# R Data Handling Cheat Sheet

## DATA IMPORT

**CSV File**

```
df1 <- read.csv("File Location\\csv_filename.csv")
```

**SAS File**

```
library(sas7bdat)
df2 <- read.sas7bdat("File Location\\sas_filename.sas7bdat")
```

**Excel File**

```
library(readxl)
df3 <- read_excel("File Location\\excel_filename.xlsx", sheet="sheetname")
```

## CHECK LIST AFTER IMPORT

```
dim(df)    check the number of rows and columns
names(df)  show column names
head(df)   shows first few rows of data
tail(df)   shows last few rows of data
str(df)    structure of the data frame
str(df$col_name)   structure of a specific column
view(df)   a snapshot of the small data frame
print(df)  prints the small data frame
```

## QUICK SUMMARY

```
summary(df)    overall data summary
summary(df$col_name)   summary of a column
table(df$col_name)  frequency table for a column
sum(is.na(df)) missing value count of data frame
sum(is.na(df$col_name)) missing count in a column
```

## DATA EXPORT

```
write.csv(df, "./File Location/csv_filename.csv")
write.table(df, "./File Location/text_filename.txt", sep="\t") export as text file
```



## SUB-SETTING DATA

```
df1 <- df[1:10, ] selecting rows 1 to 10
df2 <- df[c(1, 4, 10, 12), ] specific rows
```

Sub-setting Columns

```
df3 <- df[, 2:4] selecting column 2 to 4
df4 <- df[, c(1, 2, 4)]  specific columns
df5 <- df[, c("col_name1", "col_name2")]  selecting specific columns by column name
```

Sub-setting Rows & Columns

```
df6 <- df[5:20, c(1, 2, 4)] selecting rows 5 to 20 and columns 1st, 2nd & 4th
df7 <- df[1:5, c(-3, -5)] selecting rows 1 to 5 and all columns except 3rd & 5th
```

## IF THEN ELSE & SORTING

```
df$new <- ifelse(df$col1<100, "Low", "High") creates a new column with value "Low" when the reference column value is less than 100 and "High" otherwise
```

**Sorting**

```
df1 <- df[order(df$col_name1),] sort the data frame in ascending order of a column
df2 <- df[order(-df$col_name1),]  sort the data frame in descending order of a column
df3 <- df[order(df$col_name1, -df$col_name2),] sort the data frame based on multiple columns
```

## HANDLING DUPLICATES

**Identifying Duplicates**

```
dupes_index <- duplicated(df) fetch duplicate indexes
dupes_all <- df[dupes_index, ] access all duplicates
dupes_unique <- df[!dupes_index, ] unique records
```

**Column Based Duplicates**

```
dupes_col_name1 <- duplicated(df$col_name1) fetch duplicates based on single column
dupes_all1 <- df[dupes_col_name1, ] access all duplicates from single column
dupes_unique1 <- df[!dupes_col_name1, ] access unique records from single column
```

**Removing Duplicates**

```
unique_df1 <- unique(df) remove overall duplicates from data frame
```

## SUBSET WITH FILTER CONDITIONS

```
df_subset1 <- subset(df, col_name1>100) sub-set based on single column condition
df_subset2 <- subset(df, col_name1>100 & col_name2 =="no") sub-set based on multiple columns using AND condition
df_subset3 <- subset(df, col_name1>100 | col_name2 =="no") sub-set based on multiple columns using OR condition
df_subset4 <- subset(df, (col_name1>100 & col_name2 =="no") | col_name3 =="abc") sub-set based on multiple columns using OR / AND conditions with numeric / character filters
df_subset5 <- subset(df, (col_name1>100 & col_name2 =="no") | col_name3 =="abc", select = c(col_name4, col_name6, col_name8)) sub-set based on multiple columns using OR / AND conditions with numeric / character filters by keeping/selecting specific columns
df_subset6 <- subset(df, (col_name1>100 & col_name2 =="no") | col_name3 =="abc", select = -c(col_name5, col_name7, col_name9)) sub-set based on multiple columns using OR / AND conditions with numeric / character filters by dropping specific columns
```

## DATASET MERGE & JOIN

```
df_new1 <- merge(df_one, df_two, by="primary_col") default INNER JOIN using single primary key/column
df_new2 <- merge(df_one, df_two, by=c("primary_col1", "primary_col2") default INNER JOIN using multiple keys/columns
df_new3 <- merge(df_one, df_two, by="primary_col", all=FALSE) INNER JOIN using single primary key/column
df_new4 <- merge(df_one, df_two, by="primary_col", all=TRUE) OUTER JOIN using single primary key/column
df_new5 <- merge(df_one, df_two, by="primary_col", all.x=TRUE) LEFT OUTER JOIN using single primary key/column
df_new6 <- merge(df_one, df_two, by="primary_col", all.y=TRUE) RIGHT OUTER JOIN using single primary key/column
df_new7 <- merge(df_one, df_two, by.x=c("primary_col1"), by.y=c("primary_col2"), all.x=TRUE) LEFT OUTER JOIN using multiple keys/columns
```