**SQL & Python Ingestion and Transformation Script:**

**A screenshot of a computer

AI-generated content may be incorrect.**

* This SQL command created a schema in for my PostgreSQL database called “ai\_job\_risk\_analysis”
  + I did this so that

**A screenshot of a computer

AI-generated content may be incorrect.**

* After establishing my database schema in ADS, I used this SQL code to define and create a table in my “ai\_job\_risk\_analysis” schema for my python-generated data to be inserted into.
  + Also creates the ‘id’ column which acts as the table’s unique identifier
  + The rest of the table’s columns are the same as the columns in my python-generated data and I made sure to define the corresponding data type which should also match that of the python-generated data.

**A screenshot of a computer

AI-generated content may be incorrect.**

* This supplemental SQL code alters the table I just created and adds three new columns to our table: ‘salary\_group’, ‘experience\_level’, and ‘work\_type’
  + These are all calculated columns which are created and defined later in this script. They are derived from already existing columns based on conditions.
  + (I decided to add these columns after I had already produced a finalized output of my database in ADS, or else they would have been included in the initial table creation code)

A white background with black lines

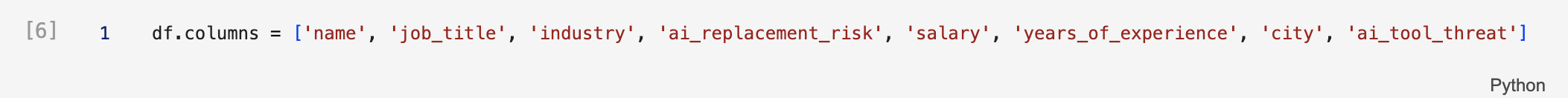
AI-generated content may be incorrect.

* This small snippet of SQL code allowed me to fix a minor issue I was having where all entries in my database with an ai tool threat of “none” were showing up as NULL instead of simply “none”.

**A screenshot of a computer

AI-generated content may be incorrect.**

* This brief code imports the pandas library as ‘pd’ for easy reference and imports the ‘create\_engine’ from the SQLAlchemy library which allows me to connect to SQL databases with python.
* After the imports (df = pd.read\_csv("fake\_ai\_survey.csv")) reads the python-generated CSV output from earlier and loads it into a pandas dataframe.
  + I also printed the dataframe to ensure the data had been loaded correctly

****

* This line of code renames the columns of the dataframe dataset, so the data is clean, consistent, and easier to work with

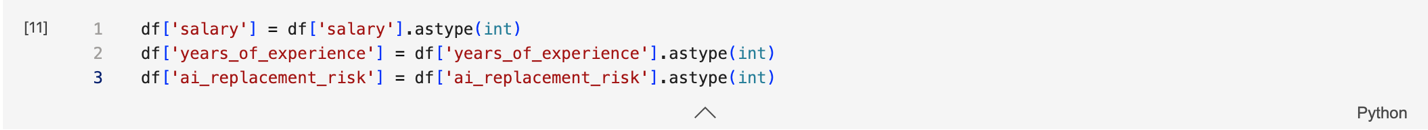
**A screenshot of a computer

AI-generated content may be incorrect.**

* This excerpt of python code is ensuring that columns with numeric values (salary, years\_of\_experience, and ai\_replacement\_risk) contain valid numbers
* And if the values are not valid the df.dropna part of the script drops any rows with missing or invalid data
  + Improves the quality of the dataset

****

* This section of code filters the dataset in the dataframe removing outlier values that could potentially be there (There aren’t any because I generated the random data within a range), but what it does for me in this situation is establish realistic ranges for my simulated data. I’m attempting to simulate a real-world data validation process and reinforce data integrity in my pipeline.

****

* Again, for the sake of ensuring consistency in my dataset, this section of code is converting the data types of the columns I just set ranges for (salary, years\_of\_experience, and ai\_replacement\_risk) to integer data types. This is further ensuring data integrity as the dataset moves through the ETL pipeline and into the PostgreSQL database.
  + Data Type Standardization

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* These three sections of code add three calculated columns: salary\_group, experience\_level, and work\_type to the dataset moving through the pipeline.
  + (I earlier added these to the database table with SQL code)
* This code takes numerical and location data and turns it into meaningful categorical groupings. This will help to provide more insights later and can make the data easier to analyze.
  + The salary\_group column categorizes salaries into four different ranges (Low, Medium, High, Very High), so depending on what range someone’s income lands in that will dictate the salary group they are assigned.
  + The experience\_level column takes numerical data (years of employment or ‘years\_of\_experience’) and similar to the salary\_group, groups different ranges of numbers into experience levels such as Entry, Mid, Senior, and Veteran-level.
  + Lastly, the work\_type column unlike the first two columns, takes location data from the city column (different cities such as Chicago, New York, Austin, etc. and Remote) and infers the work environment based on whether the value is ‘Remote’. If not ‘Remote’ the person will be assigned an ‘In-Office’ value.
* THIS ENHANCES THE INTERPRETABILITY OF THE DATA

**A black and red text

AI-generated content may be incorrect.**

* Although there is not much code, these two lines play a critical role in the pipeline! The line, engine = create\_engine, uses the SQLAlchemy library to establish a connection to my PostgreSQL database hosted on Azure.
  + The connection engine acts as a bridge between my local environment (My Mac) and the cloud-hosted database
* The df.to\_sql line uploads the dataframe (the final, cleaned dataset) to the PostgreSQL database.
  + It inserts the data into the ‘survey\_data’ table in the ‘ai\_job\_risk\_analysis’ schema.
  + If\_exists=’append’ ensures I don’t overwrite data but simply add data to the existing table. (Had to do this because I ended up going back adding the calculated columns and then reloading the data to PostgreSQL).
* LOAD step of the ETL pipeline

**SQL Calculated Field Script** (Ran in Tableau)

IF CONTAINS([Ai Tool Threat], "ChatGPT") THEN "Generative AI"

ELSEIF CONTAINS ([Ai Tool Threat], "Claude") THEN "Generative AI"

ELSEIF CONTAINS ([Ai Tool Threat], "Google Gemini") THEN "Generative AI"

ELSEIF CONTAINS ([Ai Tool Threat], "Github Copilot") THEN "Code Assistant"

ELSEIF CONTAINS ([Ai Tool Threat], "Harvey") THEN "Industry-Specific AI"

ELSEIF CONTAINS ([Ai Tool Threat], "DoNotPay") THEN "Industry-Specific AI"

ELSEIF CONTAINS ([Ai Tool Threat], "MedpaLM") THEN "Industry-Specific AI"

ELSEIF CONTAINS ([Ai Tool Threat], "Perplexity AI") THEN "Research Assistant"

ELSEIF CONTAINS ([Ai Tool Threat], "Writer.com") THEN "Productivity AI"

ELSEIF CONTAINS ([Ai Tool Threat], "Scribe AI") THEN "Productivity AI"

ELSE "Other"

END

* This SQL script was run to create a calculated field in Tableau. The calculated field (AI Tool Category) groups different AI tools based on their use case. Done to provide further insight and ease interpretability.