



Health & Nutrition, Health Science

AD07: Behind the Blind – Enhancing Clinical Data Management with Python

Health Inspired,
Quality Driven.



Disclaimer and disclosures

The views and opinions expressed in this presentation are those of the author(s) and do not necessarily reflect the opinions of PHUSE members, respective companies or organizations.

Agenda

01 Introduction

02 Challenges Behind the Blind

03 The Right Tool For The Job

04 Embracing Python in SDTM
Generation

05 Future Direction

01 Introduction





Clinical Trial SDTM Programming

SDTM programming is encountering numerous challenges:

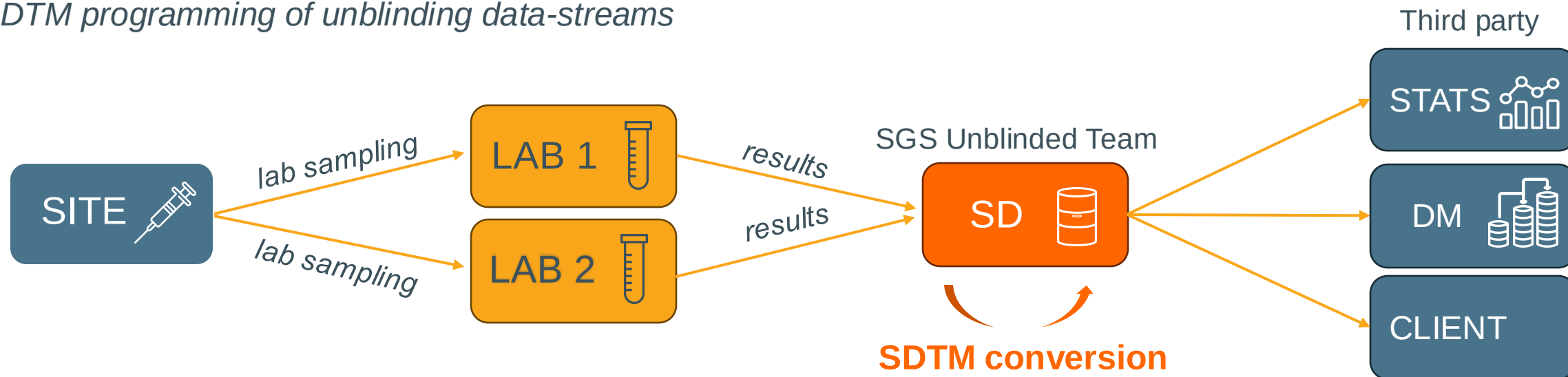
- Complex or variable study designs
- Large data
- Data validation
- CDISC standards
- Regulatory requirements
- Tight timelines



02 Challenges Behind the Blind

SGS Secure Data Team

Separate unblinded team within SGS data management
SDTM programming of unblinding data-streams



Lab samples

- PK samples (blood, stool, urine)
- Blood samples for biomarker analysis or immunogenicity
- ...

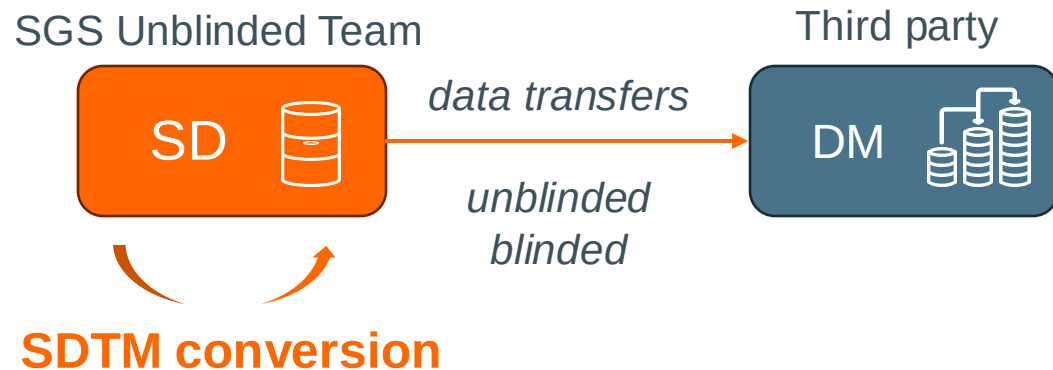
Data files in different structures and formats

- csv
- sas7bdat
- ...

SDTM datasets

- PC and PP SDTM dataset
- LB and IS SDTM dataset
- ...

Secure Data Transfer Types



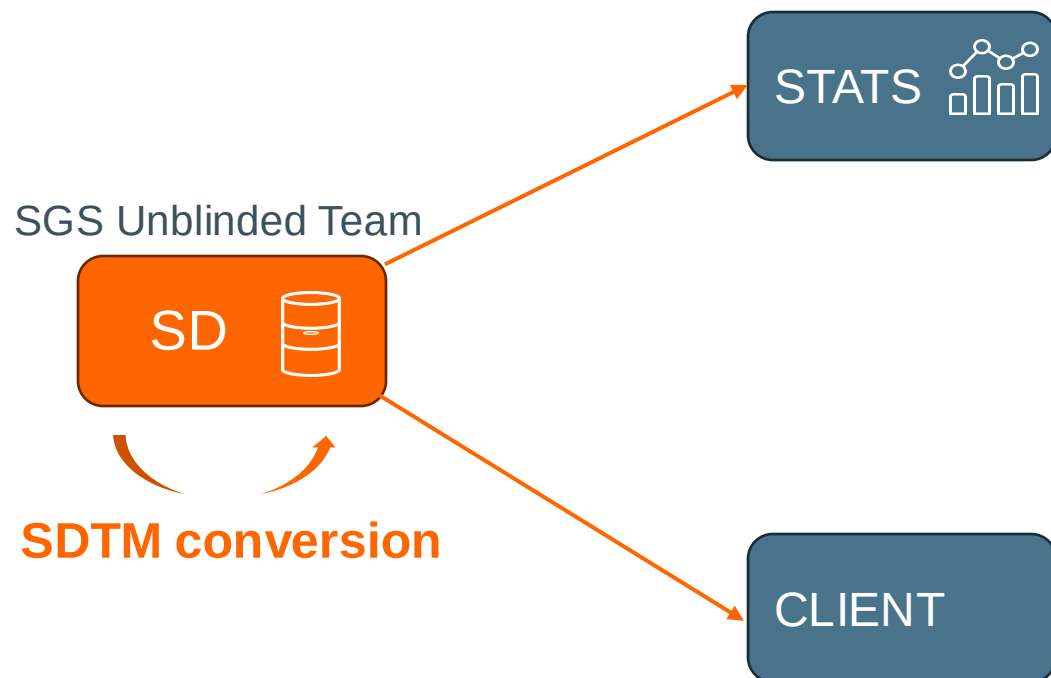
Cleaning Transfers:

- External vendor (lab) data
- Blinded
- Sample identifier
- Reconciliation against eCRF
- Ensure lab data correctness

Result Transfers:

- External vendor (lab) data
- Unblinded
- Combination of operational eCRF data and lab results
- Data for (interim) DBL
- DM CRO partner to add to full SDTM package

Secure Data Transfer Types



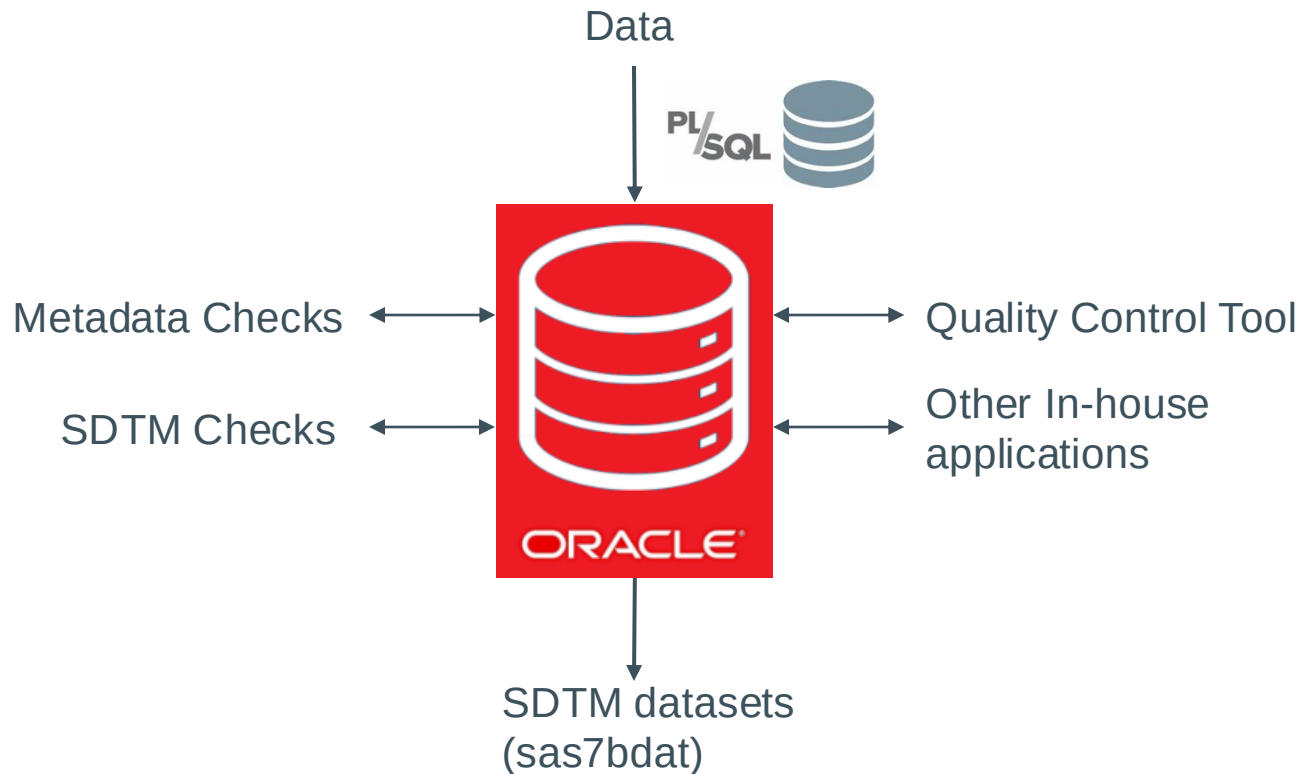
Preliminary Result Transfers:

- SDTM sent to 3rd parties
- PK analysis
- Statistical review
- Blinding or unblinding
- Scrambling/ Dummy subject IDs

Other Unblinded Transfers:

- Other unblinding SDTM domains e.g.:
 - DA (Drug Accountability)
 - DM (Demographics)
 - EX (Exposure)
 - SE (Subject Elements)
- Based on:
 - Randomization list
 - EDC
 - Subject or visit treatment reports

Old SDTM Conversion Using PL/SQL



- 1. Data loaded into database**
 - EDC data
 - External vendor data
 - Other data
- 2. Transformed using PL/SQL**
- 3. Stored in Oracle ® database**
- 4. Oracle database interacts with in-house applications**
 - Metadata
 - SDTM conformance
 - Quality Control
 - Other
- 5. SDTM datasets exported in sas7bdat format**

Challenges in Current PL/SQL Implementation

SDTM programming challenges:

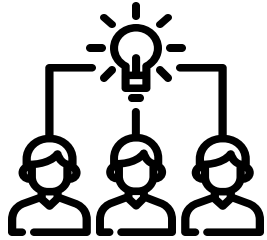
- Low versatility
- Long run-times for complex studies
- Fast turn-around
- Spaghetti-code
- Handling complexity





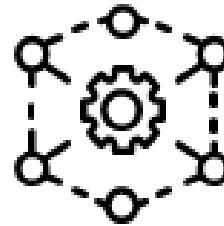
03 The Right Tool for the Job

Python Open-Source Programming



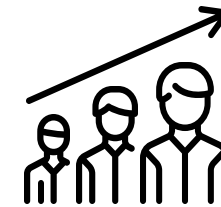
Open-Source Libraries

- Strong Data Handling capabilities
- Pandas and Numpy
- Easy to process, merge and transform data
- Support data validation and metadata management



Flexibility for customization

- Flexibility, customization and system integration
- Development of custom logic
- Complex business rules and custom transformation
- Integration with databases, file formats, platforms



Active Community

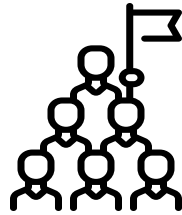
- Growing ecosystem and active community
- Continuous support through updates
- New libraries and tools

Python Open-Source Programming



Regulatory Compliance

- Support regulatory compliance
- Logging
- Version control
- Audit trail capabilities



Popular

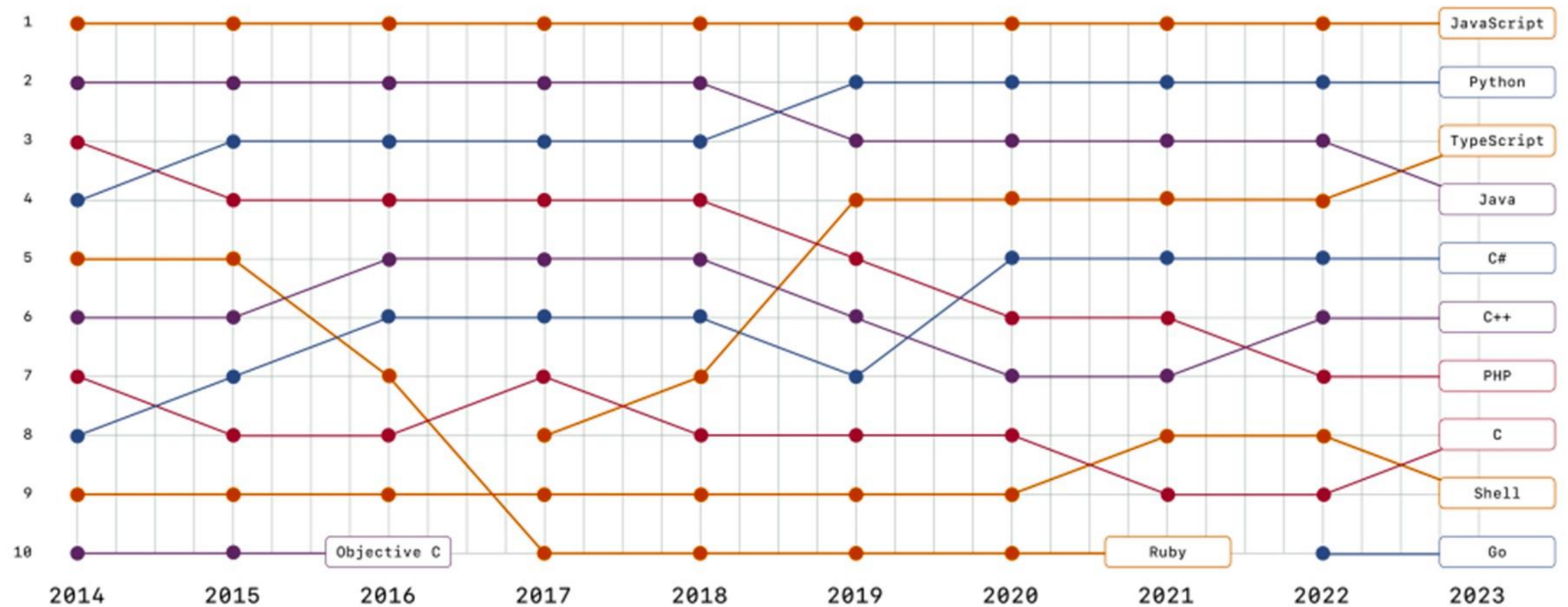
- One of most used languages worldwide
- Attractive from hiring perspective



Free to Use

- Cost effective alternative to other proprietary software

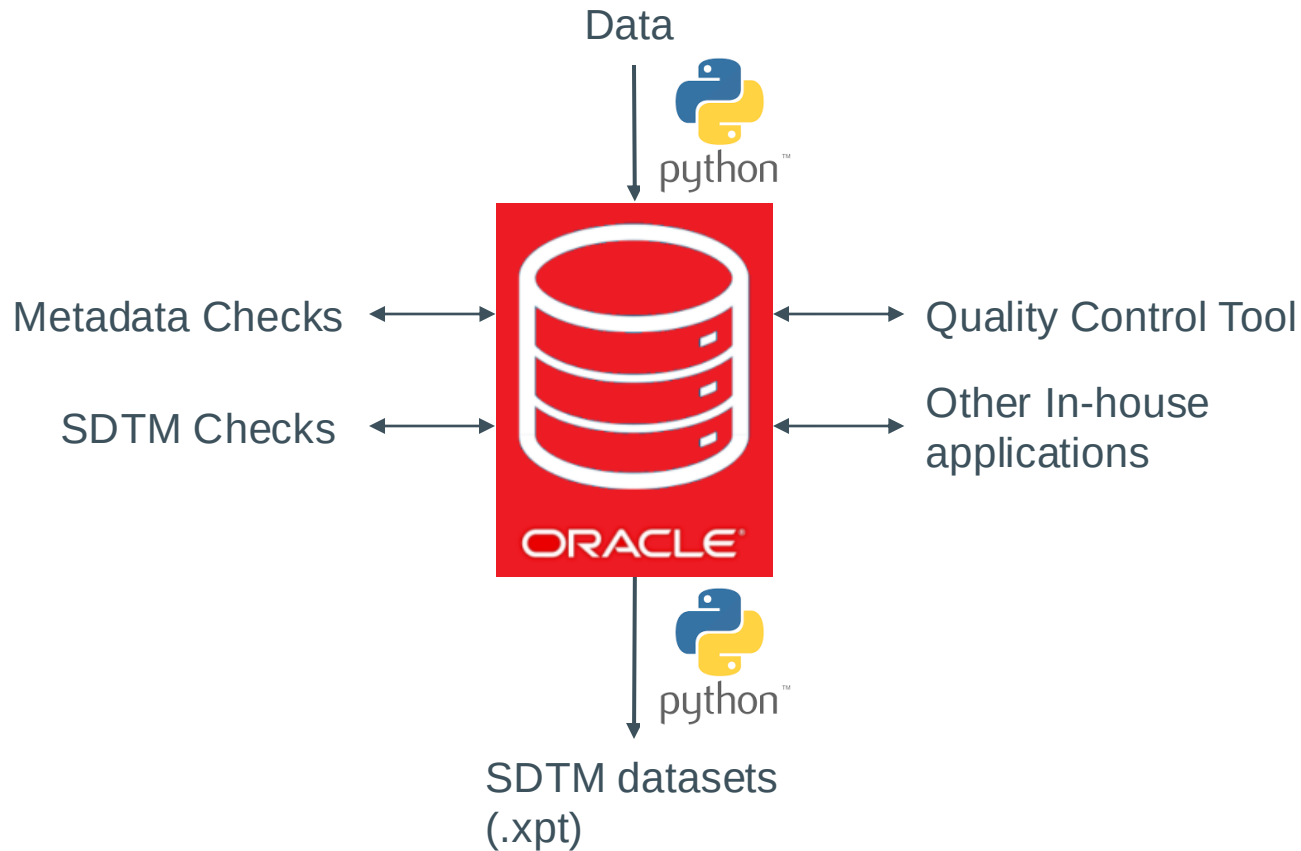
Top 10 programming languages on GitHub





04 Embracing Python in SDTM generation

New SDTM Conversion Using Python: Pilot Projects



1. Data loaded into database
2. Transformed **using Python**
3. Stored in Oracle ® database
4. Oracle database interacts with in-house applications
 - **Integration into current processes**
 - Metadata Checks
 - SDTM conformance
 - Quality Control
 - Other
5. SDTM datasets **exported using Python** in xpt format
 - **Low impact on current processes**
 - **Flexibility, efficiency, faster runtime of PyConv (Python Conversion)**



Python Practical Implementation

- **Visual Studio Code**

- Code editor

- **Ipython notebooks**

- Programming
- Pandas

- **Devops**

- Code management
- Code review



Visual Studio Code

- Code editor
- Extensions
- Language support
- PEP8 linting
- Type checking

Mapping

Use `map.get_mapping_code()` to get the mapping code

Please make sure `SOURCEID`, `LOAD_USER`, `LOAD_DATE`, `LOAD_FILE`, `LOAD_FOLDER` are

Helpful dict

```
CL_DAY = {"Day 1": 1,
          "Day 2": 2,
          "Day 3": 3
        }

CL_REASND = {"NA": "NOT ANALYZED",
            "NR": "NO RESULT",
            "NS": "NO SAMPLE",
            "SL": "SAMPLE LOST",
            "": "NOT ANALYZED"}

CL_USUBJID = dict(zip(help_randnr["QVAL"], help_randnr["USUBJID"]))
```

PC MAPPING

```
mod_dict = {}
keys = {}

for table in orig_files:
    # Keys
    keys[table] = ["USUBJID", "PCTESTCD", "VISIT", "PCTPT"]
    # Filters
    time_99 = orig_files[table]["TIME"].str.contains("99.", na=False)

    mod_dict[table] = pd.DataFrame(
        {
            "STUDYID": orig_files[table]["STUDYID"],
            "DOMAIN": "PC",
            "USUBJID": orig_files[table]["SUBJID"].astype("string").map(CL_USUBJID),
            "PCSEQ": np.NaN,
            "PCTESTCD": orig_files[table]["TEST"].map(CL_PCTESTCD),
            "PCTEST": orig_files[table]["TEST"],
            "PCCAT": "ANALYTE",
            "PCORRES": np.select(
```

Ipython notebooks

- Loading and transformation of EDC and external vendor data into SDTM format
- Code logically built-up cell by cell
- Multiple notebooks per trial, called 'cycles'
- Client-specific templates used
- Clear markup
- PEP8 style guide
- Own developed python packages
 - Pre-defined functionalities
 - Derivations
 - Copy mappings between study and transformation logic



Devops Integration

- Development life cycle tools
- Devops repositories used for trial code
- Streamline code review
- Version control
- Validation processes
- Pull requests
- For each trial, 3 branches are created:
 - Setup
 - QC
 - Live
- Code review when moving between branches

Other Key Improvements

- **XPT Dataset Creation**

- Python can read and generate XPT files (the FDA-compliant file format)
- Removes reliance on commercial third-party software

- **Data Validation**

- Validation tool programmed in Python
- Structural validation
- Content validation

- **Traceability and Source Tracking**

- Track source of data through logging
- Audit trails
- Maintain compliance with regulatory standards

05 Future Direction





Future Direction

- Full implementation
- Expanding SGS Python packages
- Automation of SDTM mapping
- Validation using CDISC Open Rules + Engine (CORE)
- Wearable and sensor data
- Azure cloud migration
- AI and machine learning



Questions?

Contact us

clinicalresearch@sgs.com

EUROPE: +32 15 27 32 45
AMERICAS: + 1 877 677 2667

www.sgs.com/cro

health-nutrition@sgs.com
www.sgs.com/healthnutrition

