[Data Manipulation with dplyr] {CheatSheet}

Basic Operations:

- **Select Columns**: `select(df, col1, col2)`
- **Rename Columns**: `rename(df, new_col = old_col)`
- **Filter Rows**: `filter(df, col > 10)`
- **Arrange Rows**: `arrange(df, col1, desc(col2))`
- **Mutate (Create/Modify Columns)**: `mutate(df, new_col = col1 + col2)`
- **Summarize Data**: `summarize(df, avg_col = mean(col))`
- **Group by**: `group_by(df, col)`
- **Ungroup Data**: `ungroup(df)`

Filtering and Selecting Rows:

- **Filter Rows by Multiple Conditions**: `filter(df, col1 > 10, col2 == "A")`
- **Filter Rows with %in%**: `filter(df, col %in% c("A", "B"))`
- **Top N Rows per Group**: `top_n(df, n = 1, wt = col)`
- **Distinct Rows**: `distinct(df, col1, col2)`

Selecting Columns:

- **Select Columns with Starts With**: `select(df, starts_with("prefix"))`
- **Select Columns with Contains**: `select(df, contains("text"))`
- **Select Columns with Matches Regex**: `select(df, matches("^col[0-9]$"))`

Sorting and Arranging:

- **Order Rows by Specific Column**: `arrange(df, desc(col))`
- **Order Rows by Multiple Columns**: `arrange(df, col1, col2)`

Creating New Variables:

- **Create a New Variable with ifelse**: `mutate(df, new_col = ifelse(col > 10, "High", "Low"))`

By: Waleed Mousa

- **Case When**: `mutate(df, category = case_when(col > 10 ~ "High", TRUE ~ "Low"))`
- **Recoding Values**: `mutate(df, new_col = recode(col, "A" = 1, "B" = 2, "C" = 3))`

Data Aggregation:

- **Summarize by Group**: `group_by(df, col) %>% summarize(avg_val = mean(val))`
- **Count by Group**: `group_by(df, col) %>% tally()`
- **Group by Multiple Columns**: `group_by(df, col1, col2) %>% summarize(avg_val = mean(val))`
- **Count Missing Values**: `summarize(df, missing_count = sum(is.na(col)))`

Joining Data:

- **Inner Join**: `inner_join(df1, df2, by = "key")`
- **Left Join**: `left_join(df1, df2, by = "key")`
- **Right Join**: `right_join(df1, df2, by = "key")`
- **Full Join**: `full_join(df1, df2, by = "key")`

Reshaping Data:

- **Pivot Longer**: `pivot_longer(df, cols = starts_with("X"), names_to = "Variable", values_to = "Value")`
- **Pivot Wider**: `pivot_wider(df, names_from = "Variable", values_from = "Value")`
- **Spread**: `spread(df, key = "Variable", value = "Value")`
- **Gather**: `gather(df, key = "Variable", value = "Value", -id)`

Window Functions:

- **Ranking Rows**: `mutate(df, rank = min_rank(col))`
- **Running Total**: `mutate(df, run_total = cumsum(col))`
- **Lag and Lead**: `mutate(df, lag_col = lag(col), lead_col = lead(col))`

Handling Missing Data:

- **Remove Missing Values**: `filter(df, !is.na(col))`
- **Impute Missing Values with Mean**: `mutate(df, col = ifelse(is.na(col), mean(col, na.rm = TRUE), col))`

Conditional Operations:

- **Conditional Mutate**: `mutate(df, new_col = ifelse(col1 > 10, col2 * 2, col2 / 2))`
- **Conditional Filter**: `filter(df, col1 > 10 & col2 == "A")`

String Operations:

- **Substring**: `mutate(df, sub_str = substr(col, start = 1, stop = 3))`
- **Concatenate Strings**: `mutate(df, new_col = paste(col1, col2, sep = "_"))`
- **String Matching**: `filter(df, str_detect(col, "pattern"))`

Statistical Functions:

- **Mean and Standard Deviation**: `summarize(df, mean_val = mean(col), sd_val = sd(col))`
- **Quantiles**: `summarize(df, q25 = quantile(col, 0.25), q75 = quantile(col, 0.75))`

Date and Time Operations:

- **Convert to Date**: `mutate(df, date_col = as.Date(date_col, format = "%Y-%m-%d"))`
- **Extract Year, Month, Day**: `mutate(df, year_col = year(date_col), month_col = month(date_col), day_col = day(date_col))`
- **Time Difference**: `mutate(df, time_diff = difftime(end_time, start_time, units = "mins"))`

Miscellaneous:

- **Sampling Rows**: `sample_n(df, size = 10)`
- **Random Sampling by Proportion**: `sample_frac(df, 0.1)`

- **Set Operations - Union**: `bind_rows(df1, df2)`
- **Set Operations - Intersection**: `semi_join(df1, df2, by = "key")`
- **Set Operations - Set Difference**: `anti_join(df1, df2, by = "key")`

Conditional Joins:

- **Conditional Inner Join**: `inner_join(df1, df2, by = "key") %>% filter(condition)`
- **Conditional Left Join**: `left_join(df1, df2, by = "key") %>% filter(condition)`

Advanced Joins:

- **Cross Join**: `crossing(df1, df2)`
- **Self Join**: `inner_join(df, df, by = "key")`

Advanced Mutate Operations:

- **Cumulative Operations**: `mutate(df, cum_sum = cumsum(col), cum_prod = cumprod(col))`
- **Rolling Mean**: `mutate(df, roll_mean = zoo::rollmean(col, k = 3, fill = NA))`

Group-wise Operations:

- **Group-wise Maximum and Minimum**: `group_by(df, group_col) %>% summarize(max_val = max(col), min_val = min(col))`
- **Group-wise Lag**: `group_by(df, group_col) %>% mutate(lag_val = lag(col))`

Combining Multiple Operations:

- **Chaining Multiple Operations**: `df %>% filter(col1 > 10) %>% arrange(col2) %>% select(col1, col2)`
- **Group-wise Summarize and Filter**: `group_by(df, group_col) %>% summarize(avg_val = mean(val)) %>% filter(avg_val > 10)`

Advanced Window Functions:

- **Moving Average**: `mutate(df, moving_avg = zoo::rollmean(col, k = 3, fill = NA, align = "right"))`
- **Rank by Group**: `group_by(df, group_col) %>% mutate(rank = dense_rank(col))`

Advanced Mutate with Row-wise Operations:

- **Row-wise Maximum and Minimum**: `mutate(df, row_max = pmax(col1, col2), row_min = pmin(col1, col2))`
- **Row-wise Cumulative Sum**: `mutate(df, row_cumsum = cumsum(c(0, col)))`

Advanced Filtering:

- **Filter by Row Number**: `filter(df, row_number() <= 10)`
- **Filter by Percentile**: `filter(df, quantile(col) > 0.75)`

Advanced Selecting:

- **Select Random Sample of Columns**: `select(df, sample(names(df), size = 3))`
- **Select Columns Matching Criteria**: `select(df, starts_with("X"), ends_with("Y"))`

Handling Duplicate Data:

- **Remove Duplicate Rows**: `distinct(df)`
- **Count Duplicate Rows**: `df %>% group_by_all() %>% tally() %>% filter(n > 1)`

Pivoting and Unpivoting:

- **Pivot Longer with Multiple Columns**: `pivot_longer(df, cols = starts_with("X"), names_to = c(".value", "variable"), names_sep = "_")`
- **Pivot Wider with Multiple Values**: `pivot_wider(df, names_from = "variable", values_from = c("value1", "value2"))`

Advanced Grouping and Aggregation:

- **Group-wise Summary with Custom Function**: `group_by(df, group_col) %>% summarize(custom_stat = my_function(col))`
- **Cumulative Sum by Group**: `group_by(df, group_col) %>% mutate(cum_sum = cumsum(col))`

Handling Time Series Data:

- **Time-based Filtering**: `filter(df, date_col > "2022-01-01")`
- **Rolling Mean by Time**: `mutate(df, rolling_mean = zoo::rollmean(col, k = 3, fill = NA, align = "right"))`

Combining Data Frames:

- **Stacking Data Frames Vertically**: `bind_rows(df1, df2)`
- **Stacking Data Frames Horizontally**: `bind_cols(df1, df2)`

Advanced Set Operations:

- **Set Union with Duplicates**: `bind_rows(df1, df2)`
- **Set Intersection with Duplicates**: `inner_join(df1, df2, by = "key")`
- **Set Difference with Duplicates**: `anti_join(df1, df2, by = "key")`

Handling Factors:

- **Convert Factor to Character**: `mutate(df, col = as.character(col))`
- **Convert Character to Factor**: `mutate(df, col = as.factor(col))`

Advanced Filtering and Selecting:

- **Filter Rows by Pattern Matching**: `filter(df, grepl("pattern", col))`
- **Select Columns by Pattern Matching**: `select(df, matches("pattern"))`

Handling List Columns:

- **Explode List Column**: `df %>% unnest(col)`
- **Creating List Column**: `mutate(df, list_col = list(1:5))`

By: Waleed Mousa

Handling Nested Data Frames:

- **Explode Nested Data Frame**: `df %>% unnest(nested_df)`
- **Creating Nested Data Frame**: `mutate(df, nested_df = list(data = df2))`

Advanced Data Imputation:

- **Impute Missing Values with Linear Interpolation**: `mutate(df, col = approx(seq_along(col), col, method = "linear", rule = 2)$y)`

Handling Spatial Data:

- **Filtering Spatial Data**: `filter(sf, st_intersects(geometry, bounding_box))`
- **Aggregating Spatial Data**: `st_buffer(sf, dist = 100) %>% summarise(total_population = sum(population))`

Handling JSON Data:

- **Extracting Values from JSON Column**: `mutate(df, value = fromJSON(col)$key)`
- **Flattening Nested JSON Column**: `bind_cols(df, map_df(df$col, as_tibble))`

Handling XML Data:

- **Parsing XML Column**: `mutate(df, value = xml_find_first(col, "//path") %>% xml_text())`

Handling Data Types:

- **Coerce to Numeric**: `mutate(df, col = as.numeric(col))`
- **Coerce to Character**: `mutate(df, col = as.character(col))`

Handling Large Data Sets:

- **Sampling Rows**: `sample_n(df, size = 1000)`

- **Lazy Evaluation**: `df %>% filter(col1 > 10) %>% arrange(col2) %>% select(col1, col2) %>% glimpse()`

Advanced Operations on Data Frames:

- **Nested Joins**: `left_join(df1, df2 %>% group_by(key) %>% summarise(new_val = mean(val)), by = "key")`
- **Conditional Aggregation**: `group_by(df, col1) %>% summarize_if(is.numeric, mean)`
- **Conditional Imputation**: `mutate(df, col = ifelse(is.na(col), mean(col, na.rm = TRUE), col))`

Handling Time Series Data:

- **Time-based Aggregation**: `df %>% group_by(floor_date(date_col, "week")) %>% summarize(avg_val = mean(val))`
- **Rolling Window Aggregation**: `df %>% arrange(date_col) %>% mutate(rolling_sum = zoo::rollapply(val, width = 5, FUN = sum, fill = NA))`

Advanced Set Operations:

- **Set Union without Duplicates**: `distinct(bind_rows(df1, df2))`
- **Set Intersection without Duplicates**: `inner_join(df1, df2, by = "key") %>% distinct()`
- **Set Difference without Duplicates**: `anti_join(df1, df2, by = "key") %>% distinct()`

Advanced Window Functions:

- **Lag and Lead by Group**: `group_by(df, group_col) %>% mutate(lag_col = lag(col), lead_col = lead(col))`
- **Conditional Cumulative Sum**: `mutate(df, cum_sum = cumsum(ifelse(condition, col, 0)))`

Advanced Joins:

- **Fuzzy Matching**: `stringdist::stringdist_join(df1, df2, by = c("col1", "col2"), method = "jaccard", max_dist = 0.2)`

- **Spatial Join**: `sf::st_join(sf1, sf2, left = FALSE, join = st_intersects)`

Data Frame Reshaping:

- **Reshape from Wide to Long Format**: `gather(df, key = "variable", value = "value", -id)`
- **Reshape from Long to Wide Format**: `spread(df, key = "variable", value = "value")`

Advanced Mutate Operations:

- **Impute Missing Values Using K-Nearest Neighbors**: `impute::impute_knn(as.matrix(df), k = 3)`
- **Conditional Mutate with Row-wise Operations**: `mutate(df, new_col = ifelse(rowSums(df[, c("col1", "col2")]) > 10, "High", "Low"))`

Advanced Filtering:

- **Filter Rows Based on Row-wise Conditions**: `filter(df, rowSums(df[, c("col1", "col2")]) > 10)`
- **Filter Rows with Anti-join**: `anti_join(df1, df2, by = "key")`

Conditional Operations:

- **Conditional Aggregation Using Case When**: `df %>% group_by(col1) %>% summarize(new_col = case_when(all(col2 > 0) ~ sum(col3), TRUE ~ NA_real_))`

Advanced Selecting:

- **Select Columns by Type**: `select(df, where(is.numeric))`
- **Select Random Sample of Columns by Percentage**: `select(df, sample_frac(0.5))`

Advanced Grouping and Aggregation:

- **Aggregating by Time Intervals and Groups**: `df %>% group_by(group_col, floor_date(date_col, "month")) %>% summarize(avg_val = mean(val))`
- **Aggregating Using Rolling Time Window**: `df %>% group_by(group_col, roll_time = rollapply(date_col, width = "1 week", by = "1 week", FUN = mean)) %>% summarize(avg_val = mean(val))`

Data Manipulation with List Columns:

- **Working with List Columns**: `mutate(df, list_col = list(1:5))`
- **Exploding List Columns and Aggregating**: `df %>% unnest(list_col) %>% group_by(group_col) %>% summarize(avg_val = mean(value))`

Data Manipulation with Nested Data Frames:

- **Working with Nested Data Frames**: `mutate(df, nested_df = list(data = df2))`
- **Exploding Nested Data Frames and Aggregating**: `df %>% unnest(nested_df) %>% group_by(group_col) %>% summarize(avg_val = mean(value))`

Handling Data Types:

- **Coerce to Date**: `mutate(df, date_col = as.Date(date_col, format = "%Y-%m-%d"))`
- **Convert Numeric to Factor with Custom Labels**: `mutate(df, factor_col = factor(num_col, labels = c("Low", "Medium", "High")))`

Combining Multiple Operations:

- **Chaining Multiple Operations with Magrittr Pipe Operator**: `df %>% filter(col1 > 10) %>% arrange(col2) %>% select(col1, col2) %>% glimpse()`

Advanced Filtering and Selecting:

- **Filter Rows by Multiple Patterns**: `filter(df, grepl("pattern1", col1) | grepl("pattern2", col2))`

- **Select Columns by Pattern and Type**: `select(df, matches("pattern"), where(is.numeric))`

Advanced Mutate with Row-wise Operations:

- **Row-wise Operations with Across**: `mutate(df, across(c(col1, col2), ~ . * 2))`
- **Row-wise Operations with Case When and Across**: `mutate(df, across(c(col1, col2), ~ case_when(. > 0 ~ . * 2, TRUE ~ .)))`

Handling Spatial Data:

- **Spatial Operations with sf Package**: `st_intersection(sf1, sf2)`

Handling JSON Data:

- **Parsing Nested JSON Columns**: `df %>% mutate(parsed_json = purrr::map(json_col, jsonlite::fromJSON))`

Handling XML Data:

- **Parsing XML Columns**: `df %>% mutate(parsed_xml = xml2::read_xml(xml_col) %>% xml2::as_list())`

Combining Data Frames:

- **Full Outer Join**: `bind_rows(df1 %>% anti_join(df2, by = "key"), df2 %>% anti_join(df1, by = "key"))`

Advanced Set Operations:

- **Set Union with Custom Logic for Duplicates**: `df1 %>% union(df2) %>% distinct()`