Lab 4: Analyze PDF Invoices using Java UDF and Snowsight

Overview

This lab is designed to help you understand the capabilities included in Snowflake's support for unstructured data and Snowpark. Although this guide is specific to processing PDF files, you can apply this pattern of processing natively in Snowflake to many types of unstructured data. All source code for this guide can be found on Github.

What You'll Need

- Snowflake account
- SnowSQL installed

What You'll Learn

- How to access PDF invoices in cloud storage from Snowflake
- How to extract text from PDFs natively using a Java User-Defined Function (UDF)

What You'll Build

- An external stage to access files in S3 from Snowflake
- A user-defined function using Snowflake's engine to process files

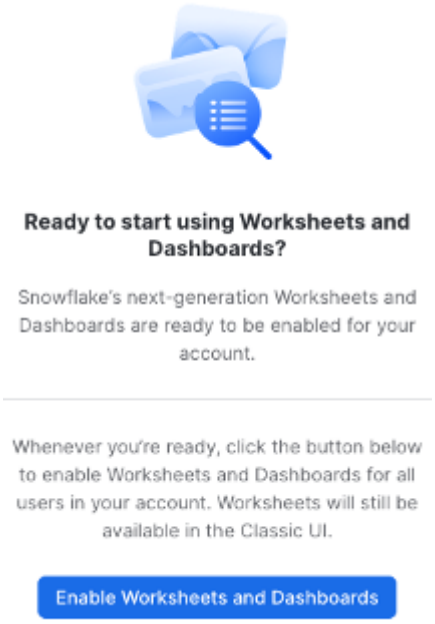
Prepare Your Environment

If you haven't already, register for a Snowflake free 30-day trial. The Snowflake edition (Standard, Enterprise, Business Critical, e.g.), cloud provider (AWS, Azure, e.g.), and Region (US East, EU, e.g.) do not matter for this lab. We suggest you select the region which is physically closest to you and the Enterprise Edition, our most popular offering. After registering, you will receive an email with an activation link and your Snowflake account URL.

Navigating to Snowsight

For this lab, you will use the latest Snowflake web interface, Snowsight.

1. Log into your Snowflake trial account
2. Click on **Snowsight** Worksheets tab. The new web interface opens in a separate tab or window.
3. Click **Worksheets** in the left-hand navigation bar. The **Ready to Start Using Worksheets and Dashboards** dialog opens.
4. Click the **Enable Worksheets and Dashboards button**.



Access the Data

Let's start by loading the PDF invoices into Snowflake. Snowflake supports two types of stages for storing data files used for loading and unloading:

- [Internal] stages store the files internally within Snowflake.
- [External] stages store the files in an external location (i.e. S3 bucket) that is referenced by the stage. An external stage specifies location and credential information, if required, for the bucket.

For this lab, we will use an external stage, but processing and analysis workflows demonstrated in this lab can also be done using an internal stage.

Create a Database, Warehouse, and Stage

Let's create a database, warehouse, and stage that will be used for loading and processing the PDFs. We will use the UI within the Worksheets tab to run the DDL that creates the database and schema. Copy the commands below into your trial environment, and execute each individually.

```
use role sysadmin;

create or replace database pdf;
create or replace warehouse quickstart;

use database pdf;
use schema public;
use warehouse quickstart;

create or replace stage pdf_external
url="s3://sfquickstarts/Analyze PDF Invoices/Invoices/"
directory = (enable = TRUE);
```

Verify if the PDF files are accessible in your external stage by entering the following command on your Snowflake worksheet.

```
ls @pdf_external;
```

You should now see an identical list of files from the S3 bucket. Make sure you see 300 files.

```
14
15  ls @pdf_external;
```

Objects | Query | Results | Chart

	name	... size
1	s3://sfquickstarts/Analyze PDF Invoices/Invoices/invoice1.pdf	13,636
2	s3://sfquickstarts/Analyze PDF Invoices/Invoices/invoice10.pdf	13,639
3	s3://sfquickstarts/Analyze PDF Invoices/Invoices/invoice100.pdf	13,653
4	s3://sfquickstarts/Analyze PDF Invoices/Invoices/invoice101.pdf	13,171
5	s3://sfquickstarts/Analyze PDF Invoices/Invoices/invoice102.pdf	12,962
6	s3://sfquickstarts/Analyze PDF Invoices/Invoices/invoice103.pdf	13,269
7	s3://sfquickstarts/Analyze PDF Invoices/Invoices/invoice104.pdf	12,683

Extract Text from PDFs

In this section, we want to extract attributes from the PDF invoices. The entities extracted are going to be fields like product names, unit cost, total cost, as well as business name. The goal is to have these fields to enrich the file-level metadata for analytics.

Creating a Java UDF in Snowflake

The Java code to parse PDFs requires some dependencies. Instead of downloading those jar files and uploading to an internal stage, you can create an external stage and reference them when creating a UDF inline.

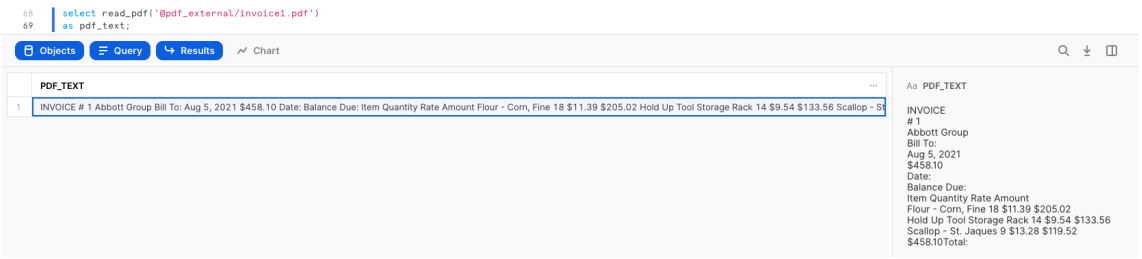
```
-- Create stage to store the JAR file
create or replace stage jars_stage_internal;

-- Create external stage to import PDFBox from S3
create or replace stage jars_stage
 url = "s3://sfquickstarts/Common JARs/"
 directory = (enable = true auto_refresh = false);

-- Create a java function to parse PDF files
create or replace function read_pdf(file string)
returns String
language java
imports = ('@jars_stage/pdfbox-app-2.0.24.jar')
```

```
HANDLER = 'PdfParser.ReadFile'
as
$$
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;
import org.apache.pdfbox.text.PDFTextStripperByArea;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import com.snowflake.snowpark_java.types.SnowflakeFile;

public class PdfParser {

    public static String ReadFile(String file_url) throws IOException {
SnowflakeFile file = SnowflakeFile.newInstance(file_url);
        try (PDDocument document = PDDocument.load(file.getInputStream())) {

            document.getClass();

            if (!document.isEncrypted()) {

                PDFTextStripperByArea stripper = new PDFTextStripperByArea();
                stripper.setSortByPosition(true);

                PDFTextStripper tStripper = new PDFTextStripper();

                String pdfFileInText = tStripper.getText(document);
                return pdfFileInText;
            }
        }

        return null;
    }
}
$$;
```

Invoking the Java UDF

The UDF can be invoked on any PDF file with a simple SQL statement. First, make sure to refresh the directory table metadata for your external stage.

```
alter stage pdf_external refresh;

select read_pdf('@pdf_external/invoice1.pdf')
as pdf_text;
```



The output is text values extracted from `invoice1.pdf`.

```
INVOICE
# 1
Abbott Group
Bill To:
Aug 5, 2021
$458.10
Date:
Balance Due:
Item Quantity Rate Amount
Flour - Corn, Fine 18 $11.39 $205.02
Hold Up Tool Storage Rack 14 $9.54 $133.56
Scallop - St. Jaques 9 $13.28 $119.52
$458.10Total:
```

UDFs are account-level objects. So if a developer familiar with Java creates a UDF, an analyst in the same account with proper permissions can invoke the UDF in their queries.

Extracting and Storing Fields

We want to store the extracted text as additional attributes for analysts to be able to select and retrieve the files of interest in their analysis, as well as perform some analytics on the attributes found.

We first need to create a table with the extracted text in its raw form. From this table, we can create views to parse the text into various fields for easier analysis.

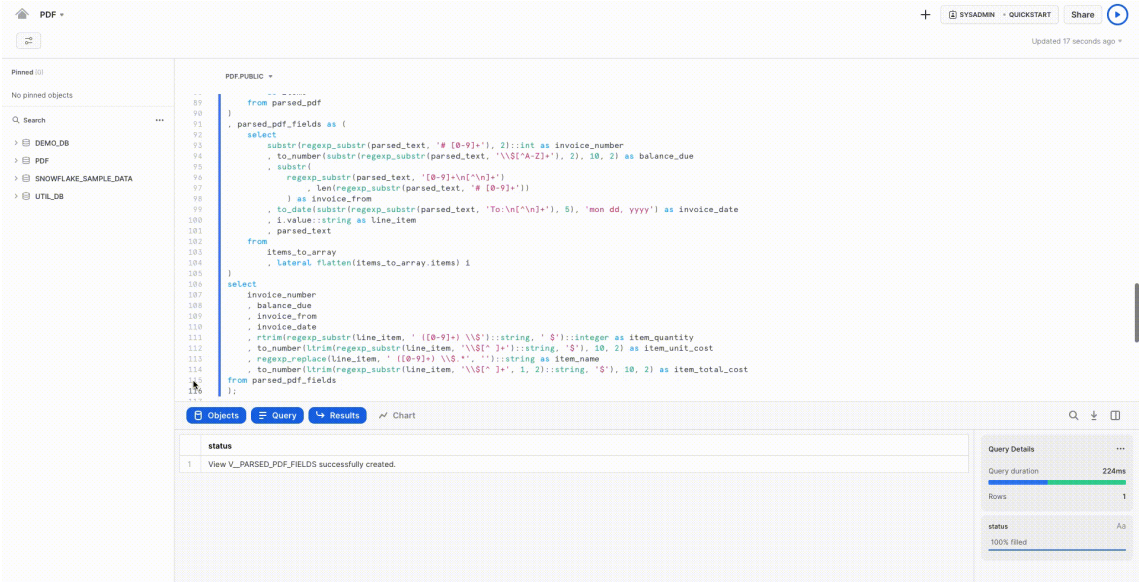
```
create or replace table parsed_pdf as
select
    relative_path
    , file_url
    , read_pdf('@pdf_external/' || relative_path) as parsed_text
from directory(@pdf_external);
```

Using Snowflake's string functions, we can parse out specific values as fields like balance due, item name, item quantity, and more.

```
create or replace view v__parsed_pdf_fields as (
with items_to_array as (
    select
        *
        , split(
            substr(
              regexp_substr(parsed_text, 'Amount\n(.*)\n(.*)\n(.*)'
```

```
              ), 8
            ), '\n'
          )
        as items
    from parsed_pdf
)
, parsed_pdf_fields as (
    select
        substr(regexp_substr(parsed_text, '# [0-9]+'), 2)::int as invoice_number
        , to_number(substr(regexp_substr(parsed_text, '\\$[^A-Z]+'), 2), 10, 2) as
balance_due
        , substr(
            regexp_substr(parsed_text, '[0-9]+\n[^\n]+')
                , len(regexp_substr(parsed_text, '# [0-9]+'))
            ) as invoice_from
        , to_date(substr(regexp_substr(parsed_text, 'To:\n[^\n]+'), 5), 'mon dd,
yyyy') as invoice_date
        , i.value::string as line_item
        , parsed_text
    from
        items_to_array
        , lateral flatten(items_to_array.items) i
)
select
    invoice_number
    , balance_due
    , invoice_from
    , invoice_date
    , rtrim(regexp_substr(line_item, ' ([0-9]+) \\$')::string, ' $')::integer as
item_quantity
    , to_number(ltrim(regexp_substr(line_item, '\\$[^ ]+')::string, '$'), 10, 2) as
item_unit_cost
    , regexp_replace(line_item, ' ([0-9]+) \\$.*', '')::string as item_name
    , to_number(ltrim(regexp_substr(line_item, '\\$[^ ]+', 1, 2)::string, '$'), 10, 2)
as item_total_cost
from parsed_pdf_fields
);
```

If you collapse and expand the `PDF` database in the Objects pane on the left, you should now see a view name `V__PARSED_PDF_FIELDS`. Click on that view, and below you should see a preview of the fields you have created along with icons to indicate the data type. You can also see a preview of the view by clicking on the button that looks like a magnifier glass.



Exploring Invoice Data

Now let's explore the data from the PDF invoices. What are the most purchased items based on quantity?

```
select
    sum(item_quantity)
    , item_name
from v__parsed_pdf_fields
group by item_name
order by sum(item_quantity) desc
limit 10;
```

```
117
118  --what are the 10 most purchased items?
119  select
120      sum(item_quantity)
121      , item_name
122  from v__parsed_pdf_fields
123  group by item_name
124  order by sum(item_quantity) desc
125  limit 10;
```

Objects | Query | Results | Chart

	SUM(ITEM_QUANTITY)	ITEM_NAME
1	203	Apron
2	178	Lettuce - California Mix
3	169	Bar Special K
4	159	Oil - Margarine
5	158	Flower - Commercial Bronze
6	155	Apple - Fuji
7	154	Wine - Rhine Riesling Wolf Blass
8	152	Garam Marsala
9	151	Phyllo Dough
10	146	Salmon - Smoked, Sliced

What are the items on which the most money was spent?

```
select
    sum(item_total_cost)
    , item_name
```

```
from v__parsed_pdf_fields
group by item_name
order by sum(item_total_cost) desc
limit 10;
```

```
126
127   --on which 10 items was the most money spent?
128   select
129       sum(item_total_cost)
130       , item_name
131   from v__parsed_pdf_fields
132   group by item_name
133   order by sum(item_total_cost) desc
134   limit 10;
```

Objects | Query | Results | Chart

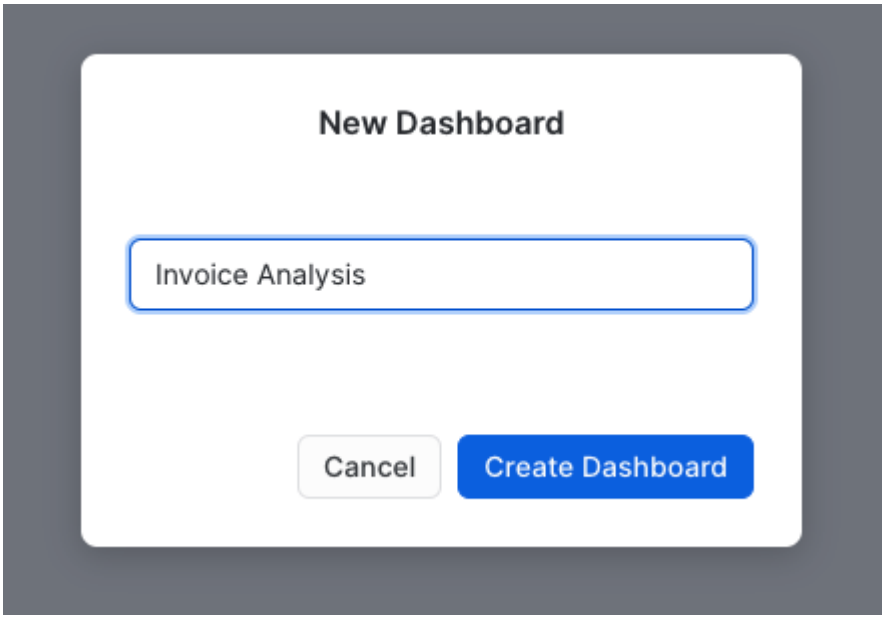
	SUM(ITEM_TOTAL_COST)	ITEM_NAME
1	3,214.68	Lettuce - California Mix
2	3,079.83	Oil - Margarine
3	2,644.85	Bar Special K
4	2,518.65	Pepper - Chili Powder
5	2,313.32	Phyllo Dough
6	2,173.98	Soup - Boston Clam Chowder
7	2,057.94	Sauce - Oyster
8	1,905.64	Cinnamon Rolls
9	1,865.92	Duck - Legs
10	1,825	Salmon - Smoked, Sliced

Analyze the Data with Snowsight

Now we can use Snowsight to visualize the data extracted from the PDF invoices.

Create a Dashboard

Let's use a dashboard as a collection of all of the visualizations we will create. Click on the Home button, then click on **Dashboards** in the pane on the left. Create a new dashboard by clicking the **+ Dashboard** button in the top-right. Name the dashboard `Invoice Analysis`, and click **Create Dashboard**.

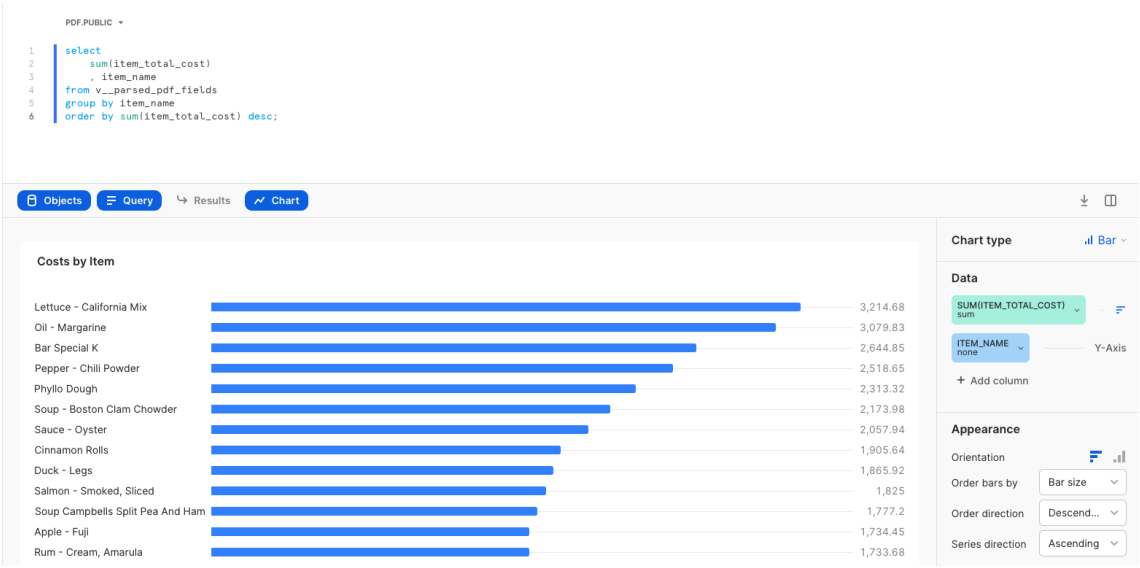


Now let's create the first tile on the Invoice Analysis dashboard by clicking the button **+ New Tile**. Give the tile a name by clicking on the timestamp at the top, and name it `Costs by Item`. In the canvas, copy/paste the SQL

below to query the data needed for the chart and run it.

```
select
    sum(item_total_cost)
    , item_name
from v__parsed_pdf_fields
group by item_name
order by sum(item_total_cost) desc;
```

Now click on **Chart**. Change the chart type from line to bar, X-Axis to `ITEM_NAME`, and orientation to horizontal. In this chart, you should see "Lettuce - California Mix" as the item on which the most money was spent, $3,214.68.



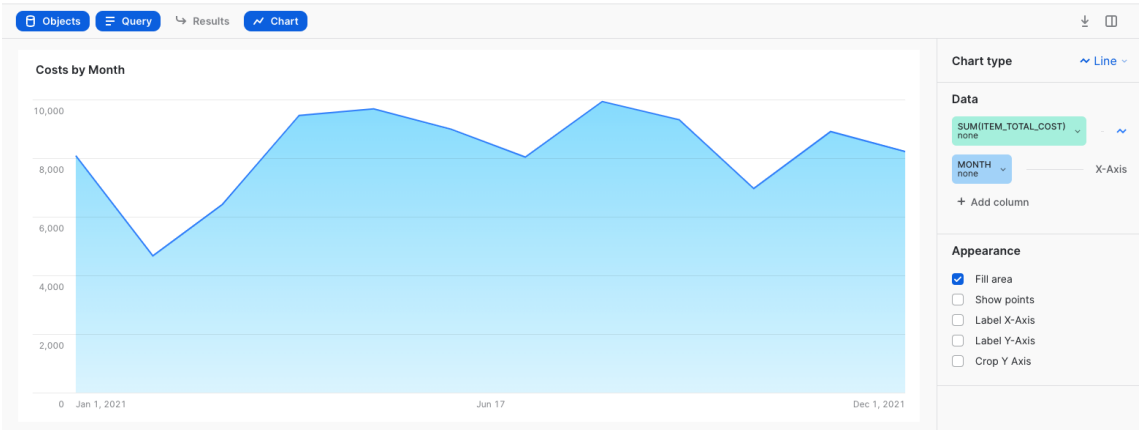
Now let's create another tile by clicking **Return to Invoice Analysis** in the top-left, then click on the **+** button, then **New Tile from Worksheet**. Name the tile `Costs by Month`. Now copy/paste and run this query.

```
select
    sum(item_total_cost)
    , date_trunc('month', invoice_date) as month
from v__parsed_pdf_fields
group by date_trunc('month', invoice_date);
```

Again, click **Chart**. You should see a line chart with `MONTH` on the x-axis and `SUM(ITEM_TOTAL_COST)` on the y-axis. Then click **Return to Invoice Analysis**. You should now see two tiles on your dashboard, which you can rearrange by clicking and dragging.

PDF.PUBLIC

```
select
    sum(item_total_cost)
    , date_trunc('month', invoice_date) as month
from v__parsed_pdf_fields
group by date_trunc('month', invoice_date);
```



Conclusion

Congratulations! You used Snowflake to analyze PDF invoices.

What we've covered

- Accessing unstructured data with an **external stage**
- Processing unstructured data with a **Java UDF**
- Visualize data with **Snowsight**