# College of AI: Document AI HOL

## Extract unstructured text from PDFs into a structured Snowflake table for downstream applications

Value Proposition:

* [Document AI](https://docs.snowflake.com/en/user-guide/snowflake-cortex/document-ai/overview) is a Snowflake AI feature designed to streamline the process of extracting insights from unstructured data (and more documents). It is powered by Arctic-TILT (Text Image Layout Transformer), a proprietary 0.8B parameter model 25, allowing customers to batch process documents of varying formats, identify key information, and materialize it in a queryable format for further analytics. With Document AI, customers can produce and automate intelligent document processing workflows entirely within Snowflake.
* [Ideal for](https://docs.snowflake.com/en/user-guide/snowflake-cortex/document-ai/overview#when-to-use-document-ai) processing PDFs and other unstructured documents for quicker processing and faster decision making at enterprise scale using a natural language interface using an industry-leading LLM that drives higher efficiency, less manual labor, lower human error, and single-click fine-tuning if needed.
* Document AI benefits include:
  + Proximity to Data → minimize latency by processing where your data already resides
  + No ML expertise or APP Dev needed → Use a natural language interface with the ability to fine-tune models.
  + Purpose-built proprietary model → Utilize a cutting-edge proprietary model purpose-built for business document extractions
  + Fully Managed and Governed → Serverless. No need to deploy models, test them, secure them, upgrade them, or scale GPU clusters.

Discovery Questions:

* Ask your customer if they would like to automate and scale-up their ability to extract data from any of the following:
  + Documents (PDF, PNG, Docx, Emails, Txt)
  + Scanned Documents (Forms)
  + Images of Documents (Photos of Documents , Receipts, Driver’s License)
  + Documents with Infographics (Checkboxes, Graphs, Images)
  + Documents with Handwritten Text

Helpful Background Information:

* Review Snowflake [Document AI Documentation](https://docs.snowflake.com/LIMITEDACCESS/document-ai/index)
  + Please note the [Document AI access control](https://docs.snowflake.com/en/user-guide/snowflake-cortex/document-ai/setting-up#label-document-ai-access-control) to grant the [SNOWFLAKE.DOCUMENT\_INTELLIGENCE\_CREATOR](https://docs.snowflake.com/en/user-guide/snowflake-cortex/document-ai/tutorials/create-processing-pipelines#set-up-the-required-objects-and-privileges) database role to an account role, and then grant the account role to users.
* [Full list of Document AI assets](https://docs.google.com/spreadsheets/d/1zFgGs3Io2KibmDd6L7Hp6I8Q2VVxH5yUmdONl0U9xTg/edit?gid=0#gid=0) including:
  + Product Overview [Deck](https://docs.google.com/presentation/d/1RZsXISQc0bL4nU5hL5W7jMgspjoEp5Jk7zzhQ7ypNIw/edit#slide=id.g1ed82e8067d_0_2803)
  + Feature Friday: [Document AI (GA July) - May 31, 2024 Feature Training](https://snowflake.seismic.com/app#/learning/library/lesson/2301849)
  + Review Document AI [PrPr Troubleshooting Guide](https://docs.google.com/document/d/1ttiPYEn1SfO2KNoKsBes2Wjvhq3s2A8hnDRx02iRig8)
  + [Document AI Best Practises Guide](https://docs.google.com/document/d/1SyaVZciVcLkPx5YDboLC5wNJJVN467bVr7ugiULvFfU/edit)
  + [Document AI FAQ](https://docs.google.com/document/d/1NylhfMWyI4rvLRTwL5_vZ8B4PLlJwlQRG0HFF5er8Mk/edit#heading=h.s9xgdcwcq7u6)
  + [Battlecard for Document AI and Intelligent document](https://docs.google.com/spreadsheets/d/1o6PpItcuCwlvw7YFo7pQ3QMNniMWWh0uvFH0lEghDR8/edit?usp=sharing) processing using Snowflake (head to head comparisons & in-detail reviews)
* More [Document AI resources](https://snowflakecomputing.atlassian.net/wiki/spaces/EN/pages/2749924956/Document+AI)
* Document AI requires you users to use a role like DOC\_AI\_ROLE. You cannot run as ACCOUNTADMIN. See this [link](https://docs.snowflake.com/en/user-guide/snowflake-cortex/document-ai/setting-up#document-ai-access-control) for more information.
  + Here is code to configure Document AI for your username:

USE ROLE ACCOUNTADMIN;

-- Create the database and schema if they do not exist

CREATE DATABASE IF NOT EXISTS AICOLLEGE;

CREATE SCHEMA IF NOT EXISTS AICOLLEGE.PUBLIC;

-- Create the stage (replace STORAGE\_INTEGRATION and URL as needed)

CREATE OR REPLACE STAGE AICOLLEGE.PUBLIC.RESUMES

DIRECTORY = ( ENABLE = true )

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

-- Create role if it does not already exist

CREATE OR REPLACE ROLE doc\_ai\_role;

-- Grant warehouse access

GRANT USAGE, OPERATE ON WAREHOUSE AICOLLEGE TO ROLE doc\_ai\_role;

-- Grant access to Document Intelligence database role

GRANT DATABASE ROLE SNOWFLAKE.DOCUMENT\_INTELLIGENCE\_CREATOR TO ROLE doc\_ai\_role;

-- Grant usage on database and schema

GRANT USAGE ON DATABASE AICOLLEGE TO ROLE doc\_ai\_role;

GRANT USAGE ON SCHEMA AICOLLEGE.PUBLIC TO ROLE doc\_ai\_role;

-- Grant read access on stage, and create privileges

GRANT READ ON STAGE AICOLLEGE.PUBLIC.RESUMES TO ROLE doc\_ai\_role;

GRANT CREATE STAGE ON SCHEMA AICOLLEGE.PUBLIC TO ROLE doc\_ai\_role;

GRANT CREATE SNOWFLAKE.ML.DOCUMENT\_INTELLIGENCE ON SCHEMA AICOLLEGE.PUBLIC TO ROLE doc\_ai\_role;

-- Grant create permissions on schema

GRANT CREATE STREAM, CREATE TABLE, CREATE TASK, CREATE VIEW ON SCHEMA AICOLLEGE.PUBLIC TO ROLE doc\_ai\_role;

-- Grant create SiS app

GRANT CREATE STREAMLIT ON SCHEMA AICOLLEGE.PUBLIC TO ROLE doc\_ai\_role;

-- Grant account-level task execution

GRANT EXECUTE TASK ON ACCOUNT TO ROLE doc\_ai\_role;

-- Grant the role to a specific user (replace with your username)

GRANT ROLE doc\_ai\_role TO USER <your username>;

-- Optional: ensure ACCOUNTADMIN can access future tables

GRANT SELECT ON FUTURE TABLES IN DATABASE AICOLLEGE TO ROLE ACCOUNTADMIN;



* Snowflake Arctic-Tilt model benchmark performance achieved a 90.20 ANLS score 26. See this [leaderboard](https://rrc.cvc.uab.es/?ch=17&com=evaluation&task=1) to compare with other models like GPT4 Vision with Amazon Textract, LayoutLM from MSFT, SMoLA-PALI-X from Google
* Official TILT paper on [arxiv](https://arxiv.org/abs/2102.09550) to provide additional information about the scope and performance of the DocAI model.
* Document AI is available in AWS and Azure commercial regions, with the exception of AWS Asia Pacific (Singapore), AWS Asia Pacific (Osaka), and AWS EU (Paris).

### Lab Overview:

In this hands-on lab, we will use Snowflake's Document AI to extract critical information from resumes. This data will be used in a downstream application to help qualify or disqualify candidates. You will:

**Extract Critical Information:**

Use Document AI to extract over 10 key items from each resume, such as name, contact information, education, work experience, skills, and certifications.

**Create a Query Pipeline (Optional):**

Set up a query pipeline with the Document AI extracting function for continuous processing of new resumes using Snowflake Streams and Tasks.

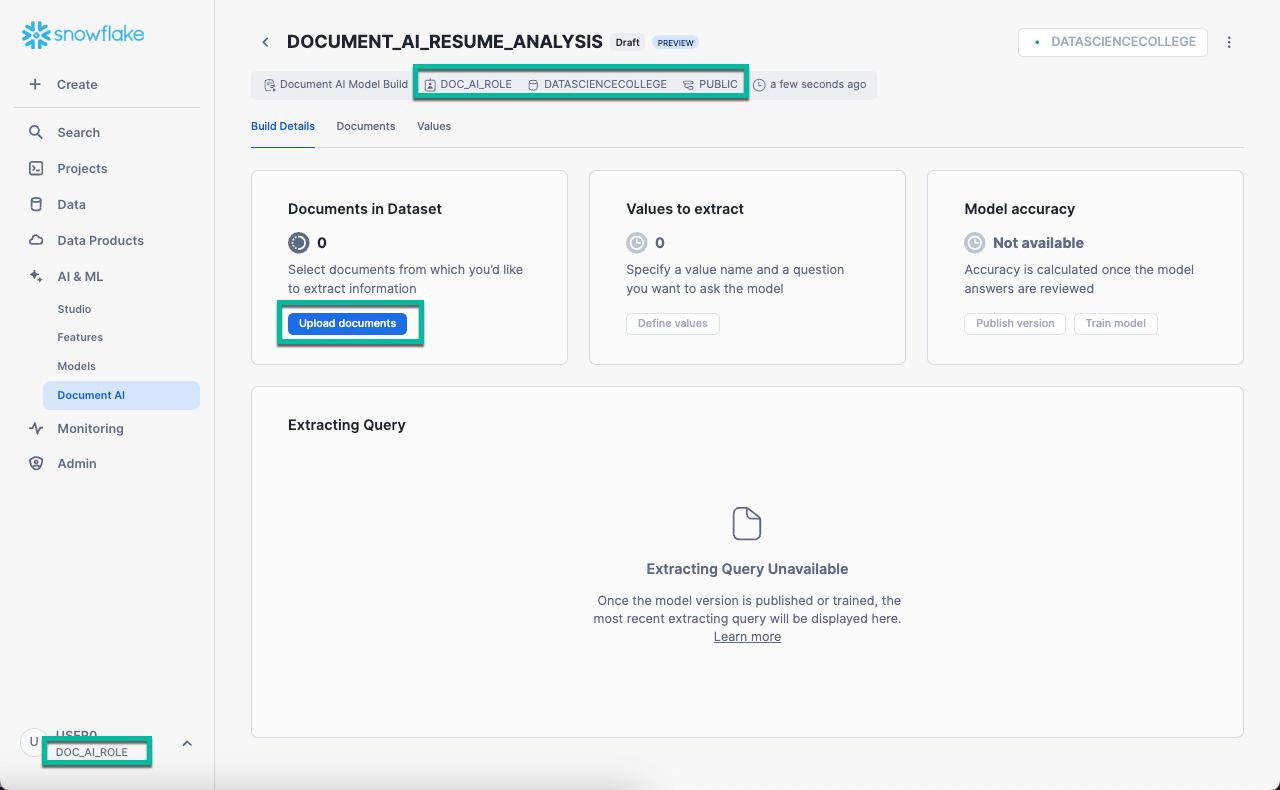
**Build a Streamlit Application (Optional):**

Develop a Streamlit app within Snowflake to visualize, review, and manage the extracted resume details, allowing for corrections and decisions on candidate qualification.

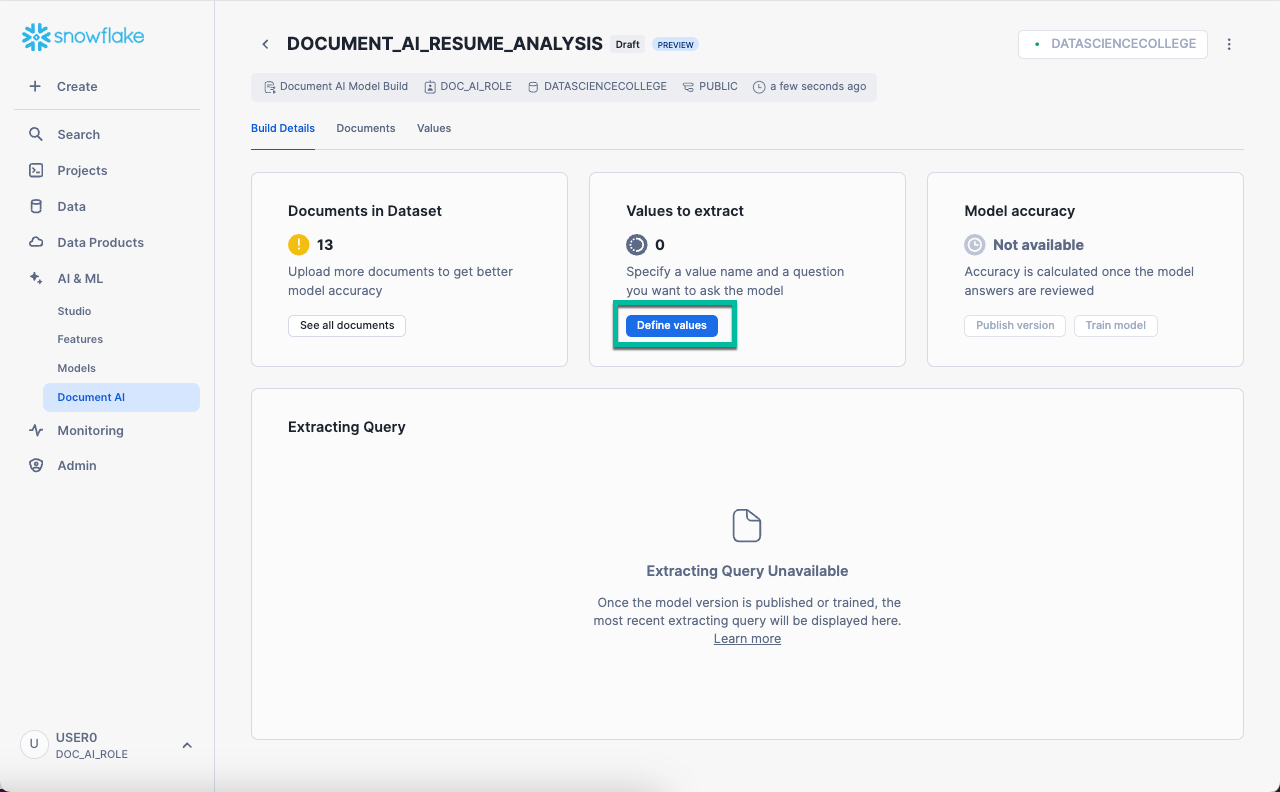
By the end of this lab, you will have hands-on experience with Snowflake's Document AI and the tools to streamline resume processing for candidate qualification.

### Part 1: Document AI Training

* Using the **DOC\_AI\_ROLE**
* Select + Build
* Build name “**DOCUMENT\_AI\_RESUME\_ANALYSIS**”
* Select "AICOLLEGE" for the database, "PUBLIC" for the schema
* Add description like “SE College of AI hands-on-lab for Document AI”
* Select Create

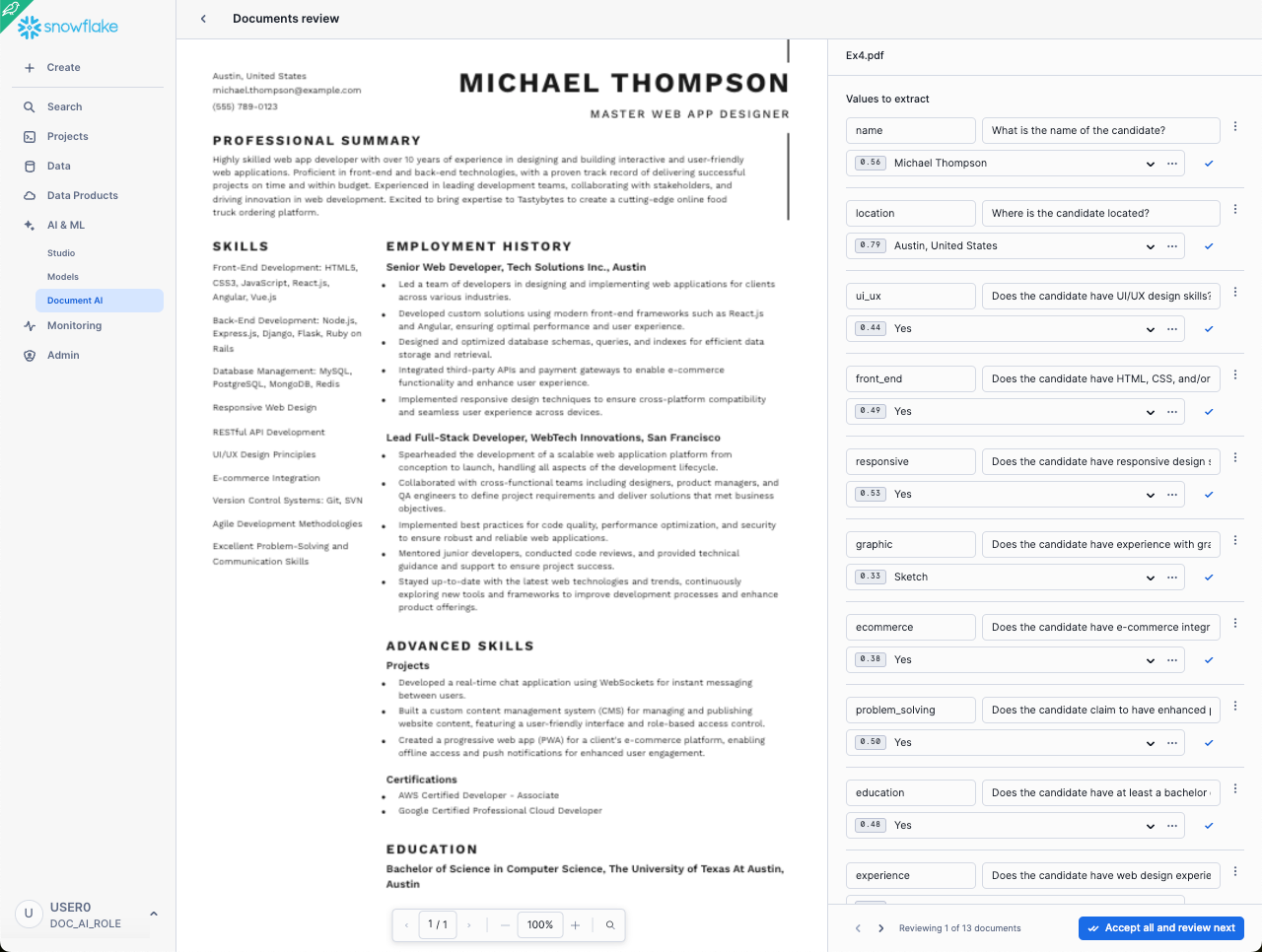


* Upload the [13 .pdf resumes available here](https://drive.google.com/drive/folders/1A90OQKKzJTuoQoZuSBMDsj_VoPUOeLeg) (Ex1.pdf through Ex13.pdf) to train the zero-shot model
* See [question optimization documentation](https://docs.snowflake.com/LIMITEDACCESS/document-ai/optimizing-questions) for extracting information with Document AI
  + Use plain English
  + Know the expected answers beforehand
  + Be specific with questions, especially for documents with multiple similar fields
  + Ask for single values in each question
  + Avoid expecting the model to make assumptions or use domain knowledge.
* Select **Define Values** to enter specific values for your model. Document AI values consist of a value name and a question you want to ask the model. If the documents are loaded and the option to select “Define Values” does not appear, refresh your browser.



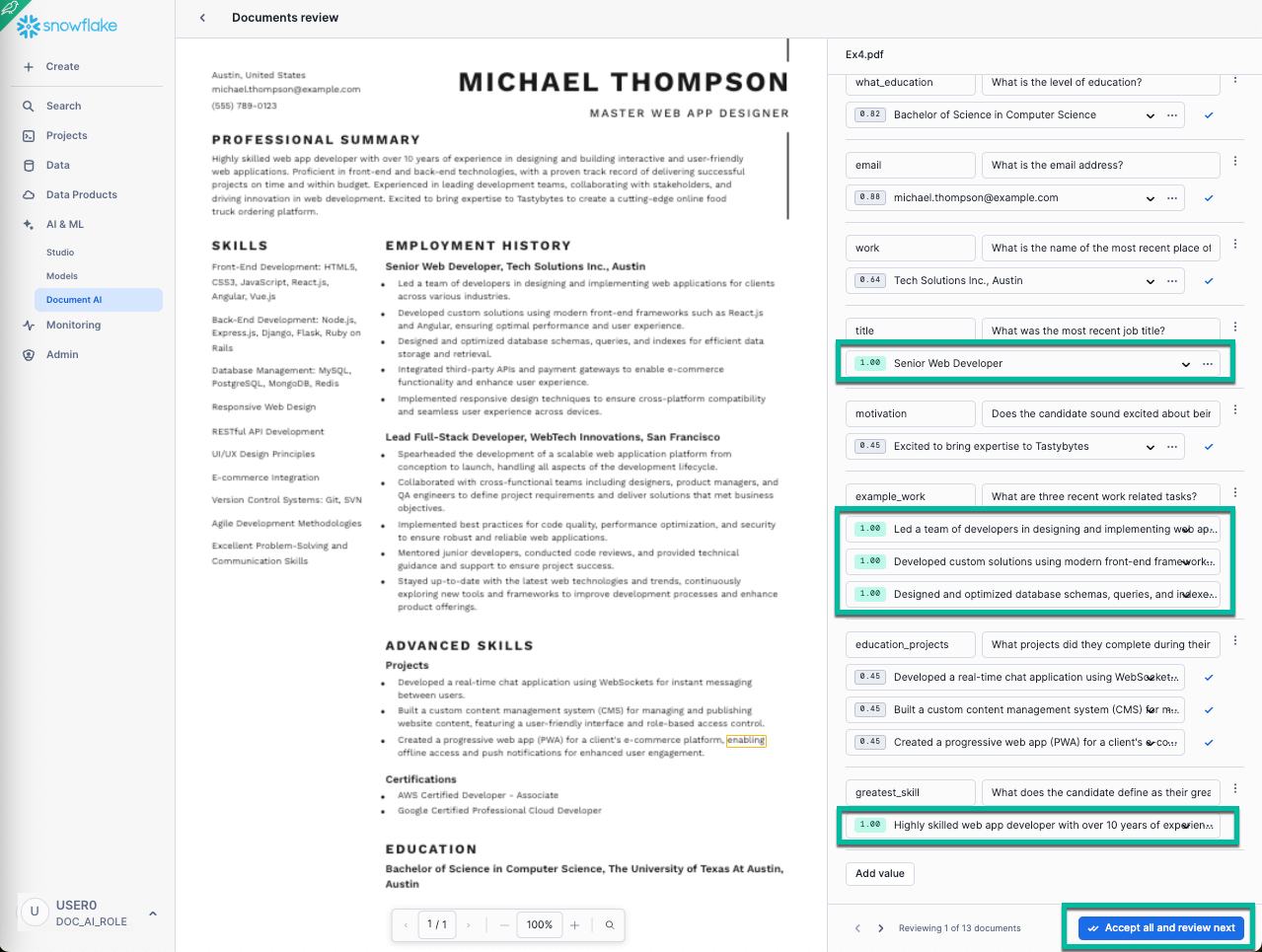
* Use the following information to create entities:

| **Entity Name** | **Entity Question** |
| --- | --- |
| name | What is the name of the candidate? |
| location | Where is the candidate located? |
| ui\_ux | Does the candidate have UI/UX design skills? |
| front\_end | Does the candidate have HTML, CSS, and/or JavaScript skills? |
| responsive | Does the candidate have responsive design skills to work across different devices? |
| graphic | Does the candidate have experience with graphic design tools like Adobe Photoshop, Illustrator, or Sketch? |
| ecommerce | Does the candidate have e-commerce integration skills? |
| problem\_solving | Does the candidate claim to have enhanced problem solving skills? |
| education | Does the candidate have at least a bachelor degree? |
| experience | Does the candidate have web design experience? |
| jobs | Does the candidate have more than one web design job experience? |
| what\_education | What is the highest level of education? |
| email | What is the email address? |
| work | What is the name of the most recent place of work? |
| title | What was the most recent job title? |
| motivation | Does the candidate sound excited about being a TastyByte web app developer? |
| example\_work | What are three recent work related tasks? |
| education\_projects | What projects did they complete during their educational studies? |
| greatest\_skill | What does the candidate define as their greatest strength? |

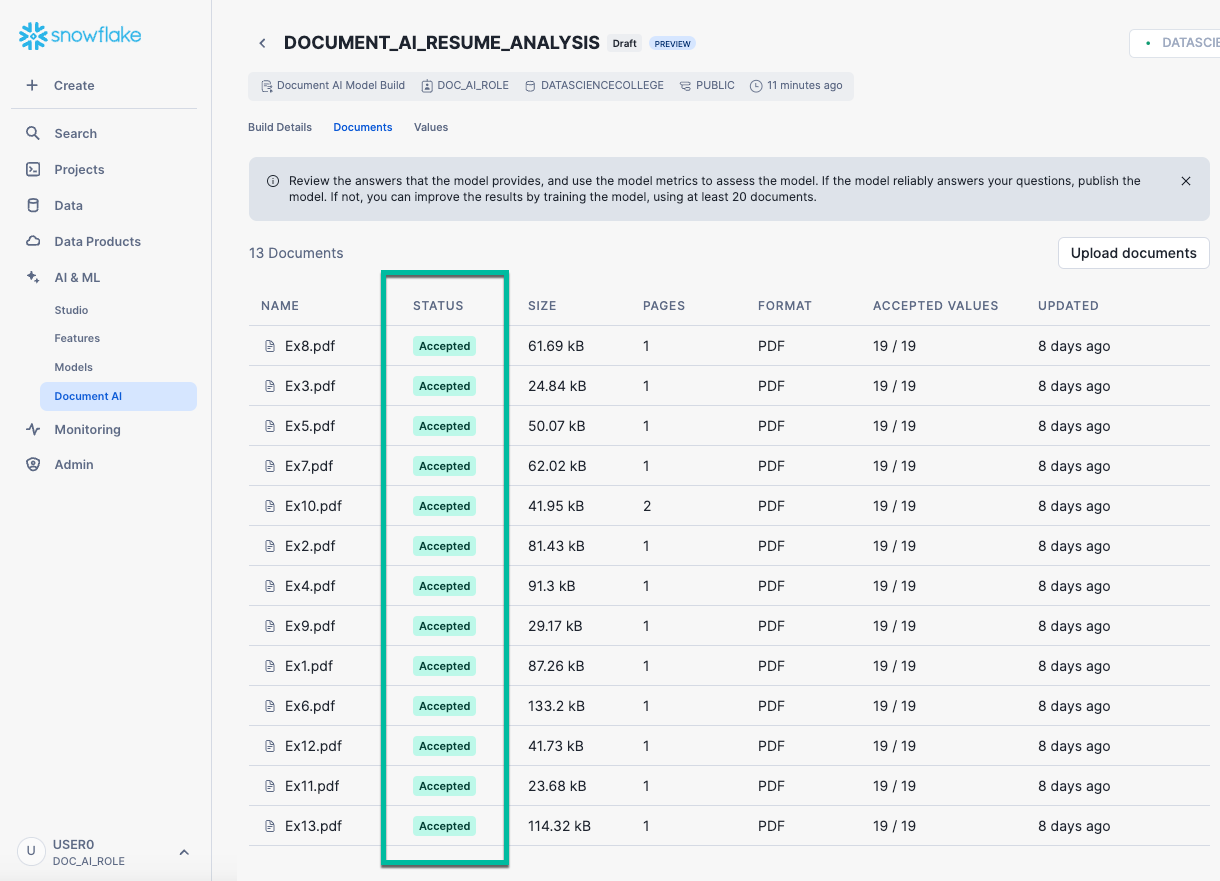


### Part 2: Fine-Tune Document AI LLM

* Select each resume and review the Document AI zero-shot results
* Correct Document AI answers as appropriate
* To assist, [refer to this file](https://docs.google.com/spreadsheets/d/162cV9GLBX4UDVgEY7g9Ovz6APKRPnRhtQaIRLa6SFVY/edit?usp=share_link) for the correct answers. Please DO NOT SPEND TOO MUCH TIME making everything match. It’s not important. Goal is to have the 13 resumes reviewed by Document AI, generate predictions (zero-shot is sufficient for the demo) and generate a downstream SiS app to explain the workflow with customers.
* You’ll notice the DORA for this HOL will be very simple to pass. That said, we want you to understand the value of Document AI and its workflow.

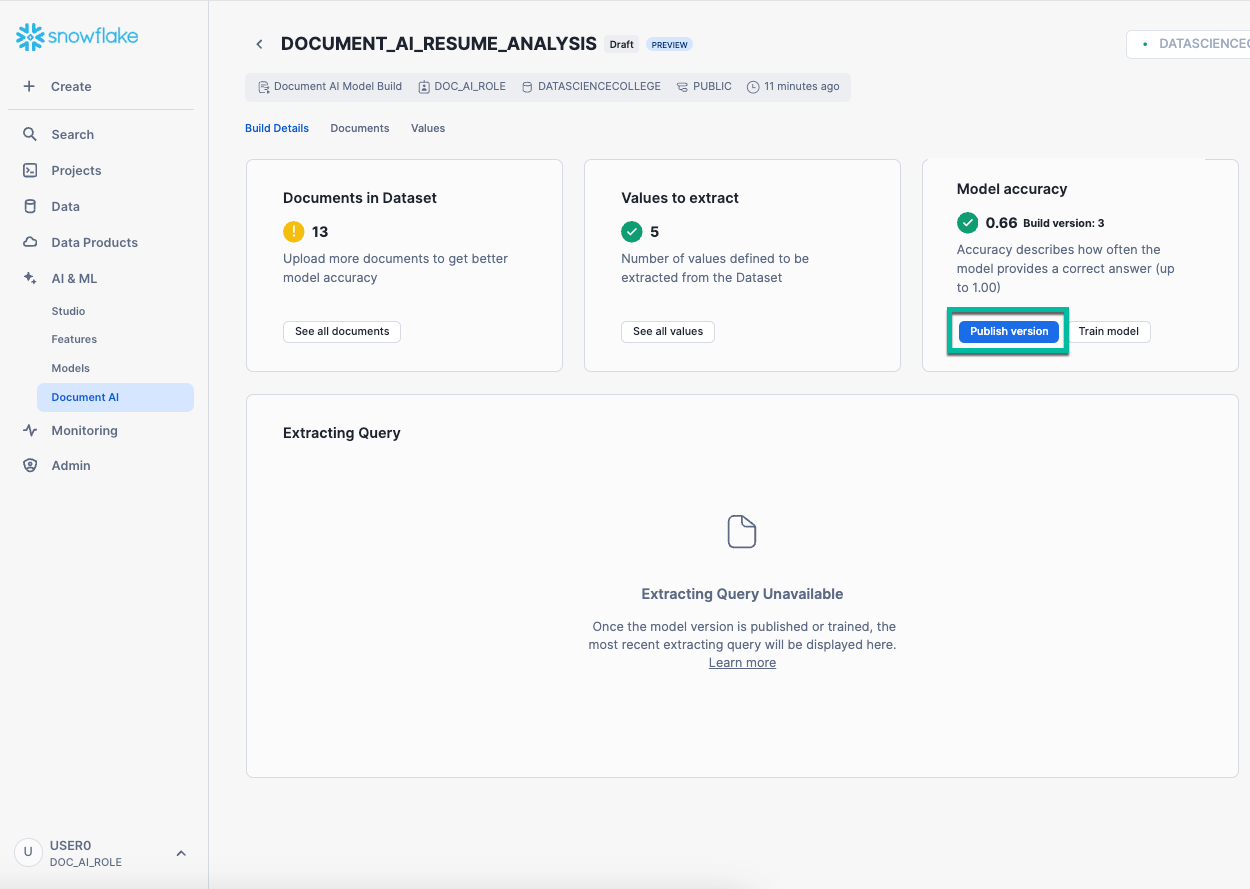


* Once you have reviewed all 13 resumes and selected “Accept all and review next”, select the **<** to the left of the Documents review, then select “**Publish**” button



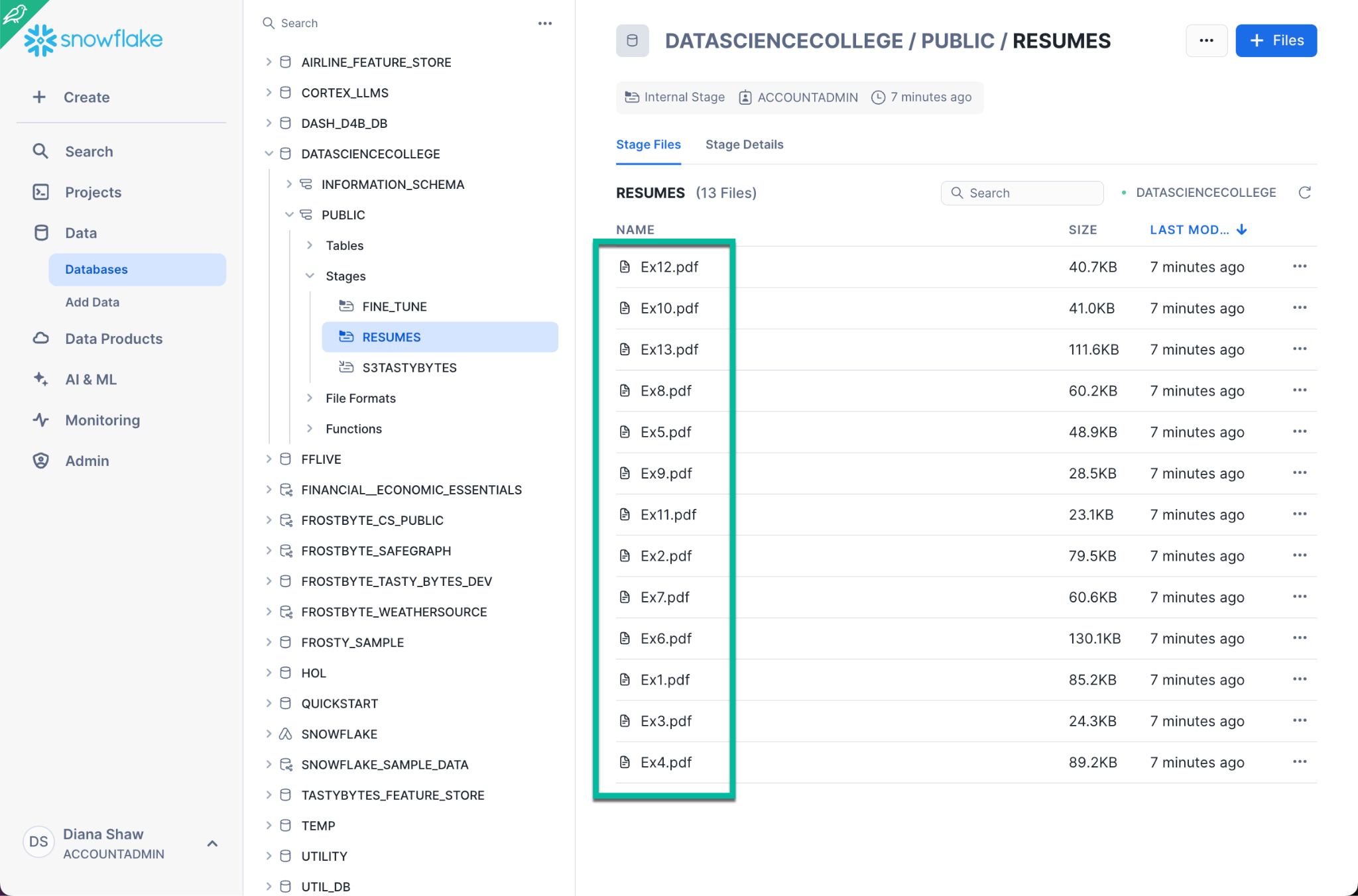
### Part 3: Assess Document AI Results

* Go to the **Build Details** tab of the DOCUMENT\_AI\_RESUME\_ANALYSIS project
* Use the model accuracy score below to evaluate the model.
* In order to improve the model quality, you can add more documents to the dataset, review and correct answers, and continue training until you get the desired quality. You will need to create new resumes if you wish to further improve the model.
  + The original 13 resumes were intended to be for example purposes only.
  + In production, customers would want to load and review at least 50 resumes.



### Part 4: Use Document AI extracting query to create text extraction

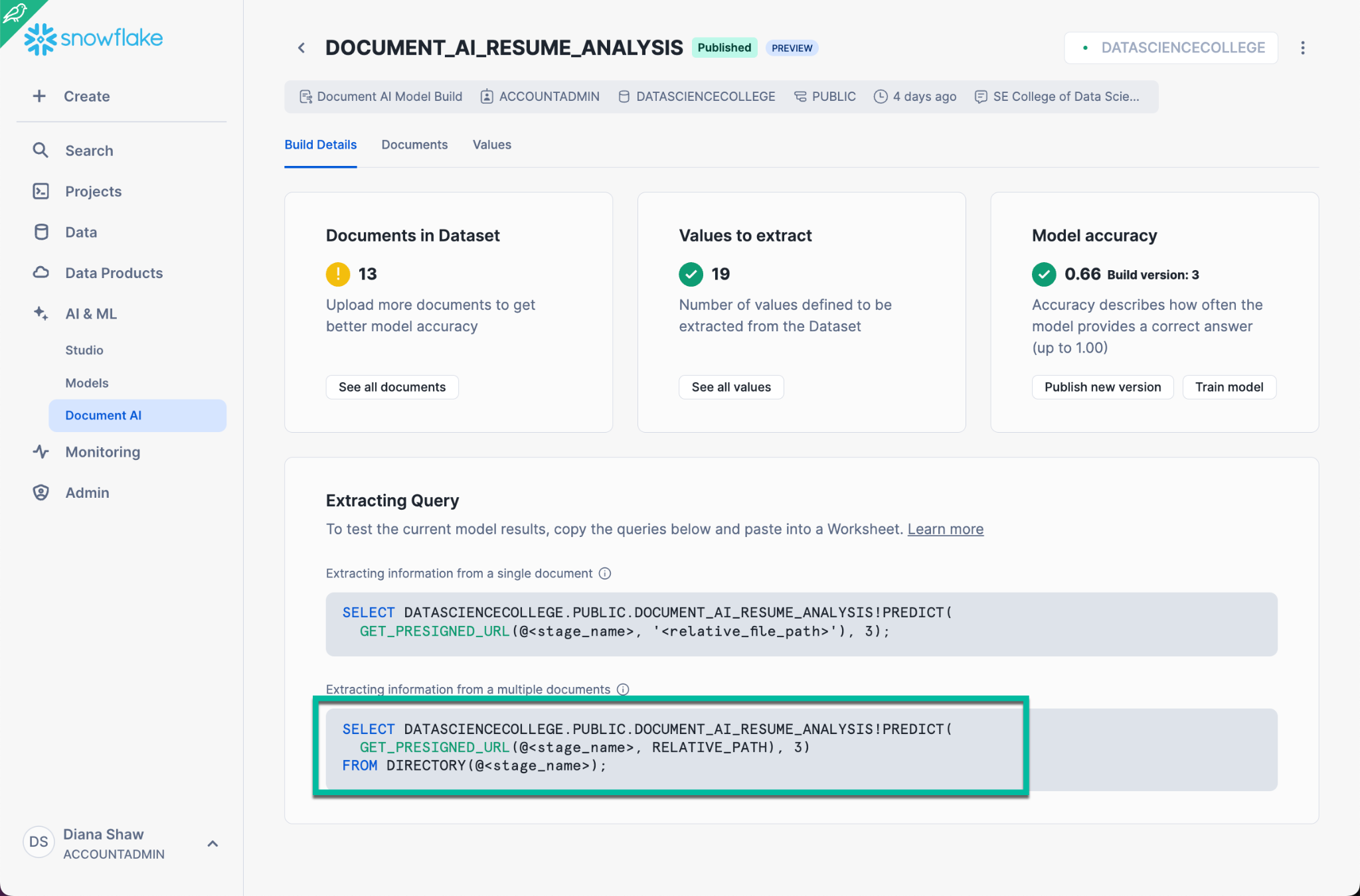
* Using the **DOC\_AI\_ROLE**
* After "Training" and “Fine-Tuning” the Document AI project, you need to run the PREDICT function
* This will extract information from documents on a stage, and will provide answers in a JSON object for each document on the stage.
* See [Document AI Documentation](https://docs.snowflake.com/LIMITEDACCESS/document-ai/predict) for more information
* Use your recently published “**DOCUMENT\_AI\_RESUME\_ANALYSIS**” model
* Use the **PREDICT** function on the 13 resumes loaded the Snowflake Managed Stage **SSE** called **RESUMES** you created in step 5 of the [HOL setup](https://docs.google.com/document/d/1z-CG06Kt2dzV2bLxTDsP55qxQY6lzwtSDE5Azz4IwJs/edit?usp=sharing)
* Navigate to the **AICOLLEGE.PUBLIC.RESUMES stage** and **load all 13 resumes** into this stage.
  + CAUTION: If you did not specify SSE when creating the stage, Document AI extracting query will fail.



* Ensure you grant READ access for your DOC\_AI\_ROLE on this stage

GRANT READ ON STAGE AICOLLEGE.PUBLIC.RESUMES TO ROLE doc\_ai\_role;

* Copy the extracting information from a multiple documents from the Build Details UI



* In a SQL worksheet, let’s use the copied extracting query that calls the Document AI PREDICT function
* Use this code to get you started and update as necessary:

-- Use Document AI PREDICT function for batch scoring your Document AI model

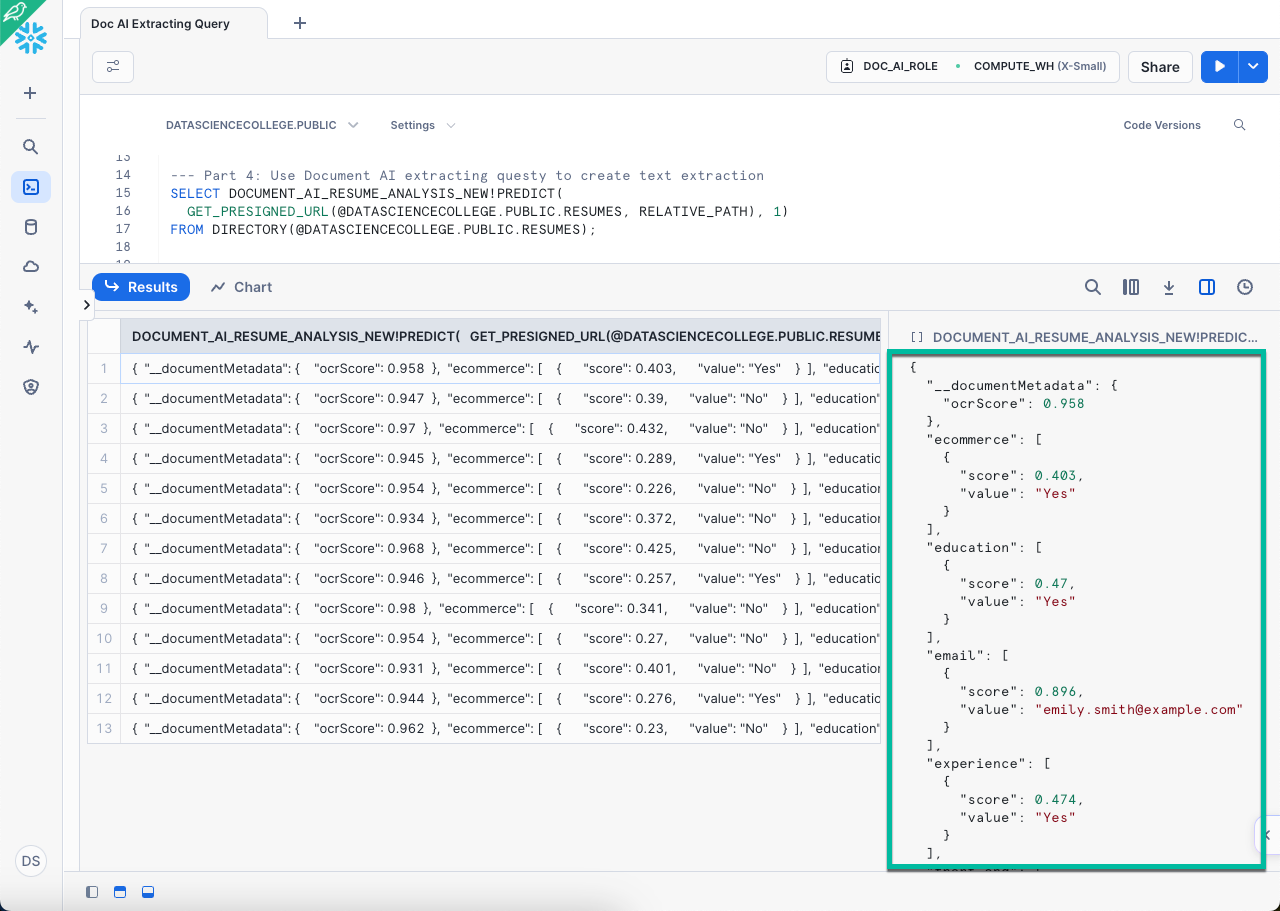
SELECT ???!???(

GET\_PRESIGNED\_URL(@???, RELATIVE\_PATH), 1)

FROM DIRECTORY(@???);



* Run the code and validate your json output includes information for each Document AI “value”



* Now let’s save the PREDICT results as a new table called “**DOCAI\_RESUMES**”
* Rename the resulting column as **src** with **variant** type for use in future instructions

-- Save results from Document AI PREDICT function into DOCAI\_RESUMES table

CREATE TABLE ??? (src variant) AS (

SELECT \* FROM (SELECT ???!PREDICT(

GET\_PRESIGNED\_URL(@AICOLLEGE.PUBLIC.RESUMES, RELATIVE\_PATH), 1) as JSON

FROM DIRECTORY(@AICOLLEGE.PUBLIC.RESUMES)

));



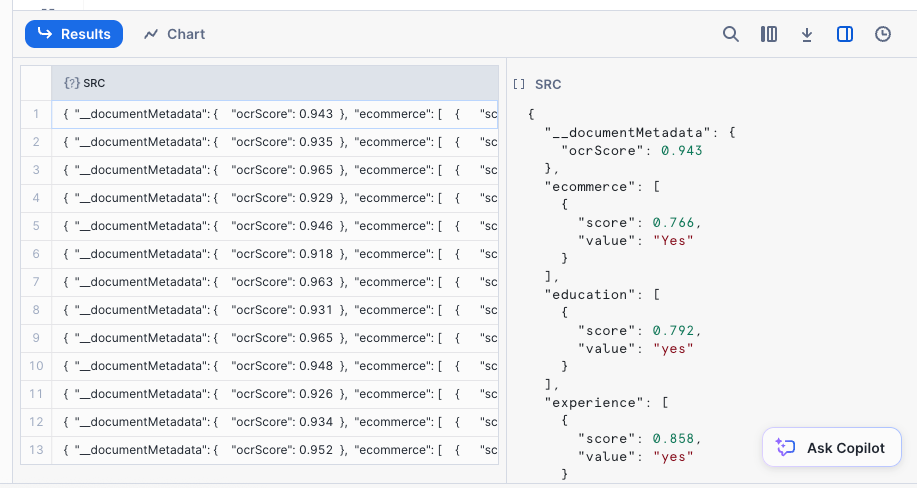
* Capture your json for updating the Lateral Flatten example below

-- View Document AI PREDICT function results

SELECT \* FROM DOCAI\_RESUMES;



Here is an example of the SRC output:



* Create a structured table called “**RESUMES\_ABT**” as a **LATERAL FLATTEN** of the JSON object
* Remove all the unnecessary quotes from the name value
* Use this code to get you started and update as necessary:

CREATE OR REPLACE TABLE RESUMES\_ABT AS

SELECT

REPLACE(name, '"', '') AS name,

REPLACE(location, '"', '') AS location,

REPLACE(email, '"', '') AS email,

REPLACE(experience, '"', '') AS experience,

REPLACE(work, '"', '') AS work,

ARRAY\_AGG(example\_work) AS example\_work,

REPLACE(jobs, '"', '') AS jobs,

REPLACE(what\_education, '"', '') AS what\_education,

ARRAY\_AGG(education\_projects) AS education\_projects,

ARRAY\_AGG(skills) AS skills,

REPLACE(motivation, '"', '') AS motivation,

REPLACE(ui\_ux, '"', '') AS ui\_ux,

REPLACE(front\_end, '"', '') AS front\_end,

REPLACE(responsive, '"', '') AS responsive,

REPLACE(title, '"', '') AS title,

REPLACE(graphic, '"', '') AS graphic,

REPLACE(ecommerce, '"', '') AS ecommerce,

REPLACE(problem\_solving, '"', '') AS problem\_solving

FROM (

SELECT

src:name[0].value AS name,

src:location[0].value AS location,

src:email[0].value AS email,

src:experience[0].value AS experience,

src:work[0].value AS work,

src:example\_work[0].value AS example\_work,

src:jobs[0].value AS jobs,

src:what\_education[0].value AS what\_education,

src:education\_projects[0].value AS education\_projects,

src:skills[0].value AS skills,

src:motivation[0].value AS motivation,

src:ui\_ux[0].value AS ui\_ux,

src:front\_end[0].value AS front\_end,

src:responsive[0].value AS responsive,

src:title[0].value AS title,

src:graphic[0].value AS graphic,

src:ecommerce[0].value AS ecommerce,

src:problem\_solving[0].value AS problem\_solving

FROM DOCAI\_RESUMES,

LATERAL FLATTEN (input => src) AS src

) AS flattened\_data

GROUP BY

name,

location,

email,

experience,

work,

example\_work,

education\_projects,

skills,

jobs,

what\_education,

motivation,

ui\_ux,

front\_end,

responsive,

title,

graphic,

ecommerce,

problem\_solving;

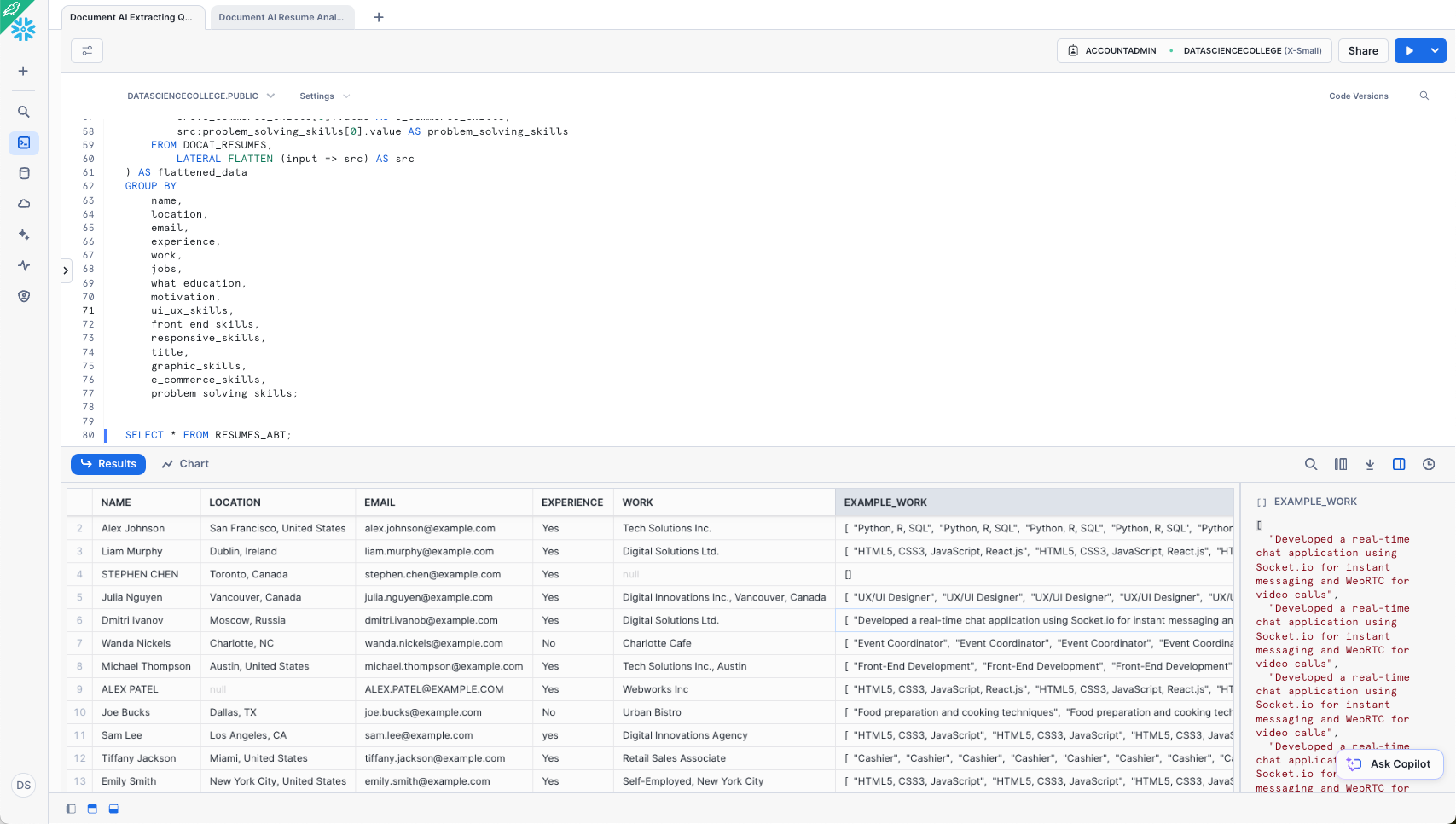


* View the resulting Document AI structured table (**RESUMES\_ABT**)
* Use this code to get you started and update as necessary:

-- View RESUMES\_ABT table

SELECT \* FROM ???;





### Part 5: Optional - Build an extracting query pipeline for new applications

* [Two additional resumes](https://drive.google.com/drive/folders/1VDfotczvzLeeOY7ZYQwSrrT_HSkgjvQ4?usp=sharing) were created for this optional part
* Load these resumes to your RESUMES stage
* Create a task or dynamic table to process new documents in the RESUMES stage
* Here is sample code to update:

-- Create stream on RESUMES stage

CREATE STREAM my\_pdf\_stream ON STAGE RESUMES;

-- Refresh the metadata on the directory table that stores the staged resumes

ALTER STAGE RESUMES REFRESH;

USE DATABASE AICOLLEGE;

USE SCHEMA PUBLIC;

CREATE OR REPLACE TABLE pdf\_reviews (

file\_name VARCHAR,

file\_size VARIANT,

last\_modified VARCHAR,

snowflake\_file\_url VARCHAR,

json\_content VARCHAR

);

CREATE OR REPLACE TASK load\_new\_file\_data

WAREHOUSE = 'aicollege'

SCHEDULE = 'USING CRON 0 0 \* \* \* America/New\_York'

COMMENT = 'Process new files in the stage and insert data into the pdf\_reviews table.'

WHEN SYSTEM$STREAM\_HAS\_DATA('my\_pdf\_stream')

AS

INSERT INTO pdf\_reviews (

SELECT

RELATIVE\_PATH as file\_name,

size AS file\_size,

last\_modified,

file\_url AS snowflake\_file\_url,

inspection\_reviews!PREDICT(GET\_PRESIGNED\_URL('@AICOLLEGE.PUBLIC.RESUMES', RELATIVE\_PATH), 1) AS json\_content

FROM my\_pdf\_stream

WHERE METADATA$ACTION = 'INSERT'

);

ALTER TASK load\_new\_file\_data RESUME;

CREATE OR REPLACE TABLE AICOLLEGE.PUBLIC.PDF\_REVIEWS\_2 AS (

WITH temp AS (

SELECT

RELATIVE\_PATH AS file\_name,

size AS file\_size,

last\_modified,

file\_url AS snowflake\_file\_url,

AICOLLEGE.PUBLIC.DOCUMENT\_AI\_RESUME\_ANALYSIS!PREDICT(get\_presigned\_url('@AICOLLEGE.PUBLIC.RESUMES', RELATIVE\_PATH), 1) AS json\_content

FROM directory(@AICOLLEGE.PUBLIC.RESUMES)

)

SELECT

file\_name,

file\_size,

last\_modified,

snowflake\_file\_url,

json\_content:\_\_documentMetadata.ocrScore::FLOAT AS ocrScore,

f.value:score::FLOAT AS inspection\_date\_score,

f.value:value::STRING AS inspection\_date\_value,

g.value:score::FLOAT AS inspection\_grade\_score,

g.value:value::STRING AS inspection\_grade\_value,

i.value:score::FLOAT AS inspector\_score,

i.value:value::STRING AS inspector\_value,

ARRAY\_TO\_STRING(ARRAY\_AGG(j.value:value::STRING), ', ') AS list\_of\_units

FROM temp,

LATERAL FLATTEN(INPUT => json\_content:inspection\_date) f,

LATERAL FLATTEN(INPUT => json\_content:inspection\_grade) g,

LATERAL FLATTEN(INPUT => json\_content:inspector) i,

LATERAL FLATTEN(INPUT => json\_content:list\_of\_units) j

GROUP BY ALL

);



### Part 6: SiS integration to visualize Document AI Predictions

* Using the **DOC\_AI\_ROLE** role
* Create a SiS app called “**Document AI Resume Review**”
* Replace the automatically created SiS code with the following Python code
* Select **Packages** and add **pypdfium2**
* Select **Run**

# Import python packages

import streamlit as st

from snowflake.snowpark.context import get\_active\_session

import pypdfium2 as pdfium

# Get the current credentials

session = get\_active\_session()

SnowflakeData = "RESUMES\_ABT"

# Set Streamlit page config

st.set\_page\_config(

page\_title="Streamlit App: Explore Document AI Extracted Text",

page\_icon=":truck:",

layout="wide",

)

# Dictionary mapping names to PDF filenames

pdfname = {

"Emily Smith": "Ex1.pdf",

"Alex Johnson": "Ex2.pdf",

"Wanda Nickels": "Ex3.pdf",

"Michael Thompson": "Ex4.pdf",

"Tiffany Jackson": "Ex5.pdf",

"Dmitri Ivanov": "Ex6.pdf",

"STEPHEN CHEN": "Ex7.pdf",

"Alex PATEL": "Ex8.pdf",

"Julia Nguyen": "Ex9.pdf",

"Liam Murphy": "Ex10.pdf",

"Joe Bucks": "Ex11.pdf",

"Emily Smith": "Ex12.pdf",

"SAM LEE": "Ex13.pdf"

}

# Write directly to the app

st.title("TastyBytes Resume Analysis App :balloon:")

st.subheader('Explore overall TastyBytes candidates')

summary1 = session.sql(f"""select count(name) as "# of Candidate" from AICOLLEGE.PUBLIC.RESUMES\_ABT;""").to\_pandas()

st.write("Total Number of Candidates:",summary1)

summary1 = session.sql(f"""select name, location, email, work, what\_education \

from RESUMES\_ABT where Experience = 'Yes';""").to\_pandas()

st.write("Candidates with Web App Design Experience:", summary1)

summary2 = session.sql(f"""select name, location, work, example\_work, what\_education, education\_projects, jobs \

from RESUMES\_ABT \

where Experience = 'Yes' and ui\_ux = 'Yes' and front\_end ='Yes' and graphic = 'Yes'

order by jobs desc ;""").to\_pandas()

st.write("Candidates with Web App Design Experience and Required Skills:", summary2)

# Explore individual candidates by dropdown list

st.subheader('Select TastyByte Candidate to review')

candidate = session.sql(f"""select distinct name from RESUMES\_ABT;""").to\_pandas()

# Candidate selection from dropdown

candidate\_selected = st.selectbox('Which to explore?', list(pdfname.keys()))

#######

# Make sure STATUS column exists

session.sql("ALTER TABLE IF EXISTS RESUMES\_ABT ADD COLUMN IF NOT EXISTS STATUS STRING").collect()

#

#######

# Query the database for selected candidate's details

candidate\_view = session.sql(f"SELECT \* FROM RESUMES\_ABT WHERE NAME = '{candidate\_selected}';").to\_pandas()

st.write("Here are Document AI extracted details for " + candidate\_selected, candidate\_view)

st.write("Revise Candidate Details for " + candidate\_selected)

revise\_details = st.data\_editor(

candidate\_view,

column\_order = (

"STATUS",

"NAME",

"LOCATION",

"EMAIL",

"EXPERIENCE",

"WORK",

"EXAMPLE\_WORK",

"JOBS",

"WHAT\_EDUCATION",

"EDUCATION\_PROJECTS",

"SKILLS",

"MOTIVATION",

"UI\_UX",

"FRONT\_END",

"RESPONSIVE",

"TITLE",

"GRAPHIC",

"ECOMMERCE",

"PROBLEM\_SOLVING"

),

column\_config={

"STATUS": st.column\_config.SelectboxColumn(

"Resume Status",

width="medium",

options=[

"Needs Review",

"Passed resume review - recommend follow up",

"Failed resume review - reject candidate",

],

required=True,

)

},

hide\_index=True,

num\_rows='dynamic'

)

if st.button('Update Snowflake Table'):

with st.spinner("Merging Data"):

try:

# Iterate over the revised details and update the Snowflake table

for index, row in revise\_details.iterrows():

# Construct the SQL UPDATE statement using Question Marks for binding variables

# This limits the chances of SQL Injections

update\_query = f"""

UPDATE RESUMES\_ABT

SET

NAME = ?,

LOCATION = ?,

EMAIL = ?,

EXPERIENCE = ?,

WORK = ?,

EXAMPLE\_WORK = ?,

JOBS = ?,

WHAT\_EDUCATION = ?,

EDUCATION\_PROJECTS = ?,

SKILLS = ?,

MOTIVATION = ?,

UI\_UX = ?,

FRONT\_END = ?,

RESPONSIVE = ?,

TITLE = ?,

GRAPHIC = ?,

ECOMMERCE = ?,

PROBLEM\_SOLVING = ?,

STATUS = ?

WHERE

NAME = ?

"""

session.sql(update\_query, params=[

row['NAME'],

row['LOCATION'],

row['EMAIL'],

row['EXPERIENCE'],

row['WORK'],

row['EXAMPLE\_WORK'],

row['JOBS'],

row['WHAT\_EDUCATION'],

row['EDUCATION\_PROJECTS'],

row['SKILLS'],

row['MOTIVATION'],

row['UI\_UX'],

row['FRONT\_END'],

row['RESPONSIVE'],

row['TITLE'],

row['GRAPHIC'],

row['ECOMMERCE'],

row['PROBLEM\_SOLVING'],

row['STATUS'],

candidate\_selected

]).collect()

st.success("Snowflake table updated successfully!")

except Exception as e:

st.error(f"Error updating the table: {e}")

# Get files from directory table

def get\_files(\_sess):

\_sess.sql("ALTER STAGE AICOLLEGE.PUBLIC.RESUMES REFRESH").collect()

return \_sess.sql(

"""SELECT \* FROM DIRECTORY('@AICOLLEGE.PUBLIC.RESUMES')

WHERE RELATIVE\_PATH ILIKE '%.pdf' ORDER BY RELATIVE\_PATH ASC"""

).to\_pandas()

# Show table of files

df = get\_files(session)

# Retrieve the associated PDF filename

if candidate\_selected in pdfname:

pdf\_filename = pdfname[candidate\_selected]

# Drop down to select a PDF

selected\_pdf = pdf\_filename

# # Drop down to select a PDF

# selected\_pdf = st.selectbox('Choose PDF', df['RELATIVE\_PATH'])

# Get the selected PDF

session.file.get(f"@AICOLLEGE.PUBLIC.RESUMES/{selected\_pdf}", f"/tmp")

# Display the PDF

pdf = pdfium.PdfDocument(f"/tmp/{selected\_pdf}")

# version = pdf.get\_version() # get the PDF standard version

# n\_pages = len(pdf) # get the number of pages in the document

page = pdf[0] # load a page

# Render the page

bitmap = page.render(

scale = 1, # 72dpi resolution

rotation = 0, # no additional rotation

# ... further rendering options

)

pil\_image = bitmap.to\_pil()

st.image(pil\_image)



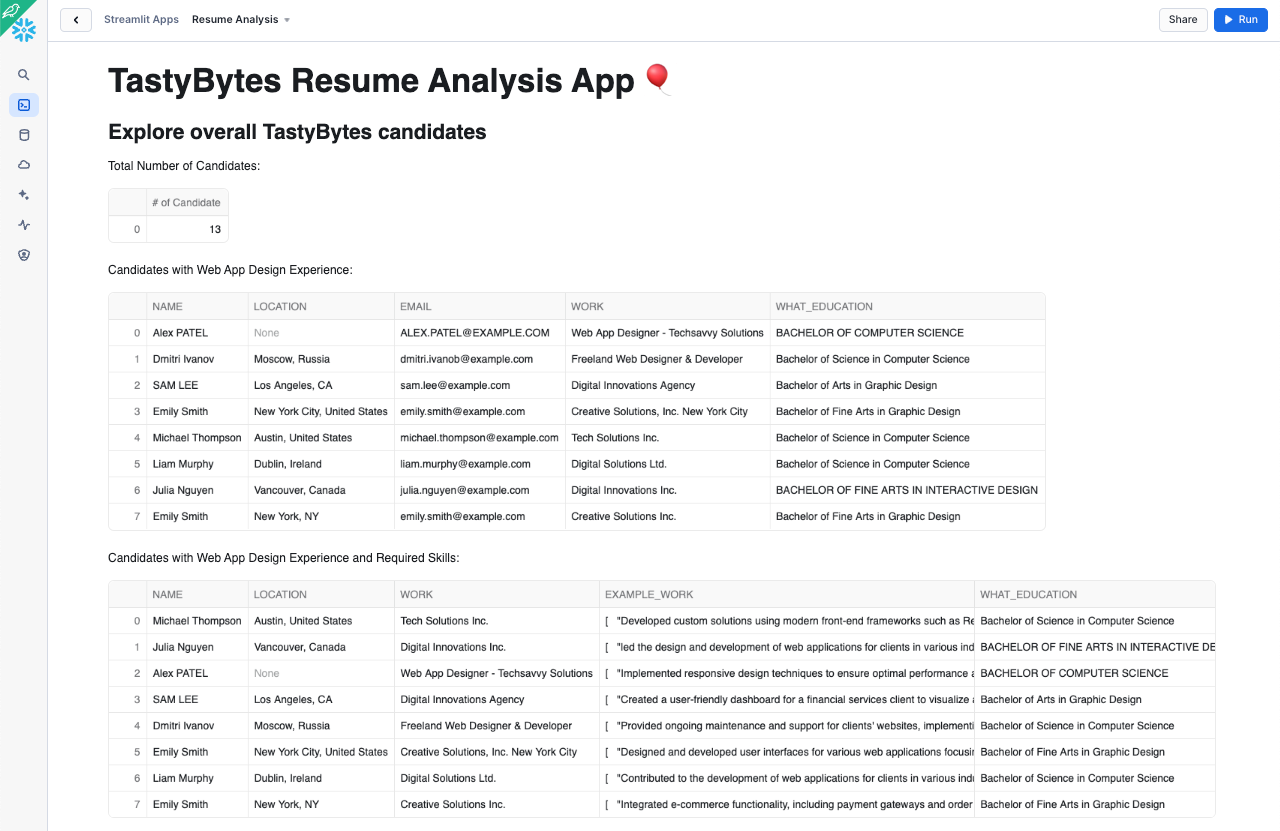
* You may need to run the following if you did not create the RESUMES or RESUMES\_ABT table as the DOC\_AI\_ROLE

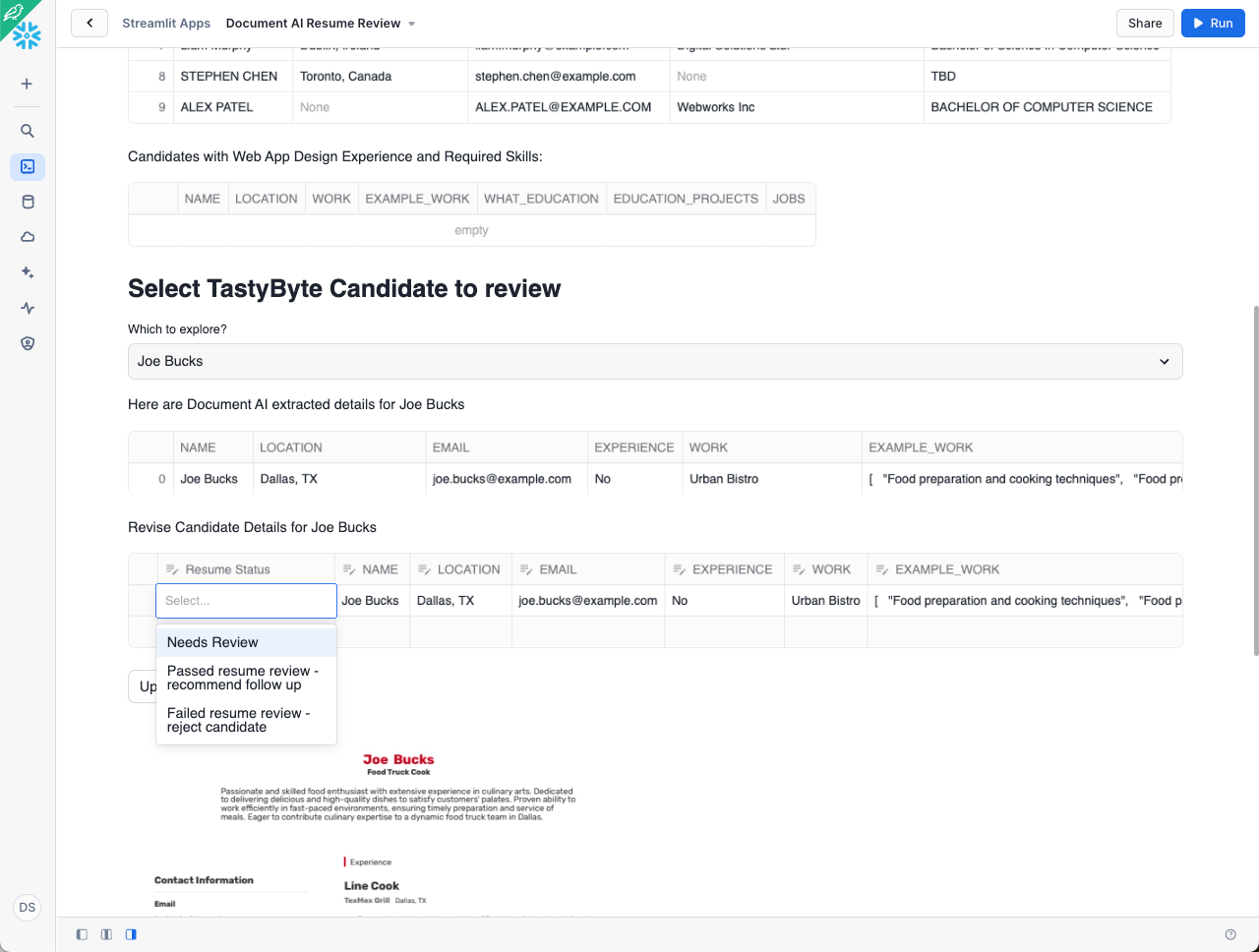
GRANT READ ON STAGE AICOLLEGE.PUBLIC.RESUMES TO ROLE DOC\_AI\_ROLE;

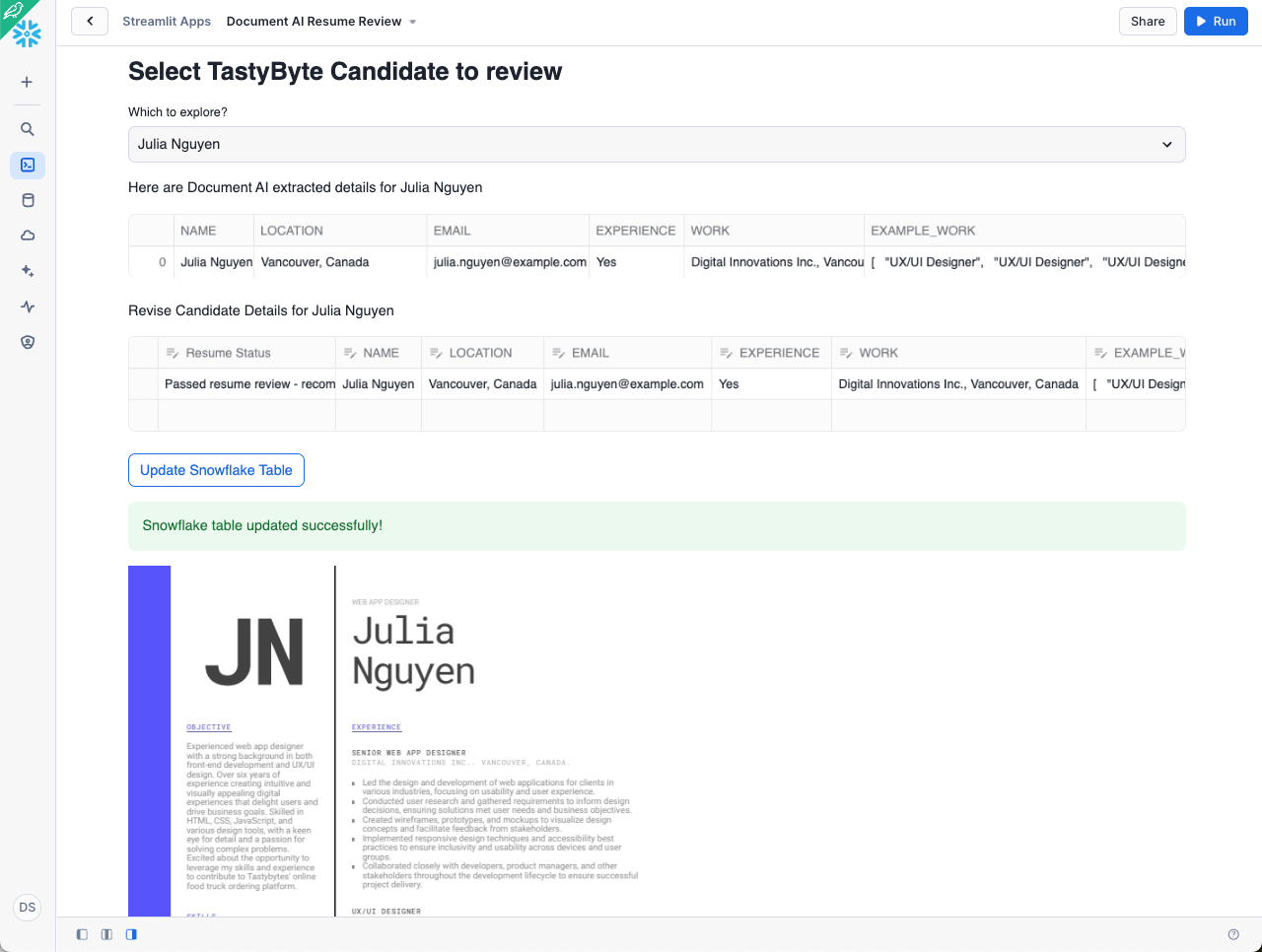
GRANT WRITE ON STAGE AICOLLEGE.PUBLIC.RESUMES TO ROLE DOC\_AI\_ROLE;

GRANT ALL PRIVILEGES ON TABLE DEMO\_DB.PUBLIC.RESUMES\_ABT TO ROLE DOC\_AI\_ROLE;









Quick [sample video](https://drive.google.com/file/d/1WAQcV721s54_y_pLMb7rX7vK0N-62o9R/view?usp=drive_link) for Doc AI Resume Review SiS app and [code](https://drive.google.com/file/d/1SBOqaJ4vX_fBrXTp6Ov4Ps5o5P2MNDJj/view?usp=drive_link)

## College of Data Science HOL DORA Evaluation Test #6

* Add the following SQL command and Run to validate your Document AI HOL is complete

GRANT SELECT ON TABLE DATASCIENCECOLLEGE.PUBLIC.RESUMES\_ABT TO ROLE ACCOUNTADMIN;

USE ROLE ACCOUNTADMIN;

select util\_db.public.se\_grader(step, (actual = expected), actual, expected, description) as graded\_results from

(SELECT 'SEAI19' as step

,( SELECT COUNT(NAME) from RESUMES\_ABT WHERE NAME LIKE 'Michael Thompson') as actual

, 1 as expected

,'HOL DORA Evaluation Test #6 IS CORRECT!' as description);

