sql_book

Ian Gow

3/25/23

Table of contents

Preface

Some stuff.

1 Preparing Data for Analysis

Chapter 2 of Tanimura (2021) provides a good foundation discussion of issues related to preparing data for analysis. While the discussion is couched in terms of SQL, in reality the issues are not specific to SQL or databases. For this reason, I recommend that you read the chapter.

While Chapter 2 of Tanimura (2021) contains many code snippets, few of these seem to be intended for users to run (in part because they assume a database set-up that users would not have). For this reason, I do not attempt to provide `dplyr` equivalents to the code there except for a couple of exceptions that I discuss below.

1.1 Missing data

```
library(tidyverse)
library(DBI)
```

```
pg <- dbConnect(RPostgres::Postgres())

dates_sql <-
   "SELECT *
    FROM generate_series('2000-01-01'::timestamp,'2030-12-31', '1 day')"

dates <-
  tbl(pg, sql(dates_sql)) %>%
  rename(date = generate_series)

dates_processed <-
  dates %>%
  mutate(date_key = as.integer(to_char(date, 'yyyymmdd')),
         day_of_month = as.integer(date_part('day',date)),
         day_of_year = as.integer(date_part('doy', date)),
         day_of_week = as.integer(date_part('dow', date)),
         day_name  = trim(to_char(date, 'Day')),
```

```r
         day_short_name = trim(to_char(date, 'Dy')),
         week_number = as.integer(date_part('week', date)),
         week_of_month = as.integer(to_char(date,'W')),
         week = as.Date(date_trunc('week', date)),
         month_number = as.integer(date_part('month',date)),
         month_name = trim(to_char(date, 'Month')),
         month_short_name = trim(to_char(date, 'Mon')),
         first_day_of_month = as.Date(date_trunc('month', date)),
         last_day_of_month = as.Date(date_trunc('month', date) +
                               sql("interval '1 month' -
                                    interval '1 day'")),
         quarter_number = as.integer(date_part('quarter', date)),
         quarter_name = trim('Q' %||% as.integer(date_part('quarter', date))),
         first_day_of_quarter = as.Date(date_trunc('quarter', date)),
         last_day_of_quarter = as.Date(date_trunc('quarter', date) +
                                 sql("interval '3 months' -
                                      interval '1 day'")),
         year = as.integer(date_part('year', date)),
         decade = as.integer(date_part('decade', date)) * 10,
         century = as.integer(date_part('century', date)))

dates_processed %>%
  collect(n = 10)
```

```
# A tibble: 10 x 22
   date                date_key day_of~1 day_o~2 day_o~3 day_n~4 day_s~5 week_~6
   <dttm>                 <int>    <int>   <int>   <int> <chr>   <chr>     <int>
 1 2000-01-01 00:00:00 20000101        1       1       6 Saturd~ Sat          52
 2 2000-01-02 00:00:00 20000102        2       2       0 Sunday  Sun          52
 3 2000-01-03 00:00:00 20000103        3       3       1 Monday  Mon           1
 4 2000-01-04 00:00:00 20000104        4       4       2 Tuesday Tue           1
 5 2000-01-05 00:00:00 20000105        5       5       3 Wednes~ Wed           1
 6 2000-01-06 00:00:00 20000106        6       6       4 Thursd~ Thu           1
 7 2000-01-07 00:00:00 20000107        7       7       5 Friday  Fri           1
 8 2000-01-08 00:00:00 20000108        8       8       6 Saturd~ Sat           1
 9 2000-01-09 00:00:00 20000109        9       9       0 Sunday  Sun           1
10 2000-01-10 00:00:00 20000110       10      10       1 Monday  Mon           2
# ... with 14 more variables: week_of_month <int>, week <date>,
#   month_number <int>, month_name <chr>, month_short_name <chr>,
#   first_day_of_month <date>, last_day_of_month <date>, quarter_number <int>,
#   quarter_name <chr>, first_day_of_quarter <date>,
#   last_day_of_quarter <date>, year <int>, decade <dbl>, century <int>, and
```

```
#   abbreviated variable names 1: day_of_month, 2: day_of_year, 3: day_of_week,
#   4: day_name, 5: day_short_name, 6: week_number
```

```r
dates_processed %>%
  show_query()
```

```
<SQL>
SELECT
  *,
  CAST(to_char("date", 'yyyymmdd') AS INTEGER) AS "date_key",
  CAST(date_part('day', "date") AS INTEGER) AS "day_of_month",
  CAST(date_part('doy', "date") AS INTEGER) AS "day_of_year",
  CAST(date_part('dow', "date") AS INTEGER) AS "day_of_week",
  trim(to_char("date", 'Day')) AS "day_name",
  trim(to_char("date", 'Dy')) AS "day_short_name",
  CAST(date_part('week', "date") AS INTEGER) AS "week_number",
  CAST(to_char("date", 'W') AS INTEGER) AS "week_of_month",
  CAST(date_trunc('week', "date") AS DATE) AS "week",
  CAST(date_part('month', "date") AS INTEGER) AS "month_number",
  trim(to_char("date", 'Month')) AS "month_name",
  trim(to_char("date", 'Mon')) AS "month_short_name",
  CAST(date_trunc('month', "date") AS DATE) AS "first_day_of_month",
  CAST(date_trunc('month', "date") + interval '1 month' -
                                          interval '1 day' AS DATE) AS "last_day_of_month"
  CAST(date_part('quarter', "date") AS INTEGER) AS "quarter_number",
  trim('Q' || CAST(date_part('quarter', "date") AS INTEGER)) AS "quarter_name",
  CAST(date_trunc('quarter', "date") AS DATE) AS "first_day_of_quarter",
  CAST(date_trunc('quarter', "date") + interval '3 months' -
                                            interval '1 day' AS DATE) AS "last_day_of_quar
  CAST(date_part('year', "date") AS INTEGER) AS "year",
  CAST(date_part('decade', "date") AS INTEGER) * 10.0 AS "decade",
  CAST(date_part('century', "date") AS INTEGER) AS "century"
FROM (
  SELECT "generate_series" AS "date"
  FROM (
SELECT *
    FROM generate_series('2000-01-01'::timestamp,'2030-12-31', '1 day')
  ) "q01"
) "q02"
```

```r
ctry_pops <-
  tribble(
  ~country, ~year_1980,  ~year_1990, ~year_2000, ~year_2010,
  "Canada", 24593, 27791, 31100, 34207,
  "Mexico", 68347, 84634, 99775, 114061,
  "United States", 227225, 249623, 282162, 309326
)

ctry_pops %>%
  pivot_longer(cols = -country,
               names_to = "year",
               names_prefix = "year_",
               values_ptypes = integer(),
               values_to = "population")
```

```
# A tibble: 12 x 3
   country       year  population
   <chr>         <chr>      <int>
 1 Canada        1980       24593
 2 Canada        1990       27791
 3 Canada        2000       31100
 4 Canada        2010       34207
 5 Mexico        1980       68347
 6 Mexico        1990       84634
 7 Mexico        2000       99775
 8 Mexico        2010      114061
 9 United States 1980      227225
10 United States 1990      249623
11 United States 2000      282162
12 United States 2010      309326
```

```r
ctry_pops_db <- copy_to(pg, ctry_pops)
```

```r
ctry_pops_db %>%
  pivot_longer(cols = -country,
               names_to = "year",
               names_prefix = "year_",
               values_to = "population") %>%
  show_query()
```

<SQL>

```
(
  (
    (
      SELECT "country", '1980' AS "year", "year_1980" AS "population"
      FROM "ctry_pops"
    )
    UNION ALL
    (
      SELECT "country", '1990' AS "year", "year_1990" AS "population"
      FROM "ctry_pops"
    )
  )
  UNION ALL
  (
    SELECT "country", '2000' AS "year", "year_2000" AS "population"
    FROM "ctry_pops"
  )
)
UNION ALL
(
  SELECT "country", '2010' AS "year", "year_2010" AS "population"
  FROM "ctry_pops"
)
```

2 Time Series Analysis

2.1 Date, Datetime, and Time Manipulations

Tanimura (2021) points out that often "timestamps in the database are not encoded with the time zone, and you will need to consult with the source or developer to figure out how your data was stored." When pushing data to a PostgreSQL database, I use the `timestamp with time zone` type as much as possible.

Tanimura (2021) provides the following example, which is interesting because the west coast of the United States would not be on the PST time zone at that time of year. Instead, it would be on PDT.

```sql
SELECT '2020-09-01 00:00:00 -0' AT TIME ZONE 'pst';
```

Table 2.1: 1 records

timezone
2020-08-31 16:00:00

```sql
SELECT '2020-09-01 00:00:00 -0' AT TIME ZONE 'pdt';
```

Table 2.2: 1 records

timezone
2020-08-31 17:00:00

I think most people barely know the difference between PST and PDT and even fewer would know the exact dates that one switches from one to the other. A better approach is to use a time zone that encodes information about when PDT is used and when PST is used. In PostgreSQL, the table `pg_timezone_names` has information that we need.

```
SELECT *
FROM pg_timezone_names
WHERE name ~ '^US/';
```

Table 2.3: Displaying records 1 - 10

name	abbrev	utc_offset	is_dst
US/Alaska	AKDT	-08:00:00	TRUE
US/Pacific	PDT	-07:00:00	TRUE
US/Eastern	EDT	-04:00:00	TRUE
US/Michigan	EDT	-04:00:00	TRUE
US/Arizona	MST	-07:00:00	FALSE
US/Indiana-Starke	CDT	-05:00:00	TRUE
US/Aleutian	HDT	-09:00:00	TRUE
US/Hawaii	HST	-10:00:00	FALSE
US/East-Indiana	EDT	-04:00:00	TRUE
US/Central	CDT	-05:00:00	TRUE

```
SELECT *
FROM pg_timezone_names
WHERE abbrev IN ('PDT', 'PST')
ORDER BY name DESC
LIMIT 5;
```

Table 2.4: 5 records

name	abbrev	utc_offset	is_dst
US/Pacific	PDT	-07:00:00	TRUE
PST8PDT	PDT	-07:00:00	TRUE
Mexico/BajaNorte	PDT	-07:00:00	TRUE
Canada/Pacific	PDT	-07:00:00	TRUE
Asia/Manila	PST	08:00:00	FALSE

```
SELECT
    '2020-09-01 00:00:00 -0' AT TIME ZONE 'US/Pacific',
    '2020-09-01 00:00:00 -0' AT TIME ZONE 'PDT';
```

Table 2.5: 1 records

timezone	timezone..2
2020-08-31 17:00:00	2020-08-31 17:00:00

```
SELECT
    '2020-12-01 00:00:00 -0' AT TIME ZONE 'US/Pacific',
    '2020-12-01 00:00:00 -0' AT TIME ZONE 'PST';
```

Table 2.6: 1 records

timezone	timezone..2
2020-11-30 16:00:00	2020-11-30 16:00:00

2.1.1 Date and Timestamp Format Conversions

As discussed in Tanimura (2021), PostgreSQL has a rich array of functions for converting dates and times and extracting such information as months and days of the week.

```
SELECT date_trunc('month','2020-10-04 12:33:35'::timestamp);
```

Table 2.7: 1 records

date_trunc
2020-10-01

One such function

```
a_time_df <- tbl(pg, sql("SELECT '2020-10-04 12:33:35'::timestamp AS a_time"))
a_time_df %>%
  mutate(a_trunced_time = date_trunc('month', a_time))
```

```
# Source:   SQL [1 x 2]
# Database: postgres  [iangow@/tmp:5432/iangow]
  a_time              a_trunced_time
  <dttm>              <dttm>
1 2020-10-04 12:33:35 2020-10-01 00:00:00
```

```r
a_time_df %>%
  mutate(a_trunced_time = date_trunc('month', a_time)) %>%
  show_query()
```

```
<SQL>
SELECT *, date_trunc('month', "a_time") AS "a_trunced_time"
FROM (SELECT '2020-10-04 12:33:35'::timestamp AS a_time) "q01"
```

```r
a_time_df %>%
  collect()
```

```
# A tibble: 1 x 1
  a_time
  <dttm>
1 2020-10-04 12:33:35
```

```r
a_time_df <- tbl(pg, sql("SELECT '2020-10-04 12:33:35 US/Pacific'::timestamp with time zon

a_time_df %>%
  mutate(a_trunced_time = date_trunc('month', a_time))
```

```
# Source:   SQL [1 x 2]
# Database: postgres  [iangow@/tmp:5432/iangow]
  a_time              a_trunced_time
  <dttm>              <dttm>
1 2020-10-04 19:33:35 2020-10-01 00:00:00
```

```r
a_time_df %>%
  mutate(a_trunced_time = date_trunc('month', a_time)) %>%
  show_query()
```

```
<SQL>
SELECT *, date_trunc('month', "a_time") AS "a_trunced_time"
FROM (SELECT '2020-10-04 12:33:35 US/Pacific'::timestamp with time zone AS a_time) "q01"
```

```r
a_time_df %>%
  collect()
```

```
# A tibble: 1 x 1
  a_time
  <dttm>
1 2020-10-04 19:33:35
```

```r
a_time_df %>%
  mutate(new_time = a_time + sql("interval '3 hours'")) %>%
  collect()
```

```
# A tibble: 1 x 2
  a_time              new_time
  <dttm>              <dttm>
1 2020-10-04 19:33:35 2020-10-04 22:33:35
```

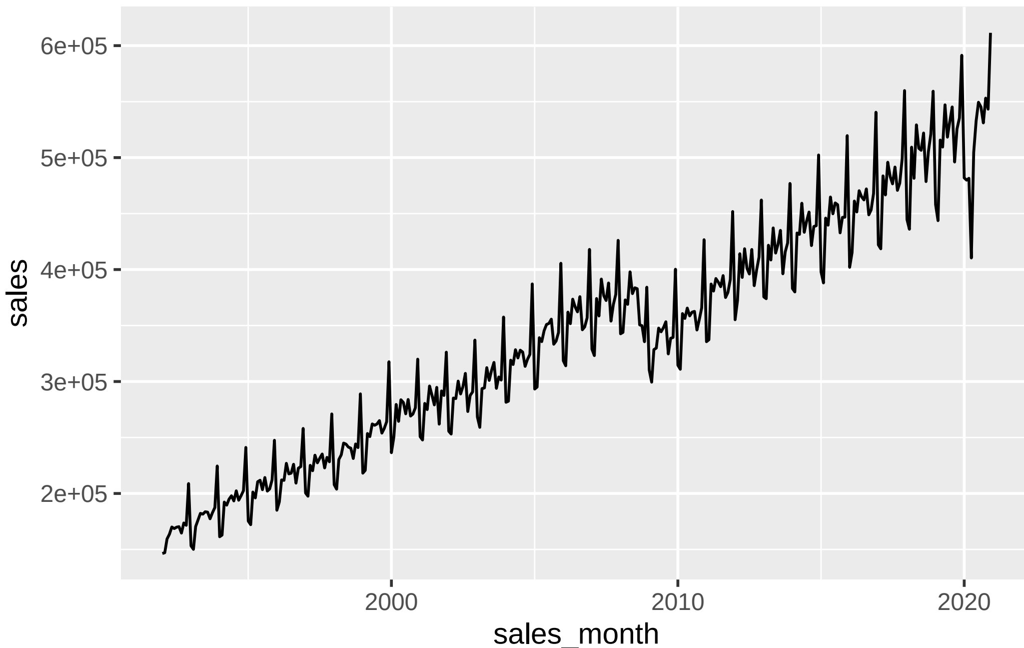
2.2 The Retail Sales Data Set

```sql
SELECT sales_month, sales
FROM retail_sales
WHERE kind_of_business = 'Retail and food services sales, total'
ORDER BY 1
```

Table 2.8: Displaying records 1 - 10

sales_month	sales
1992-01-01	146376
1992-02-01	147079
1992-03-01	159336
1992-04-01	163669
1992-05-01	170068
1992-06-01	168663
1992-07-01	169890
1992-08-01	170364
1992-09-01	164617
1992-10-01	173655

```r
retail_sales <- tbl(pg, "retail_sales")
retail_sales %>%
  filter(kind_of_business == 'Retail and food services sales, total') %>%
  select(sales_month, sales) %>%
  arrange(sales_month) %>%
  ggplot(aes(x = sales_month, y = sales)) +
  geom_line()
```



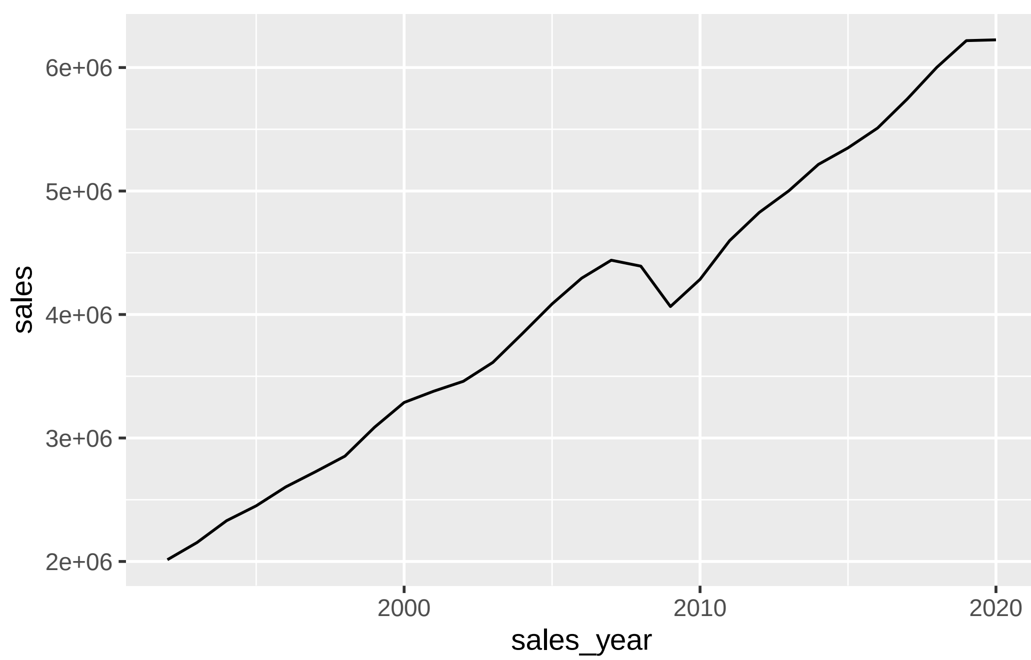

```sql
SELECT date_part('year',sales_month) as sales_year,
    sum(sales) as sales
FROM retail_sales
WHERE kind_of_business = 'Retail and food services sales, total'
GROUP BY 1
;
```

Table 2.9: Displaying records 1 - 10

sales_year	sales
2007	4439733
2005	4085746
1992	2014102

sales_year	sales
2011	4598302
2014	5215656
2006	4294359
2010	4284968
2001	3378906
2019	6218002
2018	6001623

```r
retail_sales %>%
  filter(kind_of_business == 'Retail and food services sales, total') %>%
  mutate(sales_year = date_part('year', sales_month)) %>%
  group_by(sales_year) %>%
  summarize(sales = sum(sales, na.rm = TRUE)) %>%
  arrange(sales_year) %>%
  ggplot(aes(x = sales_year, y = sales)) +
  geom_line()
```



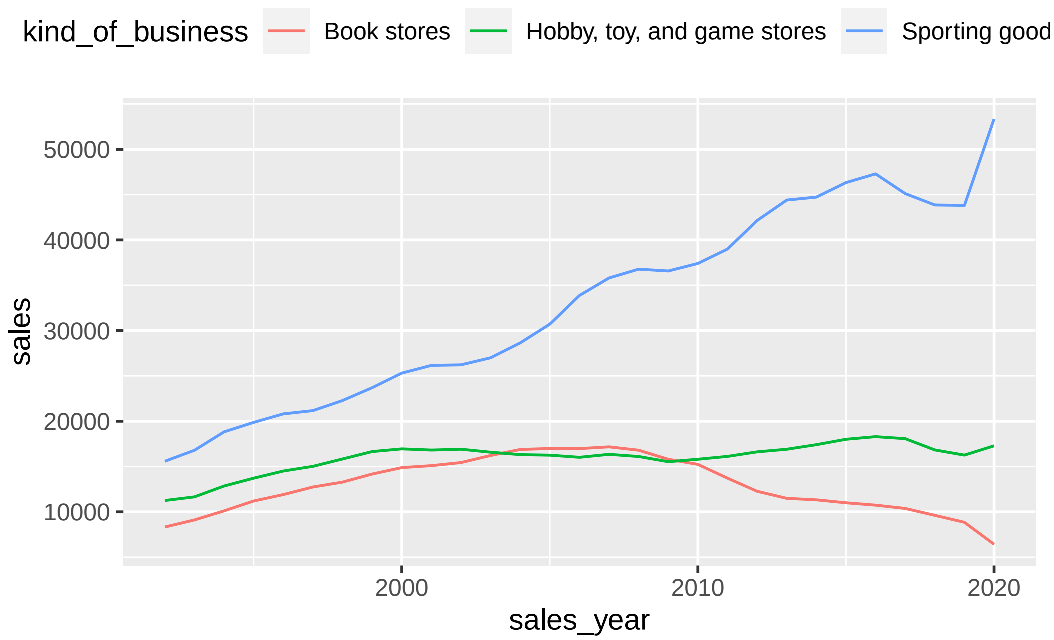

```sql
SELECT date_part('year',sales_month) as sales_year,
  kind_of_business, sum(sales) as sales
```

```sql
FROM retail_sales
WHERE kind_of_business IN
          ('Book stores',
           'Sporting goods stores',
           'Hobby, toy, and game stores')
GROUP BY 1,2
ORDER BY 1;
```

Table 2.10: Displaying records 1 - 10

sales_year	kind_of_business	sales
1992	Sporting goods stores	15583
1992	Hobby, toy, and game stores	11251
1992	Book stores	8327
1993	Hobby, toy, and game stores	11651
1993	Sporting goods stores	16791
1993	Book stores	9108
1994	Sporting goods stores	18825
1994	Book stores	10107
1994	Hobby, toy, and game stores	12850
1995	Hobby, toy, and game stores	13714

```r
retail_sales %>%
  filter(kind_of_business %in%
           c('Book stores',
             'Sporting goods stores',
             'Hobby, toy, and game stores')) %>%
  mutate(sales_year = date_part('year', sales_month)) %>%
  group_by(sales_year, kind_of_business) %>%
  summarize(sales = sum(sales, na.rm = TRUE), .groups = "drop") %>%
  arrange(sales_year) %>%
  ggplot(aes(x = sales_year, y = sales, color = kind_of_business)) +
  geom_line() +
  theme(legend.position = "top")
```



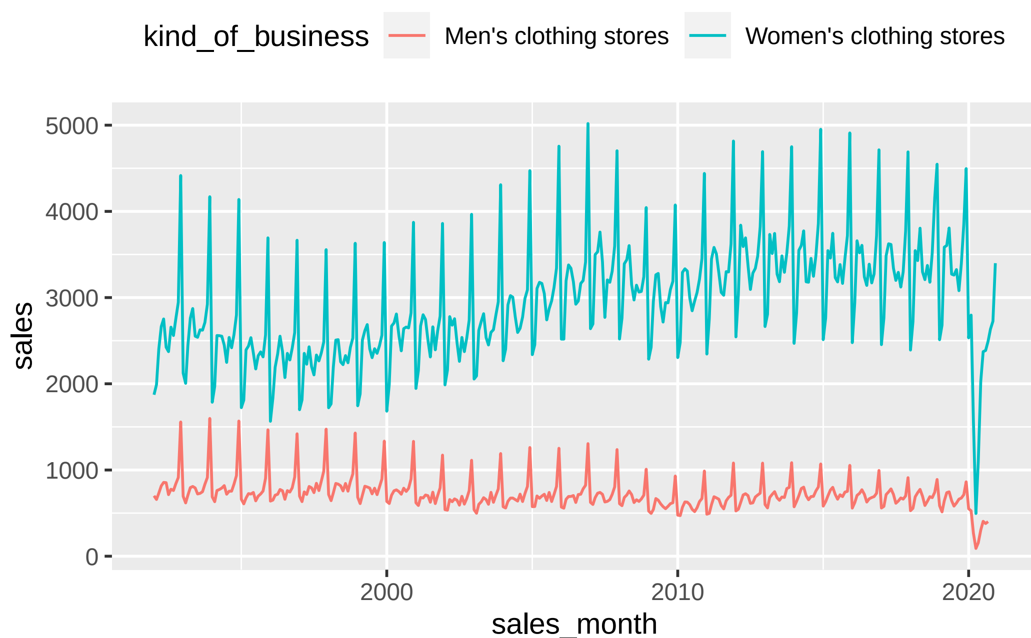

```sql
SELECT sales_month, kind_of_business, sales
FROM retail_sales
WHERE kind_of_business IN ('Men''s clothing stores','Women''s clothing stores')
ORDER BY 1,2;
```

Table 2.11: Displaying records 1 - 10

sales_month	kind_of_business	sales
1992-01-01	Men’s clothing stores	701
1992-01-01	Women’s clothing stores	1873
1992-02-01	Men’s clothing stores	658
1992-02-01	Women’s clothing stores	1991
1992-03-01	Men’s clothing stores	731
1992-03-01	Women’s clothing stores	2403
1992-04-01	Men’s clothing stores	816
1992-04-01	Women’s clothing stores	2665
1992-05-01	Men’s clothing stores	856
1992-05-01	Women’s clothing stores	2752

```r
retail_sales %>%
  filter(kind_of_business %in% c("Men's clothing stores",
```

```
                                "Women's clothing stores")) %>%
  select(sales_month, kind_of_business, sales) %>%
  arrange(sales_month) %>%
  ggplot(aes(x = sales_month, y = sales, color = kind_of_business)) +
  geom_line() +
  theme(legend.position = "top")
```




```sql
SELECT date_part('year',sales_month) as sales_year,
  kind_of_business, sum(sales) as sales
FROM retail_sales
WHERE kind_of_business IN
        ('Men''s clothing stores',
        'Women''s clothing stores')
GROUP BY 1, 2
ORDER BY 1, 2;
```

Table 2.12: Displaying records 1 - 10

sales_year	kind_of_business	sales
1992	Men's clothing stores	10179
1992	Women's clothing stores	31815
1993	Men's clothing stores	9962

sales_year	kind_of_business	sales
1993	Women's clothing stores	32350
1994	Men's clothing stores	10032
1994	Women's clothing stores	30585
1995	Men's clothing stores	9315
1995	Women's clothing stores	28696
1996	Men's clothing stores	9546
1996	Women's clothing stores	28238

```
retail_sales %>%
  filter(kind_of_business %in%
           c("Men's clothing stores",
             "Women's clothing stores")) %>%
  mutate(sales_year = date_part('year', sales_month)) %>%
  group_by(sales_year, kind_of_business) %>%
  summarize(sales = sum(sales, na.rm = TRUE), .groups = "drop") %>%
  arrange(sales_year) %>%
  ggplot(aes(x = sales_year, y = sales, color = kind_of_business)) +
  geom_line() +
  theme(legend.position = "top")
```

```sql
SELECT date_part('year', sales_month) AS sales_year,
  sum(CASE WHEN kind_of_business = 'Women''s clothing stores'
          then sales
          END) AS womens_sales,
  sum(CASE WHEN kind_of_business = 'Men''s clothing stores'
          then sales
          END) AS mens_sales
FROM retail_sales
WHERE kind_of_business IN
   ('Men''s clothing stores',
    'Women''s clothing stores')
GROUP BY 1
ORDER BY 1;
```

Table 2.13: Displaying records 1 - 10

sales_year	womens_sales	mens_sales
1992	31815	10179
1993	32350	9962
1994	30585	10032
1995	28696	9315
1996	28238	9546
1997	27822	10069
1998	28332	10196
1999	29549	9667
2000	31447	9507
2001	31453	8625

```r
pivoted_sales <-
  retail_sales %>%
  filter(kind_of_business %in%
           c("Men's clothing stores",
             "Women's clothing stores")) %>%
  mutate(kind_of_business = if_else(kind_of_business == "Women's clothing stores",
                                    "womens", "mens"),
         sales_year = date_part('year', sales_month)) %>%
  group_by(sales_year, kind_of_business) %>%
  summarize(sales = sum(sales, na.rm = TRUE), .groups = "drop") %>%
  pivot_wider(id_cols = "sales_year",
              names_from = "kind_of_business",
```

```
               names_glue = "{kind_of_business}_{.value}",
               values_from = "sales")

pivoted_sales %>%
  show_query()
```

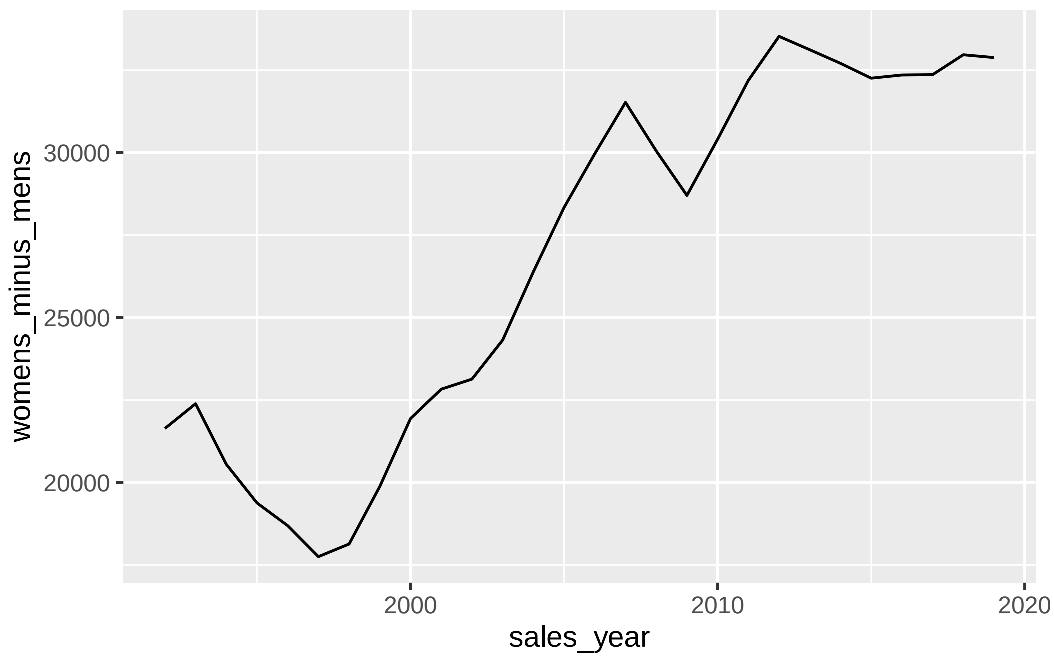
```
<SQL>
SELECT
  "sales_year",
  MAX(CASE WHEN ("kind_of_business" = 'mens') THEN "sales" END) AS "mens_sales",
  MAX(CASE WHEN ("kind_of_business" = 'womens') THEN "sales" END) AS "womens_sales"
FROM (
  SELECT "sales_year", "kind_of_business", SUM("sales") AS "sales"
  FROM (
    SELECT
      "sales_month",
      "naics_code",
      CASE WHEN ("kind_of_business" = 'Women''s clothing stores') THEN 'womens' WHEN NOT ("ki
      "reason_for_null",
      "sales",
      date_part('year', "sales_month") AS "sales_year"
    FROM "retail_sales"
    WHERE ("kind_of_business" IN ('Men''s clothing stores', 'Women''s clothing stores'))
  ) "q01"
  GROUP BY "sales_year", "kind_of_business"
) "q02"
GROUP BY "sales_year"
```

```
pivoted_sales %>%
  arrange(sales_year) %>%
  collect(n = 10) %>%
  knitr::kable()
```

sales_year	mens_sales	womens_sales
1992	10179	31815
1993	9962	32350
1994	10032	30585
1995	9315	28696
1996	9546	28238
1997	10069	27822

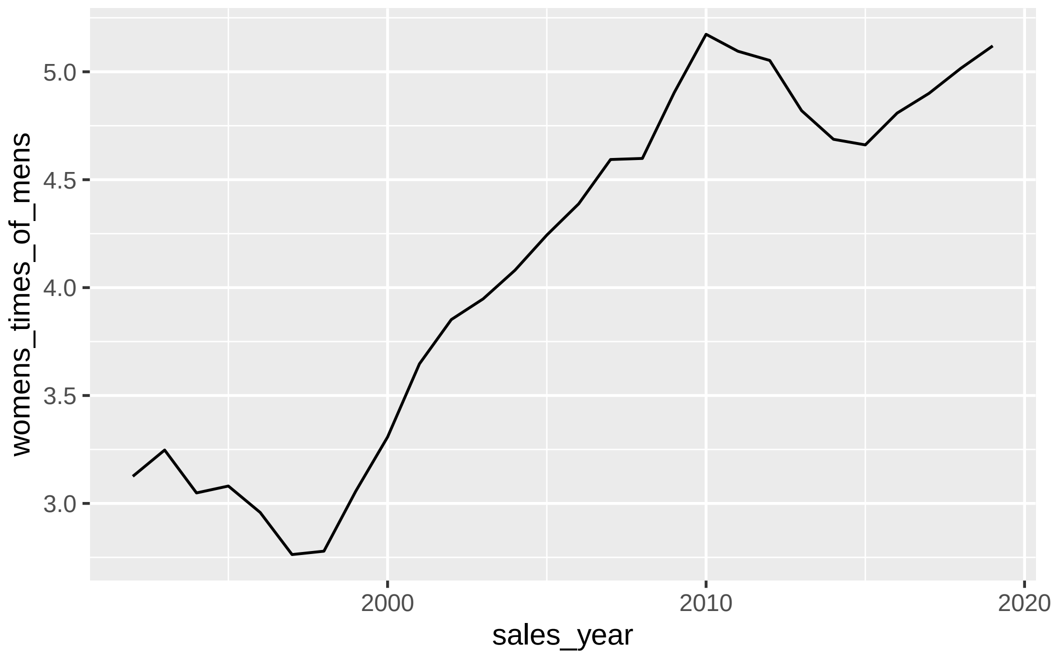
sales_year	mens_sales	womens_sales
1998	10196	28332
1999	9667	29549
2000	9507	31447
2001	8625	31453

```r
pivoted_sales %>%
  filter(sales_year <= 2019) %>%
  group_by(sales_year) %>%
  mutate(womens_minus_mens = womens_sales - mens_sales,
         mens_minus_womens = mens_sales - womens_sales) %>%
  select(sales_year, womens_minus_mens, mens_minus_womens) %>%
  arrange(sales_year) %>%
  ggplot(aes(y = womens_minus_mens, x = sales_year)) +
  geom_line()
```

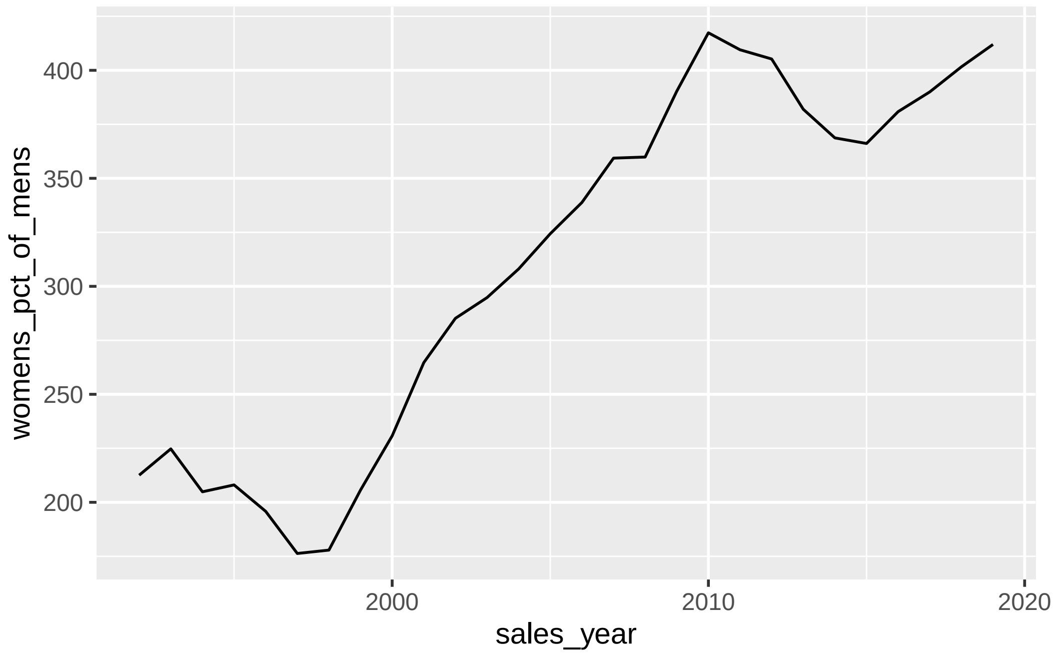



```r
pivoted_sales %>%
  filter(sales_year <= 2019) %>%
  group_by(sales_year) %>%
  mutate(womens_times_of_mens = womens_sales / mens_sales) %>%
```

```
  arrange(sales_year) %>%
  ggplot(aes(y = womens_times_of_mens, x = sales_year)) +
  geom_line()
```




```
pivoted_sales %>%
  filter(sales_year <= 2019) %>%
  group_by(sales_year) %>%
  mutate(womens_pct_of_mens = (womens_sales / mens_sales - 1) * 100) %>%
  arrange(sales_year) %>%
  ggplot(aes(y = womens_pct_of_mens, x = sales_year)) +
  geom_line()
```




```r
retail_sales %>%
  filter(kind_of_business %in%
           c("Men's clothing stores",
             "Women's clothing stores")) %>%
  group_by(sales_month) %>%
  mutate(total_sales = sum(sales)) %>%
  ungroup() %>%
  mutate(pct_total_sales = sales * 100 / total_sales) %>%
  select(sales_month, kind_of_business, pct_total_sales) %>%
  collect(n = 3)
```

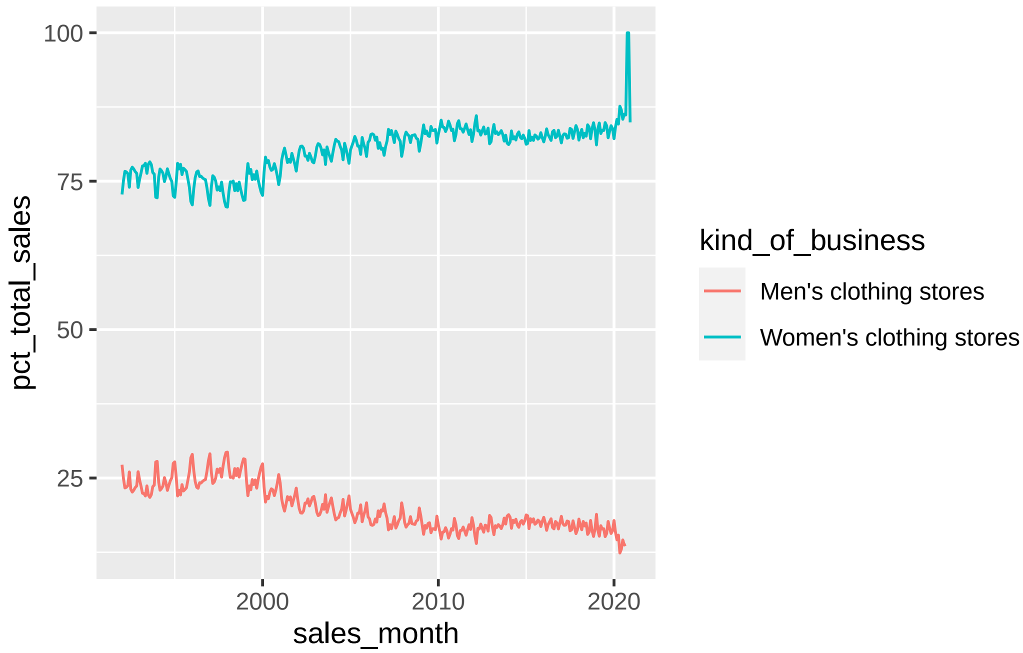
```
Warning: Missing values are always removed in SQL aggregation functions.
Use `na.rm = TRUE` to silence this warning
This warning is displayed once every 8 hours.
```

```
# A tibble: 3 x 3
  sales_month kind_of_business        pct_total_sales
  <date>      <chr>                             <dbl>
1 1992-01-01  Women's clothing stores            72.8
2 1992-01-01  Men's clothing stores              27.2
3 1992-02-01  Men's clothing stores              24.8
```

```r
retail_sales %>%
  filter(kind_of_business %in%
           c("Men's clothing stores",
             "Women's clothing stores")) %>%
  group_by(sales_month) %>%
  mutate(total_sales = sum(sales)) %>%
  ungroup() %>%
  mutate(pct_total_sales = sales * 100 / total_sales) %>%
  show_query()
```

```
<SQL>
SELECT *, ("sales" * 100.0) / "total_sales" AS "pct_total_sales"
FROM (
  SELECT *, SUM("sales") OVER (PARTITION BY "sales_month") AS "total_sales"
  FROM "retail_sales"
  WHERE ("kind_of_business" IN ('Men''s clothing stores', 'Women''s clothing stores'))
) "q01"
```

```r
retail_sales %>%
  filter(kind_of_business %in%
           c("Men's clothing stores",
             "Women's clothing stores")) %>%
  group_by(sales_month) %>%
  mutate(total_sales = sum(sales)) %>%
  ungroup() %>%
  mutate(pct_total_sales = sales * 100 / total_sales) %>%
  ggplot(aes(y = pct_total_sales, x = sales_month, color = kind_of_business)) +
  geom_line()
```




```r
retail_sales %>%
  filter(kind_of_business == "Women's clothing stores") %>%
  mutate(sales_year = date_part('year',sales_month)) %>%
  group_by(sales_year) %>%
  summarize(sales = sum(sales, na.rm = TRUE)) %>%
  ungroup() %>%
  window_order(sales_year) %>%
  mutate(index_sales = first(sales),
         pct_from_index = (sales/index_sales - 1) * 100)
```

```
# Source:     SQL [?? x 4]
# Database:   postgres  [iangow@/tmp:5432/iangow]
# Ordered by: sales_year
   sales_year sales index_sales pct_from_index
        <dbl> <dbl>       <dbl>          <dbl>
 1       1992 31815       31815           0
 2       1993 32350       31815           1.68
 3       1994 30585       31815          -3.87
 4       1995 28696       31815          -9.80
 5       1996 28238       31815         -11.2
 6       1997 27822       31815         -12.6
 7       1998 28332       31815         -10.9
```

```
 8        1999 29549       31815          -7.12
 9        2000 31447       31815          -1.16
10        2001 31453       31815          -1.14
# ... with more rows
```

```
retail_sales %>%
  filter(kind_of_business %in% c("Women's clothing stores",
                                 "Men's clothing stores"),
         sales_month <= '2019-12-31') %>%
  mutate(sales_year = date_part('year',sales_month)) %>%
  group_by(kind_of_business, sales_year) %>%
  summarize(sales = sum(sales, na.rm = TRUE), .groups = "drop") %>%
  group_by(kind_of_business) %>%
  window_order(sales_year) %>%
  mutate(index_sales = first(sales),
         pct_from_index = (sales/index_sales - 1) * 100) %>%
  ungroup() %>%
  ggplot(aes(y = pct_from_index, x = sales_year, color = kind_of_business)) +
  geom_line()
```

```r
retail_sales %>%
  filter(kind_of_business == "Women's clothing stores") %>%
  window_order(sales_month) %>%
  window_frame(-11, 0) %>%
  mutate(moving_avg = mean(sales, na.rm = TRUE),
         records_count = n()) %>%
  select(sales_month, moving_avg, records_count) %>%
  collect(n = 10)
```

```
# A tibble: 10 x 3
   sales_month moving_avg records_count
   <date>           <dbl>         <int>
 1 1992-01-01       1873              1
 2 1992-02-01       1932              2
 3 1992-03-01       2089              3
 4 1992-04-01       2233              4
 5 1992-05-01       2337.             5
 6 1992-06-01       2351.             6
 7 1992-07-01       2354.             7
 8 1992-08-01       2392.             8
 9 1992-09-01       2411.             9
10 1992-10-01       2445.            10
```

References

Tanimura, C. 2021. *SQL for Data Analysis*. O'Reilly Media. https://books.google.com.au/books?id=ojhCEAAAQBAJ.