**CORTEX SEARCH VIA STORED PROCEDURE → REST API**

**(WITH PROCEDURE)**

# **1.1 Use the new credentials only use cortex\_search**

<https://app.snowflake.com/tufbwjl/qeb33118/#/homepage>

username : CortexSearchToken

password : CortexSearchToken@2024

# **1.2 Creating and Assigning the Role**

* use role accountadmin;
* -- Grant CREATE DATABASE privilege

GRANT CREATE DATABASE ON ACCOUNT TO ROLE CORTEX\_SEARCH\_TOKEN;

* -- Grant CREATE WAREHOUSE privilege

GRANT CREATE WAREHOUSE ON ACCOUNT TO ROLE CORTEX\_SEARCH\_TOKEN;

* use role CORTEX\_SEARCH\_TOKEN
* -- create demo database

CREATE OR REPLACE DATABASE CORTEX\_SEARCH\_DB;

* -- create schema
* CREATE OR REPLACE SCHEMA CORTEX\_SEARCH\_SCHEMA;
* CREATE OR REPLACE WAREHOUSE CORTEX\_SEARCH\_WH WITH

WAREHOUSE\_SIZE='X-SMALL'

AUTO\_SUSPEND = 120

AUTO\_RESUME = TRUE

INITIALLY\_SUSPENDED=TRUE;

* USE WAREHOUSE CORTEX\_SEARCH\_WH;

# **1.3 Creating a Stage**

CREATE OR REPLACE STAGE CORTEX\_STAGE

DIRECTORY = (ENABLE = TRUE)

ENCRYPTION = (TYPE = 'SNOWFLAKE\_SSE');

list @CORTEX\_STAGE;

# **1.4 Create function**

CREATE OR REPLACE FUNCTION pdf\_text\_chunker(file\_url STRING)

RETURNS TABLE (chunk VARCHAR)

LANGUAGE PYTHON

RUNTIME\_VERSION = '3.9'

HANDLER = 'pdf\_text\_chunker'

PACKAGES = ('snowflake-snowpark-python', 'PyPDF2', 'langchain')

AS

$$

from snowflake.snowpark import types as T

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from snowflake.snowpark.files import SnowflakeFile

import PyPDF2, io

import logging

# Define the PDF text chunker class

class pdf\_text\_chunker:

def read\_pdf(self, file\_url: str) -> str:

logger = logging.getLogger("udf\_logger")

logger.info(f"Opening file {file\_url}")

# Open and read the PDF from the Snowflake stage

with SnowflakeFile.open(file\_url, 'rb') as f:

buffer = io.BytesIO(f.read())

# Use PyPDF2 to extract text from the PDF

reader = PyPDF2.PdfReader(buffer)

text = ""

for page in reader.pages:

try:

text += page.extract\_text().replace('\n', ' ').replace('\0', ' ')

except:

text = "Unable to Extract"

logger.warning(f"Unable to extract from file {file\_url}, page {page}")

return text

def process(self, file\_url: str):

# Extract text from the PDF file

text = self.read\_pdf(file\_url)

# Split the text into chunks using RecursiveCharacterTextSplitter

text\_splitter = RecursiveCharacterTextSplitter(

chunk\_size=4000, # Adjust chunk size as needed

chunk\_overlap=400, # Some overlap for better context

length\_function=len

)

# Split the extracted text into chunks

chunks = text\_splitter.split\_text(text)

# Yield the chunks one by one

for chunk in chunks:

yield (chunk,)

$$;

# **1.5 Create Chunk table**

CREATE OR REPLACE TABLE CHUNK\_CORTEX\_PDF (

RELATIVE\_PATH VARCHAR(16777216),

SIZE NUMBER(38, 0),

FILE\_URL VARCHAR(16777216),

SCOPED\_FILE\_URL VARCHAR(16777216),

CHUNK VARCHAR(16777216),

CHUNK\_VEC VECTOR(FLOAT, 768)

);

INSERT INTO CHUNK\_CORTEX\_PDF (relative\_path, size, file\_url, scoped\_file\_url, chunk, chunk\_vec)

SELECT

relative\_path,

size,

file\_url,

build\_scoped\_file\_url(@CORTEX\_STAGE, relative\_path) AS scoped\_file\_url,

func.chunk AS chunk, -- Chunk from the pdf\_text\_chunker function

SNOWFLAKE.CORTEX.EMBED\_TEXT\_768('e5-base-v2', func.chunk) AS chunk\_vec

FROM

DIRECTORY(@CORTEX\_STAGE) AS dir,

TABLE(pdf\_text\_chunker(build\_scoped\_file\_url(@CORTEX\_STAGE, dir.relative\_path))) AS func;

# **1.6 Create Cortex Search Service**

CREATE OR REPLACE CORTEX SEARCH SERVICE CORTEX\_SEARCH\_DB.CORTEX\_SEARCH\_SCHEMA.PDF\_CHUNK\_SERVICE

ON CHUNK

ATTRIBUTES RELATIVE\_PATH, FILE\_URL, SCOPED\_FILE\_URL, CHUNK

WAREHOUSE = CORTEX\_SEARCH\_WH

TARGET\_LAG = '1 hour'

AS (

SELECT \* FROM CHUNK\_CORTEX\_PDF

);

# 

# **1.7 Create Cortex Search Procedure**

CREATE OR REPLACE PROCEDURE cortex\_search\_procedure(

search\_query STRING,

filter\_column STRING,

filter\_value STRING,

result\_limit INT

)

RETURNS VARIANT

LANGUAGE SQL

EXECUTE AS CALLER

AS

$$

DECLARE

json\_query STRING;

BEGIN

-- Construct the JSON payload without explicit casting

json\_query := OBJECT\_CONSTRUCT(

'query', search\_query,

'columns', ARRAY\_CONSTRUCT('CHUNK', 'RELATIVE\_PATH', 'FILE\_URL'),

'filter', OBJECT\_CONSTRUCT('@eq', OBJECT\_CONSTRUCT(filter\_column, filter\_value)),

'limit', result\_limit

);

-- Call SEARCH\_PREVIEW function

RETURN PARSE\_JSON(

SNOWFLAKE.CORTEX.SEARCH\_PREVIEW(

'CORTEX\_SEARCH\_DB.CORTEX\_SEARCH\_SCHEMA.PDF\_CHUNK\_SERVICE',

json\_query

)

)['results'];

END;

$$;

CALL cortex\_search\_procedure('Explain Oncology', 'RELATIVE\_PATH', 'Cardiology\_Comprehensive\_Overview.pdf', 5);

# 

# **1.8 DOWNLOAD SNOWSQL** <https://drive.google.com/drive/u/1/folders/1jH_xQGKLh0EpG85qQUpAMOistpPnrlQQ>

# **1.9 GENERATE KEY IN SNOWSQL**

## 1.9.1 Open local terminal

## 1.9.2 Set a local location for token

C:\Users\User\Documents\snowsql

## 1.9.3 PRIVATE KEY :

openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out rsa\_key.p8 -nocrypt

## 1.9.4 PUBLIC KEY :

openssl rsa -in rsa\_key.p8 -pubout -out rsa\_key.pub

# **1.10 CHECK PRIVATE KEY**

openssl rsa -in C:\Users\User\Documents\JWT\_TOKEN\rsa\_key.p8 -check

**Example :**

writing RSA key

-----BEGIN PRIVATE KEY-----

MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDHHupTVJTe2u4X

JzK2X9zHoqEYKrDxJPKAoNKV/8y1ADpnJAdn7UBavrFWVwZBAhcxdFH+9HDjiT5U

S6XPCWsBZ4OZ5HZ6Q3nfE2FKhPQ6JX2Sl/JXe4PHOoZR8UAdLFwWfQrTn9yN756S

MyExzZ0ThWNxcfgua2XX/YyjP91ZzdC8ROsAamQlDJGnQiZwy24/0u/Y7ddfWkkQ

YYNAKYljpAdFIMpLhT7sd8S7l5ifq9py6U0L+uXhrkB/TqtXZTNmSVx7XIlfpmwQ

XwZv9qbYg2rd/GvWTMz1RA743MjQnIUISkcolNft5oOSQi+iNy1KN7APCbRbMnK1

58OkT64dAgMBAAECggEAGG4f2Fj+iegpAZC23V7eFuVKaA1m17NNiIASzkIL8uIk

CD8IpJTcijWuytXTB9tSjqgDnmkNx7DxQ4Tjg0cXyJyTbih0a9RHCdpWG787ur2a

TkfTvS2DkchJrhiQIfQtGTo+v4+zzvtJBI1+414jiT9an9LYnRNis7EhVe8yQH/z

OU4QbazfsiYCo4Z4ZR+wYZNg42QxkSWV6S3X8ophTQjyNsWtc1jCjZ+mblpBM9Jv

yYSPTRdQwofUqBH6XJKaFe2MezouQww8UmvXrUAqvR7gi4k12wuXnblozGixO4DQ

e7QIx1ZgIA1iE7j2mG81FE+8h2Be3pMm2uvXV+HKcQKBgQDw5ZN2p0EgDmQ6Oel4

x1bGTR6JeJABRb8qkPTuboKdzTAPSFbzxZMFQJQhA/pqiOp8s85RWFOjIoapTR/L

ouvSFedajGpqBOcUgFst2SN/VlhG5gd4YwdeZ89bx/C7hrIBJd9O9hwf8emH26Mw

HCmIi5ZhGLoNqV/C8x5fNK0xbQKBgQDTmtQysvM9xJrdbnD7Kwp6C6AcxlMXprbb

IutQ0zOf52iibJ9LqWoveJug6ll/wB7EVl8tebRuL7TZXbujlek2omqrmrKfTa2W

sSUfFUYSw9KH0p5tW20k9xvfKC7RvOGHhXd9haq2lfSnTtQX8EVDhvm6KBGjn78T

okISuaMxcQKBgQDMT+piGDkEFzCqsbNy3J6cUYw0zCxW5/DbCVazYsclT4ITdCdL

Wab4LW11EwqwkK9QVLAIDgHhusa7kM4GyId6B1nRUoZupPnYSYULvb59oZ39g9uS

H2uAhks4bLugUP1oO9bKAbKY5XRL5dziUQBo7vr8HxytuZmentSDqwm5gQKBgC06

ktCjX0p8k31w9cU51etwQLDl0yzi/YKpXBLSkd5D4CGXhhTuCTxMUzd539dmgS0G

KSHXObGzkLxFRK4ahNagsJwtXLCgMdJb2stiCLBKGt3dLu/A8vtLtjFkrV03Oh11

cazqvSp5sulv9/WF7cgZOZeIqPyIUJcpv2WxOCiBAoGAaNdsoFK181PCNtoPsrAX

AZusFjXViNAhH/V+R6/iELWLlSYX+ksY3dCGyicEE0ttOEx5EzF1FgdOi15Ir98R

3m1N7HXbITTzNVsn2UaefwdiBJBPI+F9N8pbYTCZ+AAnG+nIgihbUncjMxGV7lv0

w6KrPOlvCW3IgFEg5RvJFtY=

-----END PRIVATE KEY-----

# **1.11 CHECK PUBLIC KEY**

type rsa\_key.pub

**Example :**

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxx7qU1SU3truFycytl/c

x6KhGCqw8STygKDSlf/MtQA6ZyQHZ+1AWr6xVlcGQQIXMXRR/vRw44k+VEulzwlr

AWeDmeR2ekN53xNhSoT0OiV9kpfyV3uDxzqGUfFAHSxcFn0K05/cje+ekjMhMc2d

E4VjcXH4Lmtl1/2Moz/dWc3QvETrAGpkJQyRp0ImcMtuP9Lv2O3XX1pJEGGDQCmJ

Y6QHRSDKS4U+7HfEu5eYn6vaculNC/rl4a5Af06rV2UzZklce1yJX6ZsEF8Gb/am

2INq3fxr1kzM9UQO+NzI0JyFCEpHKJTX7eaDkkIvojctSjewDwm0WzJytefDpE+u

HQIDAQAB

-----END PUBLIC KEY-----

# **1.12 RUN THE JWT GENERATED ALTER QUERY**

ALTER USER CORTEXSEARCHTOKEN SET RSA\_PUBLIC\_KEY='MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA3mYXkRLA9a5RiRuufFGA

DSgwugWLckcf0/ZnBiz3cLgYwZ23sYZzMIkBljaUndJTIShIwBka9kslTxPbOFn3

0jae8mgAF+32cNwA4WxiinkxA2Qy5by5g0UWMpw9VQQacNJMF8Qcm+Bky9b56iD8

KmHe9Wlut5vLV2xo4sCY4U9PChlEp5s5FPsiSgpgx+9uq61yzty64A7gnBuE7XBj

g/HXWPDqasya4RVVY6sGFW92e3IYzZB8RolDNEjimSXWt2OfolUy0TzcniE7BGQC

Umb++7ZD8FFil6gcZr46gVfcB74NtXNzWfDYiHxXoPd75po8mk7ht5YRGdUIiY0v

YwIDAQAB';

# 

# **1.13 LOGIN USING FOR SNOWFLAKE ACCESS IN SNOWSQL**

snowsql -a akb21464 -u CORTEXSEARCHTOKEN

# **1.14 ACCESS THE DEFAULT ROLE**

use role accountadmin;

# **1.15 GRANT THE DEFAULT ROLE TO NEW ROLE**

GRANT ROLE ACCOUNTADMIN TO ROLE CORTEX\_SEARCH\_TOKEN; ""THE SHOW TRUE IN has\_rsa\_public\_key""

NOTE : CHECK WHETHER has\_rsa\_public\_key is TRUE WITHOUT ACCESS IS GRANTED

# **1.16 GENERATE JWT TOKEN**

snowsql -a akb21464 -u CORTEXSEARCHTOKEN --private-key-path C:/Users/User/Documents/JWT\_TOKEN/rsa\_key.p8 --generate-jwt

# **1.17 USING THE GENERATED TOKEN CALL THE CORTEX SEARCH PROCEDURE**

**PYTHON FILE :** Generator\_jwt.py

import requests

# Use the JWT token you just generated

jwt\_token = "<your-jwt-token>"

# Snowflake account and API endpoint

account = "vyb82036"

api\_endpoint = f"https://{account}.[snowflakecomputing.com/api/v2/statements](http://snowflakecomputing.com/api/v2/statements)"

# SQL statement to call your stored procedure

sql\_statement = "CALL cortex\_search\_procedure('Explain Oncology', 'RELATIVE\_PATH', 'Cardiology\_Comprehensive\_Overview.pdf', 5);"

# Construct the request payload

payload = {

"statement": sql\_statement,

"timeout": 60, # timeout in seconds

"database": "CORTEX\_LAB\_DB",

"schema": "CORTEX\_LAB\_SCHEMA",

"warehouse": "CORTEX\_LAB\_WH"

}

# Set up the headers (use Bearer token format)

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {jwt\_token}" # Use Bearer token format for JWT

}

# Send the request

response = [requests.post](http://requests.post)(api\_endpoint, json=payload, headers=headers)

# Handle the response

if response.status\_code == 200:

result = response.json()

print("Stored procedure executed successfully:", result)

else:

print(f"Error executing stored procedure (Status Code: {response.status\_code}):", response.text)

Attached drive link for your reference

<https://drive.google.com/drive/u/1/folders/1jH_xQGKLh0EpG85qQUpAMOistpPnrlQQ>

# **1.18 POSTMAN REFERENCE**

Have attached the reference image in the email. Please refer to it.

## **1.18.1 Request Post URL**

<https://tufbwjl-qeb33118.snowflakecomputing.com/api/v2/statements>

## **1.18.2 Headers**

Content-Type : application/json

X-Snowflake-Authorization-Token-Type : KEYPAIR\_JWT

Authorization : Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJWWUI4MjAzNi5DT1JURVhTQS5TSEEyNTY6Rk9xRVVtQ1J2SXA3RVR6b0U2MjcwNldhRFZWTk9wVmZVRFRaNTVwYzM1Zz0iLCJzdWIiOiJWWUI4MjAzNi5DT1JURVhTQSIsImlhdCI6MTcyNzE3MzY2MywiZXhwIjoxNzI3MTc3MjAzfQ.vnqpqpgPX\_zmbx5cFbb9ImoBxMY1uvqUIIKs3J8I4MDCyh\_1YBTiA9ZpKKDAk\_uOVV6JpFuEY8-q5yxbNuEAGnny\_AR44iyB-MUiRAEbEAs14flCOreX05IXfGwu8jxVnc43n-Tum88o29JkmoQrmlmEtXiP94ovnWOYtnGuC2WXxw5Oy62ccrP0m96bmiVdloX13MyDO5yZe9SaCJN2\_FxxUy\_kxITCZpxJoU1YcXThH5-CoTA2WYNJ7sTICZUj3Psk3fYrlWzW9kCDNz4jVenTcbRakSvA30nY4nriOJHVmr2ffaQgvTLPZ\_NJqSPSvZOZmpQL-0Dr36qhp23sSw

## **1.18.3 Body**

{

"statement": "CALL cortex\_search\_procedure('Explain Oncology', 'RELATIVE\_PATH', 'Cardiology\_Comprehensive\_Overview.pdf', 5);",

"timeout": 60,

"database": "CORTEX\_SEARCH\_DB",

"schema": "CORTEX\_SEARCH\_SCHEMA",

"warehouse": "CORTEX\_SEARCH\_WH",

"role": "CORTEX\_SEARCH\_TOKEN"

}

**CORTEX SEARCH VIA STORED PROCEDURE → REST API**

**(WITHOUT PROCEDURE)**

**NOTES :** THE QUERY IS USED ON **ACCOUNTADMIN** WITHOUT NEW ROLE .CREATE A NEW DATABASE (CORTEX\_SEARCH\_DB1) , NEW SCHEMA (CORTEX\_SEARCH\_SCHEMA1) AND USE COMPUTE\_WH.

## **1.1 POSTMAN REFERENCE - EXAMPLE**

### 1.1.1 Request Post URL

### <https://tufbwjl-qeb33118.snowflakecomputing.com/api/v2/databases/CORTEX_SEARCH_DB1/schemas/CORTEX_SEARCH_SCHEMA1/cortex-search-services/PDF_CHUNK_SERVICE:query>

### 1.1.2 Headers

Content-Type : application/json

X-Snowflake-Authorization-Token-Type : KEYPAIR\_JWT

Authorization : Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJWWUI4MjAzNi5DT1JURVhTQS5TSEEyNTY6Rk9xRVVtQ1J2SXA3RVR6b0U2MjcwNldhRFZWTk9wVmZVRFRaNTVwYzM1Zz0iLCJzdWIiOiJWWUI4MjAzNi5DT1JURVhTQSIsImlhdCI6MTcyNzE3MzY2MywiZXhwIjoxNzI3MTc3MjAzfQ.vnqpqpgPX\_zmbx5cFbb9ImoBxMY1uvqUIIKs3J8I4MDCyh\_1YBTiA9ZpKKDAk\_uOVV6JpFuEY8-q5yxbNuEAGnny\_AR44iyB-MUiRAEbEAs14flCOreX05IXfGwu8jxVnc43n-Tum88o29JkmoQrmlmEtXiP94ovnWOYtnGuC2WXxw5Oy62ccrP0m96bmiVdloX13MyDO5yZe9SaCJN2\_FxxUy\_kxITCZpxJoU1YcXThH5-CoTA2WYNJ7sTICZUj3Psk3fYrlWzW9kCDNz4jVenTcbRakSvA30nY4nriOJHVmr2ffaQgvTLPZ\_NJqSPSvZOZmpQL-0Dr36qhp23sSw

### 

### 1.1.3 Body

{

"query": "Explain Oncology",

"columns": [

"CHUNK",

"RELATIVE\_PATH",

"FILE\_URL"

],

"filter": {

"@eq": {

"RELATIVE\_PATH": "Cardiology\_Comprehensive\_Overview.pdf"

}

},

"limit": 5,

"experimental": {

"ut4": **true**,

"dolore3": 37155695.962142885,

"incididuntbb9": -80406148

}

}