databricksProject_02

DSCI 417 – Project 02

Student Grade Database

Sean Graham

Part A: Set up Environment

This part of the project involves setting up the environment.

This cell sets up the environment.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, expr

spark = SparkSession.builder.getOrCreate()
```

Part B: Load the Data

This part of the projet involves loading the data.

This cell imports each data file into a Spark DataFrame.

```
schema_accepted = 'acc_term_id STRING, sid INTEGER, first_name STRING, last_name STRING, major STRING'
accepted = spark.read.option('header', True).schema(schema_accepted).csv('/FileStore/tables/univ/accepted.csv')

schema_alumni = 'sid INTEGER'
alumni = spark.read.option('header', True).schema(schema_alumni).csv('/FileStore/tables/univ/alumni.csv')

schema_courses = 'dept STRING, course STRING, prereq STRING, credits INTEGER'
courses = spark.read.option('header', True).schema(schema_courses).csv('/FileStore/tables/univ/courses.csv')

schema_expelled = 'sid INTEGER'
expelled = spark.read.option('header', True).schema(schema_expelled).csv('/FileStore/tables/univ/expelled.csv')

schema_faculty = 'fid INTEGER, first_name STRING, last_name STRING, dept STRING'
faculty = spark.read.option('header', True).schema(schema_faculty).csv('/FileStore/tables/univ/faculty.csv')

schema_grades = 'term_id STRING, course STRING, sid INTEGER, fid INTEGER, grade STRING'
grades = spark.read.option('header', True).schema(schema_grades).csv('/FileStore/tables/univ/grades.csv')

schema_unretained = 'sid INTEGER'
unretained = spark.read.option('header', True).schema(schema_unretained).csv('/FileStore/tables/univ/unretained.csv')
```

This cell prints the number of records in each DataFrame.

```
print("The number of records in accepted is " + str(accepted.count()) + ".")
print("The number of records in alumni is " + str(alumni.count()) + ".")
print("The number of records in courses is " + str(courses.count()) + ".")
print("The number of records in expelled is " + str(expelled.count()) + ".")
print("The number of records in faculty is " + str(faculty.count()) + ".")
print("The number of records in grades is " + str(grades.count()) + ".")
print("The number of records in unretained is " + str(unretained.count()) + ".")
```

```
The number of records in accepted is 12207.
The number of records in alumni is 4920.
The number of records in courses is 119.
The number of records in expelled is 403.
The number of records in faculty is 330.
The number of records in grades is 285137.
The number of records in unretained is 2289.
```

Part C: Student Count by Status

This part of the project involves counting the number of students in each of the following groups: students who have been accepted, students who actually enrolledin courses, current students, all former students, alumni, unretained students, and studentswho were expelled.

This cell will create a three new DataFrames to store student info for students in various categories. It will then generate the desired counts.

```
enrolled = accepted.join(other=grades, on='sid', how='semi')

current = enrolled.join(other=alumni, on='sid', how='anti').join(other=unretained, on='sid', how='anti').join(other=expelled, on='sid', how='anti')

former = enrolled.join(other=current, on='sid', how='anti')

print("Number of accepted students:   " + str(accepted.count()))
print("Number of enrolled students:   " + str(enrolled.count()))
print("Number of current students:    " + str(current.count()))
print("Number of former students:     " + str(former.count()))
print("Number of unretained students: " + str(unretained.count()))
print("Number of expelled students:   " + str(expelled.count()))
print("Number of alumni:              " + str(alumni.count()))
```

```
Number of accepted students:   12207
Number of enrolled students:   9667
Number of current students:    2055
Number of former students:     7612
Number of unretained students: 2289
Number of expelled students:   403
Number of alumni:              4920
```

Part D: Distribution of Students by Major

This part of the project involves determining the number of students currently in each major, as well as the proportion of the overall number of students in each major.

This cell will determine of the number of students currently in each major, as well as the proportion of the overall number of students in each major.

```
var = current.count()

(
    current
    .groupBy('major')
    .agg(expr('COUNT(*) AS n_students'))
    .withColumn('prop', expr(f'round(n_students/{var}, 4)'))
    .sort('prop', ascending = False)
    .show()
)
```

```
+-----+----------+------+
|major|n_students|  prop|
+-----+----------+------+
|  BIO|       615|0.2993|
|  CSC|       508|0.2472|
|  CHM|       405|0.1971|
|  MTH|       320|0.1557|
|  PHY|       207|0.1007|
+-----+----------+------+
```

Part E: Course Enrollments by Department

This part of the project involves determining of the number of students enrolled in courses offered by each department duringthe Spring 2021 term.

This cell will determine of the number of students enrolled in courses offered by each department duringthe Spring 2021 term.

```
sp21_enr = grades.filter(expr('term_id == "2021A"')).count()

(
    grades
    .filter(expr('term_id == "2021A"'))
    .join(other=courses, on='course', how='inner')
    .groupBy('dept')
    .agg(expr('COUNT(*) AS n_students'))
    .withColumn('prop', expr(f'round(n_students/{sp21_enr}, 4)'))
    .sort('prop', ascending = False)
    .show()
)
```

```
+----+----------+------+
|dept|n_students|  prop|
+----+----------+------+
| GEN|      5142|0.4198|
| BIO|      1786|0.1458|
| MTH|      1517|0.1238|
| CHM|      1512|0.1234|
| CSC|      1479|0.1207|
| PHY|       814|0.0664|
+----+----------+------+
```

Part F: Graduation Rates by Major

This part of the project involves determining the graduation rates for each major.

This cell will create a DataFrame containing the number of former students in each major.

```
former_by_major = (
    former
    .groupBy('major')
    .agg(expr('COUNT(*) AS n_former'))
    .sort('major', ascending = True)
)

former_by_major.show()
```

```
+-----+--------+
|major|n_former|
+-----+--------+
|  BIO|    2243|
|  CHM|    1527|
|  CSC|    1940|
|  MTH|    1139|
|  PHY|     763|
+-----+--------+
```

This cell will determine the number of alumni for each major.

```
alumni_by_major = (
    former
    .join(other=alumni, on='sid', how='semi')
    .groupBy('major')
    .agg(expr('COUNT(*) AS n_alumni'))
    .sort('major', ascending = True)
)

alumni_by_major.show()
```

```
+-----+--------+
```

```
|major|n_alumni|
+-----+--------+
|  BIO|    1485|
|  CHM|    1017|
|  CSC|    1231|
|  MTH|     723|
|  PHY|     464|
+-----+--------+
```

This cell will use the previous two DataFrames to determine the graduation rates.

```
(
    alumni_by_major
    .join(other=former_by_major, on='major', how='inner')
    .withColumn('grad_rate', expr('round(n_alumni/n_former, 4)'))
    .sort('major', ascending = True)
    .show()
)
```

```
+-----+--------+--------+---------+
|major|n_alumni|n_former|grad_rate|
+-----+--------+--------+---------+
|  BIO|    1485|    2243|   0.6621|
|  CHM|    1017|    1527|    0.666|
|  CSC|    1231|    1940|   0.6345|
|  MTH|     723|    1139|   0.6348|
|  PHY|     464|     763|   0.6081|
+-----+--------+--------+---------+
```

Part G: Number of Terms Required for Graduation

This part of the project involves finding a frequency distribution for the number of terms that alumni required for graduation.

This cell will find a frequency distribution for the number of terms that alumni required for graduation.

```
(
    grades
    .join(other=alumni, on='sid', how='semi')
    .groupBy('sid')
    .agg(expr('COUNT(DISTINCT term_id) AS n_terms'))
    .groupBy('n_terms')
    .agg(expr('COUNT(*) AS n_alumni'))
    .sort('n_terms', ascending = True)
    .show()
)
```

```
+-------+--------+
|n_terms|n_alumni|
+-------+--------+
|      7|     200|
|      8|    3045|
|      9|    1203|
|     10|     241|
|     11|     121|
|     12|      46|
|     13|      32|
|     14|      14|
|     15|       7|
|     16|       7|
|     17|       2|
```

```
|     18|       1|
|     25|       1|
+-------+--------+
```

Part H: Current Student GPA

This part of the project involves calaculating the GPA of each current student at SU and analyzing the results.

This cell will calculate the GPA of each current student at SU and will analyze the results.

```
def grade_letter(letter):
    grade_numbers = {'A': 4, 'B': 3, 'C': 2, 'D': 1, 'F': 0}
    return grade_numbers[str(letter)]
spark.udf.register('grade_letter', grade_letter)
```

```
Out[11]: <function __main__.grade_letter(letter)>
```

This cell will calculate the GPA of each student currently enrolled at SU.

```
current_gpa = (
    grades
    .join(other=courses, on='course', how='inner')
    .withColumn('num_grade', expr('grade_letter(grade)'))
    .withColumn('gp', expr('credits * num_grade'))
    .groupBy('sid')
    .agg(
        expr("SUM(gp) AS sum_gp"),
        expr("SUM(credits) AS sum_credits")
    )
    .withColumn('gpa', expr('round(sum_gp/sum_credits, 2)'))
    .join(other=current, on='sid', how='inner')
    .select('sid', 'first_name', 'last_name', 'major', 'gpa')
    .sort('gpa', ascending = True)
)

current_gpa.show(10)
```

```
+------+----------+---------+-----+----+
|   sid|first_name|last_name|major| gpa|
+------+----------+---------+-----+----+
|111582|       Amy|Alexander|  CHM|0.29|
|111316|    Harold| Mitchell|  BIO|0.45|
|111120|  Lawrence| Sullivan|  BIO|0.54|
|111084|      Emma|    Ortiz|  PHY|0.57|
|111008|     Wayne|  Coleman|  CSC| 0.6|
|111947|     Peter| Crawford|  CSC| 0.6|
|112082|   Barbara| Thompson|  PHY| 0.6|
|111250|  Margaret|   Butler|  PHY|0.62|
|111909| Christine|    Gomez|  BIO|0.65|
|111258|     Alice|   Butler|  BIO|0.66|
+------+----------+---------+-----+----+
only showing top 10 rows
```

This cell will determine the number of current students with perfect 4.0 GPAs.

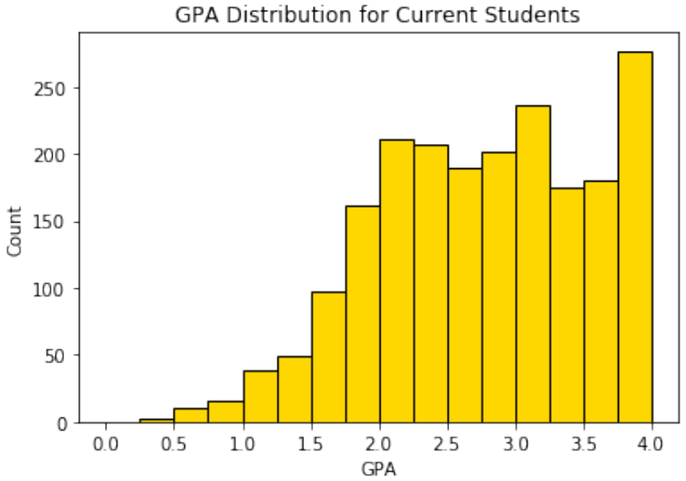
```
perfect_score = current_gpa.filter(expr('gpa == 4.0')).count()
print(perfect_score)
```

```
95
```

The next cell will create a histogram displaying the distribution of GPAs for current students.

```
current_gpa_pandas = current_gpa.toPandas()

plt.hist(current_gpa_pandas['gpa'], bins=np.arange(0, 4.25, 0.25), color='gold', edgecolor='k')
plt.title('GPA Distribution for Current Students')
plt.xlabel('GPA')
plt.ylabel('Count')
plt.show()
```



Part I: Grade Distribution by Instructor

This part of the project involves determining the proportion of A, B, C, D, and F grades given out by each faculty member at SU.

This cell will determine the proportion of A, B, C, D, and F grades given out by each faculty member at SU.

```
faculty_grade_dist = (
    grades
    .groupBy('fid')
    .agg(
        expr('COUNT(*) AS N'),
        expr('SUM(CASE WHEN grade == "A" THEN 1 ELSE 0 END) AS countA'),
        expr('SUM(CASE WHEN grade == "B" THEN 1 ELSE 0 END) AS countB'),
        expr('SUM(CASE WHEN grade == "C" THEN 1 ELSE 0 END) AS countC'),
        expr('SUM(CASE WHEN grade == "D" THEN 1 ELSE 0 END) AS countD'),
        expr('SUM(CASE WHEN grade == "F" THEN 1 ELSE 0 END) AS countF')
    )
    .join(other=faculty, on='fid', how='inner')
    .select('fid', 'first_name', 'last_name', 'dept', 'N',
           expr('round(countA / N, 2) AS propA'),
           expr('round(countB / N, 2) AS propB'),
           expr('round(countC / N, 2) AS propC'),
           expr('round(countD / N, 2) AS propD'),
           expr('round(countF / N, 2) AS propF'))
)

faculty_grade_dist.show(5)
```

```
+----+----------+---------+----+----+-----+-----+-----+-----+-----+
| fid|first_name|last_name|dept|   N|propA|propB|propC|propD|propF|
+----+----------+---------+----+----+-----+-----+-----+-----+-----+
|1088| Stephanie| Williams| MTH|1666| 0.17| 0.32| 0.36| 0.13| 0.01|
|1238|    Willie|    Black| BIO| 682| 0.48| 0.33| 0.15| 0.04| 0.01|
|1829|     Bobby|   Wilson| GEN| 640| 0.13| 0.26| 0.37|  0.2| 0.04|
|1025|  Patricia|   Rogers| CSC|2950|  0.3| 0.33|  0.3| 0.07|  0.0|
|1084|     Susan|  Edwards| MTH|  80| 0.14| 0.36| 0.38| 0.11| 0.01|
+----+----------+---------+----+----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

This cell will identify the 10 faculty members who assign the fewest A grades.

```
faculty_grade_dist.filter(expr('N >= 100')).sort('propA', ascending = True).show(10)
```

```
+----+----------+---------+----+---+-----+-----+-----+-----+-----+
| fid|first_name|last_name|dept|  N|propA|propB|propC|propD|propF|
+----+----------+---------+----+---+-----+-----+-----+-----+-----+
|1628|   Rebecca|  Stewart| GEN|395| 0.03|  0.2| 0.37| 0.28| 0.12|
|1481|   Abigail|   Brooks| BIO|311| 0.05| 0.23| 0.37| 0.28| 0.08|
|3187|    Joshua|  Griffin| GEN|154| 0.05| 0.24| 0.38| 0.21| 0.12|
|1264|     Carol|   Martin| CSC|302| 0.07| 0.22| 0.34| 0.25| 0.13|
|1039|      Joan|      Lee| BIO|147| 0.07| 0.24| 0.31| 0.33| 0.05|
|1479|     Karen|  Simmons| GEN|310| 0.08| 0.31|  0.3| 0.21|  0.1|
|1212|   Michael| Martinez| MTH|775| 0.09| 0.23| 0.31| 0.27| 0.11|
|1591|    Amanda| Mitchell| CHM|395| 0.09| 0.29| 0.35| 0.18| 0.08|
|1462|     Ralph|    Perez| BIO|195| 0.09| 0.21| 0.39| 0.23| 0.09|
|2925|   Cynthia|    Lewis| GEN|302| 0.09| 0.22| 0.35| 0.25| 0.09|
+----+----------+---------+----+---+-----+-----+-----+-----+-----+
only showing top 10 rows
```

The next cell will identify the 10 faculty members who award A's most frequently.

```
faculty_grade_dist.filter(expr('N >= 100')).sort('propA', ascending = False).show(10)
```

```
+----+----------+---------+----+----+-----+-----+-----+-----+-----+
| fid|first_name|last_name|dept|   N|propA|propB|propC|propD|propF|
+----+----------+---------+----+----+-----+-----+-----+-----+-----+
|1092|    Hannah|   Morgan| GEN|1953| 0.59|  0.3| 0.09| 0.01|  0.0|
|1262|      Sara|   Hunter| GEN|2266| 0.57| 0.28| 0.12| 0.02|  0.0|
|1094|      Judy|Patterson| GEN|1904| 0.55| 0.32| 0.12| 0.02|  0.0|
```

```
|1548|    Donald|   Gibson| GEN|1478| 0.55| 0.27| 0.14| 0.03| 0.01|
|1484|     Billy|   Cooper| BIO| 434| 0.54| 0.33| 0.09| 0.04| 0.01|
|1328|     David|   Parker| GEN|1543| 0.53| 0.28| 0.14| 0.04| 0.01|
|1058|      John|  Simpson| GEN|2729| 0.53| 0.33| 0.12| 0.02|  0.0|
|1038|   Theresa|  Stevens| CHM| 233| 0.52| 0.31| 0.15| 0.02| 0.01|
|1291|     Joyce|   Butler| GEN|2212| 0.51| 0.31| 0.14| 0.03| 0.01|
|1305|     Betty|  Stewart| GEN|2081| 0.51| 0.31| 0.14| 0.04| 0.01|
+----+----------+---------+----+----+-----+-----+-----+-----+-----+
only showing top 10 rows
```

Part J: First Term GPA

This part of the project involves calculating the first-term GPA for each student who has enrolledin classesat SU.

The next cell will calculate the first-term GPA for each student who has enrolled in classes at SU.

```
first_term_gpa = (
    grades
    .join(other=accepted, on='sid', how='inner')
    .filter(expr('term_id == acc_term_id'))
    .join(other=courses, on='course', how='inner')
    .withColumn('num_grade', expr('grade_letter(grade)'))
    .withColumn('gp', expr('credits * num_grade'))
    .groupBy('sid')
    .agg(
        expr("SUM(gp) AS sum_gp"),
        expr("SUM(credits) AS sum_credits")
    )
    .withColumn('first_term_gpa', expr('round(sum_gp/sum_credits, 2)'))
    .select('sid', 'first_term_gpa')
)

first_term_gpa.show(5)
```

```
+------+--------------+
|   sid|first_term_gpa|
+------+--------------+
|100170|           2.0|
|100446|          3.83|
|100800|          0.39|
|100884|           2.0|
|100986|          0.33|
+------+--------------+
only showing top 5 rows
```

Part K: Graduation Rates and First Term GPA

This part of the project invovles calculating graduation rates for students whose first term GPA falls into each of four different grade ranges.

This cell will establish bins for the four grade ranges.

```
def gpa_bin(gpa):
  if gpa < 1:
    return "[0,1)"
  elif gpa < 2:
    return "[1,2)"
  elif gpa < 3:
    return "[2,3)"
  else:
    return "[3,4]"

spark.udf.register('gpa_bin', gpa_bin)
```

```
Out[19]: <function __main__.gpa_bin(gpa)>
```

This cell will calculate the number of alumni whose first-term GPA falls into each bin.

```
alumni_ft_gpa = (
    first_term_gpa
    .join(other=alumni, on='sid', how='semi')
    .withColumn('gpa_bin', expr('gpa_bin(first_term_gpa)'))
    .groupBy('gpa_bin')
    .agg(expr('COUNT(*) AS n_alumni'))
    .sort('gpa_bin', ascending = True)
)

alumni_ft_gpa.show()
```

```
+-------+--------+
|gpa_bin|n_alumni|
+-------+--------+
|  [0,1)|       4|
|  [1,2)|     549|
|  [2,3)|    1887|
|  [3,4]|    2480|
+-------+--------+
```

This cell will determine the number of former students whose first-term GPA falls into each bin.

```
former_ft_gpa = (
    first_term_gpa
    .join(other=former, on='sid', how='semi')
    .withColumn('gpa_bin', expr('gpa_bin(first_term_gpa)'))
    .groupBy('gpa_bin')
    .agg(expr('COUNT(*) AS n_former'))
    .sort('gpa_bin', ascending = True)
)

former_ft_gpa.show()
```

```
+-------+--------+
|gpa_bin|n_former|
+-------+--------+
|  [0,1)|     822|
|  [1,2)|    1735|
|  [2,3)|    2433|
|  [3,4]|    2622|
+-------+--------+
```

This cell will use the previous two DataFrames to determine the graduation rates for each of the GPA bins.

```
+-------+--------+--------+---------+
|gpa_bin|n_alumni|n_former|grad_rate|
+-------+--------+--------+---------+
|  [0,1)|       4|     822|   0.0049|
|  [1,2)|     549|    1735|   0.3164|
|  [2,3)|    1887|    2433|   0.7756|
|  [3,4]|    2480|    2622|   0.9458|
+-------+--------+--------+---------+
```