**#THE INNER JOIN VERB**

-Inner Join :

**\*sets%>%**

**\*inner\_join(themes, by = c(“theme\_id” = “id”))**

-Customizing Join :

**\*sets%>%**

**\*inner\_join(themes, by = c(“theme\_id” = “id”), suffix = c(“\_set”, “\_theme”))**

-Melihat banyaknya tema yg digunakan

**\*sets%>%**

**\*inner\_join(themes, by = c(“theme\_id” = “id”), suffix = c(“\_set”, “\_theme”)) %>%**

**\*count(name\_theme, sort = TRUE)**

-Joining Part and Part\_categories

# Use the suffix argument to replace .x and .y suffixes

parts %>%

    inner\_join(part\_categories, by = c("part\_cat\_id" = "id"), suffix = c("\_part", "\_category"))

-Joining if the column has same name :

**\*sets %>%**

**\*inner\_join (inventories, by = “set\_num”)**

-Filter Joining table :

**\*sets %>%**

**\*inner\_join (inventories, by = “set\_num”)%>%**

**\*filter(version == 1)**

-Joining Parts and Inventory

# Combine the parts and inventory\_parts tables

parts %>%

inner\_join(inventory\_parts, "part\_num")

-Join more than 2 table :

**\*sets %>%**

**\*inner\_join(inventories, by = “set\_num”)%>%**

**\*inner\_join(themes, by = c(“theme\_id” = “id”), suffix = c(“\_set”, “\_theme”))**

-Joining 3 tables :

sets %>%

    # Add inventories using an inner join

    inner\_join(inventories, "set\_num") %>%

    # Add inventory\_parts using an inner join

    inner\_join(inventory\_parts, c("id"="inventory\_id"))

-Joining 4 Tables :

# Count the number of colors and sort

sets %>%

    inner\_join(inventories, by = "set\_num") %>%

    inner\_join(inventory\_parts, by = c("id" = "inventory\_id")) %>%

    inner\_join(colors, by = c("color\_id" = "id"), suffix = c("\_set", "\_color")) %>%

count(name\_color, sort=TRUE)

**#LEFT AND RIGHT JOIN**

-Combine join with dplyr :

**\*inventory\_parts\_joined <- inventories %>>**

**\*inner\_join (inventory\_parts, by = c(“id” = “inventory\_id”)) %>%**

**\*select(-id, -version) %>%**

**\*count(desc(quantitiy))**

-Joining with multiple column :

**\*batmobile %>%  
\*inner\_join (batwing, by = c(“part\_num”, “color\_id”), suffix = c(“\_batmobile”, “\_batwing”))**

-Left Join two set by part and color

# Combine the star\_destroyer and millennium\_falcon tables

millennium\_falcon %>%

left\_join(star\_destroyer, by = c("part\_num", "color\_id"), suffix = c("\_falcon", "\_star\_destroyer"))

-Left Joining two sets by color

# Aggregate Millennium Falcon for the total quantity in each part

millennium\_falcon\_colors <- millennium\_falcon %>%

  group\_by(color\_id) %>%

  summarize(total\_quantity = sum(quantity))

# Aggregate Star Destroyer for the total quantity in each part

star\_destroyer\_colors <- star\_destroyer %>%

  group\_by(color\_id) %>%

  summarize(total\_quantity = sum(quantity))

# Left join the Millennium Falcon colors to the Star Destroyer colors

millennium\_falcon\_colors %>%

  left\_join(star\_destroyer\_colors, by = "color\_id", suffix = c("\_falcon", "\_star\_destroyer"))

-Finding an observation that doesn’t have a match

inventory\_version\_1 <- inventories %>%

  filter(version == 1)

# Join versions to sets

sets %>%

  left\_join(inventory\_version\_1, by = "set\_num") %>%

  # Filter for where version is na

  filter(is.na(version))

-Right Join :

**\*batmobile%>%**

**\*right\_join(batwing, by = c(“part\_num”, “color\_id”), suffix = c(“\_batmobile”, “\_batwing”))**

-Replace NA from right join :

**\*sets %>%**

**\*count(theme\_id, sort = TRUE) %>%**

**\*right\_join(themes, by = c(“theme\_id” =”id”)) %>%**

**\*replace\_na(list(n=0)) -> replace column n with 0**

\*Counting Part color :

parts %>%

    count(part\_cat\_id) %>%

    right\_join(part\_categories, by = c("part\_cat\_id" = "id")) %>%

    # Filter for NA

    filter(is.na(n))

\*Cleaning up your count :

parts %>%

    count(part\_cat\_id) %>%

    right\_join(part\_categories, by = c("part\_cat\_id" = "id")) %>%

    # Use replace\_na to replace missing values in the n column

    replace\_na(list(n=0))

-Join with themeself

**\*themes %>%  
\*inner\_join (themes, by = c(“parent\_id” = “id”), suffix = c(“\_child”, “\_parent”)) %>%**

**\*filter (name\_child == “ The Lord of the Rings”) -> filter to know the parent name**

**\*filter (name\_parent == “The Lord of the Rings”) -> Filter to know child name**

-Joining themes to their children :

themes %>%

    # Inner join the themes table

    inner\_join(themes, by = c("id"="parent\_id"), suffix=c("\_parent","\_child"))%>%

    # Filter for the "Harry Potter" parent name

    filter(name\_parent=="Harry Potter")

-Joining themes to their grand children :

# Join themes to itself again to find the grandchild relationships

themes %>%

  inner\_join(themes, by = c("id" = "parent\_id"), suffix = c("\_parent", "\_child"))%>%

  inner\_join(themes, by = c("id\_child" = "parent\_id"), suffix = c("\_parent","\_grandchild"))

-Left Joining table to itself :

themes %>%

  # Left join the themes table to its own children

  left\_join(themes, by = c("id" = "parent\_id"), suffix = c("\_parent", "\_child"))%>%

  # Filter for themes that have no child themes

  filter(is.na(id\_child))

**#THE FULL JOIN VERB**

-Full Join : Kepp all column whether it matched or not

**\*library(tidyr)**

**\*batmobile %>%**

**\*full\_join (batwing, by = c(“part\_num”, “color\_id”), suffix = c(“batmobile”, “batwing”)) %>%**

**\*replace\_na(list(quantity\_batmobile =0, quantity\_batwing = 0))**

-Differences between batman and star wars

# Start with inventory\_parts\_joined table

inventory\_parts\_joined %>%

  # Combine with the sets table

inner\_join(sets, by = "set\_num")%>%

  # Combine with the themes table

inner\_join(themes, by=c("theme\_id"="id"), suffix=c("\_set","\_theme"))

-Aggregating each theme

# Count the part number and color id, weight by quantity

batman %>%

count(part\_num, color\_id, wt = quantity)

star\_wars %>%

count(part\_num, color\_id, wt = quantity)

-Full Joining Batman and star wars lego parts

batman\_parts %>%

  # Combine the star\_wars\_parts table

  full\_join(star\_wars\_parts, by = c("part\_num", "color\_id"), suffix = c("\_batman", "\_star\_wars")) %>%

  # Replace NAs with 0s in the n\_batman and n\_star\_wars columns

  replace\_na(list(n\_batman = 0, n\_star\_wars = 0))

\*disini joinnya dengan menggabungkan 2 kolom dalam satu table, yaitu part\_num dan color\_id

- Comparing Batman and Star Wars Lego Parts

parts\_joined %>%

  # Sort the number of star wars pieces in descending order

arrange(desc(n\_star\_wars)) %>%

  # Join the colors table to the parts\_joined table

inner\_join(colors, by = c("color\_id" = "id")) %>%

  # Join the parts table to the previous join

inner\_join(parts, by = "part\_num", suffix = c("\_color", "\_part"))

-Filtering Join :

1.Semi\_Join : akan memfilter pada 2 table hanya yg memiliki kesamaan

2.Anti\_Join : akan memfilter yg ada di table pertama tapi tidak ada di kolom ke-2

-The Semi\_Join

**\*batmobile %>%**

**\*semi\_join (batwing, by = c(“color\_id”, “part\_num”)**

\*the result of column in semi join will be result only batmobile column

-The Anti Join

**\*batmobile %>%**

**\*anti\_join (batwing, by = c(“color\_id”, “part\_num”))**

\*Result in table 1 that not match in table 2

-Something within one set but not another

# Filter the batwing set for parts that are also in the batmobile set

batwing %>%

semi\_join(batmobile, by="part\_num")

# Filter the batwing set for parts that aren't in the batmobile set

batwing %>%

anti\_join(batmobile, by="part\_num")

-What colors are included in at least one set

# Use inventory\_parts to find colors included in at least one set

colors %>%

semi\_join(inventory\_parts, by=c("id" = "color\_id"))

-Which set is missing in version 1

# Use filter() to extract version 1

version\_1\_inventories <- inventories %>%

filter(version==1)

# Use anti\_join() to find which set is missing a version 1

sets %>%

anti\_join(version\_1\_inventories, by="set\_num")

-Visualizing set difference :

-Adding Perecentage :

**\*colors\_joined <- batmobile\_colors %>%**

**\*full\_join (batwing\_colors, by = “color\_id”, suffix = c(“\_batmobile”, “\_batwing”))%>%  
\*replace\_na(list(total\_batmobile = 0, total\_batwing = 0)) %>%**

**\*inner\_join(colors, by = c(“color\_id”, “id”) %>%  
\*mutate(total\_batmobile = total\_batmobile/sum(total\_batmobile),**

**total\_batwing = total\_batwing/sum(total\_batwing),**

**difference = total\_batmobile – total\_batwing)**

-Visualizing The Data :

**\*library(ggplot2)**

**\*library(forcats)**

**\*color\_palette <- setNames(colors\_joined$rgb, colors\_joined$name)**

**\*colors\_joined %>%**

**\*mutate(name = fct\_reorder(name, difference)) %>%**

**\*ggplot(aes(name, difference, fill = name))+**

**\*geom\_col() +**

**\*coord\_flip() +**

**\*scale\_fill\_manual(values = color\_palette, guide = FALSE)**

-Aggregating sets to look a difference

batman\_colors <- inventory\_parts\_themes %>%

  # Filter the inventory\_parts\_themes table for the Batman theme

  filter(name\_theme == "Batman") %>%

  group\_by(color\_id) %>%

  summarize(total = sum(quantity)) %>%

  # Add a percent column of the total divided by the sum of the total

  mutate(percent=total/sum(total))

# Filter and aggregate the Star Wars set data; add a percent column

star\_wars\_colors <- inventory\_parts\_themes %>%

  filter(name\_theme == "Star Wars") %>%

  group\_by(color\_id) %>%

  summarize(total = sum(quantity)) %>%

  mutate(percent=total/sum(total))

  batman\_colors

  star\_wars\_colors

-Combining Sets

batman\_colors %>%

  full\_join(star\_wars\_colors, by = "color\_id", suffix = c("\_batman", "\_star\_wars")) %>%

  replace\_na(list(total\_batman = 0, total\_star\_wars = 0)) %>%

  inner\_join(colors, by = c("color\_id" = "id")) %>%

  # Create the difference and total columns

  mutate(difference = percent\_batman-percent\_star\_wars,

         total = total\_batman+total\_star\_wars) %>%

  # Filter for totals greater than 200

  filter(total>=200)

-Visualizing the difference : Batman and Star Wars

colors\_joined <- batman\_colors %>%

full\_join(star\_wars\_colors, by = "color\_id", suffix = c("\_batman", "\_star\_wars")) %>%

replace\_na(list(total\_batman = 0, total\_star\_wars = 0)) %>%

inner\_join(colors, by = c("color\_id" = "id")) %>%

mutate(difference = percent\_batman - percent\_star\_wars,

total = total\_batman + total\_star\_wars) %>%

filter(total >= 200) %>%

mutate(name = fct\_reorder(name, difference))

# Create a bar plot using colors\_joined and the name and difference columns

ggplot(colors\_joined, aes(name, difference, fill = name)) +

  geom\_col() +

  coord\_flip() +

  scale\_fill\_manual(values = color\_palette, guide = FALSE) +

  labs(y = "Difference: Batman - Star Wars")

**#CASE STUDY : JOINS ON STACK OVERFLOW DATA**

-Left Joining Question, Question tags, and tags

# Replace the NAs in the tag\_name column

questions %>%

  left\_join(question\_tags, by = c("id" = "question\_id")) %>%

  left\_join(tags, by = c("tag\_id" = "id")) %>%

  replace\_na(list(tag\_name="only-r"))

-Comparing scores across tags

questions\_with\_tags %>%

    # Group by tag\_name

    group\_by(tag\_name) %>%

    # Get mean score and num\_questions

    summarize(score = mean(score),

              num\_questions = n()) %>% -> untuk menghitung jumlah data

    # Sort num\_questions in descending order

    arrange(desc(num\_questions))

-What tags never appeared on R Question

# Using a join, filter for tags that are never on an R question

tags %>%

anti\_join(question\_tags, by=c("id"="tag\_id"))

-Finding Gaps between question and answer

questions %>%

    # Inner join questions and answers with proper suffixes

    inner\_join(answers, by = c("id" = "question\_id"), suffix = c("\_question", "\_answer"))%>%

    # Subtract creation\_date\_question from creation\_date\_answer to create gap

    mutate(gap = as.integer(creation\_date\_answer-creation\_date\_question))

-Joining Question answer and tags

question\_answer\_counts %>%

    # Join the question\_tags tables

    inner\_join(question\_tags, by=c("id"="question\_id"))%>%

    # Join the tags table

    inner\_join(tags, by=c("tag\_id"="id"))

-Average answer by question

tagged\_answers %>%

    # Aggregate by tag\_name

    group\_by(tag\_name)%>%

    # Summarize questions and average\_answers

    summarize(questions = n(),

              average\_answers = mean(n)) %>%

    # Sort the questions in descending order

    arrange(desc(questions))

-creating date variable :

**\*library(lubridate)**

**\*post %>%**

**\*mutate(year = year(creation\_date)) -> hanya mengambil tahunnya saja**

-Counting date variable :

**\*library(lubridate)**

**\*post %>%**

**\*mutate(year = year(creation\_date))%>%**

**\*count(year, type) -> menghitung variable pada tahun dan tipe tersebut. Fungsinya mirip dengan group**

-Binding Rows

**\*questions %>%  
\*bind\_rows (answers)**

-Another bind\_rows :

**\*post <- bind\_rows(question\_type, answer\_type)**

-Joining question and answer with tags

# Inner join the question\_tags and tags tables with the questions table

questions %>%

  inner\_join(question\_tags, by = c("id"="question\_id")) %>%

  inner\_join(tags, by = c("tag\_id" = "id"))

# Inner join the question\_tags and tags tables with the answers table

answers %>%

  inner\_join(question\_tags, by = "question\_id") %>%

  inner\_join(tags, by= c("tag\_id" = "id"))

-Binding and counting post with tags

# Combine the two tables into posts\_with\_tags

posts\_with\_tags <- bind\_rows(questions\_with\_tags %>%

                    mutate(type = "question"),

                    answers\_with\_tags %>%

                    mutate(type = "answer"))

posts\_with\_tags

# Add a year column, then aggregate by type, year, and tag\_name

posts\_with\_tags %>%

mutate(year=year(creation\_date))%>%

count(type, year, tag\_name)

-Visualizing question and answer in tags

# Filter for the dplyr and ggplot2 tag names

by\_type\_year\_tag\_filtered <- by\_type\_year\_tag %>%

  filter(tag\_name %in% c("dplyr", "ggplot2"))

# Create a line plot faceted by the tag name

ggplot(by\_type\_year\_tag\_filtered, aes(year, n, color = type)) +

  geom\_line() +

  facet\_wrap(~ tag\_name)