Week 3 Assignment: Normalization and Character Manipulation

Alina Vikhnevich

2025-02-15

Introduction

Normalization and character manipulation are essential for structuring and analyzing data efficiently. Using MySQL in R, I normalized movie rating data into structured tables, ensuring consistency and reducing redundancy. For text analysis, I extracted majors containing "DATA" or "STATISTICS" and used regular expressions to detect word patterns. These techniques are valuable for data cleaning, querying, and preparing unstructured data for deeper analysis.

Task #1: Normalization in R

(Using MySQL Database)

```
# Secure connection to MySQL database
password <- Sys.getenv("MYSQL_PWD")

conn <- tryCatch(
  dbConnect(MySQL(),
            user = "root",
            password = password,
            host = "127.0.0.1",
            dbname = "movies"),
  error = function(e) {
    message("Error: ", e$message)
    return(NULL)
  }
)

if (!is.null(conn)) {
  print("Database connection successful.")
  ratings_df <- dbGetQuery(conn, "SELECT * FROM ratings;")
} else {
  stop("Database connection failed. Check credentials and try again.")
}
```

```
## [1] "Database connection successful."
```

Normalization ensures data is stored in a structured and efficient way. Below are three normalized dataframes:

Users Table

```
users <- dbGetQuery(conn, "SELECT user_id, user_name, user_preference FROM users;")
users
```

```
##    user_id user_name   user_preference
## 1        1     Saqib            Action
## 2        2     Irina Romantic Comedies
## 3        3     Helen      Supernatural
## 4        4  Viktoria             Drama
## 5        5      Vlad           History
## 6        6      Lana           K-Drama
## 7        7     Jason        Conspiracy
## 8        8    Nataly         Nostalgic
## 9        9      Alex       Suspenseful
## 10      10 Michaella          Arthouse
```

Movies Table

```
movies <- dbGetQuery(conn, "SELECT movie_id, movie_title, movie_genre FROM movies;")
movies
```

```
##   movie_id          movie_title movie_genre
## 1        1 Deadpool & Wolverine      Action
## 2        2             Parasite    Thriller
## 3        3       Dune: Part Two      Sci-Fi
## 4        4         Inside Out 2   Animation
## 5        5        The Substance      Horror
## 6        6          Oppenheimer       Drama
```

Ratings Table

```
ratings <- dbGetQuery(conn, "SELECT rating_id, user_id, movie_id, rating FROM ratings_records;")
ratings
```

```
##    rating_id user_id movie_id rating
## 1          1       1        1      5
## 2          2       2        2      5
## 3          3       3        4      4
## 4          4       5        6      5
## 5          5       1        1      5
## 6          6       1        2      3
## 7          7       1        3      4
## 8          8       1        4      2
## 9          9       1        5      3
## 10        10       1        6      5
## 11        11       2        1      3
## 12        12       2        2      5
## 13        13       2        3      2
## 14        14       2        4      5
## 15        15       2        5      4
## 16        16       2        6      3
## 17        17       3        1      2
## 18        18       3        2      5
```

```
## 19        19       3        3       3
## 20        20       3        4       4
## 21        21       3        5       5
## 22        22       3        6       2
## 23        23       5        1       3
## 24        24       5        2       4
## 25        25       5        3       2
## 26        26       5        4       3
## 27        27       5        5       3
## 28        28       5        6       5
## 29        29       6        1       2
## 30        30       6        2       5
## 31        31       6        3       3
## 32        32       6        4       5
## 33        33       6        5       4
## 34        34       6        6       3
## 35        35       7        1       4
## 36        36       7        2       3
## 37        37       7        3       5
## 38        38       7        4       2
## 39        39       7        5       4
## 40        40       7        6       5
## 41        41       1        1       5
## 42        42       1        2       3
## 43        43       1        3       4
## 44        44       1        4       2
## 45        45       1        5       3
## 46        46       1        6       5
## 47        47       2        1       3
## 48        48       2        2       5
## 49        49       2        3       2
## 50        50       2        4       5
## 51        51       2        5       4
## 52        52       2        6       3
## 53        53       3        1       2
## 54        54       3        2       5
## 55        55       3        3       3
## 56        56       3        4       4
## 57        57       3        5       5
## 58        58       3        6       2
## 59        59       5        1       3
## 60        60       5        2       4
## 61        61       5        3       2
## 62        62       5        4       3
## 63        63       5        5       3
## 64        64       5        6       5
## 65        65       6        1       2
## 66        66       6        2       5
## 67        67       6        3       3
## 68        68       6        4       5
## 69        69       6        5       4
## 70        70       6        6       3
## 71        71       7        1       4
## 72        72       7        2       3
```

```
## 73        73       7        3      5
## 74        74       7        4      2
## 75        75       7        5      4
## 76        76       7        6      5
## 77        77       8        1      3
## 78        78       8        2      4
## 79        79       8        3      2
## 80        80       8        4      5
## 81        81       8        5      3
## 82        82       8        6      3
## 83        83       9        1      5
## 84        84       9        2      3
## 85        85       9        3      5
## 86        86       9        4      2
## 87        87       9        5      4
## 88        88       9        6      4
## 89        89      10        1      3
## 90        90      10        2      4
## 91        91      10        3      2
## 92        92      10        4      5
## 93        93      10        5      5
## 94        94      10        6      3
```

This normalized structure prevents data redundancy and update anomalies. Each table represents a single entity and references other tables using foreign keys.

Task #2: Character Manipulation:

College Majors Dataset

We extract majors containing **DATA** or **STATISTICS** from the dataset available at FiveThirtyEight.

```
# Read in the dataset
majors <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/majors

# Find majors containing "DATA" or "STATISTICS"
data_statistics_majors <- majors %>%
  filter(str_detect(Major, "DATA|STATISTICS")) %>%
  select(Major)

data_statistics_majors
```

```
##                                        Major
## 1 MANAGEMENT INFORMATION SYSTEMS AND STATISTICS
## 2      COMPUTER PROGRAMMING AND DATA PROCESSING
## 3               STATISTICS AND DECISION SCIENCE
```

Task #3: Understanding Regular Expressions

```
(.)\1\1
```

This pattern matches any single character repeated three times in a row.

Breakdown:

- `(.)` → Captures any one character.
- `\1` → Refers back to the same captured character.
- `\1` → Again, repeats the captured character.

Examples that match: `"aaa"`, `"111"`, `"ccc"`

```
"(.)(.)\\2\\1"
```

This pattern matches **four-character palindromes** (mirror patterns).

Breakdown:

- `(.)` → Captures the first character.
- `(.)` → Captures the second character.
- `\2` → Must match the second captured character again.
- `\1` → Must match the first captured character again.

Examples that match: `"abba"`, `"1221"`, `"boob"`

```
(..)\1
```

This pattern matches any **two-character sequence repeated twice in a row**.

Breakdown:

- `(..)` → Captures two characters.
- `\1` → Must repeat the same two characters.

Examples that match: `"abab"`, `"1212"`, `"cfcf"`

```
"(.).\\1.\\1"
```

This pattern matches a character appearing at positions 1, 3, and 5.

Breakdown:

- `(.)` → Captures one character.
- `.` → Matches any character (acts as a placeholder).
- `\1` → Must match the first captured character.
- `.` → Matches any character.
- `\1` → Again, must match the first captured character.

Examples that match: `"abaca"`, `"1x1x1"`, `"momom"`

```
"(.)(.)(.).*\\3\\2\\1"
```

This pattern matches a **six-character palindrome** (with any number of characters in between).

Breakdown:

- `(.)` → Captures the first character.
- `(.)` → Captures the second character.
- `(.)` → Captures the third character.
- `.*` → Matches any number of characters in between.
- `\3` → Must match the third captured character.
- `\2` → Must match the second captured character.
- `\1` → Must match the first captured character.

Examples that match: `"xyzzyx"`, `"123xx321"`, `"abcddcba"`

Task #4: Constructing Regular Expressions

Start and end with the same character

```
start_end_same <- "^(.).*\\1$"
```

Contain a repeated pair of letters

```
repeated_pair <- "(..).*\\1"
```

Contain one letter repeated in at least three places

```
letter_repeated_three <- "(.).*\\1.*\\1"
```

Test:

```
# Sample test words
words <- c("level", "noon", "church", "mississippi", "banana", "racecar")

# Test each pattern
matches_1 <- words[str_detect(words, start_end_same)]
matches_2 <- words[str_detect(words, repeated_pair)]
matches_3 <- words[str_detect(words, letter_repeated_three)]

# Print results
list(
  "Start and end with the same character" = matches_1,
  "Contains a repeated pair of letters" = matches_2,
  "Contains a letter repeated 3 times" = matches_3
)
```

```
## $`Start and end with the same character`
## [1] "level"   "noon"    "racecar"
##
## $`Contains a repeated pair of letters`
## [1] "church"      "mississippi" "banana"
##
## $`Contains a letter repeated 3 times`
## [1] "mississippi" "banana"
```

Conclusion

This assignment focused on normalization, text filtering, and regular expressions to clean and structure data effectively. Using SQL in R, we normalized datasets to improve consistency and avoid redundancy. Character manipulation techniques helped extract relevant text data, while regex patterns allowed complex string analysis. These skills are crucial for efficient data processing and analysis.