ADS-500B Data Science Programming

Assignment 5.1 MySQL Output

Created By: Gabe Duffy

SQL Commands

1.1) How many countries became independent in the twentieth century?

```
1 •  SELECT COUNT(*) AS num_country_independent_20th_century
2    FROM world.country
3    WHERE YEAR(IndepYear) BETWEEN 1901 AND 2000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_country_independent_20th_century
1

1.2) How many people in the world are expected to live for 75 years or more?

```
1 •  SELECT COUNT(*) AS num_people_over_75
2    FROM world.country
3    WHERE LifeExpectancy >= 75;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_people_over_75
62

1.3) List the 10 most populated countries in the world with their populations as a percentage of the world population

```
    name AS country_name,
    Population,
    (Population / (SELECT SUM(Population) FROM world.country)) * 100 AS percentage_of_world_population
FROM
    world.country
ORDER BY
    Population DESC
LIMIT
    10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

country_name	Population	percentage_of_world_population
China	1277558000	21.0168
India	1013662000	16.6755
United States	278357000	4.5792
Indonesia	212107000	3.4893
Brazil	170115000	2.7985
Pakistan	156483000	2.5743
Russian Federation	146934000	2.4172
Bangladesh	129155000	2.1247
Japan	126714000	2.0845
Nigeria	111506000	1.8344

1.4) List the top 10 countries with the highest population density.

```
SELECT
    name AS country_name,
    Population / SurfaceArea AS population_density
FROM
    world.country
ORDER BY
    population_density DESC
LIMIT
    10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conter

country_name	population_density
Macao	26277.777778
Monaco	22666.666667
Hong Kong	6308.837209
Singapore	5771.844660
Gibraltar	4166.666667
Holy See (Vatican City State)	2499.999963
Bermuda	1226.415094
Malta	1203.164557
Maldives	959.731544
Bangladesh	896.922179

1.5) How many countries are there in each “Region” ?

```
1  SELECT
2      Region,
3      COUNT(*) AS country_count
4  FROM
5      world.country
6  GROUP BY
7      Region
8  ORDER BY
9      country_count DESC;
```

Result Grid | Filter Rows:

Region	country_count
Caribbean	24
Eastern Africa	20
Middle East	18
Western Africa	17
Southern Europe	15
Southern and Central Asia	14
South America	14
Southeast Asia	11
Polynesia	10
Eastern Europe	10

1.6) What countries have more than 10 languages represented?

```
41  -- 1.6
42  SELECT
43      Name,
44      COUNT(cl.Language) AS language_count
45  FROM
46      world.countrylanguage cl
47  JOIN
48      world.country c ON cl.CountryCode = c.Code
```

Result Grid | Filter Rows: | Export: | Wrap

Name	language_count
Canada	12
China	12
India	12
Russian Federation	12
United States	12
Tanzania	11
South Africa	11

Python Commands

2.1) Use Python to explore the relationship of different variables to models per gallon (mpg). Find out which of the variables have high correlation with mpg. Report those values. Build a regression model using one of those variables to predict mpg. Do the same using two of those variables. Report your models along with the regression line equations

Step # 1

```
import pandas as pd
from sqlalchemy import create_engine

# SQLAlchemy engine
engine = create_engine("mysql+mysqlconnector://root:GDcoug2024@localhost/auto")

# Pull data from MySQL database
query = "SELECT * FROM mpg"  # Adjust the table name if needed
auto = pd.read_sql(query, engine)

# Understand the data
print(auto.head())
```

```
   mpg  cylinders  displacement  horsepower  weight acceleration  model year  \
0   18          8           307         130    3504           12          70
1   15          8           350         165    3693         11.5          70
2   18          8           318         150    3436           11          70
3   16          8           304         150    3433           12          70
4   17          8           302         140    3449         10.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

Step # 2 Find out which of the variables have high correlation with mpg. Report those values. Build a regression model using one of those variables to predict mpg. Do the same using two of those variables. Report your models along with the regression line equations.

Visual Below:

```python
import pandas as pd
from sklearn.linear_model import LinearRegression

# Fetch data from MySQL database
query = "SELECT * FROM mpg"  # Adjust the table name if needed
auto = pd.read_sql(query, engine)

#Calculate correlation coefficients
correlation_matrix = auto.corr()
mpg_correlation = correlation_matrix['mpg'].sort_values(ascending=False)

# Identify variables with high correlation with mpg
high_corr_variables = mpg_correlation[(mpg_correlation > 0.5) | (mpg_correlation < -0.5)]
print("Variables with high correlation with mpg:")
print(high_corr_variables)

# Start with one variable
variable1 = 'horsepower'  # Example variable with high correlation
X_single = auto[[variable1]]
y = auto['mpg']

# Initialize and fit the model
regression_single = LinearRegression()
regression_single.fit(X_single, y)

# one variable report
print("\nRegression Model with One Variable ({})".format(variable1))
print("Coefficient:", regression_single.coef_[0])
print("Intercept:", regression_single.intercept_)
print("Regression Line Equation: mpg = {:.2f} * {} + {:.2f}".format(regression_single.coef_[0], variable1, regression_single.int


# Example with two variables
variable2 = 'weight'
X_double = auto[[variable1, variable2]]

# Initialize and fit the model
regression_double = LinearRegression()
regression_double.fit(X_double, y)

# Two variable Report
print("\nRegression Model with Two Variables ({}, {})".format(variable1, variable2))
print("Coefficients:", regression_double.coef_)
print("Intercept:", regression_double.intercept_)
print("Regression Line Equation: mpg = {:.2f} * {} + {:.2f} * {} + {:.2f}".format(regression_double.coef_[0], variable1, regress
```

```
Variables with high correlation with mpg:
mpg             1.000000
model year      0.582750
origin          0.563667
cylinders      -0.776796
horsepower     -0.777683
displacement   -0.804304
weight         -0.831535
Name: mpg, dtype: float64

Regression Model with One Variable (horsepower)
Coefficient: -0.1575912226389155
Intercept: 39.955805483441
Regression Line Equation: mpg = -0.16 * horsepower + 39.96

Regression Model with Two Variables (horsepower, weight)
Coefficients: [-0.04716711 -0.00578798]
Intercept: 45.65407841469383
Regression Line Equation: mpg = -0.05 * horsepower + -0.01 * weight + 45.65
```

R Commands

2.2) Use R to understand how horsepower and weights are related to each other. Plot them using a scatter plot and color the data points using mpg. Do you see anything interesting/useful here? Report your observations with this plot. Now let us cluster the data on this plane in a “reasonable” number of groups. Show your plot where the data points are now colored with the cluster information and provide your interpretations.

Step # 1 Load RMySQL package and establish connection

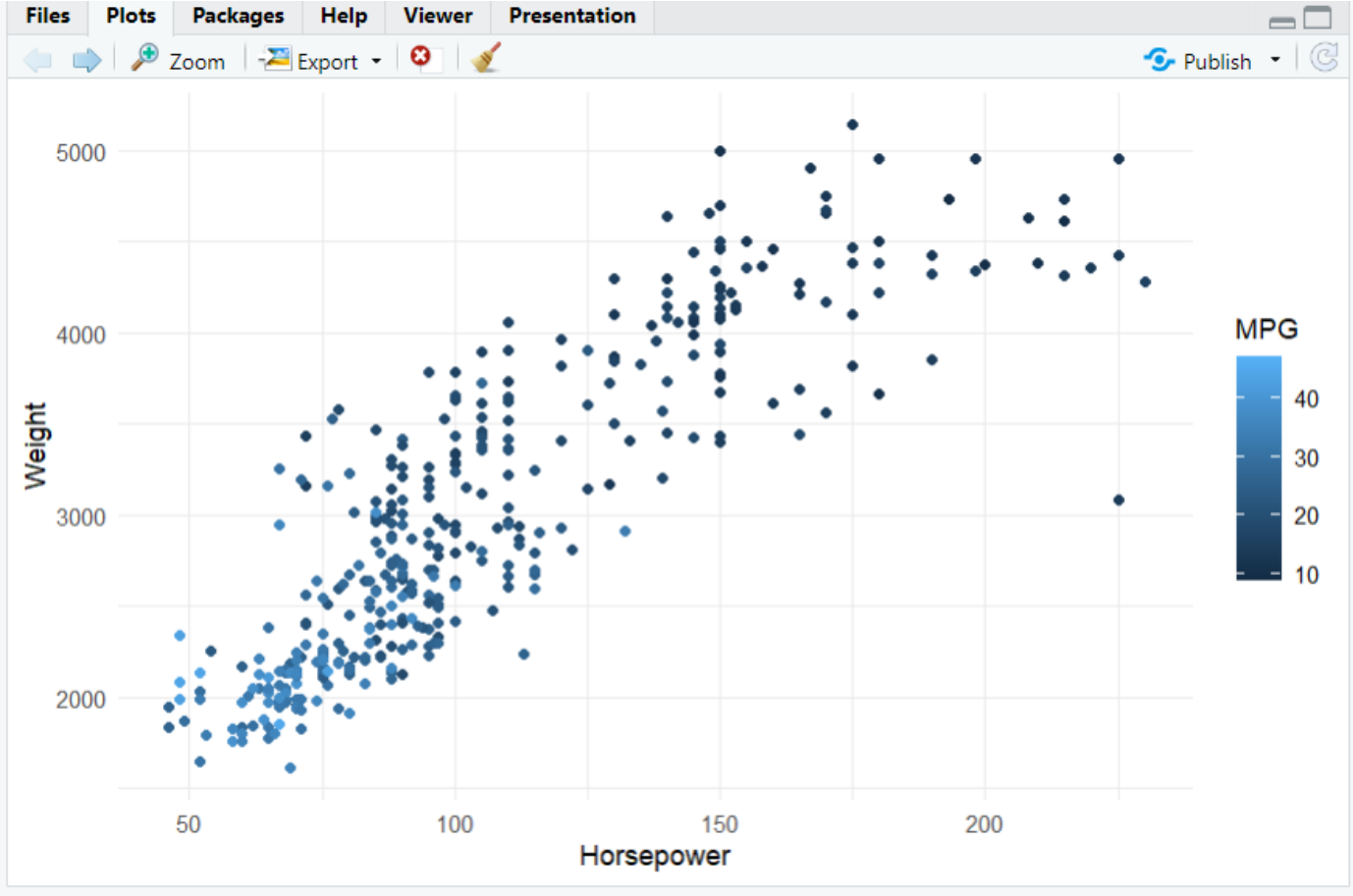
```
> # Load the RMySQL package
> library(RMySQL)
Loading required package: DBI
Warning messages:
1: package ‘RMySQL’ was built under R version 4.3.3
2: package ‘DBI’ was built under R version 4.3.3
>
> # Establish a connection to your MySQL database
> con <- dbConnect(MySQL(),
+                  dbname = "auto",
+                  host = "localhost",
+                  port = 3306,
+                  user = "root",
+                  password = "GDcoug2024")
>
> # Execute SQL queries
> result <- dbGetQuery(con, "SELECT * FROM mpg")
>
> # Close the connection
> dbDisconnect(con)
[1] TRUE
```

```
> # Fetch data from MySQL database
> mpg_data <- dbGetQuery(con, "SELECT horsepower, weight, mpg FROM mpg")
Error in .local(dbObj, ...) :
  internal error in RS_DBI_getConnection: corrupt connection handle
> # Establish connection to MySQL database
> con <- dbConnect(MySQL(),
+                  dbname = "auto",
+                  host = "localhost",
+                  port = 3306,
+                  user = "root",
+                  password = "GDcoug2024")
>
> # Fetch data from MySQL database
> mpg_data <- dbGetQuery(con, "SELECT horsepower, weight, mpg FROM mpg")
>
> # Close the database connection
> dbDisconnect(con)
[1] TRUE
`
```

Step # 2 Plot them using a scatter plot and color the data points using mpg. Do you see anything interesting/useful here? Report your observations with this plot.

```
> # Create scatter plot
> scatter_plot <- ggplot(mpg_data, aes(x = horsepower, y = weight, color = mpg)) +
+     geom_point() +
+     labs(x = "Horsepower", y = "Weight", color = "MPG") +
+     theme_minimal()
Error in ggplot(mpg_data, aes(x = horsepower, y = weight, color = mpg)) :
  could not find function "ggplot"
> # Load necessary libraries
> library(RMySQL)
> library(ggplot2)
Warning message:
package ‘ggplot2’ was built under R version 4.3.3
>
> # Establish connection to MySQL database
> con <- dbConnect(MySQL(),
+                  dbname = "auto",
+                  host = "localhost",
+                  port = 3306,
+                  user = "root",
+                  password = "GDcoug2024")
>
> # Fetch data from MySQL database
> mpg_data <- dbGetQuery(con, "SELECT horsepower, weight, mpg FROM mpg")
>
> # Close the database connection
> dbDisconnect(con)
[1] TRUE
```

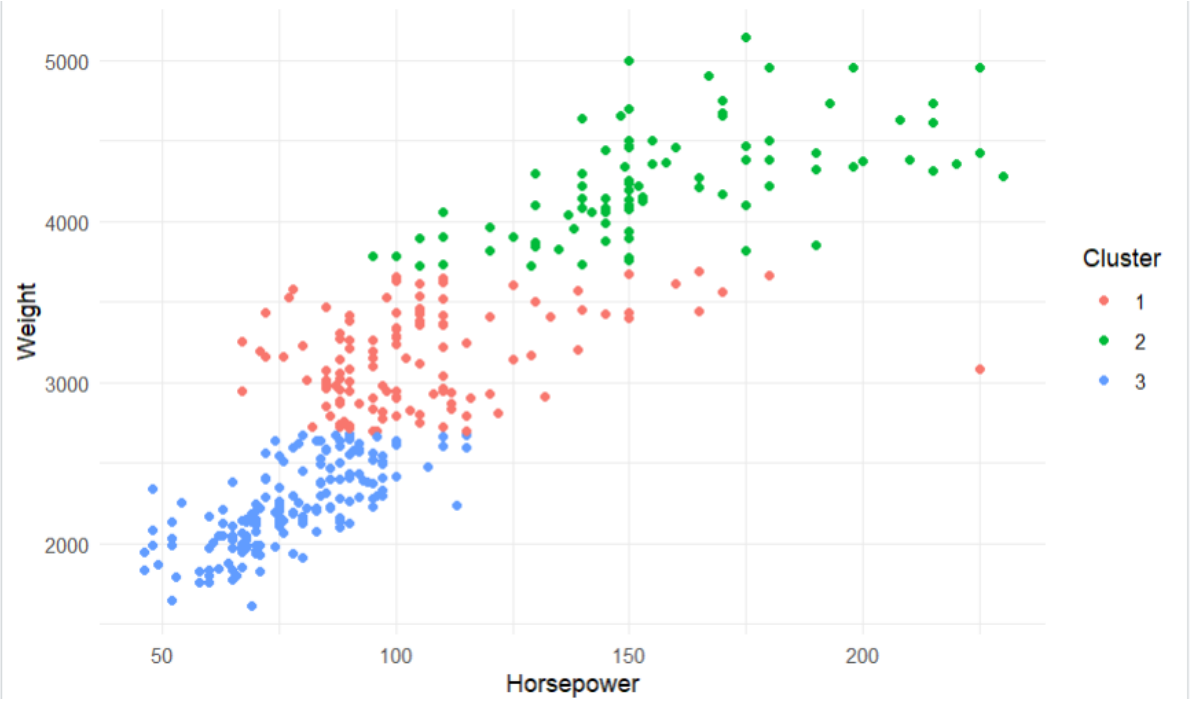
Step # 2 Scatter Plot Visualization and findings: As you will see in the scatter plot below depicting the correlation between miles per gallon (MPG), and weight, and horsepower of the vehicle. The less the vehicle weighs and the less horsepower the vehicle has the more miles per gallon (MPG) the vehicle will get. The more the vehicle weighs and the more horsepower the vehicle has the less miles per gallon (MPG) the vehicle will get. This scatterplot does a great job of proving my hypothesis correct, the more gas efficient vehicle will often be the less powerful in horsepower and lighter in weight.



Step # 3 Now let us cluster the data on this plane in a “reasonable” number of groups. Show your plot where the data points are now colored with the cluster information and provide your interpretations.

```
> # Create scatter plot
> scatter_plot <- ggplot(mpg_data, aes(x = horsepower, y = weight, color = mpg)) +
+     geom_point() +
+     labs(x = "Horsepower", y = "Weight", color = "MPG") +
+     theme_minimal()
>
> # Display the scatter plot
> print(scatter_plot)
> # Perform k-means clustering
> set.seed(123) # for reproducibility
> kmeans_result <- kmeans(mpg_data[, c("horsepower", "weight")], centers = 3)
>
> # Add cluster information to the data
> mpg_data$cluster <- as.factor(kmeans_result$cluster)
>
> # Create scatter plot with cluster information
> scatter_plot_clustered <- ggplot(mpg_data, aes(x = horsepower, y = weight, color = cluster)) +
+     geom_point() +
+     labs(x = "Horsepower", y = "Weight", color = "Cluster") +
+     theme_minimal()
>
> # Display the scatter plot with cluster information
> print(scatter_plot_clustered)
>
>
>
> # Perform k-means clustering
> set.seed(123) # for reproducibility
> kmeans_result <- kmeans(mpg_data[, c("horsepower", "weight")], centers = 3)
>
> # Add cluster information to the data
> mpg_data$cluster <- as.factor(kmeans_result$cluster)
>
> # Create scatter plot with cluster information
> scatter_plot_clustered <- ggplot(mpg_data, aes(x = horsepower, y = weight, color = cluster)) +
+     geom_point() +
+     labs(x = "Horsepower", y = "Weight", color = "Cluster") +
+     theme_minimal()
>
> # Display the scatter plot with cluster information
> print(scatter_plot_clustered)
`
```

Step # 3 Cluster Visualization and findings: As you will see in the cluster visualization below clustered into three groups by (Weight and Horsepower), the first cluster group being in the middle of the weight and horsepower vehicles. The second cluster group being the highest of the weight and horsepower vehicles. The third and final cluster group being at the lowest of the weight and horsepower vehicles. The clustering method makes it easy on the viewer by separating the data points into three cluster for easy comprehension of correlation between MPG, weight, and horsepower.


5000
4000
3000
2000
Weight
50
100
150
200
Horsepower
Cluster
1
2
3