

PROJECT GLADIATOR



Team Members	PS No.
Abilash Kamaraj	10709301
Praveen Chandar	10709235
Sejal Mahadev Jadhav	10709498

AIRLINES CASE STUDY

Data sources: Flat files ,AWS S3 ,JSON,parquet and ORC

Scheduler : Task

Streaming : Streams / Snowpipes

Visualisation : PowerBI

Language : SQL/Javascript (Stored procedure)

Problem Statement:

Flight delay and cancellation are inevitable and they often lead to concern in passengers as well as profit loss of the airlines and airports. An accurate estimation of flight delays and cancellation is critical for airlines because the results can be applied to increase customer satisfaction and income of airline agencies. The passengers' dissatisfaction and distrust of airlines seriously damage the airlines' corporate reputation and then affect passengers' loyalty. Based on the data published by the US department of transportation, this project focuses on analysing the various reasons for flight cancellation and delays.

Dataset:

Source Dataset consists of 3 csv files(comma separated values) downloaded from kaggle website.

Flights.csv

Rows	Columns	Size
5.8M	31	592MB

Column Name	Description	Data Type
YEAR	Year of the Flight Trip (Contains only 2015 data)	Number
MONTH	Month of the Flight Trip	Number

DAY	Day of the Flight Trip	Number
DAY_OF_WEEK	Day of week of the Flight Trip	Number
AIRLINE	Airline Identifier containing two letter code of the airline.	Plain Text
FLIGHT_NUMBER	Flight Identifier	Number
TAIL_NUMBER	Aircraft Identifier containing six letter code of the aircraft. This column contains 4898 unique values.	Plain Text
ORIGIN_AIRPORT	An IATA airport code, also known as an IATA location identifier, IATA station code, or simply a location identifier, is a three-character alphanumeric geocode designating many airports.	Plain Text
DESTINATION_AIRPORT	An IATA airport code, also known as an IATA location identifier, IATA station code, or simply a location identifier, is a three-character alphanumeric geocode designating many airports.	Plain Text
SCHEDULED_DEPARTURE	Planned Departure Time	Number
DEPARTURE_TIME	Actual Departure Time	Number
DEPARTURE_DELAY	Amount of Time indicating the total delay on departure	Number
TAXI_OUT	The time duration elapsed between departure from the origin airport gate and wheels off	Number
WHEELS_OFF	The time point that the aircraft's wheels leave the ground	Number

SCHEDULED_TIME	Planned time amount needed for the flight trip	Number
ELAPSED_TIME	Total time duration containing the sum of air_time,taxi_in and taxi_out	Number
AIR_TIME	The time duration between wheels_off and wheels_on time	Number
DISTANCE	Distance between two airports	Number
WHEELS_ON	The time point that the aircraft's wheels touch on the ground	Number
TAXI_IN	The time duration elapsed between wheels-on and gate arrival at the destination airport	Number
SCHEDULED_ARRIVAL	Planned arrival time	Number
ARRIVAL_TIME	Actual arrival time	Number
ARRIVAL_DELAY	Amount of time indicating the difference between arrival_time and scheduled_arrival	Number
DIVERTED	Column indicating whether the flight was diverted (1=diverted)	Number
CANCELLED	Column indicating whether the flight was cancelled (1=cancelled)	Number
CANCELLATION_REASON	Reason for Cancellation of flight: A - Airline/Carrier; B - Weather; C - National Air System; D - Security	Plain Text
AIR_SYSTEM_DELAY	Delay caused by air system	Number

SECURITY_DELAY	Delay caused due to security issues	Number
AIRLINE_DELAY	Delay caused by the airline	Number
LATE_AIRCRAFT_DELAY	Delay caused by the aircraft	Number
WEATHER_DELAY	Delay caused due to bad weather	Number

Airports.csv

Rows	Columns	Size
323	7	23KB

Column Name	Description	Data Type
IATA_CODE	An IATA airport code, also known as an IATA location identifier, IATA station code, or simply a location identifier, is a three-character alphanumeric geocode designating many airports.	Plain text
AIRPORT	Name of the airport (Contains 322 unique values)	Plain text
CITY	Name of the city in which airport is located (Contains 308 unique values)	Plain text
STATE	State in which airport is located	Plain text
COUNTRY	Country in which airport is located (Contains only one unique value - USA)	Plain Text
LATITUDE	Latitude of the airport	Number
LONGITUDE	Longitude of the airport	Number

Airlines.csv

Rows	Columns	Size
15	2	360B

Column Name	Description	Data Type
IATA_CODE	IATA airline designators, sometimes called IATA reservation codes, are two-character codes assigned by the International Air Transport Association (IATA) to the world's airlines. (Contains 14 unique values)	Plain text
AIRLINE	Name of the airline	Plain text

Task 1: Create database `airlines_db` and schema `airlines_schema`

1. Creating two users and granting privileges

```

12 --user creation
13 use role useradmin;
14
15 create or replace user Abilash password = 'Abilash@123' default_role = 'sysadmin' default_namespace = 'airlines_db.airlines_schema' default_warehouse = 'compute_wh'
16 must_change_password = FALSE;
17
18 create or replace user sejal password = 'Sejal@123' default_role = 'sysadmin' default_namespace = 'airlines_db.airlines_schema' default_warehouse = 'compute_wh'
19 must_change_password = FALSE;
20
21 use role securityadmin;
22 grant role sysadmin to user Abilash;
23
24 grant role sysadmin to user sejal;
25

```

- Two users named ‘Abilash’ and ‘sejal’ were created in the team leader’s account.
- SYSADMIN role is granted to both the users.

2. Creating database

A database named ‘airlines_db’ is created.

```
1  
2  
3 use role sysadmin;  
4  
5 --creating database  
6 create or replace database airlines_db;  
7  
8  
9  
10  
11  
12
```

Results Data Preview

✓ Query ID SQL 110ms 1 rows

Filter result... Columns ▾

Row	status
1	Database AIRLINES_DB successfully created.

3. Creating schema

A schema named ‘airlines_schema’ is created.

```
1  
2  
3 use role sysadmin;  
4  
5 --creating database  
6 create or replace database airlines_db;  
7  
8  
9  
10  
11  
12
```

Results Data Preview

✓ Query ID SQL 110ms 1 rows

Filter result... Columns ▾

Row	status
1	Database AIRLINES_DB successfully created.

Task 2: Create 3 tables airlines,airports and flights based on the given csv files and apply cluster keys on the most used column for querying.

Task 3: Add the table definition with constraints , unicode and data retention.

1. Creating CSV file format

```
1 CREATE OR REPLACE FILE FORMAT "AIRLINES_DB"."AIRLINES_SCHEMA".CSV TYPE = 'CSV' COMPRESSION = 'AUTO' FIELD_DELIMITER = ',' RECORD_DELIMITER = '\n' SKIP_HEADER = 1  
2 FIELD_OPTIONALLY_ENCLOSED_BY = 'NONE' TRIM_SPACE = FALSE ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE ESCAPE = 'NONE' ESCAPE_UNENCLOSED_FIELD = '\134'  
3 DATE_FORMAT = 'AUTO' TIMESTAMP_FORMAT = 'AUTO' NULL_IF = ('\\N');  
4
```

Results Data Preview

✓ Query ID SQL 65ms 1 rows

Filter result... Columns ▾

Row	status
1	File format CSV successfully created.

- A CSV file format named ‘CSV’ is created.
- Data in the dataset is delimited by comma and separated by newline, therefore the ‘field_delimiter’ and ‘record_delimiter’ parameters are chosen accordingly.
- Each dataset contains a header, so it is skipped by setting the ‘skip_header’ parameter as 1.

2. Creating airlines table

The screenshot shows a database interface with two separate query results. The first result is for creating the 'airlines' table:

```

5 create or replace table airlines(
6   airlineId varchar(10) primary key,
7   airlineName varchar(100) not null
8 ) cluster by (airlineId)
9 DATA_RETENTION_TIME_IN_DAYS=90
10 STAGE_FILE_FORMAT=(FORMAT_NAME='CSV' ENCODING=UTF8);
11

```

The second result is for altering the table to set a data retention period of 30 days:

```

42 alter table airlines set data_retention_time_in_days=30;
43
44

```

Both results show a single row with status 'Table successfully created.' and 'Statement executed successfully.' respectively.

- A table named ‘airlines’ is created with 2 fields.
- The field ‘airlineId’ is set as the primary key because it uniquely identifies each airline and it acts as a reference to ‘airlineId’ in the ‘flights’ table.
- The table is clustered by ‘airlineId’ because it is used in join predicate.
- The retention period for the table is set as 30 days.
- ‘CSV’ file format is used for loading the table and ‘UTF8’ is set as the character set of the source data when loading data into the table.

3. Creating airports table

```
12 create or replace table airports(
13     airportId varchar(10) primary key,
14     airportName varchar(100) not null,
15     city varchar(50) not null,
16     state varchar(10) not null,
17     country varchar(10) not null,
18     latitude float null,
19     longitude float null
20 ) cluster by (airportId)
21 DATA_RETENTION_TIME_IN_DAYS=90
22 STAGE_FILE_FORMAT=(FORMAT_NAME='CSV' ENCODING=UTF8);
```

Results Data Preview [Open History](#)

✓ Query ID SQL 133ms 1 rows

Filter result... [Download](#) [Copy](#) Columns ▾

Row	status
1	Table AIRPORTS successfully created.

```
42
43 alter table airports set data_retention_time_in_days=30;
44
```

Results Data Preview [Open History](#)

✓ Query ID SQL 55ms 1 rows

Filter result... [Download](#) [Copy](#) Columns ▾

Row	status
1	Statement executed successfully.

- A table named ‘airports’ is created with 7 fields.
- The field ‘airportsId’ is set as the primary key because it uniquely identifies each airport and it acts as a reference to ‘orgAirport’ and ‘destAirport’ in the flights table.
- The table is clustered by ‘airportId’ because it is used in join predicate.
- The retention period for the table is set as 30 days.
- ‘CSV’ file format is used for loading the table and ‘UTF8’ is set as the character set of the source data when loading data into the table.

4. Creating flights table

```
27 --create flights table
28 create or replace table flights(
29   travelDate date not null, dayOfWeek number not null, airlineId varchar(10) not null, flightNumber number not null, tailNumber varchar(10) null,
30   orgAirport varchar(10) not null, destAirport varchar(10) not null, scheduledDeparture time null,
31   departureTime time null, departureDelay number null, taxiOut number null, wheelsOff time null, scheduledTime number null,
32   elapsedTime number null, airTime number null, distance number not null, wheelsOn time null, taxiIn number null,
33   scheduledArrival time null, arrivalTime time null, arrivalDelay number null, diverted number not null, cancelled number not null,
34   cancellationReason varchar(10) null, airSystemDelay number null, securityDelay number null, airlineDelay number null,
35   lateAircraftDelay number null, weatherDelay number null,
36   constraint fk_airline_id foreign key (airlineId) references airlines(airlineId),
37   constraint fk_orgportid foreign key (orgAirport) references airports(airportId),
38   constraint fk_destport_id foreign key (destAirport) references airports(airportId)
39 ) cluster by (travelDate)
40 DATA_RETENTION_TIME_IN_DAYS=30
41 STAGE_FILE_FORMAT=(FORMAT_NAME='CSV' ENCODING=UTF8);
```

Results		Data Preview	Open History
Row	status		
1	Table FLIGHTS successfully created.		Columns ▾

- A table named ‘flights’ is created with 29 fields.
- The field ‘airlineId’ is set as foreign key and it references ‘airportId’ in the airlines table.
- The field ‘orgAirport’ and ‘destAirport’ is set as foreign key and it references ‘airportId’ in the airports table.
- Since the data is mostly filtered by month and day, travelDate is chosen as a clustering key for the table and this improves the query performance of date-based queries.
- The retention period for the table is set as 30 days.
- ‘CSV’ file format is used for loading the table and ‘UTF8’ is set as the character set of the source data when loading data into the table.

5. Checking the clustering depth

The clustering depth for a populated table measures the average depth (1 or greater) of the overlapping micro-partitions for specified columns in a table. The smaller the average depth, the better clustered the table is with regards to the specified columns.

The screenshot shows a Snowflake query results page. The SQL query is:

```

46 --checking CLUSTERING DEPTH
47 select system$clustering_depth('FLIGHTS');

```

The results table has one row:

Row	SYSTEM\$CLUSTERING_DEPTH('FLIGHTS')
1	2

Task 4: Create internal stage and load flights and airports

Internal stages are named database objects that provide the greatest degree of flexibility for data loading. Internal Stage stores data files internally within Snowflake.

```

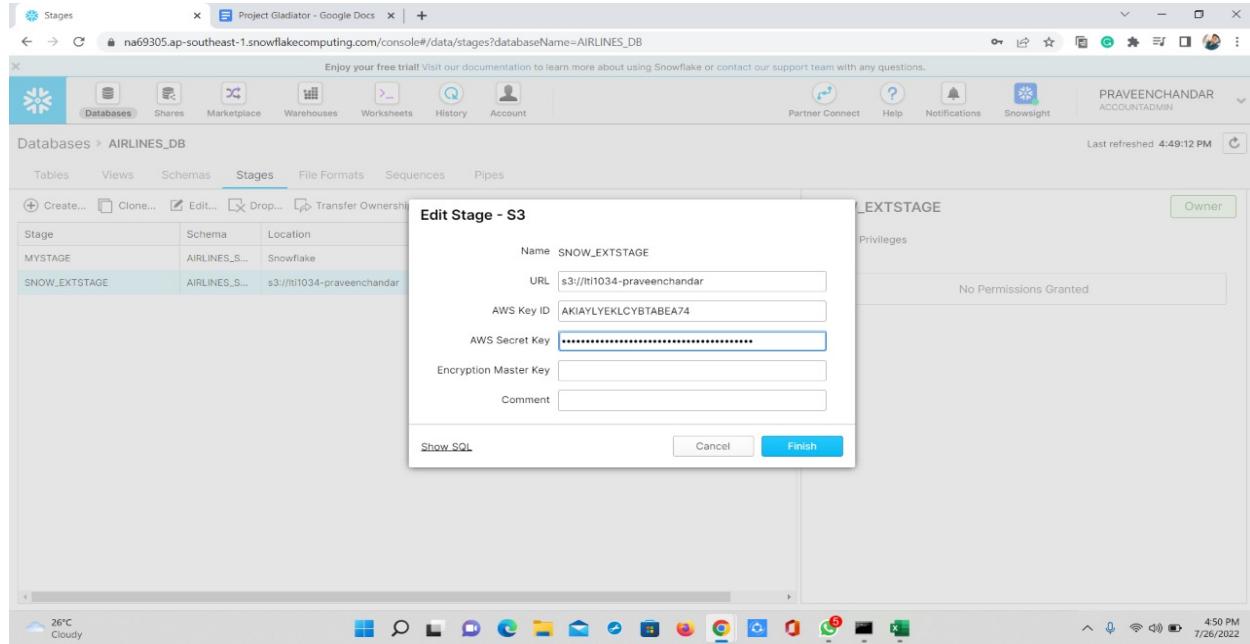
sejal#COMPUTE_WH@AIRLINES_DB.AIRLINES_SCHEMA>CREATE OR REPLACE STAGE Mystage;
+-----+
| status |
+-----+
| Stage area MYSTAGE successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.334s
sejal#COMPUTE_WH@AIRLINES_DB.AIRLINES_SCHEMA>PUT file:///C:/Users/rahul/Downloads/airports.csv @Mystage;
+-----+-----+-----+-----+-----+-----+-----+-----+
| source | target | source_size | target_size | source_compression | target_compression | status | message |
+-----+-----+-----+-----+-----+-----+-----+-----+
| airports.csv | airports.csv.gz | 23867 | 9840 | NONE | GZIP | UPLOADED |          |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 2.563s
sejal#COMPUTE_WH@AIRLINES_DB.AIRLINES_SCHEMA>LIST @Mystage;
+-----+-----+-----+
| name | size | md5 | last_modified |
+-----+-----+-----+
| mystage/airports.csv.gz | 9840 | 67457636361c8a6d8b1000e6661362ab | Tue, 26 Jul 2022 07:32:42 GMT |
+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 0.181s
sejal#COMPUTE_WH@AIRLINES_DB.AIRLINES_SCHEMA>COPY INTO AIRPORTS FROM @Mystage FILE_FORMAT=CSV PURGE=TRUE ON_ERROR=SKIP_FILE;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| file | status | rows_parsed | rows_loaded | error_limit | errors_seen | first_error | first_error_line | first_error_character | first_error_column_name |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| mystage/airports.csv.gz | LOADED | 322 | 322 | 1 | 0 | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 Row(s) produced. Time Elapsed: 3.161s

```

- A named internal stage called ‘Mystage’ has been created using the ‘create or replace stage’ command.
- The ‘airports.csv’ file has been loaded into ‘Mystage’ using the ‘PUT’ command.
- The files staged in ‘Mystage’ are listed using the ‘LIST’ command.
- The file is loaded into the table ‘airports’ from the named internal stage ‘Mystage’ using the ‘COPY INTO’ command.
- The ‘CSV’ file format is used to load and the ‘PURGE’ parameter is set ‘TRUE’ in order to remove the file from the stage after successful load.

Task 5: Load airlines files into AWS s3 bucket snow_extstage

1. Creating the external stage



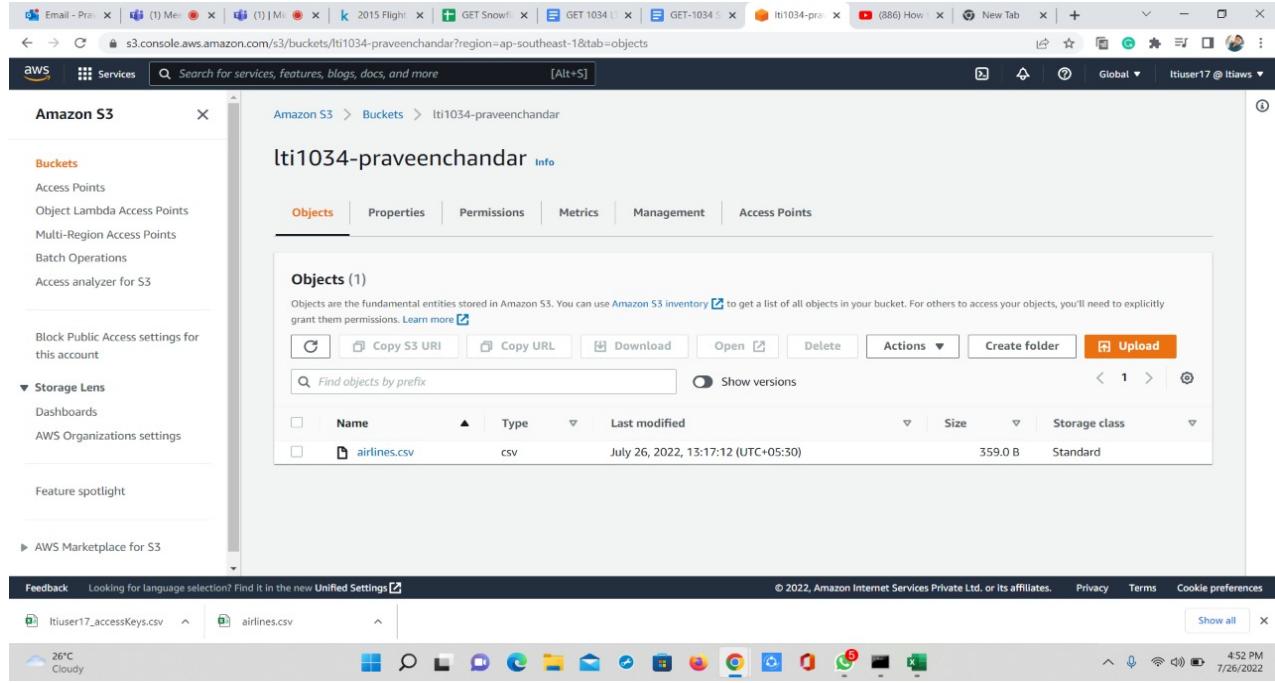
The screenshot shows the Snowflake web interface with the URL na69305.ap-southeast-1.snowflakecomputing.com/console#/data/stages?databaseName=AIRLINES_DB. The user is navigating through the AIRLINES_DB database. In the top navigation bar, the 'Stages' tab is selected. Below it, the 'Edit Stage - S3' dialog is open, showing the configuration for the 'SNOW_EXTSTAGE'. The stage details include:

- Name: SNOW_EXTSTAGE
- URL: s3://lti1034-praveenchandar
- AWS Key ID: AKIAIAYEKLKYBTAEBEA74
- AWS Secret Key: [REDACTED]
- Encryption Master Key: [REDACTED]
- Comment: [REDACTED]

The 'Privileges' section indicates 'No Permissions Granted'. At the bottom of the dialog are 'Show SQL', 'Cancel', and 'Finish' buttons.

- An external stage named 'snow_exstage' is created with the appropriate credentials.

2. Uploading file in AWS S3 bucket



The screenshot shows the AWS S3 console at the URL s3.console.aws.amazon.com/s3/buckets/lti1034-praveenchandar?region=ap-southeast-1&tab=objects. The user is viewing the 'Objects' tab for the bucket 'lti1034-praveenchandar'. The 'Objects (1)' section displays a single file named 'airlines.csv' with the following details:

Name	Type	Last modified	Size	Storage class
airlines.csv	csv	July 26, 2022, 13:17:12 (UTC+05:30)	359.0 B	Standard

The left sidebar of the S3 console shows various navigation options like Buckets, Access Points, Storage Lens, and Feature spotlight.

- The data file named ‘airlines.csv’ is uploaded in the AWS S3 bucket.

3. Listing the uploaded file



A screenshot of the Snowflake SQL interface. The query window shows the following SQL code:

```

31
32 use role sysadmin;
33
34
35
36
37
38 list @snow_extstage;
39
40
41
42
43

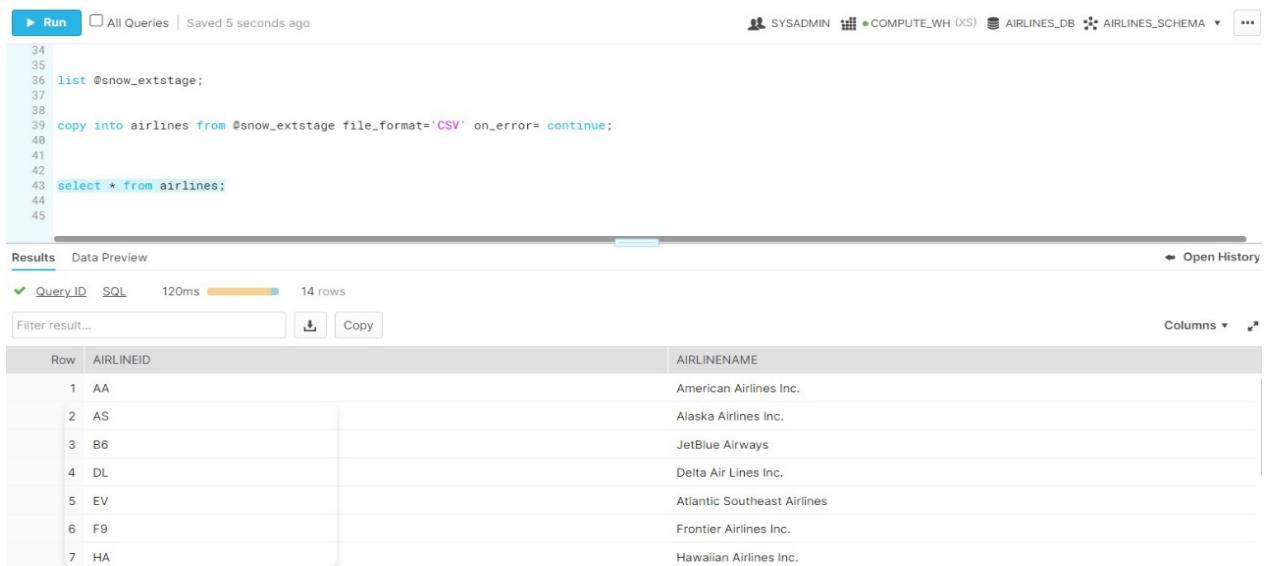
```

The results pane shows a single row from the Data Preview:

Row	name	size	md5	last_modified
1	s3://lti1034-praveenchandar/airlines.csv	359	7b5d753256599a82b9a98cc441421af0	Tue, 26 Jul 2022 07:47:12 GMT

- The data file uploaded in the external stage is listed using the ‘LIST’ command.

Task 6: Load the data from external stage



A screenshot of the Snowflake SQL interface. The query window shows the following SQL code:

```

34
35
36
37
38
39 copy into airlines from @snow_extstage file_format='CSV' on_error= continue;
40
41
42
43 select * from airlines;
44
45

```

The results pane shows the output of the SELECT * FROM airlines; query:

Row	AIRLINEID	AIRLINENAME
1	AA	American Airlines Inc.
2	AS	Alaska Airlines Inc.
3	B6	JetBlue Airways
4	DL	Delta Air Lines Inc.
5	EV	Atlantic Southeast Airlines
6	F9	Frontier Airlines Inc.
7	HA	Hawaiian Airlines Inc.

- The file in the external stage ‘snow_exstage’ is loaded into the ‘airlines’ table using ‘COPY INTO’ command.
- The file is loaded with the file format ‘CSV’ and the ‘on_error’ parameter is set to ‘continue’ in order to continue the loading process if any error occurs while loading a record.

- After uploading the data from the external stage, the table is verified by querying using ‘SELECT’ command.

Task 7: Perform bad records filtering on loading ,file format and compression

```

15 copy into flights(travelDate, dayOfWeek, airlineId, flightNumber, tailNumber, orgAirport, destAirport, scheduledDeparture, departureTime,
16     departureDelay, taxiOut, wheelsOff, scheduledTime, elapsedTime, airTime, distance, wheelsOn, taxiIn,
17     scheduledArrival, arrivalTime, arrivalDelay, diverted, cancelled, cancellationReason, airSystemDelay, securityDelay, airlineDelay,
18     lateAircraftDelay, weatherDelay) from
19     (select to_date(cast($1 as varchar(4)) || '-' || cast($2 as varchar(2)) || '-' || cast($3 as varchar(2))),
20         $4,$5,$6,NVL($7,'NA'),$8,$9,to_time(cast($10 as varchar(4)), 'HH24MI'),to_time(cast($11 as varchar(4)), 'HH24MI'),
21         $12,$13,to_time(cast($14 as varchar(4)), 'HH24MI'),$15,
22         $16,$17,$18,to_time(cast($19 as varchar(4)), 'HH24MI'),$20,to_time(cast($21 as varchar(4)), 'HH24MI'),to_time(cast($22 as varchar(4)), 'HH24MI'),
23         $23,$24,$25,NVL($26,'NA'),
24         ZEROIFNULL($27),ZEROIFNULL($28),ZEROIFNULL($29),ZEROIFNULL($30),ZEROIFNULL($31)
25         from @mystage/flightdata/flights.csv.gz
26     ) file_format=(format_name=csv) on_error=continue;
27

```

Results Data Preview Open History

✓ Query ID SQL 1m20s 1 rows

Filter result... Copy Columns ▾

Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_column_
1	mystage/flightd...	PARTIALLY_LO...	5819079	5813378	5819079	5701	Time '2400' is n...	978946	69	"FLIGHTS"["WH...

- The data file ‘flights.csv’ contains year,month and day data in separate columns in ‘number’ format. The year,month and day column is merged into a single column by type casting and concatenation. Then it is converted into ‘date’ format by the ‘to_date’ function.
- The data file ‘flights.csv’ contains time data (scheduledDeparture, departureTime, wheelsOff, scheduledArrival, arrivalTime, wheelsOn) in ‘number’ format. The time data in ‘number’ format is converted into ‘time’ format by using the ‘to_time’ function.
- The ‘tailNumber’ and ‘cancellationReason’ column is filled with ‘NA’ if it contains empty values.
- Out of ‘5819079’ records, ‘5701’ records are filtered because those records contain invalid time ‘24:00’.
- This transformation and filtering is done while loading data into the table from the named internal stage ‘mystage’.

Task 8a: Create snowpipes to continuously load flights data from external stage from s3 bucket to the flight staging table

Snowpipe enables loading data from files as soon as they're available in a stage. This means you can load data from files in micro-batches, making it available to users within minutes, rather than manually executing COPY statements on a schedule to load larger batches.

Automated data loads leverage event notifications for cloud storage to inform Snowpipe of the arrival of new data files to load. Snowpipe copies the files into a queue, from which they are loaded into the target table in a continuous, serverless fashion based on parameters defined in a specified pipe object.

Following are the steps:

1. Creating IAM roles

The screenshot shows the AWS IAM Roles page. The role 'Airlines_snowpipe' is selected. The 'Summary' section indicates that the role allows S3 to call AWS services on behalf of the user. The 'Permissions' tab is active, showing one attached policy: 'AmazonS3FullAccess'. This policy provides full access to all buckets via S3.

2. Creating storage integration in snowflake

```

46
47 create or replace storage integration s3_int type=external_stage storage_provider=s3 enabled=true
48
49 storage_aws_role.arn='arn:aws:iam::574997813424:role/Airlines_snowpipe'
50 storage_allowed_locations=('s3://lti1034-praveenchandar');
51
52 --arn:aws:iam::574997813424:role/Airlines_snowpipe
53 desc integration s3_int;
54
55

```

Results Data Preview

✓ Query_ID SQL 52ms 8 rows

Filter result... Copy

Row	property	property_type	property_value	property_default
1	ENABLED	Boolean	true *	false
2	STORAGE_PROVIDER	String	S3	
3	STORAGE_ALLOWED_LOCATIONS	List	s3://lti1034-praveenchandar	[]
4	STORAGE_BLOCKED_LOCATIONS	List		[]
5	STORAGE_AWS_IAM_USER_ARN	String	arn:aws:iam::411178425800:user/dzs9-s-sgsu6468	
6	STORAGE_AWS_ROLE_ARN	String	arn:aws:iam::574997813424:role/Airlines_snowpipe	
7	STORAGE_EXTERNAL_ID	String	NA69305_SFCRole=2_sjuqy+Ox+5Q6wTWXuZAGaeC0sU0=	
8	COMMENT	String		

3. Editing the trust relationship

The screenshot shows the AWS IAM Trust Relationships page for the 'Airlines_snowpipe' role. The left sidebar shows the IAM navigation menu. The main area displays the role's details, including its ARN and creation date. The 'Trust relationships' tab is selected, showing a JSON policy document. The policy allows the role to assume another role based on specific conditions, such as the external ID of the caller.

```

1- []
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Sid": "",
6-       "Effect": "Allow",
7-       "Principal": {
8-         "AWS": "arn:aws:iam::411178425800:user/dzs9-s-sgsu6468"
9-       },
10-      "Action": "sts:AssumeRole",
11-      "Condition": {
12-        "StringEquals": {
13-          "sts:ExternalId": "NA69305_SFCRole=2_Sj8m8IzLtrYBn+1qLl3p/HgWju="
14-        }
15-      }
16-    }
17-  ]
18- ]

```

4. Creating event notification

The screenshot shows the AWS S3 console with a green success message: "Successfully created event notification 'Airlines_event'. Operation successfully completed." Below this, the bucket overview is displayed for "lti1034-praveenchandar". Key details include:

- AWS Region: Asia Pacific (Singapore) ap-southeast-1
- Amazon Resource Name (ARN): arn:aws:s3:::lti1034-praveenchandar
- Creation date: July 18, 2022, 18:11:32 (UTC+05:30)

Bucket Versioning is enabled. The interface also shows tabs for Objects, Properties (which is selected), Permissions, Metrics, Management, and Access Points.

5. Creating an external stage

The screenshot shows the Snowflake SQL editor with the following query:

```
54
55 create or replace stage snowpipe_stage storage_integration=s3_int
56 url='s3://lti1034-praveenchandar/' file_format='CSV';
57
58
```

The results section shows a single row of data:

Row	status
1	Stage area SNOWPIPE_STAGE successfully created.

6. Creating a pipe with auto-ingest enabled

```

57
58
59 CREATE OR REPLACE PIPE snowpipe AUTO_INGEST=TRUE
60 AS copy into flightsDemo(travelDate, dayOfWeek, airlineId, flightNumber, tailNumber, orgAirport, destAirport, scheduledDeparture, departureTime,
61     departureDelay, taxiOut, wheelsOff, scheduledTime, elapsedTime, airTime, distance, wheelsOn, taxiIn,
62     scheduledArrival, arrivalTime, arrivalDelay, diverted, cancelled, cancellationReason, airSystemDelay, securityDelay, airlineDelay,
63     lateAircraftDelay, weatherDelay) from
64     (select to_date(cast($1 as varchar(4)) || '-' || cast($2 as varchar(2)) || '-' || cast($3 as varchar(2))),
65         $4,$5,$6,NVL($7,'NA'),$8,$9,$10,$11,$12,$13,$14,$15,
66         $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,NVL($26,'NA'),
67         ZEROIFNULL($27),ZEROIFNULL($28),ZEROIFNULL($29),ZEROIFNULL($30),ZEROIFNULL($31)
68     from @snowpipe_stage
69     ) file_format=(format_name=csv) on_error=continue;
70
71 show pipes;
72
73
74
75

```

Results Data Preview

Query_ID	SQL	56ms	1 rows								
	Filter result...										
Row	created_on	name	database_name	schema_name	definition	owner	notification_channel	comment	integration	pattern	error_integratio
1	2022-07-27 ...	SNOWPIPE	AIRLINES_DB	AIRLINES_S...	copy into flig...	ACCOUNTA...	arn:aws:sqs:ap-southea...		NULL	NULL	NULL

Task 8b: Maintain the change data capture and merge the data to the consumption table

A stream object records data manipulation language (DML) changes made to tables, including **inserts, updates, and deletes, as well as metadata** about each change, so that actions can be taken using the changed data. This process is referred to as **change data capture (CDC)**. An individual table stream tracks the changes made to rows in a source table. A stream makes a “change table” available of what changed, at the row level, between two transactional points of time in a table. This allows querying and consuming a sequence of change records in a transactional fashion.

A **Slowly Changing Dimension (SCD)** is a dimension that stores and manages both current and historical data over time in a data warehouse. It is considered one of the most **critical ETL tasks** in **tracking the history** of dimension records.

A **Type 2** SCD retains the full history of values. When the value of a chosen attribute changes, the current record is closed and a new record is created with the changed data values. This new record becomes the current record. Each record contains the effective time and expiration time to identify the time period between which the record was active.

1. Creating a stream on table

```

13 create or replace stream flights_stream on table flightsDemo;
14 show streams;

```

Results Data Preview Open History

✓ Query ID SQL 60ms 1 rows

Row	created_on	name	database_name	schema_name	owner	comment	table_name	source_type	base_tables	type	stale	mode
1	2022-07-28 ...	FLIGHTS_ST...	SNOWPIPE	PUBLIC	ACCOUNTA...		SNOWPIPE....	Table	SNOWPIPE....	DELTA	false	DEFAULT

- A stream named ‘flights_stream’ is created on the ‘flightsDemo’ table which captures any DML changes made to the table.

2. Uploading the data file to external stage

🕒 Upload succeeded
View details below.

Upload: status Close

The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://lti1034-praveenchandar/snowpipe/	1 file, 1.3 KB (100.00%)	0 files, 0 B (0%)

3. Table loaded through snowpipe

```

16 select * from flightsDemo;
17

```

Results Data Preview Open History

✓ Query ID SQL 1.69s 10 rows

Row	TRAVELEDATE	DAYOFWEEK	AIRLINEID	FLIGHTNUMBER	TAILNUMBER	ORGAIRPORT	DESTAIRPORT	SCHEDULEDDEI	DEPARTURETIME	DEPARTUREDEL	TAXIOUT	WHEELS
1	2015-01-01	4	AS	98	N407AS	ANC	SEA	5	2354	-11	21	
2	2015-01-01	4	AA	2336	N3KUAA	LAX	PBI	10	2	-8	12	
3	2015-01-01	4	US	840	N171US	SFO	CLT	20	18	-2	16	
4	2015-01-01	4	AA	258	N3HYAA	LAX	MIA	20	15	-5	15	
5	2015-01-01	4	AS	135	N527AS	SEA	ANC	25	24	-1	11	

- As the file is uploaded in the external stage, data is loaded according to the ‘COPY’ statement defined in the ‘snowpipe’.

4. Data change captured in the stream

```
17
18 select * from flights_stream;
19
```

Results Data Preview [Open History](#)

✓ Query ID SQL 141ms 10 rows

Filter result... [Copy](#) Columns ▾

IME	ARRIVALDELAY	DIVERTED	CANCELLED	CANCELLATION	AIRSYSTEMDEL	SECURITYDEL	AIRLINEDELAY	LATEAIRCRAFT	WEATHERDEL	METADATA\$AC	METADATA\$ISU	METADATA\$ROV
408	-22	0	0	NA	0	0	0	0	0	INSERT	FALSE	6358a34d87...
741	-9	0	0	NA	0	0	0	0	0	INSERT	FALSE	d3a844f057...
811	5	0	0	NA	0	0	0	0	0	INSERT	FALSE	51167498a7...
756	-9	0	0	NA	0	0	0	0	0	INSERT	FALSE	495b50e6f0...
259	-21	0	0	NA	0	0	0	0	0	INSERT	FALSE	164d0a082c...
810	0	0	0	NA	0	0	0	0	0	INSERT	FALSE	cc170444ed...

- The stream ‘flights_stream’ captures the insertion data of the ‘flightDemo’ table.

5. Creation of task

```
26 create or replace task merge_flights_history warehouse=compute_wh schedule='1 minute'
27 when system$stream_has_data('flights_stream')
28 as
29 merge into "SNOWPIPE"."PUBLIC"."FLIGHTS_HISTORY" fh using flights_stream fs
30 on fh.travelDate=fs.travelDate when not matched and fs.metadata$action='INSERT' then
31   insert(travelDate , dayOfWeek, airlineId, flightNumber, tailNumber, orgAirport, destAirport,typeOfUpdate,timeOfUpdate)
32   values(fs.travelDate , fs.dayOfWeek, fs.airlineId, fs.flightNumber, fs.tailNumber, fs.orgAirport, fs.destAirport,
```

Results Data Preview [Open History](#)

✓ Query ID SQL 76ms 1 rows

Filter result... [Copy](#) Columns ▾

Row	status
1	Task MERGE_FLIGHTS_HISTORY successfully created.

- A task named ‘merge_flights_history’ is created which merges any changes made in the ‘flightsDemo’ table into ‘flights_history’ table using the change data available in the stream ‘flights_stream’.

6. Running the task

```

36 alter task merge_flights_history resume;
37
38 show tasks;
39

```

Results Data Preview [Open History](#)

✓ Query ID SQL 52ms 1 rows

schema_name	owner	comment	warehouse	schedule	predecessors	state	definition	condition	allow_overlappin	error_integration	last_committed_	last_su...
PUBLIC	ACCOUNTA...		COMPUTE_...	1 minute	[]	started	merge into "...	system\$stre...	false	null	2022-07-28 ...	

7. Merged-data

```

39
40 select * from flights_history;

```

Results Data Preview [Open History](#)

✓ Query ID SQL 163ms 10 rows

Row	TRAVELEDATE	DAYOFWEEK	AIRLINEID	FLIGHTNUMBER	TAILNUMBER	ORGAIRPORT	DESTAIRPORT	TYPEOFUPDATE	TIMEOFUPDATE
1	2015-01-01	4	AS		98 N407AS	ANC	SEA	INSERT	2022-07-28 04:4...
2	2015-01-01	4	AA		2336 N3KUAA	LAX	PBI	INSERT	2022-07-28 04:4...
3	2015-01-01	4	US		840 N171US	SFO	CLT	INSERT	2022-07-28 04:4...
4	2015-01-01	4	AA		258 N3HYAA	LAX	MIA	INSERT	2022-07-28 04:4...
5	2015-01-01	4	AS		135 N527AS	SEA	ANC	INSERT	2022-07-28 04:4...
6	2015-01-01	4	DL		806 N3730B	SFO	MSP	INSERT	2022-07-28 04:4...

- After resuming the task, data gets loaded into the ‘flights_history’ table from the stream ‘flights_stream’.

8. After merge

```

41
42 select * from flights_stream;

```

Results Data Preview [Open History](#)

✓ Query ID SQL 1.31s 0 rows

Row	TRAVELEDATE	DAYOFWEEK	AIRLINEID	FLIGHTNUMBER	TAILNUMBER	ORGAIRPORT	DESTAIRPORT	SCHEDULEDDEI	DEPARTURETIM	DEPARTUREDEL	TAXIOUT	WHEELS
-----	-------------	-----------	-----------	--------------	------------	------------	-------------	--------------	--------------	--------------	---------	--------

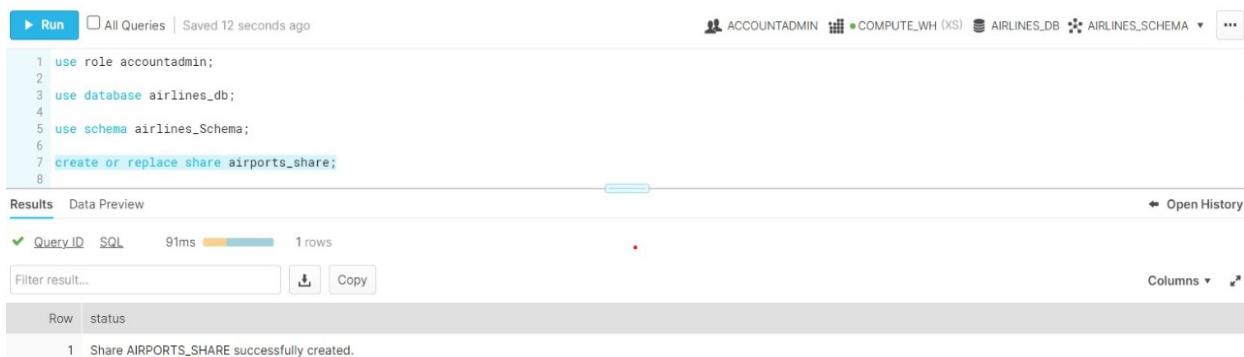
- After merging the data into the ‘flights_history’ table, the change data in the stream ‘flights_stream’ gets purged.

Task 9: Share your table to new non-snowflake user

Reader accounts provide a quick, easy, and cost-effective way to share data without requiring the consumer to become a Snowflake customer.

Each reader account belongs to the provider account that created it. Similar to standard consumer accounts, the provider account uses shares to share databases with reader accounts; however, a reader account can only consume data from the provider account that created it.

1. Creating outbound share



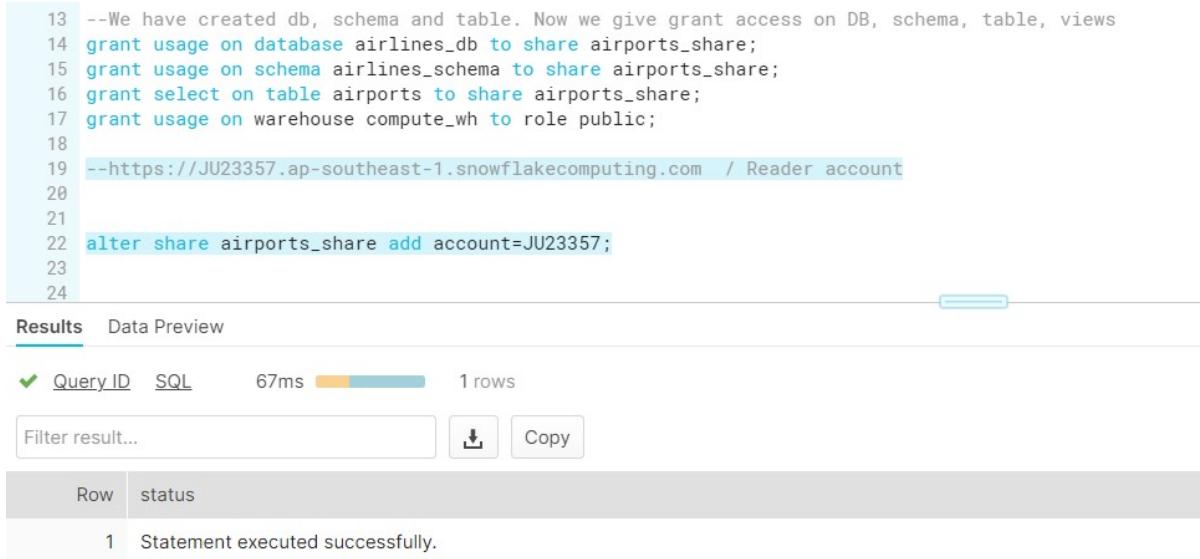
The screenshot shows a Snowflake SQL interface. The top navigation bar includes 'Run' (highlighted in blue), 'All Queries' (with a note 'Saved 12 seconds ago'), and account information ('ACCOUNTADMIN', 'COMPUTE_WH (XS)', 'AIRLINES_DB', 'AIRLINES_SCHEMA'). Below the navigation is a code editor with the following SQL script:

```
1 use role accountadmin;
2
3 use database airlines_db;
4
5 use schema airlines_schema;
6
7 create or replace share airports_share;
```

The results pane shows a single row of output:

Row	status
1	Share AIRPORTS_SHARE successfully created.

2. Granting privileges to the share



The screenshot shows a Snowflake SQL interface. The top navigation bar includes 'Run' (highlighted in blue), 'All Queries' (with a note 'Saved 12 seconds ago'), and account information ('ACCOUNTADMIN', 'COMPUTE_WH (XS)', 'AIRLINES_DB', 'AIRLINES_SCHEMA'). Below the navigation is a code editor with the following SQL script:

```
13 --We have created db, schema and table. Now we give grant access on DB, schema, table, views
14 grant usage on database airlines_db to share airports_share;
15 grant usage on schema airlines_schema to share airports_share;
16 grant select on table airports to share airports_share;
17 grant usage on warehouse compute_wh to role public;
18
19 --https://JU23357.ap-southeast-1.snowflakecomputing.com / Reader account
20
21
22 alter share airports_share add account=JU23357;
23
24
```

The results pane shows a single row of output:

Row	status
1	Statement executed successfully.

3. Creating a Reader account

The screenshot shows the Snowflake web interface with the URL na69305.ap-southeast-1.snowflakecomputing.com/console#/account/reader-accounts. The user is creating a new Reader account named "AirlinesReader". The dialog box includes fields for Account Name, Comments, Edition (ENTERPRISE), and Region (Singapore). It also contains Admin Login Information fields for User Name, Password, and Confirm Password. A "Show SQL" link is available at the bottom left. The "Create Account" button is at the bottom right.

The screenshot shows the Snowflake web interface with the URL na69305.ap-southeast-1.snowflakecomputing.com/console#/account/reader-accounts. A "Congratulations!" dialog box is displayed, stating "You have created a new reader account 'AIRLINESREADER'." It provides the Account URL (<https://ju23357.ap-southeast-1.snowflakecomputing.com>) and Locator (JU23357). Below this, a "What's next?" section lists two steps: visiting the secure shares page to create a share and logging into the account. A "Done" button is at the bottom right.

The screenshot shows the Snowflake web interface with the URL <https://na69305.ap-southeast-1.snowflakecomputing.com/console#/account/reader-accounts>. The top navigation bar includes links for Email, Meeting, Micros, Reader Account, Shares, Worksheet, IAM Manager, Project Glance, and various status indicators. The main header features the Snowflake logo and the text "Enjoy your free trial! Visit our documentation to learn more about using Snowflake or contact our support team with any questions." The user is identified as "PRAVEENCHANDAR ACCOUNTADMIN". The left sidebar has sections for Account, Usage, Billing, Users, Roles, Policies, Sessions, Resource Monitors, and Reader Accounts (which is selected). Below this, the "Reader Accounts" section displays two entries:

Account Name	Locator	Date Created	Account URL	Comments
PRAVEENCHANDAR	V57125	7/19/2022, 10:51:24 AM	https://ve57125.ap-southeast-1.snowflakecomputing.com	
AIRLINESREADER	RL90061	2:49:17 PM	https://rl90061.ap-southeast-1.snowflakecomputing.com	

The bottom status bar shows the weather as "32°C Partly sunny" and the system time as "2:59 PM 7/27/2022".

4. Reading the database from the reader account

The screenshot shows the Snowflake web interface with the URL <https://ju23357.ap-southeast-1.snowflakecomputing.com/console#/shares>. The top navigation bar and user information are identical to the previous screenshot. The main header shows the user "AIRLINESREADER ACCOUNTADMIN". The left sidebar has sections for Reader, Databases, Shares (which is selected), Marketplace, Warehouses, Worksheets, History, and Account. The "Shares" section displays an inbound secure share:

Secure Share Name	Shared By	Database	Creation Time	Owner	Comment
AIRPORTS_SHARE	ZY26775	AIRLINESREADER_DB	7/27/2022	AIRLINESREADER	Type: Inbound
ACCOUNT_USAGE	SNOWFLAKE				Creation Time: 7/27/2022

A modal window titled "Database Overview" is open, showing details for the "AIRPORTS_SHARE" secure share:

- Secure Share: AIRPORTS_SHARE
- Database: AIRLINESREADER_DB
- Roles with access: 2 granted (ACCOUNTADMIN and SYSADMIN)

The bottom status bar shows the weather as "32°C Partly sunny" and the system time as "3:27 PM 7/27/2022".

The screenshot shows the Snowflake Worksheet interface. The left sidebar displays the database schema with the AIRLINESREADER_DB selected, containing the AIRLINES_SCHEMA which has the AIRPORTS table. A query is run in the worksheet:

```

1 use warehouse compute_wh;
2
3
4 select * from airports;
    
```

The results show 322 rows of airport data:

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Mumbai	PA	USA	10.12345	20.12345
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.6819
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919
4	ABR	Aberdeen Regional Airp...	Aberdeen	SD	USA	30.12345	40.12345
5	ABY	Southwest Georgia Regi...	Albany	GA	USA	31.53552	-84.19447
6	ACK	Nantucket Memorial Air...	Nantucket	MA	USA	41.25305	-70.06018
7	ACT	Waco Regional Airport	Waco	TX	USA	31.61129	-97.23052
8	ACV	Arcata Airport	Arcata/Eureka	CA	USA	40.97812	-124.10862
9	ACY	Atlantic City Internation...	Atlantic City	NJ	USA	39.45758	-74.57717
10	ADK	Adak Airport	Adak	AK	USA	51.87796	-176.64603
11	ADQ	Kodiak Airport	Kodiak	AK	USA	57.74997	-152.49386
12	AEX	Alexandria International ...	Alexandria	LA	USA	31.32737	-92.54856

Task 10: create clone of the table with time travel before one day and write query to get history data using particular timestamp

Cloning, also referred to as “zero-copy cloning”, creates a copy of a database, schema or table. A snapshot of data present in the source object is taken when the clone is created and is made available to the cloned object. It just updates the metadata which references the source data and prevents creation of physical copy of the data.

Snowflake Time Travel enables accessing historical data (i.e. data that has been changed or deleted) at any point within a defined period. It serves as a powerful tool for performing the following tasks:

- Restoring data-related objects (tables, schemas, and databases) that might have been accidentally or intentionally deleted.
- Duplicating and backing up data from key points in the past.
- Analysing data usage/manipulation over specified periods of time.

The steps to create cloned table is as follows:

```
10
11
12
13
14
15
16
17
18
19
20
```

Results Data Preview

✓ Query ID SQL 1.5s 322 rows

Filter result... Copy

Columns ▾

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Allentown	PA	USA	40.65236	-75.44040
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919
4	ARP	Aberdeen Regional Airc...	Aberdeen	SD	USA	45.11006	-98.12123

1. Creating a clone of the table using time-travel

```
16
17
18
19
20
```

Results Data Preview

✓ Query ID SQL 1.22s 1 rows

Filter result... Copy

Row status

1 Table RESTORED_AIRPORT successfully created.

```
18
19
20
```

Results Data Preview

✓ Query ID SQL 617ms 322 rows

Filter result... Copy

Column

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Allentown	PA	USA	40.65236	-75.44040
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919
4	ARP	Aberdeen Regional Airc...	Aberdeen	SD	USA	45.11006	-98.12123

2. Changing the data in the table

```
10
11
12
13
14
15
16
17
18
19
20
```

Results Data Preview

✓ Query ID SQL 905ms 1 rows

Filter result... Copy

Row	number of rows updated
1	1

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Mumbai	PA	USA	10.12345	20.12345
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919

3. Retrieving the previous data using offset

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Allentown	PA	USA	40.65236	-75.44040
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919
4	ABR	Aberdeen Regional Airp...	Aberdeen	SD	USA	45.44906	-98.42183

Row
1

number of rows deleted 1

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Mumbai	PA	USA	10.12345	20.12345
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABR	Aberdeen Regional Airp...	Aberdeen	SD	USA	45.44906	-98.42183

Status	Query ID	SQL Text	User	Warehouse	Clust...	Size	Session ID	Start Time	End Time	Total Duration
✓	01a5f589-0000-3299-0003-cc7200013...	delete from restored_...	SEJALJADH...	COMPUTE_...	1	X-Small	1069214928...	1:23:20 AM	1:23:21 AM	1.6s

4. Retrieving the data using QueryId

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Mumbai	PA	USA	10.12345	20.12345
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919

Status	Query ID	SQL Text	User	Warehouse	Clust...	Size	Session ID	Start Time	End Time	Total Duration
This data was last updated at 1:29:32 AM. Press ⌂ at the top right to get the latest data. Close										
✓	01a5f581-0000-3295-0003-cc7200014...	update restored_airpo...	SEJALJADH...	COMPUTE_...	1	X-Small	1069214928...	1:15:30 AM	1:15:31 AM	905ms

33 `select * from restored_airport BEFORE (statement=>'01a5f581-0000-3295-0003-cc7200014aa');`

Results Data Preview [Open History](#)

✓ Query ID SQL 127ms 322 rows [Copy](#)

Filter result... [Columns](#)

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Allentown	PA	USA	40.65236	-75.44040
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919
4	ABR	Aberdeen Regional Airp...	Aberdeen	SD	USA	45.44906	-98.42183

37 `select sysdate(); --2022-07-30 20:04:39.282`

38 `select * from restored_airport AT (timestamp=> '2022-07-30 20:04:39' ::timestamp);`

Results Data Preview [Open History](#)

✓ Query ID SQL 1.04s 321 rows [Copy](#)

Filter result... [Columns](#)

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Mumbai	PA	USA	10.12345	20.12345
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABR	Aberdeen Regional Airp...	Aberdeen	SD	USA	45.44906	-98.42183

5. Query to get history data using particular timestamp

41 `select * from restored_airport AT (timestamp=> '2022-07-30 19:45:20.796' ::timestamp);`

Results Data Preview [Open History](#)

✓ Query ID SQL 252ms 322 rows [Copy](#)

Filter result... [Columns](#)

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Allentown	PA	USA	40.65236	-75.44040
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919
4	ABR	Aberdeen Regional Airp...	Aberdeen	SD	USA	45.44906	-98.42183

38 `select * from restored_airport;`

Results Data Preview [Open History](#)

✓ Query ID SQL 32ms 321 rows [Copy](#)

Filter result... [Columns](#)

Row	AIRPORTID	AIRPORTNAME	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
1	ABE	Lehigh Valley Internatio...	Mumbai	PA	USA	10.12345	20.12345
2	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
3	ABR	Aberdeen Regional Airp...	Aberdeen	SD	USA	45.44906	-98.42183
4	ABQ	Albuquerque Internation...	Albuquerque	NM	USA	35.04022	-106.60919

Task 11: Create stored procedure to insert the data into table after typecasting date format

Stored procedures allow you to write procedural code that executes SQL. In a stored procedure, you can use programmatic constructs to perform branching and looping.

1. Creating a stored procedure

```
9
10 create or replace procedure date_format(newdate varchar)
11 returns varchar as
12 $$begin
13   insert into flightsDemo values(to_date(:newdate,'YYYY-MM-DD'),1,'WN',7070,'N255WN','MAA','LAX',1615,1613,-2,7,1628,115,106,94,616,1654,5,1718,1659,-11,
14   0,0,'NA',0,0,0,0);
15   return 'success';
16 end
17 $$;
```

Results Data Preview Open History

✓ Query ID SQL 84ms 1 rows

Filter result... Copy Columns ▾

Row	status
1	Function DATE_FORMAT successfully created.

- A stored procedure named ‘date_format’ is created which takes a date in varchar format as an argument.
- This procedure uses Snowflake scripting to insert a record into the ‘flightsDemo’ table by typecasting the argument into date format using the ‘to_date’ function.

2. Calling the stored procedure

```
20 call date_format('2015-05-01');
21
22 select * from flightsDemo;
```

Results Data Preview Open History

✓ Query ID SQL 174ms 1 rows

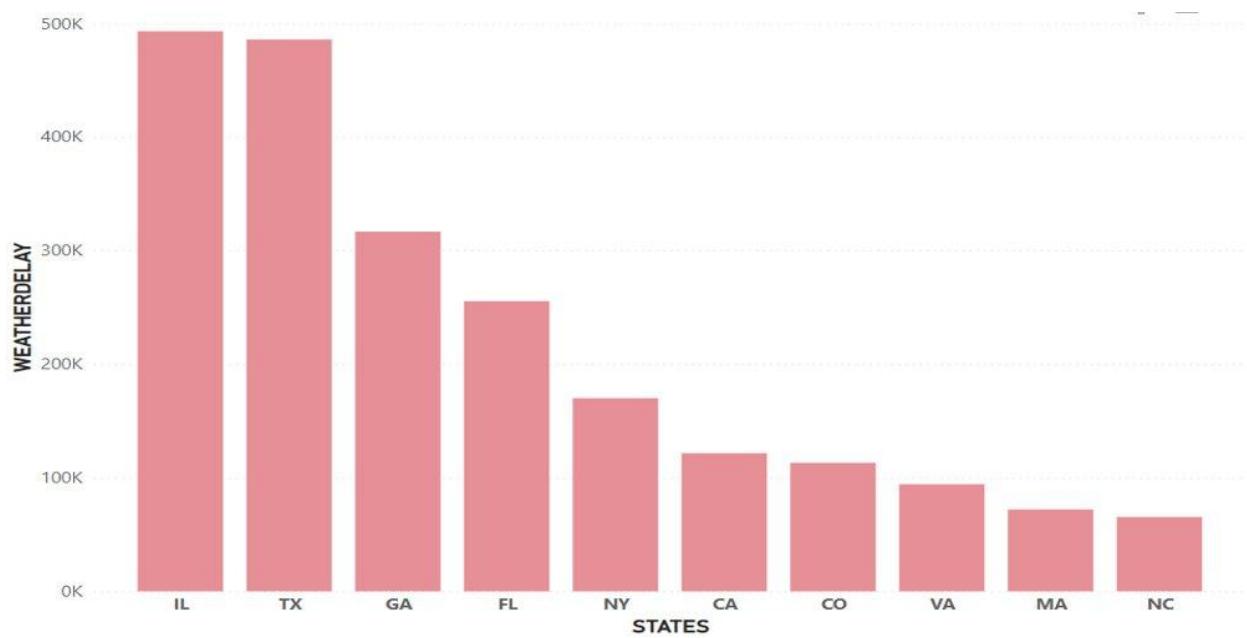
Filter result... Copy Columns ▾

Row	TRAVELDATE	DAYOFWEEK	AIRLINEID	FLIGHTNUMBER	TAILNUMBER	ORGairport	DESTAIRPORT	SCHEDULEDDEI	DEPARTURETIM	DEPARTUREDEL	TAXIOUT	WHEELS
1	2015-05-01	1	WN	7070	N255WN	MAA	LAX	1615	1613	-2	7	.

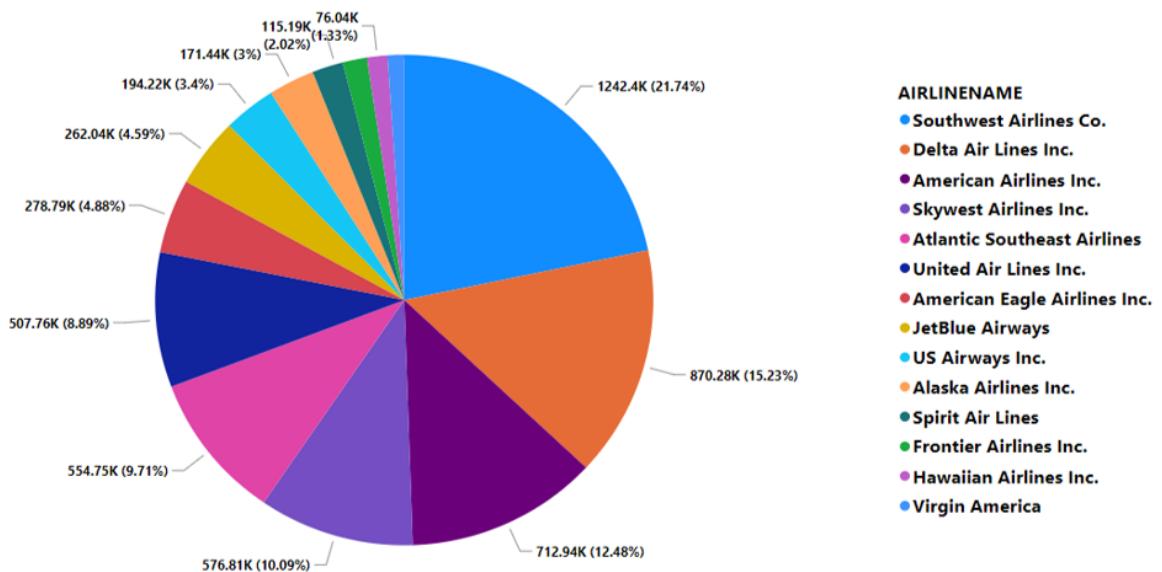
- After the stored procedure is called, a record is inserted into the ‘flightsDemo’ table.

Task 12: Create a visualisation for flight delay analysis using powerBI and prepare reports /dashboard for business queries

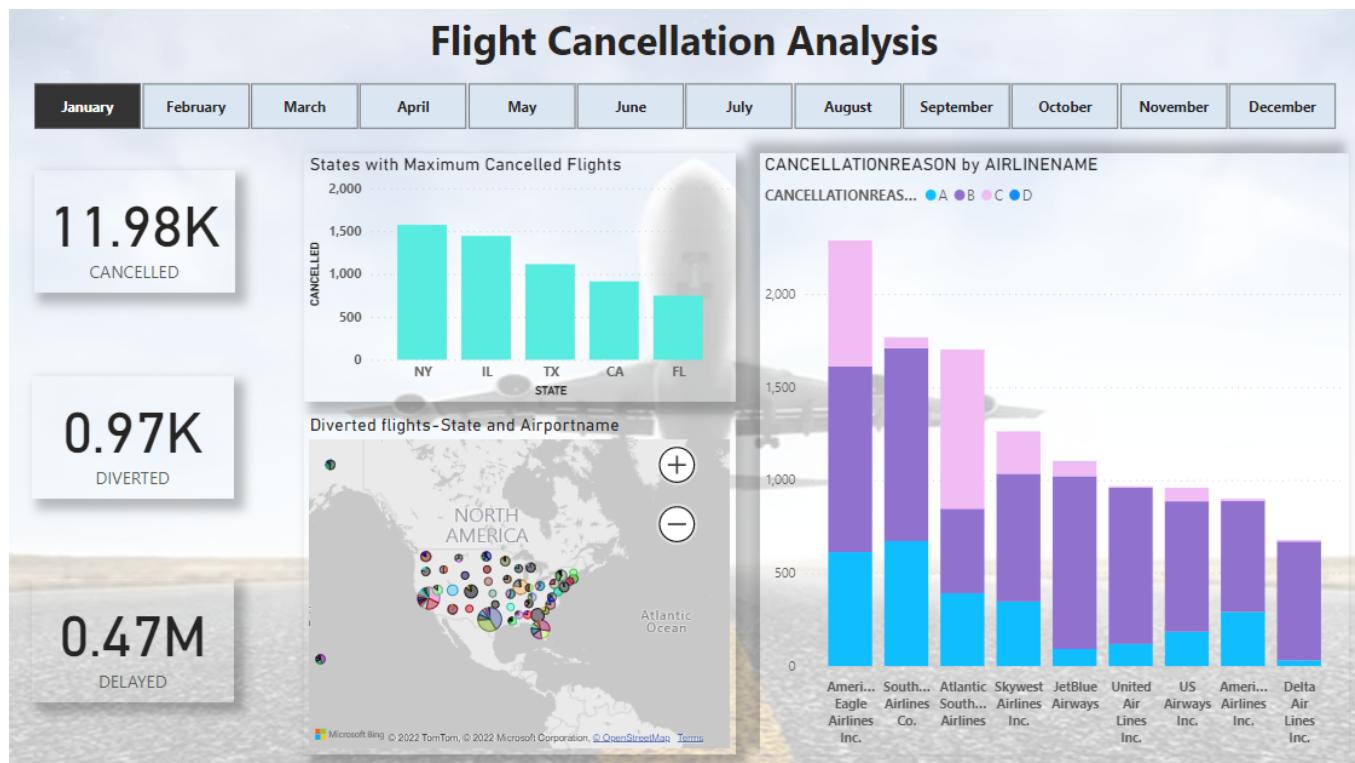
Top 10 States with weather delay



Arrival Delay by Airline Name



Flight Cancellation Analysis



Flight Delay Analysis

