toxvaldbstage

April 7, 2025

Type Package

Title Builds the ToxValDB v9.6.1 Stage Database

Version 9.6.1

Author Taylor Wall

Maintainer Taylor Wall <`wall.taylor@epa.gov`>

Description
ToxValDB is a database containing quantitative records from in vivo toxicology studies from many sources. The database has 2 main parts - toxval_source containing
source data in separate tables, and the main toxval schema which combines data from multiple sources
into a single format. This package moves data from toxval_source to toxval.
Data is read from files or other databases into toxval_source and then pulled
into toxval where terms are converted to standard values. This version is setup to build ToxValDB v9.6.1.

Imports DBI,
RMySQL,
openxlsx,
dplyr,
tidyr,
stringr,
tibble,
janitor,
XML,
miniUI,
RCurl,
gsubfn,
textclean,
data.table,
digest,
httr,
jsonlite,
magrittr,
methods,
purrr,

readr,
readxl,
stringi,
tidyselect,
writexl

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests knitr,
rmarkdown

VignetteBuilder knitr

Contents

`cas_checkSum` *cas_checkSum*

Description

Check CAS RN validity via checksum method

For a suspected CAS RN, determine validity by calculating final digit checksum

Usage

```
cas_checkSum(x, checkLEN = TRUE)
```

Arguments

`x`	chr. Input vector of values to check. Standard CAS notation using hyphens is fine, as #' all non-digit characters are stripped for checksum calculation. Each element of *x* should contain #' only one suspected CAS RN to check.
`checkLEN`	logi. Should the function check that the non-digit characters of *x* are at least 4, but no #' more than 10 digits long? Defaults to TRUE. #'

Details

This function performs a very specific type of check for CAS validity, namely whether the final digit checksum follows the CAS standard. By default, it also ensures that the digit length is compatible with CAS standards. It does nothing more.

This means that there is no check for valid CAS format. Use the `cas_detect` function to check CAS format beforehand, or write your own function if necessary.

Value

A `logical` vector of length *x* denoting whether each *x* is a valid CAS by the checksum method. `NA` input values will remain `NA`.

Note

This is a vectorized, reasonably high-performance version of the is.cas function found in the webchem package. The functionality encompasses only the actual checksum checking of `webchem::is.cas`; as mentioned in `details`, use `cas_detect` to recreate the CAS format + checksum checking in `webchem::is.cas`. See examples.

Short of looking up against the CAS registry, there is no way to be absolutely sure that even inputs that pass the checksum test are actually registered CAS RNs. The short digit length of CAS IDs combined with the modulo 10 single- digit checksum means that even within a set of randomly generated validly-formatted CAS entities, ~10% will pass checksum.

See Also

`str_detect`, `str_pad`

Examples

```
cas_good <- c("71-43-2", "18323-44-9", "7732-18-5") #benzene, clindamycin, water
cas_bad  <- c("61-43-2", "18323-40-9", "7732-18-4") #single digit change from good
cas_checkSum(c(cas_good, cas_bad))
```

`chem.check.v2` *chem.check.v2*

Description

Check the chemicals from a file Names with special characters are cleaned and trimmed CASRN are fixed (dashes put in, trimmed) and check sums are calculated The output is sent to a file called chemcheck.xlsx in the source data file One option for using this is to edit the source file until no errors are found

Usage

```
chem.check.v2(res0, source = NULL, verbose = FALSE)
```

Arguments

`res0`	The data frame in which chemicals names and CASRN will be replaced
`source`	The source to be processed. If source=NULL, process all sources
`verbose`	If TRUE, print diagnostic messages

Details

DETAILS

Value

Return a list with fixed CASRN and name and flags indicating if fixes were made: res0=res0,name.OK=name.OK,casrn.OK=c

See Also

[stri_escape_unicode][stringi::stri_escape_unicode] [str_replace_all][stringr::str_replace_all], [str_squish][stringr::str_squisl
[rowwise][dplyr::rowwise], [mutate][dplyr::mutate], [ungroup][dplyr::ungroup], [filter][dplyr::filter],
[select][dplyr::select], [rename][dplyr::rename], [distinct][dplyr::distinct] [separate][tidyr::separate]
[write_xlsx][writexl::write_xlsx]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

clean.last.character *clean.last.character*

Description

Clean unneeded characters from the end of a string

Usage

```
clean.last.character(x)
```

Arguments

x	String to be cleaned

Details

DETAILS

Value

The cleaned string

See Also

`str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`convert.fields.to.json` *convert.fields.to.json*

Description

Combine non-ID columns from audit table into JSON format for audit storage

Usage

```
convert.fields.to.json(in_dat)
```

Arguments

`in_dat`	data to translate to JSON format

Details

DETAILS

Value

Values in JSON format

See Also

`summarise`, `select`, `bind toJSON, fromJSON`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`create_source_table_SQL` *create_source_table_SQL*

Description

Input source data is used to generate the SQL for the source's toxval_source table. SQL is based off a generic SQL file

Usage

```
create_source_table_SQL(
  source,
  res,
  src_version,
  db,
  do.halt = TRUE,
  verbose = FALSE
)
```

Arguments

`source`	name of the source being processed
`res`	input dataframe of source data
`src_version`	Version date of the source
`db`	version of toxval_source to use
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information

Details

DETAILS

Value

New SQL table as a tibble

See Also

`str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

DAT.pipe.source.audit *DAT.pipe.source.audit*

Description

Processes DAT QC audit information into database

Usage

```
DAT.pipe.source.audit(
  source_table,
  db,
  live_df,
  audit_df,
  hashing_type = "vectorized"
)
```

Arguments

`source_table`	name of ToxVal source table audit information is associated with
`db`	the name of the database
`live_df`	a filepath to the DAT live data to push to the 'source' table
`audit_df`	a filepath to the DAT audit data to push to source_audit #'
`hashing_type`	character string of 'vectorized' or 'base' representing which source_hash generation approach to use for the data based on how the original source was hashed.

Details

DETAILS

Value

None

See Also

`read_excel` `rename`, `mutate`, `mutate-joins`, `select`, `filter` `reexports` `write_xlsx`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`doc_lineage_sync_clowder_metadata`
doc_lineage_sync_clowder_metadata

Description

Utility script to sync the Clowder metadata to the database based on Clowder ID

Usage

```
doc_lineage_sync_clowder_metadata(
  source_table,
  db,
  clowder_url,
  clowder_api_key,
  batch_size = 100,
  dsID = "5e31dc1e99323f93a9f5cec0",
  clowder_id_list = NULL
)
```

Arguments

`source_table`	The source table name (e.g. source_test)
`db`	the name of the database
`clowder_url`	URL to Clowder
`clowder_api_key`	API key to access Clowder resources
`batch_size`	PARAM_DESCRIPTION, Default: 100
`dsID`	Clowder Dataset ID
`clowder_id_list`	Optional input list of Clowder IDs to update.

Details

DETAILS

Value

Clowder metadata

See Also

GET, add_headers, content toJSON, fromJSON nest, reexports filter, mutate, select, bind keep str_replace, str_trim

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

export_chemicals_to_curate
export_chemicals_to_curate

Description

Export XLSX files by source for chemical curation

Usage

```
export_chemicals_to_curate(db, export_all = FALSE)
```

Arguments

db	Version of toxval_source to use
export_all	Whether to export all chemicals, Default: FALSE

Details

DETAILS

Value

None

See Also

[separate][tidyr::separate] [bind_rows][dplyr::bind_rows], [filter][dplyr::filter], [select][dplyr::select], [mutate][dplyr::mutate], [rowwise][dplyr::rowwise], [case_when][dplyr::case_when], [ungroup][dplyr::ungroup], [group_split][dplyr::group_split] [read_xlsx][readxl::read_xlsx] [write_xlsx][writexl::write_xlsx]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

fix.casrn *fix.casrn*

Description

Fix a CASRN that has one of several problems

Usage

```
fix.casrn(casrn, cname = "", verbose = FALSE)
```

Arguments

`casrn`	Input CASRN to be fixed
`cname`	An optional chemical name
`verbose`	if TRUE, print the input values

Details

DETAILS

Value

the fixed CASRN

See Also

`reexports`

Examples

```
## Not run:
if(interactive()){
 fix.casrn("107028")
 # Expected output "107-02-8"
 }

## End(Not run)
```

`fix.non_ascii.v2` *fix.non_ascii.v2*

Description

Flag and fix non-ascii characters in the database

Usage

```
fix.non_ascii.v2(df, source)
```

Arguments

`df`	The dataframe to be processed
`source`	Current ToxVal source
`The`	source to be fixed

Details

DETAILS

Value

The dataframe with non ascii characters replaced with cleaned versions

See Also

`read.xlsx`, `write.xlsx` `str_trim` `stri_escape_unicode`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`fix.replace.unicode` *fix.replace.unicode*

Description

A function to check all character fields and handle unicode symbols, either by removing them or replacing them with alphabetic equivalents.

Usage

```
fix.replace.unicode(df)
```

Arguments

`df`	Character vector to check/replace unicode symbols.

Details

DETAILS

Value

Returns a modified version of the input vector with unicode replacements.

See Also

[stri_escape_unicode][stringi::stri_escape_unicode] [str_extract][stringr::str_extract]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`get.chemical.info.by.source` *get.chemical.info.by.source*

Description

get chemical info from source db tables

Usage

```
get.chemical.info.by.source(source.db, source_table, source, file_id)
```

Arguments

`source.db`	The version of toxval source to use.
`source_table`	The name of toxval source table to use.
`source`	The name of toxval source to use.
`file_id`	The suffixed 5 digit identifiers specified in the file names in the folder ./chemical_mapping/source_chemical_files

Details

DETAILS

Value

database info collected

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

get.chemical.info.by.source.combined

get.chemical.info.by.source.combined

Description

get chemical info from source db tables for curation, create chemical table to map curated chemicals to.

Usage

```
get.chemical.info.by.source.combined(source.db, source_table, source)
```

Arguments

`source.db`	The version of toxval source to use.
`source_table`	The name of toxval source table to use.
`source`	The name of toxval source to use.

Details

DETAILS

Value

database info collected

See Also

`bind`, `mutate`, `context`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

get.num.decimal.count *get.num.decimal.count*

Description

A function count the length of an input numeric and its decimal places

Usage

```
get.num.decimal.count(in_num)
```

Arguments

in_num	PARAM_DESCRIPTION

Details

DETAILS

Value

Returns a dataframe of the length of the numeric and decimal places

See Also

str_count

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

getDBConn *getDBConn*

Description

Get the names the database server, user, and pass or returns error message

Usage

```
getDBConn()
```

Details

DETAILS

Value

print the database connection information

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

import.driver *import.driver*

Description

Function to run all import scripts to fill toxval_source

Usage

```
import.driver(db, chem.check.halt = FALSE, do.clean = FALSE)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	PARAM_DESCRIPTION, Default: FALSE
`do.clean`	If TRUE, delte data from all tables before reloading
`chem.chek.halt`	If TRUE and there are bad chemical names or casrn, #' stop to look at the results in indir/chemcheck.xlsx

Details

DETAILS

Value

None

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import.dup.log.info` *FUNCTION_TITLE*

Description

FUNCTION_DESCRIPTION

Usage

```
import.dup.log.info(db, dups_log_file = "True_Duplicate_for_DAT.xlsx")
```

Arguments

`db`	The version of toxval into which the source info is loaded.
`dups_log_file`	The name of the duplicates log file to load

Details

#' Data Profiling Dups Log Load Source Info into toxval source. The information is in the file ./data_profile/data_profile_files/data_profiling_dups_log3.xlsx

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

[read_xlsx][readxl::read_xlsx] [separate_rows][tidyr::separate_rows], [unite][tidyr::unite] [filter][dplyr::filter], [select][dplyr::select]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_actor_source` *FUNCTION_TITLE*

Description

Extract ACToR1 data to toxval source

Usage

```
import_actor_source(toxval.db, infile, filepath, verbose = F)
```

Arguments

`toxval.db`	The version of toxval source into which the tables are loaded.
`infile`	The input file ./ACToR replacements/ACToR_2021/assay_table_hazard prioritized for use.xlsx
`filepath`	The path for all the input xlsx files ./ACToR replacements/ACToR_2021
`verbose`	Whether the loaded rows should be printed to the console.
`do.init`	if TRUE, read the data in from the res_actor_2021q4 database and set up the matrix

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

`read.xlsx`, `write.xlsx` `str_replace` `group_by`, `select`, `mutate-joins`, `mutate_all` `spread` `aggregate`, `na.fail`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_atsdr_pfas_2021_source` *import_atsdr_pfas_2021_source*

Description

Load ATSDR PFAS 2021 data to toxval_source

Usage

```
import_atsdr_pfas_2021_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to ToxVal

See Also

`read_excel` `remove_empty` `bind_rows`, `summarise`, `mutate`, `arrange`, `distinct`, `case_when`, `select`, `group_by` `separate_rows`, `separate`, `pivot_longer`, `drop_na` `str_trim`, `str_extract`, `str_replace` `enframe` `all_of`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_copper_source *import_copper_source*

Description

Load Copper Manufacturers data into toxval_source

Usage

```
import_copper_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `rename`, `mutate`, `row_number`, `case_when`, `filter`, `select`, `bind_rows`, `distinct` `drop_na`, `separate` `str_extract`, `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_dod_meg_source` *import_dod_meg_source*

Description

Load DOD MEG to toxval_source.

Usage

```
import_dod_meg_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None. Data is processed into the database

See Also

`read_excel` `mutate`, `case_when`, `distinct` `str_trim` `unite`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_doe_benchmarks_source`
import_doe_benchmarks_source

Description

Load DOE Wildlife Benchmarks data into toxval_source

Usage

```
import_doe_benchmarks_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `mutate`, `row_number`, `select`, `rename`, `bind_rows`, `across`, `case_when`, `distinct` `str_trim`, `str_extract` `reexports`, `pivot_longer`, `drop_na`, `separate`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_doe_pac_source` *import_doe_pac_source*

Description

Load DOE Source into toxval_source

Usage

```
import_doe_pac_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE, stop if there are problems with the chemical mapping
`do.reset`	PARAM_DESCRIPTION, Default: FALSE
`do.insert`	PARAM_DESCRIPTION, Default: FALSE

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

[read_xlsx][readxl::read_xlsx] [str_squish][stringr::str_squish], [str_extract_all][stringr::str_extract_all] [mutate][dplyr::mutate], [across][dplyr::across], [rename][dplyr::rename], [select][dplyr::select], [rowwise][dplyr::rowwise], [ungroup][dplyr::ungroup] [pivot_longer][tidyr::pivot_longer]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_doe_source *FUNCTION_TITLE*

Description

Load doe Source into dev_toxval_source_v4.

Usage

```
import_doe_source(toxval.db, infile)
```

Arguments

`toxval.db`	The version of toxval into which the source is loaded.
`infile`	The input file ./doe/doe_files/Revision_29.xlsx

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

`read.xlsx` `distinct`, `filter-joins`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_efsa_source` *import_efsa_source*

Description

Import of EFSA OpenFoodTox 2022 source into toxval_source

Usage

```
import_efsa_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `str_trim` `rename`, `mutate`, `recode`, `across`, `select`, `distinct` `separate`, `reexports`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_envirotox_source

FUNCTION_TITLE

Description

Load EnviroTox.V2 Source data into dev_toxval_source_v4.

Usage

```
import_envirotox_source(toxval.db, infile)
```

Arguments

`toxval.db`	The version of toxval into which the source info is loaded.
`infile`	The input file ./envirotox/envirotox_files/envirotox_taxonomy.xlsx

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

`read.xlsx`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_flex_source` *FUNCTION_TITLE*

Description

Load the FLEX data (old ACToR data) from files to toxval source. This will load all Excel file in the folder ACToR replacements/

Usage

```
import_flex_source(
  db,
  filepath = "ACToR replacements",
  verbose = F,
  chem.check.halt = F,
  do.clean = F
)
```

Arguments

`db`	The version of toxval_source into which the tables are loaded.
`filepath`	The path for all the input xlsx files ./ACToR replacements
`verbose`	Whether the loaded rows should be printed to the console.
`chem.check.halt`	If TRUE and there are problems with chemicals CASRN checks, halt the program
`do.clean`	If true, remove data for these sources before reloading

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

`read.xlsx`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_generic_source *FUNCTION_TITLE*

Description

A generic template for adding data to toxval_source for a new source

Import USGS HBSL data into toxval_source

Import of WHO IPCS data

Import of WHO JECFA ADI data

Usage

```
import_generic_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)

import_source_usgs_hbsl(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)

import_who_ipcs(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)

import_source_who_jecfa_adi(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before

`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

DETAILS

DETAILS

DETAILS

Value

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

None. Data is processed into the toxval_source database

OUTPUT_DESCRIPTION

See Also

`read_excel` `str_trim`

`read_excel` `str_trim`

`read_excel` `str_trim`

`read_excel` `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_hawc_pfas_source`

import_hawc_pfas_source

Description

Load HAWC PFAS data into toxval_source

Usage

```
import_hawc_pfas_source(
  db,
  hawc_num = NULL,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`hawc_num`	The HAWC number being processed (e.g. 150, 430)
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is added to toxval_source

See Also

`read_excel` `mutate`, `select`, `distinct`, `arrange`, `count`, `mutate-joins`, `filter`, `rename`, `case_when` `all_of` `pivot_wider`, `unite`, `pivot_longer`, `separate`, `drop_na`, `separate_rows` `str_trim`, `str_detect`, `str_extract` `map2` `digest`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_hawc_source` *import_hawc_source*

Description

Load HAWC Project data into toxval_source

Usage

```
import_hawc_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`getSheetNames`, `read.xlsx` `select`, `distinct`, `mutate`, `arrange`, `count`, `mutate-joins`, `bind_rows`, `context`, `reexports`, `across`, `na_if` `pivot_wider`, `unite`, `reexports`, `drop_na` `setops` `digest` `str_extract`, `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

import_health_canada_source
import_health_canada_source

Description

Transforms and loads Health Canada data into toxval_source

Usage

```
import_health_canada_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

db	The version of toxval_source into which the source is loaded.
chem.check.halt	If TRUE and there are bad chemical names or casrn,
do.reset	If TRUE, delete data from the database for this source before
do.insert	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

read_excel mutate, across, reexports, case_when, na_if str_extract, str_trim, modifiers, str_count

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_heast_source *import_heast_source*

Description

Load HEAST data into toxval_source

Usage

```
import_heast_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `filter`, `select`, `c("rowwise", "rowwise")`, `mutate`, `group_by`, `na_if`, `row_number`, `rename`, `bind_rows`, `case_when` `all_of`, `starts_with` `str_trim`, `str_extract` `drop_na`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_hpvis_source   *import_hpvis_source*

Description

Load HPVIS data into toxval_source

Usage

```
import_hpvis_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read.xlsx` `setNames` `mutate`, `setops`, `filter`, `across`, `reexports`, `na_if`, `case_when`, `row_number`, `select`, `bind_rows` `type.convert` `str_trim`, `str_replace`, `str_extract`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

import_niosh_source *import_niosh_source*

Description

Load NIOSH data into toxval_source

Usage

```
import_niosh_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

db	The version of toxval_source into which the source is loaded.
chem.check.halt	If TRUE and there are bad chemical names or casrn,
do.reset	If TRUE, delete data from the database for this source before
do.insert	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

read_excel mutate, case_when str_trim, str_extract separate, drop_na pivot_longer

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_oppt_source` *FUNCTION_TITLE*

Description

FUNCTION_DESCRIPTION

Usage

```
import_oppt_source(db, infile = "OPPT_data_20181219.xlsx", chem.check.halt = T)
```

Arguments

`db`	The version of toxval_source into which the source info is loaded.
`infile`	The input file ./oppt/oppt_files/OPPT_data_20181219.xlsx
`chem.check.halt`	If TRUE, stop if there are problems with the chemical mapping

Details

#' Load OPPT Source Info into toxval_source

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

`read.xlsx`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_opp_source` *import_opp_source*

Description

Load EPA OPP data to toxval_source

Usage

```
import_opp_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE,
  do.summary_data = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE
`do.summary_data`	If TRUE, add OPP Summary data to table before insertion

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `pivot_longer`, `separate` `filter`, `mutate`, `row_number`, `case_when`, `select`, `bind_rows` `str_trim`, `str_extract`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_ow_dwsha_source` *import_ow_dwsha_source*

Description

Load OW Drinking Water Standards data into toxval_source

Usage

```
import_ow_dwsha_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is added to toxval_source

See Also

`read_excel` `mutate`, `case_when`, `filter`, `row_number`, `bind_rows` `separate`, `pivot_longer` `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_pfas_150_sem_v2_source` *import_pfas_150_sem_v2_source*

Description

Load PFAS 150 SEM V2 Source data into toxval_source

Usage

```
import_pfas_150_sem_v2_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_rsl_source` *import_rsl_source*

Description

Import of RSL 2023 source into toxval_source

Usage

```
import_rsl_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	PARAM_DESCRIPTION, Default: FALSE
`do.insert`	PARAM_DESCRIPTION, Default: FALSE
`infile1`	The input file ./rsl/rsl_files/rsl_thq10_nov_2022.xlsx
`infile2`	The input file ./rsl/rsl_files/rsl_thq01_nov_2022.xlsx
`infile3`	The input file ./rsl/rsl_files/rsl_subchronic_nov_2022.xlsx

Details

DETAILS

Value

None; data is sent to toxval_source

See Also

`read_excel` `bind_rows`, `mutate_all`, `mutate`, `na_if`, `across`, `reexports`, `case_when` `unite`, `pivot_longer`, `separate`, `drop_na` `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_atsdr_mrls` *import_source_astdr_mrls*

Description

Send ASTDR MRLs data to toxval_source

Usage

```
import_source_atsdr_mrls(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE,
  do.toxicological_profile = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE
`do.toxicological_profile`	If TRUE, add toxicological profile data to table before insertion

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `separate` `mutate`, `case_when` `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_caloehha` *import_source_caloehha*

Description

Load caloehha Source file into toxval_source The raw data can be exported as an Excel sheet from the web site https://oehha.ca.gov/chemicals, selecting the link "Export database as .CSV file"

This method parses that file and prepares for loading into toxval source

Usage

```
import_source_caloehha(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE,
  do.summary_data = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are problems with chemicals CASRN checks, halt the program
`do.reset`	PARAM_DESCRIPTION, Default: FALSE
`do.insert`	PARAM_DESCRIPTION, Default: FALSE
`do.summary_data`	If TRUE, add Cal OEHHA Summary data to table before insertion
`infile`	The input file ="../caloehha/caloehha_files/OEHHA-chemicals_2018-10-30T08-50-47.xlsx",

Details

DETAILS

Value

None; data is pushed to ToxVal_Source

See Also

`read.xlsx`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_epa_aegl` *import_source_epa_aegl*

Description

Import EPA AEGL data into toxval_source

Usage

```
import_source_epa_aegl(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	PARAM_DESCRIPTION
`chem.check.halt`	If TRUE, stop if there are problems with the chemical mapping
`do.reset`	PARAM_DESCRIPTION, Default: FALSE
`do.insert`	PARAM_DESCRIPTION, Default: FALSE

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

`read_excel` `filter`, `mutate`, `select`, `distinct`, `mutate-joins` `separate_rows`, `pivot_longer`, `reexports` `modifiers`, `str_remove`, `str_split`, `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_epa_hhtv` *import_source_epa_hhtv*

Description

Push EPA HHTV data to toxval_source

Usage

```
import_source_epa_hhtv(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_epa_ow_npdwr` *FUNCTION_TITLE*

Description

FUNCTION_DESCRIPTION

Usage

```
import_source_epa_ow_npdwr(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

#' Import of EPA OW NPDWR source into toxval_source

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

`read_excel` `rename`, `mutate`, `across` `pivot_longer`, `separate` `str_trim`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_source_epa_ow_nrwqc_hhc

FUNCTION_TITLE

Description

Import of EPA OW NRWQC-HHC source into toxval_source

Usage

```
import_source_epa_ow_nrwqc_hhc(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

db	The version of toxval_source into which the source is loaded.
chem.check.halt	If TRUE and there are bad chemical names or casrn,
do.reset	If TRUE, delete data from the database for this source before
do.insert	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

read_excel rename, mutate, across, c("rowwise", "rowwise", "rowwise") pivot_longer, reexports, separate str_detect, str_trim

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_gestis_dnel` *import_source_gestis_dnel*

Description

Import GESTIS DNEL into toxval_source

Usage

```
import_source_gestis_dnel(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

[read_xlsx][readxl::read_xlsx] [str_squish][stringr::str_squish] [mutate][dplyr::mutate], [across][dplyr::across], [select][dplyr::select], [distinct][dplyr::distinct], [filter][dplyr::filter] [pivot_longer][tidyr::pivot_longer]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_source_hess *import_source_hess*

Description

Load HESS data into toxval_source

Usage

```
import_source_hess(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

db	The version of toxval_source into which the source is loaded.
chem.check.halt	If TRUE and there are bad chemical names or casrn,
do.reset	If TRUE, delete data from the database for this source before
do.insert	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

read_excel pivot_longer, unite, drop_na mutate, across, reexports, na_if, case_when, tidyeval-compat, select, rename, mutate-joins str_replace, str_trim, case, str_extract

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_iris` *import_source_iris*

Description

Import of IRIS 2023-05-09 source into toxval_source

Import PPRTV (CPHEA) source data into toxval_source

Usage

```
import_source_iris(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE,
  do.summary_data = FALSE
)

import_source_pprtv_cphea(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE,
  do.summary_data = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE
`do.summary_data`	If TRUE, add PPRTV CPHEA Summary data to table before insertion

Details

DETAILS

DETAILS

Value

None; data is pushed to toxval_source

None; data is pushed to toxval_source

See Also

`read_excel` `mutate`, `filter`, `select`, `across`, `rename`, `c("rowwise", "rowwise", "rowwise")`, `distinct` `pivot_longer`, `reexports`, `separate`, `replace_na` `str_trim`, `str_replace`, `str_extract`

`read_excel` `pivot_longer`, `reexports`, `separate`, `replace_na`, `drop_na` `mutate`, `across`, `case_when`, `select`, `reexports`, `distinct` `str_trim`, `str_extract` `all_of`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_iuclid` *import_source_iuclid*

Description

Import IUCLID data to ToxVal Source

Usage

```
import_source_iuclid(
  db,
  subf,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`subf`	The subfolder containing the IUCLID subsource
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `filter`, `group_by`, `mutate`, `row_number`, `context`, `case_when`, `pull`, `rename`, `select` `separate_rows`, `reexports`, `separate`, `unite`, `pivot_longer`, `pivot_wider`, `drop_na` `str_trim`, `str_extract`, `modifiers` `str_detect` `str_split` `str_unique` `mgsub`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_iuclid_orchestrate` *import_source_iuclid_orchestrate*

Description

Load the various IUCLID subsources into ToxVal

Usage

```
import_source_iuclid_orchestrate(
  dir = paste0(toxval.config()$datapath, "iuclid")
)
```

Arguments

`dir`	directory containing the various IUCLID subsource subdirectories
`db`	The version of toxval_source into which the source is loaded.
`do.insert`	If TRUE, insert data into the database, default TRUE
`chem.check.halt`	If TRUE, stop the execution if there are errors in the

Details

DETAILS

Value

None, subsources loaded

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_mass_mmcl` *import_source_mass_mmcl*

Description

Load Mass. Drinking Water Standards into toxval_source

Usage

```
import_source_mass_mmcl(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE, stop if there are problems with the chemical mapping
`do.reset`	If TRUE, delete data from the database for this source before inserting new data
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is loaded into toxval_source

See Also

read_excel mutate, across, reexports, rename, coalesce, filter, case_when reexports pivot_longer, drop_na, separate str_match, str_trim

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_ntp_pfas` *import_source_ntp_pfas*

Description

A function for adding source NTP PFAS data to toxval_source

Usage

```
import_source_ntp_pfas(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

`read_excel` `rename`, `mutate`, `across`, `bind_rows`, `distinct`, `c("rowwise", "rowwise")`, `select`, `filter`, `mutate-joins`, `case_when` `pivot_longer`, `separate`, `unite`, `drop_na` `str_trim`, `str_extract`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`import_source_penn_dep_mscs` *import_source_penn_dep*

Description

Load Pennsylvania DEP MSCs into toxval_source

Usage

```
import_source_penn_dep_mscs(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE and there are bad chemical names or casrn,
`do.reset`	If TRUE, delete data from the database for this source before
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is loaded to MySQL server

See Also

`read_excel` `mutate`, `case_when`, `distinct` `str_trim` `unite`, `pivot_longer`, `separate`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

import_source_penn_dep_toxvalues *import_source_penn_dep_toxvalues*

Description

Load Penn DEP ToxValues Source into toxval_source

Usage

```
import_source_penn_dep_toxvalues(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`chem.check.halt`	If TRUE, stop if there are problems with the chemical mapping
`do.reset`	If TRUE, delete data from the database for this source before inserting new data
`do.insert`	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is loaded into toxval_source

See Also

read_excel mutate, filter, case_when str_trim unite, pivot_longer, separate

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

import_source_who_jecfa_tox_studies

import_source_who_jecfa_tox_studies

Description

Import of WHO JECFA Tox Studies data

Usage

```
import_source_who_jecfa_tox_studies(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

db	The version of toxval_source into which the source is loaded.
chem.check.halt	If TRUE and there are bad chemical names or casrn,
do.reset	If TRUE, delete data from the database for this source before
do.insert	If TRUE, insert data into the database, default FALS

Details

DETAILS

Value

None; data is pushed to toxval_source

See Also

read_excel mutate, case_when, filter separate_rows str_trim str_extract

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

import_test_source *import_test_source*

Description

Load TEST Source data into toxval_source

Usage

```
import_test_source(
  db,
  chem.check.halt = FALSE,
  do.reset = FALSE,
  do.insert = FALSE
)
```

Arguments

db	The version of toxval_source into which the source is loaded.
chem.check.halt	If TRUE, stop if there are problems with the chemical mapping
do.reset	If TRUE, delete data from the database for this source before inserting new data
do.insert	If TRUE, insert data into the database, default FALSE

Details

DETAILS

Value

None; data is loaded into toxval_source

See Also

read_excel read.table select, mutate-joins, join_by, mutate, case_when str_trim pivot_longer, drop_na

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`init.audit.table` *init.audit.table*

Description

Create audit table and add BEFORE UPDATE audit triggers to source_* tables

Create audit table and add BEFORE UPDATE audit triggers to source_* tables

Usage

```
init.audit.table(db, do.halt = FALSE, verbose = FALSE)
```

Arguments

`db`	the name of the database
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information
`s_tbl`	Source table name to apply changes to
`field_list`	List of current field names in source table

Details

DETAILS

Value

None

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`initialize_source_iuclid_directory` *FUNCTION_TITLE*

Description

Initialize Source IUCLID Directory into subdirectory based on input files

Usage

```
initialize_source_iuclid_directory()
```

Details

DETAILS

Value

None, file directory structure generated

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`parse_sql_file` *parse_sql_file*

Description

Function to parse SQL file into SQL query strings

Usage

```
parse_sql_file(filepath = NULL)
```

Arguments

`filepath`	Input SQL filepath

Details

DETAILS

Value

SQL query strings

See Also

read_lines str_trim

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`prep.DAT.conversion` *prep.DAT.conversion*

Description

Select and rename DAT audit columns for toxval_source, calculate new source_hash

Usage

```
prep.DAT.conversion(in_dat = NULL, hash_id_list = NULL, hashing_type = "base")
```

Arguments

in_dat	Input DAT data
hash_id_list	List of hash values to keep

Details

DETAILS

Value

Updated DAT tibble

See Also

rename, select, mutate unite, reexports map

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`printCurrentFunction` *printCurrentFunction*

Description

Print the name of the current function

Usage

```
printCurrentFunction(comment.string = NA)
```

Arguments

`comment.string`	An optional string to be printed

Details

DETAILS

Value

None

See Also

`flush.console`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

qc_prescreening_summary

qc_prescreening_summary

Description

Runs a database query and returns a result set

Usage

```
qc_prescreening_summary(
  src_tbl = NULL,
  source_name = NULL,
  outputDir = NULL,
  db = NULL
)
```

Arguments

`src_tbl`	a toxval source table name.
`source_name`	a toxval source name (used for direct load types).
`outputDir`	optional directory path to save output file in.
`db`	the name of the database.

Details

DETAILS

Value

Result set of QC prescreenig information

See Also

`pivot_longer` `reexports`, `group_by`, `summarise` `write_xlsx`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

runInsert *runInsert*

Description

Insert a record into a database. if auto.increment=TRUE, return the auto incremented primary key of the record. otherwise, return -1

Usage

```
runInsert(query, db, do.halt = F, verbose = F, auto.increment.id = F)
```

Arguments

`query`	a properly formatted SQL query as a string
`db`	the name of the database
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information
`auto.increment.id`	PARAM_DESCRIPTION, Default: F
`auto.increment`	if TRUE, add the auto increment primary key even if not part of the query

Details

DETAILS

Value

Returns the database table auto incremented primary key ID

See Also

`character(0)`, `MySQLDriver-class`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

runInsertTable *runInsertTable*

Description

Inserts multiple rows into a database table

Usage

```
runInsertTable(mat, table, db, do.halt = TRUE, verbose = FALSE, get.id = TRUE)
```

Arguments

`mat`	data frame containing the data, with the column names corresponding
`table`	name of the database table to which data will be inserted
`db`	the name of the database
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information
`get.id`	Whether to return ID or not, Default: T

Details

DETAILS

Value

ID or None

See Also

`character(0)`, `MySQLDriver-class`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`runQuery` *runQuery*

Description

Runs a database query and returns a result set

Usage

```
runQuery(query = NULL, db, do.halt = TRUE, verbose = FALSE)
```

Arguments

`query`	a properly formatted SQL query as a string
`db`	the name of the database
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information

Details

DETAILS

Value

Query results

See Also

`character(0)`, `MySQLDriver-class` `flush.console`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

runStatement *runStatement*

Description

Run a SQL statement, such as an ALTER or UPDATE

Usage

```
runStatement(query, db, do.halt = FALSE, verbose = FALSE)
```

Arguments

`query`	a properly formatted SQL query as a string
`db`	the name of the database
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information

Details

DETAILS

Value

None. SQL statement is run.

See Also

`character(0)`, `MySQLDriver-class`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`runUpdate` *runUpdate*

Description

Runs a database query and returns a result set

Usage

```
runUpdate(
  table,
  updateQuery = NULL,
  updated_df = NULL,
  db,
  do.halt = TRUE,
  verbose = FALSE,
  trigger_check = TRUE
)
```

Arguments

`table`	table to update
`updateQuery`	a properly formatted SQL query as a string in the form of an UPDATE INNER JOIN
`updated_df`	a dataframe of updated data to temporarily write to database for INNER JOIN
`db`	the name of the database
`do.halt`	if TRUE, halt on errors or warnings
`verbose`	if TRUE, print diagnostic information
`trigger_check`	if FALSE, audit triggers are ignored/bypassed

Details

DETAILS

Value

None

See Also

`character(0)`, `MySQLDriver-class` `dbSendStatement`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

set_clowder_doc_type *set_clowder_doc_type*

Description

Update documents table entries "document_type" field based on Clowder organization

Usage

```
set_clowder_doc_type(
  source_table = NULL,
  source_version_date = NULL,
  clowder_url = NULL,
  clowder_api_key = NULL,
  source.db = NULL,
  ds_id = NULL,
  clowder_id_list = NULL
)
```

Arguments

`source_table`	The source table name (e.g. source_test). Default is NULL for "all"
`source_version_date`	The version date for the source table. Default is NULL for "all"
`clowder_url`	URL to Clowder
`clowder_api_key`	API key to access Clowder resources
`source.db`	Name of the toxval_source database to apply updates to
`ds_id`	Clowder Dataset ID for ToxVal Clowder Documents.
`clowder_id_list`	Optional DataFrame with field "clowder_id" values for document records to update.

Details

DETAILS

Value

None. SQL statements are performed.

See Also

`read_excel` `rename`, `filter`, `select`, `mutate-joins`, `mutate` `separate_rows`, `unite`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`set_clowder_id_lineage` *set_clowder_id_lineage*

Description

Create document records and associations in toxval_source based on input source table and document map

Usage

```
set_clowder_id_lineage(
  source_table,
  map_clowder_id_field,
  map_file,
  clowder_url,
  clowder_api_key,
  sync_clowder_metadata = FALSE,
  source.db,
  toxval.db
)
```

Arguments

`source_table`	The source table name (e.g. source_test)
`map_clowder_id_field`	Column name for the Clowder ID field of the map
`map_file`	A dataframe of Clowder document mapping info. If NULL, will try to load a hardcoded map for the source
`clowder_url`	URL to Clowder

`clowder_api_key`	API key to access Clowder resources
`sync_clowder_metadata`	Boolean whether to sync Clowder metadata for new document records. Default is False.
`source.db`	The source database name
`toxval.db`	The database version to use

Details

DETAILS

Value

Returns an updated map with newly associated toxval_source table ID values

See Also

`read_excel` `rename`, `filter`, `select`, `mutate-joins`, `mutate` `separate_rows`, `unite`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`set_field_SQL_type` *set_field_SQL_type*

Description

Helper function to generate SQL field types based on dataframe field types

Usage

```
set_field_SQL_type(src_f = NULL, default_fields = NULL)
```

Arguments

`src_f`	Dataframe to generate field types from
`default_fields`	Default fields already handled by input generic SQL

Details

DETAILS

Value

SQL string for the input dataframe's fields

See Also

`bind`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`source.table.to.DAT` *source.table.to.DAT*

Description

Convert toxval source table to DAT format for loading to DAT application

Usage

```
source.table.to.DAT(source.db, source_table, limit = 1e+06, sample_p = NA)
```

Arguments

`source.db`	The version of toxval source to use.
`source_table`	The name of toxval source table to use. If a DataFrame, input data will be #' processing and returned without saving to file.
`limit`	Excel file grouping limit (default is max XLSX row limit)
`sample_p`	Percentage of records to sample down to
`source`	The name of toxval source to use.

Details

DETAILS

Value

Processed source table to DAT format cached and returned.

See Also

`rename`, `filter`, `slice`, `select` `pivot_longer`, `reexports` `write_xlsx`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`source_chemical.process` *source_chemical.process*

Description

Deal with the process of making the source_chemical information

Usage

```
source_chemical.process(
  db,
  res,
  source,
  table,
  chem.check.halt = FALSE,
  casrn.col = "casrn",
  name.col = "name",
  verbose = FALSE
)
```

Arguments

`db`	The version of toxval into which the source info is loaded.
`res`	The input dataframe to which chemical information will be added
`source`	The source to process
`table`	Name of the database table
`chem.check.halt`	If TRUE, stop if there are problems with the chemical mapping
`casrn.col`	The name of the column containing the CASRN
`name.col`	The name ofhte column containing hte chemical name
`verbose`	If TRUE, write out diagnostic messages #'

Details

DETAILS

Value

Returns the original dataframe with a chemical_id appended

See Also

`unite` `head` `digest`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

source_hash_vectorized *source_hash_vectorized*

Description

Generate the hash key for a source table based on hashing columns

Add the hash key to the source tables and add the new rows

Usage

```
source_hash_vectorized(res, hashing_cols)

toxval_source.hash.and.load(
  db = "dev_toxval_source_v5",
  source,
  table,
  do.reset = FALSE,
  do.insert = FALSE,
  res,
  hashing_cols = NULL
)
```

Arguments

`res`	The data frame to be processed
`hashing_cols`	Optional list of columns to use for generating source_hash
`db`	The version of toxval_source into which the source is loaded.
`source`	Name of the source
`table`	Name of the database table

`do.reset`	If TRUE, delete data from the database for this source before #' inserting new data. Default FALSE
`do.insert`	If TRUE, insert data into the database, default False

Details

DETAILS

DETAILS

Value

Input dataframe with new source_hash field

None

See Also

`digest` `distinct`

`digest` `distinct`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`source_prep_and_load` *source_prep_and_load*

Description

Prep the source data aand load

Usage

```
source_prep_and_load(
  db,
  source,
  table,
  res,
  do.reset = FALSE,
  do.insert = FALSE,
  chem.check.halt = FALSE,
  verbose = FALSE,
  hashing_cols = NULL
)
```

Arguments

`db`	The version of toxval_source into which the source is loaded.
`source`	Name of the source
`table`	Name of the database table
`res`	The data frame to be processed
`do.reset`	If TRUE, delete data from the database for this source before #' inserting new data. Default FALSE
`do.insert`	If TRUE, insert data into the database, default FALSE
`chem.check.halt`	If TRUE, stop the execution if there are errors in the #' chemical mapping
`verbose`	If TRUE, write out diagnostic messages #'
`hashing_cols`	Optional list of columns to use for generating source_hash

Details

DETAILS

Value

None

See Also

`reexports`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

source_set_defaults *source_set_defaults*

Description

Set default value for NAs - just set NA to "-" for columns of type character

Usage

```
source_set_defaults(res, source)
```

Arguments

res	The input dataframe
source	The data source name

Details

DETAILS

Value

Returns the input dataframe with defaults set

See Also

[pull][dplyr::pull]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`toxval.config` *toxval.config*

Description

Define a set of global variables. These include the source path (datapath) and the source databases (e.g. dev_toxval_version and dev_toxval_source_version).

Usage

```
toxval.config()
```

Details

DETAILS

Value

Returns a set of parameters to be used throughout the package

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`toxval.source.import.dedup` *toxval.source.import.dedup*

Description

Perform deduping on data before it is sent to toxval_source

Usage

```
toxval.source.import.dedup(
  res,
  dedup_fields = NULL,
  hashing_cols = NULL,
  delim = " |::| "
)
```

Arguments

`res`	dataframe containing the source data to dedup
`dedup_fields`	vector containing field names to dedup, Default: NULL (all fields but hashing cols)
`hashing_cols`	vector containing field names of hashing columns, Default: toxval.config()$hashing_cols
`delim`	string used to separate collapsed values, Default: ' |::| '

Details

DETAILS

Value

dataframe containing deduped source data

See Also

`select`, `group_by`, `summarise`, `context`, `filter`, `mutate`, `across`, `reexports`, `na_if`, `distinct`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`toxval.source_push_mapped_chemicals`
toxval.source_push_mapped_chemicals

Description

Orchestrates the push of mapped chemical information to the selected toxval_source database. Uses the map_curated_chemicals() helper function to generate mapped input.

Usage

```
toxval.source_push_mapped_chemicals(
  db,
  source.index,
  curated.path,
  ignore.curation.dups = FALSE,
  match.chemical.id = TRUE,
  reset.mapping = FALSE,
  bulk.push = TRUE
)
```

Arguments

`db`	The version of toxval source database to use.
`source.index`	The source chemical index. Can be full or just numeric (ex. ToxVal00001 vs. 00001)
`curated.path`	Input path to the folder directory with expected subdirectories of #' 'BIN Files', 'DSSTox Files', and 'jira_chemical_files'
`ignore.curation.dups`	Boolean whether to match with any curated records flagged as "unresolved duplicates" (Default FALSE)
`match.chemical.id`	Boolean whether to match by provided chemical_id external identifier (Default TRUE)
`reset.mapping`	Boolean whether to reset chemical mappings in source_chemical table of database
`bulk.push`	Boolean whether to bulk push updates, or one at a time. Default is TRUE

Details

DETAILS

Value

None. Update SQL statements are executed.

See Also

`read_excel` `rename`, `distinct`, `mutate`, `select`, `mutate-joins`, `filter`, `bind`

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`update_chemical_preferred_info_by_dtxsid`
update_chemical_preferred_info_by_dtxsid

Description

Set the name and casrn in the source_chemical table based on CCTE API

Usage

```
update_chemical_preferred_info_by_dtxsid(source.db)
```

Arguments

`source.db`	The database version to use

Details

DETAILS

Value

None

See Also

[GET][httr::GET], [content][httr::content], [POST][httr::POST], [accept_json][httr::accept_json], [content_type_json][httr::content_type_json], [add_headers][httr::add_headers] [bind_rows][dplyr::bind_rows], [select][dplyr::select], [mutate][dplyr::mutate], [n][dplyr::n]

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

`%>%` *Pipe operator*

Description

See `magrittr::%>%` for details.

Usage

```
lhs %>% rhs
```

Arguments

`lhs`	A value or the magrittr placeholder.
`rhs`	A function call using the magrittr semantics.

Value

The result of calling ‘rhs(lhs)‘.

Index