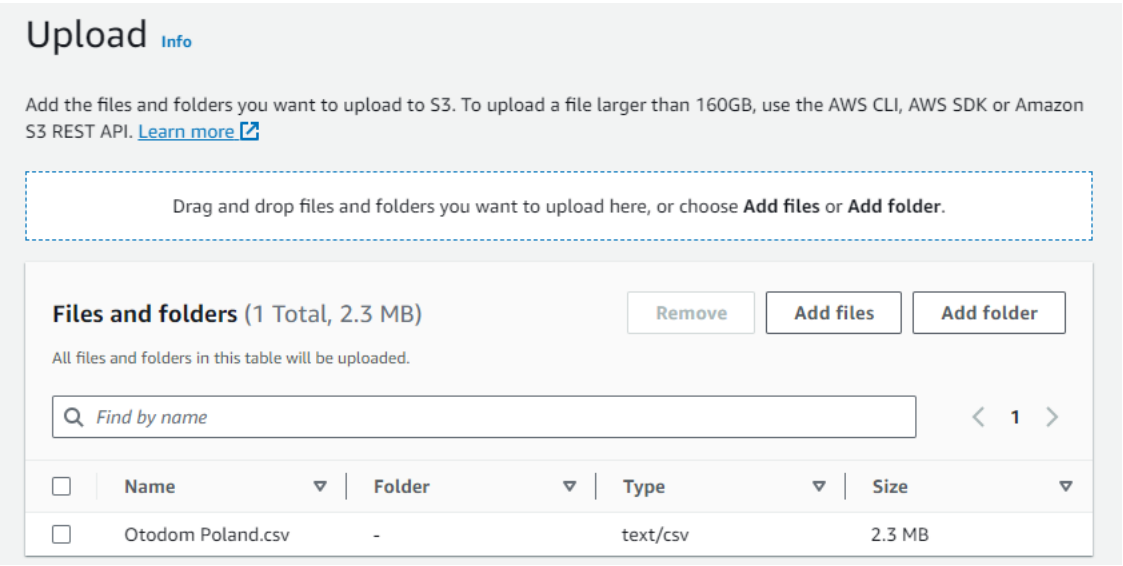


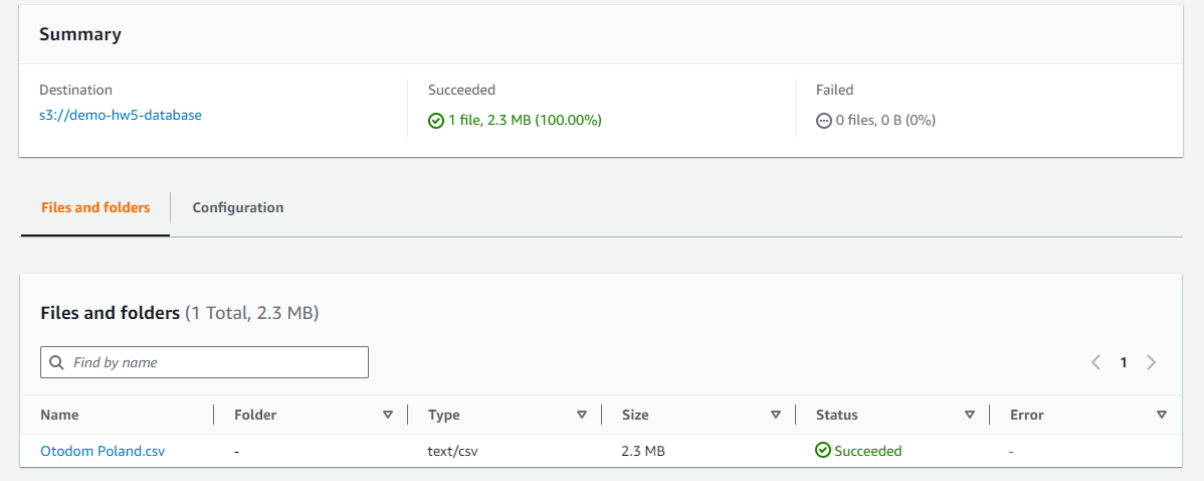
Step 1: Creating bucket in AWS S3

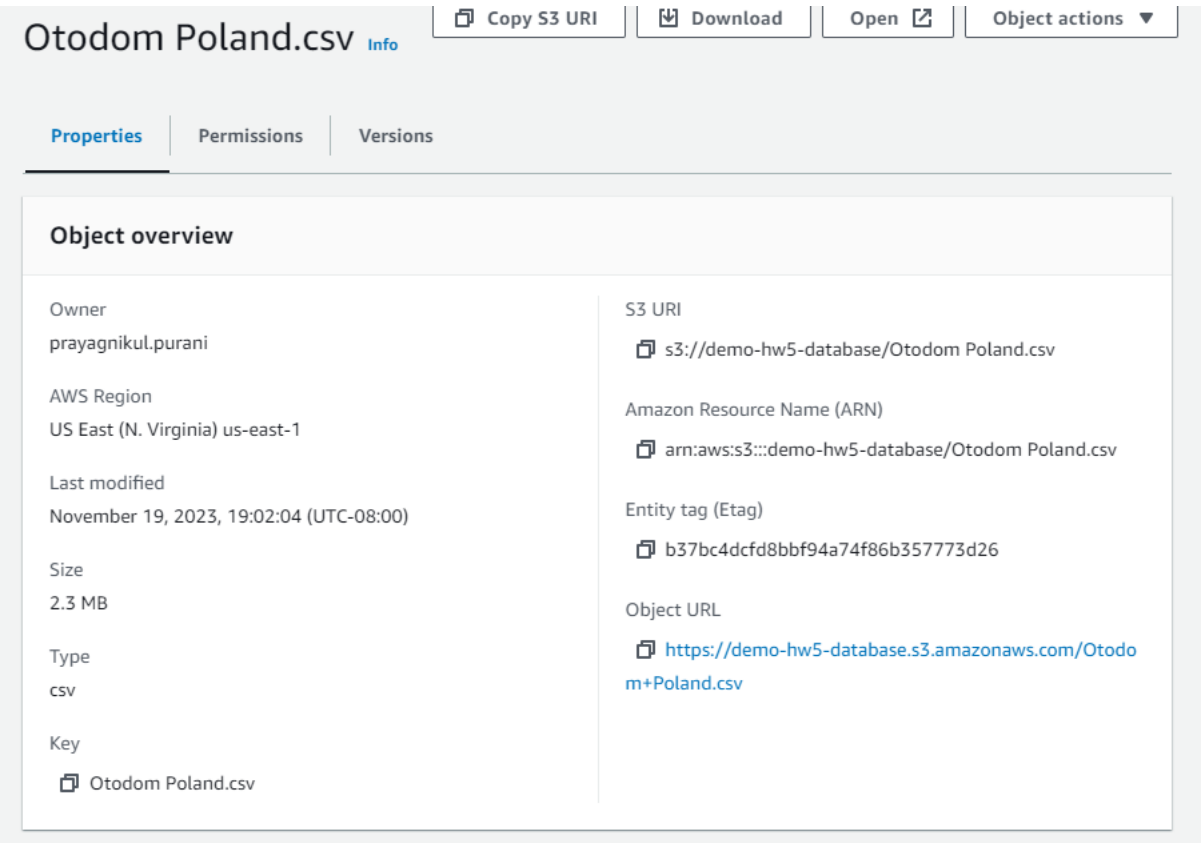




Step 2: Uploading the file







S3 Bucket Name: s3://demo-hw5-database/Otodom Poland.csv

Step 3: Creating a Role ARN

➔ **Polices**

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "Statement1",
                        "Effect": "Allow",
                        "Action": [
                                "s3:PutObject",
```

```
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::demo-hw5-database/*"
            ]
        },
        {
            "Sid": "Statement2",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::demo-hw5-database"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": "*"
                }
            }
        }
    ]
}
```

Specify permissions Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor Visual JSON Actions

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Sid": "Statement1",
6              "Effect": "Allow",
7              "Action": [
8                  "s3:PutObject",
9                  "s3:GetObject",
10                 "s3:GetObjectVersion",
11                 "s3:DeleteObject",
12                 "s3:DeleteObjectVersion"
13             ],
14             "Resource": [
15                 "arn:aws:s3:::demo-hw5-database/*"
16             ]
17         },
18         {
19             "Sid": "Statement2",
20             "Effect": "Allow",
21             "Action": [
22                 "s3:ListBucket",
23                 "s3:GetBucketLocation"
24             ],
25             "Resource": [
26                 "arn:aws:s3:::demo-hw5-database"
27             ],
28             "Condition": {
```

+ Add new statement

Edit statement Remove
Statement1

Add actions

Choose a service

s3

Included

S3

Available

S3 Object Lambda

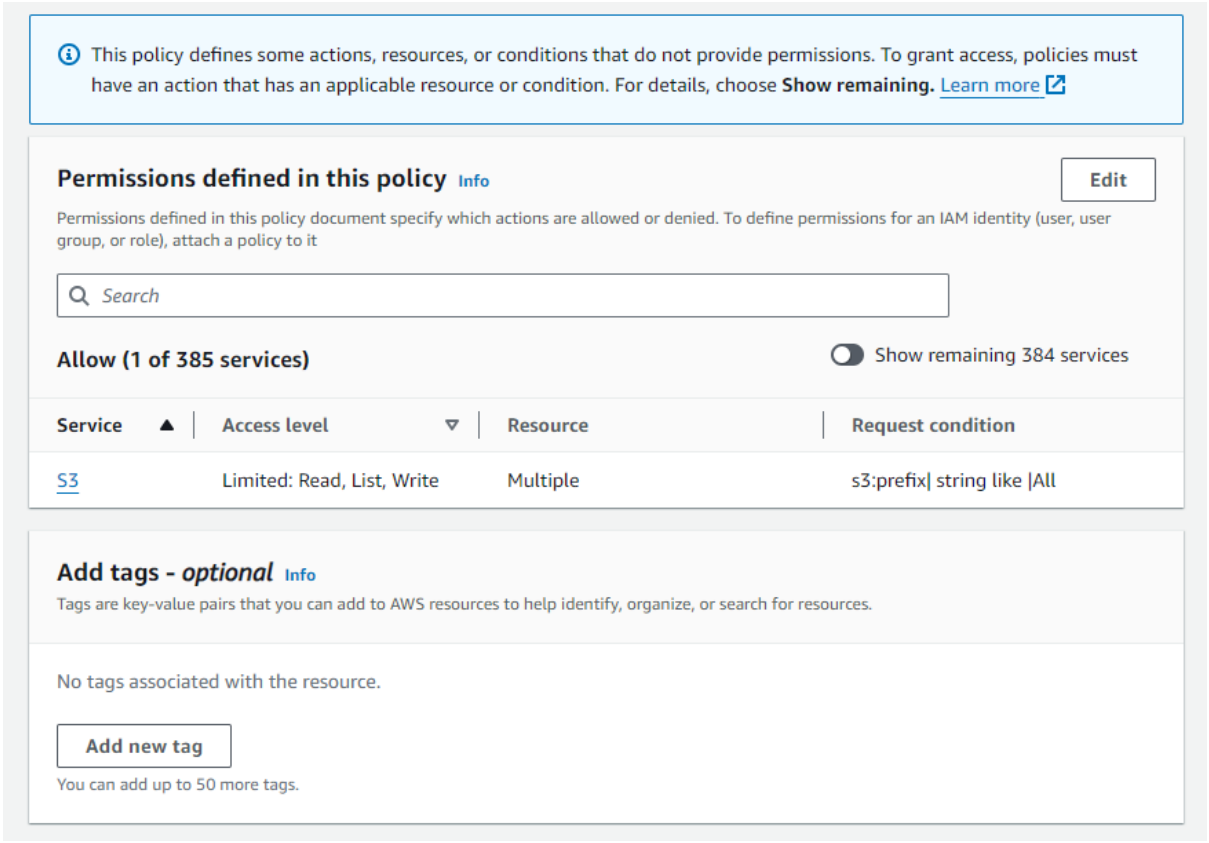
S3 Outposts

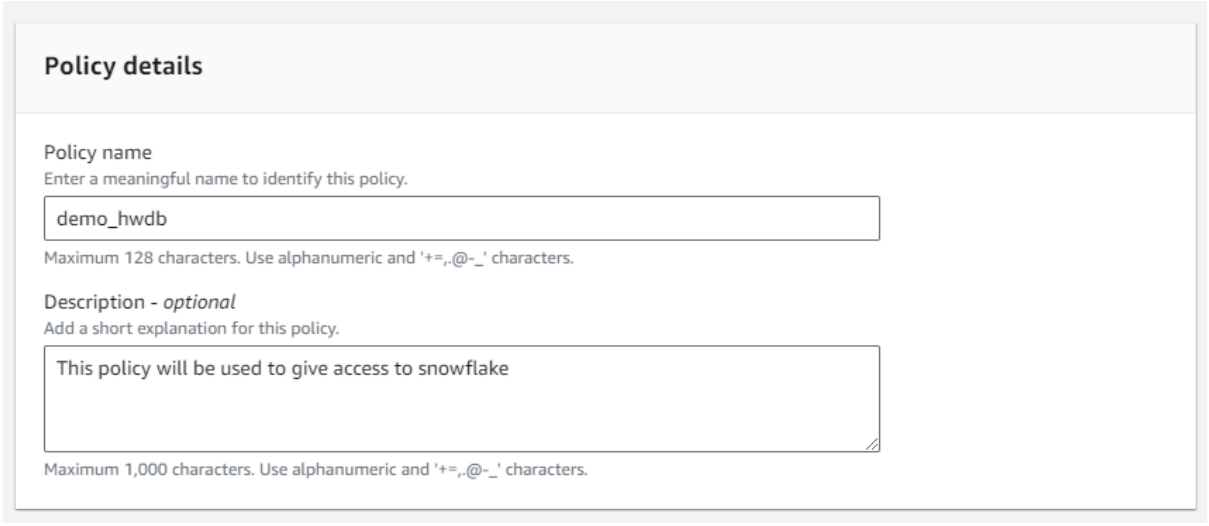
Add a resource Add

Add a condition (optional) Add

JSON Ln 14, Col 16 5735 of 6144 characters remaining

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0


This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose Show remaining. Learn more
Permissions defined in this policy Info
Edit
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it
Search
Allow (1 of 385 services)
Show remaining 384 services
Service
Access level
Resource
Request condition
S3
Limited: Read, List, Write
Multiple
s3:prefix| string like |All
Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.


Policy details
Policy name
Enter a meaningful name to identify this policy.
demo_hwdb
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.
Description - optional
Add a short explanation for this policy.
This policy will be used to give access to snowflake
Maximum 1,000 characters. Use alphanumeric and '+=,.@-_' characters.

➔ **Roles**

Select trusted entity Info

Trusted entity type

- AWS service
  Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account
  Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity
  Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation
  Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy
  Create a custom trust policy to enable others to perform actions in this account.

An AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

- This account (770342028587)
- Another AWS account

Options

- Require external ID (Best practice when a third party will assume this role)
  You can increase the security of your role by requiring an optional external identifier, which prevents "confused deputy" attacks. This is recommended if you do not own or have administrative access to the account that can assume this role. The external ID can include any characters that you choose. To assume this role, users must be in the trusted account and provide this exact external ID. Learn more

External ID

770342028587

Important: The console does not support using an external ID with the Switch Role feature. If you select this option, entities in the trusted account must use the API, CLI, or a custom federation proxy to make cross-account iam:AssumeRole calls. Learn more

- Require MFA
  Requires that the assuming entity use multi-factor authentication.

Add permissions Info

Permissions policies (1/890) Info

Choose one or more policies to attach to your new role.

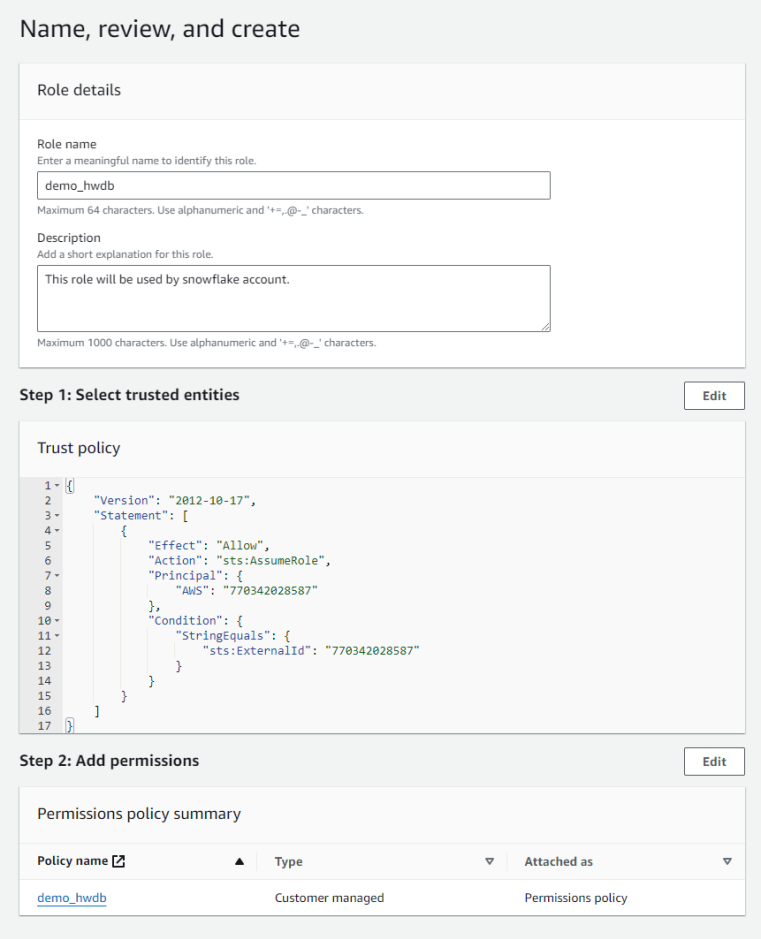
Filter by Type

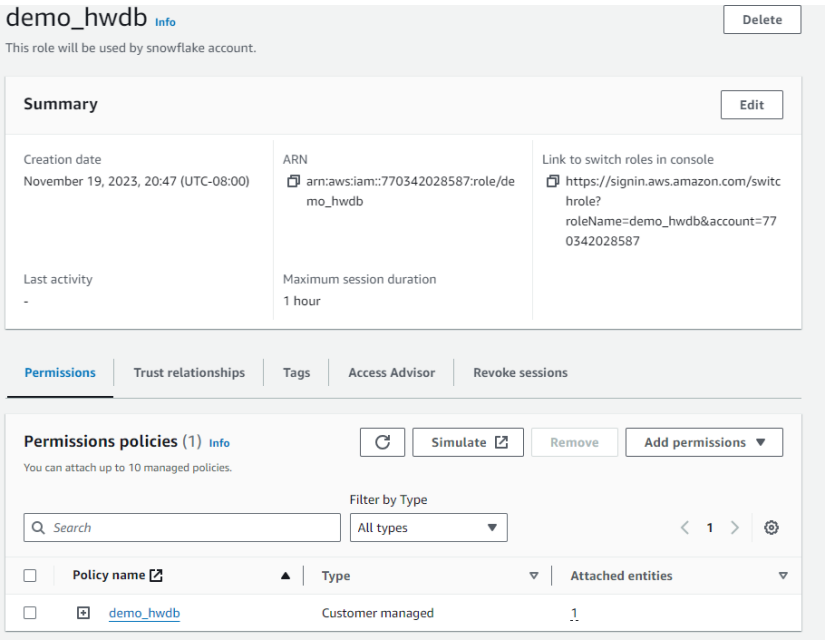
demo | All types | 1 match | 1

Policy name	Type	Description
demo_hwdb	Customer managed	This policy will be used to give access to ...

Set permissions boundary - *optional*

Cancel | Previous | Next



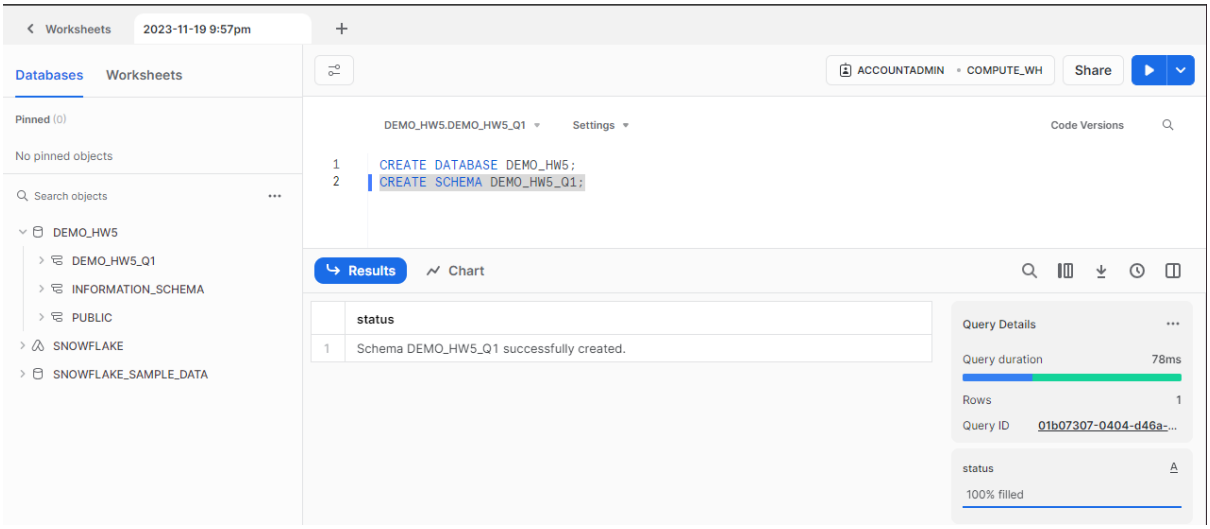


Role ARN: arn:aws:iam::770342028587:role/demo_hwdb

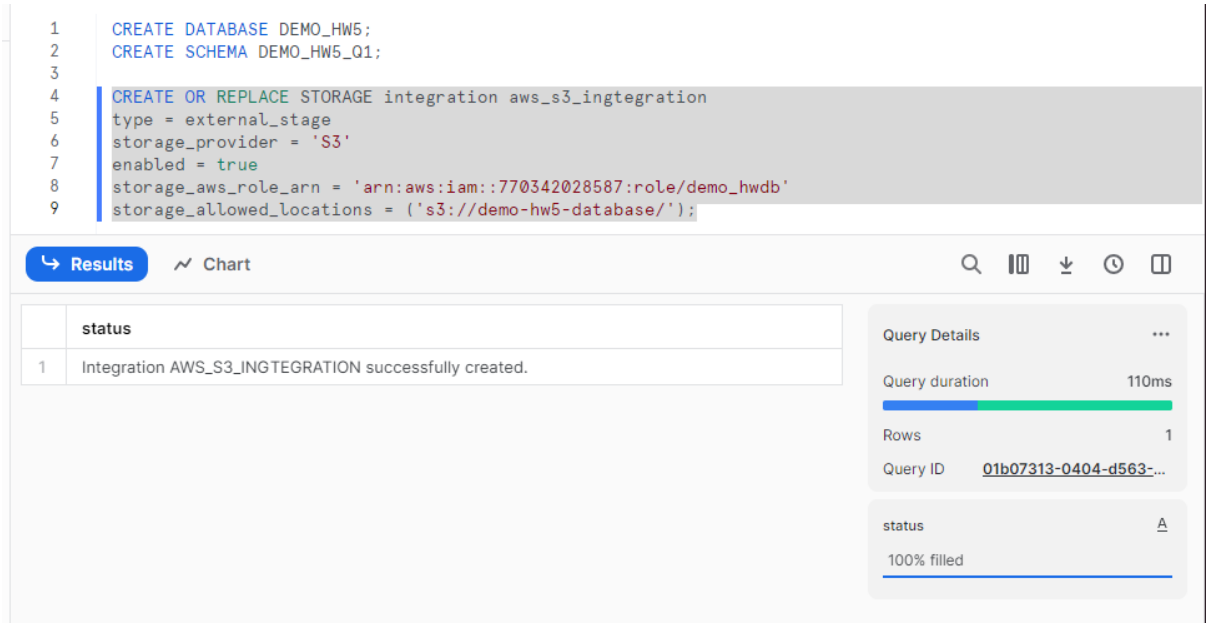
Account no: 770342028587

Everything is done from AWS the only thing left is to do the snowflake part

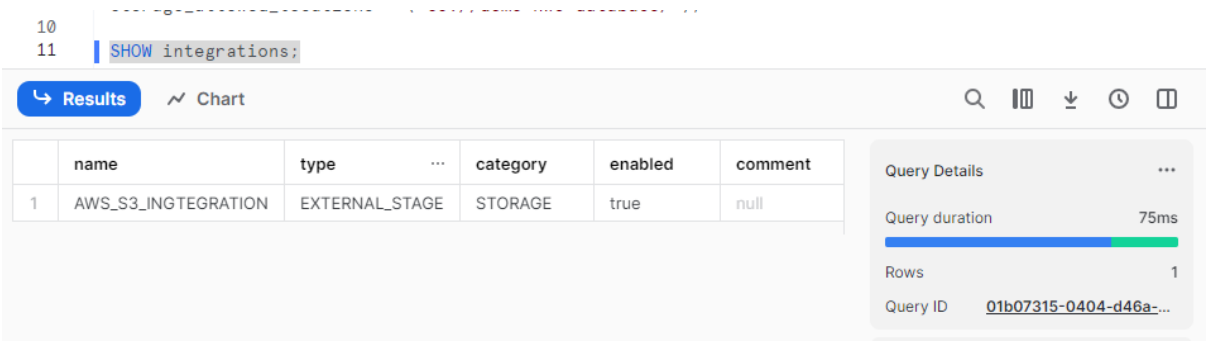
Step 4: Creating the database and schema in snowflake.



Step 5: Integration of snowflake and AWS



Integration with aws is successful.



```
10
11  SHOW integrations;
12
13  DESC integration aws_s3_ingtegration;
```

Results | Chart

	property	property_type	property_value
1	ENABLED	Boolean	true
2	STORAGE_PROVIDER	String	S3
3	STORAGE_ALLOWED_LOCATIONS	List	s3://demo-hw5-database/
4	STORAGE_BLOCKED_LOCATIONS	List	
5	STORAGE_AWS_IAM_USER_ARN	String	arn:aws:iam::836948182393:user/yqyf0000-
6	STORAGE_AWS_ROLE_ARN	String	arn:aws:iam::770342028587:role/demo_hwd
7	STORAGE_AWS_EXTERNAL_ID	String	AJB00084_SFCRole=2_ioW0gJXsSR1wtOWY
8	COMMENT	String	

Query Details
Query duration 72ms
Rows 8
Query ID 01b0731a-0404-d646-...

property A
100% filled

property_type A
String 5
List 2
Boolean 1

Creating the file format

```
16
17  create or replace file format deno_format
18  type = 'CSV'
19  field_delimiter = '|'
20  skip_header = 1;
```

Results | Chart

	status
1	File format DENO_FORMAT successfully created.

Query Details
Query duration 75ms
Rows 1
Query ID 01b07635-0001-e6a5-...

status A
100% filled

Creating the stage:

```
22  create or replace stage demo_aws_stage
23  storage_integration = aws_s3_integration
24  file_format = deno_format
25  url = 's3://demo-hw5-database/';
```

Results | Chart

	status
1	Stage area DEMO_AWS_STAGE successfully created.

Query Details
Query duration 436ms
Rows 1
Query ID 01b0763a-0001-e5c4-...

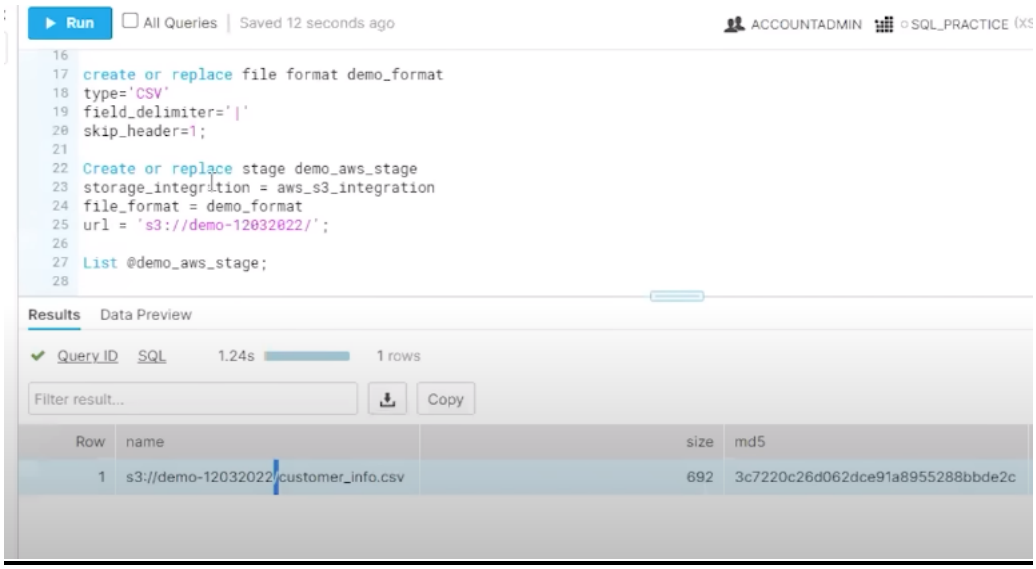
status A
100% filled

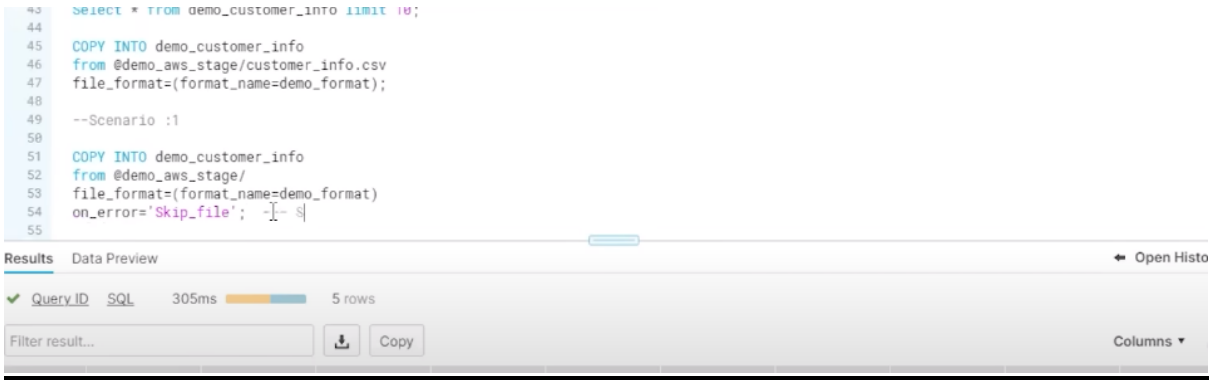
```sql
CREATE TABLE real_estate_db (
    timestamp STRING,
    title STRING,
    price FLOAT,
    market STRING,
    surface FLOAT,
    location STRING,
    remote_support BOOLEAN,
    lighting BOOLEAN,
    advertiser_type STRING,
    description STRING,
    no_of_rooms INTEGER,
    form_of_property STRING,
    url STRING,
    is_for_sale BOOLEAN,
    posting_id STRING
);
```

Results | Chart

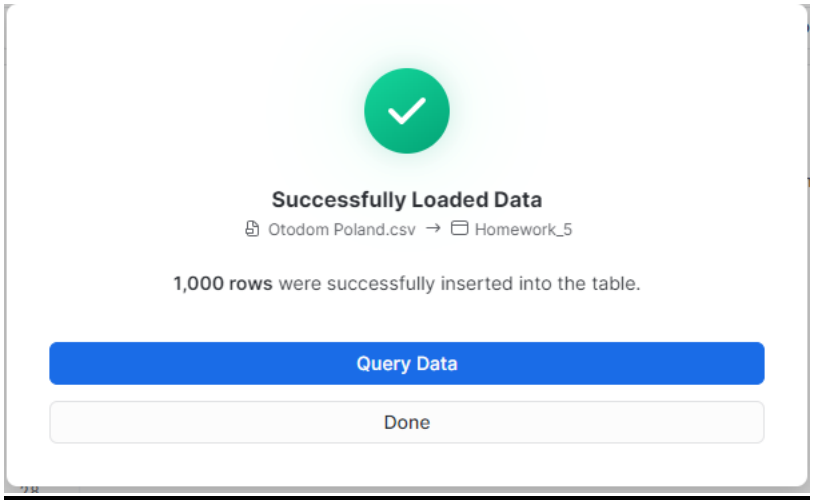
	status
1	Table REAL_ESTATE_DB successfully created.

Now we will check whether the stage is connected or not.

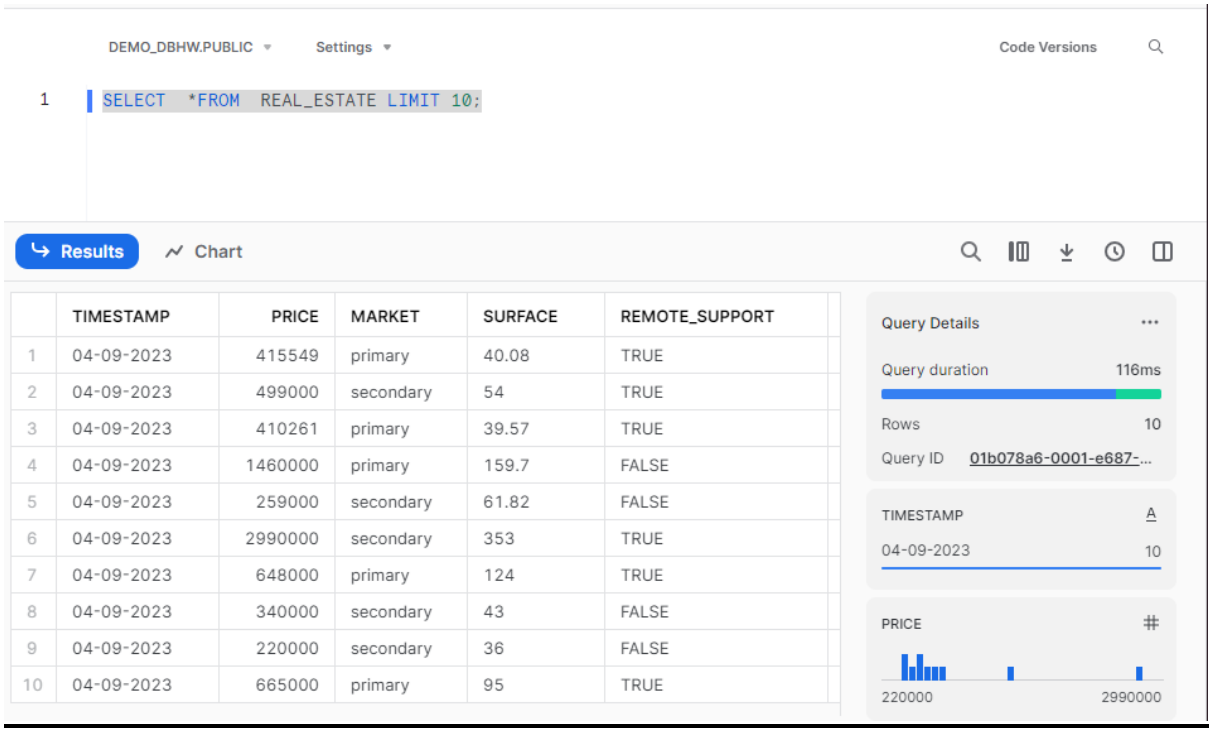




Importing the data



Know we will do some query on the data to get insights from the data.



	TIMESTAMP	PRICE	MARKET	SURFACE	REMOTE_SUPPORT
1	04-09-2023	415549	primary	40.08	TRUE
2	04-09-2023	499000	secondary	54	TRUE
3	04-09-2023	410261	primary	39.57	TRUE
4	04-09-2023	1460000	primary	159.7	FALSE
5	04-09-2023	259000	secondary	61.82	FALSE
6	04-09-2023	2990000	secondary	353	TRUE
7	04-09-2023	648000	primary	124	TRUE
8	04-09-2023	340000	secondary	43	FALSE
9	04-09-2023	220000	secondary	36	FALSE
10	04-09-2023	665000	primary	95	TRUE

Query 1

	MAX_PRICE	MIN_PRICE
1	5845000	0

Query 2

```
7  SELECT market, COUNT(*) AS property_count FROM REAL_ESTATE GROUP BY market;
8
```

Results | Chart

	MARKET	PROPERTY_COUNT
1	primary	372
2	secondary	620
3	null	8

Query Details

Query duration 588ms

Rows 3

Query ID 01b078c6-0001-e6a5-...

MARKET A

67% filled 33% null

PROPERTY_COUNT #

8 620

Query 3

```
SELECT AVG(CASE WHEN remote_support = 'Yes' THEN 1 ELSE 0 END) * 100 AS remote_support_percentage
FROM REAL_ESTATE;
```

sults | Chart

REMOTE_SUPPORT_PERCENTAGE
23.600000

Query Details

Query duration 70ms

Rows 1

Query ID 01b078cb-0001-e4af-0...

REMOTE_SUPPORT_PERCENTAGE #

100% filled

Query 4

```
SELECT advertiser_type, COUNT(*) AS property_count FROM REAL_ESTATE GROUP BY advertiser_type;
```

esults | Chart

ADVERTISER_TYPE	PROPERTY_COUNT
developer	115
agency	771
private	107
Lack of information	4
business	3

Query Details

Query duration 46ms

Rows 5

Query ID 01b078cc-0001-e4af-0...

ADVERTISER_TYPE A

100% filled

PROPERTY_COUNT #

3 771

Query 5

```
16
17  SELECT COUNT(*) AS for_sale_count FROM REAL_ESTATE WHERE is_for_sale = 'Yes';
18
```

Results | Chart

...	FOR_SALE_COUNT
1	1000

Query Details

Query duration 58ms

Rows 1

Query ID 01b078cd-0001-e5ff-0...

FOR_SALE_COUNT #

100% filled

Query 6

```
21
22  SELECT market,
23         COUNT(CASE WHEN lighting = 'Natural' THEN 1 END) AS natural_lighting_count,
24         COUNT(CASE WHEN lighting = 'Artificial' THEN 1 END) AS artificial_lighting_count
25  FROM REAL_ESTATE
26  GROUP BY market;
27
```

Results | Chart

	MARKET	NATURAL_LIGHTING_COUNT	ARTIFICIAL_LIGHTING_COUNT
1	primary	0	0
2	secondary	0	0
3	null	0	0

Query Details

Query duration 272ms

Rows 3

Query ID 01b078cf-0001-e687-0...

MARKET A

67% filled 33% null

Query 6

```sql
SELECT
    SUM(CASE WHEN remote_support = TRUE AND lighting = TRUE THEN 1 ELSE 0 END) AS
remote_and_good_lighting,
    COUNT(*) AS total_properties,
    (remote_and_good_lighting / total_properties) * 100 AS percentage
FROM real_estate;
```

Results | Chart

	REMOTE_AND_GOOD_LIGHTING	TOTAL_PROPERTIES	PERCENTAGE
1	0	1000	0.000000

Query Details
Query duration 121ms
Rows 1
Query ID 01b07b78-0001-e6c5-...

REMOTE_AND_GOOD_LIGHTING
100% filled

Query 7

```sql
SELECT  no_of_rooms, price, price / no_of_rooms AS price_per_room
FROM real_estate where no_of_rooms !=0;
```

Results | Chart

	NO_OF_ROOMS	PRICE	PRICE_PER_ROOM
4	5	1460000	292000.000000
5	2	259000	129500.000000
6	6	2990000	498333.333333
7	5	648000	129600.000000
8	2	340000	170000.000000
9	2	220000	110000.000000
10	4	665000	166250.000000
11	2	199000	99500.000000
12	2	523776	261888.000000
13	3	340000	113333.333333
14	4	474000	118500.000000
15	5	1290000	258000.000000
16	3	549000	183000.000000
17	2	452081	226040.500000
18	4	549900	137475.000000
19	4	849000	212250.000000

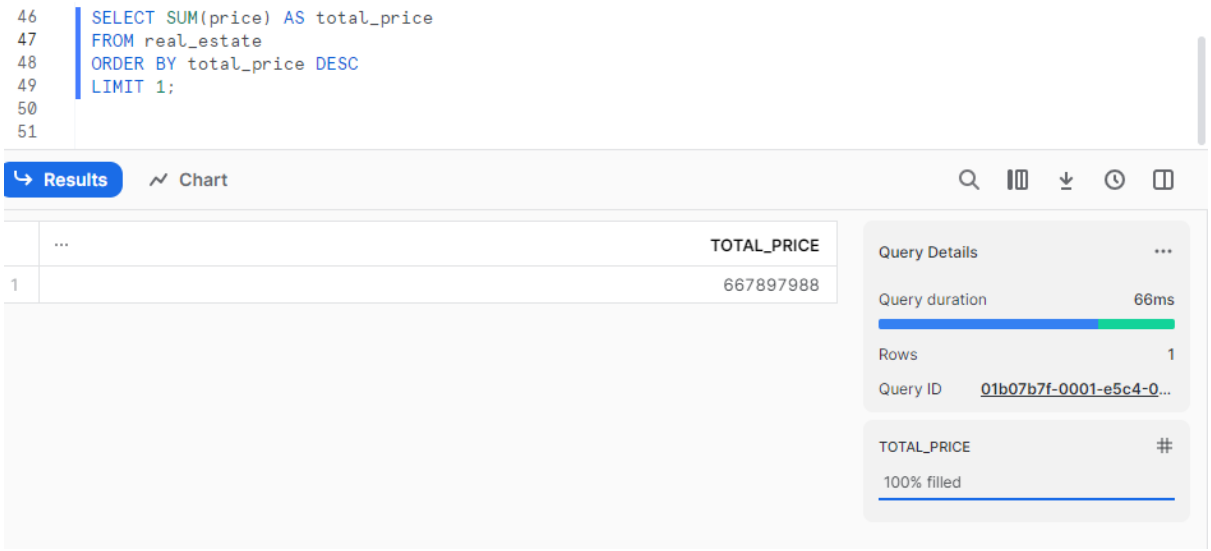
Query Details
Query duration 423ms
Rows 999
Query ID 01b07b7c-0001-e6a5-0...

NO_OF_ROOMS
1 10

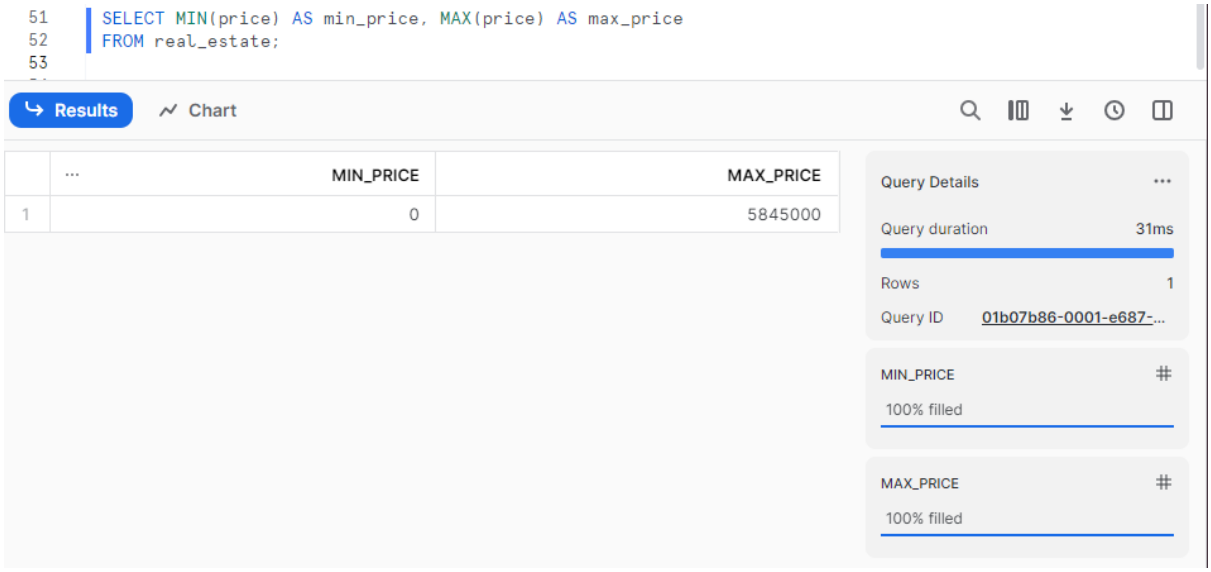
PRICE
0 5845000

PRICE_PER_ROOM
0 1461250

Query 8



Query 9



CODE:

I have added the worksheet from the snowflake in which all the code are present as we can't download the worksheet from snowflake in free version so, I have copied all the code to the notepad and have upload it's attached with files

Question 1 b:

Query 1:

```sql
SELECT
    timestamp,
    market,
    advertiser_type,
    no_of_rooms,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    CUBE(timestamp, market, advertiser_type, no_of_rooms);
```

Results | Chart

	TIMESTAMP	MARKET	ADVERTISER_TYPE	NO_OF_ROOMS	T
25	20-09-2023	secondary	private	3	
26	20-09-2023	primary	developer	4	
27	04-09-2023	primary	developer	4	
28	04-09-2023	secondary	agency	1	
29	17-10-2023	secondary	agency	3	
30	17-10-2023	primary	developer	2	
31	17-10-2023	secondary	private	3	
32	17-10-2023	primary	developer	5	
33	20-09-2023	primary	developer	3	
34	20-09-2023	secondary	agency	3	
35	20-09-2023	primary	developer	2	
36	20-09-2023	secondary	private	2	
37	20-09-2023	primary	agency	5	
38	17-10-2023	secondary	agency	2	
39	20-09-2023	secondary	agency	5	
40	26-07-2023	secondary	agency	3	
41	26-07-2023	null	Lack of information	2	
42	25-07-2023	primary	agency	3	
43	04-09-2023	secondary	private	3	

Query Details

Query duration 1.1s

Rows 501

Query ID 01b09e7d-0001-e9a9-...

TIMESTAMP

04-09-2023	95
20-09-2023	58
17-10-2023	55

+ 9 more

MARKET

secondary	146
primary	116

ADVERTISER_TYPE

agency	155
private	88
developer	48

+ 2 more

Query 2:

```sql
SELECT
    timestamp,
    market,
    advertiser_type,
    no_of_rooms,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    ROLLUP(timestamp, market, advertiser_type, no_of_rooms);
```

Results | Chart

	MARKET	ADVERTISER_TYPE	NO_OF_ROOMS	TOTAL_PROPERTIES
1	primary	developer	2	35
2	secondary	agency	2	130
3	primary	agency	5	9
4	primary	private	5	8
5	primary	agency	2	35
6	secondary	agency	3	131
7	secondary	agency	5	20
8	primary	private	4	6
9	secondary	agency	7	2
10	primary	agency	4	24

Query 3:

```sql
SELECT
    market,
    advertiser_type,
    AVG(price) as avg_price,
    MAX(surface) as max_surface,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    CUBE(market, advertiser_type);
```

Results | Chart

	MARKET	ADVERTISER_TYPE	AVG_PRICE	MAX_SURFACE	TOTAL_PROPERTIES
1	null	Lack of information	248677.250000	81,60 m²	4
2	null	private	390000.000000	3 000 m²	1
3	secondary	agency	706947.711397	99.4	544
4	primary	private	713633.733333	96.51	30
5	primary	developer	562448.869565	98.36	115

Query 4:

```
89
90  SELECT
91      market,
92      advertiser_type,
93      no_of_rooms,
94      COUNT(*) as total_properties
95  FROM
96      real_estate
97  WHERE
98      is_for_sale = true
99  GROUP BY
100     ROLLUP(market, advertiser_type, no_of_rooms);
101
```

Results | Chart

	MARKET	ADVERTISER_TYPE	NO_OF_ROOMS	TOTAL_PROPERTIES
1	primary	developer	2	42
2	secondary	agency	2	173
3	primary	agency	5	12
4	secondary	agency	6	11
5	primary	private	5	9
6	primary	developer	4	19
7	secondary	agency	5	29
8	primary	private	4	7
9	primary	agency	3	88
10	primary	agency	1	8
11	secondary	agency	1	34
12	primary	agency	4	52
13	secondary	private	2	25
14	secondary	agency	4	104

Query 5:

```sql
SELECT
    market,
    advertiser_type,
    AVG(CASE WHEN is_for_sale = true THEN price ELSE NULL END) as avg_price_for_sale,
    AVG(CASE WHEN is_for_sale = false THEN price ELSE NULL END) as avg_price_for_rent,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    CUBE(market, advertiser_type);
```

Results | Chart

	MARKET	ADVERTISER_TYPE	AVG_PRICE_FOR_SALE	AVG_PRICE_FOR_RENT	TOTAL_PROPERTIES
1	primary	developer	562448.869565	null	115
2	secondary	agency	706947.711397	null	544
3	primary	agency	632946.048458	null	227
4	primary	private	713633.733333	null	30
5	secondary	private	669662.355263	null	76
6	null	Lack of information	248677.250000	null	4
7	null	business	423333.333333	null	3
8	null	private	390000.000000	null	1
9	primary	null	617659.637097	null	372
10	secondary	null	702377.248387	null	620
11	null	null	331838.625000	null	8
12	null	null	667897.988000	null	1000
13	null	developer	562448.869565	null	115
14	null	agency	685159.932555	null	771
15	null	private	679377.112150	null	107
16	null	Lack of information	248677.250000	null	4
17	null	business	423333.333333	null	3

Query 6:

```sql
SELECT
    remote_support,
    AVG(price) as avg_price,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    CUBE(remote_support);
```

Results | Chart

	REMOTE_SUPPORT	AVG_PRICE	TOTAL_PROPERTIES
1	TRUE	740714.440678	236
2	FALSE	645404.947644	764
3	null	667897.988000	1000

Query 7:

```sql
SELECT
    remote_support,
    AVG(price) as avg_price,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    CUBE(remote_support);
```

Results | Chart

	REMOTE_SUPPORT	AVG_PRICE	TOTAL_PROPERTIES
1	TRUE	740714.440678	236
2	FALSE	645404.947644	764
3	null	667897.988000	1000

Query 8:

```sql
SELECT
    market,
    no_of_rooms,
    AVG(price) as avg_price,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    ROLLUP(market, no_of_rooms);
```

Results | Chart

	MARKET	NO_OF_ROOMS	AVG_PRICE	TOTAL_PROPERTIES
1	primary	2	487797.125000	112
2	secondary	2	477359.030303	198
3	primary	5	842676.391304	23
4	secondary	6	1539749.916667	12
5	primary	4	730613.692308	78
6	secondary	3	634757.885321	218
7	secondary	5	1183516.096774	31
8	primary	1	492889.200000	15
9	secondary	1	372291.666667	39
10	secondary	7	985550.000000	3
11	secondary	4	1089083.165217	115
12	primary	3	623566.742857	140

Query 9:

```sql
SELECT
    lighting,
    advertiser_type,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    CUBE(lighting, advertiser_type);
```

Results | Chart

	LIGHTING	ADVERTISER_TYPE	TOTAL_PROPERTIES
1	null	developer	115
2	null	agency	771
3	null	private	106
4	null	Lack of information	4
5	Ask	business	3
6	Ask	private	1
7	null	null	996
8	Ask	null	4
9	null	null	1000
10	null	developer	115
11	null	agency	771
12	null	private	107
13	null	Lack of information	4
14	null	business	3

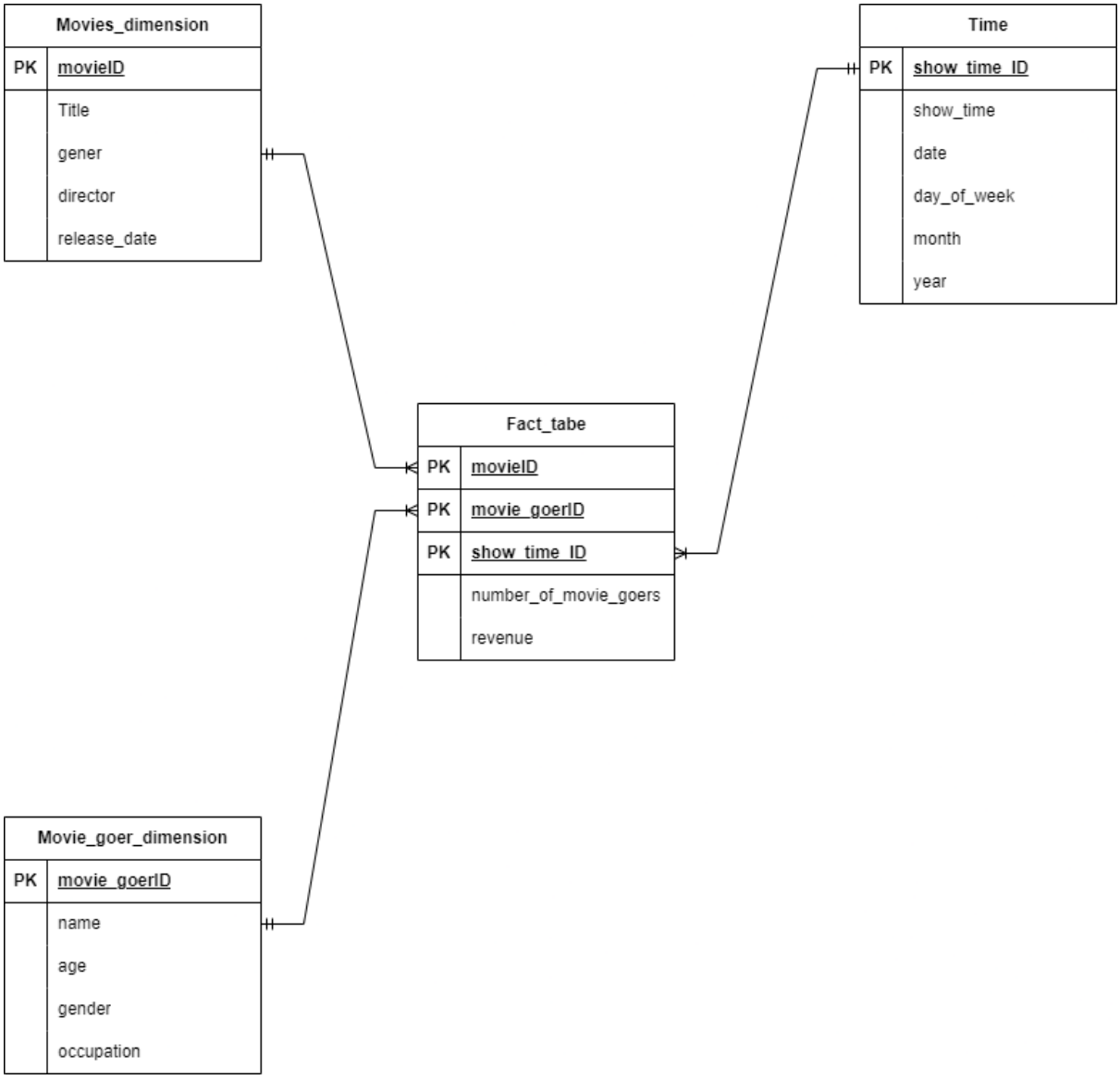
Query 10:

```sql
SELECT
    market,
    no_of_rooms,
    AVG(price) as avg_price,
    COUNT(*) as total_properties
FROM
    real_estate
GROUP BY
    ROLLUP(market, no_of_rooms);
```

Results | Chart

	MARKET	NO_OF_ROOMS	AVG_PRICE	TOTAL_PROPERTIES
1	primary	2	487797.125000	112
2	secondary	2	477359.030303	198
3	primary	5	842676.391304	23
4	secondary	6	1539749.916667	12
5	primary	4	730613.692308	78
6	secondary	3	634757.885321	218
7	secondary	5	1183516.096774	31
8	primary	1	492889.200000	15
9	secondary	1	372291.666667	39
10	secondary	7	985550.000000	3
11	secondary	4	1089083.165217	115
12	primary	3	623566.742857	140

Question 2 A:



Within this star schema:

The measurements (number of movie goers and revenue) and foreign keys referencing the dimensions are contained in the Fact_table table.

Descriptive information on movies, moviegoers, and time is contained in the Dimension_Movie, Dimension_MovieGoer, and Dimension_Time tables, respectively.

The foreign key connections build the linkages between the fact and dimension tables**.**

Question 2 b:

```
SELECT
   m.name, SUM(f.revenue) as total_revenue
FROM
   Fact_table f
```

```
JOIN
    Movies_dimension m ON f.movie_id = m.movie_id
JOIN
    Time t ON f.show_time_id = t.show_time_id
WHERE
    t.year = 2020
GROUP BY
    m.name;
```

Justification:

The Fact_table database has the total income, while the Movies_dimension table contains the name of the film. Using movie_id as the common key, it combines the fact table with the movie dimension table. Using show_time_id as the common key, it also links the fact table with the time dimension table. The WHERE clause restricts the entries that appear in the results to those from 2020. The results are grouped by movie name under the GROUP BY clause. The end result is the total amount of money made from the sale of tickets for every film in 2020.

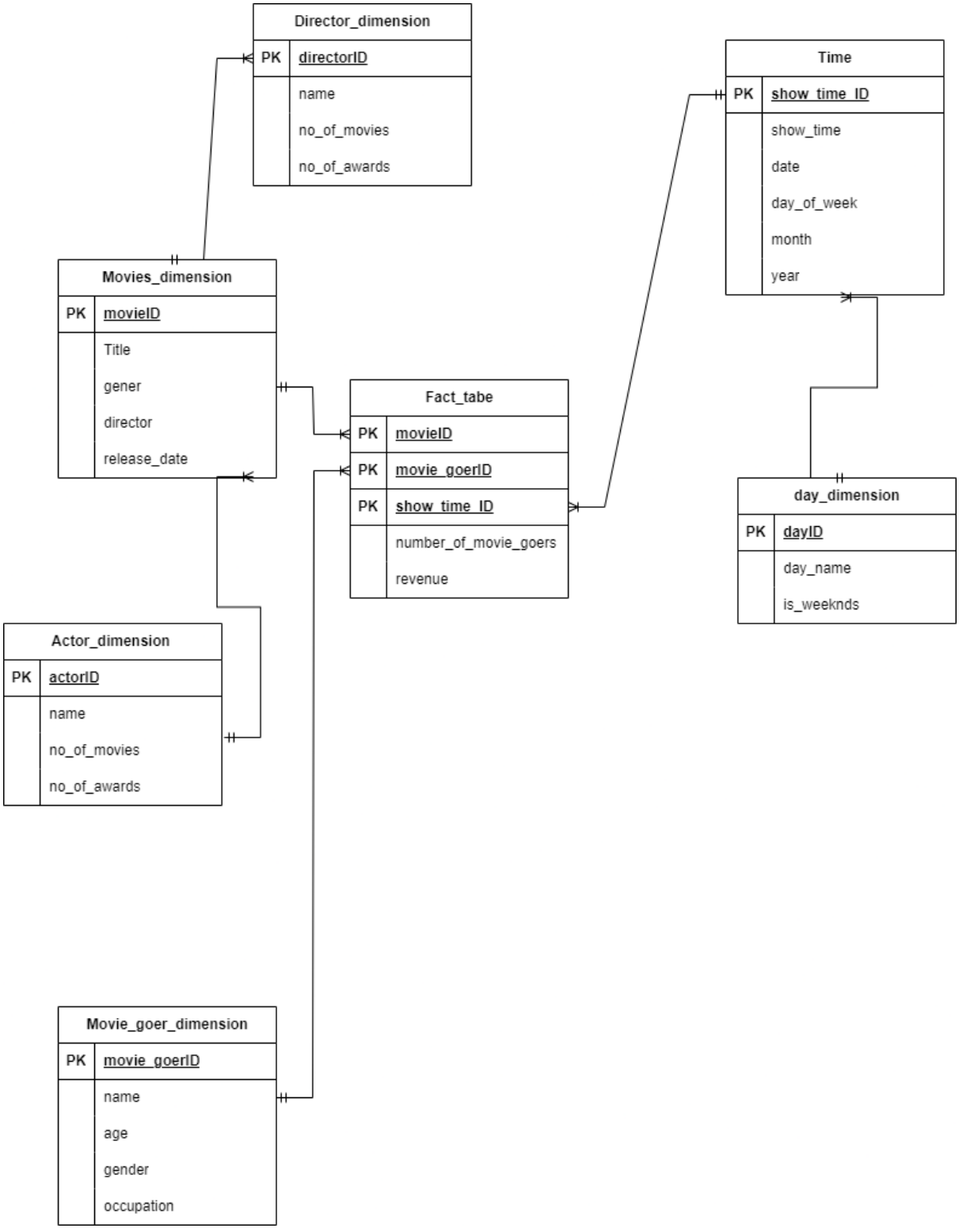
Question 2 c:

```
SELECT
    movie, SUM(ts.ticket_price) as total_revenue
FROM
    ticket_sales ts
JOIN
Movie_dimension m ON ts.movie = m.title
WHERE
    ts.year = 2020
GROUP BY
    movie;
```

Justification:

The total income field in the query is set to the sum of the ticket prices, and the movie column is taken from the ticket sales table. The WHERE clause restricts the entries that appear in the results to those from 2020. The movie column is used to group the results in the GROUP BY clause. The end result is the total amount of money made from the sale of tickets for every film in 2020.

Question 2 d:



Query 1:

```
SELECT
   mg.occupation,
   SUM(f.number_of_movie_goers) as total_movie_goers
FROM
```

```
    Fact_MovieGoerTime f
JOIN
    Dimension_MovieGoer mg ON f.movie_goer_id = mg.movie_goer_id
GROUP BY
    mg.occupation
ORDER BY
    total_movie_goers DESC;
```

Query 2:

```
SELECT
    m.genre,
    SUM(f.number_of_movie_goers) as total_movie_goers
FROM
    Fact_MovieGoerTime f
JOIN
    Dimension_Movie m ON f.movie_id = m.movie_id
GROUP BY
    m.genre
ORDER BY
    total_movie_goers DESC;
```

Question 3 a:

Because columnar databases are so good at performing analytical tasks, they have become more and more common in the business, particularly when used to construct data warehouse solutions. The following five points highlight the importance of adopting columnar databases in data warehousing:

1. **Performance Improvement for Analytic Queries:**
   Remark: Data is stored in columnar databases column-wise as opposed to row-wise. For analytical queries including aggregations, filtering, and other actions on a subset of columns, this architecture is especially helpful.As an illustration, the columnar database Vertica is renowned for its capacity to greatly improve the efficiency of intricate analytical queries. Vertica has been widely used in sectors like banking and telecoms where quick and effective data analysis is essential.

2. **Compression Techniques for Storage Efficiency:**
   Observation: Columnar databases frequently use compression strategies designed especially for columns, which results in significant storage reductions. This is so that column values in a database may be compressed more successfully because they often have comparable data types.
   Columnar storage with compression is used by the well-known cloud-based data warehousing system, Amazon Redshift. This is advantageous for enterprises with variable data quantities, such as e-commerce and online shopping, since it enables firms to handle and analyze enormous datasets more economically.

3. **Scalability for Growing Data Volumes:**
   Remark: Columnar databases are ideal for managing massive and expanding data volumes because of their horizontal scalability architecture. For businesses handling large volumes of data, including e-commerce, healthcare, and online advertising, scalability is essential.
   Columnar storage is used by Google BigQuery, a serverless, highly scalable data warehouse. As their data expands, this enables enterprises to smoothly scale their analytical workloads. BigQuery is used by businesses in industries such as internet advertising to do real-time analytics on large datasets.
4. **Optimized for Aggregations and Reporting:**
   Observation: Columnar databases are best suited for reporting and aggregations, which are common components of analytical tasks. During query execution, the ability to read only the columns that are required speeds up the retrieval and processing of data.
   An example of a cloud-based data warehousing platform that employs a columnar storage architecture with several clusters is Snowflake. Because of its nature, Snowflake is ideally suited for sectors like retail, where complicated data aggregations and reporting are standard needs.

5. **Integration with Analytical Tools and Ecosystem:**
   Observation: Columnar databases are easy to connect with popular ecosystems and analytical tools. The deployment of analytics and business intelligence (BI) systems is made easier by this compatibility.
   For instance: Online gaming and other businesses employ distributed NoSQL columnar databases like Apache Cassandra. Because of its good integration with analytical tools, businesses may examine user behaviour and enhance the game experience. High-performance and scalable applications require distributed storage, which meets the needs of columnar storage for analytical queries.

In conclusion, the industry's use of columnar databases for data warehousing has shown to be advantageous in terms of enhancing analytical query performance, attaining storage efficiency, guaranteeing scalability, enhancing reporting and aggregations, and simplifying interaction with analytical tools. These findings demonstrate the applicability and efficiency of columnar databases in meeting the intricate data analysis requirements of several businesses.

Question 3 b:

Because OLAP (Online Analytical Processing) technologies allow users to study multidimensional data interactively, they are essential to business intelligence. These are five key points about the different OLAP tools that are utilized in the market, such as Oracle OLAP, Microsoft SSAS (SQL Server Analysis Services), and IBM Cognos.

1. Multidimensional Modeling and Analysis Capabilities
2. Performance Optimization through Aggregation and Caching
3. Integration with BI Platforms for Seamless Reporting
4. Support for Both MOLAP and ROLAP Architectures
5. Advanced Analytical Features and Predictive Analytics Integration

IBM Congos:

A popular tool for corporate intelligence and performance management, IBM Cognos offers a range of applications for scorecarding, reporting, analysis, and event and metric monitoring. Based on the analysis of IBM Cognos, the following five key findings are presented, along with particular instances from sectors where the product has had a notable influence.

1. Comprehensive Business Intelligence Suite
2. Integration with Data Warehousing Solutions
3. Self-Service Analytics Empowering Business Users
4. Mobile Business Intelligence for On-the-Go Access
5. AI-Powered Insights and Predictive Analytics

Microsoft SSAS:

Online analytical processing (OLAP) and data modeling may be accomplished with Microsoft SQL Server Analysis Services (SSAS), a multidimensional and data mining tool. These are five key takeaways from Microsoft SSAS, along with particular instances from sectors where it has had a notable influence:

1. Tabular and multidimensional models
2. Attachment to Microsoft BI Stack
3. Analytics Predictive and Data Mining
4. Optimizing performance and scalability
5. Data governance and security based on roles

In conclusion, Microsoft SSAS provides a flexible range of functionalities, such as support for both multidimensional.

Oracle OLAP:

A feature of the Oracle Database called Oracle OLAP (Online Analytical Processing) offers sophisticated analytical tools for multidimensional data processing. Here are five key insights about Oracle OLAP, along with concrete instances from sectors where it has had a notable influence:

1. Oracle Database Integration
2. Support for Cubes with Multiple Dimensions
3. Features for Advanced Aggregation and Calculation
4. Flexibility in Handling Big Datasets
5. Combining Business Intelligence Tools with Integration

All things considered, Oracle OLAP offers strong analytical capabilities, smooth connection with the Oracle Database, multidimensional cube support, sophisticated calculation functions, scalability for big datasets, and business intelligence tool compatibility. Because of these characteristics, Oracle OLAP is a useful tool for businesses in a variety of sectors that want to extract important insights from their multidimensional data.

References:

https://www.youtube.com/watch?v=NXaRxwax9qA

https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-bucket.html

https://docs.aws.amazon.com/quickstarts/latest/s3backup/step-1-create-bucket.html

https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html

https://www.youtube.com/watch?v=_DRklnnJbig

https://www.youtube.com/watch?v=Q_2CIivxVVs

https://community.claris.com/en/s/question/0D50H00006dtMgASAU/upload-file-from-container-to-s3

https://community.claris.com/en/s/question/0D50H00006dtMgASAU/upload-file-from-container-to-s3

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html

https://www.strongdm.com/blog/aws-iam-roles-vs-policies

https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_access-management.html

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html

https://www.arubanetworks.com/techdocs/central/2.5.1/content/gateways/vgw/vgw-orch-aws-crte-role-arn.htm

https://docs.panther.com/cloud-scanning/cloud-resource-attributes/aws/iam-role

https://snowflakewiki.medium.com/connecting-snowflake-to-aws-ef7b6de1d6aa

https://docs.snowflake.com/en/user-guide/data-load-s3-config

https://docs.snowflake.com/en/user-guide/data-load-s3-config-storage-integration

https://www.youtube.com/watch?v=m5-78NkUzZA

https://www.ibm.com/products/cognos-analytics?utm_content=SRCWW&p1=Search&p4=43700050328201997&p5=e&gclid=Cjw

KCAiAvJarBhA1EiwAGgZl0Ax0FNUNwQO0nNXGdESkZmB0r4lT9QMbTx5aHXurX2nHMDryYdW7WhoC2TgQAvD_BwE&gclsrc=aw.ds

https://www.ibm.com/products/cognos-analytics

https://en.wikipedia.org/wiki/IBM_Cognos_Analytics

https://en.wikipedia.org/wiki/Cognos

https://www.solarwinds.com/sql-sentry/use-cases/ssas-performance-monitoring?CMP=KNC-TAD-GGL-SW_NA_X_PP_CPC_LD_EN_PRODB_TIN-XDB&17913394546~139205942626_g_c_ssas-e~614098017743~~p72469533976&s_kwcid=AL!11508!3!614098017743!e!!g!!ssas&ds_cid=71700000098707476&ds_agid=58700007958078445&network=g&device=c&keyword=Ssas&matchtype=e&creative=614098017743&feeditemid=&gad_source=1&gclid=CjwKCAiAvJarBhA1EiwAGgZl0NP8UUiukfgVU8l69U3Cp0xIK5xnL2IAvRvoVn5cWu3CwQt4hZ6rthoCO0IQAvD_BwE&gclsrc=aw.ds

https://learn.microsoft.com/en-us/analysis-services/ssas-overview?view=asallproducts-allversions

https://en.wikipedia.org/wiki/Microsoft_Analysis_Services

https://learn.microsoft.com/en-us/analysis-services/analysis-services-overview?view=asallproducts-allversions

https://www.oracle.com/database/technologies/olap.html

https://www.oracle.com/database/technologies/olap.html#:~:text=Oracle%20OLAP%20is%20a%20world,speed%20of%20thought%20response%20times.

https://www.techtarget.com/searchdatamanagement/definition/OLAP#:~:text=OLAP%20(online%20analytical%20processing)%20is,from%20different%20points%20of%20view.

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Oracle.Options.OLAP.html

https://www.dba-oracle.com/t_olap_dimensions_cubes.htm