SmartStudy: Tailored Learning for Every Student

Introduction

It is my utmost pleasure to present to you the introduction to my capstone project for the Coursera Google Data Analytics program, Track 3. Throughout this journey, I have had the opportunity to apply my acquired knowledge and skills in data analytics to work on a hypothetical dataset of my choice. In this repository, I am sharing the highlights of my capstone project and the exciting insights I have discovered.

For my capstone project, I wanted to delve into a dataset that not only piqued my interest but also aligned with my passion for education domain. With this in mind, I selected a hypothetical dataset that focuses on tracking and analyzing student learning dataset to ensure the implementation of "SmartStudy: Tailored Learning for Every Student.".

The scenario

The Honey Bee School of Excellence is a leading educational institution known for its commitment to providing quality education and fostering the holistic development of its students. As part of their ongoing efforts to enhance the learning experience, the school administration is considering the implementation of "SmartStudy: Tailored Learning for Every Student." This innovative program aims to support students' learning journey through after-school work completion tracking and personalized assistance.

To assess the potential impact and effectiveness of the SmartStudy program, the school administration has decided to conduct an exploratory data analysis (EDA) on student performance and work completion data. The objective is to uncover insights and draw conclusions that can guide the decision-making process.

The school administration has provided a dataset containing information about student marks, work completion records, demographic details, and other relevant factors. The dataset is comprehensive, capturing data from multiple grade levels and diverse student backgrounds.

To begin the EDA process, the school administration assigns a team of data analysts and education experts. They start by importing the dataset and conducting data cleaning procedures, such as handling missing values and ensuring data consistency.

Once the data is prepared, the team begins exploring various aspects of student performance and work completion. They calculate summary statistics, visualize distributions, and examine potential relationships between variables. The team analyzes factors such as average marks, work completion rates, grade levels, available devices, and demographic characteristics.

During the analysis, the team uncovers interesting insights. They find that students with higher work completion rates tend to have better overall marks. Additionally, they observe variations in work completion rates across different grade levels and available devices. Certain demographic factors, such as socioeconomic status, might also influence work completion patterns.

As the EDA progresses, the team segments the data to further investigate performance and work completion trends within specific student groups. They examine the differences between grade levels, subject areas, and student demographics. The team also considers the impact of work completion on student motivation and engagement.

Code and Visualizations Using R Programming

Load packages

Start by installing your required package. If you have already installed and loaded `tidyverse` in this session, feel free to skip the code chunks in this step.

```
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

Once a package is installed, we can load it by running the `library()` function with the package name inside the parentheses:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(ggplot2)
library(dplyr)
```

Data Import

```
data <- read.csv("student_learning_data.csv")
```

This code chunk imports the student learning data from a CSV file using the read.csv function. The file path to the CSV file is specified in the read.csv function. Make sure to provide the correct path to your student_learning_data.csv file in order to successfully import the data.

View the First Rows of the Data

```
head(data)
```

```
##   student_id gender grade parent_education    lunch available_device
## 1     GR8001 female     8  bachelor degree standard           laptop
## 2     GR9001 female     9   primary school standard   android_device
## 3     GR8002 female     8    master degree standard           laptop
## 4     GR7001 female     7    master degree standard           laptop
## 5     GR9002 female     9    middle school standard           laptop
## 6     GR8003 female     8          diploma standard           laptop
##   work_completion math science social average   Tier
## 1      incomplete   72      72     74      73 Tire 1
## 2       completed   69      90     88      82 Tire 1
## 3      incomplete   90      95     93      93 Tire 1
## 4      incomplete   50      53     58      54 Tire 2
## 5      incomplete   69      75     78      74 Tire 1
## 6      incomplete   71      83     78      77 Tire 1
```

In this code chunk, the head function is used to display the first few rows of the data dataframe. The head function allows you to quickly preview the structure and contents of the dataset.

By calling head(data), the R console will show the top rows of the dataframe. The number of rows displayed by default is usually 6, but you can modify this by passing an optional parameter to the head function, specifying the desired number of rows to be shown.

This code chunk is helpful for inspecting the data and gaining a quick understanding of its columns, values, and overall structure.

Create Average of Students Tiers

```
data$average_tier <- cut(data$average, breaks = c(-Inf, 40, 70, Inf), labels = c("Tier 3", "Tier 2", "T
```

In this code chunk, the cut function is used to create average tiers based on the values in the 'average' column of the data. The cut function divides the data into three tiers using the specified breaks and labels.

Count Students by Grade and Tier

```
grades <- c(7, 8, 9, 10)

tier_counts <- data.frame(Grade = grades,
                          Tier1 = rep(0, length(grades)),
                          Tier2 = rep(0, length(grades)),
                          Tier3 = rep(0, length(grades)))

for (i in 1:length(grades)) {
  grade_data <- subset(data, grade == grades[i])
  grade_counts <- table(grade_data$average_tier)
  tier_counts[i, c("Tier1", "Tier2", "Tier3")] <- grade_counts
}

rownames(tier_counts)
```

```
## [1] "1" "2" "3" "4"
```

This code chunk counts the number of students in each grade and tier category. It initializes a data frame called tier_counts with columns for grades and each tier category. Then, using a for loop, it subsets the data for each grade and counts the occurrences of each tier using the table function. The counts are stored in the tier_counts data frame.

View Tier Counts

```
tier_counts
```

```
##   Grade Tier1 Tier2 Tier3
## 1     7     6   116   107
## 2     8     8   111    71
## 3     9    10   177   132
## 4    10     8   125   129
```

This code chunk displays the tier_counts data frame in the R console, showing the counts of students in each grade and tier category.

Reshape Data for Visualization

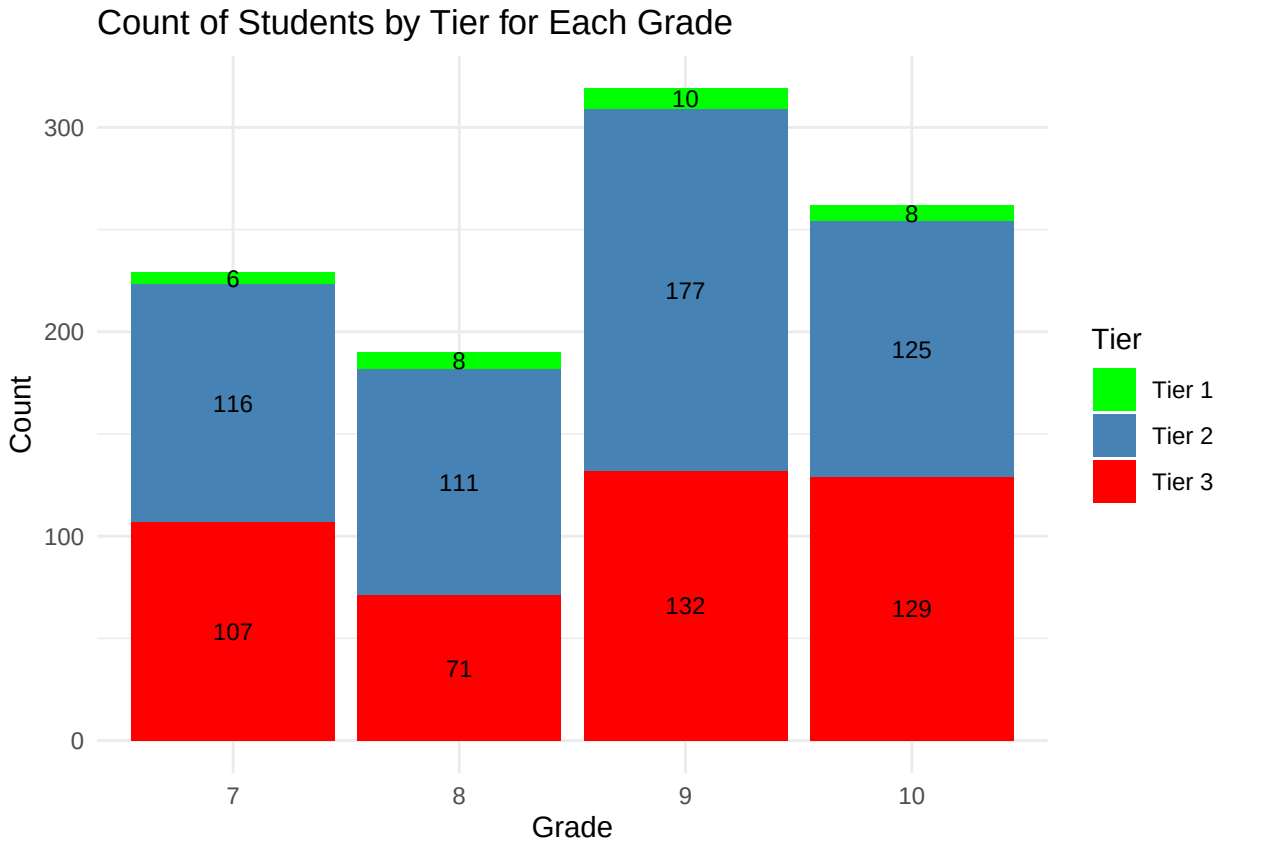
```
tier_counts_long <- tidyr::pivot_longer(tier_counts, cols = starts_with("Tier"), names_to = "Tier", valu
```

The pivot_longer function from the tidyr package is used in this code chunk to reshape the tier_counts data frame from wide format to long format. It converts the columns starting with "Tier" into a single column called "Tier" and the corresponding counts into a column called "Count".

Visualize Student Tiers by Grade

```
plot <- ggplot(tier_counts_long, aes(x = factor(Grade), y = Count, fill = Tier)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = Count), position = position_stack(vjust = 0.5), color = "black", size = 3) +
  labs(x = "Grade", y = "Count", title = "Count of Students by Tier for Each Grade") +
  scale_fill_manual(values = c("Tier1" = "green", "Tier2" = "steelblue", "Tier3" = "red"),
                    labels = c("Tier 1", "Tier 2", "Tier 3")) +
  theme_minimal()

print(plot)
```

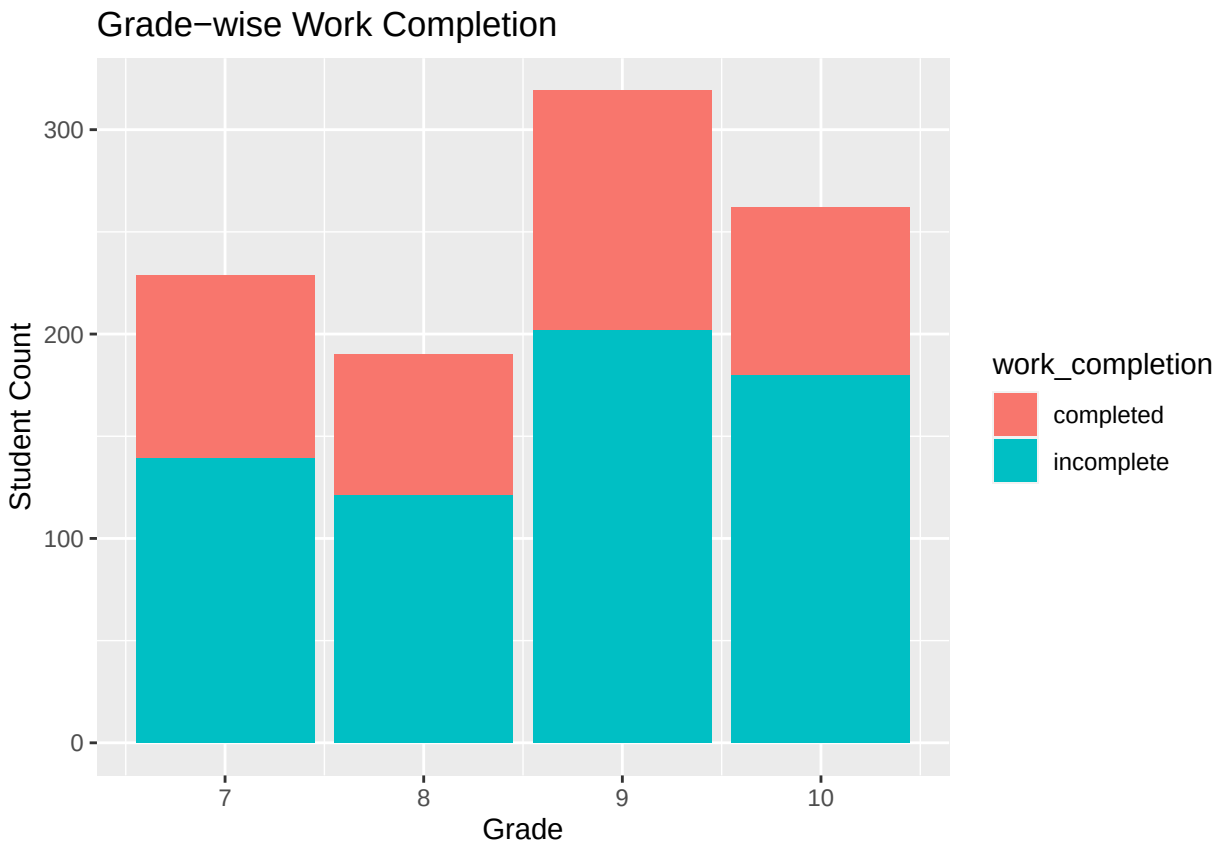


This code chunk creates a bar plot using ggplot2 to visualize the count of students in each tier for each grade. It uses the ggplot function and various geom_ layers to construct the plot, including bars and text labels. The plot is customized with labels, colors, and themes.

Filter Data and Visualize Work Completion by Grade

```
filtered_data <- data %>%
  filter(!is.na(grade), !is.na(work_completion))  # Filter out non-finite values

ggplot(filtered_data, aes(x = grade, fill = work_completion)) +
  geom_bar() +
  labs(x = "Grade", y = "Student Count") +
  ggtitle("Grade-wise Work Completion")
```



In this code chunk, the filter function from the dplyr package is used to remove rows with missing values for the grade and work_completion variables. The !is.na(grade) and !is.na(work_completion) conditions ensure that only rows without any missing values in these two variables are kept in the filtered_data dataframe.

The ggplot function is then used to create a bar plot to visualize the work completion of students in each grade. The x aesthetic is set to the grade variable, and the fill aesthetic is set to the work_completion variable. This results in bars colored based on the different work completion categories. The geom_bar function is used to display the bars.

Additionally, the labs function is used to provide labels for the x-axis and y-axis, specifying "Grade" and "Student Count" respectively. The ggtitle function is used to add a title to the plot, which is set as "Grade-wise Work Completion".

This code chunk allows you to filter out rows with missing values and visualize the work completion of students in each grade using a bar plot.

Count Work Completion by Tier

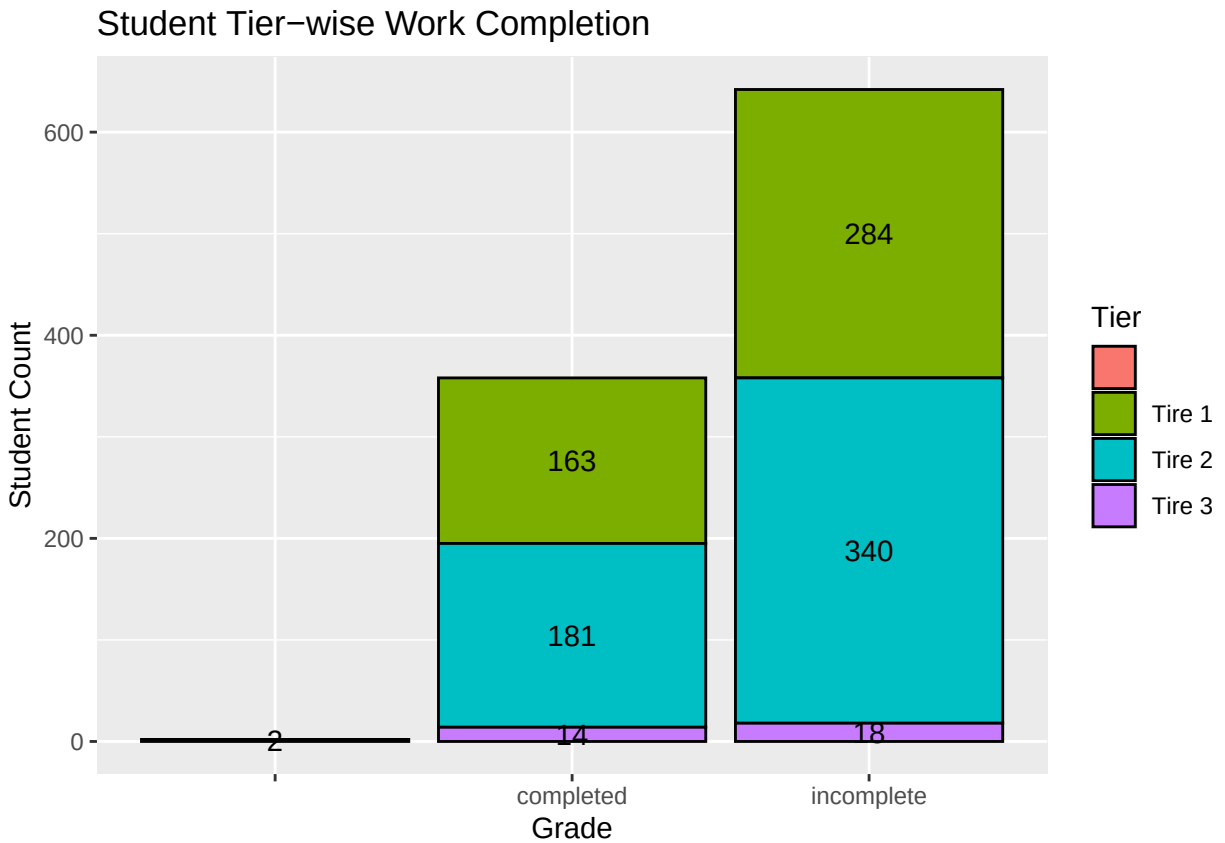
```
count_labels <- data %>%
  group_by(work_completion, Tier) %>%
  summarise(Count = n())
```

```
## `summarise()` has grouped output by 'work_completion'. You can override using
## the `.groups` argument.
```

This code chunk calculates the count of students for each work completion status and tier category. It uses the group_by and summarise functions from dplyr to group the data by work completion and tier, and then calculates the count of students using the n function.

##Visualize Work Completion by Tier

```
ggplot(data, aes(x = work_completion, fill = Tier)) +
  geom_bar(color = "black") +
  geom_text(data = count_labels, aes(label = Count, y = Count), position = position_stack(vjust = 0.5),
    labs(x = "Grade", y = "Student Count") +
  ggtitle("Student Tier-wise Work Completion")
```



This code chunk creates a bar plot to visualize the work completion of students in each tier. It uses ggplot2 and includes additional geom_text layer to display the count labels on top of the bars.

Count Available Devices by Grade

```
count_labels <- data %>%
  group_by(grade, available_device) %>%
  summarise(Count = n())
```

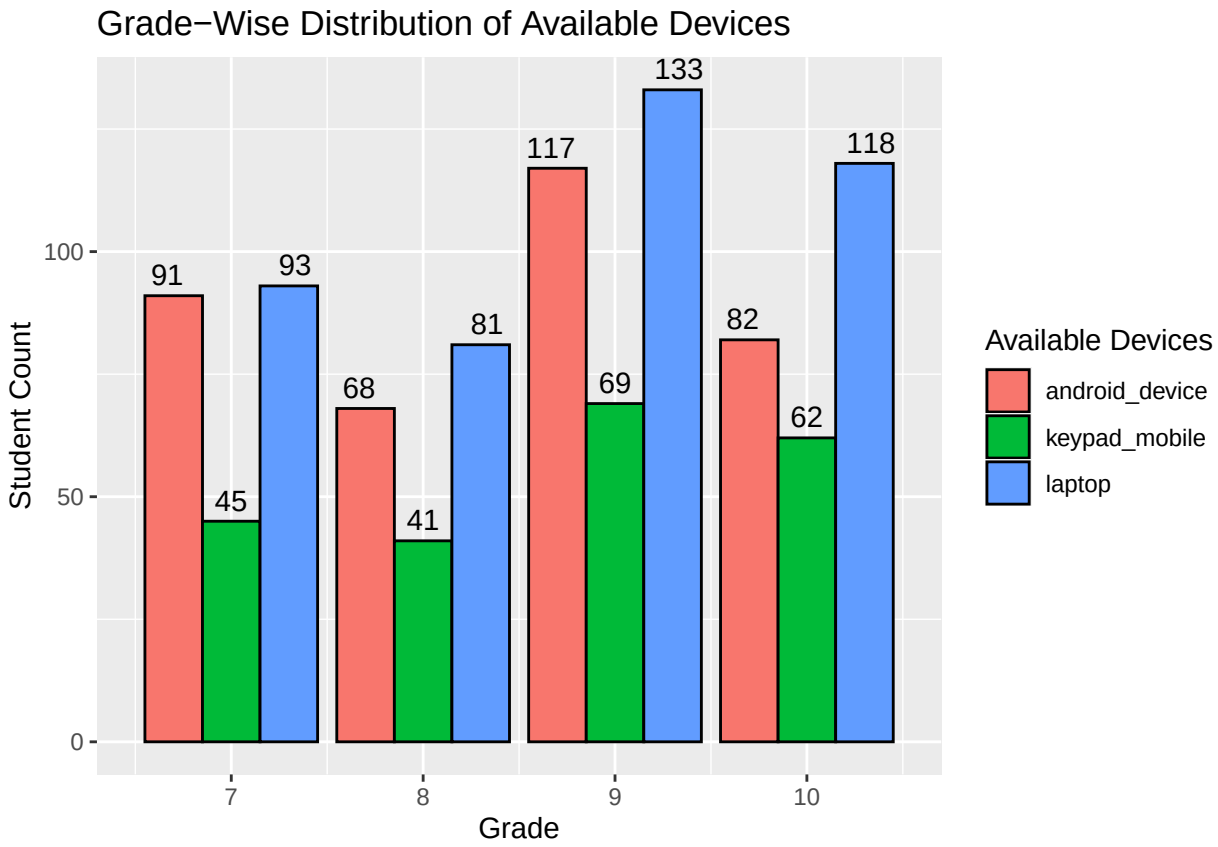
```
## `summarise()` has grouped output by 'grade'. You can override using the
## `.groups` argument.
```

In this code chunk, the count of students for each grade and available device category is calculated. It uses group_by and summarise functions to group the data by grade and available device, and then calculates the count of students using the n function.

##Filter Data and Visualize Distribution of Available Devices by Grade

```
data_filtered <- data[complete.cases(data[, c("grade", "available_device")]), ]
count_labels_filtered <- count_labels[complete.cases(count_labels), ]

ggplot(data_filtered, aes(x = grade, fill = available_device)) +
  geom_bar(position = "dodge", color = "black") +
  geom_text(data = count_labels_filtered, aes(label = Count, y = Count), position = position_dodge(widt
  labs(x = "Grade", y = "Student Count", fill = "Available Devices") +
  ggtitle("Grade-Wise Distribution of Available Devices")
```



In this code chunk, the complete.cases function is used to filter out rows in the data dataframe that have missing values in the "grade" and "available_device" columns. The resulting filtered data is stored in the data_filtered dataframe.

Similarly, the complete.cases function is used to filter out rows in the count_labels dataframe that have missing values. The resulting filtered data is stored in the count_labels_filtered dataframe.

The ggplot function is then used to create a grouped bar plot to visualize the distribution of available devices by grade. The x aesthetic is set to the "grade" variable, and the fill aesthetic is set to the "available_device" variable. This results in grouped bars where each group represents a different available device category for each grade. The geom_bar function is used to display the bars.

Furthermore, the geom_text function is used to add text labels to the plot. The data argument is set to

count_labels_filtered, which contains the count values for each combination of grade and available device. The aes argument is used to map the "Count" variable to the label and y aesthetics. The position_dodge function is used to position the text labels next to the corresponding bars. The vjust argument is set to -0.5 to adjust the vertical position of the labels.

The labs function is used to provide labels for the x-axis, y-axis, and fill legend, specifying "Grade", "Student Count", and "Available Devices" respectively. The ggtitle function is used to add a title to the plot, which is set as "Grade-Wise Distribution of Available Devices".

This code chunk filters the data to exclude rows with missing values and creates a grouped bar plot to visualize the distribution of available devices by grade, along with text labels displaying the counts for each category.

Insights & Recommendations

Grade-wise Distribution of Student Tiers:

1.The graph illustrates the distribution of student tiers (Tier 1, Tier 2, Tier3) across grades.

2.Grade 9 had the highest number of students in all tiers, indicating the need for targeted interventions to support learning in that grade.

3.The majority of students fell into Tier 2, indicating an average level of performance across all grades.

4.SmartStudy's personalized learning paths can help uplift the performance of students in Tier 3 and provide further enrichment for students in Tier 2.

Grade-wise Work Completion:

1.The bar chart showcases the work completion status (completed, incomplete) for each grade.

2.Incomplete work submissions outweighed completed work submissions across all grades.

3.Emphasizing the importance of timely and complete work submission to all students can improve overall work completion rates.

Student Tier-wise Work Completion:

1.The scatter plot presents the work completion status (completed, incomplete) based on student tiers (Tier 1, Tier 2, Tier 3).

2.The graph reveals that completed work was higher in Tier 1 and Tier 2 compared to Tier 3.

3.Encouraging and motivating students in Tier 3 to complete their assignments can bridge the gap and improve work completion rates.

Grade-wise Distribution of Available Devices:

1.The visualization depicts the distribution of available devices (keypad mobile, Android device, laptop) across grades.

2.Grade 9 had the highest availability of devices, while other grades showed variations in device access.

3.Ensuring equitable access to devices, especially in grades with lower availability, can facilitate students' access to SmartStudy's tailored learning resources.

Conclusion

Based on the findings from the EDA, the team concludes that implementing the SmartStudy program could greatly benefit the students at Honey Bee School of Excellence. By leveraging after-school work completion

tracking and personalized support, the program can address individual learning needs, improve student performance, and enhance overall academic outcomes.

The team proposes a tailored smart study platform that combines elements of gamification, personalized study plans, and interactive learning modules. The platform would provide students with real-time feedback on their work completion, suggest targeted study materials, and offer additional support based on their individual progress and learning preferences.

With the EDA results and proposed recommendations, the Honey Bee School of Excellence's administration is now equipped to make an informed decision about implementing the SmartStudy program. They can present the insights and benefits to the school board and stakeholders, demonstrating how the program aligns with the school's mission of providing a nurturing and personalized learning environment.

By integrating technology and data-driven approaches, the Honey Bee School of Excellence aims to empower every student on their educational journey and foster a culture of continuous improvement and academic excellence.