SEN603 · DATA ANALYTICS AND STATISTICS

Week 1

Statistics, Data Science and Python Setup

Assoc. Prof. Onur Polat

Hacettepe University · Systems Engineering M.Sc. (Thesis, English)

Fall 2026 · Lecture 1 of 14

Welcome to SEN603

A practical, code-first course in statistics for engineers

- This is a graduate-level, applied course — every concept is paired with Python code.
- You will leave with a reproducible Colab notebook for every method we cover.
- The course serves your thesis: you will end able to analyse and defend real data.
- Please ask questions in class, on the discussion board, and in office hours.
- Everything (slides, notebooks, data) will be shared on the course Drive folder.

SEN603 at a glance

Course code	SEN603
Programme	Systems Engineering M.Sc. (Thesis) — English
Credits	3 (3-0-3) · ECTS 8
Type	Compulsory · Face-to-face
Language	English
Prerequisites	Basic Python, basic calculus & algebra
Instructor (section)	Assoc. Prof. Onur Polat
Main textbook	Bruce, Gedeck & Dobbins (2020), Practical Statistics for Data Scientists, 2nd ed.

By the end of this course you will...

1. Perform fundamental statistical analyses using Python
2. Create and interpret standard data visualisations
3. Conduct hypothesis tests and understand p-values
4. Build simple regression models and read diagnostics
5. Communicate statistical findings clearly and reproducibly

SYLLABUS MAP

Where we are going — 14 weeks

W1	Statistics, DS & Python setup	**W2**	Data exploration & EDA
W3	Inference basics · permutation, p-values	**W4**	Random numbers & simulation
W5	Probability & the Normal distribution	**W6**	Categorical data · χ^2 tests
W7	MIDTERM	**W8**	Sampling & bootstrap CIs
W9	ANOVA basics	**W10**	Correlation
W11	Simple linear regression	**W12**	Multiple regression
W13–14	Research presentations	**W15–16**	Review & final exam

How our 3-hour lectures run

00:00 – 01:20	**Lecture — Part I** Theory, intuition, worked examples on the board
01:20 – 01:35	**Break ☕** Stretch, questions, informal discussion
01:35 – 02:30	**Lecture — Part II + live Colab demo** Method in action on a real dataset
02:30 – 03:00	**Application + homework brief** You code with me; I hand out that week's HW

How you will be graded

- Weekly HW (13 × ~2%)
- Midterm (Week 7)
- Research presentations
- Final exam (Week 16)

Note Attendance is expected. Late homework: −20%/day, cap 3 days.

What to read

MAIN TEXTBOOK

Bruce, P., Gedeck, P. & Dobbins, K. (2020).

Practical Statistics for Data Scientists — 50+ Essential Concepts Using R and Python (2nd ed.). O'Reilly.

SUPPLEMENTARY

- Wasserman, L. (2004). All of Statistics. Springer.
- McKinney, W. (2022). Python for Data Analysis, 3rd ed. O'Reilly.
- James, Witten, Hastie, Tibshirani (2023). An Introduction to Statistical Learning with Python. Springer.
- VanderPlas, J. (2023). Python Data Science Handbook, 2nd ed. O'Reilly.
- Official docs: pandas, NumPy, SciPy, statsmodels, scikit-learn.

SOFTWARE STACK

What we will install and use

Python 3.11+

Language runtime

Google Colab

Primary notebook environment

numpy

Arrays, vectorised math

pandas

Tabular data manipulation

matplotlib

Base plotting

seaborn

Statistical plots on matplotlib

scipy.stats

Distributions & classical tests

statsmodels

Regression, ANOVA, time series

scikit-learn

Predictive modelling & CV

PART I

Why statistics matters in data science

Slides 11 – 24

CONCEPT

What is data science?

- Data science extracts knowledge and useful decisions from data using code, statistics, and domain expertise.
- It spans data collection, cleaning, exploration, modelling, and communication.
- It is empirical: models are proposed, tested against data, and revised.
- For a systems engineer, it is the toolbox for reasoning under uncertainty from measurements.

Working definition Data science = statistical thinking + programming + domain knowledge, delivered as reproducible analyses.

Statistics, Data Science, Machine Learning

Statistics

Formal inference about populations from samples; quantifies uncertainty.

Data Science

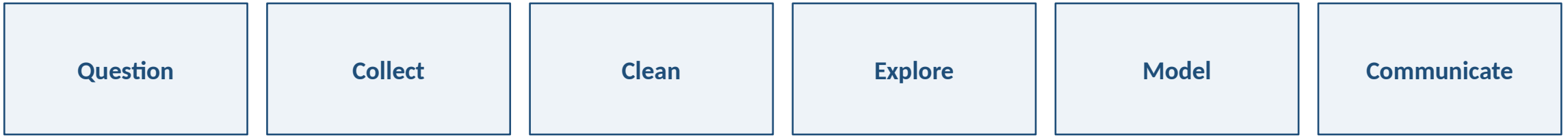
The end-to-end practice: get data → clean → explore → model → communicate.

Machine Learning

Algorithms that learn patterns to predict on new data at scale.

Take-away ML is powerful but blind without statistical thinking. DS glues them together.

The data-science pipeline



Statistics enters at every step:

- Collect: sampling design, bias, sample size.
- Clean: outliers, missingness, measurement error.
- Explore: summaries, distributions, associations.
- Model: inference, prediction, uncertainty quantification.
- Communicate: p-values, CIs, effect sizes — honestly.

Kinds of data problems

Descriptive	What does the data look like? Summaries and visualisations.
Inferential	What can we say about the population from a sample?
Predictive	Given inputs, what output do we expect for new cases?
Causal	Would changing X change Y? Requires experiments or careful design.

Numeric vs. categorical data

Numeric

- Continuous — real values (temperature, current).
- Discrete — integer counts (defects per unit).
- Also called quantitative.
- Support arithmetic operations meaningfully.
- Typical stats: mean, std, correlation.

Categorical

- Nominal — unordered labels (city, sensor id).
- Ordinal — ordered labels (low/med/high).
- Binary — two categories (pass/fail).
- Arithmetic on codes is usually meaningless.
- Typical stats: counts, proportions, χ^2 tests.

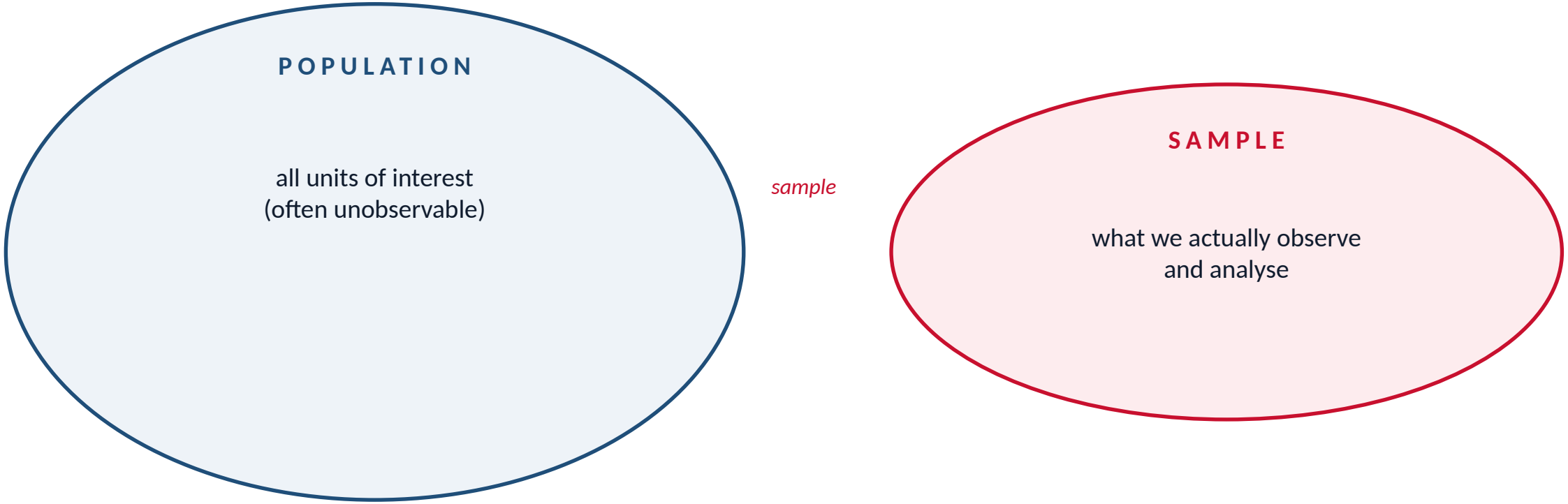
Rectangular data is the default

- A rectangular dataset = rows (records) × columns (features).
- Each row is one observation; each column is one variable of one type.
- In Python, this is a pandas DataFrame.
- Non-rectangular data (text, images, time series, graphs) is often flattened into rectangular form for analysis.

id	temperature	vibration	status
1	72.4	0.31	OK
2	78.1	0.47	OK
3	84.9	0.92	FAIL

CORE CONCEPT

Population vs. sample



Key idea We use sample statistics (like the sample mean) to estimate population parameters (like the true mean).

Random sampling and why it matters

- A simple random sample gives every unit an equal chance of selection.
- Random samples let us quantify uncertainty and generalise honestly.
- Non-random samples can look large yet be systematically misleading.
- In engineering: convenience sampling (only the machines you can access) is a common trap.
- Design of experiments (Week 11 context) is randomisation done deliberately.

Warning Size fixes noise, not bias. A million biased observations still give a biased answer.

Classic pitfalls

Survivorship bias	Analysing only machines still running ignores those that failed early.
Self-selection	Voluntary responses come from atypical users (very happy / very angry).
Convenience	Sampling from the lab bench, not the plant floor.
Non-response	Missing observations are rarely missing at random.

Reproducibility is non-negotiable

- A result you cannot re-run is not a result — it is folklore.
- Every analysis lives in a notebook with: data source, code, environment, and version.
- Set a random seed for any stochastic step.
- Keep raw data untouched; write derived data as a separate file.
- Use version control (Git) even for solo projects — your future self will thank you.

Rule of thumb If a colleague on another machine cannot reproduce your figure in 10 minutes, it is not done.

MINDSET

Think statistically — three habits

1. Quantify uncertainty

Never report a number without asking how sure you are.

2. Show the distribution

Any single summary hides the shape it came from.

3. Beware of stories from noise

Small samples produce beautiful, meaningless patterns.

WHY YOU CARE

Statistics in systems engineering — quick cases

Quality control

Is our defect rate drifting? Control charts + hypothesis tests.

Reliability

How long until failure? Survival analysis + confidence bounds.

A/B testing

Does interface B outperform A? Two-sample tests and effect sizes.

Forecasting

Next month's demand? Time-series models and prediction intervals.

Anomaly detection

Which sensor readings look weird? Distributions + robust statistics.

Model validation

Does my ML model actually generalise? Cross-validation + tests.

WATCH OUT

Six pitfalls we will fight all semester

- Correlation is not causation — cover in Week 10.
- p-hacking — testing many things and reporting only what "worked".
- Overfitting — chasing patterns that will not repeat.
- Ignoring assumptions — running tests without checking them.
- Cherry-picking outliers — deleting inconvenient data.
- Beautiful plots that lie — misleading axes, cropped scales.

PART II

The Python environment — Colab, NumPy, pandas

Slides 25 – 45

Why Python for statistics and data analytics?

- General-purpose language with a huge scientific ecosystem.
- Free, open source, and the current lingua franca of data science.
- Great for the full pipeline: ingestion, cleaning, analysis, ML, deployment.
- First-class notebook experience (Jupyter, Colab) for reproducibility.
- Widely adopted in industry — a skill worth keeping.

What is Colab and why we use it

What it is

- Google's hosted Jupyter notebook.
- Runs in your browser — no install needed.
- Comes with numpy, pandas, matplotlib, scipy, sklearn preinstalled.
- Free tier is enough for this course.
- Notebooks live in your Google Drive.

Why for SEN603

- Zero setup friction — you code from Day 1.
- Easy sharing (a link is enough).
- Every student has the same environment.
- Free GPU/TPU if you want to explore ML later.
- You can download notebooks as .ipynb / .pdf / .py.

This week Open colab.research.google.com and sign in with your Hacettepe Google account.

Anatomy of a Colab notebook

Menu bar

File, Edit, View, Insert, Runtime, Tools, Help.

Toolbar

+Code / +Text buttons to add cells.

Cells

Code cells (Python) and text cells (Markdown).

Runtime

The Python process behind the notebook.

Files pane

Left sidebar: upload, mount Drive, browse files.

Table of contents

Auto-generated from your Markdown headings.

Code cells vs. text cells

Code cell — Python; run with Shift+Enter.

```
# A code cell
import numpy as np
x = np.arange(10)
print(x.mean(), x.std())
```

Text cell — Markdown for notes, equations, and headings.

```
# Week 1 — first analysis
The sample mean is $\bar{x}=\frac{1}{n}\sum_{i=1}^n x_i$.

- Load data
- Summarise
- Plot
```

Tip Alternate code and text cells — treat your notebook as a report, not a script.

Runtimes, sessions, and files

- The runtime is a temporary VM — it disconnects after inactivity.
- When it restarts, variables are lost. Rerun cells from the top.
- Uploaded files live only in this session; save to Drive to keep them.
- Restart runtime after installing a big package.
- GPU/TPU is optional (Runtime → Change runtime type).

Installing and importing packages

```
# Most libraries are preinstalled on Colab; when you need one:
!pip install pmdarima

# Then import as usual:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression

print("numpy:", np.__version__)
print("pandas:", pd.__version__)
```

Convention One import block near the top of every notebook — grouped: standard library, third-party, local.

NumPy — arrays for fast math

- The foundation of pandas, scipy, sklearn, statsmodels.
- Provides ndarray: a homogeneous, n-dimensional array.
- Vectorised operations are C-fast — no Python for-loops.
- Broadcasting = elegant arithmetic across shapes.
- Random number generation, linear algebra, FFT built in.

NumPy essentials

```
import numpy as np

# create arrays
a = np.array([1, 2, 3, 4, 5])
z = np.zeros(5); o = np.ones((2, 3))
r = np.arange(0, 10, 2)          # 0,2,4,6,8
lin = np.linspace(0, 1, 5)       # 0, 0.25, ... 1

# summaries
a.mean(), a.std(ddof=1), a.min(), a.max()

# vectorised math (no loops)
b = a ** 2 + 3

# reproducible random draws
rng = np.random.default_rng(seed=42)
x = rng.normal(loc=0, scale=1, size=1000)
print(x.mean(), x.std())
```

pandas — the DataFrame library

- pandas gives us two workhorses: Series (1-D) and DataFrame (2-D).
- Think of Series as a labelled NumPy array.
- Think of DataFrame as a table — like an Excel sheet or SQL table.
- Both are indexed (labels on rows/columns) — indexing is a superpower.
- Fast, expressive, and the default for tabular data in Python.

The Series

```
import pandas as pd

# A Series = values + index
temps = pd.Series(
    [72.4, 78.1, 84.9, 69.3],
    index=["M1", "M2", "M3", "M4"],
    name="temperature"
)

temps                 # labelled 1-D array
temps["M3"]           # label-based access -> 84.9
temps.mean(), temps.std()
temps[temps > 75]     # boolean filtering
```

The DataFrame at a glance

	machine	temp	vibration	status
0	M1	72.4	0.31	OK
1	M2	78.1	0.47	OK
2	M3	84.9	0.92	FAIL
3	M4	69.3	0.22	OK

row index →

↑ column labels

In one sentence A DataFrame is a dictionary of Series that share the same index.

Ways to build a DataFrame

```
import pandas as pd

# 1) From a dict of lists
df = pd.DataFrame({
    "machine":   ["M1", "M2", "M3", "M4"],
    "temp":      [72.4, 78.1, 84.9, 69.3],
    "vibration": [0.31, 0.47, 0.92, 0.22],
    "status":    ["OK", "OK", "FAIL", "OK"],
})

# 2) From a CSV file
df = pd.read_csv("sensors.csv")

# 3) From a URL
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv"
iris = pd.read_csv(url)
```

Getting a CSV into Colab — three ways

```
# A) From a URL (simplest)
import pandas as pd
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/tips.csv"
df = pd.read_csv(url)

# B) Upload from your computer
from google.colab import files
uploaded = files.upload()          # opens a file picker
df = pd.read_csv(next(iter(uploaded)))

# C) Mount Google Drive (persistent)
from google.colab import drive
drive.mount("/content/drive")
df = pd.read_csv("/content/drive/MyDrive/SEN603/data/sensors.csv")
```

First look at any new dataset

```
df.shape                # (rows, columns)
df.head()               # first 5 rows
df.tail(3)              # last 3 rows
df.info()               # dtypes, non-null counts, memory
df.describe()           # numeric summary
df.describe(include="object")   # summary for categoricals
df.dtypes               # types per column
df.columns.tolist()     # list column names
df.sample(5, random_state=42)   # random 5 rows
```

Habit Run this exact block on every new dataset — before any analysis.

Selecting columns and rows

```
# Column selection
df["temp"]                      # one column -> Series
df[["temp", "vibration"]]       # multiple columns -> DataFrame

# Row selection
df.head(3)                      # first 3 rows
df.iloc[0]                      # 1st row by position
df.iloc[0:5, 1:3]               # rows 0–4, cols 1–2 (positional)
df.loc[0, "temp"]               # by label
df.loc[df["status"] == "FAIL"]  # boolean row filter
```

Rule Use .loc for labels, .iloc for positions — never mix.

Boolean filtering — the workhorse

```
# Single condition
hot = df[df["temp"] > 80]

# Combining conditions (& = AND, | = OR, ~ = NOT)
critical = df[(df["temp"] > 80) & (df["vibration"] > 0.5)]

# .query is often more readable
critical = df.query("temp > 80 and vibration > 0.5")

# isin / between
df[df["status"].isin(["FAIL", "WARN"])]
df[df["temp"].between(70, 85)]
```

Creating and dropping columns

```
# Create a new column from existing ones
df["temp_c"] = (df["temp"] - 32) * 5 / 9
df["risk"]   = df["vibration"] * df["temp"]

# Conditional column
import numpy as np
df["flag"] = np.where(df["temp"] > 80, "hot", "normal")

# Drop columns (axis=1) or rows (axis=0)
df = df.drop(columns=["flag"])
df = df.rename(columns={"temp": "temp_f"})
```

Detecting missing values

```
df.isna().sum()                  # NA count per column
df.isna().mean().round(3)        # NA proportion per column

# quick strategies (we cover more in Week 2)
df.dropna()                      # drop rows with any NA
df.fillna(df.median(numeric_only=True))   # median-impute numerics
```

Preview Missingness handling is a Week 2 topic — for now, just detect and report it.

Split → apply → combine

```
# Average temperature by status
df.groupby("status")["temp"].mean()

# Multiple aggregates
df.groupby("status").agg(
    n=("temp", "size"),
    mean_temp=("temp", "mean"),
    mean_vib=("vibration", "mean"),
)
```

Coming next week GroupBy is the heart of EDA. We will use it heavily in Week 2.

One line for a first plot

```
import matplotlib.pyplot as plt
import seaborn as sns

df["temp"].hist(bins=20)
plt.xlabel("Temperature"); plt.ylabel("Count"); plt.title("Temperature distribution"); plt.show()

# seaborn (nicer defaults)
sns.boxplot(data=df, x="status", y="temp"); plt.show()
```

Deep dive Histograms, boxplots, and outlier plots are the full topic of Week 2.

A minimal end-to-end analysis in ~10 lines

```
import pandas as pd, seaborn as sns, matplotlib.pyplot as plt

url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/tips.csv"
tips = pd.read_csv(url)                          # 1) load
print(tips.shape); print(tips.head())            # 2) inspect
print(tips.describe())                           # 3) summarise
print(tips.groupby("day")["tip"].mean())         # 4) group
sns.boxplot(data=tips, x="day", y="tip")         # 5) visualise
plt.show()
```

This is the shape of every weekly notebook Load → inspect → summarise → group → visualise → interpret.

Application (last 30 minutes)

- Open Colab, create a new notebook titled SEN603_W1_YourName.ipynb.
- Add a Markdown title cell and your name/student number.
- Load the classic tips dataset from a URL (see previous slide).
- Run .info(), .describe(), and .head() and add one sentence of interpretation under each.
- Compute the average tip by day of week using groupby.
- Save the notebook to your Drive under /SEN603/W1/.
- Share the link with the instructor via the course form.

HW1 — Environment check and first EDA

Deliverable

A single Colab notebook, shared with the instructor.

Tasks

- T1. Pick any dataset with ≥ 5 columns and ≥ 200 rows (UCI / Kaggle / seaborn).
- T2. Print shape, dtypes, and a numeric + categorical describe().
- T3. Compute mean, median, and std for two numeric columns.
- T4. Filter and count rows that meet a rule you invent (justify it in one sentence).
- T5. Use groupby on one categorical column and report the group means for one numeric column.
- T6. Write a short Markdown 'Findings' cell (3–5 sentences) summarising what you saw.

Due Before Week 2 lecture. Late policy: −20% / day, capped at 3 days.

AVOID THESE

Beginner traps to avoid

- Running cells out of order — always Runtime → Restart and run all before submitting.
- Hard-coded file paths from your laptop — use a URL or Drive path.
- Reading the CSV with default settings without checking dtypes.
- Silent data loss from df = df.dropna() without noting how many rows disappeared.
- No random seed — results change every run, and no one can reproduce you.
- One giant cell doing everything — split into small, titled cells.

Week 1 — what you should be able to do

- ✓ Explain the role of statistics in data science.
- ✓ Distinguish population, sample, and sampling bias.
- ✓ Distinguish numeric vs. categorical data types.
- ✓ Open Colab and run a Python cell.
- ✓ Load a CSV into a pandas DataFrame from a URL.
- ✓ Inspect a DataFrame with head / info / describe.
- ✓ Select, filter, and group with pandas.

NEXT WEEK

Week 2 — Data Exploration (EDA)

Mean, median, std, percentiles · histograms · boxplots · bar charts · outliers

---

Before Week 2

- Read Bruce et al., Chapter 1 (Exploratory Data Analysis).
- Complete HW1 and share your notebook.
- Bring at least one question about your dataset.

Questions?

onur.polat@hacettepe.edu.tr · Office hours: TBA