**Snowflake Data Loading using all File Formats**

<https://uaubrbj-yb48348.snowflakecomputing.com/console#/internal/worksheet>

SAARA

Saaranya@123

# 1.FLAT FILES or HUMAN READABLE FILES using CSV,PSV,TSV and Compression gzip files :

## 1.1 CSV - Using Snowflake WebUI

### Loading with good data

### When comma,double quotes in data

### Date and Timestamp issues

### Special Character Data Loading

### Column Count Mismatch & Data Loading

### Null Handling Into Snowflake

### Duplicate Data Loading

## 1.2 CSV - Using Snowsql CLI

## 1.3 CSV - Using External Storage (AWS S3,Azure,GCP)

## CSV - Using Snowflake WebUI

### Loading with good data

Query to create table with all data types which includes number,varchar,date,Boolean, timestamp\_ltz

**create** **or** **replace** transient **table** customer\_csv (

customer\_pk number(38,0),

salutation varchar(10),

first\_name varchar(20),

last\_name varchar(30),

gender varchar(1),

marital\_status varchar(1),

day\_of\_birth date,

birth\_country varchar(60),

email\_address varchar(50),

city\_name varchar(60),

zip\_code varchar(10),

country\_name varchar(20),

gmt\_timezone\_offset number(10,2),

preferred\_cust\_flag boolean,

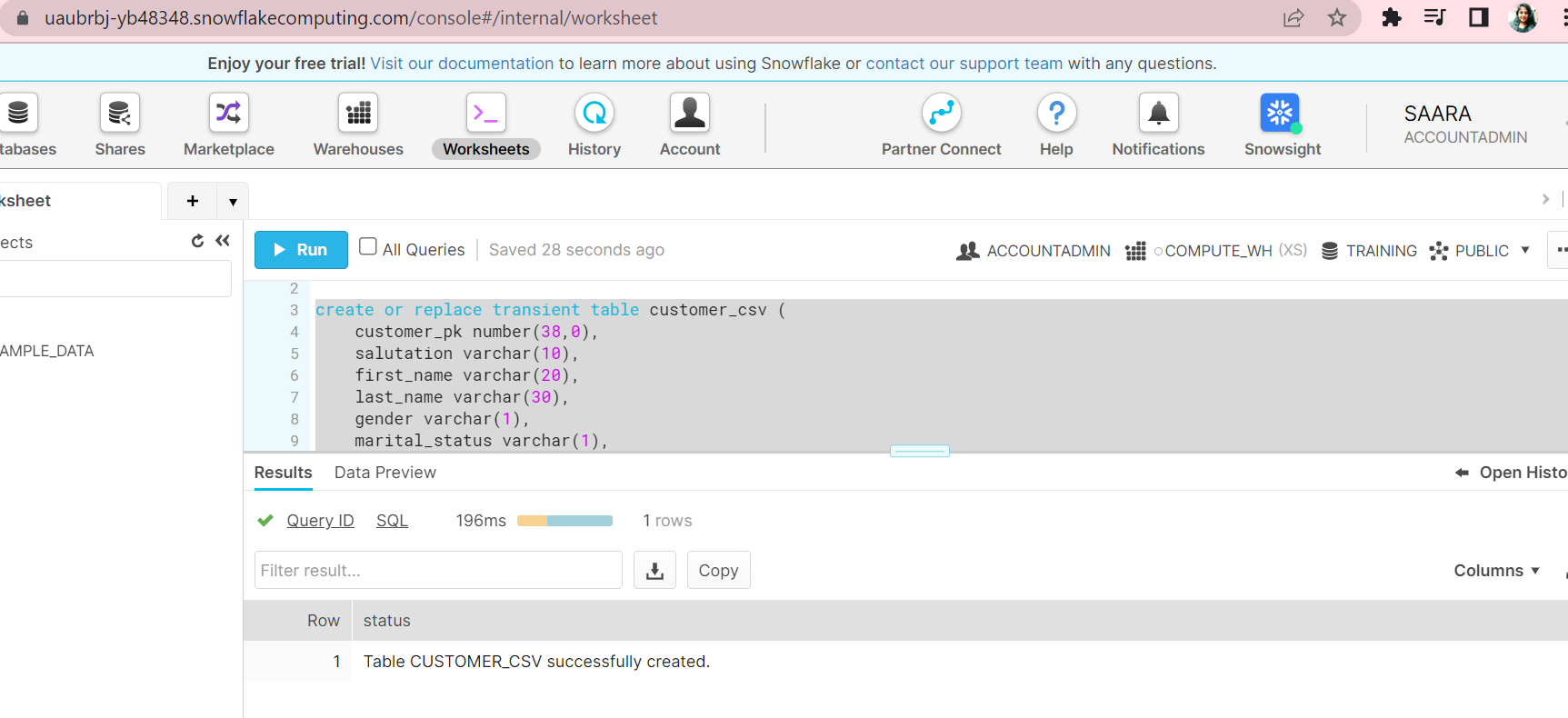
registration\_time timestamp\_ltz(9)

);

Why creating as Transient table instead of temporary and permanent ?

Transient tables are similar to permanent tables with the key difference that the fail-safe storage is not applicable to transient tables.   
Because transient tables do not have a Fail-safe period, they provide a good option for managing the cost of large tables used to store transformation data. However, with an underlying risk that data cannot be recovered after the Time Travel retention period passes. Temporary tables do not support some standard features such as cloning.

Table CUSTOMER\_CSV created successfully.



Query to create file format:

**create** **or** **replace** file format customer\_csv\_ff

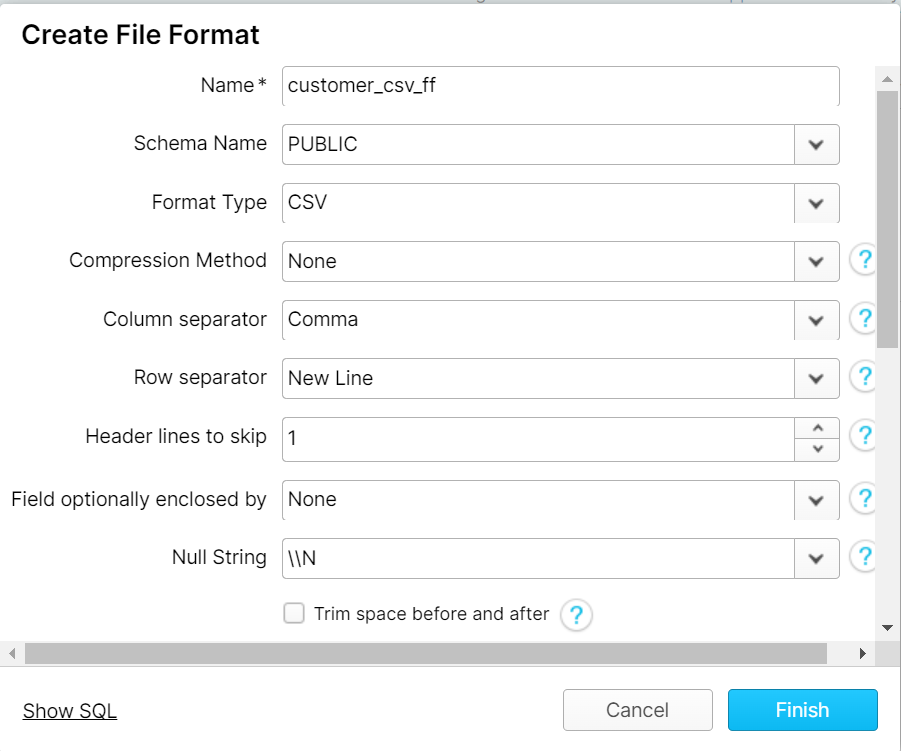
**type** **=** 'csv'

compression **=** 'none'

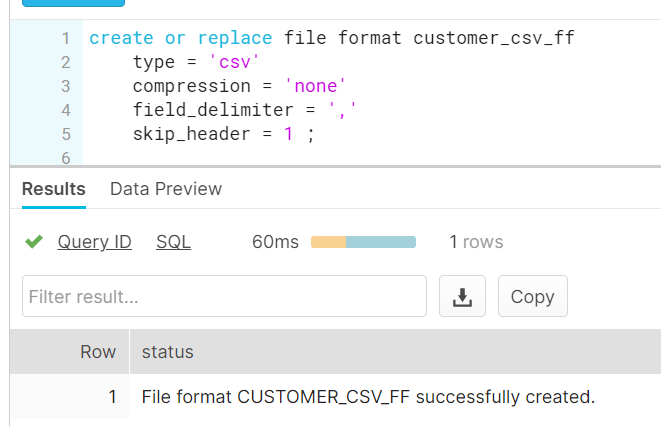
field\_delimiter **=** ','

skip\_header **=** 1 ;

Same using WebUI Classic Console :

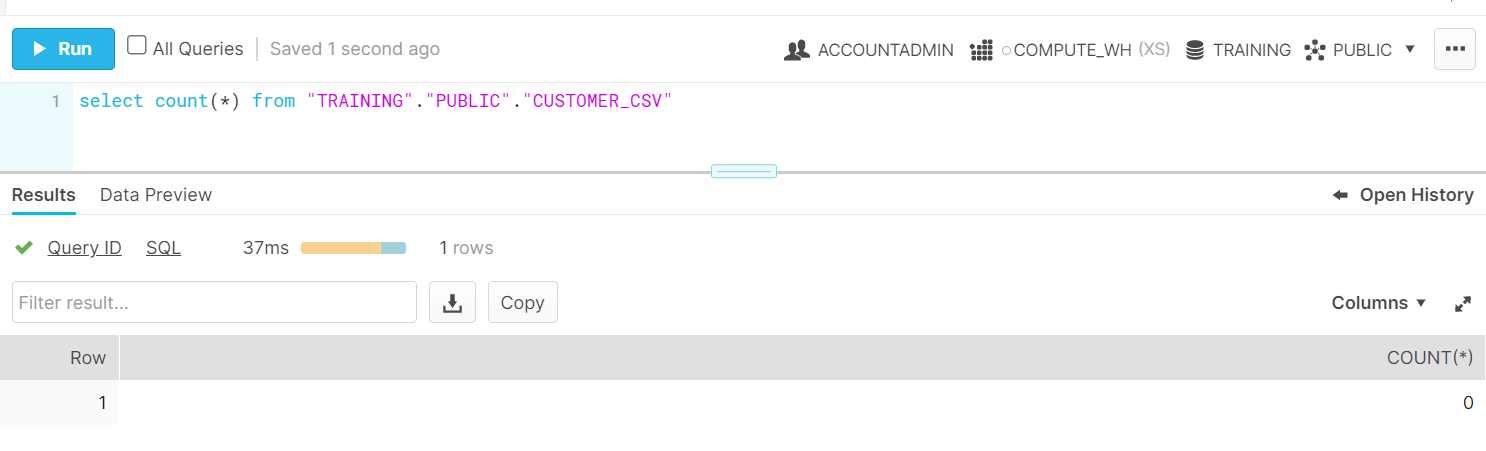


File format Created successfully:

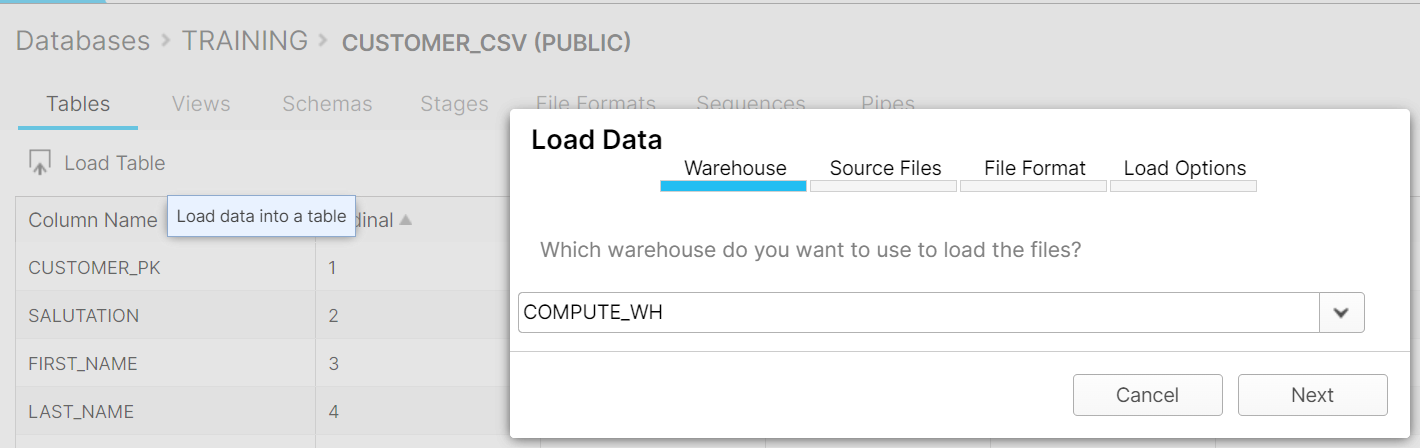


Now loading starts:

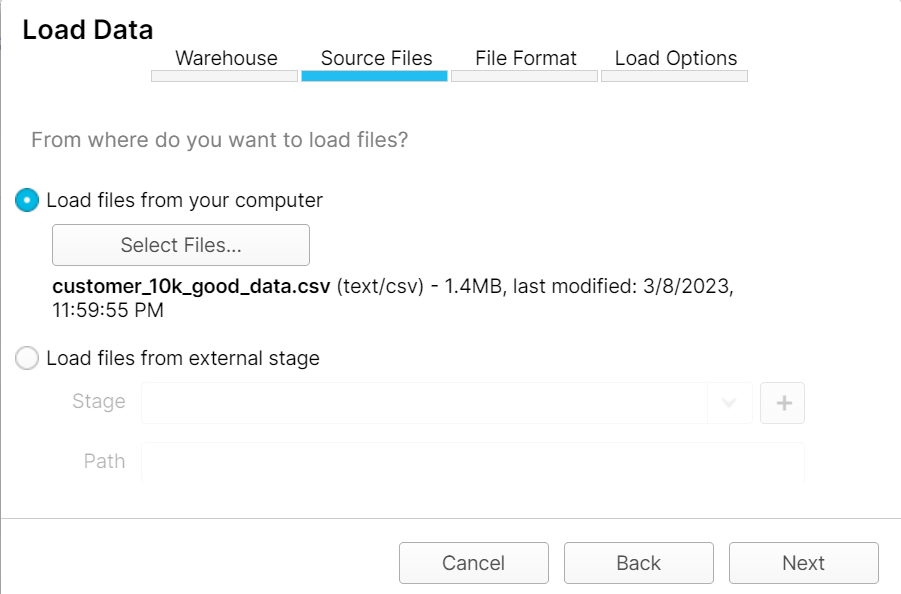
Count is zero before loading:



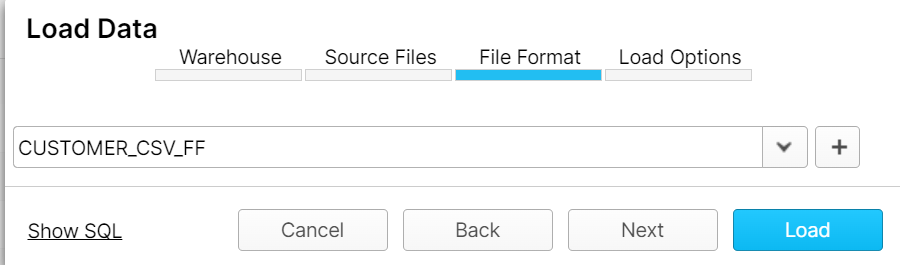
Using WebUI :

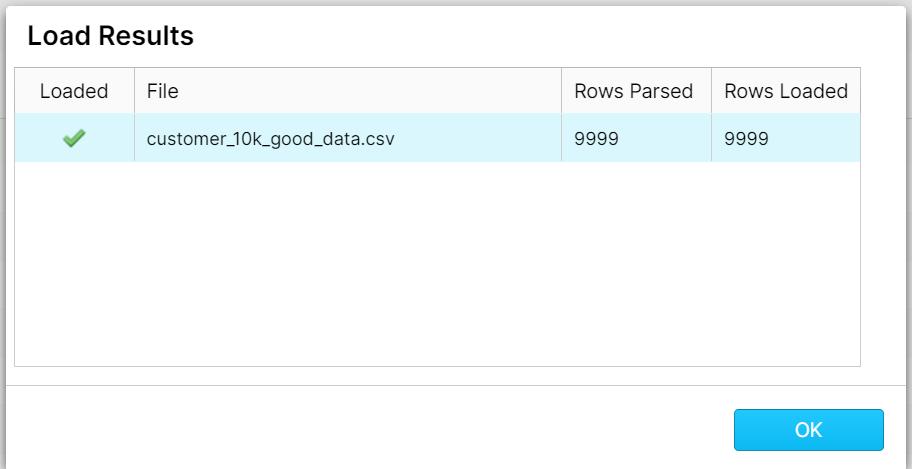


Choose file from local drive :

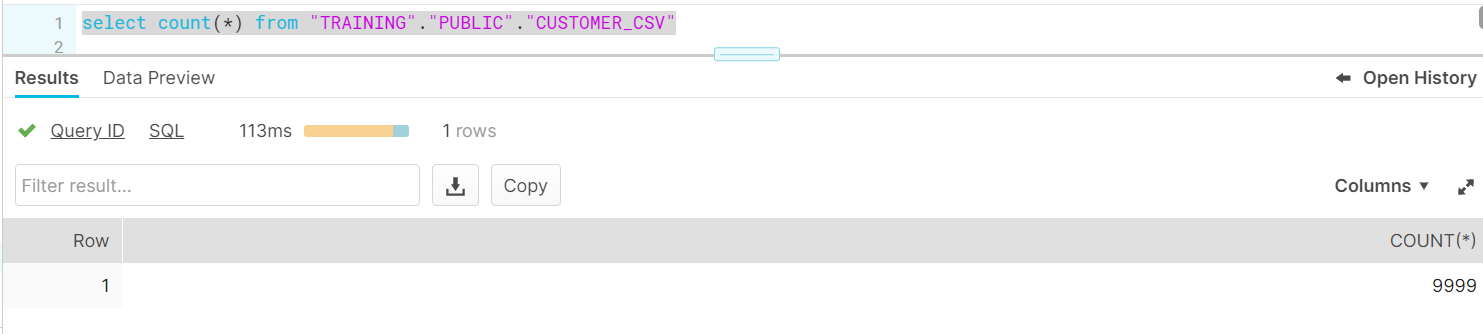


Choose file format :



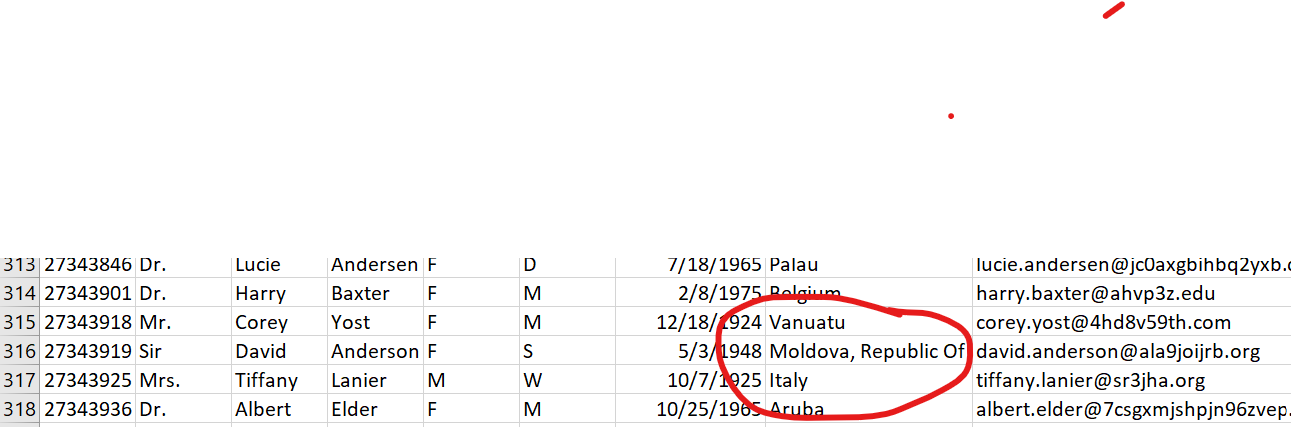


9999 records loaded successfully to table(excluding header)



### When comma,double quotes in data

Now add comma in the data file:



Change file format with field\_optionally\_enclosed\_by = '\042'

create or replace file format customer\_csv\_ff2

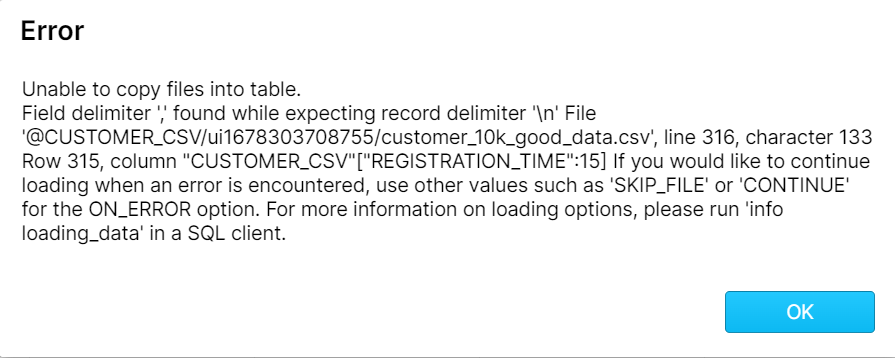
type = 'csv'

compression = 'none'

field\_delimiter = ','

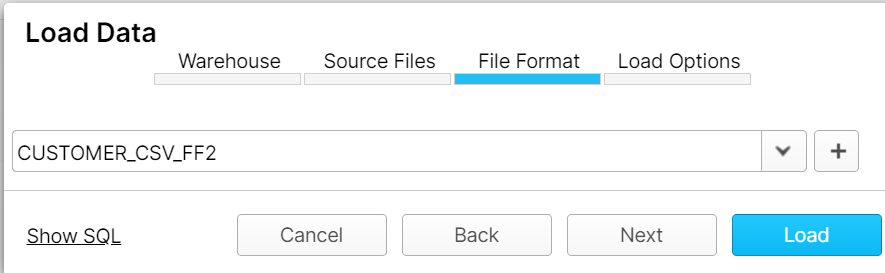
field\_optionally\_enclosed\_by = '\042'

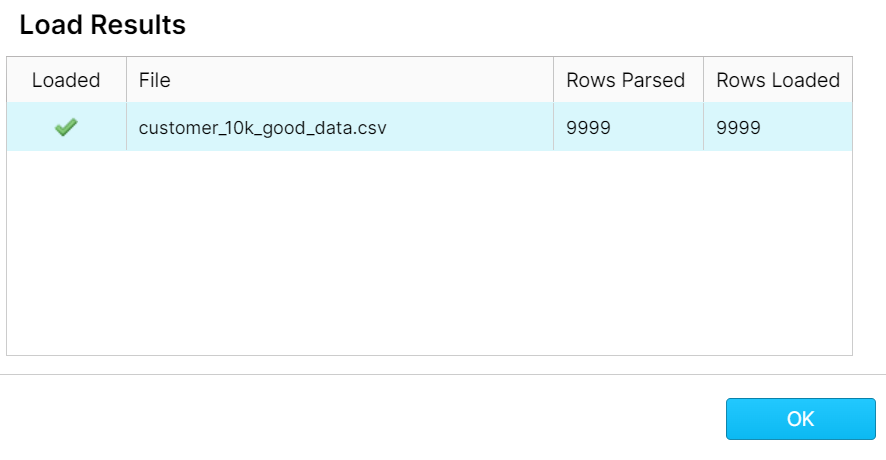
skip\_header = 1 ;



Loading after updating file format :

Choose new file format from the drop down below :





Date and Timestamp issues :

*-- Step-01 : lets start with a table called bookings*

*-- it has 5 columns, accomodate data typles like date, time, datetime & timestamp.*

**create** **or** **replace** transient **table** bookings(

booking\_id number,

booking\_dt date,

booking\_time time,

booking\_dt\_time datetime,

booking\_timestamp timestamp

);

*-- csv file format call csv\_ff with standard paramters.*

*-- no additional paramter for date and time is used to*

*-- see how does it work with default settings.*

**create** **or** **replace** file format csv\_ff

**type** **=** 'csv'

compression **=** 'none'

field\_delimiter **=** ','

record\_delimiter **=** '\n'

skip\_header **=** 1

field\_optionally\_enclosed\_by **=** '\042';

Input File :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| booking\_id | booking\_dt | booking\_time | booking\_dt\_time | booking\_timestamp |
| 1 | 8/29/2022 | 23:59:59 | 8/29/2022 23:59 | 2022-08-29 23:59:59 +01:00 |
| 2 | 9/29/2022 | 23:59:59 | 9/29/2022 23:59 | 2022-09-29 23:59:59 +02:00 |
| 3 | 10/29/2022 | 23:59:59 | 10/29/2022 23:59 | 2022-10-29 23:59:59 +03:00 |
| 4 | 11/29/2022 | 23:59:59 | 11/29/2022 23:59 | 2022-11-29 23:59:59 +04:00 |
| 5 | 12/29/2022 | 23:59:59 | 12/29/2022 23:59 | 2022-12-29 23:59:59 +05:00 |

PUT file://<file\_path>/01-bookings-std.csv @BOOKINGS/ui1678706575840

COPY INTO "TRAINING"."PUBLIC"."BOOKINGS" FROM @/ui1678706575840 FILE\_FORMAT = '"TRAINING"."PUBLIC"."CSV\_FF"' ON\_ERROR = 'ABORT\_STATEMENT' PURGE = TRUE;

Table

Description automatically generated

Graphical user interface, application

Description automatically generated

There is no timestamp for milli and nano seconds in the field BOOKING\_DT\_TIME and BOOKING\_TIMESTAMP in input file, but the table is loaded with milli and nano seconds by default.

Alter statement to change timestamp in session:

**alter** **session** **set** TIMESTAMP\_NTZ\_OUTPUT\_FORMAT **=** 'YYYY-MM-DD HH24:MI:SS.FF9';

## Loading a TSV File using WebUI:

Same customer table created with name \_tsv :

create or replace transient table customer\_tsv (

customer\_pk number(38,0),

salutation varchar(10),

first\_name varchar(20),

last\_name varchar(30),

gender varchar(1),

marital\_status varchar(1),

day\_of\_birth date,

birth\_country varchar(60),

email\_address varchar(50),

city\_name varchar(60),

zip\_code varchar(10),

country\_name varchar(20),

gmt\_timezone\_offset number(10,2),

preferred\_cust\_flag boolean,

registration\_time timestamp\_ltz(9)

);

Query for file format :

Though the file type is csv ,field delimiter is \t

**create** **or** **replace** file format customer\_tsv\_ff

**type** **=** 'csv'

compression **=** 'none'

field\_delimiter **=** '\t'

field\_optionally\_enclosed\_by **=** '\042'

skip\_header **=** 1 ;

Choose CSV file format :

Graphical user interface

Description automatically generated with medium confidence

Table

Description automatically generated

Graphical user interface, application

Description automatically generated

## Loading a PSV File using WebUI:

Same customer table created with name \_psv :

create or replace transient table customer\_psv (

customer\_pk number(38,0),

salutation varchar(10),

first\_name varchar(20),

last\_name varchar(30),

gender varchar(1),

marital\_status varchar(1),

day\_of\_birth date,

birth\_country varchar(60),

email\_address varchar(50),

city\_name varchar(60),

zip\_code varchar(10),

country\_name varchar(20),

gmt\_timezone\_offset number(10,2),

preferred\_cust\_flag boolean,

registration\_time timestamp\_ltz(9)

);

PSV File format:

**create** **or** **replace** file format customer\_psv\_ff

**type** **=** 'csv'

compression **=** 'none'

field\_delimiter **=** '|'

field\_optionally\_enclosed\_by **=** '\042'

skip\_header **=** 1 ;

Loaded Successfully :

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Loading using JSON File :

Basic Structure of a JSON file :

|  |  |
| --- | --- |
| 1  2  3  4 | - <KEY>:<VALUE> Data is in key/value pairs  - Data is separated by commas  - Objects are enclosed in curly braces ({})  - Arrays are enclosed in square brackets ([]) |

Created one Publicly accessible S3 bucket :

Graphical user interface, text, application, chat or text message

Description automatically generated

Load using Parquet File :

create or replace table cities

(

continent varchar default null,

country varchar default null,

city variant default null

);

Create file format and stage:

create or replace file format parquet\_ff type ='parquet';

Create or replace stage parquet\_stage file\_format = parquet\_ff;

put file://C:\Users\X024271\cities.parquet @parquet\_stage;

A picture containing graphical user interface

Description automatically generated

copy into cities from (select $1:continent::varchar,$1:country:name::varchar,$1:country:city::variant from @parquet\_stage/cities.parquet);

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated with medium confidence

Parquet File Loaded successfully.

Unloading Parquet file :

In snowsql:

copy into @parquet\_stage/out/parquet\_ from

(

select continent,country,c.value::string as city from cities

,lateral flatten(input => city)c

) file\_format =(type ='parquet')

header =true;

Text

Description automatically generated

select t.$1 from @parquet\_stage/out/t;

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated