DATA MANIPULATION AND WRANGLING IN R

Statistical Computing in R

Lecturer:Ivan Innocent Sekibenga

1. What is Data Manipulation and Wrangling?

Data manipulation and wrangling is the process of transforming and mapping data from one "raw" form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics. This process is a fundamental step in data analysis and is often necessary to prepare data for visualization, statistical modeling, or machine learning. The goal of data manipulation and wrangling is to clean, structure, and enrich the data to ensure its quality and usability.

In R, several packages are commonly used for data manipulation and wrangling, with the `dplyr` and `tidyr` packages from the `tidyverse` being among the most popular. These packages provide a set of functions that make it easy to perform tasks such as filtering, selecting, mutating, summarizing, and reshaping data. Here are some common data manipulation and wrangling verbs in R using the `dplyr` and `tidyr` packages:`select()`, `filter()`, `mutate()`, `arrange()`, `summarize()`, `group_by()`, `pivot_longer()`, `pivot_wider()`, `unite()`, `separate()`.

2. Loading necessary packages

```
library(tidyverse) # Load the tidyverse package
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- 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 errors
```

```
library(readxl)    # for importing excel files
```

3. Importing dataset to manipulate and wrangle

```
employees <- read_excel("Employees.xlsx")
View(employees) # View the dataset
# this dataset contains information about employees in a company,
# including their names, departments, salaries, and hire dates etc
```

4. The Pipe Operator

The **pipe operator** is the foundation of tidyverse workflows. It allows you to **chain multiple operations together** in a readable way. Instead of nesting functions, you write them in a sequence that flows like a sentence.

- `%>%` → the `magrittr` pipe, widely used in tidyverse.
- `|>` → the base R pipe, available in R 4.1+.

Both work similarly, but `%>%` is more common in tidyverse code.

4.1 Windows keyboard shortcuts for pipes

- `%>%` (magrittr pipe): In RStudio, use `Ctrl + Shift + M`
- `|>` (base R pipe): In RStudio, newer versions also allow `Ctrl + Shift + M`

4.2 Why use pipes?

Pipes make your code **easier to read and understand** by breaking down complex operations into clear, sequential steps. Each step takes the output of the previous step as its input.

- **Readability**: Code reads from left to right, top to bottom, like a story.
- **Maintainability**: Easier to modify or debug individual steps.
- **Clarity**: Each operation is explicit, reducing confusion about data flow.

4.3 How to use pipes

The first step of a sequence of pipes can be a value, a variable, or a function, including arguments. The code below shows a series of examples of different ways of achieving the same result. The examples use the function `round`, which also allows for a second argument: `digits = 2`. Note that, when using the pipe operator, only the nominally second argument is provided to the function `round` – that is `round(digits = 2)`

```
# No pipe, using variables or objects
tmp_variable_A <- 2
tmp_variable_B <- sqrt(tmp_variable_A)
round(tmp_variable_B, digits = 2)
```

```
## [1] 1.41
```

```
# No pipe, using functions only
round(sqrt(2), digits = 2)
```

```
## [1] 1.41
```

```
# Pipe starting from a value
2 %>%
  sqrt() %>%
  round(digits = 2)
```

```
## [1] 1.41
```

```
# Pipe starting from a variable
the_value_two <- 2
the_value_two %>%
  sqrt() %>%
  round(digits = 2)
```

```
## [1] 1.41
```

```r
# Pipe starting from a function
sqrt(2) %>%
 round(digits = 2)
```

```
## [1] 1.41
```

A complex operation created through the use of `%>%` can be used on the right side of `<-`, to assign the outcome of the operation to a variable(object).

```r
sqrt_of_two <-
  2 %>%
  sqrt() %>%
  round(digits = 2)
```

Example: Calculating average salary for IT employees

```r
# Without pipe: nesting makes it harder to read

summarise(filter(employees, Department == "IT"),
  avg_salary = mean(`Annual Salary`, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   avg_salary
##        <dbl>
## 1     97790.
```

```r
# With pipe: each step flows into the next, easier to follow

employees %>%
  filter(Department == "IT") %>%                # Step 1: keep only IT employees
  summarise(avg_salary = mean(`Annual Salary`,# Step 2: calculate average salary
                              na.rm = TRUE))    # Remove missing values
```

```
## # A tibble: 1 x 1
##   avg_salary
##        <dbl>
## 1     97790.
```

When applicable: Whenever your analysis involves **multiple tidyverse verbs in sequence**, pipes make the workflow **clean and logical**.

NB: FOR MOST OF THE EXAMPLES THAT FOLLOW,MODIFY THE CODE BY FORMING OBJECTS IN EACH CHUNK SO THAT YOU CAN VIEW THE ENTIRE OUTPUT AS A DATAFRAME INSTEAD OF A TIBBLE, WHERE NECESSARY.Use the `View()` function if needed.

5. select() – Choosing columns

`select()` helps you **pick or drop columns** from a dataset. It's useful when you want to **focus only on variables of interest**.

Example 5.1: Select specific columns by name

```r
employees %>%
  select(`Full Name`, Department, `Annual Salary`)
```

```
## # A tibble: 1,000 x 3
```

```
##    `Full Name`     Department `Annual Salary`
##    <chr>           <chr>                <dbl>
##  1 Emily Davis     IT                  141604
##  2 Theodore Dinh   IT                   99975
##  3 Luna Sanders    Finance             163099
##  4 Penelope Jordan IT                   84913
##  5 Austin Vo       Finance              95409
##  6 Joshua Gupta    Sales                50994
##  7 Ruby Barnes     IT                  119746
##  8 Luke Martin     Finance              41336
##  9 Easton Bailey   Accounting          113527
## 10 Madeline Walker Finance              77203
## # i 990 more rows
```

```r
# Keep only three variables
```

Explanation: Keeps only `Full Name`, `Department`, and `Annual Salary`.
Situation: Useful when preparing a **salary report** for HR that doesn't need extra details.

Example 5.2: Exclude one column

```r
 employees %>%
  select(-`Exit Date`)    # Drop the "Exit Date" column
```

```
## # A tibble: 1,000 x 13
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 6 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>
```

Explanation: Removes the `Exit Date` column from the dataset.
Situation: When analyzing current employees, the exit date is irrelevant.

Example 5.3: Exclude multiple columns

```r
employees %>%
  select(-c(`Exit Date`, Gender))    # Drop both Exit Date and Gender
```

```
## # A tibble: 1,000 x 12
##    EEID   `Full Name`     `Job Title` Department `Business Unit` Ethnicity   Age
##    <chr>  <chr>           <chr>       <chr>      <chr>           <chr>     <dbl>
##  1 E02387 Emily Davis     Sr. Manger  IT         Research & Dev~ Black        55
##  2 E04105 Theodore Dinh   Technical ~ IT         Manufacturing   Asian        59
##  3 E02572 Luna Sanders    Director    Finance    Speciality Pro~ Caucasian    50
```

```
##  4 E02832 Penelope Jordan Computer S~ IT         Manufacturing   Caucasian    26
##  5 E01639 Austin Vo       Sr. Analyst Finance    Manufacturing   Asian        55
##  6 E00644 Joshua Gupta    Account Re~ Sales      Corporate       Asian        57
##  7 E01550 Ruby Barnes     Manager     IT         Corporate       Caucasian    27
##  8 E04332 Luke Martin     Analyst     Finance    Manufacturing   Black        25
##  9 E04533 Easton Bailey   Manager     Accounting Manufacturing   Caucasian    29
## 10 E03838 Madeline Walker Sr. Analyst Finance    Speciality Pro~ Caucasian    34
## # i 990 more rows
## # i 5 more variables: `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>
```

Explanation: Removes both `Exit Date` and `Gender` columns.
Situation: When doing performance analysis, these variables may not matter.

Example 5.4: Select a range of columns

```r
employees %>%
  select(`Full Name`:Gender)   # Select all columns between Full Name and Gender
```

```
## # A tibble: 1,000 x 5
##    `Full Name`     `Job Title`              Department `Business Unit`    Gender
##    <chr>           <chr>                    <chr>      <chr>              <chr>
##  1 Emily Davis     Sr. Manger               IT         Research & Develo~ Female
##  2 Theodore Dinh   Technical Architect      IT         Manufacturing      Male
##  3 Luna Sanders    Director                 Finance    Speciality Produc~ Female
##  4 Penelope Jordan Computer Systems Manager IT         Manufacturing      Female
##  5 Austin Vo       Sr. Analyst              Finance    Manufacturing      Male
##  6 Joshua Gupta    Account Representative   Sales      Corporate          Male
##  7 Ruby Barnes     Manager                  IT         Corporate          Female
##  8 Luke Martin     Analyst                  Finance    Manufacturing      Male
##  9 Easton Bailey   Manager                  Accounting Manufacturing      Male
## 10 Madeline Walker Sr. Analyst              Finance    Speciality Produc~ Female
## # i 990 more rows
```

Explanation: Selects all columns from `Full Name` to `Gender`.
Situation: Handy for quickly grabbing employee personal details stored together.

Example 5.5: Select columns starting with a prefix

```r
employees %>%
  select(starts_with("Job"))   # Keep all columns starting with "Job"
```

```
## # A tibble: 1,000 x 1
##    `Job Title`
##    <chr>
##  1 Sr. Manger
##  2 Technical Architect
##  3 Director
##  4 Computer Systems Manager
##  5 Sr. Analyst
##  6 Account Representative
##  7 Manager
##  8 Analyst
##  9 Manager
## 10 Sr. Analyst
```

```
## # i 990 more rows
```

Explanation: Selects columns like `Job Title`, `Job Level`, etc.
Situation: Useful when focusing on job-related information or attributes.

Example 5.6: Select columns ending with "Date"

```
employees %>%
  select(ends_with("Date"))   # Keep only date fields
```

```
## # A tibble: 1,000 x 2
##    `Hire Date`         `Exit Date`
##    <dttm>              <dttm>
##  1 2016-04-08 00:00:00 2021-10-16 00:00:00
##  2 1997-11-29 00:00:00 NA
##  3 2006-10-26 00:00:00 NA
##  4 2019-09-27 00:00:00 NA
##  5 1995-11-20 00:00:00 NA
##  6 2017-01-24 00:00:00 NA
##  7 2020-07-01 00:00:00 NA
##  8 2020-05-16 00:00:00 2021-05-20 00:00:00
##  9 2019-01-25 00:00:00 NA
## 10 2018-06-13 00:00:00 NA
## # i 990 more rows
```

Explanation: Selects columns like `Hire Date`, `Exit Date`
Situation: When analyzing employment timelines or tenure i.e helpful when analyzing hiring versus exit trends.

Example 5.7: Select columns containing "Unit"

```
employees %>%
  select(contains("Unit"))   # Selects columns that have "Unit" in the name
```

```
## # A tibble: 1,000 x 1
##    `Business Unit`
##    <chr>
##  1 Research & Development
##  2 Manufacturing
##  3 Speciality Products
##  4 Manufacturing
##  5 Manufacturing
##  6 Corporate
##  7 Corporate
##  8 Manufacturing
##  9 Manufacturing
## 10 Speciality Products
## # i 990 more rows
```

Explanation: Selects columns like `Business Unit`, `Cost Unit`
Situation: For reports that need organizational units.

Example 5.8: Select only numeric columns

```
employees %>%
  select(where(is.numeric))   # Keeps only numeric variables
```

```
## # A tibble: 1,000 x 3
##      Age `Annual Salary` `Bonus %`
##    <dbl>           <dbl>     <dbl>
##  1    55          141604      0.15
##  2    59           99975      0
##  3    50          163099      0.2
##  4    26           84913      0.07
##  5    55           95409      0
##  6    57           50994      0
##  7    27          119746      0.1
##  8    25           41336      0
##  9    29          113527      0.06
## 10    34           77203      0
## # i 990 more rows
```

Explanation: Selects all columns with numeric data types i.e eliminates text fields, keeps salaries,ages etc .
Situation: When performing statistical analysis or calculations requiring numbers only.

Example 5.9: Select only text columns

```
employees %>%
  select(where(is.character))   # Keeps only character (text) variables
```

```
## # A tibble: 1,000 x 9
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 2 more variables: Country <chr>, City <chr>
```

Explanation: Selects all columns with text data types i.e eliminates numeric fields, keeps names,departments etc .
Situation: When preparing reports or summaries that focus on descriptive information and its good for text cleaning or name standardization.

Example 5.10: Rename columns while selecting

```
employees %>%
  select(EmployeeName = `Full Name`, Dept = Department, Salary = `Annual Salary`)   # Rename while sele
```

```
## # A tibble: 1,000 x 3
```

```
##    EmployeeName    Dept       Salary
##    <chr>           <chr>       <dbl>
##  1 Emily Davis     IT         141604
##  2 Theodore Dinh   IT          99975
##  3 Luna Sanders    Finance    163099
##  4 Penelope Jordan IT          84913
##  5 Austin Vo       Finance     95409
##  6 Joshua Gupta    Sales       50994
##  7 Ruby Barnes     IT         119746
##  8 Luke Martin     Finance     41336
##  9 Easton Bailey   Accounting 113527
## 10 Madeline Walker Finance     77203
## # i 990 more rows
```

Explanation: Selects and renames columns for clarity.
Situation: When preparing data for reports, sharing or presentations where concise or shorter names are preferred.This operation does not change the original dataset.

Example 5.11: Select first three columns by index

```
employees %>%
  select(1:3)    # Choose first three columns by position
```

```
## # A tibble: 1,000 x 3
##    EEID   `Full Name`     `Job Title`
##    <chr>  <chr>           <chr>
##  1 E02387 Emily Davis     Sr. Manger
##  2 E04105 Theodore Dinh   Technical Architect
##  3 E02572 Luna Sanders    Director
##  4 E02832 Penelope Jordan Computer Systems Manager
##  5 E01639 Austin Vo       Sr. Analyst
##  6 E00644 Joshua Gupta    Account Representative
##  7 E01550 Ruby Barnes     Manager
##  8 E04332 Luke Martin     Analyst
##  9 E04533 Easton Bailey   Manager
## 10 E03838 Madeline Walker Sr. Analyst
## # i 990 more rows
```

Explanation: Selects the first three columns based on their position in the dataset.
Situation: When the structure of the dataset is known and you want to quickly grab the first few columns without specifying names.

Example 5.12: Reorder columns with Department first

```
employees %>%
  select(Department, everything())    # Move Department to front, keep all others
```

```
## # A tibble: 1,000 x 14
##    Department EEID   `Full Name`     `Job Title` `Business Unit` Gender Ethnicity
##    <chr>      <chr>  <chr>           <chr>       <chr>           <chr>  <chr>
##  1 IT         E02387 Emily Davis     Sr. Manger  Research & Dev~ Female Black
##  2 IT         E04105 Theodore Dinh   Technical ~ Manufacturing   Male   Asian
##  3 Finance    E02572 Luna Sanders    Director    Speciality Pro~ Female Caucasian
##  4 IT         E02832 Penelope Jord~ Computer S~ Manufacturing   Female Caucasian
##  5 Finance    E01639 Austin Vo       Sr. Analyst Manufacturing   Male   Asian
```

```
##  6 Sales      E00644 Joshua Gupta   Account Re~ Corporate       Male   Asian
##  7 IT         E01550 Ruby Barnes    Manager     Corporate       Female Caucasian
##  8 Finance    E04332 Luke Martin    Analyst     Manufacturing   Male   Black
##  9 Accounting E04533 Easton Bailey  Manager     Manufacturing   Male   Caucasian
## 10 Finance    E03838 Madeline Walk~ Sr. Analyst Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

Explanation: Moves the `Department` column to the front while keeping all other columns in their original order.
Situation: When you want to highlight the department information in reports or analyses i.e For reports sorted by Department.

6. filter() – Keep only the rows that meet conditions

`filter()` helps you **keep only the rows that meet certain conditions**. It's useful for **subsetting data** based on specific criteria.

Example 6.1: Filter employees in the IT department

```r
employees %>%
  filter(Department == "IT")   # keep rows where Department equals "IT"
```

```
## # A tibble: 241 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  4 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  5 E04116 David Barnes   Director    IT         Corporate       Male   Caucasian
##  6 E03680 Elias Alvarado Sr. Manger  IT         Manufacturing   Male   Latino
##  7 E03484 Logan Rivera   Director    IT         Research & Dev~ Male   Latino
##  8 E00304 Dylan Choi     Vice Presi~ IT         Corporate       Male   Asian
##  9 E02594 Ezekiel Kumar  IT Coordin~ IT         Research & Dev~ Male   Asian
## 10 E02074 Nora Brown     Enterprise~ IT         Manufacturing   Female Caucasian
## # i 231 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns only employees in the IT department.
When applicable: when building a report for the IT leadership team.

Example 6.2: Filter or keep employees with salary > 100000

```r
employees %>%
  filter(`Annual Salary` > 100000)    # Keep employees with salary > 100000
```

```
## # A tibble: 460 x 14
##    EEID   `Full Name`   `Job Title`  Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>         <chr>        <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis   Sr. Manger   IT         Research & Dev~ Female Black
##  2 E02572 Luna Sanders  Director     Finance    Speciality Pro~ Female Caucasian
```

```
##  3 E01550 Ruby Barnes   Manager      IT         Corporate       Female Caucasian
##  4 E04533 Easton Bailey Manager      Accounting Manufacturing   Male   Caucasian
##  5 E00591 Savannah Ali  Sr. Manger   Human Res~ Manufacturing   Female Asian
##  6 E03344 Camila Rogers Controls En~ Engineeri~ Speciality Pro~ Female Caucasian
##  7 E00530 Eli Jones     Manager      Human Res~ Manufacturing   Male   Caucasian
##  8 E04239 Everleigh Ng  Sr. Manger   Finance    Research & Dev~ Female Asian
##  9 E00549 Isabella Xi   Vice Presid~ Marketing  Research & Dev~ Female Asian
## 10 E00163 Bella Powell  Director     Finance    Research & Dev~ Female Black
## # i 450 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns employees earning more than 100,000.
When applicable: when analyzing high earners for bonus considerations.

Example 6.3: Finance employees under age 40 (AND condition)

```
employees %>%
  filter(Department == "Finance" & Age < 40)    # Keep Finance employees under 40
```

```
## # A tibble: 45 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  2 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
##  3 E03247 Caroline Jenk~ Analyst     Finance    Research & Dev~ Female Caucasian
##  4 E03824 Penelope Cole~ Analyst     Finance    Corporate       Female Black
##  5 E01499 Jade Rojas     Director    Finance    Speciality Pro~ Female Latino
##  6 E00254 Samuel Morales Analyst II  Finance    Corporate       Male   Latino
##  7 E00595 Everly Chow    Sr. Manger  Finance    Speciality Pro~ Female Asian
##  8 E00972 Amelia Salazar Analyst II  Finance    Corporate       Female Latino
##  9 E02872 Liam Jung      Manager     Finance    Corporate       Male   Asian
## 10 E00417 Athena Carril~ Analyst II  Finance    Speciality Pro~ Female Latino
## # i 35 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns Finance employees younger than 40.
When applicable: when targeting younger finance staff for a training program.

Example 6.4: Employees in IT or HR departments (OR condition)

```
 employees %>%
  filter(Department == "IT" | Department == "Human Resources")    # Keep employees in IT or Human Resour
```

```
## # A tibble: 366 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  4 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  5 E00591 Savannah Ali   Sr. Manger  Human Res~ Manufacturing   Female Asian
##  6 E00530 Eli Jones      Manager     Human Res~ Manufacturing   Male   Caucasian
##  7 E04116 David Barnes   Director    IT         Corporate       Male   Caucasian
```

```
##  8 E03680 Elias Alvarado Sr. Manger  IT         Manufacturing   Male   Latino
##  9 E03484 Logan Rivera   Director    IT         Research & Dev~ Male   Latino
## 10 E02206 Jose Henderson Director    Human Res~ Speciality Pro~ Male   Black
## # i 356 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns employees in either IT or HR.
When applicable: when preparing a cross-departmental collaboration report or cross-department policy roll out for Human Resources & IT.

Example 6.5: Employees hired after 2020-01-01

```
employees %>%
  filter(`Hire Date` > as.Date("2020-01-01"))   # Keep employees hired after Jan 1, 2020
```

```
## # A tibble: 152 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  2 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  3 E03344 Camila Rogers  Controls E~ Engineeri~ Speciality Pro~ Female Caucasian
##  4 E04239 Everleigh Ng   Sr. Manger  Finance    Research & Dev~ Female Asian
##  5 E04732 Eva Rivera     Director    Sales      Manufacturing   Female Latino
##  6 E00415 Leilani Butler Analyst II  Marketing  Manufacturing   Female Black
##  7 E02862 Peyton Huang   Sr. Manger  IT         Manufacturing   Female Asian
##  8 E00716 John Chow      Sr. Manger  Marketing  Research & Dev~ Male   Asian
##  9 E03824 Penelope Cole~ Analyst     Finance    Corporate       Female Black
## 10 E03349 Anna Mehta     Cloud Infr~ IT         Speciality Pro~ Female Asian
## # i 142 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns employees hired after January 1, 2020.
When applicable: when analyzing recent hires for onboarding feedback.

Example 6.6: Employees with missing exit dates (current employees)

```
employees %>%
  filter(is.na(`Exit Date`))   # Keep employees with missing Exit Date
```

```
## # A tibble: 915 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  2 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  3 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  4 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  5 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  6 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  7 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
##  8 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
##  9 E00591 Savannah Ali   Sr. Manger  Human Res~ Manufacturing   Female Asian
## 10 E03344 Camila Rogers  Controls E~ Engineeri~ Speciality Pro~ Female Caucasian
## # i 905 more rows
```

```
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns employees who are still with the company (no exit date).
When applicable: when generating a list of current employees for internal communications.

Example 6.7: Employees with non-missing exit dates (former employees)

```
employees %>%
  filter(!is.na(`Exit Date`))   # Keep employees with non-missing Exit Date
```

```
## # A tibble: 85 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  3 E03496 Robert Yang    Sr. Analyst Accounting Speciality Pro~ Male   Asian
##  4 E01754 Owen Lam       Sr. Busine~ Human Res~ Speciality Pro~ Male   Asian
##  5 E00502 Natalia Salaz~ Sr. Analyst Accounting Manufacturing   Female Latino
##  6 E04000 Skylar Carril~ Engineerin~ Engineeri~ Corporate       Female Latino
##  7 E00436 Everly Walker  HRIS Analy~ Human Res~ Speciality Pro~ Female Caucasian
##  8 E02966 William Foster Field Engi~ Engineeri~ Manufacturing   Male   Caucasian
##  9 E01540 Miles Salazar  IT Coordin~ IT         Manufacturing   Male   Latino
## 10 E04474 Mila Hong      Test Engin~ Engineeri~ Research & Dev~ Female Asian
## # i 75 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

```
                                #  i.e these are former employees
```

What this does: returns employees who have left the company (have an exit date).
When applicable: when analyzing turnover rates or exit interviews.

Example 6.8:Employees with age between 30 and 40 (inclusive)

```
employees %>%
  filter(between(Age, 30, 40))   # Keep employees aged between 30 and 40
```

```
## # A tibble: 263 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
##  2 E00591 Savannah Ali   Sr. Manger  Human Res~ Manufacturing   Female Asian
##  3 E03496 Robert Yang    Sr. Analyst Accounting Speciality Pro~ Male   Asian
##  4 E04732 Eva Rivera     Director    Sales      Manufacturing   Female Latino
##  5 E00671 Leonardo Dixon Analyst     Sales      Speciality Pro~ Male   Caucasian
##  6 E03549 Mateo Vu       Account Re~ Sales      Speciality Pro~ Male   Asian
##  7 E02074 Nora Brown     Enterprise~ IT         Manufacturing   Female Caucasian
##  8 E04152 Adeline Huang  Controls E~ Engineeri~ Manufacturing   Female Asian
##  9 E04285 Riley Padilla  Technical ~ IT         Manufacturing   Female Latino
## 10 E01754 Owen Lam       Sr. Busine~ Human Res~ Speciality Pro~ Male   Asian
## # i 253 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

```r
                    # shorthand for Age >=30 & Age <=40
```

What this does: returns employees aged 30 to 40.
When applicable: when targeting a specific age group for wellness programs or benefits.

Example 6.9: Employees in specific departments (using %in%)

```r
sa <- employees %>%
  filter(Department %in% c("IT", "Finance", "Human Resources"))
View(sa)# Keep employees in IT, Finance, or Human Resources
```

What this does: returns employees in IT, Finance, or Human Resources.
When applicable: when preparing a report for multiple departments or cross-functional teams.

Example 6.10: Employees not in specific departments (using !%in%)

```r
employees %>%
  filter(!Department %in% c("IT", "Finance"))   # Keep employees not in IT or Finance
```

```
## # A tibble: 639 x 14
##    EEID   `Full Name`   `Job Title`  Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>         <chr>        <chr>      <chr>           <chr>  <chr>
##  1 E00644 Joshua Gupta  Account Rep~ Sales      Corporate       Male   Asian
##  2 E04533 Easton Bailey Manager      Accounting Manufacturing   Male   Caucasian
##  3 E00591 Savannah Ali  Sr. Manger   Human Res~ Manufacturing   Female Asian
##  4 E03344 Camila Rogers Controls En~ Engineeri~ Speciality Pro~ Female Caucasian
##  5 E00530 Eli Jones     Manager      Human Res~ Manufacturing   Male   Caucasian
##  6 E03496 Robert Yang   Sr. Analyst  Accounting Speciality Pro~ Male   Asian
##  7 E00549 Isabella Xi   Vice Presid~ Marketing  Research & Dev~ Female Asian
##  8 E00884 Camila Silva  Sr. Manger   Marketing  Speciality Pro~ Female Latino
##  9 E04625 Adam Dang     Director     Sales      Research & Dev~ Male   Asian
## 10 E04732 Eva Rivera    Director     Sales      Manufacturing   Female Latino
## # i 629 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns employees not in IT or Finance.
When applicable: when analyzing departments outside of IT and Finance for organizational changes.

Example 6.11: Female managers (string equality and Job Title check)

```r
employees %>%
  filter(Gender == "Female", `Job Title` == "Manager") # keep all female managers
```

```
## # A tibble: 44 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  2 E03749 Kennedy Foster Manager     Marketing  Speciality Pro~ Female Caucasian
##  3 E04798 Aurora Ali     Manager     Marketing  Research & Dev~ Female Asian
##  4 E00105 Isla Espinoza  Manager     Accounting Speciality Pro~ Female Latino
##  5 E03061 Vivian Lewis   Manager     Marketing  Manufacturing   Female Caucasian
##  6 E04630 Maria Griffin  Manager     Marketing  Manufacturing   Female Caucasian
##  7 E04348 Natalia Owens  Manager     Human Res~ Manufacturing   Female Caucasian
```

```
##  8 E02147 Allison Medina Manager     Finance    Speciality Pro~ Female Latino
##  9 E04168 Mila Juarez    Manager     Sales      Speciality Pro~ Female Latino
## 10 E03328 Lucy Johnson   Manager     IT         Research & Dev~ Female Caucasian
## # i 34 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: returns all female employees with the job title "Manager".
When applicable: when preparing diversity reports or leadership analyses.

Example 6.12: Employees whose city is Miami (string equality)

```r
employees %>%
  filter(City == "Miami")   # keep employees located in Miami
```

```
## # A tibble: 112 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  2 E00591 Savannah Ali   Sr. Manger  Human Res~ Manufacturing   Female Asian
##  3 E04732 Eva Rivera     Director    Sales      Manufacturing   Female Latino
##  4 E04285 Riley Padilla  Technical ~ IT         Manufacturing   Female Latino
##  5 E01848 Zoey Jackson   Business P~ Human Res~ Manufacturing   Female Black
##  6 E03824 Penelope Cole~ Analyst     Finance    Corporate       Female Black
##  7 E00935 Joseph Martin  Analyst II  Marketing  Corporate       Male   Black
##  8 E01525 Jose Ross      Engineerin~ Engineeri~ Research & Dev~ Male   Caucasian
##  9 E01258 Gabriel Brooks Network En~ IT         Manufacturing   Male   Caucasian
## 10 E04959 Noah King      Developmen~ Engineeri~ Speciality Pro~ Male   Black
## # i 102 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: keeps employees located in Miami.
When applicable: when analyzing location-specific data or planning local events.

Example 6.13: Employees still employed (no Exit Date)

```r
employees %>%
  filter(is.na(`Exit Date`))   # keeps rows where Exit Date is missing (NA)
```

```
## # A tibble: 915 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  2 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  3 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  4 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  5 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  6 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  7 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
##  8 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
##  9 E00591 Savannah Ali   Sr. Manger  Human Res~ Manufacturing   Female Asian
## 10 E03344 Camila Rogers  Controls E~ Engineeri~ Speciality Pro~ Female Caucasian
## # i 905 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
```

```
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: retains active employees only. **When applicable:** payroll processing or headcount reporting.

7. summarise() –collapse rows to summary values

The `summarise()` function is used to compute single-value summaries such as mean, sum, counts, min/max and distinct counts. This verb or function is often used with `group_by()`

Example 7.1: Overall average annual salary

```
employees %>%
  summarise(avg_salary = mean(`Annual Salary`, na.rm = TRUE))   # mean of `Annual Salary` column
```

```
## # A tibble: 1 x 1
##   avg_salary
##        <dbl>
## 1    113217.
```

```
# na.rm = TRUE ignores missing values
# returns a single value: the average salary across all employees
# avg_salary is the assigned name of the new summary column
# avg_salary is also the object storing the value 113217.4,
# which is the average salary across all employees
```

What this does: single-row result with avg_salary.
When applicable: when need to compute average salary for all employees

Example 7.2: Count of employees

```
employees %>%
  summarise(total_employees = n())   # n() returns number of rows
```

```
## # A tibble: 1 x 1
##   total_employees
##             <int>
## 1            1000
```

What this does: total headcount.
When applicable: workforce size reporting.

Example 7.2: Minimum and maximum salary in one call

```
employees %>%
  summarise(min_salary = min(`Annual Salary`, na.rm = TRUE),
            max_salary = max(`Annual Salary`, na.rm = TRUE))   # multiple summaries
```

```
## # A tibble: 1 x 2
##   min_salary max_salary
##        <dbl>      <dbl>
## 1      40063     258498
```

```
# min_salary and max_salary are new column names
# na.rm = TRUE ignores missing values
```

What this does: returns lowest and highest pay or salary values.
When applicable: salary range checks.

Example 7.3: Median age

```
employees %>%
  summarise(median_age = median(Age, na.rm = TRUE)) # median of Age column
```

```
## # A tibble: 1 x 1
##   median_age
##        <dbl>
## 1         45
```

```
# median_age is the assigned name of the new summary column
# na.rm = TRUE ignores missing values
```

What this does: returns median age to identify the center of age distribution. .
When applicable: workforce demographic summary or age distribution analysis.

Example 7.4: Standard deviation of salaries

```
employees %>%
  summarise(sd_salary = sd(`Annual Salary`, na.rm = TRUE))  # standard deviation of Annual Salary
```

```
## # A tibble: 1 x 1
##   sd_salary
##       <dbl>
## 1    53546.
```

```
# sd_salary is the assigned name of the new summary column
# na.rm = TRUE ignores missing values
```

What this does: dispersion of salaries.
When applicable: assessing pay equity or variability.

Example 7.4: Number of unique departments

```
employees %>%
  summarise(unique_departments = n_distinct(Department))   # count distinct departments
```

```
## # A tibble: 1 x 1
##   unique_departments
##                <int>
## 1                  7
```

```
# unique_departments is the assigned name of the new summary column
# n_distinct() counts unique values
```

What this does: returns count of unique departments.
When applicable: organizational structure analysis or diversity reporting.

Example 7.5: Total payroll (sum of all salaries)

```
employees %>%
  summarise(total_payroll = sum(`Annual Salary`, na.rm = TRUE))   # sum of Annual Salary column
```

```
## # A tibble: 1 x 1
##   total_payroll
##           <dbl>
## 1     113217365
```

```
# total_payroll is the assigned name of the new summary column
# na.rm = TRUE ignores missing values
```

What this does: total annual payroll cost.
When applicable: finance budgeting and headcount cost.

Example 7.6: Ratio of males to females

```
employees %>%
  summarise(male_count = sum(Gender == "Male"),
            female_count = sum(Gender == "Female"),
            male_to_female = male_count / female_count)
```

```
## # A tibble: 1 x 3
##   male_count female_count male_to_female
##        <int>        <int>          <dbl>
## 1        482          518          0.931
```

```
# male_count is the count of all male employees
# female_count is the count of all female employees
# male_to_female is the ratio of males to females
```

What this does: returns the ratio
When applicable: diversity and inclusion reporting or gender balance reporting.

Example 7.7: Summarise multiple numeric columns at once using across()

```
employees %>%
  summarise(across(c(Age, `Annual Salary`, `Bonus %`),
                   list(mean = ~mean(.x, na.rm = TRUE),
                        sd   = ~sd(.x, na.rm = TRUE))))
```

```
## # A tibble: 1 x 6
##   Age_mean Age_sd `Annual Salary_mean` `Annual Salary_sd` `Bonus %_mean`
##      <dbl>  <dbl>                <dbl>              <dbl>          <dbl>
## 1     44.4   11.2              113217.             53546.         0.0887
## # i 1 more variable: `Bonus %_sd` <dbl>
```

```
# across() applies functions to multiple columns
# list() defines multiple summary functions
# .x refers to each column being summarized
```

What this does: returns mean and standard deviation for Age, Annual Salary, and Bonus %.
When applicable: comprehensive numeric summaries for key metrics.

Example 7.8: Count of employees by department

```
employees %>%
  group_by(Department) %>%   # Step 1: group by Department
  summarise(count = n())     # Step 2: count employees in each department
```

```
## # A tibble: 7 x 2
##   Department      count
##   <chr>           <int>
## 1 Accounting         96
## 2 Engineering       158
## 3 Finance           120
## 4 Human Resources   125
## 5 IT                241
## 6 Marketing         120
## 7 Sales             140
```

What this does: returns employee counts per department.
When applicable: workforce distribution analysis or departmental reporting.

Example 7.9: Average salary by department

```
employees %>%
  group_by(Department) %>%   # Step 1: group by Department
  summarise(avg_salary = mean(`Annual Salary`, na.rm = TRUE))   # Step 2: average salary per department
```

```
## # A tibble: 7 x 2
##   Department      avg_salary
##   <chr>                <dbl>
## 1 Accounting         123147.
## 2 Engineering        109035.
## 3 Finance            122803.
## 4 Human Resources    118058.
## 5 IT                  97790.
## 6 Marketing          129663.
## 7 Sales              111050.
```

What this does: returns average salary for each department.
When applicable: salary benchmarking or departmental budget planning.

Example 7.10: Maximum and minimum age by department

```
employees %>%
  group_by(Department) %>%   # Step 1: group by Department
  summarise(min_age = min(Age, na.rm = TRUE),
            max_age = max(Age, na.rm = TRUE))   # Step 2: min and max age per department
```

```
## # A tibble: 7 x 3
##   Department      min_age max_age
##   <chr>             <dbl>   <dbl>
## 1 Accounting           25      64
## 2 Engineering          25      65
## 3 Finance              25      65
## 4 Human Resources      25      65
## 5 IT                   25      65
## 6 Marketing            25      65
## 7 Sales                25      65
```

What this does: returns minimum and maximum age for each department.
When applicable: demographic analysis or age diversity reporting by department.

8. mutate() — add or transform columns (does not reduce row count)

`mutate()` helps you **add new columns or transform existing ones**. It's useful for **creating derived variables** or modifying data. Used to create derived features, flags, cleaned strings and computed numeric columns.

Example 8.1: Add a new column for monthly salary

```
employees %>%
  mutate(Monthly_Salary = `Annual Salary` / 12)    # New column: Monthly_Salary
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
## #   Monthly_Salary <dbl>
```

What this does: adds a `Monthly_Salary` column by dividing annual salary by 12.
When applicable: when preparing payroll reports or budgeting monthly expenses.

Example 8.2: Create a flag for high earners (salary > 100000)

```
employees %>%
  mutate(High_Earner = ifelse(`Annual Salary` > 100000, TRUE, FALSE))    # New column: High_Earner flag
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
```

```
## #   High_Earner <lgl>
```

```
# ifelse() is used to create conditional values
# TRUE if salary > 100000, else FALSE
```

What this does: adds a `High_Earner` boolean column.
When applicable: when identifying employees for bonus eligibility or special programs.

Example 8.3: Standardize job titles to lowercase

```
employees %>%
  mutate(Job_Title = tolower(`Job Title`))   # Transform Job Title to lowercase
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
## #   Job_Title <chr>
```

What this does: converts all job titles to lowercase for consistency.
When applicable: when cleaning data for analysis or reporting.

Example 8.4:Bonus amount in currency

```
employees %>%
  mutate(Bonus_Amount = `Bonus %`  * `Annual Salary`)   # New column: Bonus_Amount
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
```

```
## #   Bonus_Amount <dbl>
```

What this does: calculates the actual bonus amount based on the bonus percentage and annual salary.
When applicable: when preparing compensation reports or financial planning.

Example 8.5: Salary in thousands (easier readability)

```
employees %>%
  mutate(salary_k = `Annual Salary` / 1000)    # scale down salary
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
## #   salary_k <dbl>
```

```
# New column: salary_k (salary in thousands)
```

What this does: adds a `salary_k` column for salary in thousands.
When applicable: when creating reports where large numbers are easier to read in thousands or plots or tables where thousands improve readability. .

Example 8.6: Extract hire year from hire date

```
employees %>%
  mutate(Hire_Year = lubridate::year(`Hire Date`))    # New column: Hire_Year
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
```

```
## #   Hire_Year <dbl>
```

What this does: adds a `Hire_Year` column by extracting the year from the hire date.
When applicable: when analyzing hiring trends over time or cohort analyses.

Example 8.7: Age group (categorical) using case_when

```
employees %>%
  mutate(age_group = case_when(
    Age < 30 ~ "Young",
    Age < 50 ~ "Mid-career",
    TRUE     ~ "Senior"))   # create categorical column from numeric Age
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
## #   age_group <chr>
```

```
# case_when() allows multiple conditions
# TRUE is the default case for Age >= 50
```

What this does: adds an `age_group` column categorizing employees into “Young”, “Mid-career”, and “Senior” i.e assigns age group labels
When applicable: when segmenting workforce for targeted programs or analyses.

Example 8.8: Create combined location string

```
employees %>%
  mutate(Location = paste(Country, City, sep = ", "))
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
```

```
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
## #   Location <chr>
```

```r
# New column: Location (Country, City)
```

What this does: adds a `Location` column combining country and city.
When applicable: when preparing location-based reports or visualizations.

Example 8.9: Fill missing Exit Date with a sentinel value (e.g., "Still Employed")

```r
employees %>%
  mutate(ExitDate_clean = coalesce(as.character(`Exit Date`), "Still Employed"))
```

```
## # A tibble: 1,000 x 15
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 8 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
## #   ExitDate_clean <chr>
```

```r
# coalesce() returns first non-NA value; convert date to string for consistency
# New column: ExitDate_clean
# as.character() converts date to string for consistency
# a string value "Still Employed" is used for missing exit dates
```

What this does: adds an `ExitDate_clean` column replacing missing exit dates with "Still Employed".
When applicable: when preparing reports that need a clear indication of current employees and reporting where NA would be confusing to non-technical audiences.

9. arrange() —reorder rows (sorting)

arrange() helps you **sort rows** based on one or more columns. It's useful for **organizing data** for reports or analyses.

Example 9.1: Sort employees by annual salary (ascending)

```
employees %>%
  arrange(`Annual Salary`)    # Sort by Annual Salary (lowest to highest)
```

```
## # A tibble: 1,000 x 14
##    EEID   `Full Name`   `Job Title`   Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>         <chr>         <chr>      <chr>           <chr>  <chr>
##  1 E03928 Miles Dang    IT Coordina~ IT         Speciality Pro~ Male   Asian
##  2 E04109 Leah Bryant   IT Coordina~ IT         Manufacturing   Female Caucasian
##  3 E00862 Levi Moreno   Systems Ana~ IT         Research & Dev~ Male   Latino
##  4 E03719 Jack Brown    Analyst      Marketing  Corporate       Male   Caucasian
##  5 E02732 Alice Tran    Analyst      Marketing  Corporate       Female Asian
##  6 E04332 Luke Martin   Analyst      Finance    Manufacturing   Male   Black
##  7 E02183 Sarah Ayala   Analyst      Sales      Corporate       Female Latino
##  8 E01895 Peyton Walker Analyst      Marketing  Research & Dev~ Female Caucasian
##  9 E01832 Ezra Singh    Analyst      Finance    Manufacturing   Male   Asian
## 10 E01361 Emma Hill     IT Coordina~ IT         Manufacturing   Female Caucasian
## # i 990 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: sorts employees from lowest to highest salary.
When applicable: when preparing salary reports or identifying low earners.

Example 9.2: Sort employees by annual salary (descending)

```
employees %>%
  arrange(desc(`Annual Salary`))    # Sort by Annual Salary (highest to lowest)
```

```
## # A tibble: 1,000 x 14
##    EEID   `Full Name`   `Job Title`   Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>         <chr>         <chr>      <chr>           <chr>  <chr>
##  1 E04354 Raelynn Rios  Vice Presid~ Sales      Manufacturing   Female Latino
##  2 E04742 Kinsley Vega  Vice Presid~ Accounting Corporate       Female Latino
##  3 E02522 Silas Rivera  Vice Presid~ Sales      Corporate       Male   Latino
##  4 E01371 Dominic Le    Vice Presid~ Marketing  Corporate       Male   Asian
##  5 E04170 Grayson Chin  Vice Presid~ IT         Research & Dev~ Male   Asian
##  6 E01628 Jackson Perry Vice Presid~ Marketing  Research & Dev~ Male   Caucasian
##  7 E00917 Skylar Bell   Vice Presid~ Engineeri~ Manufacturing   Female Caucasian
##  8 E04103 Isabella Soto Vice Presid~ Finance    Corporate       Female Latino
##  9 E01584 Henry Zhu     Vice Presid~ Marketing  Speciality Pro~ Male   Asian
## 10 E02825 Wyatt Li      Vice Presid~ Engineeri~ Manufacturing   Male   Asian
## # i 990 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: sorts employees from highest to lowest salary.
When applicable: when preparing executive reports or identifying top earners.

Example 9.3: Sort by department, then by salary within each department

```r
employees %>%
  arrange(Department, desc(`Annual Salary`))   # Sort by Department, then by Annual Salary (desc)
```

```
## # A tibble: 1,000 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E04742 Kinsley Vega   Vice Presi~ Accounting Corporate       Female Latino
##  2 E02563 Emily Clark    Vice Presi~ Accounting Corporate       Female Caucasian
##  3 E03289 Christopher L~ Vice Presi~ Accounting Manufacturing   Male   Asian
##  4 E02202 Emilia Bailey  Vice Presi~ Accounting Speciality Pro~ Female Caucasian
##  5 E04249 Hadley Dang    Vice Presi~ Accounting Corporate       Female Asian
##  6 E01249 Samuel Bailey  Vice Presi~ Accounting Speciality Pro~ Male   Caucasian
##  7 E00380 Alice Thompson Vice Presi~ Accounting Speciality Pro~ Female Caucasian
##  8 E03545 Sofia Cheng    Vice Presi~ Accounting Corporate       Female Asian
##  9 E04359 Greyson Lam    Vice Presi~ Accounting Manufacturing   Male   Asian
## 10 E00103 Nora Park      Director    Accounting Speciality Pro~ Female Asian
## # i 990 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

```r
# desc() sorts in descending order
```

What this does: sorts first by department, then by salary within each department.
When applicable: when preparing departmental salary reports or analyses that require hierarchical sorting.

Example 9.4: Sort by hire date (oldest to newest)

```r
employees %>%
  arrange(`Hire Date`)   # Sort by Hire Date (oldest to newest) with earliest hire at top
```

```
## # A tibble: 1,000 x 14
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02710 Silas Huang    Engineerin~ Engineeri~ Research & Dev~ Male   Asian
##  2 E01967 John Dang      Director    Sales      Corporate       Male   Asian
##  3 E03042 Ava Nelson     Systems An~ IT         Manufacturing   Female Caucasian
##  4 E01525 Jose Ross      Engineerin~ Engineeri~ Research & Dev~ Male   Caucasian
##  5 E01591 Paisley Trinh  Technical ~ IT         Corporate       Female Asian
##  6 E00788 Emily Contrer~ Analyst II  Sales      Manufacturing   Female Latino
##  7 E02440 Grayson Turner Solutions ~ IT         Corporate       Male   Caucasian
##  8 E00245 Benjamin Delg~ Test Engin~ Engineeri~ Corporate       Male   Latino
##  9 E03402 Isaac Liu      Field Engi~ Engineeri~ Manufacturing   Male   Asian
## 10 E04247 Camila Evans   Manager     Marketing  Research & Dev~ Female Black
## # i 990 more rows
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

What this does: sorts employees from oldest to newest hire date.
When applicable: find longest-tenured employees.

10. group by()

The `group_by()` function is used to group data by one or more variables. It is often used in conjunction with `summarise()` to perform calculations on each group separately.

Example 10.1: Group by department and calculate average salary

```
employees %>%
  group_by(Department) %>%   # Step 1: group by Department
  summarise(avg_salary = mean(`Annual Salary`, na.rm = TRUE))   # Step 2: average salary per department
```

```
## # A tibble: 7 x 2
##   Department      avg_salary
##   <chr>                <dbl>
## 1 Accounting         123147.
## 2 Engineering        109035.
## 3 Finance            122803.
## 4 Human Resources    118058.
## 5 IT                  97790.
## 6 Marketing          129663.
## 7 Sales              111050.
```

What this does: returns average salary for each department.
When applicable: salary benchmarking or departmental budget planning.

Example 10.2: Group by department and count employees

```
employees %>%
  group_by(Department) %>%   # Step 1: group by Department
  summarise(count = n())     # Step 2: count employees in each department
```

```
## # A tibble: 7 x 2
##   Department      count
##   <chr>           <int>
## 1 Accounting         96
## 2 Engineering       158
## 3 Finance           120
## 4 Human Resources   125
## 5 IT                241
## 6 Marketing         120
## 7 Sales             140
```

What this does: returns employee counts per department.
When applicable: workforce distribution analysis or departmental reporting.

Example 10.3: Salary summary statistics by Gender

```
employees %>%
  group_by(Gender) %>%
  summarise(min_salary = min(`Annual Salary`, na.rm = TRUE), # minimum salary
            max_salary = max(`Annual Salary`, na.rm = TRUE), # maximum salary
            avg_salary = mean(`Annual Salary`, na.rm = TRUE)) # average salary
```

```
## # A tibble: 2 x 4
##   Gender min_salary max_salary avg_salary
```

```
##   <chr>       <dbl>      <dbl>      <dbl>
## 1 Female      40124      258498     112314.
## 2 Male        40063      258081     114188.
```

What this does: minimum,maximum and average salary by gender
When applicable: pay equity analysis or diversity reporting or gender pay gap analysis.

Example 10.4: Total bonus payout by Department

```
employees %>%
  group_by(Department) %>%
  summarise(total_bonus = sum(`Bonus %` * `Annual Salary`, na.rm = TRUE))   # total bonus payout
```

```
## # A tibble: 7 x 2
##   Department      total_bonus
##   <chr>                 <dbl>
## 1 Accounting         1978470.
## 2 Engineering        1986376.
## 3 Finance            2407170.
## 4 Human Resources    2357839.
## 5 IT                 2334997.
## 6 Marketing          2743498.
## 7 Sales              2065452.
```

```
# sum of (Bonus % * Annual Salary) gives total bonus amount
# total_bonus is the assigned name of the new summary column
```

What this does: returns total bonus payout for each department.
When applicable: budgeting for bonuses or departmental compensation planning.

Example 10.5: Average age and salary by Department

```
employees %>%
  group_by(Department) %>%
  summarise(avg_age = mean(Age, na.rm = TRUE),    # average age
            avg_salary = mean(`Annual Salary`, na.rm = TRUE)) # average salary
```

```
## # A tibble: 7 x 3
##   Department      avg_age avg_salary
##   <chr>             <dbl>      <dbl>
## 1 Accounting         43.7    123147.
## 2 Engineering        45.7    109035.
## 3 Finance            45.3    122803.
## 4 Human Resources    44.5    118058.
## 5 IT                 44.3     97790.
## 6 Marketing          43.2    129663.
## 7 Sales              43.6    111050.
```

What this does: returns average age and salary for each department.
When applicable: demographic and compensation analysis by department.

Example 10.6: Proportion of workforce in each Department

```
employees %>%
  group_by(Department) %>%
```

```r
  summarise(count = n()) %>%
  mutate(proportion = count / sum(count))    # proportion of total workforce
```

```
## # A tibble: 7 x 3
##   Department      count proportion
##   <chr>           <int>      <dbl>
## 1 Accounting         96      0.096
## 2 Engineering       158      0.158
## 3 Finance           120      0.12
## 4 Human Resources   125      0.125
## 5 IT                241      0.241
## 6 Marketing         120      0.12
## 7 Sales             140      0.14
```

What this does: returns the proportion of employees in each department or proportion of workforce in each Department .
When applicable: organizational structure analysis or workforce distribution reporting.

Example 10.7: Highest-paid employee per Department (use slice_max)

```r
employees %>%
  group_by(Department) %>%
  slice_max(order_by = `Annual Salary`, n = 1)    # get row(s) with max salary per group
```

```
## # A tibble: 7 x 14
## # Groups:   Department [7]
##   EEID   `Full Name`    `Job Title`   Department `Business Unit` Gender Ethnicity
##   <chr>  <chr>          <chr>         <chr>      <chr>           <chr>  <chr>
## 1 E04742 Kinsley Vega   Vice Preside~ Accounting Corporate       Female Latino
## 2 E00917 Skylar Bell    Vice Preside~ Engineeri~ Manufacturing   Female Caucasian
## 3 E04103 Isabella Soto  Vice Preside~ Finance    Corporate       Female Latino
## 4 E02599 Daniel Huang   Vice Preside~ Human Res~ Corporate       Male   Asian
## 5 E04170 Grayson Chin   Vice Preside~ IT         Research & Dev~ Male   Asian
## 6 E01371 Dominic Le     Vice Preside~ Marketing  Corporate       Male   Asian
## 7 E04354 Raelynn Rios   Vice Preside~ Sales      Manufacturing   Female Latino
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

```r
# slice_max() returns rows with highest values in specified column
# n = 1 returns the top row only
# order_by specifies the column to determine the maximum
```

What this does: returns the highest-paid employee in each department.
When applicable: identifying top talent or benchmarking salaries by department.

Example 10.8: Lowest-paid employee per Department (use slice_min)

```r
employees %>%
  group_by(Department) %>%
  slice_min(order_by = `Annual Salary`, n = 1)    # get row(s) with min salary per group
```

```
## # A tibble: 7 x 14
## # Groups:   Department [7]
##   EEID   `Full Name`    `Job Title`   Department `Business Unit` Gender Ethnicity
##   <chr>  <chr>          <chr>         <chr>      <chr>           <chr>  <chr>
```

```
## 1 E01877 Abigail Garza Analyst       Accounting Manufacturing   Female Latino
## 2 E04484 Vivian Thao   Quality Engi~ Engineeri~ Research & Dev~ Female Asian
## 3 E04332 Luke Martin   Analyst       Finance    Manufacturing   Male   Black
## 4 E01388 Cooper Gupta  Business Par~ Human Res~ Speciality Pro~ Male   Asian
## 5 E03928 Miles Dang    IT Coordinat~ IT         Speciality Pro~ Male   Asian
## 6 E03719 Jack Brown    Analyst       Marketing  Corporate       Male   Caucasian
## 7 E02183 Sarah Ayala   Analyst       Sales      Corporate       Female Latino
## # i 7 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>
```

```
# slice_min() returns rows with lowest values in specified column
# n = 1 returns the bottom row only
# order_by specifies the column to determine the minimum
```

What this does: returns the lowest-paid employee in each department.
When applicable: identifying entry-level roles or benchmarking salaries by department.

Example 10.9: Mark high earners per Department (create a per-group flag)

```
employees %>%
  group_by(Department) %>%
  mutate(dept_median = median(`Annual Salary`, na.rm = TRUE),
         above_median = `Annual Salary` > dept_median) %>%
  ungroup()   # remove grouping after per-group operations
```

```
## # A tibble: 1,000 x 16
##    EEID   `Full Name`    `Job Title` Department `Business Unit` Gender Ethnicity
##    <chr>  <chr>          <chr>       <chr>      <chr>           <chr>  <chr>
##  1 E02387 Emily Davis    Sr. Manger  IT         Research & Dev~ Female Black
##  2 E04105 Theodore Dinh  Technical ~ IT         Manufacturing   Male   Asian
##  3 E02572 Luna Sanders   Director    Finance    Speciality Pro~ Female Caucasian
##  4 E02832 Penelope Jord~ Computer S~ IT         Manufacturing   Female Caucasian
##  5 E01639 Austin Vo      Sr. Analyst Finance    Manufacturing   Male   Asian
##  6 E00644 Joshua Gupta   Account Re~ Sales      Corporate       Male   Asian
##  7 E01550 Ruby Barnes    Manager     IT         Corporate       Female Caucasian
##  8 E04332 Luke Martin    Analyst     Finance    Manufacturing   Male   Black
##  9 E04533 Easton Bailey  Manager     Accounting Manufacturing   Male   Caucasian
## 10 E03838 Madeline Walk~ Sr. Analyst Finance    Speciality Pro~ Female Caucasian
## # i 990 more rows
## # i 9 more variables: Age <dbl>, `Hire Date` <dttm>, `Annual Salary` <dbl>,
## #   `Bonus %` <dbl>, Country <chr>, City <chr>, `Exit Date` <dttm>,
## #   dept_median <dbl>, above_median <lgl>
```

What this does: adds a flag indicating if an employee earns above the median salary for their department.
When applicable: identifying high performers or salary benchmarking within departments or identify internal high performers relative to peers.