**Case Study**

**Name: Priyeshwar**

**Mail:** [**priyesh2664@gmail.com**](mailto:priyesh2664@gmail.com)

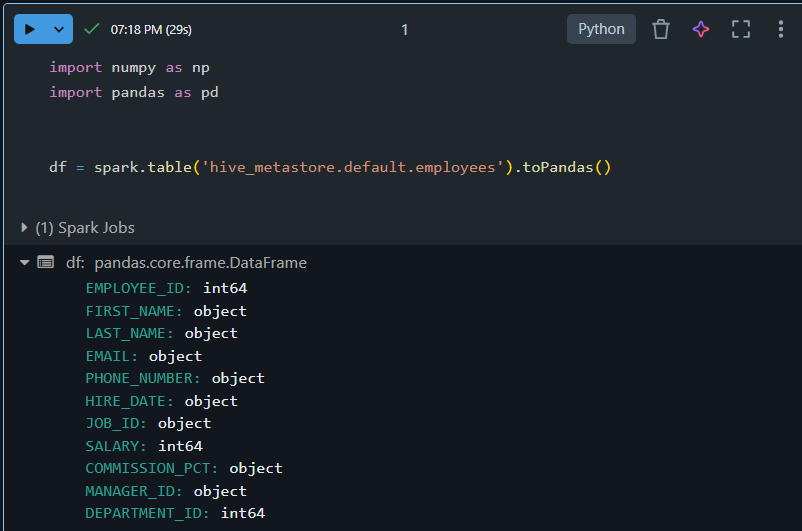
**Load CSV file**

Create a pandas DataFrame from the dataset for easier processing and visualization. Replace the file path below with your actual file location.

**import numpy as np**

**import pandas as pd**

**df = spark.table('hive\_metastore.default.employees').toPandas()**

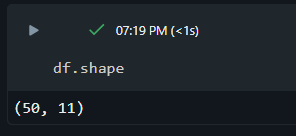


**Use pandas for data insights**

**Check shape of the DataFrame**

**df.shape**

The df.shape command returns the number of rows and columns.



**Generate descriptive statistics**

**df.describe()**

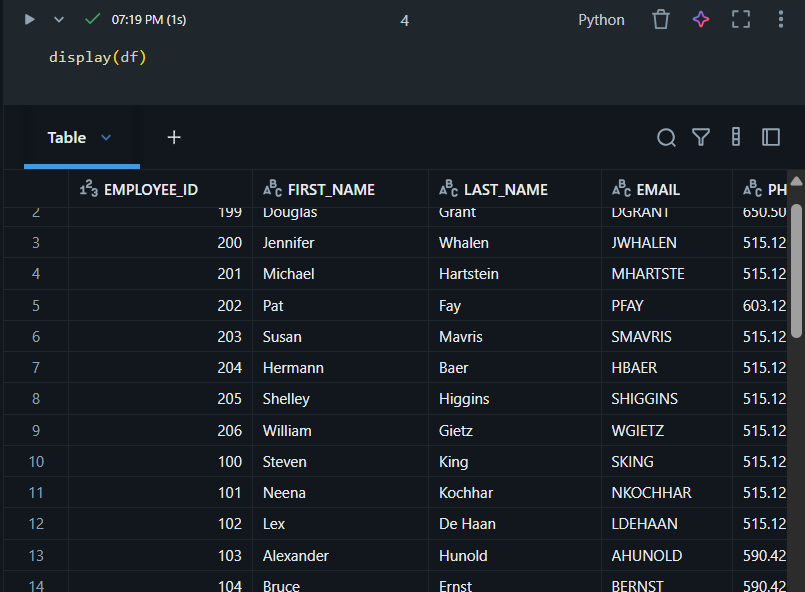
The df.describe() command provides numerical summaries such as mean, standard deviation, min, max, and percentiles.



**Generate a data profile**

display(df)

This displays a full interactive table of the data for profiling.



**Clean the data**

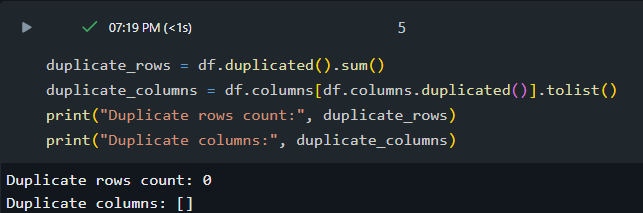
**Remove duplicate data**

duplicate\_rows = df.duplicated().sum()

duplicate\_columns = df.columns[df.columns.duplicated()].tolist()

print("Duplicate rows count:", duplicate\_rows)

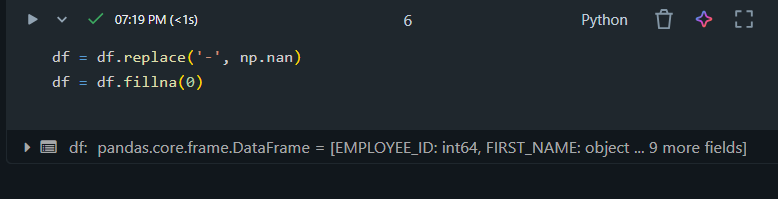
print("Duplicate columns:", duplicate\_columns)



**Handle null or missing values**

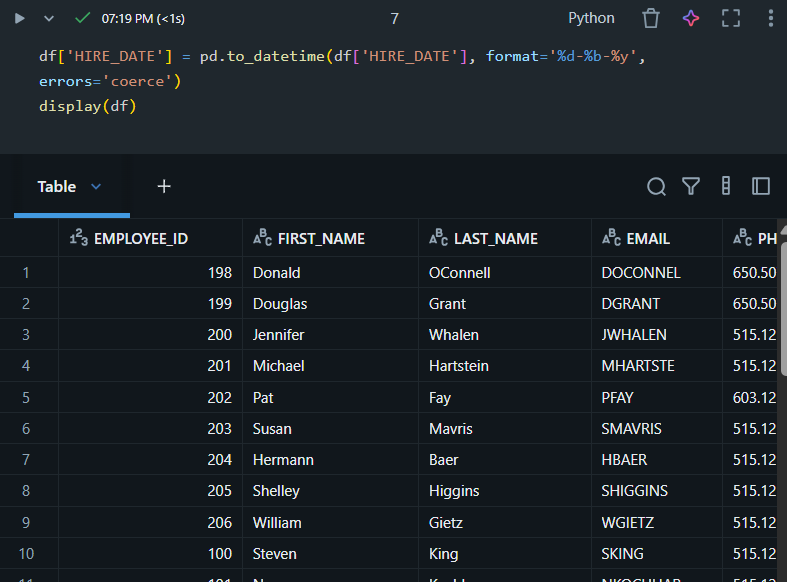
df = df.replace('-', np.nan)

df = df.fillna(0)



**Reformat dates**

df['HIRE\_DATE'] = pd.to\_datetime(df['HIRE\_DATE'], format='%d-%b-%y', errors='coerce')

display(df)

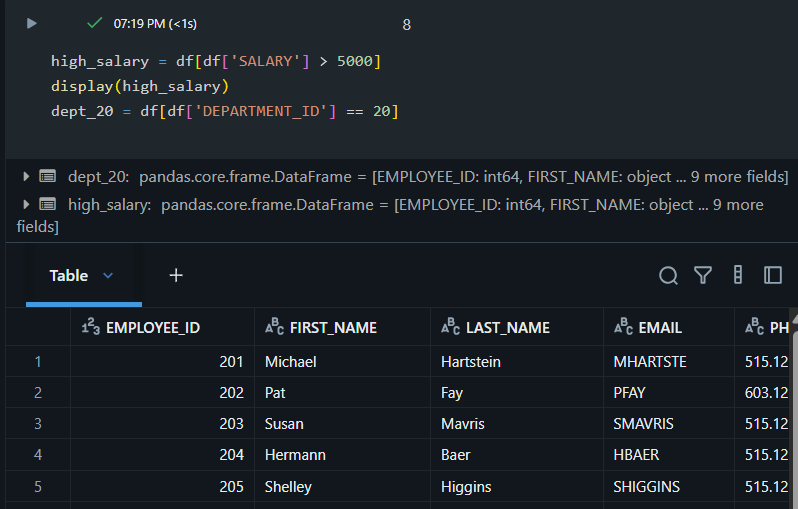
**Filter for specific conditions**

# Employees with salary > 5000

high\_salary = df[df['SALARY'] > 5000]

# Employees in department 20

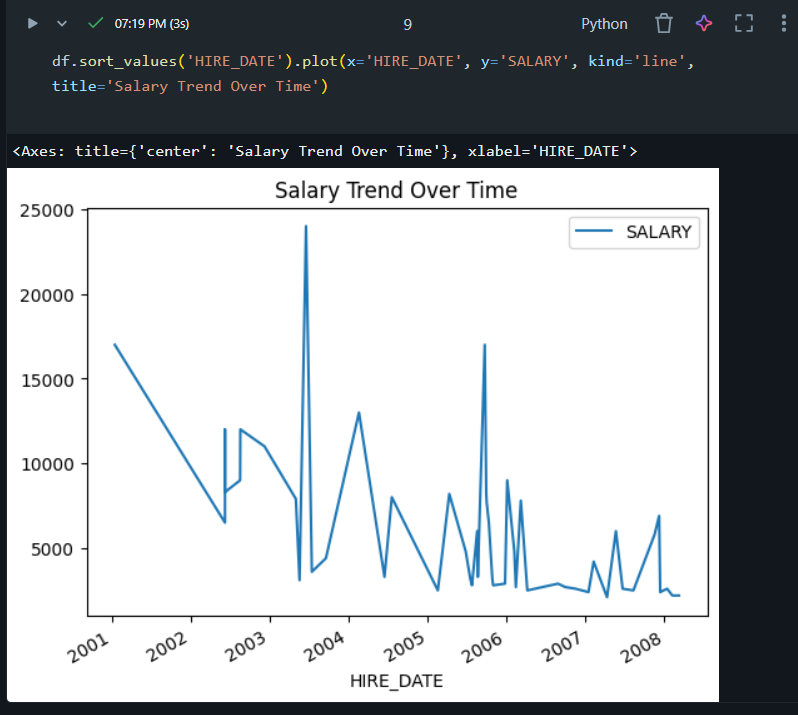
dept\_20 = df[df['DEPARTMENT\_ID'] == 20]



**Create visualizations using the dataset**

**Line Chart**

df.sort\_values('HIRE\_DATE').plot(x='HIRE\_DATE', y='SALARY', kind='line', title='Salary Trend Over Time')



**Area Chart**

df.groupby('DEPARTMENT\_ID')['SALARY'].sum().plot(kind='area', title='Total Salary by Department')



**Delta Table Operations**

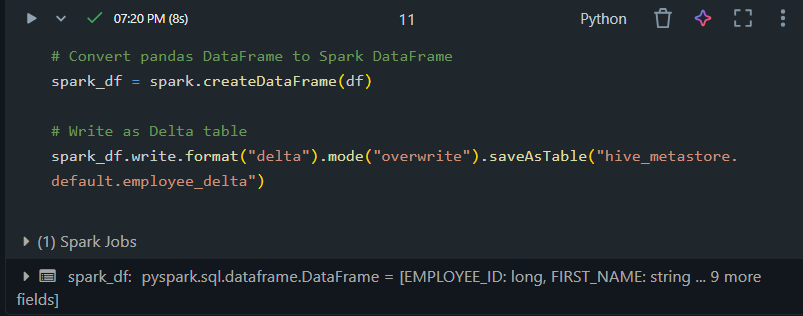
**Create a Delta Table**

# Convert pandas DataFrame to Spark DataFrame

spark\_df = spark.createDataFrame(df)

# Write as Delta table

spark\_df.write.format("delta").mode("overwrite").saveAsTable("hive\_metastore.default.employee\_delta")



**Merge & Upsert to a Delta Table**

from delta.tables import DeltaTable

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType

from datetime import datetime

# Define schema explicitly

schema = StructType([

    StructField("EMPLOYEE\_ID", IntegerType(), False),

    StructField("FIRST\_NAME", StringType(), True),

    StructField("LAST\_NAME", StringType(), True),

    StructField("EMAIL", StringType(), True),

    StructField("PHONE\_NUMBER", StringType(), True),

    StructField("HIRE\_DATE", DateType(), True),

    StructField("JOB\_ID", StringType(), True),

    StructField("SALARY", DoubleType(), True),

    StructField("COMMISSION\_PCT", DoubleType(), True),

    StructField("MANAGER\_ID", IntegerType(), True),

    StructField("DEPARTMENT\_ID", IntegerType(), True)

])

# Create source DataFrame with explicit schema

source = spark.createDataFrame([

    (205, "Alex", "Smith", "ASMITH", "650.123.9999", datetime.strptime("12-Jan-09", "%d-%b-%y"), "IT\_PROG", 9000.0, None, 101, 60)

], schema=schema)

# Merge into Delta table

target = DeltaTable.forName(spark, "hive\_metastore.default.employee\_delta")

(

    target.alias("t")

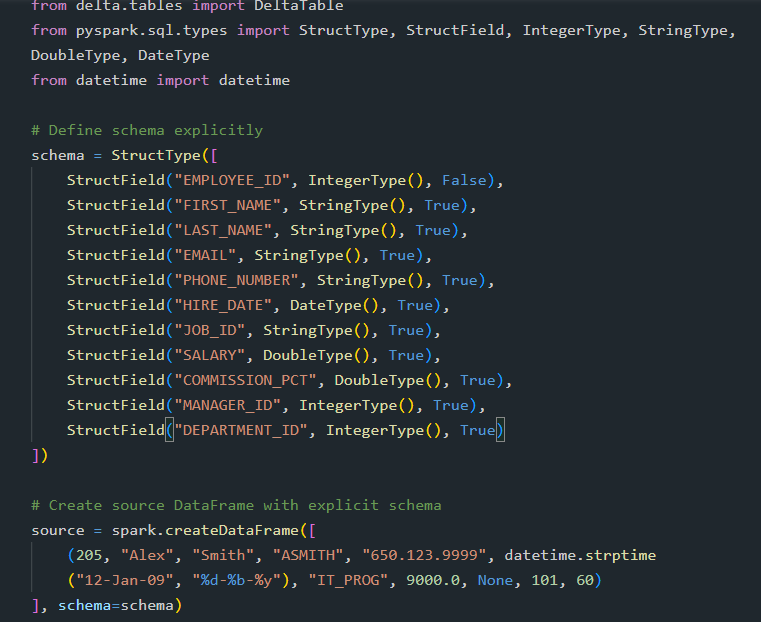
    .merge(source.alias("s"), "t.EMPLOYEE\_ID = s.EMPLOYEE\_ID")

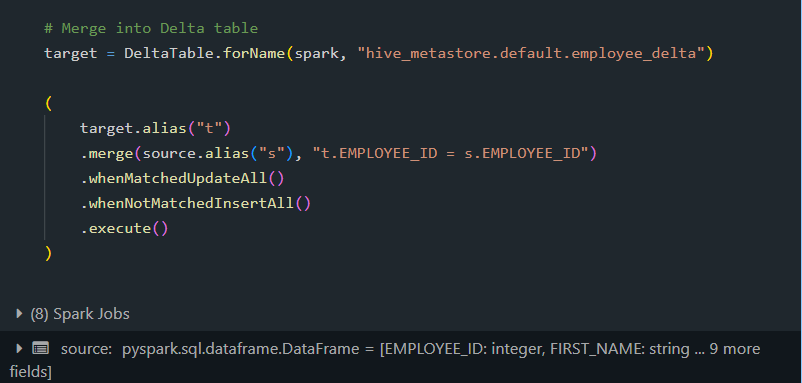
    .whenMatchedUpdateAll()

    .whenNotMatchedInsertAll()

    .execute()

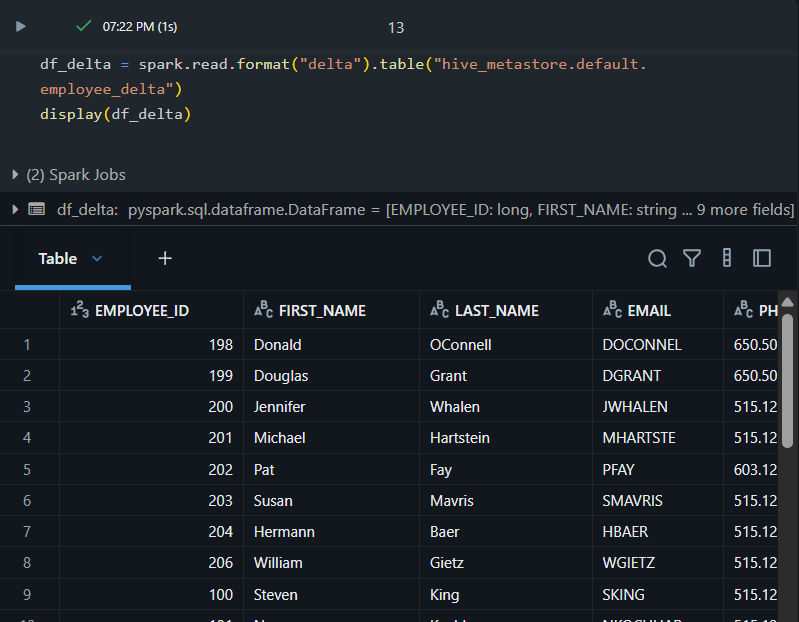
)





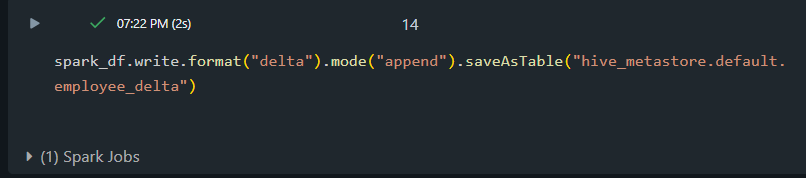
**Read a Delta Table**

df\_delta = spark.read.format("delta").table("hive\_metastore.default.employee\_delta")

display(df\_delta)

**Write to a Delta Table**

spark\_df.write.format("delta").mode("append").saveAsTable("hive\_metastore.default.employee\_delta")



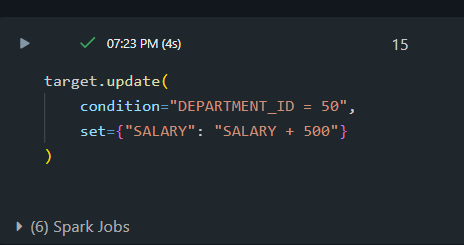
**Update a Delta Table**

target.update(

condition="DEPARTMENT\_ID = 50",

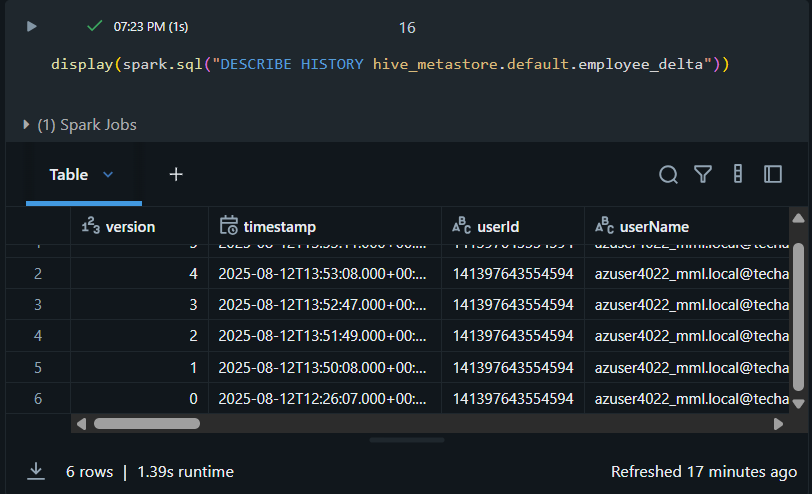
set={"SALARY": "SALARY + 500"}

)



**Display Table History**

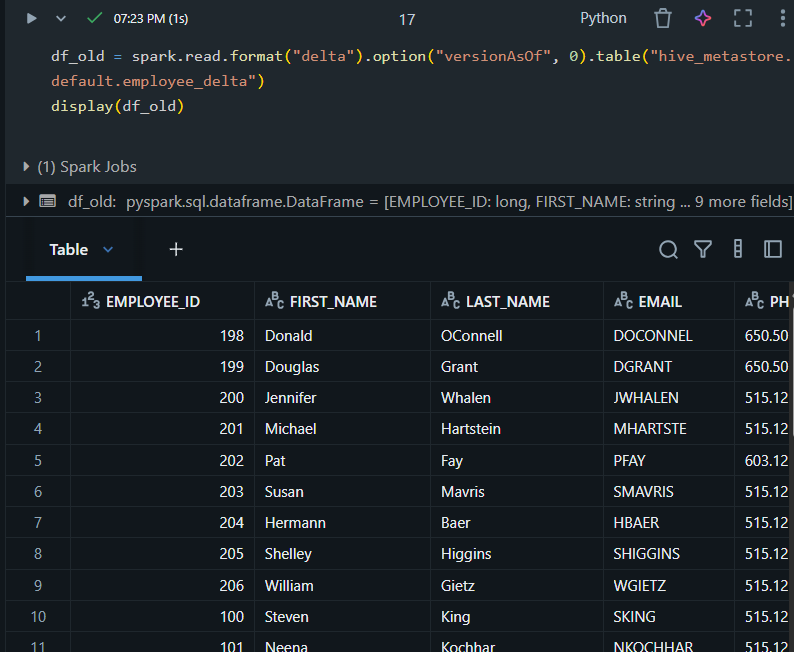
display(spark.sql("DESCRIBE HISTORY hive\_metastore.default.employee\_delta"))



**Query Earlier Version (Time Travel)**

df\_old = spark.read.format("delta").option("versionAsOf", 0).table("hive\_metastore.default.employee\_delta")

display(df\_old)



**Optimize a Delta Table**

spark.sql("OPTIMIZE hive\_metastore.default.employee\_delta")



**Clean Up Snapshots with VACUUM**

spark.sql("VACUUM hive\_metastore.default.employee\_delta RETAIN 168 HOURS")

