Lecture 18: Oct 17, 2018

SQL

- *Connecting to a Database*
- *Structured Query Language (SQL)*
  - *Creating and Filling Tables*
  - *Updating a Record*
  - *Deleting Records*
  - *Writing Queries*
- *Resources*

James Balamuta
STAT 385 @ UIUC

Announcements

- **hw07** is due **Friday, Oct 26th, 2018** at **6:00 PM**
- **Office Hour Changes**
  - **John Lee's** are now from **4 - 5 PM** on **WF**
  - **Hassan Kamil's** are now from **2:30 - 3:30 PM** on **TR**
- **Quiz 08** covers Week 7 contents @ CBTF.
  - Window: Oct 16th - 18th
  - Sign up: https://cbtf.engr.illinois.edu/sched
- Want to review your homework or quiz grades?
  Schedule an appointment.

Last Time

- **Databases**
  - Collection of data
  - Data tables mirror *R*'s data frame
- **Keys and Relationships**
  - Keys are unique field(s) to identify rows in data.
  - Relationships show how data is connected between tables
- **Joins**
  - Naively merging data is rarely a good idea.
  - Mutating, Filtering, and Set Joins are better for a varying number of rows between data sets.

Lecture Objectives

- **Connect** to a SQL Database.
- **Create** database table schema.
- **Write** SQL queries that select and summarize data.

Connecting to a Database

Previously

Definition:
Database refers to a collection of different tabular pieces of data.

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Grades

student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

Courses

course_id	acronym
STAT385	SPM
STAT432	BSL
HIST100	GH

Definition:
Relational Database Management Systems (RDBMS) is an underlying system that oversees data with interconnections between tables.

Students

Primary Key

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Grades

Foreign Key Foreign Key

student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

Courses

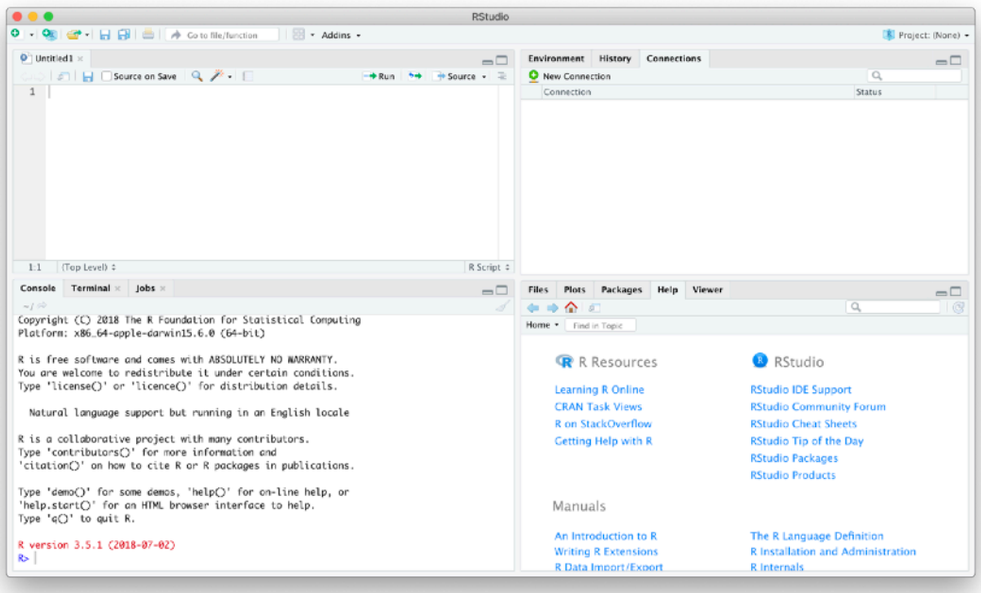
Primary Key

course_id	acronym
STAT385	SPM
STAT432	BSL
HIST100	GH

Connection Overview

… working with a database in *R* …

Analysis in *R*



Query

```
SELECT *
FROM subject_heights
```

Database Server*

Response

id	sex	height
1	M	6.1
2	F	5.5
3	F	5.2
…	…	…
55	M	5.9

---

* Databases are drawn as either a stack of "discs" or "cans" to pay tribute to hard drives, which use platters to store information long-term.

Communicating with a Database

- *RDBMS* implement different dialects of data manipulation languages.
    - e.g. *SQLite* is different from *Oracle*.

- The **DBI** package provides a common interface for interacting with each RDBMS.

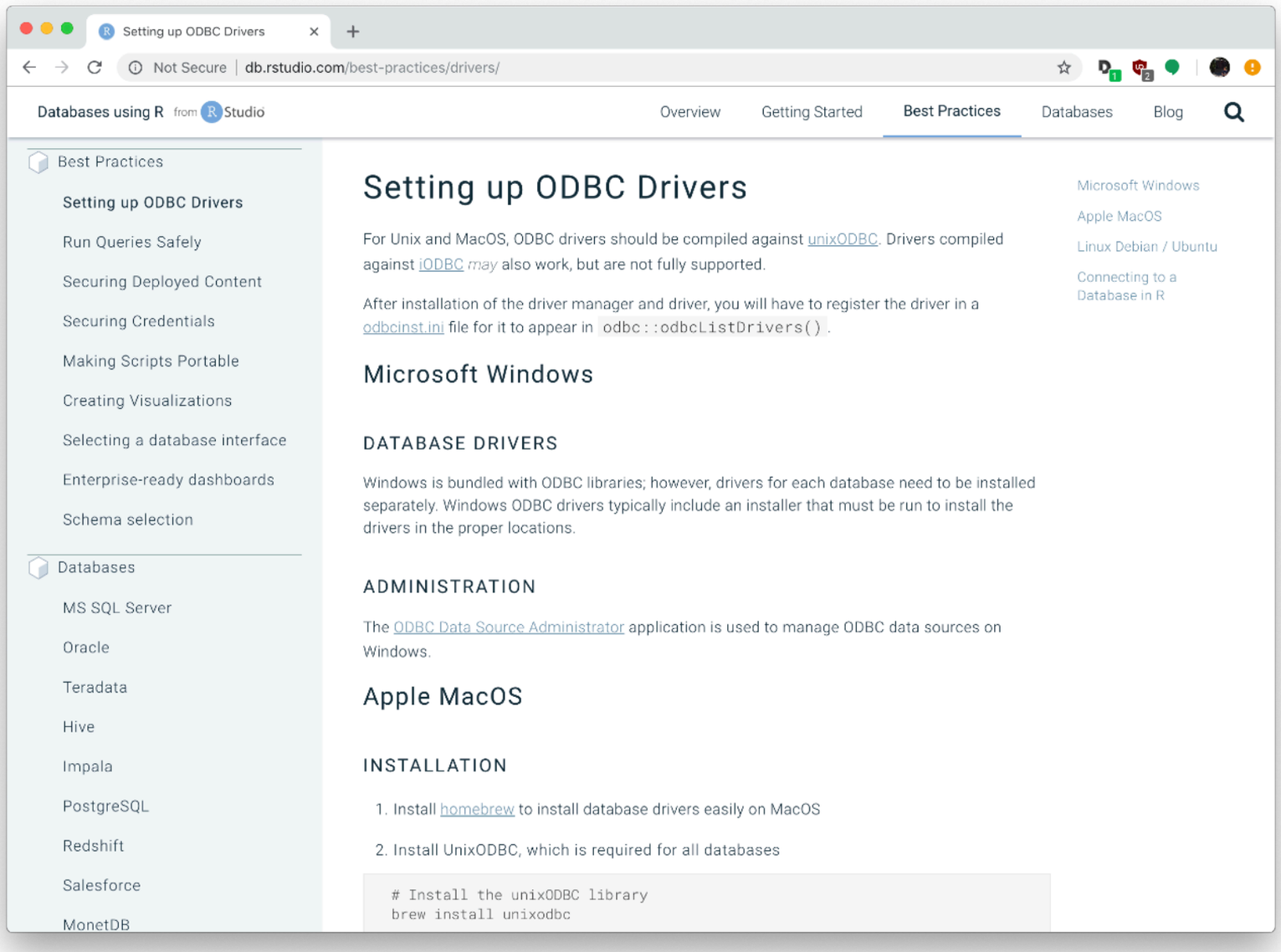
- Each RDBMS requires a software driver that follows the **ODBC** standard to connect with it.
    - e.g. Think of your USB drives…

Install Database Packages

```
# Install the
# Database Interface
# Package that provides a
# common interface to databases
install.packages("DBI")

# Install the
# Open Database Connectivity
# package that provides a
# communication driver layer for
# database interactions.
install.packages("odbc")

# Install DBI-compliant database
# packages
install.packages(
   c("RSQLite", "bigrquery",
      "RMySQL",  "RMariaDB",
       "ROracle",  "RPostgres")
)

# Install a `dplyr` to SQL translator package
install.packages("dbplyr")
```

ODBC Drivers*

… driver communication with a database …



http://db.rstudio.com/best-practices/drivers/

---

* These are pre-installed on RStudio Cloud. To access a database via ODBC, these drivers must be installed on your local computer. Follow the appropriate installation guide given above.

General Connections

```
# Establish a Connection via ODBC
con = DBI::dbConnect(odbc::odbc(),

    # Name of Database (e.g. MySQL, SQLite)
    Driver   = "[your driver's name]",

    # URL to Server Location
    Server   = "[your server's path]",

    # Database Name
    Database = "[your database's name]",

     # Username
    UID      = rstudioapi::askForPassword(
                   "Database user"),
    # Password
    PWD      = rstudioapi::askForPassword(
                   "Database password"),

    # Port to connect on
    Port     = 5432
)
```

---

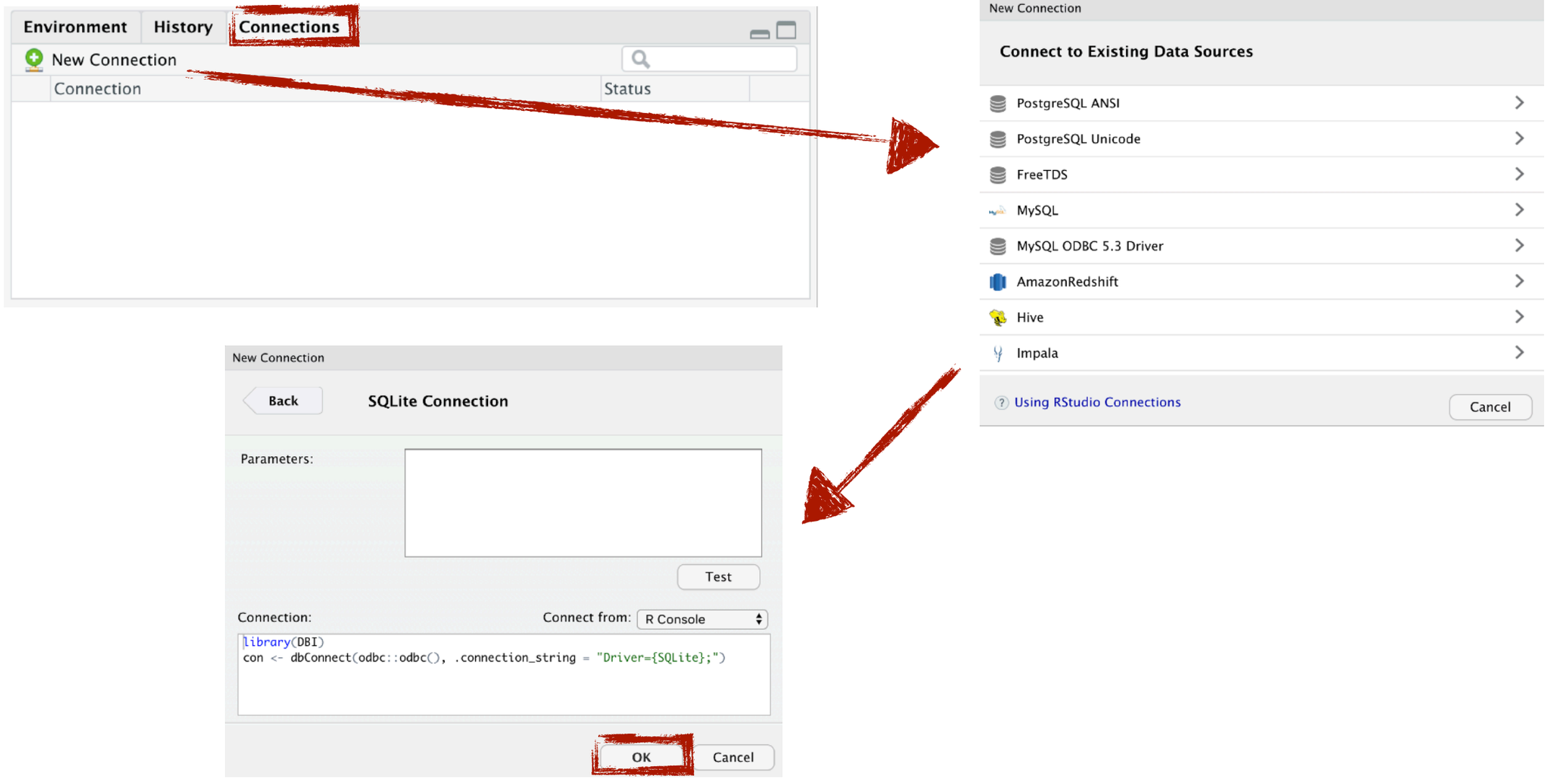
* Why are we prompting for a password for both the username and password?

… embedded database …

```
# Working with a Database Locally

# In memory connection…
con = DBI::dbConnect(
        RSQLite::SQLite(), ":memory:"
)

# File-backed connection…
con = DBI::dbConnect(
        RSQLite::SQLite(),
        "my_db.sqlite"
)
```

Connections Panel*

… obtaining access to a database …

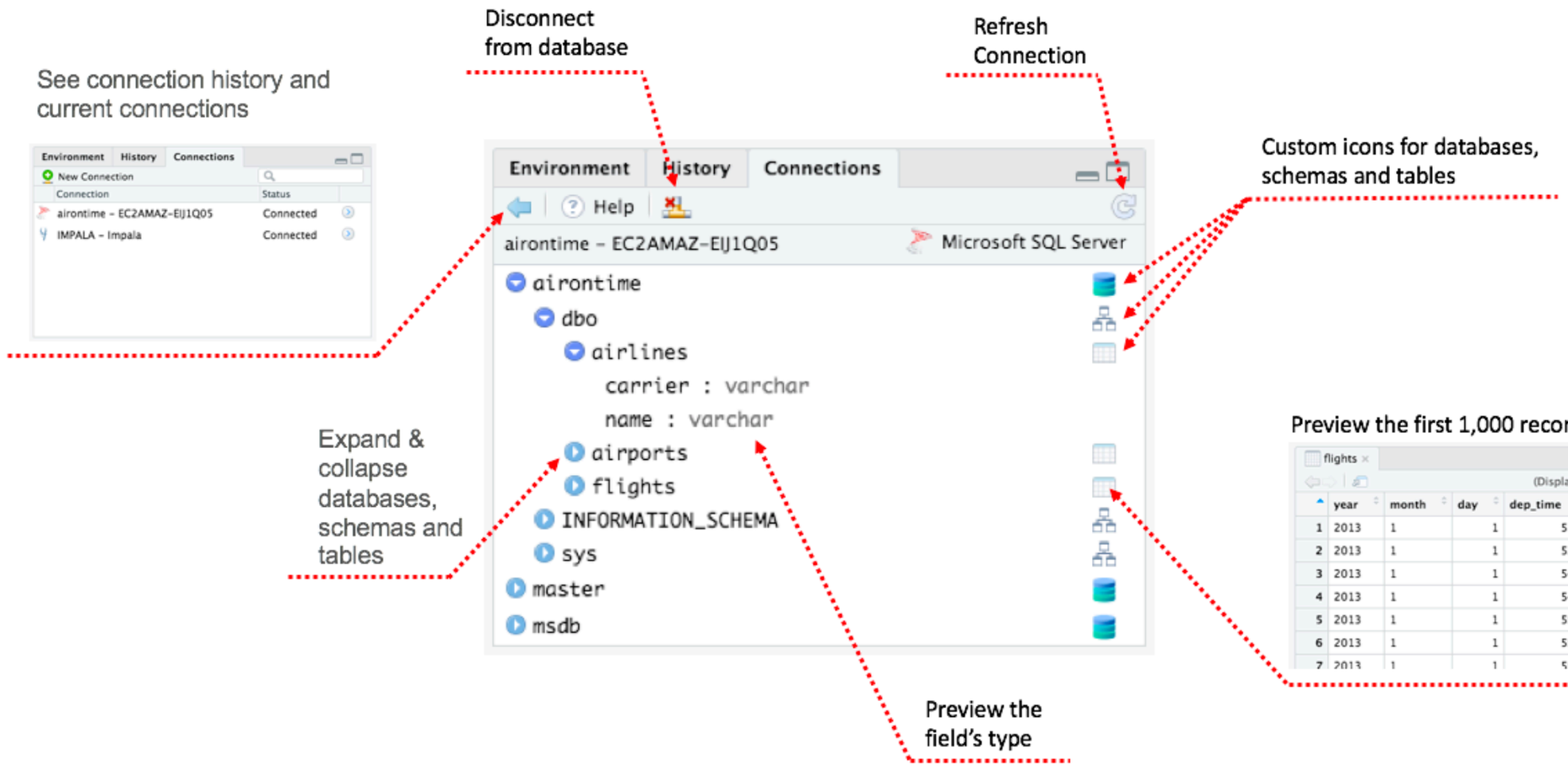


---

* Requires RStudio v1.1 or greater. Lots of great SQL improvements in v1.2 (in preview).
Note: SQLite odbc driver is **NOT** currently available on RStudio Cloud.

Overview of Panel

… meaning of each icon …



https://db.rstudio.com/rstudio/connections/

---

* Requires RStudio v1.1 or greater. Lots of great SQL improvements in v1.2 (in preview).
Note: This is an example of a filled Connections Panel.

Structured Query Language (SQL)

Definition:
Structured Query Language (SQL) refers to a declarative approach for manipulating data held inside of an *RDBMS*.

SELECT * FROM things

Definition:
Declarative Languages refers to a paradigm where the objective of what should be done is given without specifying how to obtain the results.

e.g. "May I please have a table for one?"

Definition:
Imperative Languages refers to a paradigm where you explicitly state the step by step instructions to obtain the desired result.

e.g. "I see an empty table at a restaurant. I walk over to the table. I sit down."

Previously

Table to Data Frame

… database logic vs *R's* data structures …

Students

Table
(data.frame)

Field (Column)

	id	firstname	lastname	age	instate
Record (Row)	1	Billy	Joe	23	FALSE
	2	Theodore	Squirrel	25	TRUE
	3	Keeya	Nod	21	TRUE
	Integer	Character	Character	Integer	Logical

Table Scheme
(Data Types)

SQLite Data Types

- **NULL:** The value is a **NULL** value.
- **INTEGER:** The value is a signed integer.
  - e.g. -8, -4, 0, 1, 7, 25, …
- **REAL:** The value is a floating point value
  - e.g. -10.5, -2.2, 0, 3.8, 4, 5.9, 18.7, …
- **TEXT:** The value is a text string
  - e.g. NetID, Username, FirstName, LastName, …
- **BLOB:** The value is a blob of data
  - e.g. Essays, Books, Text Messages, …

https://www.sqlite.org/datatype3.html

---

* Each database has its own way of specifying a data type. For type information, please see the official documentation for the database. (MySQL, PostgresSQL)

Creating Tables

… writing the schema …

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Courses

course_id	acronym
STAT385	SPM
STAT432	BSL
HIST100	GH

Grades

student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

```
-- Here we are writing the schema for each table
-- and specifying the relationship.
-- Note the change in comment style from # to --

CREATE TABLE Students (
  id INTEGER,
  firstname TEXT,
  lastname TEXT,
  age REAL,
  instate INTEGER,
  PRIMARY KEY (id)
);

CREATE TABLE Courses (
  course_id INTEGER,
  acronym TEXT,
  PRIMARY KEY (course_id)
);

CREATE TABLE Grades (
  student_id INTEGER,
  course_id TEXT,
  grade TEXT,
  FOREIGN KEY (student_id) REFERENCES Students(id),
  FOREIGN KEY (course_id) REFERENCES
Courses(course_id),
  PRIMARY KEY (student_id, course_id)
);
```

Specify **Field Name** and **Data Type**

Denotes **Primary Key**

Denote **Foreign Key**

Semicolon marks the end of a command

R with SQL

… augmented features of *R Markdown* …

Load Database

```{r connect-db}
library(DBI)
db = dbConnect(RSQLite::SQLite(), dbname = "my_db.sqlite")
```

Query Database

```{sql query-db, connection = db}
CREATE TABLE Courses (
  course_id INTEGER,
  acronym TEXT,
  PRIMARY KEY (course_id)
);
```

---

* By using R Markdown's SQL engine feature, we can directly embed SQL code in the the *R Markdown* file and specify the output variable (e.g. **output.var**) that should hold the returned data.frame in *R*.

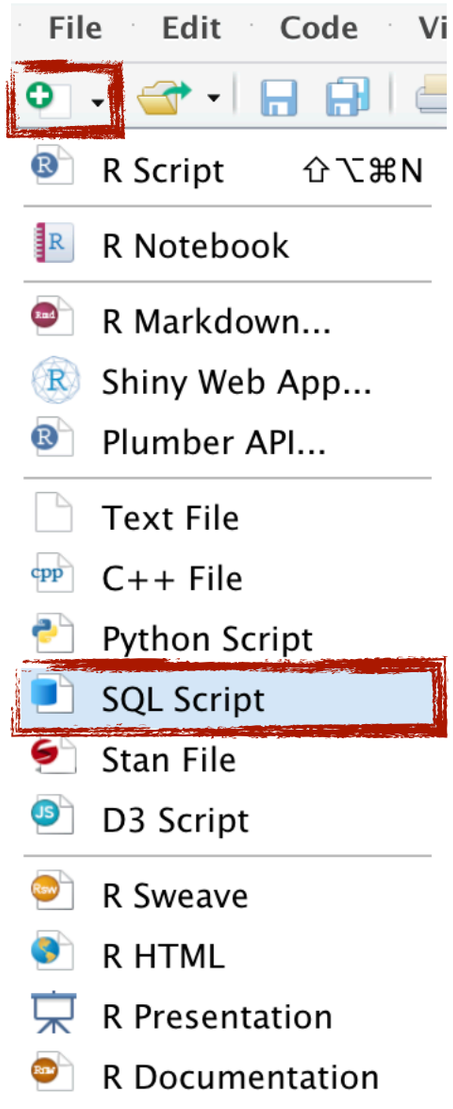
```r
# Verify data importation
# Done in R

# Check all tables were created.
DBI::dbListTables(db)
# [1] "Courses"  "Grades"   "Students"

# The variable names for the Grades Table.
DBI::dbListFields(db, "Grades")
# [1] "student_id" "course_id"  "grade"

# Retrieve a table - Note: It's empty!
DBI::dbReadTable(db, "Grades")
# data frame with 3 columns and 0 rows
```

Verifying Schema

… checking that structure is present …

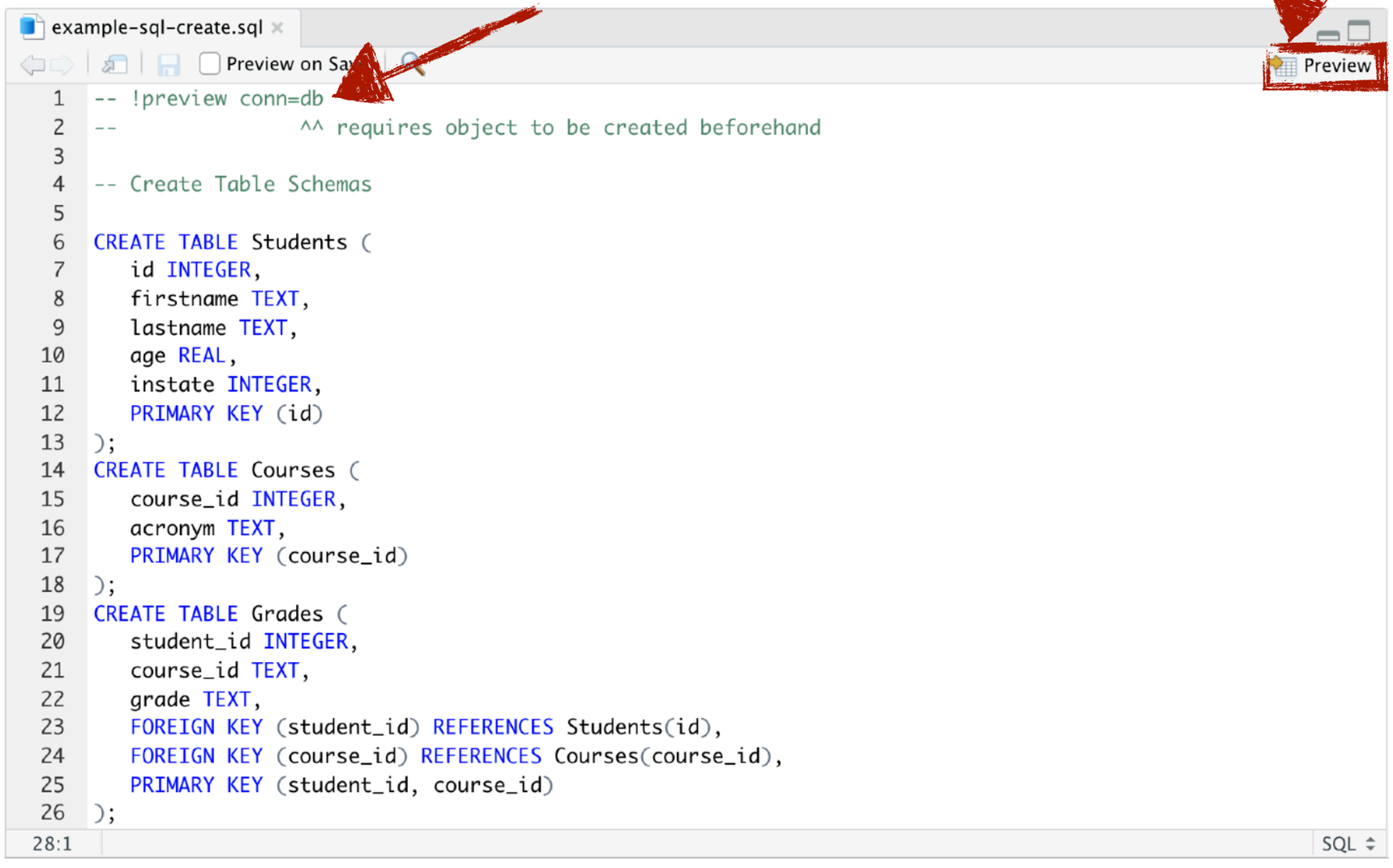
SQL in RStudio

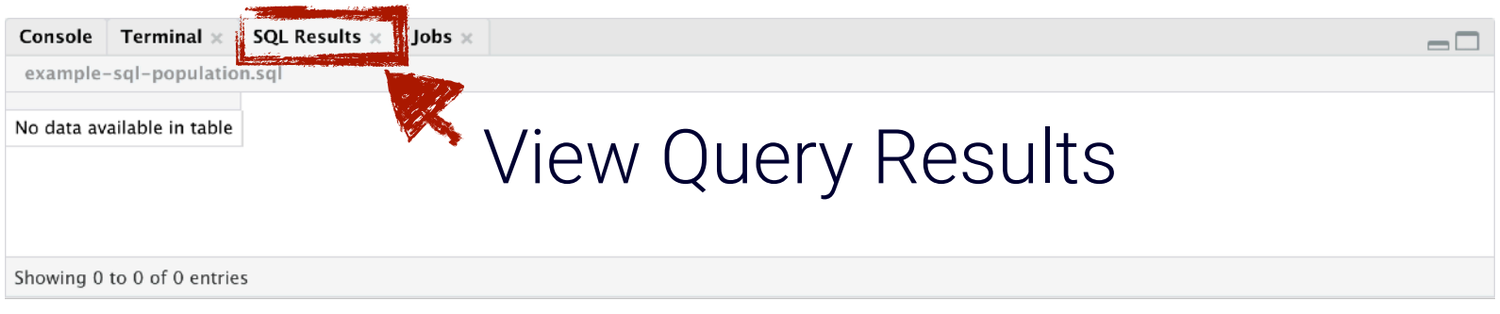
… new sql editing in v1.2 …



Specify Connection

Run Query





View Query Results

Executing a Command

… using DBI to execute a command …

```
# In memory connection...
db = DBI::dbConnect(RSQLite::SQLite(), ":memory:")

# Create a single table
DBI::dbExecute(db,
"CREATE TABLE Students (
   id INTEGER,
   firstname TEXT,
   lastname TEXT,
   age REAL,
   instate INTEGER,
   PRIMARY KEY (id)
);")
# [1] 0

DBI::dbListTables(con)
# [1] "Students"
```

Populate Table with Values

… inserting records …

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Courses

course_id	acronym
STAT385	SPM
STAT432	BSL
HIST100	GH

Grades

student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

```
-- Insert
-- Single-table insertion of data.
```

Table to insert into

```
INSERT INTO Students VALUES
    (1, "Billy", "Joe", 23, 0),
    (2, "Theodore", "Squirrel", 25, 1),
    (3, "Keeya", "Nod", 21, 0);
```

Values for each record

```
INSERT INTO Courses VALUES
    ("STAT385", "SPM"),
    ("STAT432", "BSL"),
    ("HIST100", "GH");
```

Comma between records

```
INSERT INTO Grades VALUES
    (1, "STAT385", "A+"),
    (2, "STAT432", "A-"),
    (1, "HIST100", "A"),
    (3, "STAT385", "B+");
```

Semicolon marks the end of a command

Modifying Records

… changing a value …

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Students

id	firstname	lastname	age	instate
1	James	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Grades

student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	CS374	A
3	HIST101	C-

Grades

student_id	course_id	grade
1	STAT385	A+
2	STAT432	B
1	CS374	A
3	HIST101	B

-- **Update**

-- Single-table value change.

```
UPDATE Students
SET firstname = "James"
WHERE id = 1;
```

SET specifies what the new value should be

WHERE provides a way to search for values

```
UPDATE Grades
SET grade = "B"
WHERE id > 1;
```

Semicolon marks the end of a command

Your Turn

Change the ages of anyone greater than 22 to 18 in the Students table.

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

```
-- Fill me in!
UPDATE _____
SET _____ = _____
WHERE _____ __ _____;
```

Deleting a Record

… removing a row …

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
3	Keeya	Nod	21	TRUE

-- **DELETE**
-- Single-table record removal.

-- Be very careful when
-- removing records.

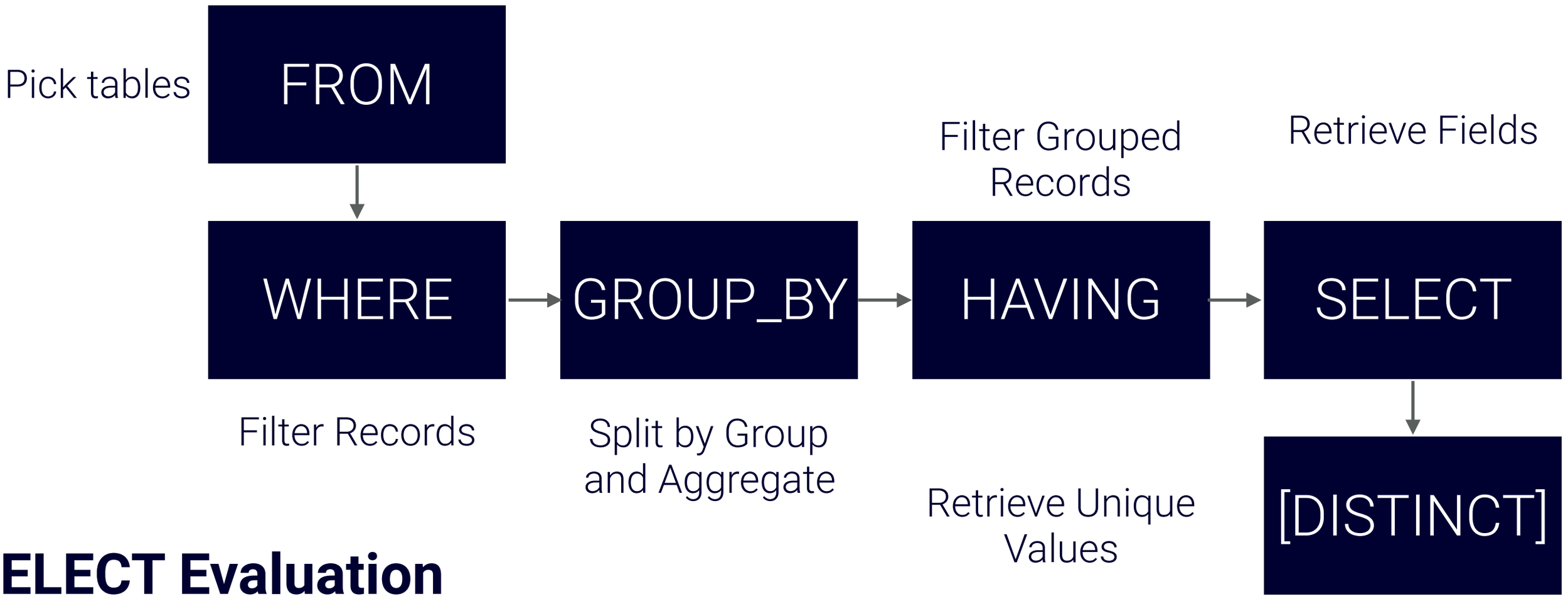
```
DELETE FROM Students
WHERE id = 2;
```

Semicolon marks the end of a command

WHERE provides a way to search for values

-- **Select**
-- Retrieval of data from a table.

```
SELECT columns or calculations
  FROM table
  [WHERE condition]
  [GROUP BY columns]
  [HAVING condition]
  [ORDER BY column <ASC | DESC> ]
  [LIMIT offset, count];

-- Statements inside of [ ] are optional.
```



SELECT Evaluation

Finding Data

… searching records …

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Students

firstname	lastname
Billy	Joe
Theodore	Squirrel
Keeya	Nod

MeanAge
23

```
-- SELECT
-- Single-table selection of fields.

-- Retrieve all fields from the table
SELECT
    *
FROM
    Students;

-- Retrieve only the firstname
-- and lastname from the table
SELECT
    firstname, lastname
FROM
    Students;

-- Compute a mean of variable
SELECT
    AVG(age) as MeanAge
FROM
    Students;
```

Finding Data

… searching records …

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Students

firstname	lastname
Billy	Joe
Theodore	Squirrel
Keeya	Nod

MeanAge
23

Equivalents in R to SQL Queries

```
# Typing name of data.frame
Students

# Retrieve variables
Students[,
  c("firstname", "lastname")
]

# dplyr way
Students %>%
  select(firstname, lastname)

# Create a summary statistic
MeanAge = mean(Students$age)

# And with dplyr...
Students %>%
  summarise(MeanAge = mean(age))
```

Finding Data By Type

… searching records by type …

Students

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE

Students

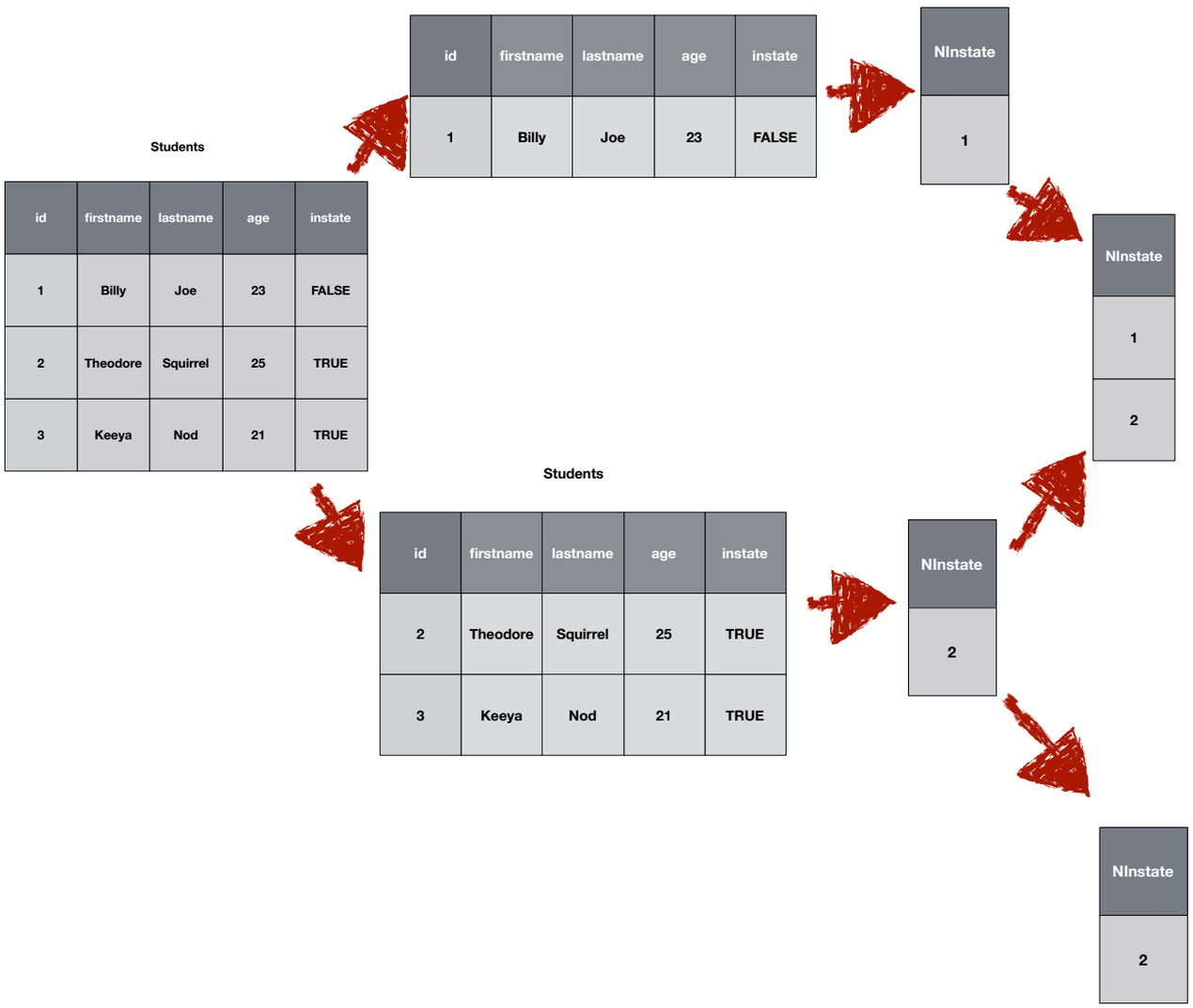
id	firstname	lastname	age	instate
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

NInstate
2

```
-- SELECT with WHERE
-- Single-table selection of fields.

-- Retrieve all records that
-- match a criteria
SELECT
    *
FROM
    Students
WHERE
    instate = 0;

-- Count number of
-- students instate
SELECT
    COUNT(*) as NInstate
FROM
    Students
WHERE
    instate = 1;
```

Group Aggregation

… searching records by type …



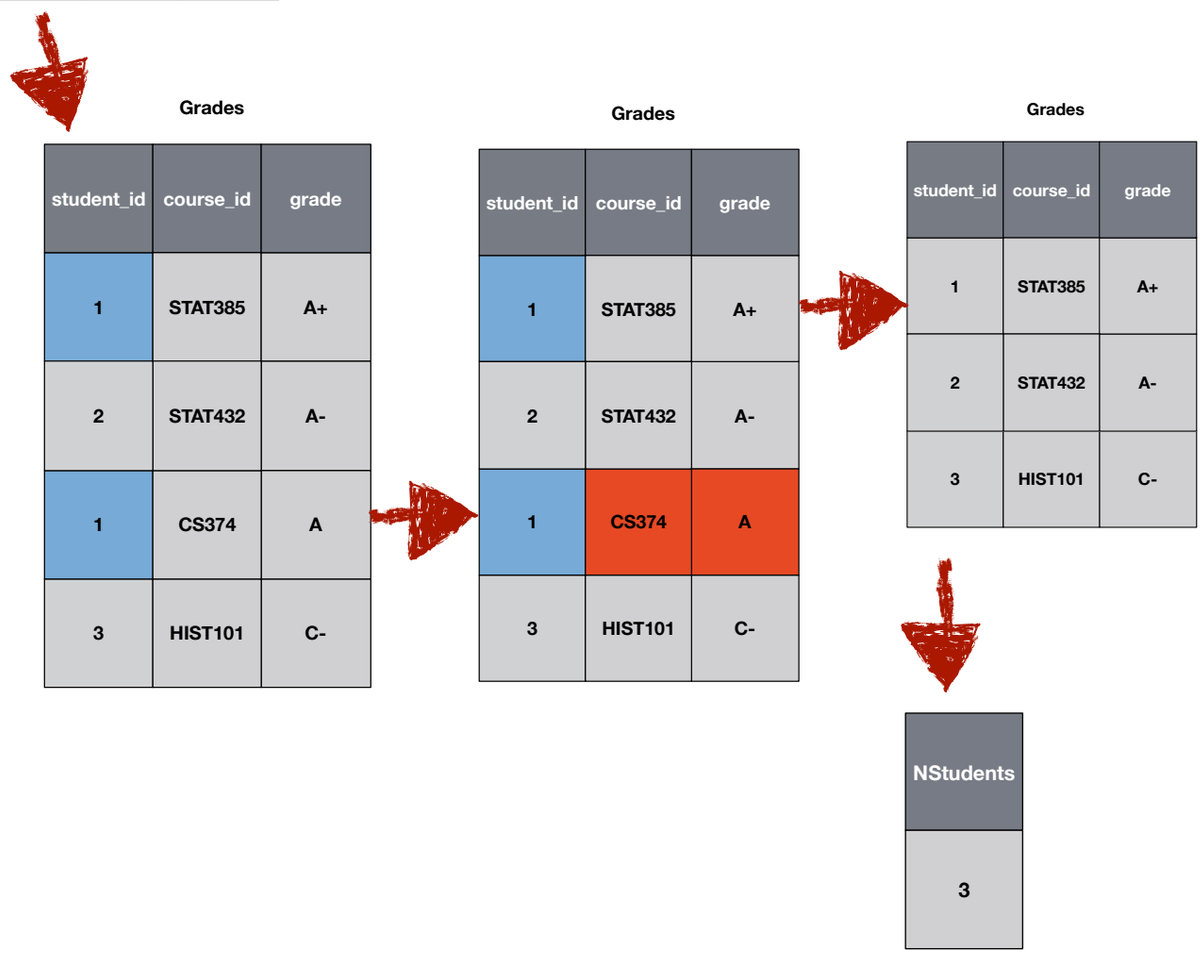

```
-- SELECT with GROUP BY
-- Single-table selection of fields.

SELECT
   COUNT(*) as NInstate
FROM
   Students
GROUP BY
   instate
ORDER BY
   NInstate ASC;


-- Apply filters on group-by result.
SELECT
   COUNT(*) as NInstate
FROM
   Students
GROUP BY
   instate
HAVING NInstate > 1;
```

Distinct Elements

… picking out only unique values …

Grades

student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	CS374	A
3	HIST101	C-

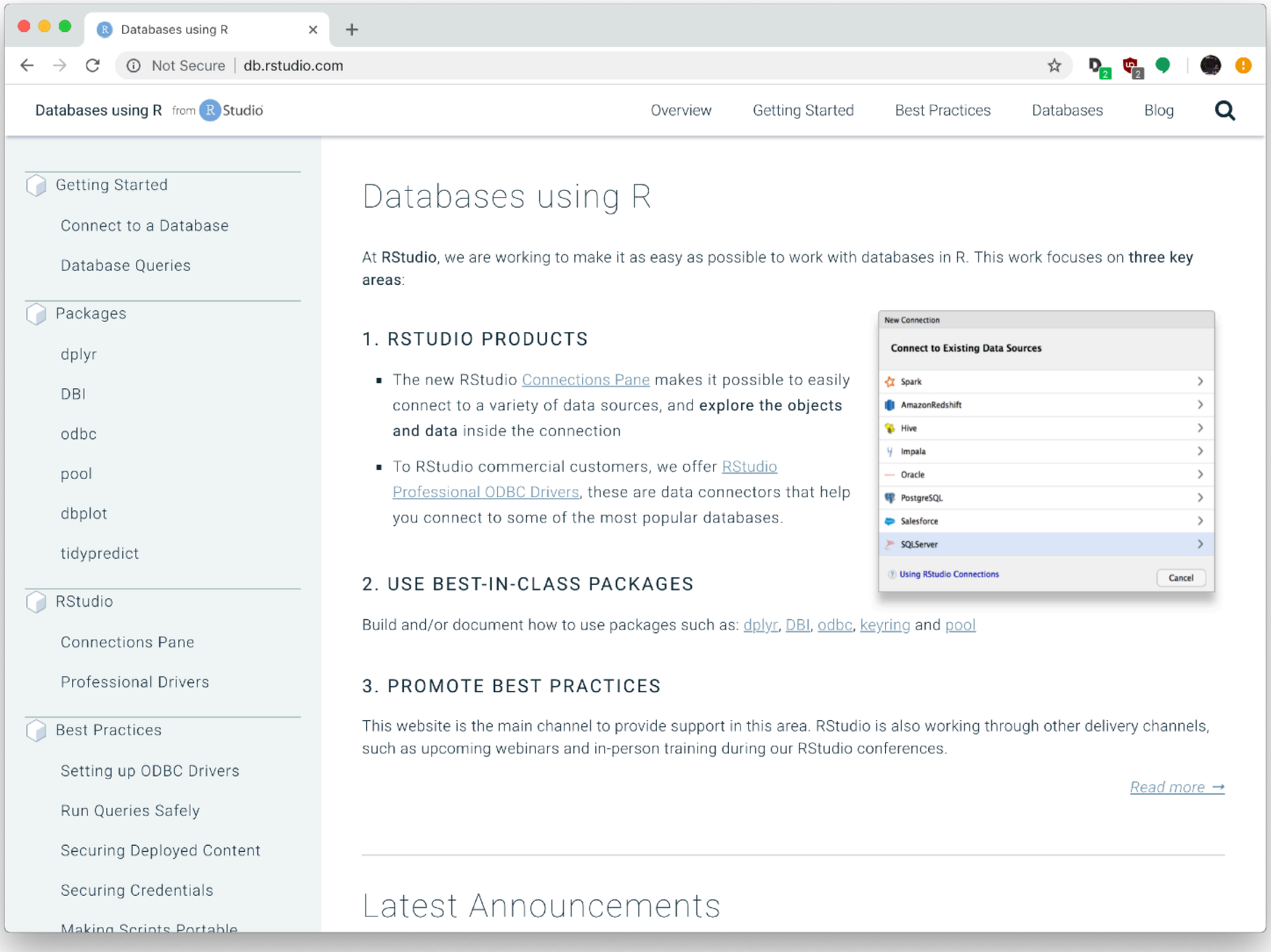



```
-- SELECT with DISTINCT
-- Single-table selection of fields.

SELECT
    COUNT(DISTINCT student_id)
    as NStudents
FROM
    Grades
```

Resources

Databases in R

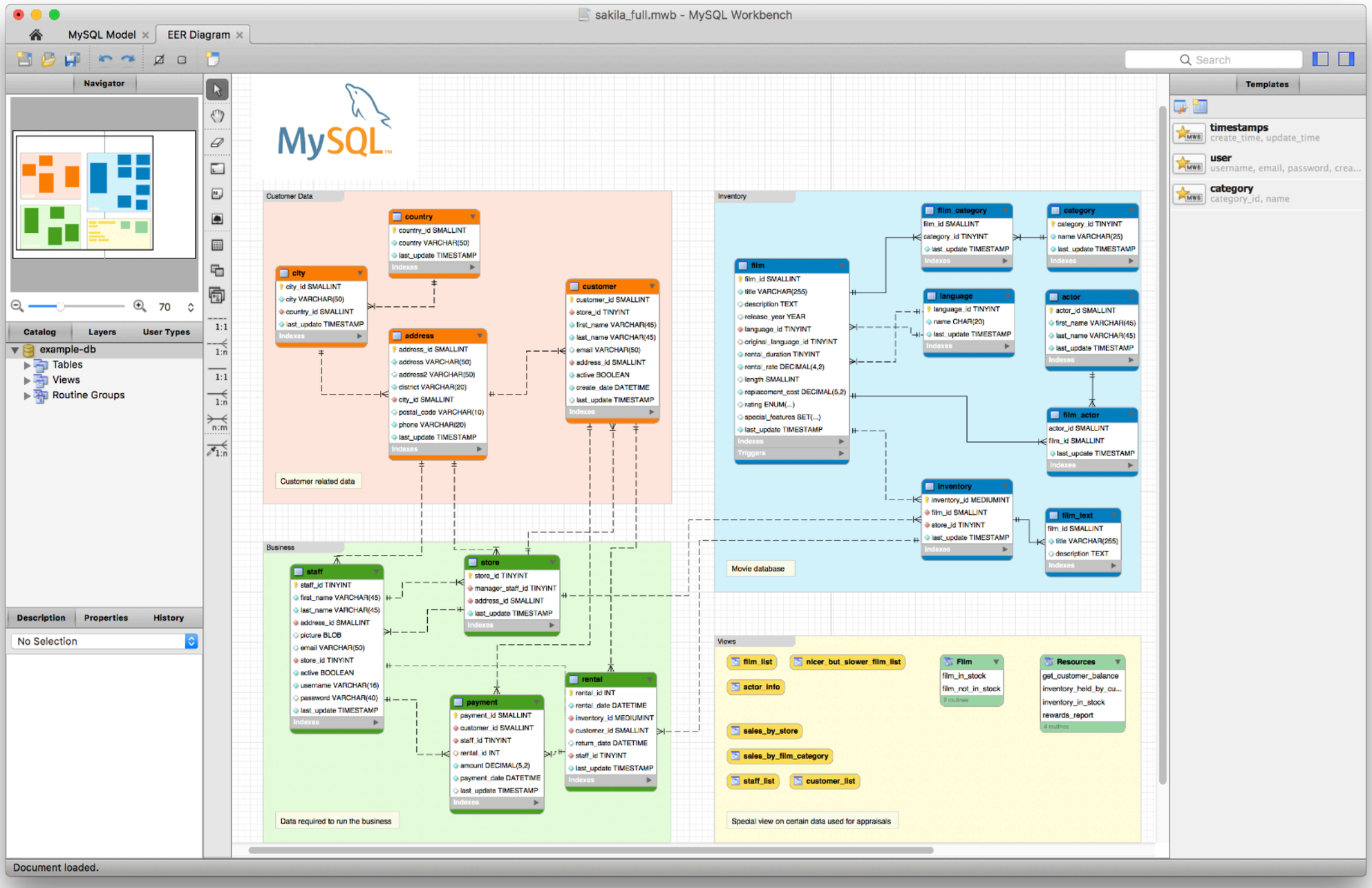
… an overview of drivers, querying, and more !!!



https://db.rstudio.com/

MySQL Workbench

… GUI for Designing Databases …



https://www.mysql.com/products/workbench/

Recap

- **Connecting to a Database**
  - Interactively obtaining and updating data
- **Structured Query Language**
  - Declarative domain-specific language that handles data querying, manipulation, access, and definitions.

Acknowledgements

Acknowledgements

- Edgar Ruiz for both db.rstudio.com and the **dbplyr** package.
- Kirill Müller for the **DBI** package.
- Hadley Wickham for the **dbplyr** package.