```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

spark = SparkSession.builder.appName("read").getOrCreate()
```

Problem

```
tweets = [(1, "let us code"),
          (2, "More than fifteen chars are here!")]

tweets_schema = """ tweet_id int , content string"""

df_tweets = spark.createDataFrame(tweets, tweets_schema)
df_tweets.display()

df_result = df_tweets.filter(length("content")<= 15)

df_result.display()

# Using col method
df_tweets.filter(length(col("content"))<= 15).display()


project = [(1, 1),
           (1,2),
           (1,3),
           (2,1),
           (2,4)]

project_schema = ''' project_id int, employee_id int'''

df_project = spark.createDataFrame(project , project_schema)
df_project.display()

employee = [(1, "Khaled", 3),
            (2, "ALi", 2),
            (3, "John", 1),
            (4, "Doe", 2)]

employee_schema = ''' employee_id int, name string , experience_years
int'''

df_employee = spark.createDataFrame(employee, employee_schema)
df_employee.display()
```

Write an SQL query that reports the average experience years of all the employees for each project, rounded to 2 digits.

Return the result table in any order.

SQL Solution

```
df_project.createOrReplaceTempView("project")
df_employee.createOrReplaceTempView("employee")

df_result = spark.sql("""
select
    p.project_id,
    ROUND(SUM(e.experience_years) / COUNT(e.employee_id), 2) AS
avg_years
FROM
    employee e
JOIN
    project p
ON
    e.employee_id = p.employee_id
GROUP BY
    p.project_id

    """)

df_result.display()

df_joined = df_employee.join(df_project,
df_employee.employee_id==df_project.employee_id, "inner")\
    .select(df_employee.employee_id.alias("Id"),
df_employee.experience_years, df_project.project_id)

df_avg =
df_joined.groupBy("project_id").agg(round(sum("experience_years")/
count("Id"), 2).alias("average_years"))

df_avg.display()
```

Problem 577

```
employee_1= [(3, "Brad", None, 4000),
             (1, "John", 3, 1000),
             (2, "Dan", 3, 2000),
             (4, "Thomas", 3, 4000)]
employee_1_schema= """ empId int, name string, supervisor int, salary
int"""

df_employee1= spark.createDataFrame(employee_1, employee_1_schema)
df_employee1.display()
```

```
bonus = [(2,500),
         (4,2000)]
bonus_schema = """empId int, bonus int """
df_bonus= spark.createDataFrame(bonus, bonus_schema)
df_bonus.display()
```

Write a answer to report the name and bonus amount of each employee with a bonus less than 1000.

Return the result table in any order.

```
df_employee1.createOrReplaceTempView("employee1")
df_bonus.createOrReplaceTempView("bonus")

df_result1= spark.sql(""" select e.name , b.bonus

          from employee1 e
         left join bonus b
          on e.empId=b.empId
          where b.bonus < 1000 or b.bonus is null
          """)
df_result1.display()

df_joined1 = df_employee1.join(df_bonus,
df_employee1.empId==df_bonus.empId, "left")

df_solution = df_joined1.filter((col("bonus") < 1000 )|
col("bonus").isNull()).select("name", "bonus")
df_solution.display()
```