Package: contoso (via r-universe)

October 30, 2026

Type Package

Title Dataset of the 'Contoso' Company

Version 2.4.0

Description A collection of synthetic datasets simulating sales transactions from a fictional company. The dataset includes various related tables that contain essential business and operational data, useful for analyzing sales performance and other business insights. Key tables included in the package are: - ``sales'': Contains data on individual sales transactions, including order details, pricing, quantities, and customer information. - ``customer'': Stores customer-specific details such as demographics, geographic location, occupation, and birthday. - ``store'': Provides information about stores, including location, size, status, and operational dates. - ``orders'': Contains details about customer orders, including order and delivery dates, store, and customer data. - ``product'': Contains data on products, including attributes such as product name, category, price, cost, and weight. - ``calendar'': A time-based table that includes date-related attributes like year, month, quarter, day, and working day indicators. The package also includes 1,127 fictional internal business reports (weekly sales flashes, monthly reviews, regional reviews, pricing committee minutes, annual operating plans and industry notes) written about the trading from 2016 to 2025 in the larger cloud dataset, with a glossary of the terms they use, for building and testing document search and financial planning and analysis workflows with large language models. This dataset is ideal for practicing data analysis, performing time-series analysis, creating reports, or simulating business intelligence scenarios.

License MIT + file LICENSE

Imports DBI, dplyr, cli, duckdb (>= 1.4.0), purrr, tibble, yaml12

Suggests testthat (>= 3.0.0), withr, dbplyr, yaml, knitr, quarto

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

URL https://codeberg.org/usrbinr/contoso

Config/testthat/edition 3

BugReports https://codeberg.org/usrbinr/contoso/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Alejandro Hagan [aut, cre]

Maintainer Alejandro Hagan <alejandro.hagan@outlook.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-30 07:17:22 UTC

RemoteUrl https://github.com/cran/contoso

RemoteRef HEAD

RemoteSha bdbb3ecfd434881da0c4e0d369f7e30fd06dae45

Contents

calendar . . . 2
create_contoso_duckdb . . . 4
customer . . . 5
export_reports . . . 6
fx . . . 7
launch_ui . . . 8
orderrows . . . 9
orders . . . 10
product . . . 10
read_reports . . . 11
sales . . . 12
store . . . 13

Index **15**

`calendar` *Calendar Dimension Data from the Contoso Dataset*

Description

This dataset contains calendar-related information used for time-based analysis in the Contoso dataset. It includes various representations of date-related attributes, such as year, quarter, month, and day, along with indicators for working days. It is useful for time-series analysis and aggregating data by different time periods.

Usage

```
calendar
```

Format

A data frame with 1,461 rows and 17 columns:

date `Date` The actual date for the record.

date_key `double` The date as `YYYYMMDD`.

year `double` The year part of the date.

year_quarter `character` Quarter and year, formatted `"Q1-2021"`.

year_quarter_number `double` A continuous quarter counter, `year * 4 + quarter`, running from 8085. **Not** 1-4 — use `quarter` for the within-year number.

quarter `character` The quarter within the year (e.g., "Q1", "Q2").

year_month `character` Month and year, formatted `"January 2021"`.

year_month_short `character` Month and year, formatted `"Jan 2021"`.

year_month_number `double` A continuous month counter, `year * 12 + month`, running from 24253. **Not** `YYYYMM`.

month `character` The month name (e.g., "March").

month_short `character` The abbreviated month name (e.g., "Mar").

month_number `double` The month within the year, 1-12.

day_of_week `character` The full name of the day of the week (e.g., "Monday").

day_of_week_short `character` The abbreviated day of the week (e.g., "Mon").

day_of_week_number `double` The day within the week, starting **Sunday = 1**, not Monday.

working_day `double` Indicator of whether the date is a working day (1 for working day, 0 for non-working day). This is the flag.

working_day_number `double` A **cumulative** count of working days elapsed, running 0-1003. Not a flag: it increments on working days and repeats on non-working days.

Details

Spans whole calendar years, so it extends past the trading period at both ends. Join from `calendar` when you need a dense series with explicit zeros.

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

`create_contoso_duckdb` *Creates DuckDB database with Contoso datasets*

Description

Creates a DuckDB connection with Contoso datasets loaded from cloud storage. The datasets are stored as Parquet files on Cloudflare R2 and streamed directly into DuckDB.

Usage

```
create_contoso_duckdb(size = "small")
```

Arguments

`size`	Dataset size: "small", "medium", "large", or "mega"

Details

The `create_contoso_duckdb()` function creates views for the following Contoso datasets:

- `sales`: Contains sales transaction data.
- `product`: Contains details about products, including attributes like product name, manufacturer, and category.
- `customer`: Contains customer demographic and geographic information.
- `store`: Contains information about store locations and attributes.
- `fx`: Contains foreign exchange rate data for currency conversion.
- `calendar`: Contains various date-related information, including day, week, month, and year.
- `orders`: Contains order header information.
- `orderrows`: Contains order line items.

Available sizes (sales rows):

- `small`: 7,794
- `medium`: 2,349,091
- `large`: 23,719,935
- `mega`: 237,245,485

Value

A list containing:

- `sales`, `product`, `customer`, `store`, `fx`, `calendar`, `orders`, `orderrows`: lazy `tbl` objects
- `con`: the DuckDB connection (use `DBI::dbDisconnect(db$con, shutdown = TRUE)` when done)

Schemas

Column names, order and types are identical across all four sizes: date columns are `DATE`, keys and counts are `BIGINT`, and measures are `DOUBLE`. Only the row counts differ.

`size = "small"` serves the same data as the bundled sales, customer and friends. The larger sizes come from separate generator runs, so they cover a different date range and a much larger customer catalogue – including customers who never place an order, which the bundled data excludes.

Examples

```
## Not run:
  db <- create_contoso_duckdb(size = "small")
  db$sales |> head()
  DBI::dbDisconnect(db$con, shutdown = TRUE)

## End(Not run)
```

`customer` *Customer Data from the Contoso Dataset*

Description

This dataset contains information about customers from the Contoso dataset, including demographic details, geographical information, contact information, and other personal attributes. It provides insights into customer profiles, including location, age, occupation, and more.

Usage

```
customer
```

Format

A data frame with 3,165 rows and 24 columns:

customer_key `double` Unique identifier for each customer. A primary key: unique and never missing.

geo_area_key `double` Unique identifier for the geographical area the customer resides in.

start_date `Date` Date when the customer relationship began.

end_date `Date` Date when the customer relationship ended, if applicable.

continent `character` The continent where the customer resides.

gender `character` The gender of the customer (e.g., 'Male', 'Female').

title `character` The title of the customer (e.g., 'Mr.', 'Ms.').

given_name `character` The given (first) name of the customer.

middle_initial `character` The middle initial of the customer, if applicable.

surname `character` The surname (last name) of the customer.

street_address `character` The street address of the customer.

city `character` The city where the customer resides.

state `character` The state or province where the customer resides.

state_full `character` The full name of the state or province.

zip_code `character` The postal (ZIP) code of the customer's address.

country `character` The country where the customer resides, using the country code.

country_full `character` The full name of the country where the customer resides.

birthday `Date` The date of birth of the customer.

age `double` The age of the customer.

occupation `character` The customer's occupation or profession.

company `character` The company the customer is associated with, if applicable.

vehicle `character` The type or make of vehicle the customer owns or drives.

latitude `double` The latitude of the customer's address.

longitude `double` The longitude of the customer's address.

Details

Each row is one customer. Only customers that placed at least one order are included, so every `customer_key` here appears in sales.

Changed in 2.2.0

Up to version 2.1.0 this table was built by joining the customer catalogue onto `sales`, which repeated each customer once per sales line they appeared on: 7,794 rows for 3,165 customers. `customer_key` was therefore not unique and any join to this table silently multiplied rows. The duplicates have been collapsed. No attribute varied within a customer, so nothing was lost.

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

---

`export_reports` *Write the Contoso business reports to files*

Description

Writes each report from `read_reports()` as a YAML file named `<id>.yaml`, holding the metadata fields and the Markdown text in `body`. `yaml12::read_yaml()` reads a file back in one call.

Usage

```
export_reports(dir, overwrite = FALSE)
```

Arguments

`dir`	Directory to write to. Created if it does not exist.
`overwrite`	Replace files that already exist?

Value

The paths of the written files, invisibly.

Examples

```
dir <- file.path(tempdir(), "contoso-reports")
paths <- export_reports(dir)
yaml12::read_yaml(paths[1])$title
unlink(dir, recursive = TRUE)
```

`fx` *Foreign Exchange Data from the Contoso Dataset*

Description

This dataset contains information about foreign exchange (FX) rates between different currencies. It includes details about the exchange rate for a given date, as well as the currencies involved. This dataset is useful for analyzing currency conversions and understanding the exchange rates between different currencies over time.

Usage

```
fx
```

Format

A data frame with 36,525 rows and 4 columns:

date `Date` The date of the exchange rate.

from_currency `character` The code of the source currency (e.g., "USD", "EUR").

to_currency `character` The code of the target currency (e.g., "GBP", "JPY").

exchange `double` The exchange rate between the source and target currencies on the given date.

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

launch_ui *Launch the DuckDB UI in your browser*

Description

The `launch_ui()` function installs and launches the DuckDB UI extension for an active DuckDB database connection. This allows users to interact with the database via a web-based graphical interface.

Your connection from `create_contoso_duckdb()` is returned in the list.

Usage

```
launch_ui(.con)
```

Arguments

`.con`	A valid `DBIConnection` object connected to a DuckDB database. The function will check that the connection is valid before proceeding.

Details

The function performs the following steps:

- Checks that the provided DuckDB connection is valid. If the connection is invalid, it aborts with a descriptive error message.
- Installs the `ui` extension into the connected DuckDB instance.
- Calls the `start_ui()` procedure to launch the DuckDB UI in your browser.

This provides a convenient way to explore and manage DuckDB databases interactively without needing to leave the R environment.

Value

The function is called for its side effects and does not return a value. It launches the DuckDB UI and opens it in your default web browser.

See Also

- `create_contoso_duckdb()` for creating example Contoso datasets in DuckDB.
- `DBI::dbConnect()` and `DBI::dbDisconnect()` for managing DuckDB connections.
- `duckdb::duckdb()` for creating a DuckDB driver instance.

Examples

```
## Not run:
# Connect to DuckDB
db <- create_contoso_duckdb()

# Launch the DuckDB UI
launch_ui(db$con)

# Clean up
DBI::dbDisconnect(db$con, shutdown = TRUE)

## End(Not run)
```

`orderrows` *Order Rows Data from the Contoso Dataset*

Description

This dataset contains detailed information about the individual items (rows) within each order in the Contoso dataset. It includes details such as the product, quantity, pricing, and cost of each item in an order. This dataset is useful for analyzing the breakdown of order components and individual product sales.

Usage

```
orderrows
```

Format

A data frame with 7,794 rows and 7 columns:

order_key `double` Unique identifier for the order to which the item belongs.

line_number `double` Line number within the order, identifying each product line.

product_key `double` Unique identifier for the product in the order row.

quantity `double` The quantity of the product ordered.

unit_price `double` The price per unit of the product.

net_price `double` The price *per unit* after discount, in USD. Despite the name this is a unit figure, not a line total.

unit_cost `double` The cost per unit of the product.

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

orders *Order Data from the Contoso Dataset*

Description

This dataset contains information about customer orders, including order dates, delivery dates, and store details.

Usage

```
orders
```

Format

A data frame with 3,242 rows and 6 columns:

order_key `double` Unique identifier for the order.

customer_key `double` Unique identifier for the customer who placed the order.

store_key `double` Unique identifier for the store where the order was placed.

order_date `Date` The date when the order was placed.

delivery_date `Date` The date when the order is expected to be delivered.

currency_code `character` The currency code used for the order (e.g., USD, EUR).

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

product *Product Data from the Contoso Dataset*

Description

This dataset contains information about products in the Contoso dataset. It includes product details such as identifiers, descriptions, pricing, weight, and categorization. This dataset is useful for analyzing product characteristics, pricing, and product-related sales insights.

Usage

```
product
```

Format

A data frame with 2,517 rows and 14 columns:

product_key `double` Unique identifier for each product.

product_code `character` A code that uniquely identifies the product.

product_name `character` The name or description of the product.

manufacturer `character` The name of the manufacturer of the product.

brand `character` The brand of the product.

color `character` The color of the product.

weight_unit `character` The unit `weight` is expressed in: `"grams"`, `"ounces"` or `"pounds"` — never `"kg"`. Metric and imperial units are mixed, so `weight` is **not comparable across rows** without converting. Missing where the product has no weight.

weight `double` The weight of the product, in `weight_unit`. Not on a common scale — see that column.

cost `double` The cost price of the product.

price `double` The selling price of the product.

category_key `double` Unique identifier for the category to which the product belongs.

category_name `character` The name of the category to which the product belongs.

sub_category_key `double` Unique identifier for the subcategory to which the product belongs.

sub_category_name `character` The name of the subcategory to which the product belongs.

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

---

`read_reports` *Read the Contoso business reports*

Description

Returns 1,127 fictional internal business documents written about Contoso's trading from May 2016 to the end of 2025, one row per report: the metadata a document system would record, plus the Markdown text in `body`. The figures in the reports come from the "mega" dataset (`create_contoso_duckdb("mega")`), not the bundled tables, which end in 2024. Use `export_reports()` to get them as files. The terms the reports use are defined in `system.file("extdata", "reports", "glossary.yaml", package = "contoso")`.

Usage

```
read_reports()
```

Value

A tibble with columns `id`, `title`, `unit`, `author`, `report_type`, `period_start`, `period_end`, `published` (the last three are `Date`) and `body`.

Examples

```
reports <- read_reports()
reports[, c("id", "published")]
```

---

sales *Sales Data from the Contoso Dataset*

Description

This dataset contains information about sales orders, including order details, pricing, and customer data from the Contoso dataset. It provides insights into the transactions that have occurred, including order dates, delivery dates, customer and store information, as well as product details.

Usage

```
sales
```

Format

A data frame with 7,794 rows and 20 columns:

order_key `double` Unique identifier for each order.

line_number `double` Line number within the order (for multi-line orders).

order_date `Date` Date when the order was placed.

delivery_date `Date` Date when the order was delivered.

customer_key `double` Unique identifier for the customer who placed the order.

store_key `double` Unique identifier for the store where the order was placed.

product_key `double` Unique identifier for the product in the order.

quantity `double` The quantity of the product ordered.

unit_price `double` The price per unit of the product.

net_price `double` The price *per unit* after discount, in USD. Despite the name this is a unit figure, not a line total.

unit_cost `double` The cost per unit of the product.

currency_code `character` The currency the order was placed in (e.g., USD, EUR). Does **not** describe the money columns, which are all USD.

exchange_rate `double` The USD-to-`currency_code` rate on `order_date`, from fx. 1.0 for USD orders. It has not been applied to the money columns — multiply if you want local currency.

gross_revenue `double` A product's unit_price multiplied by quantity.

net_revenue `double` A product's net_price multiplied by quantity.

unit_discount `double` A product's unit_price minus net_price.

discounts `double` A product's unit_discount multiplied by quantity.

cogs `double` Cost of goods sold. A product's unit_cost multiplied by quantity.

gross_margin `double` A product's net_revenue minus cogs.

unit_margin `double` A product's gross_margin divided by quantity.

Details

Each row is one line item on one order — the same grain as orderrows. This is the denormalised fact table: it repeats every column of orders and orderrows, adds the day's `exchange_rate`, and pre-computes seven revenue, cost and margin measures.

Currency

Every money column is in **USD**, regardless of the order's `currency_code`, so amounts sum directly across orders. Multiply by `exchange_rate` to express an amount in the currency the order was placed in.

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

---

`store` *Store Data from the Contoso Dataset*

Description

This dataset contains information about stores within the Contoso dataset. It includes details about the store's geographic location, operational status, and physical characteristics such as size and opening/closing dates. It provides insights into the store network of the company.

Usage

```
store
```

Format

A data frame with 74 rows (73 physical stores plus the online channel) and 11 columns:

store_key `double` Unique identifier for each store. 999999 is the online channel sentinel.

store_code `double` A code identifying the *site*. **Not unique** — a site with successive premises shares one code across several `store_key` values. -1 on the online row.

geo_area_key `double` The geographical area, on the same numbering as `customer$geo_area_key`. No geo-area table ships with the package, so this resolves to nothing on its own. -1 on the online row.

country_code `character` The country code where the store is located (e.g., "US", "DE"), plus the placeholder `"--"` on the online row.

country_name `character` The full name of the country, plus the pseudo-country `"Online"`.

state `character` The state or region, always spelled out — unlike `customer$state`, which is often abbreviated. Literally `"Online"` on the online row.

open_date `Date` The date when the store was opened.

close_date `Date` The date when the store was closed. Missing for stores still trading.

description `character` Always `"Contoso Store <state>"` for physical stores, and `"Online store"` for the online row. Carries no store-format information.

square_meters `double` The physical size of the store in square meters. Missing on the online row.

status `character` Exceptional status only: `"Closed"` or `"Restructured"`. **Missing, never "Open"**, for stores trading normally — test with `is.na()` rather than comparing to a string.

The online channel

`store_key` 999999 is the online channel, not a location. It carries sentinel values throughout: `store_code` and `geo_area_key` are -1, `country_code` is `"--"`, `country_name` and `state` are literally `"Online"`, and `square_meters` is missing. Filter it out before any geographic or floor-space aggregate. It accounts for a little over half of all sales lines.

Source

https://github.com/sql-bi/Contoso-Data-Generator-V2-Data/releases/tag/ready-to-use-data

Index

∗ **datasets**
    calendar, 2
    customer, 5
    fx, 7
    orderrows, 9
    orders, 10
    product, 10
    sales, 12
    store, 13

calendar, 2
create_contoso_duckdb, 4
create_contoso_duckdb(), *8*
customer, 5, *5*

DBI::dbConnect(), *8*
DBI::dbDisconnect(), *8*
duckdb::duckdb(), *8*

export_reports, 6
export_reports(), *11*

fx, 7, *12*

launch_ui, 8

orderrows, 9, *13*
orders, 10, *13*

product, 10

read_reports, 11
read_reports(), *6*

sales, *5, 6*, 12
store, 13