FOR THE ENTIRETY OF THIS ASSIGNMENT, DO NOT LOAD A PACKAGE. *These are the only packages you should need to get started:.*

```
library(tidyverse)
library(dm)
library(DiagrammeR)
library(RSQLite)
library(RMariaDB)
library(duckdb)
library(duckplyr)
library(progress)
library(pixarfilms)
library(nycflights13)
library(parquetize)
```

.....

5 Question 10

5.1 Setup

```
# attach relevant packages
library(DBI)

### First steps ##################################################################

# Connection ------------------------------------------------------------------

con <- dbConnect(duckdb::duckdb())
con
```

```
<duckdb_connection cf300 driver=<duckdb_driver dbdir=':memory:' read_only=FALSE bigint=numeric
```

```
# Discover tables -------------------------------------------------------------

dbListTables(con)
```

```
character(0)
```

```
# Populate database (normally done by other people) ---------------------------

# Magic: import tables into the database
dm::copy_dm_to(
```

```
  con,
  dm::dm_pixarfilms(),
  set_key_constraints = FALSE,
  temporary = FALSE
)
```

```
Note: method with signature 'DBIConnection#Id' chosen for function 'dbExistsTable',
 target signature 'duckdb_connection#Id'.
 "duckdb_connection#ANY" would also be valid
```

```
# Discover tables ------------------------------------------------------------

dbListTables(con)
```

```
[1] "academy"         "box_office"      "genres"          "pixar_films"
[5] "pixar_people"    "public_response"
```

```
dbListFields(con, "box_office")
```

```
[1] "film"                 "budget"               "box_office_us_canada"
[4] "box_office_other"     "box_office_worldwide"
```

5.2 Exercises

```
con
```

```
<duckdb_connection cf300 driver=<duckdb_driver dbdir=':memory:' read_only=FALSE bigint=numeric>
```

5.2.1 1. List all columns from the `pixar_films` table.

```
dbListFields(con, "pixar_films")
```

```
[1] "number"       "film"         "release_date" "run_time"     "film_rating"
```

5.2.2 2. Review the help for `dbListFields()` and `dbListTables()` and the index on https://dbi.r-dbi.org/reference/.

```
help("dbListFields")
```

```
starting httpd help server ... done
```

```r
help("dbListTables")
browseURL("https://dbi.r-dbi.org/reference/")
```

5.3 Question 11

5.3.1

```r
# Discover tables ------------------------------------------------------------

dbListTables(con)
```

```
[1] "academy"         "box_office"      "genres"          "pixar_films"
[5] "pixar_people"    "public_response"
```

```r
dbListFields(con, "pixar_films")
```

```
[1] "number"       "film"         "release_date" "run_time"     "film_rating"
```

```r
dbListFields(con, "academy")
```

```
[1] "film"       "award_type" "status"
```

```r
# Read table -----------------------------------------------------------------

df_pixar_films <- dbReadTable(con, "pixar_films")
df_pixar_films
```

```
   number                film release_date run_time film_rating
1       1           Toy Story   1995-11-22       81           G
2       2        A Bug's Life   1998-11-25       95           G
3       3         Toy Story 2   1999-11-24       92           G
4       4      Monsters, Inc.   2001-11-02       92           G
5       5        Finding Nemo   2003-05-30      100           G
6       6     The Incredibles   2004-11-05      115          PG
7       7                Cars   2006-06-09      117           G
8       8         Ratatouille   2007-06-29      111           G
9       9              WALL-E   2008-06-27       98           G
10     10                  Up   2009-05-29       96          PG
11     11         Toy Story 3   2010-06-18      103           G
12     12              Cars 2   2011-06-24      106           G
13     13               Brave   2012-06-22       93          PG
14     14 Monsters University   2013-06-21      104           G
```

```
15     15          Inside Out   2015-06-19       95          PG
16     16   The Good Dinosaur   2015-11-25       93          PG
17     17        Finding Dory   2016-06-17       97          PG
18     18              Cars 3   2017-06-16      102           G
19     19                Coco   2017-11-22      105          PG
20     20       Incredibles 2   2018-06-15      118          PG
21     21         Toy Story 4   2019-06-21      100           G
22     22              Onward   2020-03-06      102          PG
23     23                Soul   2020-12-25      100          PG
24     24                Luca   2021-06-18      151         N/A
25     25         Turning Red   2022-03-11       NA         N/A
26     26           Lightyear   2022-06-17       NA         N/A
27     27                <NA>   2023-06-16      155   Not Rated
```

```
as_tibble(df_pixar_films)
```

```
# A tibble: 27 x 5
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i 17 more rows
```

```
# Execute queries ------------------------------------------------------------

dbGetQuery(con, "SELECT * FROM pixar_films")
```

```
   number               film release_date run_time film_rating
1       1          Toy Story   1995-11-22       81           G
2       2       A Bug's Life   1998-11-25       95           G
3       3        Toy Story 2   1999-11-24       92           G
4       4     Monsters, Inc.   2001-11-02       92           G
5       5       Finding Nemo   2003-05-30      100           G
6       6    The Incredibles   2004-11-05      115          PG
7       7               Cars   2006-06-09      117           G
8       8        Ratatouille   2007-06-29      111           G
9       9             WALL-E   2008-06-27       98           G
10     10                 Up   2009-05-29       96          PG
```

```
11     11         Toy Story 3   2010-06-18      103           G
12     12              Cars 2   2011-06-24      106           G
13     13               Brave   2012-06-22       93          PG
14     14 Monsters University   2013-06-21      104           G
15     15          Inside Out   2015-06-19       95          PG
16     16   The Good Dinosaur   2015-11-25       93          PG
17     17        Finding Dory   2016-06-17       97          PG
18     18              Cars 3   2017-06-16      102           G
19     19                Coco   2017-11-22      105          PG
20     20       Incredibles 2   2018-06-15      118          PG
21     21         Toy Story 4   2019-06-21      100           G
22     22              Onward   2020-03-06      102          PG
23     23                Soul   2020-12-25      100          PG
24     24                Luca   2021-06-18      151         N/A
25     25         Turning Red   2022-03-11       NA         N/A
26     26           Lightyear   2022-06-17       NA         N/A
27     27                <NA>   2023-06-16      155   Not Rated
```

```
# Assign SQL queries to character strings
sql <- "SELECT * FROM pixar_films WHERE release_date >= '2020-01-01'"

# new in R 4.1: r"()" syntax
# Kirill has used "" to indicate column names and '' for character strings
# sql <- r"(SELECT * FROM "pixar_films" WHERE "release_date" >= '2020-01-01')"
dbGetQuery(con, sql)
```

```
  number        film release_date run_time film_rating
1     22      Onward   2020-03-06      102          PG
2     23        Soul   2020-12-25      100          PG
3     24        Luca   2021-06-18      151         N/A
4     25 Turning Red   2022-03-11       NA         N/A
5     26   Lightyear   2022-06-17       NA         N/A
6     27        <NA>   2023-06-16      155   Not Rated
```

```
# Further pointers ------------------------------------------------------------

# Quoting identifiers
dbQuoteIdentifier(con, "academy")
```

```
<SQL> academy
```

```
dbQuoteIdentifier(con, "from")
```

```
<SQL> "from"
```

```r
# Quoting literals
dbQuoteLiteral(con, "Toy Story")
```

```
<SQL> 'Toy Story'
```

```r
dbQuoteLiteral(con, as.Date("2020-01-01"))
```

```
<SQL> '2020-01-01'::date
```

```r
# Paste queries with glue_sql()

# Parameterized queries
sql <- "SELECT count(*) FROM pixar_films WHERE release_date >= ?"
dbGetQuery(con, sql, params = list(as.Date("2020-01-01")))
```

```
  count_star()
1            6
```

```r
# Incomplete sql query
# sql <- paste0(
#   "SELECT * FROM",
#   dbQuoteIdentifier(con, "academy"),
#   " ",
#   "pixar_films WHERE release_date >= ?"
# )
#
# dbGetQuery(
#   con,
#   sql,
#   params = list(
#     c("Won", "Won"),
#     c("Animated Feature", "Original Song")
#   )
# )
```

```r
# Reading tables: Exercises ------------------------------------------------

con
```

```
<duckdb_connection cf300 driver=<duckdb_driver dbdir=':memory:' read_only=FALSE bigint=numeric
```

```r
# 1. Read the `academy` table.
# 2. Read all records from the `academy` table that correspond to awards won
#     - Hint: Use the query "SELECT * FROM academy WHERE status = 'Won'"
# 3. Use quoting and/or a query parameter to make the previous query more robust.
#     - Hint: `sql <- paste0("SELECT * FROM academy WHERE ", quoted_column, " = ?")`
```

5.4 Question 12

5.5 Setup

```
### Downsizing on the database ######################################################

# Connection ------------------------------------------------------------------

con <- DBI::dbConnect(duckdb::duckdb())
dm::copy_dm_to(con, dm::dm_pixarfilms(), set_key_constraints = FALSE, temporary = FALSE)

# Lazy tables -----------------------------------------------------------------

pixar_films <- tbl(con, "pixar_films")
pixar_films
```

```
# Source:   table<pixar_films> [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i more rows
```

```
# Get all data ----

df_pixar_films <-
  pixar_films |>
  collect()
df_pixar_films
```

```
# A tibble: 27 x 5
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
```

```
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i 17 more rows
```

```
# Get first 10 rows
pixar_films |>
  collect(n = 10)
```

```
# A tibble: 10 x 5
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
```

```
# Get first 10 rows
pixar_films |>
  slice_sample(n = 10)
```

```
# Source:   SQL [10 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film           release_date run_time film_rating
   <chr>  <chr>          <date>          <dbl> <chr>
 1 5      Finding Nemo   2003-05-30        100 G
 2 1      Toy Story      1995-11-22         81 G
 3 15     Inside Out     2015-06-19         95 PG
 4 17     Finding Dory   2016-06-17         97 PG
 5 18     Cars 3         2017-06-16        102 G
 6 24     Luca           2021-06-18        151 N/A
 7 19     Coco           2017-11-22        105 PG
 8 22     Onward         2020-03-06        102 PG
 9 8      Ratatouille    2007-06-29        111 G
10 4      Monsters, Inc. 2001-11-02         92 G
```

```
# Why does this work? Show_query helps
pixar_films |>
  head() |>
  show_query()
```

```
<SQL>
SELECT pixar_films.*
FROM pixar_films
LIMIT 6
```

```
# setting a seed in R session has no effect on database.
# Thus, we will need to set a seed in the database
dbExecute(con, "SELECT setseed(.42)")
```

```
[1] 0
```

```
pixar_films |>
  slice_sample(n = 10) |>
  show_query()
```

```
<SQL>
SELECT number, film, release_date, run_time, film_rating
FROM (
  SELECT pixar_films.*, ROW_NUMBER() OVER (ORDER BY RANDOM()) AS col01
  FROM pixar_films
) q01
WHERE (col01 <= 10)
```

```
# Projection (column selection)  ----------------------------------------------

pixar_films |>
  select(1:3)
```

```
# Source:   SQL [?? x 3]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date
   <chr>  <chr>           <date>
 1 1      Toy Story       1995-11-22
 2 2      A Bug's Life    1998-11-25
 3 3      Toy Story 2     1999-11-24
 4 4      Monsters, Inc.  2001-11-02
 5 5      Finding Nemo    2003-05-30
 6 6      The Incredibles 2004-11-05
 7 7      Cars            2006-06-09
```

```
 8 8      Ratatouille     2007-06-29
 9 9      WALL-E          2008-06-27
10 10     Up              2009-05-29
# i more rows
```

```
# Computations happens on the database!
pixar_films |>
  select(1:3) |>
  show_query()
```

```
<SQL>
SELECT number, film, release_date
FROM pixar_films
```

```
# Bring the data into the R session
df_pixar_films_3 <-
  pixar_films |>
  select(1:3) |>
  collect()
df_pixar_films_3
```

```
# A tibble: 27 x 3
   number film            release_date
   <chr>  <chr>           <date>
 1 1      Toy Story       1995-11-22
 2 2      A Bug's Life    1998-11-25
 3 3      Toy Story 2     1999-11-24
 4 4      Monsters, Inc.  2001-11-02
 5 5      Finding Nemo    2003-05-30
 6 6      The Incredibles 2004-11-05
 7 7      Cars            2006-06-09
 8 8      Ratatouille     2007-06-29
 9 9      WALL-E          2008-06-27
10 10     Up              2009-05-29
# i 17 more rows
```

```
# Immutable data: original data unchanged
pixar_films |>
  collect()
```

```
# A tibble: 27 x 5
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
```

```
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i 17 more rows
```

```
# regex can work
pixar_films |>
  filter(grepl("^Toy ", film)) |>
  collect()
```

```
# A tibble: 4 x 5
  number film        release_date run_time film_rating
  <chr>  <chr>       <date>          <dbl> <chr>
1 1      Toy Story   1995-11-22         81 G
2 3      Toy Story 2 1999-11-24         92 G
3 11     Toy Story 3 2010-06-18        103 G
4 21     Toy Story 4 2019-06-21        100 G
```

```
# Hypothetically, if it didn't, just modify the data frame in R
pixar_films |>
  collect() |>
  filter(grepl("^Toy ", film))
```

```
# A tibble: 4 x 5
  number film        release_date run_time film_rating
  <chr>  <chr>       <date>          <dbl> <chr>
1 1      Toy Story   1995-11-22         81 G
2 3      Toy Story 2 1999-11-24         92 G
3 11     Toy Story 3 2010-06-18        103 G
4 21     Toy Story 4 2019-06-21        100 G
```

```
# Filtering (row selection)  ------------------------------------------------

pixar_films |>
  filter(release_date >= "2020-01-01")
```

```
# Source:   SQL [6 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  number film        release_date run_time film_rating
  <chr>  <chr>       <date>          <dbl> <chr>
```

```
1 22     Onward      2020-03-06        102 PG
2 23     Soul        2020-12-25        100 PG
3 24     Luca        2021-06-18        151 N/A
4 25     Turning Red 2022-03-11         NA N/A
5 26     Lightyear   2022-06-17         NA N/A
6 27     <NA>        2023-06-16        155 Not Rated
```

```
# Computations happens on the database!
pixar_films |>
  filter(release_date >= "2020-01-01") |>
  show_query()
```

```
<SQL>
SELECT pixar_films.*
FROM pixar_films
WHERE (release_date >= '2020-01-01')
```

```
# Bring the data into the R session
df_pixar_films_202x <-
  pixar_films |>
  filter(release_date >= "2020-01-01") |>
  collect()
df_pixar_films_202x
```

```
# A tibble: 6 x 5
  number film        release_date run_time film_rating
  <chr>  <chr>       <date>          <dbl> <chr>
1 22     Onward      2020-03-06        102 PG
2 23     Soul        2020-12-25        100 PG
3 24     Luca        2021-06-18        151 N/A
4 25     Turning Red 2022-03-11         NA N/A
5 26     Lightyear   2022-06-17         NA N/A
6 27     <NA>        2023-06-16        155 Not Rated
```

```
# Immutable data: original data unchanged
pixar_films |>
  collect()
```

```
# A tibble: 27 x 5
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
```

```
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i 17 more rows
```

5.5.1 Exercises

```
# Downsizing on the database: Exercises ---------------------------------------

# `select()` ------------------------------------------------------------------

pixar_films
```

```
# Source:   table<pixar_films> [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i more rows
```

```
# *  Find several ways to select the 3 first columns
## base R
pixar_films |>
  collect() %>%
  .[, 1:3]
```

```
# A tibble: 27 x 3
   number film            release_date
   <chr>  <chr>           <date>
 1 1      Toy Story       1995-11-22
 2 2      A Bug's Life    1998-11-25
 3 3      Toy Story 2     1999-11-24
 4 4      Monsters, Inc.  2001-11-02
```

```
 5 5      Finding Nemo    2003-05-30
 6 6      The Incredibles 2004-11-05
 7 7      Cars            2006-06-09
 8 8      Ratatouille     2007-06-29
 9 9      WALL-E          2008-06-27
10 10     Up              2009-05-29
# i 17 more rows
```

```
## dplyr
pixar_films |>
  select(1:3)
```

```
# Source:   SQL [?? x 3]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date
   <chr>  <chr>           <date>
 1 1      Toy Story       1995-11-22
 2 2      A Bug's Life    1998-11-25
 3 3      Toy Story 2     1999-11-24
 4 4      Monsters, Inc.  2001-11-02
 5 5      Finding Nemo    2003-05-30
 6 6      The Incredibles 2004-11-05
 7 7      Cars            2006-06-09
 8 8      Ratatouille     2007-06-29
 9 9      WALL-E          2008-06-27
10 10     Up              2009-05-29
# i more rows
```

```
## dplyr ugly
pixar_films |>
  select(!4:ncol(pixar_films))
```

```
# Source:   SQL [?? x 3]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date
   <chr>  <chr>           <date>
 1 1      Toy Story       1995-11-22
 2 2      A Bug's Life    1998-11-25
 3 3      Toy Story 2     1999-11-24
 4 4      Monsters, Inc.  2001-11-02
 5 5      Finding Nemo    2003-05-30
 6 6      The Incredibles 2004-11-05
 7 7      Cars            2006-06-09
 8 8      Ratatouille     2007-06-29
 9 9      WALL-E          2008-06-27
10 10     Up              2009-05-29
# i more rows
```

```
## dplyr need to know column names
pixar_films |>
  select(number:release_date)
```

```
# Source:   SQL [?? x 3]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date
   <chr>  <chr>           <date>
 1 1      Toy Story       1995-11-22
 2 2      A Bug's Life    1998-11-25
 3 3      Toy Story 2     1999-11-24
 4 4      Monsters, Inc.  2001-11-02
 5 5      Finding Nemo    2003-05-30
 6 6      The Incredibles 2004-11-05
 7 7      Cars            2006-06-09
 8 8      Ratatouille     2007-06-29
 9 9      WALL-E          2008-06-27
10 10     Up              2009-05-29
# i more rows
```

```
# *  What happens if you include the name of a variable multiple times in a `select()` call?
pixar_films |>
  select(number, number)
```

```
# Source:   SQL [?? x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number
   <chr>
 1 1
 2 2
 3 3
 4 4
 5 5
 6 6
 7 7
 8 8
 9 9
10 10
# i more rows
```

```
# *  Select all columns that contain underscores (use `contains()`)
pixar_films |>
  select(contains("_"))
```

```
# Source:   SQL [?? x 3]
```

```
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   release_date run_time film_rating
   <date>          <dbl> <chr>
 1 1995-11-22         81 G
 2 1998-11-25         95 G
 3 1999-11-24         92 G
 4 2001-11-02         92 G
 5 2003-05-30        100 G
 6 2004-11-05        115 PG
 7 2006-06-09        117 G
 8 2007-06-29        111 G
 9 2008-06-27         98 G
10 2009-05-29         96 PG
# i more rows
```

```
# *  Use `all_of()` to select 2 columns of your choice
columns_of_interest = pixar_films |> colnames() |> head(n = 2)
pixar_films |>
  select(columns_of_interest)
```

```
Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
i Please use `all_of()` or `any_of()` instead.
  # Was:
  data %>% select(columns_of_interest)

  # Now:
  data %>% select(all_of(columns_of_interest))

See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```
# Source:   SQL [?? x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film
   <chr>  <chr>
 1 1      Toy Story
 2 2      A Bug's Life
 3 3      Toy Story 2
 4 4      Monsters, Inc.
 5 5      Finding Nemo
 6 6      The Incredibles
 7 7      Cars
 8 8      Ratatouille
 9 9      WALL-E
10 10     Up
# i more rows
```

```
pixar_films |>
  select(all_of(columns_of_interest))
```

```
# Source:   SQL [?? x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film
   <chr>  <chr>
 1 1      Toy Story
 2 2      A Bug's Life
 3 3      Toy Story 2
 4 4      Monsters, Inc.
 5 5      Finding Nemo
 6 6      The Incredibles
 7 7      Cars
 8 8      Ratatouille
 9 9      WALL-E
10 10     Up
# i more rows
```

```
pixar_films |>
  select(!!columns_of_interest)
```

```
# Source:   SQL [?? x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film
   <chr>  <chr>
 1 1      Toy Story
 2 2      A Bug's Life
 3 3      Toy Story 2
 4 4      Monsters, Inc.
 5 5      Finding Nemo
 6 6      The Incredibles
 7 7      Cars
 8 8      Ratatouille
 9 9      WALL-E
10 10     Up
# i more rows
```

```
# `filter()` ----------------------------------------------------------------

pixar_films
```

```
# Source:   table<pixar_films> [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date run_time film_rating
```

```
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i more rows
```

```r
# Find all films that
# 1. Are rated "PG"

filter(pixar_films, film_rating == "PG")
```

```
# Source:   SQL [10 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film              release_date run_time film_rating
   <chr>  <chr>             <date>          <dbl> <chr>
 1 6      The Incredibles   2004-11-05        115 PG
 2 10     Up                2009-05-29         96 PG
 3 13     Brave             2012-06-22         93 PG
 4 15     Inside Out        2015-06-19         95 PG
 5 16     The Good Dinosaur 2015-11-25         93 PG
 6 17     Finding Dory      2016-06-17         97 PG
 7 19     Coco              2017-11-22        105 PG
 8 20     Incredibles 2     2018-06-15        118 PG
 9 22     Onward            2020-03-06        102 PG
10 23     Soul              2020-12-25        100 PG
```

```r
# 2. Had a run time below 95

filter(pixar_films, run_time < 95)
```

```
# Source:   SQL [5 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  number film              release_date run_time film_rating
  <chr>  <chr>             <date>          <dbl> <chr>
1 1      Toy Story         1995-11-22         81 G
2 3      Toy Story 2       1999-11-24         92 G
3 4      Monsters, Inc.    2001-11-02         92 G
4 13     Brave             2012-06-22         93 PG
5 16     The Good Dinosaur 2015-11-25         93 PG
```

```r
# 3. Had a rating of "N/A" or "Not Rated"

filter(pixar_films, film_rating %in% c("N/A", "Not Rated"))
```

```
# Source:   SQL [4 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  number film        release_date run_time film_rating
  <chr>  <chr>       <date>          <dbl> <chr>
1 24     Luca        2021-06-18        151 N/A
2 25     Turning Red 2022-03-11         NA N/A
3 26     Lightyear   2022-06-17         NA N/A
4 27     <NA>        2023-06-16        155 Not Rated
```

```r
# 4. Were released after and including year 2020

filter(pixar_films, release_date >= as.Date("2020-01-01"))
```

```
# Source:   SQL [6 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  number film        release_date run_time film_rating
  <chr>  <chr>       <date>          <dbl> <chr>
1 22     Onward      2020-03-06        102 PG
2 23     Soul        2020-12-25        100 PG
3 24     Luca        2021-06-18        151 N/A
4 25     Turning Red 2022-03-11         NA N/A
5 26     Lightyear   2022-06-17         NA N/A
6 27     <NA>        2023-06-16        155 Not Rated
```

```r
# 5. Have a missing name (`film` column) or `run_time`

filter(pixar_films, is.na(film) | is.na(run_time))
```

```
# Source:   SQL [3 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  number film        release_date run_time film_rating
  <chr>  <chr>       <date>          <dbl> <chr>
1 25     Turning Red 2022-03-11         NA N/A
2 26     Lightyear   2022-06-17         NA N/A
3 27     <NA>        2023-06-16        155 Not Rated
```

```r
# 6. Are a first sequel (the name ends with "2", as in "Toy Story 2")
#     - Hint: Bring the data into the R session before filtering

filter(collect(pixar_films), grepl("2$", film))
```

```
# A tibble: 3 x 5
  number film          release_date run_time film_rating
  <chr>  <chr>         <date>          <dbl> <chr>
1 3      Toy Story 2   1999-11-24         92 G
2 12     Cars 2        2011-06-24        106 G
3 20     Incredibles 2 2018-06-15        118 PG
```

```
# `count()`, `summarize()`, `group_by()`, `ungroup()` --------------------------

pixar_films
```

```
# Source:   table<pixar_films> [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i more rows
```

```
# 1. How many films are stored in the table?

count(pixar_films)
```

```
# Source:   SQL [1 x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
      n
  <dbl>
1    27
```

```
# 2. How many films released after 2005 are stored in the table?

filter(pixar_films, release_date >= as.Date("2006-01-01")) |>
  count()
```

```
# Source:   SQL [1 x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
      n
  <dbl>
1    21
```

```r
# 3. What is the total run time of all films?
#     - Hint: Use `summarize(sum(...))`, watch out for the warning

summarize(pixar_films, total_time = sum(run_time, na.rm = TRUE))
```

```
# Source:   SQL [1 x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  total_time
       <dbl>
1       2621
```

```r
# 4. What is the total run time of all films, per rating?
#     - Hint: Use `group_by()` or `.by`

pixar_films |>
  summarize(.by = film_rating, total_time = sum(run_time, na.rm = TRUE))
```

```
# Source:   SQL [4 x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  film_rating total_time
  <chr>            <dbl>
1 G                 1301
2 N/A                151
3 Not Rated          155
4 PG                1014
```

5.6 Question 13

```r
### Downsizing on the database ######################################################

# Connection ------------------------------------------------------------------

con <- DBI::dbConnect(duckdb::duckdb())
dm::copy_dm_to(
  con,
  dm::dm_pixarfilms(),
  set_key_constraints = FALSE,
  temporary = FALSE
)

# Lazy tables -----------------------------------------------------------------

pixar_films <- tbl(con, "pixar_films")
pixar_films
```

```
# Source:   table<pixar_films> [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i more rows
```

```
# Aggregation ------------------------------------------------------------------

pixar_films |>
  summarize(
    .by = film_rating,
    n = n()
  )
```

```
# Source:   SQL [4 x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  film_rating     n
  <chr>       <dbl>
1 G              13
2 PG             10
3 Not Rated       1
4 N/A             3
```

```
# Shortcut
pixar_films |>
  count(film_rating)
```

```
# Source:   SQL [4 x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  film_rating     n
  <chr>       <dbl>
1 G              13
2 Not Rated       1
3 N/A             3
4 PG             10
```

```r
# Computations happens on the database!
pixar_films |>
  count(film_rating) |>
  show_query()
```

```
<SQL>
SELECT film_rating, COUNT(*) AS n
FROM pixar_films
GROUP BY film_rating
```

```r
# Bring the data into the R session
df_pixar_films_by_rating <-
  pixar_films |>
  count(film_rating) |>
  collect()
df_pixar_films_by_rating
```

```
# A tibble: 4 x 2
  film_rating     n
  <chr>       <dbl>
1 Not Rated       1
2 G              13
3 N/A             3
4 PG             10
```

```r
# Immutable data: original data unchanged
pixar_films |>
  collect()
```

```
# A tibble: 27 x 5
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
10 10     Up              2009-05-29         96 PG
# i 17 more rows
```

```r
# Second lazy table ------------------------------------------------------------

academy <- tbl(con, "academy")

academy
```

```
# Source:   table<academy> [?? x 3]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   film         award_type          status
   <chr>        <chr>               <chr>
 1 Toy Story    Animated Feature    Award not yet introduced
 2 Toy Story    Original Screenplay Nominated
 3 Toy Story    Adapted Screenplay  Ineligible
 4 Toy Story    Original Score      Nominated
 5 Toy Story    Original Song       Nominated
 6 Toy Story    Other               Won Special Achievement
 7 A Bug's Life Animated Feature    Award not yet introduced
 8 A Bug's Life Adapted Screenplay  Ineligible
 9 A Bug's Life Original Score      Nominated
10 Toy Story 2  Animated Feature    Award not yet introduced
# i more rows
```

```r
academy |>
  count(status)
```

```
# Source:   SQL [5 x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  status                      n
  <chr>                   <dbl>
1 Won Special Achievement     1
2 Award not yet introduced    3
3 Nominated                  36
4 Won                        17
5 Ineligible                 23
```

```r
# Left join ------

academy |>
  left_join(pixar_films)
```

```
Joining with `by = join_by(film)`
```

```
# Source:   SQL [?? x 7]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   film         award_type       status number release_date run_time film_rating
```

```
   <chr>        <chr>            <chr>  <chr>  <date>          <dbl> <chr>
 1 Toy Story    Animated Feature Award~ 1      1995-11-22         81 G
 2 Toy Story    Original Screen~ Nomin~ 1      1995-11-22         81 G
 3 Toy Story    Adapted Screenp~ Ineli~ 1      1995-11-22         81 G
 4 Toy Story    Original Score   Nomin~ 1      1995-11-22         81 G
 5 Toy Story    Original Song    Nomin~ 1      1995-11-22         81 G
 6 Toy Story    Other            Won S~ 1      1995-11-22         81 G
 7 A Bug's Life Animated Feature Award~ 2      1998-11-25         95 G
 8 A Bug's Life Adapted Screenp~ Ineli~ 2      1998-11-25         95 G
 9 A Bug's Life Original Score   Nomin~ 2      1998-11-25         95 G
10 Toy Story 2  Animated Feature Award~ 3      1999-11-24         92 G
# i more rows
```

```
academy |>
  left_join(pixar_films, join_by(film))
```

```
# Source:   SQL [?? x 7]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   film         award_type       status number release_date run_time film_rating
   <chr>        <chr>            <chr>  <chr>  <date>          <dbl> <chr>
 1 Toy Story    Animated Feature Award~ 1      1995-11-22         81 G
 2 Toy Story    Original Screen~ Nomin~ 1      1995-11-22         81 G
 3 Toy Story    Adapted Screenp~ Ineli~ 1      1995-11-22         81 G
 4 Toy Story    Original Score   Nomin~ 1      1995-11-22         81 G
 5 Toy Story    Original Song    Nomin~ 1      1995-11-22         81 G
 6 Toy Story    Other            Won S~ 1      1995-11-22         81 G
 7 A Bug's Life Animated Feature Award~ 2      1998-11-25         95 G
 8 A Bug's Life Adapted Screenp~ Ineli~ 2      1998-11-25         95 G
 9 A Bug's Life Original Score   Nomin~ 2      1998-11-25         95 G
10 Toy Story 2  Animated Feature Award~ 3      1999-11-24         92 G
# i more rows
```

```
academy |>
  left_join(pixar_films, join_by(film)) |>
  show_query()
```

```
<SQL>
SELECT academy.*, number, release_date, run_time, film_rating
FROM academy
LEFT JOIN pixar_films
  ON (academy.film = pixar_films.film)
```

```
# Join with prior computation ------

academy_won <-
  academy |>
```

```
  filter(status == "Won") |>
  count(film, name = "n_won")
academy_won
```

```
# Source:   SQL [?? x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   film            n_won
   <chr>           <dbl>
 1 Finding Nemo        1
 2 Ratatouille         1
 3 Toy Story 4         1
 4 The Incredibles     2
 5 WALL-E              1
 6 Toy Story 3         2
 7 Coco                2
 8 Up                  2
 9 Inside Out          1
10 Monsters, Inc.      1
# i more rows
```

```
pixar_films |>
  left_join(academy_won, join_by(film))
```

```
# Source:   SQL [?? x 6]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date run_time film_rating n_won
   <chr>  <chr>           <date>          <dbl> <chr>       <dbl>
 1 4      Monsters, Inc.  2001-11-02         92 G               1
 2 5      Finding Nemo    2003-05-30        100 G               1
 3 6      The Incredibles 2004-11-05        115 PG              2
 4 8      Ratatouille     2007-06-29        111 G               1
 5 9      WALL-E          2008-06-27         98 G               1
 6 10     Up              2009-05-29         96 PG              2
 7 11     Toy Story 3     2010-06-18        103 G               2
 8 13     Brave           2012-06-22         93 PG              1
 9 15     Inside Out      2015-06-19         95 PG              1
10 19     Coco            2017-11-22        105 PG              2
# i more rows
```

```
academy_won |>
  right_join(pixar_films, join_by(film)) |>
  arrange(release_date)
```

```
# Source:     SQL [?? x 6]
# Database:   DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
```

```
# Ordered by: release_date
   film            n_won number release_date run_time film_rating
   <chr>           <dbl> <chr>  <date>          <dbl> <chr>
 1 Toy Story          NA 1      1995-11-22         81 G
 2 A Bug's Life       NA 2      1998-11-25         95 G
 3 Toy Story 2        NA 3      1999-11-24         92 G
 4 Monsters, Inc.      1 4      2001-11-02         92 G
 5 Finding Nemo        1 5      2003-05-30        100 G
 6 The Incredibles     2 6      2004-11-05        115 PG
 7 Cars               NA 7      2006-06-09        117 G
 8 Ratatouille         1 8      2007-06-29        111 G
 9 WALL-E              1 9      2008-06-27         98 G
10 Up                  2 10     2009-05-29         96 PG
# i more rows
```

```
academy_won |>
  right_join(pixar_films, join_by(film)) |>
  mutate(n_won = coalesce(n_won, 0L)) |>
  arrange(release_date)
```

```
# Source:     SQL [?? x 6]
# Database:   DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
# Ordered by: release_date
   film            n_won number release_date run_time film_rating
   <chr>           <dbl> <chr>  <date>          <dbl> <chr>
 1 Toy Story           0 1      1995-11-22         81 G
 2 A Bug's Life        0 2      1998-11-25         95 G
 3 Toy Story 2         0 3      1999-11-24         92 G
 4 Monsters, Inc.      1 4      2001-11-02         92 G
 5 Finding Nemo        1 5      2003-05-30        100 G
 6 The Incredibles     2 6      2004-11-05        115 PG
 7 Cars                0 7      2006-06-09        117 G
 8 Ratatouille         1 8      2007-06-29        111 G
 9 WALL-E              1 9      2008-06-27         98 G
10 Up                  2 10     2009-05-29         96 PG
# i more rows
```

```
# important point: this SQL statement is not necessarily what we would want to
# write by hand; if putting into production, would want to simplify SQL
pixar_films |>
  left_join(academy_won, join_by(film)) |>
  mutate(n_won = coalesce(n_won, 0L)) |>
  arrange(release_date) |>
  show_query()
```

```
<SQL>
```

```
SELECT
  number,
  film,
  release_date,
  run_time,
  film_rating,
  COALESCE(n_won, 0) AS n_won
FROM (
  SELECT pixar_films.*, n_won
  FROM pixar_films
  LEFT JOIN (
    SELECT film, COUNT(*) AS n_won
    FROM (
      SELECT academy.*
      FROM academy
      WHERE (status = 'Won')
    ) q01
    GROUP BY film
  ) RHS
    ON (pixar_films.film = RHS.film)
) q01
ORDER BY release_date
```

```
# Caveat: tables must be on the same source ----------------------------------

try(
  academy |>
    left_join(pixarfilms::pixar_films, join_by(film))
)
```

```
Error in auto_copy(x, y, copy = copy, indexes = if (auto_index) list(by$y)) :
  `x` and `y` must share the same src.
i `x` is a <tbl_duckdb_connection/tbl_dbi/tbl_sql/tbl_lazy/tbl> object.
i `y` is a <tbl_df/tbl/data.frame> object.
i Set `copy = TRUE` if `y` can be copied to the same source as `x` (may be
  slow).
```

```
academy |>
  left_join(pixarfilms::pixar_films, join_by(film), copy = TRUE)
```

```
# Source:   SQL [?? x 7]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   film         award_type       status number release_date run_time film_rating
   <chr>        <chr>            <chr>  <chr>  <date>          <dbl> <chr>
 1 Toy Story    Animated Feature Award~ 1      1995-11-22         81 G
 2 Toy Story    Original Screen~ Nomin~ 1      1995-11-22         81 G
```

```
 3 Toy Story    Adapted Screenp~ Ineli~ 1      1995-11-22         81 G
 4 Toy Story    Original Score   Nomin~ 1      1995-11-22         81 G
 5 Toy Story    Original Song    Nomin~ 1      1995-11-22         81 G
 6 Toy Story    Other            Won S~ 1      1995-11-22         81 G
 7 A Bug's Life Animated Feature Award~ 2      1998-11-25         95 G
 8 A Bug's Life Adapted Screenp~ Ineli~ 2      1998-11-25         95 G
 9 A Bug's Life Original Score   Nomin~ 2      1998-11-25         95 G
10 Toy Story 2  Animated Feature Award~ 3      1999-11-24         92 G
# i more rows
```

```
academy |>
  left_join(pixarfilms::pixar_films, join_by(film), copy = TRUE) |>
  show_query()
```

```
<SQL>
SELECT academy.*, number, release_date, run_time, film_rating
FROM academy
LEFT JOIN dbplyr_2oPOuRyraA
  ON (academy.film = dbplyr_2oPOuRyraA.film)
```

```
try(
  pixarfilms::academy |>
    left_join(pixar_films, join_by(film))
)
```

```
Error in auto_copy(x, y, copy = copy) :
  `x` and `y` must share the same src.
i `x` is a <tbl_df/tbl/data.frame> object.
i `y` is a <tbl_duckdb_connection/tbl_dbi/tbl_sql/tbl_lazy/tbl> object.
i Set `copy = TRUE` if `y` can be copied to the same source as `x` (may be
  slow).
```

```
pixarfilms::academy |>
  left_join(pixar_films, join_by(film), copy = TRUE)
```

```
# A tibble: 80 x 7
   film         award_type       status number release_date run_time film_rating
   <chr>        <chr>            <chr>  <chr>  <date>          <dbl> <chr>
 1 Toy Story    Animated Feature Award~ 1      1995-11-22         81 G
 2 Toy Story    Original Screen~ Nomin~ 1      1995-11-22         81 G
 3 Toy Story    Adapted Screenp~ Ineli~ 1      1995-11-22         81 G
 4 Toy Story    Original Score   Nomin~ 1      1995-11-22         81 G
 5 Toy Story    Original Song    Nomin~ 1      1995-11-22         81 G
 6 Toy Story    Other            Won S~ 1      1995-11-22         81 G
 7 A Bug's Life Animated Feature Award~ 2      1998-11-25         95 G
```

```
 8 A Bug's Life Adapted Screenp~ Ineli~ 2      1998-11-25         95 G
 9 A Bug's Life Original Score   Nomin~ 2      1998-11-25         95 G
10 Toy Story 2  Animated Feature Award~ 3      1999-11-24         92 G
# i 70 more rows
```

```
pixar_films_db <-
  copy_to(con, pixarfilms::pixar_films)

academy |>
  left_join(pixar_films_db, join_by(film))
```

```
# Source:   SQL [?? x 7]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   film         award_type       status number release_date run_time film_rating
   <chr>        <chr>            <chr>  <chr>  <date>          <dbl> <chr>
 1 Toy Story    Animated Feature Award~ 1      1995-11-22         81 G
 2 Toy Story    Original Screen~ Nomin~ 1      1995-11-22         81 G
 3 Toy Story    Adapted Screenp~ Ineli~ 1      1995-11-22         81 G
 4 Toy Story    Original Score   Nomin~ 1      1995-11-22         81 G
 5 Toy Story    Original Song    Nomin~ 1      1995-11-22         81 G
 6 Toy Story    Other            Won S~ 1      1995-11-22         81 G
 7 A Bug's Life Animated Feature Award~ 2      1998-11-25         95 G
 8 A Bug's Life Adapted Screenp~ Ineli~ 2      1998-11-25         95 G
 9 A Bug's Life Original Score   Nomin~ 2      1998-11-25         95 G
10 Toy Story 2  Animated Feature Award~ 3      1999-11-24         92 G
# i more rows
```

```
# Downsizing on the database: Exercises ----------------------------------------

# `count()`, `summarize()`, `group_by()`, `ungroup()` --------------------------

pixar_films
```

```
# Source:   table<pixar_films> [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film            release_date run_time film_rating
   <chr>  <chr>           <date>          <dbl> <chr>
 1 1      Toy Story       1995-11-22         81 G
 2 2      A Bug's Life    1998-11-25         95 G
 3 3      Toy Story 2     1999-11-24         92 G
 4 4      Monsters, Inc.  2001-11-02         92 G
 5 5      Finding Nemo    2003-05-30        100 G
 6 6      The Incredibles 2004-11-05        115 PG
 7 7      Cars            2006-06-09        117 G
 8 8      Ratatouille     2007-06-29        111 G
 9 9      WALL-E          2008-06-27         98 G
```

```
10 10     Up               2009-05-29         96 PG
# i more rows
```

```
# 1. How many films are stored in the table?

pixar_films |>
  count()
```

```
# Source:   SQL [1 x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
      n
  <dbl>
1    27
```

```
# 2. How many films released after 2005 are stored in the table?
## their solution
pixar_films |>
  filter(release_date >= as.Date("2006-01-01"))
```

```
# Source:   SQL [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film                 release_date run_time film_rating
   <chr>  <chr>                <date>          <dbl> <chr>
 1 7      Cars                 2006-06-09        117 G
 2 8      Ratatouille          2007-06-29        111 G
 3 9      WALL-E               2008-06-27         98 G
 4 10     Up                   2009-05-29         96 PG
 5 11     Toy Story 3          2010-06-18        103 G
 6 12     Cars 2               2011-06-24        106 G
 7 13     Brave                2012-06-22         93 PG
 8 14     Monsters University  2013-06-21        104 G
 9 15     Inside Out           2015-06-19         95 PG
10 16     The Good Dinosaur    2015-11-25         93 PG
# i more rows
```

```
## better solution
pixar_films |>
  filter(year(release_date) > 2005)
```

```
# Source:   SQL [?? x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film                 release_date run_time film_rating
   <chr>  <chr>                <date>          <dbl> <chr>
 1 7      Cars                 2006-06-09        117 G
 2 8      Ratatouille          2007-06-29        111 G
```

```
 3 9      WALL-E              2008-06-27         98 G
 4 10     Up                  2009-05-29         96 PG
 5 11     Toy Story 3         2010-06-18        103 G
 6 12     Cars 2              2011-06-24        106 G
 7 13     Brave               2012-06-22         93 PG
 8 14     Monsters University 2013-06-21        104 G
 9 15     Inside Out          2015-06-19         95 PG
10 16     The Good Dinosaur   2015-11-25         93 PG
# i more rows
```

```
# 3. What is the total run time of all films?
#     - Hint: Use `summarize(sum(...))`, watch out for the warning

pixar_films |>
  summarize(total_run_time = sum(run_time))
```

```
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.

# Source:   SQL [1 x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  total_run_time
           <dbl>
1           2621
```

```
# 4. What is the total run time of all films, per rating?
#     - Hint: Use `group_by()` or `.by`

pixar_films |>
  summarize(total_run_time = sum(run_time), .by = film_rating)
```

```
# Source:   SQL [4 x 2]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  film_rating total_run_time
  <chr>                <dbl>
1 G                     1301
2 PG                    1014
3 Not Rated              155
4 N/A                    151
```

```
# `left_join()` ----------------------------------------------------------------

pixar_films |>
  left_join(academy, join_by(film))
```

```
# Source:   SQL [?? x 7]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   number film         release_date run_time film_rating award_type       status
   <chr>  <chr>        <date>          <dbl> <chr>       <chr>            <chr>
 1 1      Toy Story    1995-11-22         81 G           Animated Feature Award~
 2 1      Toy Story    1995-11-22         81 G           Original Screen~ Nomin~
 3 1      Toy Story    1995-11-22         81 G           Adapted Screenp~ Ineli~
 4 1      Toy Story    1995-11-22         81 G           Original Score   Nomin~
 5 1      Toy Story    1995-11-22         81 G           Original Song    Nomin~
 6 1      Toy Story    1995-11-22         81 G           Other            Won S~
 7 2      A Bug's Life 1998-11-25         95 G           Animated Feature Award~
 8 2      A Bug's Life 1998-11-25         95 G           Adapted Screenp~ Ineli~
 9 2      A Bug's Life 1998-11-25         95 G           Original Score   Nomin~
10 3      Toy Story 2  1999-11-24         92 G           Animated Feature Award~
# i more rows
```

```
# 1. How many rows does the join between `academy` and `pixar_films` contain?
#    Try to find out without loading all the data into memory. Explain.

left_join(pixar_films, academy, join_by(film)) |>
  count()
```

```
# Source:   SQL [1 x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
      n
  <dbl>
1    84
```

```
count(academy)
```

```
# Source:   SQL [1 x 1]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
      n
  <dbl>
1    80
```

```
# 2. Which films are not yet listed in the `academy` table? What does the
#    resulting SQL query look like?
#    - Hint: Use `anti_join()`

anti_join(pixar_films, academy, join_by(film))
```

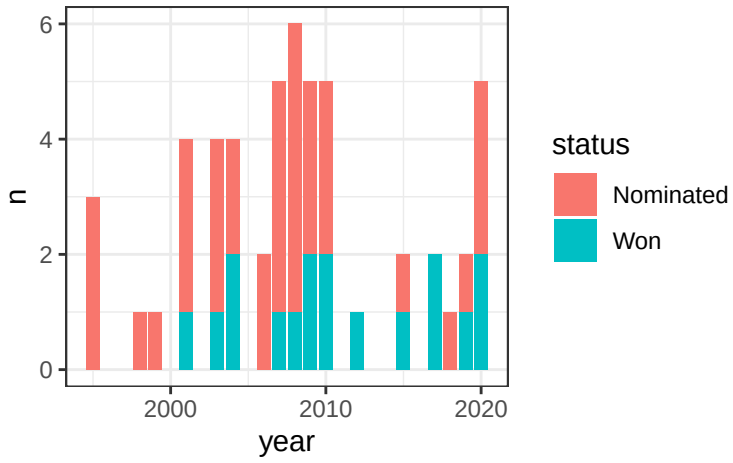
```
# Source:   SQL [4 x 5]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
  number film        release_date run_time film_rating
```

```
  <chr>  <chr>       <date>          <dbl> <chr>
1 24     Luca        2021-06-18        151 N/A
2 25     Turning Red 2022-03-11         NA N/A
3 26     Lightyear   2022-06-17         NA N/A
4 27     <NA>        2023-06-16        155 Not Rated
```

```
# 3. Plot a bar chart with the number of awards won and nominated per year.
#    Compute as much as possible on the database.
#    - Hint: "Long form" or "wide form"?


academy_won_nominated <-
  academy |>
  filter(status %in% c("Nominated", "Won")) |>
  select(film, status)

per_year_won_nominated <-
  pixar_films |>
  transmute(film, year = year(release_date)) |>
  inner_join(academy_won_nominated, join_by(film)) |>
  count(year, status) |>
  collect()
per_year_won_nominated
```

```
# A tibble: 27 x 3
    year status        n
   <dbl> <chr>     <dbl>
 1  2018 Nominated     1
 2  2003 Nominated     3
 3  1999 Nominated     1
 4  2015 Won           1
 5  2017 Won           2
 6  2020 Nominated     3
 7  2006 Nominated     2
 8  2007 Won           1
 9  2009 Won           2
10  2008 Won           1
# i 17 more rows
```

```
ggplot(per_year_won_nominated, aes(x = year, y = n, fill = status)) +
  geom_col()
```



5.7 Question 21

5.7.1

```
library(DBI)
library(tidyverse)
requireNamespace("duckplyr")
```

```
### Working with database dumps ########################################################

# Create data ------------------------------------------------------------------

arrow::write_parquet(nycflights13::flights, "flights.parquet")

fs::file_size("flights.parquet")
```

```
5.43M
```

```
object.size(nycflights13::flights)
```

```
40650104 bytes
```

```
# Processing the local data ----

# Read as tibble ----

df <- arrow::read_parquet("flights.parquet")
df
```

```
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      517            515         2      830            819
 2  2013     1     1      533            529         4      850            830
 3  2013     1     1      542            540         2      923            850
 4  2013     1     1      544            545        -1     1004           1022
 5  2013     1     1      554            600        -6      812            837
 6  2013     1     1      554            558        -4      740            728
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# i 336,766 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```r
# Read as Arrow dataset ----

ds <- arrow::open_dataset("flights.parquet")
ds
```

```
FileSystemDataset with 1 Parquet file
19 columns
year: int32
month: int32
day: int32
dep_time: int32
sched_dep_time: int32
dep_delay: double
arr_time: int32
sched_arr_time: int32
arr_delay: double
carrier: string
flight: int32
tailnum: string
origin: string
dest: string
air_time: double
distance: double
hour: double
minute: double
time_hour: timestamp[us, tz=America/New_York]
```

```
ds |>
  count(year, month, day) |>
  collect()
```

```
# A tibble: 365 x 4
    year month   day     n
   <int> <int> <int> <int>
 1  2013     1     1   842
 2  2013     1     2   943
 3  2013     1     3   914
 4  2013     1     4   915
 5  2013     1     5   720
 6  2013     1     6   832
 7  2013     1     7   933
 8  2013     1     8   899
 9  2013     1     9   902
10  2013     1    10   932
# i 355 more rows
```

```
# Register as duckdb lazy table ----

con_memory <- dbConnect(duckdb::duckdb(), dbdir = ":memory:")

tbl <- duckdb::tbl_file(con_memory, "flights.parquet")
tbl
```

```
# Source:   SQL [?? x 19]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      517            515         2      830            819
 2  2013     1     1      533            529         4      850            830
 3  2013     1     1      542            540         2      923            850
 4  2013     1     1      544            545        -1     1004           1022
 5  2013     1     1      554            600        -6      812            837
 6  2013     1     1      554            558        -4      740            728
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# i more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
class(tbl)
```

```
[1] "tbl_duckdb_connection" "tbl_dbi"               "tbl_sql"
[4] "tbl_lazy"              "tbl"
```

```
tbl |>
  count(year, month, day)
```

```
# Source:   SQL [?? x 4]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
    year month   day     n
   <int> <int> <int> <dbl>
 1  2013     1     1   842
 2  2013     1     2   943
 3  2013     1     3   914
 4  2013     1     4   915
 5  2013     1     5   720
 6  2013     1     6   832
 7  2013     1     7   933
 8  2013     1     8   899
 9  2013     1     9   902
10  2013     1    10   932
# i more rows
```

```
tbl |>
  count(year, month, day) |>
  filter(month == 1) |>
  explain()
```

```
<SQL>
SELECT "year", "month", "day", COUNT(*) AS n
FROM (FROM 'flights.parquet') q01
GROUP BY "year", "month", "day"
HAVING ("month" = 1.0)

<PLAN>
physical_plan

          PROJECTION

 __internal_decompress_integ
    ral_integer(#0, 2013)
 __internal_decompress_integ
      ral_integer(#1, 1)
 __internal_decompress_integ
```

```
      ral_integer(#2, 1)
              #3


   PERFECT_HASH_GROUP_BY

              #0
              #1
              #2
         count_star()


          PROJECTION

             year
            month
             day


          PROJECTION

__internal_compress_integra
    l_utinyint(#0, 2013)
__internal_compress_integra
     l_utinyint(#1, 1)
__internal_compress_integra
     l_utinyint(#2, 1)


        PARQUET_SCAN

             year
            month
             day

 Filters: month=1 AND month
          IS NOT NULL

          EC: 67355
```

```
# The future: Register as duckplyr lazy data frame ----

duckplyr_df <- duckplyr::duckplyr_df_from_parquet("flights.parquet")
class(duckplyr_df)
```

```
[1] "duckplyr_df" "tbl_df"      "tbl"         "data.frame"
```

```r
filtered <-
  duckplyr_df |>
  count(year, month, day) |>
  filter(month == 1)

filtered |>
  explain()
```

```
          PROJECTION

__internal_decompress_integ
   ral_integer(#0, 2013)
__internal_decompress_integ
     ral_integer(#1, 1)
__internal_decompress_integ
     ral_integer(#2, 1)
              #3


           ORDER_BY

           ORDERS:
  read_parquet."year" ASC
  read_parquet."month" ASC
   read_parquet."day" ASC


          PROJECTION

__internal_compress_integra
    l_utinyint(#0, 2013)
__internal_compress_integra
     l_utinyint(#1, 1)
__internal_compress_integra
     l_utinyint(#2, 1)
              #3


          PROJECTION

             year
            month
             day
              n
```

```
        PROJECTION

__internal_decompress_integ
   ral_integer(#0, 2013)
__internal_decompress_integ
     ral_integer(#1, 1)
__internal_decompress_integ
     ral_integer(#2, 1)
             #3


   PERFECT_HASH_GROUP_BY

             #0
             #1
             #2
        count_star()


         PROJECTION

            year
           month
            day


         PROJECTION

__internal_compress_integra
    l_utinyint(#0, 2013)
__internal_compress_integra
     l_utinyint(#1, 1)
__internal_compress_integra
     l_utinyint(#2, 1)


           FILTER

   r_base::==(month, 1.0)

         EC: 67355


       READ_PARQUET

            year
           month
            day
```

```
          EC: 336776
```

```
filtered
```

```
materializing:
---------------------
--- Relation Tree ---
---------------------
Filter [==("month", 1.0)]
  Order ["year" ASC, "month" ASC, "day" ASC]
    Aggregate ["year", "month", "day", n()]
      read_parquet(flights.parquet)

---------------------
-- Result Columns  --
---------------------
- year (INTEGER)
- month (INTEGER)
- day (INTEGER)
- n (INTEGER)

# A tibble: 31 x 4
    year month   day     n
   <int> <int> <int> <int>
 1  2013     1     1   842
 2  2013     1     2   943
 3  2013     1     3   914
 4  2013     1     4   915
 5  2013     1     5   720
 6  2013     1     6   832
 7  2013     1     7   933
 8  2013     1     8   899
 9  2013     1     9   902
10  2013     1    10   932
# i 21 more rows
```

```
filtered |>
  explain()
```

```
      R_DATAFRAME_SCAN

          data.frame
```

```
             year
            month
             day
              n

            EC: 31
```

```
duckplyr_df |>
  count(year, month, day) |>
  filter(month == 1L) |>
  explain()
```

```
          PROJECTION

 __internal_decompress_integ
    ral_integer(#0, 2013)
 __internal_decompress_integ
      ral_integer(#1, 1)
 __internal_decompress_integ
      ral_integer(#2, 1)
               #3


           ORDER_BY

           ORDERS:
   read_parquet."year" ASC
   read_parquet."month" ASC
    read_parquet."day" ASC


          PROJECTION

 __internal_compress_integra
     l_utinyint(#0, 2013)
 __internal_compress_integra
      l_utinyint(#1, 1)
 __internal_compress_integra
      l_utinyint(#2, 1)
               #3


          PROJECTION

             year
```

```
          month
           day
            n


         PROJECTION

__internal_decompress_integ
   ral_integer(#0, 2013)
__internal_decompress_integ
     ral_integer(#1, 1)
__internal_decompress_integ
     ral_integer(#2, 1)
             #3


   PERFECT_HASH_GROUP_BY

             #0
             #1
             #2
        count_star()


         PROJECTION

            year
           month
            day


         PROJECTION

__internal_compress_integra
    l_utinyint(#0, 2013)
__internal_compress_integra
     l_utinyint(#1, 1)
__internal_compress_integra
     l_utinyint(#2, 1)


           FILTER

    r_base::==(month, 1)

         EC: 67355
```

```
        READ_PARQUET

             year
            month
             day

          EC: 336776
```

```r
# Create partitioned data ------------------------------------------------------

arrow::write_dataset(
  nycflights13::flights,
  "flights-part/",
  partitioning = c("year", "month")
)

fs::dir_tree("flights-part")
```

```
flights-part
\-- year=2013
    +-- month=1
    |   \-- part-0.parquet
    +-- month=10
    |   \-- part-0.parquet
    +-- month=11
    |   \-- part-0.parquet
    +-- month=12
    |   \-- part-0.parquet
    +-- month=2
    |   \-- part-0.parquet
    +-- month=3
    |   \-- part-0.parquet
    +-- month=4
    |   \-- part-0.parquet
    +-- month=5
    |   \-- part-0.parquet
    +-- month=6
    |   \-- part-0.parquet
    +-- month=7
    |   \-- part-0.parquet
    +-- month=8
    |   \-- part-0.parquet
    \-- month=9
        \-- part-0.parquet
```

```r
# Read partitioned data ------------------------------------------------------

tbl_part <- duckdb::tbl_query(
  con_memory,
  "read_parquet('flights-part/*/*/*.parquet', hive_partitioning = true)"
)
tbl_part
```

```
# Source:   SQL [?? x 19]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
     day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
   <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>
 1     1      447            500       -13      614            648       -34
 2     1      522            517         5      735            757       -22
 3     1      536            545        -9      809            855       -46
 4     1      539            545        -6      801            827       -26
 5     1      539            545        -6      917            933       -16
 6     1      544            550        -6      912            932       -20
 7     1      549            600       -11      653            716       -23
 8     1      550            600       -10      648            700       -12
 9     1      550            600       -10      649            659       -10
10     1      551            600        -9      727            730        -3
# i more rows
# i 12 more variables: carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>, month <dbl>, year <dbl>
```

```r
class(tbl_part)
```

```
[1] "tbl_duckdb_connection" "tbl_dbi"               "tbl_sql"
[4] "tbl_lazy"              "tbl"
```

```r
tbl_part |>
  count(year, month, day)
```

```
# Source:   SQL [?? x 4]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
    year month   day     n
   <dbl> <dbl> <int> <dbl>
 1  2013    11     2   689
 2  2013    11    10   895
 3  2013    12    19   974
 4  2013    12    27   963
 5  2013     6    11   980
 6  2013     6    15   801
```

```
 7  2013     6    16   918
 8  2013     6    22   812
 9  2013     6    26   995
10  2013     5     7   955
# i more rows
```

```
tbl_part |>
  filter(month %in% 1:3) |>
  explain()
```

```
<SQL>
SELECT q01.*
FROM (FROM read_parquet('flights-part/*/*/*.parquet', hive_partitioning = true)) q01
WHERE ("month" IN (1, 2, 3))

<PLAN>
physical_plan

        READ_PARQUET

             day
          dep_time
        sched_dep_time
          dep_delay
           arr_time
        sched_arr_time
          arr_delay
           carrier
            flight
           tailnum
            origin
             dest
           air_time
           distance
             hour
            minute
          time_hour
            month
             year

 File Filters: (month IN (1,
           2, 3))

          EC: 86667
```

```r
# Create CSV data ----------------------------------------------------------------

readr::write_csv(nycflights13::flights, "flights.csv")

# Read CSV data ----------------------------------------------------------------

tbl_csv <- duckdb::tbl_file(con_memory, "flights.csv")

tbl_csv |>
  count(year, month, day)
```

```
# Source:   SQL [?? x 4]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
    year month   day     n
   <dbl> <dbl> <dbl> <dbl>
 1  2013    12     4   958
 2  2013    12    23   985
 3  2013    12    24   761
 4  2013     2     5   896
 5  2013     2     9   684
 6  2013     2    14   956
 7  2013     2    17   848
 8  2013     3     1   958
 9  2013     3    21   980
10  2013     3    25   978
# i more rows
```

```r
tbl_csv |>
  count(year, month, day) |>
  explain()
```

```
<SQL>
SELECT "year", "month", "day", COUNT(*) AS n
FROM (FROM 'flights.csv') q01
GROUP BY "year", "month", "day"

<PLAN>
physical_plan

        HASH_GROUP_BY

              #0
              #1
              #2
         count_star()
```

```
          PROJECTION

             year
            month
             day


        READ_CSV_AUTO

             year
            month
             day

          EC: 326882
```

```
duckplyr_df_csv <- duckplyr::duckplyr_df_from_csv("flights.csv")

duckplyr_df_csv |>
  count(year, month, day)
```

```
materializing:
---------------------
--- Relation Tree ---
---------------------
Order ["year" ASC, "month" ASC, "day" ASC]
  Aggregate ["year", "month", "day", n()]
    read_csv_auto(flights.csv)

---------------------
-- Result Columns  --
---------------------
- year (BIGINT)
- month (BIGINT)
- day (BIGINT)
- n (INTEGER)

# A tibble: 365 x 4
    year month   day     n
   <dbl> <dbl> <dbl> <int>
 1  2013     1     1   842
 2  2013     1     2   943
 3  2013     1     3   914
 4  2013     1     4   915
 5  2013     1     5   720
 6  2013     1     6   832
```

```
 7  2013     1     7   933
 8  2013     1     8   899
 9  2013     1     9   902
10  2013     1    10   932
# i 355 more rows
```

```
duckplyr_df_csv |>
  count(year, month, day) |>
  explain()
```

```
          ORDER_BY

          ORDERS:
   read_csv_auto."year" ASC
  read_csv_auto."month" ASC
   read_csv_auto."day" ASC


         PROJECTION

            year
           month
            day
             n


       HASH_GROUP_BY

            #0
            #1
            #2
         count_star()


         PROJECTION

            year
           month
            day


       READ_CSV_AUTO

            year
           month
            day
```

```
          EC: 326882
```

```
# Create derived Parquet data with duckplyr ------------------------------------------------

duckplyr_df_csv |>
  count(year, month, day) |>
  duckplyr::df_to_parquet("flights-count.parquet")
```

```
materializing:
---------------------
--- Relation Tree ---
---------------------
r_dataframe_scan(0x193623e87a0)

---------------------
-- Result Columns  --
---------------------
- year (DOUBLE)
- month (DOUBLE)
- day (DOUBLE)
- dep_time (VARCHAR)
- sched_dep_time (DOUBLE)
- dep_delay (VARCHAR)
- arr_time (VARCHAR)
- sched_arr_time (DOUBLE)
- arr_delay (VARCHAR)
- carrier (VARCHAR)
- flight (DOUBLE)
- tailnum (VARCHAR)
- origin (VARCHAR)
- dest (VARCHAR)
- air_time (VARCHAR)
- distance (DOUBLE)
- hour (DOUBLE)
- minute (DOUBLE)
- time_hour (TIMESTAMP)
```

```
fs::file_size("flights-count.parquet")
```

```
2.03K
```

```
duckplyr_df_count <-
  duckplyr::duckplyr_df_from_parquet("flights-count.parquet")
```

```
duckplyr_df_count |>
  explain()
```

```
        READ_PARQUET

             year
            month
             day
              n

           EC: 365
```

```
duckplyr_df_count
```

```
materializing:
---------------------
--- Relation Tree ---
---------------------
read_parquet(flights-count.parquet)

---------------------
-- Result Columns  --
---------------------
- year (DOUBLE)
- month (DOUBLE)
- day (DOUBLE)
- n (INTEGER)

# A tibble: 365 x 4
    year month   day     n
   <dbl> <dbl> <dbl> <int>
 1  2013     1     1   842
 2  2013     1     2   943
 3  2013     1     3   914
 4  2013     1     4   915
 5  2013     1     5   720
 6  2013     1     6   832
 7  2013     1     7   933
 8  2013     1     8   899
 9  2013     1     9   902
10  2013     1    10   932
# i 355 more rows
```

```
duckplyr_df_count |>
  explain()
```

```
      R_DATAFRAME_SCAN

          data.frame

             year
            month
             day
              n

           EC: 365
```

5.7.2 Exercises

```
# Exercises ------------------------------------------------------------------

arrow::write_parquet(nycflights13::flights, "flights.parquet")

# 1. From the Parquet file, compute a lazy dbplyr tables
#    showing the mean and median departure delay
#    for each month.

con <- dbConnect(duckdb::duckdb(), dbdir = ":memory:")

flights <- duckdb::tbl_file(con, "flights.parquet")

month_delay <-
  flights |>
  summarise(
    .by = month,
    mean_delay = mean(dep_delay),
    median_delay = median(dep_delay)
  )

month_delay
```

```
# Source:   SQL [?? x 3]
# Database: DuckDB v1.0.0 [rmcshane@Windows 10 x64:R 4.4.1/:memory:]
   month mean_delay median_delay
   <int>      <dbl>        <dbl>
```

```
 1     1      10.0            -2
 2     2      10.8            -2
 3     3      13.2            -1
 4     4      13.9            -2
 5     5      13.0            -1
 6     6      20.8             0
 7     7      21.7             0
 8     8      12.6            -1
 9     9       6.72           -3
10    10       6.24           -3
# i more rows
```

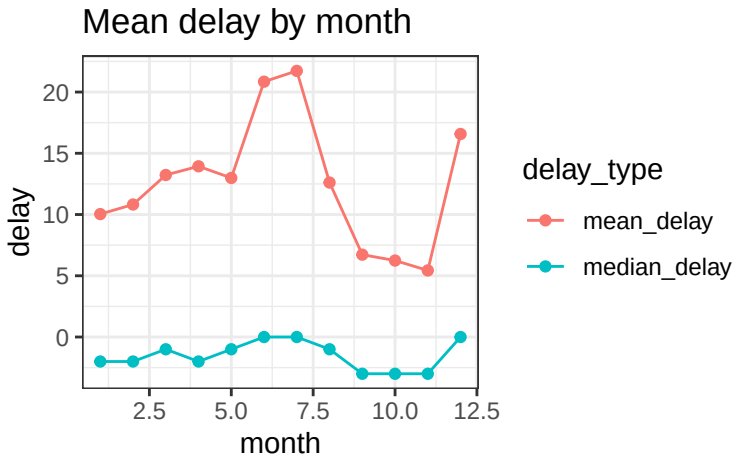
```
# 2. Compute the same data as duckplyr lazy data frames.

nycflights13::flights |>
  select(month, dep_delay) |>
  duckplyr::as_duckplyr_df() |>
  summarise(
    .by = month,
    mean_delay = mean(dep_delay),
    median_delay = median(dep_delay)
  )
```

```
materializing:
---------------------
--- Relation Tree ---
---------------------
Aggregate ["month", mean(dep_delay), median(dep_delay)]
  r_dataframe_scan(0x1936ab44628)

---------------------
-- Result Columns  --
---------------------
- month (INTEGER)
- mean_delay (DOUBLE)
- median_delay (DOUBLE)

# A tibble: 12 x 3
   month mean_delay median_delay
   <int>      <dbl>        <dbl>
 1    11       5.44           -3
 2     1      10.0            -2
 3     4      13.9            -2
 4    10       6.24           -3
 5     2      10.8            -2
 6     7      21.7             0
 7     8      12.6            -1
```

```
 8     5      13.0            -1
 9     9       6.72           -3
10    12      16.6             0
11     3      13.2            -1
12     6      20.8             0
```

```r
# 3. Store this data as a Parquet file.

nycflights13::flights |>
  select(month, dep_delay) |>
  duckplyr::as_duckplyr_df() |>
  summarise(
    .by = month,
    mean_delay = mean(dep_delay),
    median_delay = median(dep_delay),
  ) |>
  duckplyr::df_to_parquet("delay-by-month.parquet")

# 4. Read the Parquet file and plot the data.

library(ggplot2)

duckplyr::duckplyr_df_from_parquet("delay-by-month.parquet") |>
  pivot_longer(cols = c(mean_delay, median_delay), names_to = "delay_type", values_to = "delay
  ggplot(aes(x = month, y = delay, color = delay_type)) +
  geom_point() +
  geom_line() +
  labs(title = "Mean delay by month")
```

```
materializing:
---------------------
--- Relation Tree ---
---------------------
read_parquet(delay-by-month.parquet)

---------------------
-- Result Columns  --
---------------------
- month (INTEGER)
- mean_delay (DOUBLE)
- median_delay (DOUBLE)
```


Mean delay by month
delay
20
15
10
5
0
2.5
5.0
7.5
10.0
12.5
month
delay_type
mean_delay
median_delay