Tidyverse

In data science, we spend most of our time doing four things:

1. **Importing** data
2. **Tidying** data
3. **Transforming/manipulating** data
4. **Visualising** results

The **Tidyverse** is a collection of R packages designed exactly for these tasks.

Core Packages

Task	Package
Data Import	readr
Tidying	tidyr
Transform	dplyr
Visualizing	ggplot2
Strings	stringr
Factors	forcats
Dates	lubridate
Programming	purrr, tibble

```
install.packages("tidyverse")
library(readr)

students <- readr::read_csv("students_large.csv")
View(students)
```

	student_id	name	age	gender	department	math_score	english_score	attendance
1	1001	Student_1	24	F	Computer Science	63	69	77
2	1002	Student_2	21	F	Data Science	77	77	73
3	1003	Student_3	28	M	Data Science	54	57	93
4	1004	Student_4	25	M	Computer Science	96	90	94
5	1005	Student_5	22	F	Data Science	98	88	72
6	1006	Student_6	24	F	Computer Science	79	50	72
7	1007	Student_7	27	F	Mathematics	95	52	96
8	1008	Student_8	20	F	Computer Science	54	62	98
9	1009	Student_9	24	M	Mathematics	61	77	87
10	1010	Student_10	28	F	Computer Science	65	98	98

✅ tidyr → Data Tidying:

What is Tidy Data?

1 row = 1 observation
1 column = 1 variable
1 table = 1 type of information

📌 pivot_longer()

Converts data from wide format to long format by turning multiple columns into key–value pairs.

```
# Used for reshaping data (wide ↔ long), filling missing values.
library(tidyr)
students_long <- students %>%
  pivot_longer(col=c("math_score","english_score"),
               names_to = "subject",
               values_to = "score")
View(students_long)
```

	student_id	name	age	gender	department	attendance	subject	score
1	1001	Student_1	24	F	Computer Science	77	math_score	63
2	1001	Student_1	24	F	Computer Science	77	english_score	69
3	1002	Student_2	21	F	Data Science	73	math_score	77
4	1002	Student_2	21	F	Data Science	73	english_score	77
5	1003	Student_3	28	M	Data Science	93	math_score	54
6	1003	Student_3	28	M	Data Science	93	english_score	57
7	1004	Student_4	25	M	Computer Science	94	math_score	96
8	1004	Student_4	25	M	Computer Science	94	english_score	90
9	1005	Student_5	22	F	Data Science	72	math_score	98
10	1005	Student_5	22	F	Data Science	72	english_score	88

tidyr takes messy data with many columns and reshapes it into one clean, tidy format that's easier to analyse and plot.To convert two columns (math_score, english_score) into one tidy column.

⭐ tidyr takes messy data with many columns and reshapes it into one clean, tidy format that's easier to analyse and plot.

📌 pivot_wider()

Converts data from long format back to wide format by spreading key–value pairs into separate columns.

```
stu_wider <- students_long %>% pivot_wider(names_from =subject,
                                values_from=score)
View(stu_wider)
```

	student_id	name	age	gender	department	attendance	math_score	english_score
1	1001	Student_1	24	F	Computer Science	77	63	69
2	1002	Student_2	21	F	Data Science	73	77	77
3	1003	Student_3	28	M	Data Science	93	54	57
4	1004	Student_4	25	M	Computer Science	94	96	90
5	1005	Student_5	22	F	Data Science	72	98	88
6	1006	Student_6	24	F	Computer Science	72	79	50
7	1007	Student_7	27	F	Mathematics	96	95	52
8	1008	Student_8	20	F	Computer Science	98	54	62
9	1009	Student_9	24	M	Mathematics	87	61	77
10	1010	Student_10	28	F	Computer Science	98	65	98

Data Manipulation with dplyr:

- **dplyr** is the heart of tidy data manipulation.
- Its functions can be grouped into three categories:

1. Operations on ROWS
2. Operations on Columns
3. Grouped Operations

Operations on ROWS:

1. filter() — keep rows based on conditions

```
student_DataScience <- students %>% filter(department=='Data Science')
View(student_DataScience)
```

	student_id	name	age	gender	department	math_score	english_score	attendance
1	1002	Student_2	21	F	Data Science	77	77	73
2	1003	Student_3	28	M	Data Science	54	57	93
3	1005	Student_5	22	F	Data Science	98	88	72
4	1011	Student_11	28	M	Data Science	75	74	95
5	1013	Student_13	22	M	Data Science	97	87	88

Multiple logical expressions:

```
stud_g <- filter(students, attendance > 90, english_score > 90, math_score > 90)
View(stud_g)
```

	student_id	name	age	gender	department	math_score	english_score	attendance
1	1143	Student_143	18	F	Computer Science	91	93	92

2. arrange() — sort rows:

```
## 2. arrange() – sort rows
student_ascmarks <- students %>% arrange(attendance)
View(student_ascmarks)
```

.

	student_id	name	age	gender	department	math_score	english_score	attendance
1	1023	Student_23	23	M	Data Science	85	60	70
2	1050	Student_50	24	F	Mathematics	61	57	70
3	1137	Student_137	28	M	Computer Science	70	75	70
4	1192	Student_192	24	F	Mathematics	59	60	70
5	1197	Student_197	22	F	Data Science	94	60	70
6	1029	Student_29	27	F	Data Science	50	67	71
7	1066	Student_66	25	M	Computer Science	72	62	71
8	1116	Student_116	28	F	Data Science	71	59	71
9	1129	Student_129	24	M	Computer Science	80	98	71
10	1162	Student_162	21	F	Statistics	95	83	71

Computer Science

⭐ Operations on COLUMNS:

select() — choose columns:

```
student_markColumn <- students %>% select(name, english_score, math_score, attendance)
View(student_markColumn)
```

	name	english_score	math_score	attendance
1	Student_1	69	63	77
2	Student_2	77	77	73
3	Student_3	57	54	93

 # To select ALL columns from the dataset students EXCEPT the columns math_score and english_score

```
Stud_Without_scores <- students %>% select(-english_score, -math_score)
View(Stud_Without_scores)
```

	student_id	name	age	gender	department	attendance
1	1001	Student_1	24	F	Computer Science	77
2	1002	Student_2	21	F	Data Science	73
3	1003	Student_3	28	M	Data Science	93
4	1004	Student_4	25	M	Computer Science	94
5	1005	Student_5	22	F	Data Science	72

mutate() — create/modify columns:

```
students_totalMarks <- students %>% mutate(totalMarks = english_score + math_score)
View(students_totalMarks)
```

	student_id	name	age	gender	department	math_score	english_score	attendance	totalMarks
1	1001	Student_1	24	F	Computer Science	63	69	77	132
2	1002	Student_2	21	F	Data Science	77	77	73	154
3	1003	Student_3	28	M	Data Science	54	57	93	111
4	1004	Student_4	25	M	Computer Science	96	90	94	186
5	1005	Student_5	22	F	Data Science	98	88	72	186
6	1006	Student_6	24	F	Computer Science	79	50	72	129
7	1007	Student_7	27	F	Mathematics	95	52	96	147
8	1008	Student_8	20	F	Computer Science	54	62	98	116
9	1009	Student_9	24	M	Mathematics	61	77	87	138
10	1010	Student_10	28	F	Computer Science	65	98	98	163

3. Grouped Operations:

```
group_summarise <- students %>% group_by(department) %>%
  summarise(count=n(),
            avg_math = mean(math_score),
            median_mathscore= median(math_score),
            sd_math = sd(math_score),
            max_math_score = max(math_score),
            min_math_score = min(math_score))
 View(group_summarise)
```

	department	count	avg_math	median_mathscore	sd_math	max_math_score	min_math_score
1	Computer Science	53	71.88679	70.0	13.56276	98	52
2	Data Science	52	75.26923	75.5	14.28988	99	50
3	Mathematics	44	75.15909	77.0	14.00489	98	52
4	Statistics	51	78.90196	82.0	14.54408	99	50