

Give me R code to read data from the following database:  
SQL database: pgAdmin4  
Database Name: DATA607  
Table Name: rating\_final

Sure! Since your database is **PostgreSQL** (pgAdmin4 is just the GUI), you can connect from R using the `RPostgres` or `RPostgreSQL` package. Here's a clean example using `RPostgres`, which is more modern:

```
r

# Install the package if not already installed
# install.packages("RPostgres")
# install.packages("DBI")

library(DBI)
library(RPostgres)

# Database connection details
host <- "localhost"          # or your server IP
port <- 5432                  # default PostgreSQL port
dbname <- "DATA607"
user <- "your_username"      # replace with your username
password <- "your_password"  # replace with your password
```

```
# Connect to the database
con <- dbConnect(
  RPostgres::Postgres(),
  dbname = dbname,
  host = host,
  port = port,
  user = user,
  password = password
)

# Read the table into R
rating_data <- dbReadTable(con, "rating_final")

# View the first few rows
head(rating_data)

# Disconnect when done
dbDisconnect(con)
```

Copy code

### Notes:

- Replace "your\_username" and "your\_password" with your PostgreSQL credentials.
- If you only want a **subset or a custom query**, you can use dbGetQuery :

r

```
rating_data <- dbGetQuery(con, "SELECT * FROM rating_final LIMIT 10;")
```

Copy code

I can also write a version using `tidyverse` syntax for easier data handling if you want.

Do you want me to do that?



I ran the following R code:

```
{r}  
library(tidyverse)  
glimpse(df)
```

The output is as follows:

```
Rows: 7  
Columns: 11  
$ Email <chr>  
"feva0706@gmail.com", "jahidneel10@gmail.com", "...  
$ Name <chr> "Foizunnesa  
Eva", "Jahid Hasan", "Mahmudul Hasan...  
$ Age <int> 33, 23, 36, 32,  
40, 49, 39  
$ Gender <chr> "Female",  
"Male", "Male", "Male", "Male", "Male"...  
$ Timestamp <chr> "2/4/2026  
20:12", "2/4/2026 21:37", "2/5/2026 10...  
$ Superman..2025..Rating <chr> "3.5",  
"4.5", NA, NA, "3.6", NA, NA  
$ F1..The.Movie..2025..Rating <chr> "4.2",  
"4", NA, "4.9", NA, "3.5", NA  
$ Mission..Impossible.The.Final.Reckoning..2025..Rating  
<chr> "4.5", "4.1", "4.8", "4", "4.8", "3.7", NA  
$ Jurassic.World..Rebirth..2025..Rating <chr> "3",
```

```
"3", "3.5", NA, NA, NA, NA  
$ Sinners..2025..Rating <chr> NA, NA,  
NA, "4.5", "4", "4.1", NA  
$ Zootopia.2..2025..Rating <chr> NA,  
"4.3", NA, "3.9", NA, NA, "4.6"
```

What is your suggestion to handle the missing values in this dataset?

Thanks for sharing the `glimpse()` output. I can see a few things:

- Most of your **rating columns** are stored as **character**, not numeric.
- There are **NA values scattered across rating columns**, which is common in survey/movie rating data.

Here's a concise approach to handle the missing values:

---

## 1 Convert rating columns to numeric

Right now, they are `<chr>`. You can convert all rating columns at once:

```
r  
  
# Identify rating columns  
rating_cols <- names(df)[6:11]  
  
# Convert to numeric
```

```
df <- df %>%
  mutate(across(all_of(rating_cols), as.numeric))
```

---

## 2 Decide on a missing value strategy

Common strategies:

a) Keep NAs

- If you are doing **analysis that can handle NAs** (like `mean(..., na.rm = TRUE)` ), you can leave them as is.
- Pros: preserves original data.
- Cons: some operations like modeling may need complete cases.

b) Remove rows with all ratings NA

- If someone hasn't rated any movie, that row is not useful:

r

```
df <- df %>%
  filter(if_any(all_of(rating_cols), ~ !is.na(.)))
```

Copy code

c) Impute missing values

- **Simple imputation:** Replace NA with **column mean or median**:

r

```
df <- df %>%
  mutate(across(all_of(rating_cols), ~ ifelse(is.na(.), mean(. , na.rm = TRUE) )
```

Copy code

- **Advanced imputation:** Use **KNN** or **mice** packages for predictive imputation if your analysis requires it.
- 

### 3 Optional: Create a summary

You can quickly see missing values per column:

```
r  
  
df %>%  
  summarise(across(all_of(rating_cols), ~ sum(is.na(.))))
```

 Copy code

---

#### My recommendation:

- **Keep NAs for analysis** like averages or distribution plots ( `ggplot` can handle NAs).
  - **Only impute if you need a complete dataset** for modeling or correlation analysis.
  - **Always convert ratings to numeric first**, otherwise calculations will fail.
- 

If you want, I can write a **ready-to-run R snippet** that cleans this dataset, converts ratings to numeric, and either imputes missing values or filters them depending on your goal.

Do you want me to do that?



Give me R code to visualize the ratings of each movie. A horizontal bar chart should be good

Perfect! Since you have multiple movies and ratings in **wide format**, the best approach is to **reshape the data into long format** and then plot a horizontal bar chart using `ggplot2`. Here's a ready-to-run R snippet:

```
r

library(tidyverse)

# 1. Convert rating columns to numeric (if not done already)
rating_cols <- names(df)[6:11]

df <- df %>%
  mutate(across(all_of(rating_cols), as.numeric))

# 2. Reshape data to long format
df_long <- df %>%
  pivot_longer(
    cols = all_of(rating_cols),
    names_to = "Movie",
    values_to = "Rating"
  )

# 3. Optional: clean movie names for better display
```

```
df_long <- df_long %>%  
  mutate(Movie = str_replace_all(Movie, "\\\.\.\.2025\\\.\.\.Rating", "")) %>%  
  mutate(Movie = str_replace_all(Movie, "\\.\\.", " "))  
  
# 4. Plot horizontal bar chart (average rating per movie)  
df_long %>%  
  group_by(Movie) %>%  
  summarise(Average_Rating = mean(Rating, na.rm = TRUE)) %>%  
  ggplot(aes(x = Average_Rating, y = fct_reorder(Movie, Average_Rating))) +  
  geom_col(fill = "steelblue") +  
  labs(  
    title = "Average Ratings of Movies",  
    x = "Average Rating",  
    y = "Movie"  
  ) +  
  theme_minimal()
```

 Copy code

---

## Explanation:

1. `pivot_longer()` converts wide format to long format for plotting.
  2. `str_replace_all()` cleans up the messy movie column names.
  3. `fct_reorder()` orders bars from lowest to highest rating.
  4. `geom_col()` creates horizontal bars.
- 

If you want, I can also make a **version showing all individual ratings as dots over the bars**, so you can see the distribution of ratings per movie—not just the average.

Do you want me to do that?



Add different color to the bars and anotate

Got it! We can modify the previous code to:

- Assign **different colors** to each movie.
- **Annotate** each bar with the average rating value.

Here's the updated R code:

```
r

library(tidyverse)

# 1. Convert rating columns to numeric
rating_cols <- names(df)[6:11]

df <- df %>%
  mutate(across(all_of(rating_cols), as.numeric))

# 2. Reshape data to long format
df_long <- df %>%
  pivot_longer(
    cols = all_of(rating_cols),
    names_to = "Movie",
```

ChatGPT ▾  Free offer × Memory full  Share ...

```
mutate(Movie = str_replace_all(Movie, "\.\.\.Z0Z0\.\.\.Rating", " \n\n"))  
mutate(Movie = str_replace_all(Movie, "\.", " "))  
  
# 3. Calculate average ratings per movie  
avg_ratings <- df_long %>%  
  group_by(Movie) %>%  
  summarise(Average_Rating = mean(Rating, na.rm = TRUE))  
  
# 4. Plot horizontal bar chart with different colors and annotation  
ggplot(avg_ratings, aes(x = Average_Rating, y = fct_reorder(Movie, Average_Rating  
  geom_col() +  
  geom_text(aes(label = round(Average_Rating, 2)),  
    hjust = -0.1, # position outside the bar  
    size = 4) +  
  scale_fill_brewer(palette = "Set3") + # nice color palette  
  labs(  
    title = "Average Ratings of Movies",  
    x = "Average Rating",  
    y = "Movie"  
  ) +  
  theme_minimal() +  
  theme(legend.position = "none") + # remove Legend as colors already indicate mo  
  xlim(0, max(avg_ratings$Average_Rating) + 1) # adjust x-axis for labels
```

 Copy code

## ✓ Features:

- Different colors for each movie using `Set3` palette.
  - Annotations show the exact average rating on each bar.
  - Horizontal bars ordered by rating.
- 

If you want, I can also make a **version with a gradient color based on the rating value**, so higher-rated movies are visually darker/brighter. This often looks more intuitive.

Do you want me to do that too?

