Dataset: HR dataset <https://docs.google.com/spreadsheets/d/1PW-b9PF0j_x6yTwgLFWdv5-mjm2egnyYmfDftW9H0Pg/edit?usp=sharing>

Sheets:

    Employee

    Department

    Locations

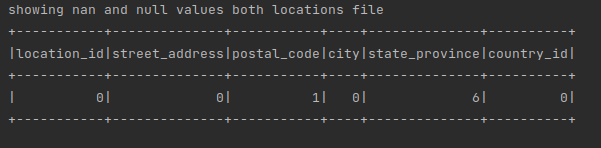
    Job\_history

    Jobs

**Tasks:**

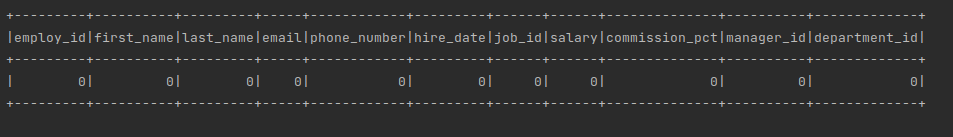
# **Count the NaN values of all the columns of locations file**

print("showing nan and null values both locations file")  
locations.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in locations.columns]).show()



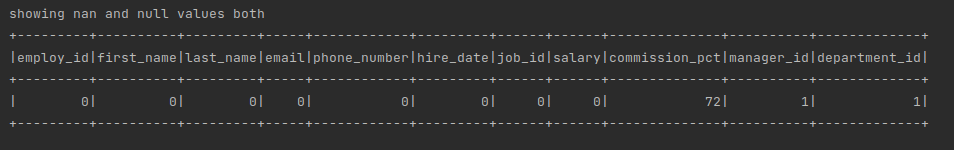
showing nan values only employee

from pyspark.sql.functions import isnan, when, count, col  
  
employee.select([count(when(isnan(c), c)).alias(c) for c in employee.columns]).show()



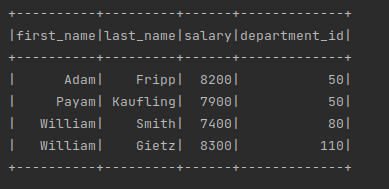
showing nan and null both values:

print("showing nan and null values both")  
employee.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in employee.columns]).show()



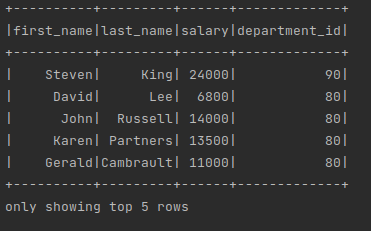
# **Display the name, salary and department number for those employees whose first name ends with the letter 'm'**

employee\_m = employee.select("first\_name", "last\_name", "salary", "department\_id").filter(employee.first\_name.like("%m"))  
employee\_m.show(5)



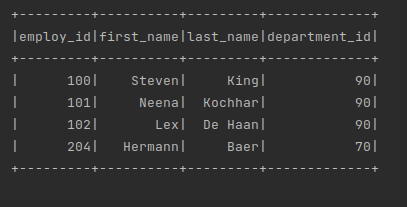
1. **display the first name, last name, salary and department number for those employees whose first name ends with the letter 'd' or 'n' or 's' and also arrange the result in descending order by department id.**

employee.select("first\_name", "last\_name", "salary", "department\_id") \ .filter(employee.first\_name.like("%d") | employee.first\_name.like("%n") | employee.first\_name.like("%s")) \ .sort(desc("department\_id")) \ .show(5)

****

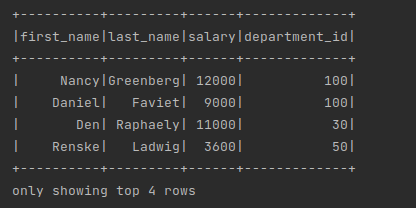
1. **Find the employees who work either in department 70 or 90.**

employee.select("employ\_id", "first\_name", "last\_name", "department\_id") \ .filter(employee.department\_id.isin(70, 90)) \ .show(5)

****

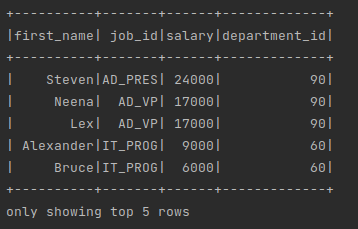
1. **Display the first, last name, salary and department number for those employees who hold a letter n as a 3rd character in their first name.**

employee.select("first\_name", "last\_name", "salary", "department\_id") \  
 .filter(employee.first\_name.like("\_\_n%")) \  
 .show(4)



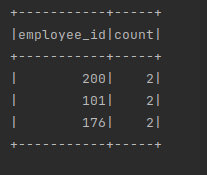
1. **Display the first name, job id, salary and department for those employees not working in the departments 50,30 and 80**

employee.select("first\_name", "job\_id", "salary", "department\_id") \  
 .filter(~employee.department\_id.isin(50, 30, 80)) \  
 .show(5)



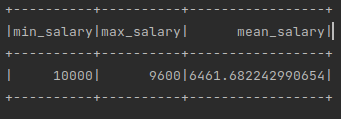
1. **Display the ID for those employees who did two or more jobs in the past.**

history.select("employee\_id") \  
 .groupBy(history.employee\_id) \  
 .count().filter("count>=2")\  
 .show()



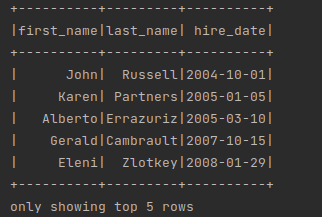
1. **Calculate minimum, maximum and mean salary from employees file**

employee.select("salary") \  
 .agg(f.min("salary").alias("min\_salary"), \  
 f.max("salary").alias("max\_salary"), \  
 f.mean("salary").alias("mean\_salary"))\  
 .show()



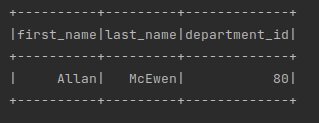
1. **display the first and last name and date of joining of the employees who is either Sales Representative or Sales Man.**

employee.select("first\_name", "last\_name", "hire\_date") \ .filter(employee.job\_id.isin('SA\_REP', 'SA\_MAN')) \ .show(5)



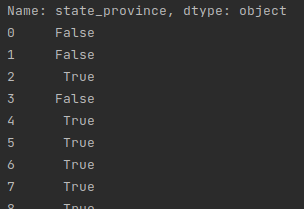
1. **display the first and last name, and department number for all employees whose last name is "McEwen".**

employee.select("first\_name", "last\_name", "department\_id") \  
 .filter(employee.last\_name=="McEwen") \  
 .show()



1. **create and display a boolean series, where True for not null and False for null values or missing values in state\_province column of locations file.**

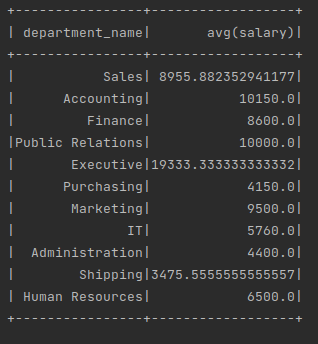
pd\_locations = pd.read\_csv('/home/aman/Downloads/locations.csv')  
print(pd\_locations.state\_province)  
print(pd\_locations.state\_province.notnull())



1. **Compute the average salary earned by each department with department name**

emp\_dep=employee.join(department, employee.department\_id == department.department\_id, "inner")

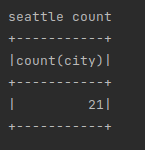
avg\_department\_name=emp\_dep.select("department\_name", "salary") \  
 .groupBy(emp\_dep.department\_name) \  
 .agg({"salary":"avg"}) \  
 .show()



1. **Compute the number of departments located in Seattle city.**

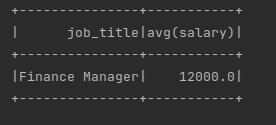
dep\_location\_join=department.join(locations, department.location\_id == locations.location\_id, "inner")  
dep\_location\_join.show()

print("seattle count")  
dep\_seattle =dep\_location\_join.filter(dep\_location\_join.city=='Seattle') \  
 .select(count("city")) \  
 .show()



1. **What is the average salary of job ‘Finance Manager’**

job\_finance\_avg = empl\_jobs\_join.filter(empl\_jobs\_join.job\_title == 'Finance Manager') \  
 .groupBy("job\_title") \  
 .agg({"salary": "avg"}) \  
 .show()



1. **You are offered a position either as ‘Sales Representative’ or ‘Marketing Representative’. You want to join a position with a smaller number of employees. Which position will you join?**

epml\_mark\_sales = empl\_jobs\_join.filter((empl\_jobs\_join.job\_title == 'Sales Representative') | (empl\_jobs\_join.job\_title == 'Marketing Representative')) \  
 .groupBy("job\_title") \  
 .count()\  
 .show()

