ID511M · DATA CURATION AND VISUALIZATION · LABORATORY

The IITs Timetable

Part A · Data Acquisition Part B · The Unified Schema

Collect the timetables of 23 IITs — 23 formats, 23 slot systems — and forge them into one analytical dataset. Acquisition first; schema second: you cannot design what you have not seen.

3-hour session · Department of Mathematics and Statistics · *Panchatcharam Mariappan, Associate Professor*, IIT Tirupati

LABORATORY OUTCOMES

What you will be able to do by 5 pm

Acquire structured data from hostile formats

Pull timetable tables out of live HTML pages, Excel workbooks and PDF documents into pandas — and know which extraction tool fits which format.

Capture provenance at the moment of ingest

Write a machine-readable manifest — source URL, retrieval time, checksum, parser version — so every raw file can be traced and re-fetched.

Design a unified schema from evidence

Derive a single relational schema for course meetings after inspecting 23 incompatible slot systems — and defend each column, type and key.

State and test integrity constraints

Express 'no room hosts two courses at once' as a denial constraint and implement it as an executable validation check on your own data.

THE MISSION

Six institutes, six dialects, one dataset per team

6

IITs assigned

Tirupati · Madras · Jodhpur · Gandhinagar · Hyderabad · Guwahati (Team 1)

3

raw formats

live HTML pages, Excel workbooks, PDF timetables — as published

1

unified schema

a single long table of course meetings that any IIT maps into

0

hand edits

every transformation is code; raw files are never modified

Why this order — acquisition before schema. A schema designed before seeing the data encodes your assumptions; a schema designed after acquisition encodes the evidence. Today you first collect the raw material exactly as published, then let its actual structure — and its actual pathologies — dictate the unified design.

This dataset returns in the wrangling lab (reshaping slot grids) and the visualization labs (occupancy heatmaps, calendar views).

THE MISSION

Six institutes, six dialects, one dataset per team

6

IITs assigned

Palakkad · Roorkee · Ropar · Mandi · Indore · Bombay (Team 2)

3

raw formats

live HTML pages, Excel workbooks, PDF timetables — as published

1

unified schema

a single long table of course meetings that any IIT maps into

0

hand edits

every transformation is code; raw files are never modified

Why this order — acquisition before schema. A schema designed before seeing the data encodes your assumptions; a schema designed after acquisition encodes the evidence. Today you first collect the raw material exactly as published, then let its actual structure — and its actual pathologies — dictate the unified design.

This dataset returns in the wrangling lab (reshaping slot grids) and the visualization labs (occupancy heatmaps, calendar views).

THE MISSION

Six institutes, six dialects, one dataset per team

6

IITs assigned

BHU · ISM Dhanbad · Delhi · Bhilai · Patna · Jammu (Team 3)

3

raw formats

live HTML pages, Excel workbooks, PDF timetables — as published

1

unified schema

a single long table of course meetings that any IIT maps into

0

hand edits

every transformation is code; raw files are never modified

Why this order — acquisition before schema. A schema designed before seeing the data encodes your assumptions; a schema designed after acquisition encodes the evidence. Today you first collect the raw material exactly as published, then let its actual structure — and its actual pathologies — dictate the unified design.

This dataset returns in the wrangling lab (reshaping slot grids) and the visualization labs (occupancy heatmaps, calendar views).

THE MISSION

Six institutes, six dialects, one dataset per team

6

IITs assigned

Kharagpur · Kanpur · Dharwad · Goa · Hyderabad · Bhubaneswar (Team 4)

3

raw formats

live HTML pages, Excel workbooks, PDF timetables — as published

1

unified schema

a single long table of course meetings that any IIT maps into

0

hand edits

every transformation is code; raw files are never modified

Why this order — acquisition before schema. A schema designed before seeing the data encodes your assumptions; a schema designed after acquisition encodes the evidence. Today you first collect the raw material exactly as published, then let its actual structure — and its actual pathologies — dictate the unified design.

This dataset returns in the wrangling lab (reshaping slot grids) and the visualization labs (occupancy heatmaps, calendar views).

SESSION PLAN · 3 HOURS

How the afternoon runs

0:00 — **Briefing & team/IIT assignment**

This deck; inspect your institute's published timetable together.

0:20 — **Part A — Acquisition sprint**

Fetch raw files; parse HTML / Excel / PDF into raw DataFrames; write the provenance manifest as you go.

1:30 — **Schema huddle (whole class)**

Each team reports its slot system in 10 minutes; we negotiate the unified schema on the board.

1:50 — **Part B — Mapping sprint**

Build your institute's slot dictionary; transform raw tables into the agreed schema; run validation checks.

2:45 — **Wrap & submission**

Push notebook + manifest + mapped table; state one constraint your data violates.

PART A

Data Acquisition

Get the timetables onto disk exactly as the institutes publish them — with a paper trail. Raw first, honest always; interpretation comes in Part B.

90 minutes · one IIT per team

The source zoo: same information, three physics

Live HTML pages

- Institute academic portals render the timetable as an HTML <table> (sometimes several, nested).
- Tool: requests + pandas.read_html.
- Pathologies: merged header cells, slot codes as colour, tables split by department, session cookies.

Excel workbooks

- Registrars love one-sheet-per-department workbooks with merged cells and colour-coded slots.
- Tool: pandas.read_excel / openpyxl (openpyxl sees merges and fills; read_excel does not).
- Pathologies: merged ranges become NaN, hidden sheets, formatting-as-data.

PDF documents

- Some institutes publish only a typeset or scanned PDF grid.
- Tool: pdfplumber for text-layer PDFs; flag scanned pages for OCR — do not hand-transcribe.
- Pathologies: columns detected by x-coordinates, line-wrapped course codes, page breaks mid-table.

First act of curation: record which physics your IIT uses — before writing any parser.

Rules of engagement

Scrape like a guest, not a bot

Check robots.txt; identify yourself in the User-Agent; sleep ≥ 2 s between requests; fetch each page once and cache it. A timetable is a few pages — there is no excuse for hammering a server.

Personal data stays out of the shared set

Instructor names and room assignments are personal/operational data even when publicly posted. Keep them in your private raw layer; the shared dataset carries course-and-slot structure only (or pseudonymised instructor IDs).

Respect terms and cite sources

Note each portal's terms of use in your manifest; every derived table must name its source URL and retrieval date. Data without citation is rumour.

Timetables are snapshots

Institutes revise timetables mid-semester. Your manifest's timestamp is not bureaucracy — it is the version identifier of your entire dataset.

The landing zone: raw is immutable

- One directory per institute; raw files are written once and never edited — not even to 'fix one cell'.
- Every fix happens downstream in code, so it is repeatable, reviewable and reversible.
- File names carry institute, semester and retrieval date — the file system is your first metadata store.
- A checksum (SHA-256) freezes each file: if the portal changes tomorrow, you can prove what you saw today.

```
timetable-project/
  raw/
    iit-tirupati/2026-jan/
      timetable_2026-08-01.html
      manifest.json
    iit-bombay/2026-jan/
      tt_all_depts_2026-08-01.xlsx
      manifest.json
  src/            # parsers, one module per IIT
  staging/        # parsed but not yet unified
  curated/        # unified-schema tables (Part B)
  checks/         # validation reports
```

Golden rule: you must be able to delete staging/ and curated/ and rebuild both with one command.

HTML: fetch once, parse from the cached copy

```
import requests, pandas as pd, hashlib, json, time
from pathlib import Path

URL = "https://intranet.iittp.ac.in/timetable/2026-jan"
raw = Path("raw/iit-tirupati/2026-jan"); raw.mkdir(parents=True,
exist_ok=True)

hdrs = {"User-Agent": "ID511M-lab (student@iittp.ac.in)"}
r = requests.get(URL, headers=hdrs, timeout=30)
r.raise_for_status()
html_file = raw / "timetable_2026-08-01.html"
html_file.write_bytes(r.content)          # cache first ...

tables = pd.read_html(html_file)          # ... parse from the cache
print(len(tables), [t.shape for t in tables])
```

- **Save bytes, then parse the file — never re-hit the server while debugging a parser.**
- read_html returns every table on the page; selecting the right ones is a logged decision.
- Multi-row headers: pass header=[0,1] and flatten column tuples yourself.
- If the page needs a login, download manually via browser and record that in the manifest — do not script credentials.

Excel and PDF: where formatting is data

```
# Excel: merged cells hide the slot grid
import openpyxl
wb = openpyxl.load_workbook(x, data_only=True)
ws = wb["CSE"]
for rng in ws.merged_cells.ranges:
    v = ws.cell(rng.min_row, rng.min_col).value
    for row in ws[rng.coord]:
        for cell in row:
            merged[cell.coordinate] = v
# then rebuild the grid before DataFrame-ing
```

```
# PDF: tables live in the text layer
import pdfplumber
with pdfplumber.open(p) as pdf:
    rows = []
    for page in pdf.pages:
        t = page.extract_table()
        if t: rows += t
df = pd.DataFrame(rows[1:], columns=rows[0])

# scanned pages -> extract_table() is None:
# STOP, flag for OCR, log it. No hand typing.
```

Merged cells and colour codes are meaning encoded as formatting. Your parser must translate that meaning into values — and your notebook must say where it did so, because that translation is the first place errors enter the pipeline.

The manifest: your dataset's birth certificate

```
{
  "institute": "IIT Tirupati",
  "semester":  "2026-Jan",
  "source_url": "https://intranet.iittp.ac.in/...",
  "retrieved_utc": "2026-08-01T09:12:44Z",
  "retrieved_by": "team-3",
  "method": "requests 2.32; cached to disk",
  "files": [{
    "name": "timetable_2026-08-01.html",
    "sha256": "9f2c...ab41",
    "bytes": 184223
  }],
  "terms_note": "internal academic use",
  "anomalies": ["PH slots table missing Sat row"]
}
```

- **Written at retrieval time, not reconstructed later — memory is not provenance.**
- sha256 lets anyone verify the raw file is untouched: hashlib.sha256(path.read_bytes()).hexdigest().
- anomalies is your field notebook: record oddities the moment you see them.
- In PROV-DM terms (Lecture 2): the file is an Entity, your fetch an Activity, the team an Agent. This JSON is that graph, flattened.

PART A · GATE

Acquisition gate — clear it before touching schema work

- [ ] Raw files on disk, byte-identical to source (checksums recorded)
- [ ] manifest.json complete for every file, anomalies included
- [ ] Parser produces a raw DataFrame per department/page — no value edits
- [ ] Instructor names / rooms separated from the shareable layer
- [ ] One paragraph in the notebook: this IIT's slot-system dialect, described in your own words

That last paragraph is Part B's raw material — write it while the pain is fresh.

PART B

The Unified Schema

Six slot dialects become one relational design. The schema is negotiated as a class, from the evidence each team just collected — then every team maps its institute into it.

75 minutes · huddle, then mapping sprint

The slot-system zoo (why a naive union fails)

Letter slots

IIT Madras style

- A course occupies slot "A": a named bundle of meetings, e.g. Mon 8:00 + Wed 12:00 + Thu 11:00.
- One code ⇒ many (day, time) pairs; the mapping lives in a separate legend table.

Numbered periods

Grid style

- Day × period grid: P1 = 09:00–09:50, P2 = 10:00–10:50 …
- Cell content = course code; a 2-period lab spans P4–P6 as a merged cell.

Hybrid / half-slots

IIT Hyderabad & others

- Letter slots plus half-slots (A1/A2), dedicated lab slots, and tutorial hours attached to a parent slot.
- Same letter ≠ same times across institutes — "A" is a local name.

The unifying insight: every dialect, however exotic, is a compressed encoding of the same atomic fact — ***a course meets on a day, from a start time to an end time, in a room, in a mode (L/T/P).*** Decompress each dialect to atoms and the union is trivial. That atom is our schema's row.

meetings — one row per course-meeting atom

Column	Type	Scale (L1!)	Example	Note
`institute`	str (enum)	nominal	`IITT`	controlled vocabulary, 6 values
`semester`	str	nominal	`2026-Jan`	part of the natural key
`course_code`	str	nominal	`ID511M`	as published, uppercased
`component`	enum	nominal	`P`	L / T / P — lecture, tutorial, practical
`day`	int 0–5	ordinal (cyclic)	`0`	0 = Monday; ISO order
`start_time / end_time`	time	interval	`14:00 / 16:50`	24 h, institute-local; durations are ratio
`room_id`	str	nominal	`R-A21h`	pseudonymised in shared layer
`native_slot`	str	nominal	`"P4-P6"`	the dialect's own code — never discard it
`source_file`	str	—	`tt_...xlsx`	foreign key into the manifest

Natural key: (institute, semester, course_code, component, day, start_time). native_slot preserves the original encoding — unification must be lossless, or it is destruction.

Slot dictionaries: decompress the dialect

```
# One legend per institute: slot code -> atoms
SLOTS_IITT = {
  "A":  [(0,"08:00","08:50"), (2,"12:00","12:50"),
         (3,"11:00","11:50")],
  "A1": [(0,"08:00","08:50")],          # half-slot
  "LAB-M2": [(0,"14:00","16:50")],      # 3 h practical
}

def decompress(course, code, legend):
    for day, t0, t1 in legend[code]:
        yield dict(course_code=course, day=day,
                   start_time=t0, end_time=t1,
                   native_slot=code)

meetings = pd.DataFrame(
    m for c, code in offerings
      for m in decompress(c, code, SLOTS_IITT))
```

- **The legend is data, not code logic — store it as a table so a non-programmer can audit it.**
- Build the legend from the institute's published slot chart, and cite that chart in the manifest.
- Grid dialects go the other way: melt the day × period grid to long form, then map periods to clock times.
- Unknown code ⇒ raise, log, continue — an unmapped slot is a finding, not an exception to swallow.

Constraints: the schema's laws, as executable checks

- Domain constraints — day ∈ {0..5}; end_time > start_time; component ∈ {L,T,P}.
- Key constraint — the natural key is unique: duplicated rows mean a parser bug or a real timetable clash.
- Denial constraint (room clash) — no two rows may share (institute, semester, day, room_id) with overlapping [start, end).
- Cross-table constraint — every native_slot appears in the institute's legend; every source_file appears in the manifest.
- **Violations are reported, never silently dropped — some 'violations' are true facts (shared sessions, cross-listed courses).**

```
def room_clashes(df):
    g = df.sort_values(["institute","day",
                        "room_id","start_time"])
    same = (["institute","semester",
             "day","room_id"])
    prev = g.groupby(same).shift()
    clash = prev.end_time > g.start_time
    return g[clash.fillna(False)]

viol = room_clashes(meetings)
viol.to_csv("checks/room_clashes.csv")
print(len(viol), "potential clashes")
```

Preview of Lecture 7: this is a denial constraint, and 'which cells to change to restore consistency' is exactly the Fellegi–Holt problem.

SUBMISSION

Deliverables and marking (10 marks)

3	**Raw layer + manifest**	Untouched raw files, complete manifests with checksums and anomaly notes.
3	**Parser notebook**	Reproducible end-to-end: raw → staging → curated meetings table; decisions logged in prose cells.
2	**Unified table + legend**	Your institute's meetings in the agreed schema; slot legend stored as data with its source cited.
2	**Validation report**	All constraint checks run; violations listed and each one classified: parser bug, source error, or true fact.

Bonus (+1): document a genuine ambiguity in your institute's published timetable that the class schema cannot represent — and propose the schema change.

WHERE THIS DATASET GOES NEXT

You now own a dataset nobody else has

Integration lab

All six institutes union into one table; cross-institute conflicts become the record-linkage exercise.

Wrangling lab

Reshape meetings into slot-occupancy grids, per-department load tables, free-slot finders.

Visualization labs

Occupancy heatmaps, calendar small-multiples, cross-IIT comparisons of teaching-hour distributions.

Before you leave: state, in one sentence, the estimand your team's first visualization will target.