Test Quarto Julio Vargas

Julio Vargas Garcia

September 17, 2025

# 1. import data

Importing Dataset using Pyspark. The code shows the schema and first rows of the dataset. Schema is shown below but partial dataframe is muted.

from pyspark.sql import SparkSession  
from pyspark.sql.types import StructType  
import json  
  
spark = SparkSession.builder.appName("JobPostingsAnalysis").getOrCreate()  
  
# Load schema from JSON file  
with open("data/schema\_lightcast.json") as f:  
 schema = StructType.fromJson(json.load(f))  
  
df = (spark.read  
 .option("header", "true")  
 .option("inferSchema","false")   
 .schema(schema) # saved schema  
 .option("multiLine", "true")  
 .option("escape", "\"")  
 .csv("data/lightcast\_job\_postings.csv")  
 .limit(5000))  
  
df.createOrReplaceTempView("jobs")  
  
  
# save schema to pretty JSON  
import json  
# convert schema to dict first  
schema\_dict = df.schema.jsonValue()  
# write it nicely formatted (indent=2 gives pretty print)  
with open("data/schema\_lightcast.json", "w") as f:  
 json.dump(schema\_dict, f, indent=2)  
  
  
df.printSchema()  
  
  
# Count total rows in the DataFrame  
row\_count = df.count()  
print(row\_count)

root  
 |-- ID: string (nullable = true)  
 |-- LAST\_UPDATED\_DATE: string (nullable = true)  
 |-- LAST\_UPDATED\_TIMESTAMP: timestamp (nullable = true)  
 |-- DUPLICATES: integer (nullable = true)  
 |-- POSTED: string (nullable = true)  
 |-- EXPIRED: string (nullable = true)  
 |-- DURATION: integer (nullable = true)  
 |-- SOURCE\_TYPES: string (nullable = true)  
 |-- SOURCES: string (nullable = true)  
 |-- URL: string (nullable = true)  
 |-- ACTIVE\_URLS: string (nullable = true)  
 |-- ACTIVE\_SOURCES\_INFO: string (nullable = true)  
 |-- TITLE\_RAW: string (nullable = true)  
 |-- BODY: string (nullable = true)  
 |-- MODELED\_EXPIRED: string (nullable = true)  
 |-- MODELED\_DURATION: integer (nullable = true)  
 |-- COMPANY: integer (nullable = true)  
 |-- COMPANY\_NAME: string (nullable = true)  
 |-- COMPANY\_RAW: string (nullable = true)  
 |-- COMPANY\_IS\_STAFFING: boolean (nullable = true)  
 |-- EDUCATION\_LEVELS: string (nullable = true)  
 |-- EDUCATION\_LEVELS\_NAME: string (nullable = true)  
 |-- MIN\_EDULEVELS: integer (nullable = true)  
 |-- MIN\_EDULEVELS\_NAME: string (nullable = true)  
 |-- MAX\_EDULEVELS: integer (nullable = true)  
 |-- MAX\_EDULEVELS\_NAME: string (nullable = true)  
 |-- EMPLOYMENT\_TYPE: integer (nullable = true)  
 |-- EMPLOYMENT\_TYPE\_NAME: string (nullable = true)  
 |-- MIN\_YEARS\_EXPERIENCE: integer (nullable = true)  
 |-- MAX\_YEARS\_EXPERIENCE: integer (nullable = true)  
 |-- IS\_INTERNSHIP: boolean (nullable = true)  
 |-- SALARY: integer (nullable = true)  
 |-- REMOTE\_TYPE: integer (nullable = true)  
 |-- REMOTE\_TYPE\_NAME: string (nullable = true)  
 |-- ORIGINAL\_PAY\_PERIOD: string (nullable = true)  
 |-- SALARY\_TO: integer (nullable = true)  
 |-- SALARY\_FROM: integer (nullable = true)  
 |-- LOCATION: string (nullable = true)  
 |-- CITY: string (nullable = true)  
 |-- CITY\_NAME: string (nullable = true)  
 |-- COUNTY: integer (nullable = true)  
 |-- COUNTY\_NAME: string (nullable = true)  
 |-- MSA: integer (nullable = true)  
 |-- MSA\_NAME: string (nullable = true)  
 |-- STATE: integer (nullable = true)  
 |-- STATE\_NAME: string (nullable = true)  
 |-- COUNTY\_OUTGOING: integer (nullable = true)  
 |-- COUNTY\_NAME\_OUTGOING: string (nullable = true)  
 |-- COUNTY\_INCOMING: integer (nullable = true)  
 |-- COUNTY\_NAME\_INCOMING: string (nullable = true)  
 |-- MSA\_OUTGOING: integer (nullable = true)  
 |-- MSA\_NAME\_OUTGOING: string (nullable = true)  
 |-- MSA\_INCOMING: integer (nullable = true)  
 |-- MSA\_NAME\_INCOMING: string (nullable = true)  
 |-- NAICS2: integer (nullable = true)  
 |-- NAICS2\_NAME: string (nullable = true)  
 |-- NAICS3: integer (nullable = true)  
 |-- NAICS3\_NAME: string (nullable = true)  
 |-- NAICS4: integer (nullable = true)  
 |-- NAICS4\_NAME: string (nullable = true)  
 |-- NAICS5: integer (nullable = true)  
 |-- NAICS5\_NAME: string (nullable = true)  
 |-- NAICS6: integer (nullable = true)  
 |-- NAICS6\_NAME: string (nullable = true)  
 |-- TITLE: string (nullable = true)  
 |-- TITLE\_NAME: string (nullable = true)  
 |-- TITLE\_CLEAN: string (nullable = true)  
 |-- SKILLS: string (nullable = true)  
 |-- SKILLS\_NAME: string (nullable = true)  
 |-- SPECIALIZED\_SKILLS: string (nullable = true)  
 |-- SPECIALIZED\_SKILLS\_NAME: string (nullable = true)  
 |-- CERTIFICATIONS: string (nullable = true)  
 |-- CERTIFICATIONS\_NAME: string (nullable = true)  
 |-- COMMON\_SKILLS: string (nullable = true)  
 |-- COMMON\_SKILLS\_NAME: string (nullable = true)  
 |-- SOFTWARE\_SKILLS: string (nullable = true)  
 |-- SOFTWARE\_SKILLS\_NAME: string (nullable = true)  
 |-- ONET: string (nullable = true)  
 |-- ONET\_NAME: string (nullable = true)  
 |-- ONET\_2019: string (nullable = true)  
 |-- ONET\_2019\_NAME: string (nullable = true)  
 |-- CIP6: string (nullable = true)  
 |-- CIP6\_NAME: string (nullable = true)  
 |-- CIP4: string (nullable = true)  
 |-- CIP4\_NAME: string (nullable = true)  
 |-- CIP2: string (nullable = true)  
 |-- CIP2\_NAME: string (nullable = true)  
 |-- SOC\_2021\_2: string (nullable = true)  
 |-- SOC\_2021\_2\_NAME: string (nullable = true)  
 |-- SOC\_2021\_3: string (nullable = true)  
 |-- SOC\_2021\_3\_NAME: string (nullable = true)  
 |-- SOC\_2021\_4: string (nullable = true)  
 |-- SOC\_2021\_4\_NAME: string (nullable = true)  
 |-- SOC\_2021\_5: string (nullable = true)  
 |-- SOC\_2021\_5\_NAME: string (nullable = true)  
 |-- LOT\_CAREER\_AREA: integer (nullable = true)  
 |-- LOT\_CAREER\_AREA\_NAME: string (nullable = true)  
 |-- LOT\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_SPECIALIZED\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_SPECIALIZED\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_OCCUPATION\_GROUP: integer (nullable = true)  
 |-- LOT\_OCCUPATION\_GROUP\_NAME: string (nullable = true)  
 |-- LOT\_V6\_SPECIALIZED\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_V6\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_V6\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_V6\_OCCUPATION\_GROUP: integer (nullable = true)  
 |-- LOT\_V6\_OCCUPATION\_GROUP\_NAME: string (nullable = true)  
 |-- LOT\_V6\_CAREER\_AREA: integer (nullable = true)  
 |-- LOT\_V6\_CAREER\_AREA\_NAME: string (nullable = true)  
 |-- SOC\_2: string (nullable = true)  
 |-- SOC\_2\_NAME: string (nullable = true)  
 |-- SOC\_3: string (nullable = true)  
 |-- SOC\_3\_NAME: string (nullable = true)  
 |-- SOC\_4: string (nullable = true)  
 |-- SOC\_4\_NAME: string (nullable = true)  
 |-- SOC\_5: string (nullable = true)  
 |-- SOC\_5\_NAME: string (nullable = true)  
 |-- LIGHTCAST\_SECTORS: string (nullable = true)  
 |-- LIGHTCAST\_SECTORS\_NAME: string (nullable = true)  
 |-- NAICS\_2022\_2: integer (nullable = true)  
 |-- NAICS\_2022\_2\_NAME: string (nullable = true)  
 |-- NAICS\_2022\_3: integer (nullable = true)  
 |-- NAICS\_2022\_3\_NAME: string (nullable = true)  
 |-- NAICS\_2022\_4: integer (nullable = true)  
 |-- NAICS\_2022\_4\_NAME: string (nullable = true)  
 |-- NAICS\_2022\_5: integer (nullable = true)  
 |-- NAICS\_2022\_5\_NAME: string (nullable = true)  
 |-- NAICS\_2022\_6: integer (nullable = true)  
 |-- NAICS\_2022\_6\_NAME: string (nullable = true)  
  
5000

# 2. Step 2 - Creating Relational Tables

We will split the main dataframe into four relational tables based on the dataset:

* Job Postings
* Industries (based on NAICS 2022 and SOC 5)
* Companies
* Location (including MSA - Metropolitan Statistical Area)

Each table should have a primary key and the necessary foreign keys to maintain relationships. Here’s a table outlining the four relational tables, along with their relevant columns:

## 2.1 Example

Let’s get 5 rows of id, title, and company\_name from the job postings table.

spark.sql("SELECT ID AS job\_id, title\_raw FROM jobs LIMIT 5").show(truncate=False)

+----------------------------------------+------------------------------------------------------------+  
|job\_id |title\_raw |  
+----------------------------------------+------------------------------------------------------------+  
|1f57d95acf4dc67ed2819eb12f049f6a5c11782c|Enterprise Analyst (II-III) |  
|0cb072af26757b6c4ea9464472a50a443af681ac|Oracle Consultant - Reports (3592) |  
|85318b12b3331fa490d32ad014379df01855c557|Data Analyst |  
|1b5c3941e54a1889ef4f8ae55b401a550708a310|Sr. Lead Data Mgmt. Analyst - SAS Product Owner |  
|cb5ca25f02bdf25c13edfede7931508bfd9e858f|Comisiones de $1000 - $3000 por semana... Comiensa Rapido!!!|  
+----------------------------------------+------------------------------------------------------------+

## 2.2 Locations Table

Lets extract columns from the main dataframe to create a locations table. The columns are as follows:

I also sorted the locations table by MSA in ascending order and then added a location\_id as primary key.

LOCATION\_ID (PK), LOCATION, CITY\_NAME, STATE\_NAME, COUNTY\_NAME, MSA, MSA\_NAME.

from pyspark.sql.functions import col, monotonically\_increasing\_id  
  
# using inbuilt pyspark select  
# locations\_df = df.select(  
# col("location"),  
# col("city\_name"),  
# col("state\_name"),  
# col("county\_name"),  
# col("msa"),  
# col("msa\_name")  
# ).distinct().withColumn("location\_id", monotonically\_increasing\_id())  
  
#alternative using selectExpr  
locations\_df = df.selectExpr("monotonically\_increasing\_id() AS LOCATION\_ID",  
 "location",  
 "city\_name",  
 "state\_name",  
 "county\_name",  
 "msa",  
 "msa\_name")  
  
locations\_df.createOrReplaceTempView("locations")  
  
locations\_df.show(truncate=False)

+-----------+-------------------------------------------------+------------------+-------------+--------------------+-----+-------------------------------------+  
|LOCATION\_ID|location |city\_name |state\_name |county\_name |msa |msa\_name |  
+-----------+-------------------------------------------------+------------------+-------------+--------------------+-----+-------------------------------------+  
|0 |{\n "lat": 33.20763,\n "lon": -92.6662674\n} |El Dorado, AR |Arkansas |Union, AR |20980|El Dorado, AR |  
|1 |{\n "lat": 44.3106241,\n "lon": -69.7794897\n} |Augusta, ME |Maine |Kennebec, ME |12300|Augusta-Waterville, ME |  
|2 |{\n "lat": 32.7766642,\n "lon": -96.7969879\n} |Dallas, TX |Texas |Dallas, TX |19100|Dallas-Fort Worth-Arlington, TX |  
|3 |{\n "lat": 33.4483771,\n "lon": -112.0740373\n}|Phoenix, AZ |Arizona |Maricopa, AZ |38060|Phoenix-Mesa-Chandler, AZ |  
|4 |{\n "lat": 37.6392595,\n "lon": -120.9970014\n}|Modesto, CA |California |Stanislaus, CA |33700|Modesto, CA |  
|5 |{\n "lat": 0,\n "lon": 0\n} |[Unknown City], AR|Arkansas |[Unknown county], AR|NULL |NULL |  
|6 |{\n "lat": 33.4941704,\n "lon": -111.9260519\n}|Scottsdale, AZ |Arizona |Maricopa, AZ |38060|Phoenix-Mesa-Chandler, AZ |  
|7 |{\n "lat": 39.7589478,\n "lon": -84.1916069\n} |Dayton, OH |Ohio |Montgomery, OH |19430|Dayton-Kettering, OH |  
|8 |{\n "lat": 41.1220409,\n "lon": -74.5804378\n} |Franklin, NJ |New Jersey |Sussex, NJ |35620|New York-Newark-Jersey City, NY-NJ-PA|  
|9 |{\n "lat": 40.7501,\n "lon": -73.997\n} |New York, NY |New York |New York, NY |35620|New York-Newark-Jersey City, NY-NJ-PA|  
|10 |{\n "lat": 35.6224561,\n "lon": -117.6708966\n}|Ridgecrest, CA |California |Kern, CA |12540|Bakersfield, CA |  
|11 |{\n "lat": 21.3069444,\n "lon": -157.8583333\n}|Honolulu, HI |Hawaii |Honolulu, HI |46520|Urban Honolulu, HI |  
|12 |{\n "lat": 0,\n "lon": 0\n} |[Unknown City], GA|Georgia |[Unknown county], GA|NULL |NULL |  
|13 |{\n "lat": 42.331427,\n "lon": -83.0457538\n} |Detroit, MI |Michigan |Wayne, MI |19820|Detroit-Warren-Dearborn, MI |  
|14 |{\n "lat": 32.2987573,\n "lon": -90.1848103\n} |Jackson, MS |Mississippi |Hinds, MS |27140|Jackson, MS |  
|15 |{\n "lat": 42.3600825,\n "lon": -71.0588801\n} |Boston, MA |Massachusetts|Suffolk, MA |14460|Boston-Cambridge-Newton, MA-NH |  
|16 |{\n "lat": 0,\n "lon": 0\n} |[Unknown City], AZ|Arizona |[Unknown county], AZ|NULL |NULL |  
|17 |{\n "lat": 58.3019444,\n "lon": -134.4197221\n}|Juneau, AK |Alaska |Juneau Borough, AK |27940|Juneau, AK |  
|18 |{\n "lat": 33.5185892,\n "lon": -86.8103567\n} |Birmingham, AL |Alabama |Jefferson, AL |13820|Birmingham-Hoover, AL |  
|19 |{\n "lat": 37.7749295,\n "lon": -122.4194155\n}|San Francisco, CA |California |San Francisco, CA |41860|San Francisco-Oakland-Berkeley, CA |  
+-----------+-------------------------------------------------+------------------+-------------+--------------------+-----+-------------------------------------+  
only showing top 20 rows

## 2.3 Industries Table

industries INDUSTRY\_ID (Primary Key), NAICS\_2022\_6, NAICS\_2022\_6\_NAME, SOC\_5, SOC\_5\_NAME, LOT\_SPECIALIZED\_OCCUPATION\_NAME, LOT\_OCCUPATION\_GROUP

industries\_df = df.selectExpr("monotonically\_increasing\_id() AS INDUSTRY\_ID",  
 "naics\_2022\_4",  
 "naics\_2022\_4\_name",  
 "naics\_2022\_5",  
 "naics\_2022\_5\_name",  
 "naics\_2022\_6",  
 "naics\_2022\_6\_name",  
 "soc\_5 AS SOC\_5",  
 "soc\_5\_name AS SOC\_5\_NAME",  
 "lot\_specialized\_occupation\_name",  
 "lot\_occupation\_group")  
  
industries\_df.createOrReplaceTempView("industries")  
  
industries\_df.show(truncate=False)

+-----------+------------+------------------------------------------------------------+------------+------------------------------------------------------------------------+------------+------------------------------------------------------------------------+-------+---------------+--------------------------------+--------------------+  
|INDUSTRY\_ID|naics\_2022\_4|naics\_2022\_4\_name |naics\_2022\_5|naics\_2022\_5\_name |naics\_2022\_6|naics\_2022\_6\_name |SOC\_5 |SOC\_5\_NAME |lot\_specialized\_occupation\_name |lot\_occupation\_group|  
+-----------+------------+------------------------------------------------------------+------------+------------------------------------------------------------------------+------------+------------------------------------------------------------------------+-------+---------------+--------------------------------+--------------------+  
|0 |4413 |Automotive Parts, Accessories, and Tire Retailers |44133 |Automotive Parts and Accessories Retailers |441330 |Automotive Parts and Accessories Retailers |15-2051|Data Scientists|General ERP Analyst / Consultant|2310 |  
|1 |5613 |Employment Services |56132 |Temporary Help Services |561320 |Temporary Help Services |15-2051|Data Scientists|Oracle Consultant / Analyst |2310 |  
|2 |5242 |Agencies, Brokerages, and Other Insurance Related Activities|52429 |Other Insurance Related Activities |524291 |Claims Adjusting |15-2051|Data Scientists|Data Analyst |2311 |  
|3 |5221 |Depository Credit Intermediation |52211 |Commercial Banking |522110 |Commercial Banking |15-2051|Data Scientists|Data Analyst |2311 |  
|4 |9999 |Unclassified Industry |99999 |Unclassified Industry |999999 |Unclassified Industry |15-2051|Data Scientists|Oracle Consultant / Analyst |2310 |  
|5 |5178 |All Other Telecommunications |51781 |All Other Telecommunications |517810 |All Other Telecommunications |15-2051|Data Scientists|Data Analyst |2311 |  
|6 |3344 |Semiconductor and Other Electronic Component Manufacturing |33441 |Semiconductor and Other Electronic Component Manufacturing |334413 |Semiconductor and Related Device Manufacturing |15-2051|Data Scientists|Data Analyst |2311 |  
|7 |5242 |Agencies, Brokerages, and Other Insurance Related Activities|52429 |Other Insurance Related Activities |524291 |Claims Adjusting |15-2051|Data Scientists|Data Analyst |2311 |  
|8 |9999 |Unclassified Industry |99999 |Unclassified Industry |999999 |Unclassified Industry |15-2051|Data Scientists|General ERP Analyst / Consultant|2310 |  
|9 |5415 |Computer Systems Design and Related Services |54151 |Computer Systems Design and Related Services |541511 |Custom Computer Programming Services |15-2051|Data Scientists|Data Analyst |2311 |  
|10 |4238 |Machinery, Equipment, and Supplies Merchant Wholesalers |42383 |Industrial Machinery and Equipment Merchant Wholesalers |423830 |Industrial Machinery and Equipment Merchant Wholesalers |15-2051|Data Scientists|Data Analyst |2311 |  
|11 |5613 |Employment Services |56132 |Temporary Help Services |561320 |Temporary Help Services |15-2051|Data Scientists|Data Analyst |2311 |  
|12 |5223 |Activities Related to Credit Intermediation |52232 |Financial Transactions Processing, Reserve, and Clearinghouse Activities|522320 |Financial Transactions Processing, Reserve, and Clearinghouse Activities|15-2051|Data Scientists|Data Analyst |2311 |  
|13 |5416 |Management, Scientific, and Technical Consulting Services |54161 |Management Consulting Services |541611 |Administrative Management and General Management Consulting Services |15-2051|Data Scientists|General ERP Analyst / Consultant|2310 |  
|14 |5239 |Other Financial Investment Activities |52394 |Portfolio Management and Investment Advice |523940 |Portfolio Management and Investment Advice |15-2051|Data Scientists|Enterprise Architect |2315 |  
|15 |6113 |Colleges, Universities, and Professional Schools |61131 |Colleges, Universities, and Professional Schools |611310 |Colleges, Universities, and Professional Schools |15-2051|Data Scientists|Data Analyst |2311 |  
|16 |5415 |Computer Systems Design and Related Services |54151 |Computer Systems Design and Related Services |541512 |Computer Systems Design Services |15-2051|Data Scientists|General ERP Analyst / Consultant|2310 |  
|17 |5613 |Employment Services |56132 |Temporary Help Services |561320 |Temporary Help Services |15-2051|Data Scientists|Oracle Consultant / Analyst |2310 |  
|18 |6214 |Outpatient Care Centers |62149 |Other Outpatient Care Centers |621493 |Freestanding Ambulatory Surgical and Emergency Centers |15-2051|Data Scientists|Enterprise Architect |2315 |  
|19 |9999 |Unclassified Industry |99999 |Unclassified Industry |999999 |Unclassified Industry |15-2051|Data Scientists|Data Analyst |2311 |  
+-----------+------------+------------------------------------------------------------+------------+------------------------------------------------------------------------+------------+------------------------------------------------------------------------+-------+---------------+--------------------------------+--------------------+  
only showing top 20 rows

## 2.4 Companies Table

#   
# Step 1: Create Companies Table (Primary Key: company\_id)  
companies\_df = df.select(  
 col("company"),  
 col("company\_name"),  
 col("company\_raw"),  
 col("company\_is\_staffing")  
).distinct().withColumn("company\_id", monotonically\_increasing\_id())  
# companies\_df.show(5)  
companies = companies\_df.toPandas()  
companies.drop(columns=["company"], inplace=True)  
companies.rename(columns={"company\_is\_staffing": "is\_staffing"},  
inplace=True)  
companies.to\_csv("./output/companies.csv", index=False)  
companies.head()

|  | company\_name | company\_raw | is\_staffing | company\_id |
| --- | --- | --- | --- | --- |
| 0 | Murphy USA | Murphy USA | False | 0 |
| 1 | Smx Corporation Limited | SMX | True | 1 |
| 2 | Sedgwick | Sedgwick | False | 2 |
| 3 | Wells Fargo | Wells Fargo | False | 3 |
| 4 | Unclassified | LH/GM | False | 4 |

## 2.5 Job postings Table

# Step 4: Create Job Postings Table (Adding Foreign Keys)  
job\_postings\_df = df.select(  
 col("id").alias("job\_id"),  
 col("title\_clean"),  
 col("body"),  
 col("company"),  
 col("employment\_type\_name"),  
 col("remote\_type\_name"),  
 col("min\_years\_experience"),  
 col("max\_years\_experience"),  
 col("salary"),  
 col("salary\_from"),  
 col("salary\_to"),  
 col("location"),  
 col("naics\_2022\_6"),  
 col("posted"),  
 col("expired"),  
 col("duration")  
)  
  
job\_postings\_df.show(5)

+--------------------+--------------------+--------------------+--------+--------------------+----------------+--------------------+--------------------+------+-----------+---------+--------------------+------------+--------+---------+--------+  
| job\_id| title\_clean| body| company|employment\_type\_name|remote\_type\_name|min\_years\_experience|max\_years\_experience|salary|salary\_from|salary\_to| location|naics\_2022\_6| posted| expired|duration|  
+--------------------+--------------------+--------------------+--------+--------------------+----------------+--------------------+--------------------+------+-----------+---------+--------------------+------------+--------+---------+--------+  
|1f57d95acf4dc67ed...|enterprise analys...|31-May-2024\n\nEn...| 894731|Full-time (> 32 h...| [None]| 2| 2| NULL| NULL| NULL|{\n "lat": 33.20...| 441330|6/2/2024| 6/8/2024| 6|  
|0cb072af26757b6c4...|oracle consultant...|Oracle Consultant...| 133098|Full-time (> 32 h...| Remote| 3| 3| NULL| NULL| NULL|{\n "lat": 44.31...| 561320|6/2/2024| 8/1/2024| NULL|  
|85318b12b3331fa49...| data analyst|Taking care of pe...|39063746|Full-time (> 32 h...| [None]| 5| NULL| NULL| NULL| NULL|{\n "lat": 32.77...| 524291|6/2/2024| 7/7/2024| 35|  
|1b5c3941e54a1889e...|sr lead data mgmt...|About this role:\...|37615159|Full-time (> 32 h...| [None]| 3| NULL| NULL| NULL| NULL|{\n "lat": 33.44...| 522110|6/2/2024|7/20/2024| 48|  
|cb5ca25f02bdf25c1...|comisiones de por...|Comisiones de $10...| 0|Part-time / full-...| [None]| NULL| NULL| 92500| 35000| 150000|{\n "lat": 37.63...| 999999|6/2/2024|6/17/2024| 15|  
+--------------------+--------------------+--------------------+--------+--------------------+----------------+--------------------+--------------------+------+-----------+---------+--------------------+------------+--------+---------+--------+  
only showing top 5 rows

Adding Foreign keys to job postings table

# Join with Companies Table to get company\_id  
job\_postings\_df = job\_postings\_df.join(companies\_df.select("company", "company\_id"), on="company", how="left")  
  
# Join with Locations Table to get location\_id  
job\_postings\_df = job\_postings\_df.join(locations\_df.select("location", "location\_id"), job\_postings\_df.location == locations\_df["location"],  
how="left").drop("location")  
  
# Join with Industries Table to get industry\_id  
job\_postings\_df = job\_postings\_df.join(industries\_df.select("naics\_2022\_6", "industry\_id"),  
 job\_postings\_df.naics\_2022\_6 == industries\_df.naics\_2022\_6,  
 how="left").drop("naics\_2022\_6")  
  
# Drop redundant columns  
job\_postings\_df = job\_postings\_df.drop("company", "lat-long")  
job\_postings\_df.createOrReplaceTempView("job\_postings")  
  
# Show final job\_postings\_df structure  
job\_postings\_df.show(5)

25/09/21 04:38:11 WARN Column: Constructing trivially true equals predicate, 'location == location'. Perhaps you need to use aliases.  
25/09/21 04:38:11 WARN Column: Constructing trivially true equals predicate, 'naics\_2022\_6 == naics\_2022\_6'. Perhaps you need to use aliases.  
[Stage 45:> (0 + 1) / 1][Stage 46:> (0 + 1) / 1][Stage 47:> (0 + 0) / 1][Stage 46:> (0 + 1) / 1][Stage 47:> (0 + 1) / 1]

+--------------------+--------------------+--------------------+--------------------+----------------+--------------------+--------------------+------+-----------+---------+--------+--------+--------+----------+-----------+-----------+  
| job\_id| title\_clean| body|employment\_type\_name|remote\_type\_name|min\_years\_experience|max\_years\_experience|salary|salary\_from|salary\_to| posted| expired|duration|company\_id|location\_id|industry\_id|  
+--------------------+--------------------+--------------------+--------------------+----------------+--------------------+--------------------+------+-----------+---------+--------+--------+--------+----------+-----------+-----------+  
|1f57d95acf4dc67ed...|enterprise analys...|31-May-2024\n\nEn...|Full-time (> 32 h...| [None]| 2| 2| NULL| NULL| NULL|6/2/2024|6/8/2024| 6| 0| 1399| 4561|  
|1f57d95acf4dc67ed...|enterprise analys...|31-May-2024\n\nEn...|Full-time (> 32 h...| [None]| 2| 2| NULL| NULL| NULL|6/2/2024|6/8/2024| 6| 0| 1399| 2770|  
|1f57d95acf4dc67ed...|enterprise analys...|31-May-2024\n\nEn...|Full-time (> 32 h...| [None]| 2| 2| NULL| NULL| NULL|6/2/2024|6/8/2024| 6| 0| 1399| 1399|  
|1f57d95acf4dc67ed...|enterprise analys...|31-May-2024\n\nEn...|Full-time (> 32 h...| [None]| 2| 2| NULL| NULL| NULL|6/2/2024|6/8/2024| 6| 0| 1399| 0|  
|1f57d95acf4dc67ed...|enterprise analys...|31-May-2024\n\nEn...|Full-time (> 32 h...| [None]| 2| 2| NULL| NULL| NULL|6/2/2024|6/8/2024| 6| 0| 0| 4561|  
+--------------------+--------------------+--------------------+--------------------+----------------+--------------------+--------------------+------+-----------+---------+--------+--------+--------+----------+-----------+-----------+  
only showing top 5 rows

# 3. Query 1: Industry-Specific Salary Trends Grouped by Job Title

Identify **median salary trends** for job postings in the **Technology industry (NAICS\_2022\_5 = '51821')**, grouped by **specialized occupation**.

* **Filter the dataset**
  + Select job postings where **naics\_2022\_5 = '51821'** (Technology industry).
  + Ensure salary values are **not NULL and greater than 0**.
* **Join the relevant tables**
  + Use **job\_postings** as the base table.
  + Join with **industries** using **industry\_id**.
* **Aggregate data**
  + Group results by **industry name (naics\_2022\_5\_name)** and **specialized occupation (specialized\_occupation)**.
  + Compute the **median salary** using **PERCENTILE\_APPROX()** for specialized occupation.
* **Order the results**
  + Sort by **median salary** in descending order.
* **Visualize results** using **Plotly**
  + Create a **grouped bar chart** where:
    - **X-axis** = lot\_specialized\_occupation\_name
    - **Y-axis** = median\_salary
    - **Color** = industry\_name
* Ensure different specialized occupations are distinguishable across the industry.

|  |
| --- |
| Query compute median salary by specialized occupation in Tech in progress |
| from pyspark.sql import functions as F  # Filter for Tech industry tech = (  df.where(  (F.col("NAICS\_2022\_5") == "51821") &  F.col("SALARY").isNotNull() &  (F.col("SALARY") > 0)  )  .withColumn("salary\_d", F.col("SALARY").cast("double"))  .groupBy(  F.col("NAICS\_2022\_5\_NAME"),  F.col("LOT\_SPECIALIZED\_OCCUPATION\_NAME") # ← aquí nombre real  )  .agg(F.expr("percentile\_approx(salary\_d, 0.5)").alias("median\_salary"))  .orderBy(F.desc("median\_salary")) )  tech.show(truncate=False)  +--------------------------------------------------------------------------------------+--------------------------------+-------------+ |NAICS\_2022\_5\_NAME |LOT\_SPECIALIZED\_OCCUPATION\_NAME |median\_salary| +--------------------------------------------------------------------------------------+--------------------------------+-------------+ |Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services|Enterprise Architect |190000.0 | |Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services|SAP Analyst / Admin |145600.0 | |Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services|Oracle Consultant / Analyst |143000.0 | |Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services|General ERP Analyst / Consultant|118140.0 | |Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services|Data Analyst |80000.0 | |Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services|Business Intelligence Analyst |74298.0 | |Computing Infrastructure Providers, Data Processing, Web Hosting, and Related Services|Data Quality Analyst |40000.0 | +--------------------------------------------------------------------------------------+--------------------------------+-------------+ 4. PLOT import pandas as pd import plotly.express as px import plotly.io as pio  #pio.renderers.default = "plotly\_mimetype" pdf = tech.toPandas()  pdf = pdf.sort\_values("median\_salary", ascending=True)  fig = px.bar(  pdf,  x="LOT\_SPECIALIZED\_OCCUPATION\_NAME",   y="median\_salary",   title="Median Salary by Specialized Occupation" )  fig  Unable to display output for mime type(s): text/html  Unable to display output for mime type(s): text/html  from pyspark.sql import functions as F import plotly.express as px  # Filtrar y agrupar tech2 = (df  .where( (F.col("NAICS\_2022\_6") == F.lit("518210")) &  F.col("SALARY").isNotNull() &  (F.col("SALARY") > 0) )  .groupBy(  F.coalesce(F.col("NAICS\_2022\_6\_NAME"), F.lit("NAICS 518210")).alias("industry\_name"),  F.col("LOT\_SPECIALIZED\_OCCUPATION\_NAME")  )  .agg(F.expr("percentile\_approx(SALARY, 0.5)").alias("median\_salary"))  .orderBy(F.desc("median\_salary")) )  import matplotlib.pyplot as plt #from pyspark.sql.functions import col  df\_pandas = df.select("SALARY").toPandas() plt.hist(df\_pandas['SALARY'].dropna(), bins=30) plt.title("Distribución de salarios") plt.xlabel("Salario") plt.ylabel("Frecuencia") plt.show() |