

# Badge 4 Data Lake

Friday, December 16, 2022 4:58 PM

<https://learn.snowflake.com/courses/course-v1:snowflake+ESS-DLKW+A/courseware/db1d329ab8ac4aa8aafef3d649db66a9/7f31163e95e7487d95957d9827836fc5/?child=first>

- Go to [www.openstreetmap.org](http://www.openstreetmap.org)
- Go to WKT Playground located at: [clydedacruz.github.io/openstreetmap-wkt-playground/](http://clydedacruz.github.io/openstreetmap-wkt-playground/)

From <<https://learn.snowflake.com/courses/course-v1:snowflake+ESS-DLKW+A/courseware/98dc14e86fea4477bcaeda95aa5e766f/7427ce8924a84e8fba06f0972de1688e/>>

Did you see Row 19? It's caused by an extra CRLF at the end of the file. Add a WHERE clause to your SELECT to nix that row!

17	Men's XL
18	Men's XXL
19	

```
select REPLACE($1,chr(13)||chr(10)) as sizes_available
from @uni_klaus_zmd/sweatsuit_sizes.txt
(file_format => zmd_file_format_1)
where sizes_available <> '';
```

## 1 Load file with empty string telling SF how to handle empty string:

create file format "STREAMSETS\_DB"."PRADEEP"."MY\_FILE\_FORMAT" type = CSV

empty\_field\_as\_null=false FIELD\_DELIMITER=';' TRIM\_SPACE = true FIELD\_OPTIONALLY\_ENCLOSED\_BY =  
'\" ESCAPE = 'NONE' ESCAPE\_UNENCLOSED\_FIELD = 'NONE';

From <<https://community.streamsets.com/got-a-question-7/how-to-load-or-unload-empty-field-as-non-nulls-into-snowflake-destination-1161>>

## 2 Load file with empty string telling SF how to handle empty string:

<https://docs.snowflake.com/en/user-guide/data-unload-considerations.html>

### Example: Unloading and Loading Data with Enclosing Quotes%

In the following example, a set of data is unloaded from the null\_empty1 table to the user's stage. The output data file is then used to load data into the null\_empty2 table:

```
-- Source table (null_empty1) contents
+-----+
| i | V | D |
+-----+
| 1 | NULL | NULL value |
| 2 | | Empty string |
+-----+

-- Create a file format that describes the data and the guidelines for processing it
create or replace file format my_csv_format
  field_optionally_enclosed_by='0x27' null_if=('null');

-- Unload table data into a stage
copy into @mystage
  from null_empty1
  file_format = (format_name = 'my_csv_format');

-- Output the data file contents
1,'null','NULL value'
2,'','Empty string'

-- Load data from the staged file into the target table (null_empty2)
copy into null_empty2
  from @mystage/data_0_0_0.csv.gz
  file_format = (format_name = 'my_csv_format');

select * from null_empty2;

+-----+
| i | V | D |
+-----+
| 1 | NULL | NULL value |
| 2 | | Empty string |
+-----+
```

Bac

## Example: Unloading and Loading Data Without Enclosing Quotes

In the following example, a set of data is unloaded from the `null_empty1` table to the user's stage. The output data file is then used to load data into the `null_empty2` table:

```
-- Source table (null_empty1) contents
+-----+
| 1 | V | D |
+-----+
| 1 | NULL | NULL value |
| 2 | | Empty string |
+-----+

-- Create a file format that describes the data and the guidelines for processing it
create or replace file format my_csv_format
  empty_field_as_null=false null_if=('null');

-- Unload table data into a stage
copy into @mystage
  from null_empty1
  file_format = (format_name = 'my_csv_format');

-- Output the data file contents
1,null,null value
2,,empty string

-- Load data from the staged file into the target table (null_empty2)
copy into null_empty2
  from @mystage/data_0_0_0.csv.gz
  file_format = (format_name = 'my_csv_format');

select * from null_empty2;

+-----+
| 1 | V | D |
+-----+
| 1 | NULL | NULL value |
| 2 | | Empty string |
+-----+
```

## NULL handling in Snowflake

### NULL values and NULL handling in Snowflake

October 12, 2022

#### Issue

Write the issue or the “How to” question. Do not repeat the title of the article. Give more information about the scope of this article based on the customer case. You can add prerequisites in this section

Each relational database engine has a slightly different architecture and different ways to handle some of the key concepts, including handling the special NULL value. The Snowflake database uses the following rules:

#### Cause

Write the background information or the root cause of the problem

#### Solution

Write resolution instructions: Use bullets, numbers and additional headings Add Screenshots to explain the resolution Add diagrams to explain complicated technical details, keep the diagrams in lucidchart or in google slide (keep it shared with entire Snowflake), and add the link of the source material in the Internal comment section Go in depth if required Add links and other resources as required Add tables if required

- An equality or inequality comparison like 'a'=NULL, 'a'>NULL or NULL=NULL will always return NULL.
- IS NULL or IS NOT NULL check is the proper way to compare with NULL.
- The EQUAL\_NULL function can be used to check for NULL-safe equality, so equal\_null('a',null) will return false.
- A COUNT(\*) will return a total count of rows in the table, while COUNT(<column\_name>) will return a count of rows with a non-NULL value in that particular column. COUNT(A,B) only counts the rows that have no NULL values in either the A or B column while COUNT(<ALIAS>.\*) can be used to count all the rows containing no NULL columns.
- Aggregate functions (like AVG) also dismisses rows with NULL value; so an AVG from 3 rows containing 1, 5 and NULL values results in 3 as the NULL row is dismissed.
- If you want a real average, you need to do a SUM(<VALUE>)/COUNT(\*), which treats the NULL values as 0.

- An empty string in Snowflake is not equal to NULL, so " IS NULL returns FALSE.

From <<https://community.snowflake.com/s/article/NULL-handling-in-Snowflake>>

File formats come in 6 flavors - CSV, JSON, XML, PARQUET, ORC, & AVRO. Notice that nothing in that list says "PDF" or "Image" or "PNG" or "JPG."

From <<https://learn.snowflake.com/courses/course-v1:snowflake+ESS-DLKW+A/courseware/82cb5bfb82184d68afa5bdfbaad61f31/7c38d67274824db7b7c76664e763f9f3/?child=first>>

Did you notice that Zena was able to define a column using the AS syntax, and then use that column name in the very next line of the same SELECT? This is not true in many other database systems and can be very convenient when developing complex syntax.

```
select UPPER(RELATIVE_PATH) as uppercase_filename
, REPLACE(uppercase_filename, '/') as no_slash_filename
, REPLACE(no_slash_filename, '_', ' ') as no_underscores_filename
, REPLACE(no_underscores_filename, '.PNG') as just_words_filename
from directory(@uni_klaus_clothing);
```

## Nest 4 Functions into 1 Statement

We did the first one for you as an example.

```
-- nest 4 statements into 3 statements
select REPLACE(UPPER(RELATIVE_PATH), '/') as no_slash_filename
, REPLACE(no_slash_filename, '_', ' ') as no_underscores_filename
, REPLACE(no_underscores_filename, '.PNG') as just_words_filename
from directory(@uni_klaus_clothing);
```

	NO_SLASH_FILENAME	NO_UNDERSCORES_FILENAME	JUST_WORDS_FILENAME
1	90S_TRACKSUIT.PNG	90S TRACKSUIT.PNG	90S TRACKSUIT
2	BURGUNDY_SWEATSUIT.PNG	BURGUNDY SWEATSUIT.PNG	BURGUNDY SWEATSUIT

```
select REPLACE(REPLACE(REPLACE(UPPER(RELATIVE_PATH), '/'), '_', ' '), '.PNG') as just_words_filename
from directory(@uni_klaus_clothing);
```

```
select color_or_style, direct_url, price, size as image_size, last_modified as image_last_modified
from directory(@uni_klaus_clothing) as d
join SWEATSUITS as s
on concat('https://uni-klaus.s3.us-west-2.amazonaws.com/clothing', d.relative_path) = s.DIRECT_URL;
```

Even though this workshop is called the Data Lake Workshop (DLKW), we haven't yet discussed what exactly we mean when we say "Data Lake." The Data Lake metaphor was introduced to the world in 2011 by James Dixon, who was the Chief Technology Officer for a company called Pentaho, at that time.

Dixon said:

*If you think of a data mart as a store of bottled water -- cleansed and packaged and structured for easy consumption -- the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples.*

When we talk about Data Lakes at Snowflake, we tend to mean data that has not been loaded into traditional Snowflake tables. We might also call these traditional tables "native" Snowflake tables, or "regular" tables.

As we've already seen, Structured and Semi-structured data that is sitting outside of Snowflake can be easily accessed and analyzed using familiar Snowflake tools like views, file formats, and SQL queries.

We've also seen how Unstructured data, not loaded into Snowflake, can be accessed with a special Snowflake tool called a Directory Table. We've also seen how Directory Tables can be used in combination with functions, joins, internal tables, and standard views to access that non-loaded data.

And through all this, we've seen that bringing together loaded and non-loaded data is a simple and seamless process. When some data is loaded and some is left in a non-loaded state the two types can be joined and queried together, this is sometimes referred to as a **Data Lakehouse**.

## WHAT You Can, WHERE You Can - With Snowflake's Many HOW-You-Cans

Depending on the your role in a data-driven organization:

- You may have the power to move data into Snowflake internal tables, or you may not.
- You may have the power to update data already loaded in internal tables, or you may not.
- You may have the power to copy data from one stage to another, or you may not.

Snowflake makes it possible for you to do WHAT you can, WHERE you can, because

Snowflake keeps adding new HOW-You-Cans to the Snowflake toolset.

So, while a worker in one department might think "I'll just run an update statement on that table,"

another worker trying to achieve the same goal might need to modify a view in their little corner of Snowflake, while another might be able to add a file to a cloud folder and put a stage and view on top of that file, and then join it to data someone else has loaded.

All the different Hands-On Essentials workshops help you learn about various WHATs and HOWs of Snowflake.

This workshop focuses more specifically on one of the WHEREs. In this case, the WHERE is external. The WHERE is external to Snowflake's native tables.

From <<https://learn.snowflake.com/courses/course-v1:snowflake+ESS-DLKW+A/courseware/82cb5bf82184d68afa5bdfbaad61f31/2a7570e170f84cb5a9303187d262c747/?child=first>>

<https://geojson.io/#map=6/39.765/-104.973>

[https://wiki.openstreetmap.org/wiki/Main\\_Page](https://wiki.openstreetmap.org/wiki/Main_Page)  
OSM

```

580 create or replace view COMPETITION as
581 select *
582 from SONRA_DENVER_CO_USA_FREE.DENVER.V_OSM_DEN_AMENITY_SUSTENANCE
583 where
584   ((amenity in ('fast_food','cafe','restaurant','juice_bar'))
585    and
586    (name ilike '%jamba%' or name ilike '%juice%'
587     or name ilike '%superfruit%'))
588   or
589   (cuisine like '%smoothie%' or cuisine like '%juice%');
590
591 select * from competition;
592 desc view competition;
593

```

	name	type	kind	null?	...	default	prim	Aa type
1	ID	NUMBER(38,0)	COLUMN	Y		null	N	GEOGRAPHY
2	COORDINATES	GEOGRAPHY	COLUMN	Y		null	N	
3	AMENITY	VARCHAR(16777216)	COLUMN	Y		null	N	
4	NAME	VARCHAR(16777216)	COLUMN	Y		null	N	
5	ADDR_CITY	VARCHAR(16777216)	COLUMN	Y		null	N	

- Iceberg is an open-source table type, which means a private company does not own the technology. Iceberg Table technology is not proprietary.
- Iceberg Tables are a layer of functionality you can lay on top of parquet files (just like the Cherry Creek Trails file we've been using) that will make files behave more like loaded data. In this way, it's like a file format, but also MUCH more.
- Iceberg Table data will be editable via Snowflake! Read that again. Not just the tables are editable (like the table name), but the data they make available (like the data values in columns and rows). So, you will be able to create an Iceberg Table in Snowflake, on top of a set of parquet files that have NOT BEEN LOADED into Snowflake, and then run INSERT and UPDATE statements on the data using SQL 🤖.

Iceberg Tables will make Snowflake's Data Lake options incredibly powerful!!



## THIS CHANGES EVERYTHING

People sometimes think of Snowflake as a solution for structured, normalized data (which they often call a Data Warehouse). For a while, people said Data Lakes were the only path forward. Lately, many people say the best solution is a Data Lakehouse (they're just mushing the two terms together and saying you need both).

Snowflake can be all of those things and Iceberg tables will be an amazing addition.

☒ You can create a Materialized View on top of a Snowflake External Stage as long as you include an External Table layer in-between.

☐ Snowflake named its proprietary Iceberg tables based on icebergs because the founders love to Snow Ski 🎿!

☒ Iceberg tables are not a proprietary technology, they are open source.

☐ Iceberg tables in Snowflake were rolled out for customers in June of 2022.

☒ Iceberg tables in Snowflake were announced in June of 2022 as "Coming Soon!"

☐ External tables in Snowflake can be used to edit Parquet files using INSERT and UPDATE statements.

☒ Iceberg tables in Snowflake will make it possible to edit Parquet files using INSERT and UPDATE statements.

L4: When we talk about Data Lakes in Snowflake, what do we mean?

☐ Data that flows easily and is always fresh because it doesn't stagnate.

☐ Data that is loaded into internal Snowflake tables using a Snowflake object called a Stream.

☒ Data that is left outside of Snowflake but can be accessed using Snowflake tools.

☐ Data that cannot be accessed using a tool called a Stream.

L6: What GeoSpatial Data Formats have we used in this workshop so far?



Well Known Text (WKT)



Keyhole Markup Language (KML)



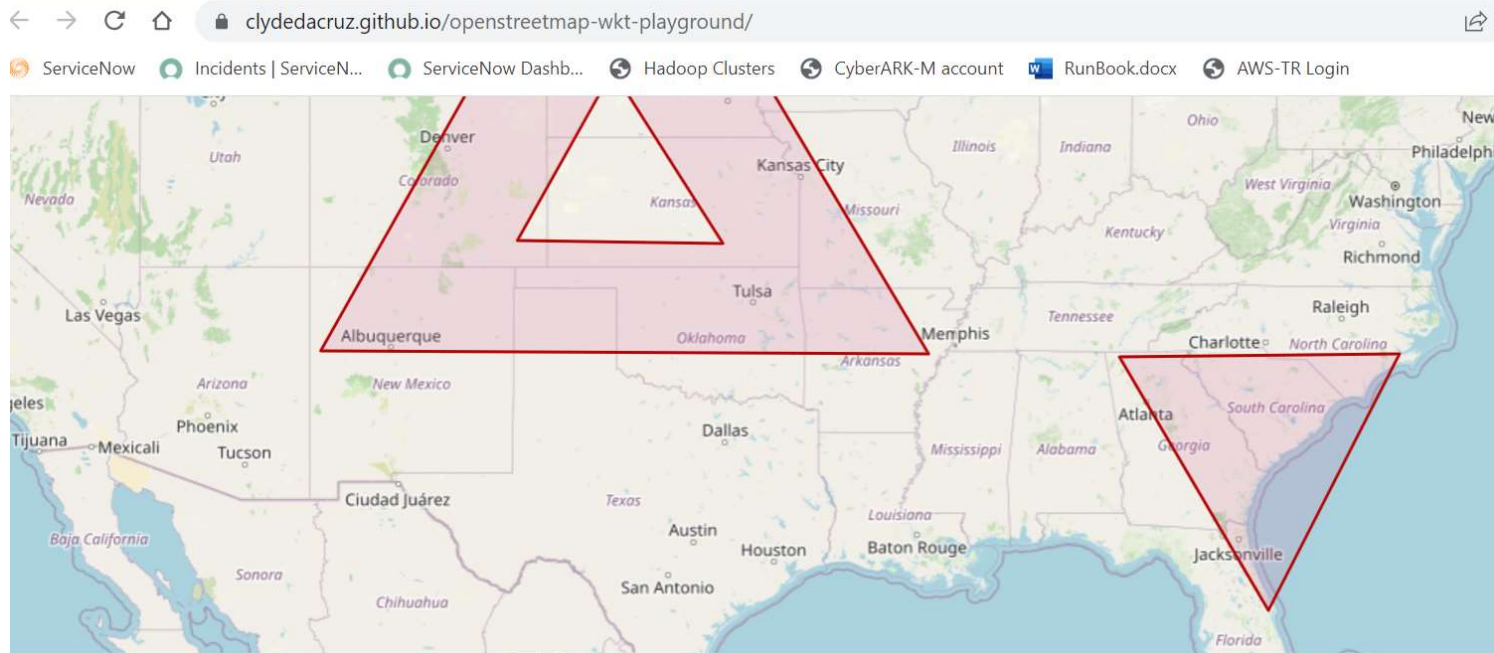
GeoJSON







Well Known Binary (WKB)






WKT: Well-known text (WKT)

<https://clydedacruz.github.io/openstreetmap-wkt-playground/>



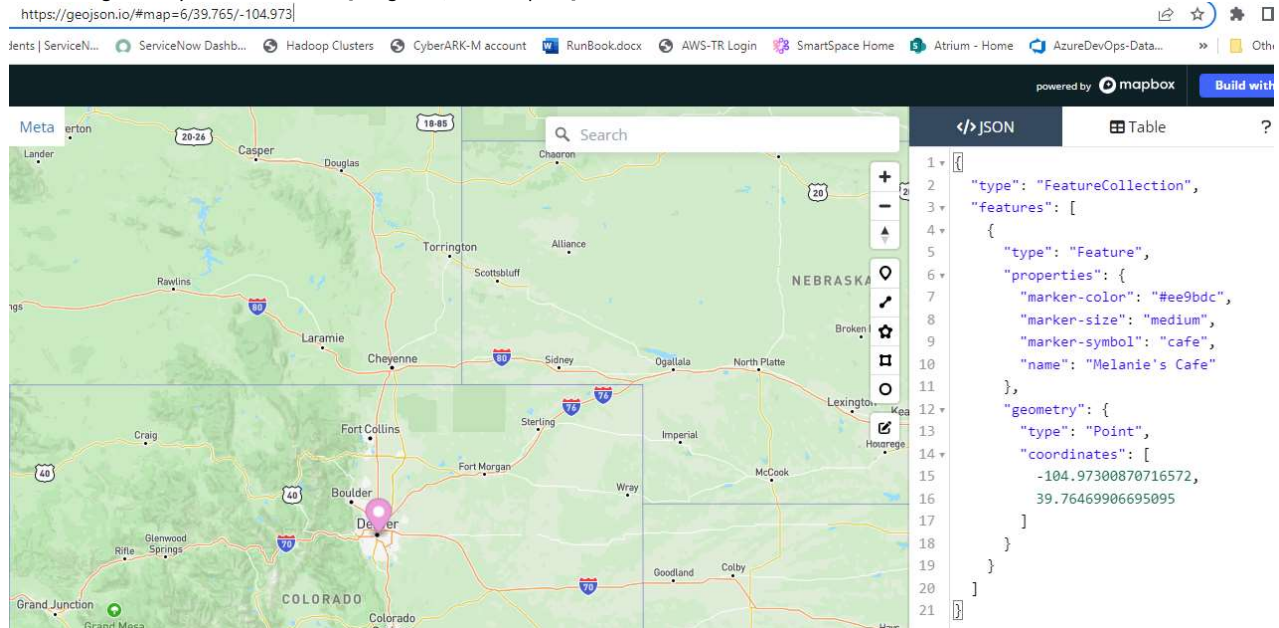
```
MULTIPOLYGON((( -108.72070312499997 34.99400375757577, -100.01953124999997 46.58906908309183, -90.79101562499996 34.92197103616377, -108.72070312499997 34.99400375757577), (-100.10742187499997 41.47566020027821, -102.91992187499996 37.61423141542416, -96.85546874999996 37.54457732085582, -100.10742187499997 41.47566020027821)), ((-85.16601562499999 34.84987503195417, -80.771484375 28.497660832963476, -76.904296875 34.92197103616377, -85.16601562499999 34.84987503195417)))
```

Geometry primitives (2D)		
Type	Examples	
Point		POINT (30 10)
LineString		LINESTRING (30 10, 10 30, 40 40)
Polygon		POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))
		POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))

Multipart geometries (2D)		
Type	Examples	
MultiPoint		MULTIPOINT ((10 40), (40 30), (20 20), (30 10)) MULTIPOINT (10 40, 40 30, 20 20, 30 10)
MultiLineString		MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))
MultiPolygon		MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))
		MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))
GeometryCollection		GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))

GeoJSON: geometry -> coordinates -> [ longitude, latitude pairs]

<https://geojson.io/#map=6/39.765/-104.973>



```

1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {
7         "marker-color": "#ee9bdc",
8         "marker-size": "medium",
9         "marker-symbol": "cafe",
10        "name": "Melanie's Cafe"
11      },
12      "geometry": {
13        "type": "Point",
14        "coordinates": [
15          -104.97300870716572,
16          39.76469906695095
17        ]
18      }
19    ]
20  }
21 }

```

Badge 4 worksheet:

use role accountadmin;

```

select demo_db.public.grader(step, (actual = expected), actual, expected, description) as graded_results
from
(SELECT
'DORA_IS_WORKING' as step
,(select 123 ) as actual
,123 as expected
,'Dora is working!' as description
);

```

select current\_account();

use role sysadmin;

create database ZENAS\_ATHLEISURE\_DB;



```

create schema zenas_athleisure_db.PRODUCTS ;
drop schema ZENAS_ATHLEISURE_DB.public;
create or replace stage ZENAS_ATHLEISURE_DB.products.uni_klaus_clothing
  url = 's3://uni-klaus/clothing';
list @ZENAS_ATHLEISURE_DB.products.uni_klaus_clothing;
show stages;

create or replace stage ZENAS_ATHLEISURE_DB.products.UNI_KLAUS_ZMD
  url = 's3://uni-klaus/zenas_metadata';
list @ZENAS_ATHLEISURE_DB.products.UNI_KLAUS_ZMD;

create or replace stage ZENAS_ATHLEISURE_DB.products.UNI_KLAUS_SNEAKERS
  url = 's3://uni-klaus/sneakers';
list @ZENAS_ATHLEISURE_DB.products.UNI_KLAUS_SNEAKERS;

select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
  SELECT
    'DLKW01' as step
    ,(
      select count(*)
      from ZENAS_ATHLEISURE_DB.INFORMATION_SCHEMA.STAGES
      where stage_url ilike ('%/clothing%')
      or stage_url ilike ('%/zenas_metadata%')
      or stage_url ilike ('%/sneakers%')
    ) as actual
    , 3 as expected
    , 'Stages for Klaus bucket look good' as description
  );

select $1
from @uni_klaus_zmd;
select $1
from @uni_klaus_zmd/product_coordination_suggestions.txt;

create file format zmd_file_format_1
RECORD_DELIMITER = '^';
select $1
from @uni_klaus_zmd/product_coordination_suggestions.txt
(file_format => zmd_file_format_1);

create file format zmd_file_format_2
FIELD_DELIMITER = '^';
select $1, $2,$3,$4,$5, $6,$7
from @uni_klaus_zmd/product_coordination_suggestions.txt
(file_format => zmd_file_format_2);

create or replace file format zmd_file_format_3
FIELD_DELIMITER = '='
RECORD_DELIMITER = '^';
select $1, $2
from @uni_klaus_zmd/product_coordination_suggestions.txt
(file_format => zmd_file_format_3);

create or replace file format zmd_file_format_1
RECORD_DELIMITER = ',';
trim_space = true;
select replace($1, char(13)||char(10)) as sizes_available
from @uni_klaus_zmd/sweatsuit_sizes.txt
(file_format => zmd_file_format_1 )
where sizes_available <> '';

create or replace file format zmd_file_format_2
FIELD_DELIMITER = '|'

```

```

RECORD_DELIMITER = ',';
trim_space = true;
select replace($1, char(13) || char(10)), $2, $3
from @uni_klaus_zmd/swt_product_line.txt
(file_format => zmd_file_format_2);

```

-- Many data files use CRLF (Carriage Return Line Feed) as the record delimiter, so if a different record delimiter is used, the CRLF can end up displayed or loaded! When strange characters appear in your data, you can refine your select statement to deal with them.

-- In SQL we can use ASCII references to deal with these characters.

-- 13 is the ASCII for Carriage return  
 -- 10 is the ASCII for Line Feed  
 -- SQL has a function, CHR() that will allow you to reference ASCII characters by their numbers. So, chr(13) is the same as the Carriage Return character and chr(10) is the same as the Line Feed character.

-- In Snowflake, we can CONCATENATE two values by putting || between them (a double pipe). So we can look for CRLF by telling Snowflake to look for:

```
-- chr(13) || chr(10)
```

```

create or replace file format zmd_file_format_1
RECORD_DELIMITER = ',';
trim_space = true;

```

```

create view zenas_athleisure_db.products.sweatsuit_sizes as
select replace($1, char(13) || char(10)) as sizes_available
from @uni_klaus_zmd/sweatsuit_sizes.txt
(file_format => zmd_file_format_1)
where sizes_available <> '';
select * from zenas_athleisure_db.products.sweatsuit_sizes;

```

```

-----
create or replace file format zmd_file_format_2
FIELD_DELIMITER = '|'
RECORD_DELIMITER = ',';
trim_space = true;

create or replace view zenas_athleisure_db.products.SWEATBAND_PRODUCT_LINE as
select replace($1, char(13) || char(10)) as product_code, $2 as headband_description, $3 as
wristband_description
from @uni_klaus_zmd/swt_product_line.txt
(file_format => zmd_file_format_2);
select * from zenas_athleisure_db.products.SWEATBAND_PRODUCT_LINE;

```

```

-----
create or replace file format zmd_file_format_3
FIELD_DELIMITER = '='
RECORD_DELIMITER = '^';

create or replace view zenas_athleisure_db.products.SWEATBAND_COORDINATION as
select $1 as product_line, $2 as has_matching_sweatsuit
from @uni_klaus_zmd/product_coordination_suggestions.txt
(file_format => zmd_file_format_3);

```

```
select * from zenas_athleisure_db.products.SWEATBAND_COORDINATION;
```

```

select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
  SELECT
    'DLKW02' as step
    ,(select sum(tally) from
      (select count(*) as tally
       from ZENAS_ATHLEISURE_DB.PRODUCTS.SWEATBAND_PRODUCT_LINE
       where length(product_code) > 7
      union

```

```

select count(*) as tally
from ZENAS_ATHLEISURE_DB.PRODUCTS.SWEATSUIT_SIZES
where LEFT(sizes_available,2) = char(13) || char(10))
) as actual
,0 as expected
,'Leave data where it lands.' as description
);

list @uni_klaus_zmd;
list @uni_klaus_sneakers;
list @uni_klaus_clothing;

select $1 from @uni_klaus_clothing/90s_tracksuit.png;

select metadata$filename, count(metadata$file_row_number)
from @uni_klaus_clothing
group by metadata$filename ;

--Directory Tables
select * from directory(@uni_klaus_clothing);

-- Oh Yeah! We have to turn them on, first
alter stage uni_klaus_clothing
set directory = (enable = true);

--Now?
select * from directory(@uni_klaus_clothing);

--Oh Yeah! Then we have to refresh the directory table!
alter stage uni_klaus_clothing refresh;

--Now?
select * from directory(@uni_klaus_clothing);

--testing UPPER and REPLACE functions on directory table
select UPPER(RELATIVE_PATH) as uppercase_filename
, REPLACE(uppercase_filename, '/') as no_slash_filename
, REPLACE(no_slash_filename, '_') as no_underscores_filename
, REPLACE(no_underscores_filename, '.PNG') as just_words_filename
from directory(@uni_klaus_clothing);

select REPLACE(REPLACE(REPLACE(UPPER(RELATIVE_PATH), '/'), '_'), '.PNG') as just_words_filename
from directory(@uni_klaus_clothing);

-----join directory table and other table?
--create an internal table for some sweat suit info
create or replace TABLE ZENAS_ATHLEISURE_DB.PRODUCTS.SWEATSUITS (
    COLOR_OR_STYLE VARCHAR(25),
    DIRECT_URL VARCHAR(200),
    PRICE NUMBER(5,2)
);

--fill the new table with some data
insert into ZENAS_ATHLEISURE_DB.PRODUCTS.SWEATSUITS
(COLOR_OR_STYLE, DIRECT_URL, PRICE)
values
('90s', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/90s\_tracksuit.png',500)
,('Burgundy', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/burgundy\_sweatsuit.png',65)
,('Charcoal Grey', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/charcoal\_grey\_sweatsuit.png',65)
,('Forest Green', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/forest\_green\_sweatsuit.png',65)
,('Navy Blue', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/navy\_blue\_sweatsuit.png',65)
,('Orange', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/orange\_sweatsuit.png',65)
,('Pink', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/pink\_sweatsuit.png',65)
,('Purple', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/purple\_sweatsuit.png',65)
,('Red', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/red\_sweatsuit.png',65)
,('Royal Blue', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/royal\_blue\_sweatsuit.png',65)
,('Yellow', 'https://uni-klaus.s3.us-west-2.amazonaws.com/clothing/yellow\_sweatsuit.png',65);

```

```

select color_or_style, direct_url, price, size as image_size, last_modified as image_last_modified
from directory(@uni_klaus_clothing) as d
join SWEATSUITS as s
on concat('https://uni-klaus.s3.us-west-2.amazonaws.com/clothing', d.relative_path) = s.DIRECT_URL;

```

```

---- cross join or cartesian join
-- 3 way join - internal table, directory table, and view based on external data
create or replace view catalog as
select color_or_style
, direct_url
, price
, size as image_size
, last_modified as image_last_modified
, sizes_available
from sweatsuits
join directory(@uni_klaus_clothing)
on relative_path = SUBSTR(direct_url,54,50)
cross join sweatsuit_sizes;

```

```

select * from catalog;

```

```

----
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
  SELECT
    'DLKW03' as step
    ,(select count(*) from ZENAS_ATHLEISURE_DB.PRODUCTS.CATALOG) as actual
    ,198 as expected
    ,'Cross-joined view exists' as description
);

```

```

-- Add a table to map the sweat suits to the sweat band sets
create table ZENAS_ATHLEISURE_DB.PRODUCTS.UPSELL_MAPPING
(
  SWEATSUIT_COLOR_OR_STYLE varchar(25)
  ,UPSELL_PRODUCT_CODE varchar(10)
);

```

```

--populate the upsell table
insert into ZENAS_ATHLEISURE_DB.PRODUCTS.UPSELL_MAPPING
(
  SWEATSUIT_COLOR_OR_STYLE
  ,UPSELL_PRODUCT_CODE
)
VALUES
('Charcoal Grey','SWT_GRY')
,('Forest Green','SWT_FGN')
,('Orange','SWT_ORG')
,('Pink','SWT_PNK')
,('Red','SWT_RED')
,('Yellow','SWT_YLW');

```

```

select * from UPSSELL_MAPPING;

```

```

----
-- Zena needs a single view she can query for her website prototype
create view catalog_for_website as
select color_or_style
,price
,direct_url
,size_list
,coalesce('BONUS: ' || headband_description || ' & ' || wristband_description, 'Consider White, Black
or Grey Sweat Accessories') as upsell_product_desc
from
(
  select color_or_style, price, direct_url, image_last_modified,image_size
  ,listagg(sizes_available, ' | ') within group (order by sizes_available) as size_list
  from catalog
  group by color_or_style, price, direct_url, image_last_modified, image_size
)

```

```

) as c
left join upsell_mapping as u
on u.sweatsuit_color_or_style = c.color_or_style
left join sweatband_coordination as sc
on sc.product_line = u.upsell_product_code
left join sweatband_product_line as spl
on spl.product_code = sc.product_line
where price < 200 -- high priced items like vintage sweatsuits aren't a good fit for this website
and image_size < 1000000 -- large images need to be processed to a smaller size
;

```

```

--select * from sweatband_coordination;

```

```

select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
SELECT
'DLKW04' as step
,(select count(*)
from zenas_athleisure_db.products.catalog_for_website
where upsell_product_desc like '%NUS:%') as actual
,6 as expected
,'Relentlessly resourceful' as description
);

```

```

create database MELS_SMOOTHIE_CHALLENGE_DB;
drop schema MELS_SMOOTHIE_CHALLENGE_DB.public;
create schema MELS_SMOOTHIE_CHALLENGE_DB.trails;

```

```

create or replace stage MELS_SMOOTHIE_CHALLENGE_DB.trails.trails_geojson
url = 's3://uni-lab-files-more/dlkw/trails/trails_geojson';
list @MELS_SMOOTHIE_CHALLENGE_DB.trails.trails_geojson;

```

```

create or replace stage MELS_SMOOTHIE_CHALLENGE_DB.trails.trails_parquet
url = 's3://uni-lab-files-more/dlkw/trails/trails_parquet';
list @MELS_SMOOTHIE_CHALLENGE_DB.trails.trails_parquet;

```

```

create or replace file format MELS_SMOOTHIE_CHALLENGE_DB.trails.ff_json
type = 'json';

```

```

create or replace file format MELS_SMOOTHIE_CHALLENGE_DB.trails.ff_parquet
type = 'parquet';

```

```

select $1 from @trails_geojson
(file_format => ff_json);

```

```

select $1 from @MELS_SMOOTHIE_CHALLENGE_DB.trails.trails_parquet
(file_format => ff_parquet);

```

```

----
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
SELECT
'DLKW05' as step
,(select sum(tally)
from
(select count(*) as tally
from mels_smoothie_challenge_db.information_schema.stages
union all
select count(*) as tally
from mels_smoothie_challenge_db.information_schema.file_formats)) as actual
,4 as expected
,'Camila\'s Trail Data is Ready to Query' as description
);
----

```

```

select $1:sequence_1 as sequence_1,
$1:trail_name::varchar as trail_name,
$1:latitude as latitude,

```



```

    $1:longitude as longitude,
    $1:sequence_2 as sequence_2,
    $1:elevation as elevation
from @MELS_SMOOTHIE_CHALLENGE_DB.trails.trails_parquet
(file_format => ff_parquet)
order by sequence_1;

```

```

create or replace view MELS_SMOOTHIE_CHALLENGE_DB.trails.CHERRY_CREEK_TRAIL as
select $1:sequence_1 as point_id,
    $1:trail_name::varchar as trail_name,
    $1:latitude::number(11,8) as lng,
    $1:longitude::number(11,8) as lat
from @MELS_SMOOTHIE_CHALLENGE_DB.trails.trails_parquet
(file_format => ff_parquet)
order by point_id;

```

```

select top 100 lng || ' ' || lat as coord_pair, 'POINT(' || coord_pair || ')' as trail_point
from MELS_SMOOTHIE_CHALLENGE_DB.trails.CHERRY_CREEK_TRAIL;

```

--To add a column, we have to replace the entire view  
--changes to the original are shown in red

```

create or replace view cherry_creek_trail as
select
    $1:sequence_1 as point_id,
    $1:trail_name::varchar as trail_name,
    $1:latitude::number(11,8) as lng,
    $1:longitude::number(11,8) as lat,
    lng || ' ' || lat as coord_pair
from @trails_parquet
(file_format => ff_parquet)
order by point_id;
select * from cherry_creek_trail;

```

```

select
'LINESTRING(' ||
listagg(coord_pair, ',')
within group (order by point_id)
|| ')' as my_linestring
from cherry_creek_trail
where point_id <= 10
group by trail_name;

```

```

select
'LINESTRING(' ||
listagg(coord_pair, ',')
within group (order by point_id)
|| ')' as my_linestring
from cherry_creek_trail
group by trail_name;

```

```

----
select $1 from @trails_geojson (file_format => ff_json);

```

```

create or replace view DENVER_AREA_TRAILS as
select
    $1:features[0]:properties:Name::string as feature_name
    , $1:features[0]:geometry:coordinates::string as feature_coordinates
    , $1:features[0]:geometry::string as geometry
    , $1:features[0]:properties::string as feature_properties
    , $1:crs:properties:name::string as specs
    , $1 as whole_object
from @trails_geojson (file_format => ff_json);

```

```

select * from DENVER_AREA_TRAILS;

```

```

----
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
SELECT

```

```

'DLKW06' as step
,(select count(*) as tally
  from mels_smoothie_challenge_db.information_schema.views
  where table_name in ('CHERRY_CREEK_TRAIL','DENVER_AREA_TRAILS')) as actual
,2 as expected
,'Mel\'s views on the geospatial data from Camila' as description
);

----

--Remember this code?
select
'LINESTRING(' ||
listagg(coord_pair, ',')
within group (order by point_id)
|| ')' as my_linestring
,st_length(TO_GEOGRAPHY(my_linestring)) as length_of_trail --this line is new! but it won't work!
from cherry_creek_trail
group by trail_name;

```

```

select feature_name ,st_length(to_geography(WHOLE_OBJECT)) as trail_length from
DENVER_AREA_TRAILS;

```

```

select get_ddl('view', 'DENVER_AREA_TRAILS');

```

```

create or replace view DENVER_AREA_TRAILS(
  FEATURE_NAME,
  FEATURE_COORDINATES,
  GEOMETRY,
  TRAIL_LENGTH,
  FEATURE_PROPERTIES,
  SPECS,
  WHOLE_OBJECT
) as
select
$1:features[0]:properties::Name::string as feature_name
,$1:features[0]:geometry::coordinates::string as feature_coordinates
,$1:features[0]:geometry::string as geometry
,st_length(to_geography(geometry)) as trail_length
,$1:features[0]:properties::string as feature_properties
,$1:crs:properties::name::string as specs
,$1 as whole_object
from @trails_geojson (file_format => ff_json);

```

```

select * from DENVER_AREA_TRAILS;

```

```

select $1 from @trails_geojson (file_format => ff_json);

```

```

----

--Create a view that will have similar columns to DENVER_AREA_TRAILS
--Even though this data started out as Parquet, and we're joining it with geoJSON data
--So let's make it look like geoJSON instead.
create view DENVER_AREA_TRAILS_2 as
select
trail_name as feature_name
,'{"coordinates":[' || listagg('[' || lng || ',' || lat || '],',',') || '],"type":"LineString"}' as geometry
,st_length(to_geography(geometry)) as trail_length
from cherry_creek_trail
group by trail_name;

```

```

select * from DENVER_AREA_TRAILS_2;

```

```

--Create a view that will have similar columns to DENVER_AREA_TRAILS
select feature_name, geometry, trail_length
from DENVER_AREA_TRAILS
union all
select feature_name, geometry, trail_length
from DENVER_AREA_TRAILS_2;

```

```

--Add more GeoSpatial Calculations to get more GeoSpecial Information!
create or replace view trails_and_boundaries as

```

```

select feature_name
, to_geography(geometry) as my_linestring
, st_xmin(my_linestring) as min_eastwest
, st_xmax(my_linestring) as max_eastwest
, st_ymin(my_linestring) as min_northsouth
, st_ymax(my_linestring) as max_northsouth
, trail_length
from DENVER_AREA_TRAILS
union all
select feature_name
, to_geography(geometry) as my_linestring
, st_xmin(my_linestring) as min_eastwest
, st_xmax(my_linestring) as max_eastwest
, st_ymin(my_linestring) as min_northsouth
, st_ymax(my_linestring) as max_northsouth
, trail_length
from DENVER_AREA_TRAILS_2;

select * from trails_and_boundaries;

select 'polygon((' ||
min(min_eastwest) || ' ' || max(max_northsouth) || ' ' ||
max(max_eastwest) || ' ' || max(max_northsouth) || ' ' ||
max(max_eastwest) || ' ' || min(min_northsouth) || ' ' ||
min(min_eastwest) || ' ' || min(min_northsouth) || ' '))' as my_polygon
from trails_and_boundaries;

----
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
SELECT
'DLKW07' as step
,(select round(max(max_northsouth))
from MELS_SMOOTHIE_CHALLENGE_DB.TRAILS.TRAILS_AND_BOUNDARIES)
as actual
,40 as expected
,'Trails Northern Extent' as description
);

----
-- Melanie's Location into a 2 Variables (mc for melanies cafe)
set mc_lat='-104.97300245114094';
set mc_lng='39.76471253574085';

--Confluence Park into a Variable (loc for location)
set loc_lat='-105.00840763333615';
set loc_lng='39.754141917497826';

--Test your variables to see if they work with the Makepoint function
select st_makepoint($mc_lat,$mc_lng) as melanies_cafe_point;
select st_makepoint($loc_lat,$loc_lng) as confluent_park_point;

--use the variables to calculate the distance from
--Melanie's Cafe to Confluent Park
select st_distance(
st_makepoint($mc_lat,$mc_lng)
,st_makepoint($loc_lat,$loc_lng)
) as mc_to_cp;

create or replace schema mels_smoothie_challenge_db.LOCATIONS;

create or replace function distance_to_mc(loc_lng number(38,32), loc_lat number(38,32))
returns float
as
$$
st_distance(
st_makepoint('-104.97300245114094','39.76471253574085'),
st_makepoint(loc_lng, loc_lat))
$$;

```

```

set tc_lat='39.74548137398218';
set tc_lng='-105.00532059763648';
select distance_to_mc($tc_lng, $tc_lat);

----
create or replace view COMPETITION as
select *
from SONRA_DENVER_CO_USA_FREE.DENVER.V_OSM_DEN_AMENITY_SUSTENANCE
where
  ((amenity in ('fast_food','cafe','restaurant','juice_bar'))
  and
  (name ilike '%jamba%' or name ilike '%juice%'
  or name ilike '%superfruit%'))
or
  (cuisine like '%smoothie%' or cuisine like '%juice%');

select * from competition;
desc view competition;

SELECT
name
,cuisine
, ST_DISTANCE(
  st_makepoint('-104.97300245114094','39.76471253574085')
  , coordinates
) AS distance_from_melanies
,*
FROM competition
ORDER by distance_from_melanies;

CREATE OR REPLACE FUNCTION distance_to_mc(lng_and_lat GEOGRAPHY)
RETURNS FLOAT
AS
$$
  st_distance(
    st_makepoint('-104.97300245114094','39.76471253574085')
    ,lng_and_lat
  )
$$
;

SELECT
name
,cuisine
,distance_to_mc(coordinates) AS distance_from_melanies
,*
FROM competition
ORDER by distance_from_melanies;

-- Tattered Cover Bookstore McGregor Square
set tcb_lng='-104.9956203';
set tcb_lat='39.754874';

--this will run the first version of the UDF
select distance_to_mc($tcb_lng,$tcb_lat);

--this will run the second version of the UDF, bc it converts the coords
--to a geography object before passing them into the function
select distance_to_mc(st_makepoint($tcb_lng,$tcb_lat));

--this will run the second version bc the Sonra Coordinates column
-- contains geography objects already
select name
, distance_to_mc(coordinates) as distance_to_melanies
, ST_ASWKT(coordinates)
from SONRA_DENVER_CO_USA_FREE.DENVER.V_OSM_DEN_SHOP
where shop='books'
and name like '%Tattered Cover%'
and addr_street like '%Wazee%';

```

```

-- bike shops --
desc view SONRA_DENVER_CO_USA_FREE.DENVER.V_OSM_DEN_SHOP;
select * from SONRA_DENVER_CO_USA_FREE.DENVER.V_OSM_DEN_SHOP limit 5;

create or replace view DENVER_BIKE_SHOPS as
select name, shop, distance_to_mc(coordinates) as DISTANCE_TO_MELANIES
from SONRA_DENVER_CO_USA_FREE.DENVER.V_OSM_DEN_SHOP
where shop = 'bicycle'
order by DISTANCE_TO_MELANIES asc;

select * from DENVER_BIKE_SHOPS;
desc view DENVER_BIKE_SHOPS;

--
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
  SELECT
    'DLKW08' as step
    ,(select truncate(distance_to_melanies)
      from mels_smoothie_challenge_db.locations.denver_bike_shops
      where name like '%Mojo%') as actual
    ,14084 as expected
    ,'Bike Shop View Distance Calc works' as description
);
--

select * from CHERRY_CREEK_TRAIL;
alter view CHERRY_CREEK_TRAIL rename to V_CHERRY_CREEK_TRAIL;

create or replace external table T_CHERRY_CREEK_TRAIL(
  my_filename varchar(50) as (metadata$filename::varchar(50))
)
location= @trails_parquet
auto_refresh = true
file_format = (type = parquet);

select get_ddl('view','mels_smoothie_challenge_db.trails.v_cherry_creek_trail');

create or replace external table mels_smoothie_challenge_db.trails.T_CHERRY_CREEK_TRAIL(
  POINT_ID number as ($1:sequence_1::number),
  TRAIL_NAME varchar(50) as ($1:trail_name::varchar),
  LNG number(11,8) as ($1:latitude::number(11,8)),
  LAT number(11,8) as ($1:longitude::number(11,8)),
  COORD_PAIR varchar(50) as (lng::varchar || ' ' || lat::varchar)
)
location= @mels_smoothie_challenge_db.trails.trails_parquet
auto_refresh = true
file_format = mels_smoothie_challenge_db.trails.ff_parquet;

create or replace secure materialized view SMV_CHERRY_CREEK_TRAIL as
select * from mels_smoothie_challenge_db.trails.T_CHERRY_CREEK_TRAIL;

select * from SMV_CHERRY_CREEK_TRAIL;

select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as
graded_results from
(
  SELECT
    'DLKW09' as step
    ,(select row_count
      from mels_smoothie_challenge_db.information_schema.tables
      where table_schema = 'TRAILS'
      and table_name = 'SMV_CHERRY_CREEK_TRAIL')
      as actual
    ,3526 as expected
    ,'Secure Materialized View Created' as description
);

```



