

The background image is a wide-angle aerial photograph of the San Francisco skyline during sunset. The city's iconic buildings, including the Transamerica Pyramid and the Coit Tower, are visible against a backdrop of rolling hills and a body of water. The sky is a warm, golden color.

# My Organization's First R Package

## 1 – Working with Data in a Database

### Why use databases?

---



Databases can be great. Here are some advantages:

- we know where the data is
  - the data is in one place (in that, this is the source of truth)
  - many people can have simultaneous access to the same data
  - database tables can be updated with new data, with minimal disruption
  - are there others? let me know!
-

### Why use databases?

---



Databases can be *annoying* too. Here are some bad things I've experienced:

- queries can get stuck in the queue sometimes
  - tables can be accidentally deleted or badly altered
  - availability may only be through certain IP addresses
  - some **dplyr** statements may not work at all (no SQL equivalent)
  - others? (speak up those with bad experiences)
-

## WORKING WITH DATA IN A DATABASE

# Which databases are available and (easily) accessible by R?



	MySQL MariaDB	Postgres Redshift	SQLite
R Package	RMariaDB	RPostgres	RSQLite
Website	<a href="http://rmariadb.r-dbi.org">rmariadb.r-dbi.org</a>	<a href="http://rpostgres.r-dbi.org">rpostgres.r-dbi.org</a>	<a href="http://r-dbi.github.io/RSQLite">r-dbi.github.io/RSQLite</a>
Libraries Required			
Mac	<a href="#">mysql-connector-c++</a>	<a href="#">libpq</a>	—
Debian, Ubuntu	<a href="#">libmysqlclient-dev</a> <a href="#">Libmariadb-client-lgpl-dev</a>	<a href="#">libpq-dev</a>	—
Fedora, CentOS, RHEL	<a href="#">mysql-devel</a> <a href="#">mariadb-devel</a>	<a href="#">postgresql-devel</a>	—

Mac: `brew install <lib>` // Debian/Ubuntu: `sudo apt-get install -y <lib>` // Fedora et al.: `sudo yum install <lib>`

### Which packages are required?

---



For our workshop, we'll need the RMariaDB package to access the MySQL database.

You'll also need to install some system libraries if you're using a Mac or Linux system.

#### Mac

```
brew install mysql-connector-c++
```

#### Debian/Ubuntu

```
sudo apt-get install -y libmysqlclient-dev  
sudo apt-get install -y Libmariadb-client-lgpl-dev
```

#### Fedora et al.

```
sudo yum install mysql-devel  
sudo yum install mariadb-devel
```

---

# Which packages are required?

---



We need the **DBI** package. It can be thought of as the central package for connecting to databases in R.

We also need packages from the *Tidyverse*. In particular, **dplyr** and **dbplyr** are essential for working with databases.

R Package	DBI	dplyr	dbplyr
Website	<a href="http://dbi.r-dbi.org">dbi.r-dbi.org</a>	<a href="http://dplyr.tidyverse.org">dplyr.tidyverse.org</a>	<a href="http://dbplyr.tidyverse.org">dbplyr.tidyverse.org</a>

## WORKING WITH DATA IN A DATABASE

CODE DEMO: `01_01_databases.Rmd`

---



We are going to move over to the `.Rmd` file named:

`exercises/01_01_databases.Rmd`

In that, we will create a simple **SQLite** database, create a table with `nycflights13 :: flights` data, and use `dplyr` to work with the in-memory DB table.

---

## WORKING WITH DATA IN A DATABASE

R

### Introducing our database and our data tables.

Just for today, we all work for a company called *Intendo*.



They have a database with data available for the **2015** year.  
It has all sorts of information for their only game: *Super Jetroid*.



# WORKING WITH DATA IN A DATABASE

## Introducing our database and our data tables.



The database is hosted on AWS using RDS (Relational Database Service).

It's a MySQL database with 3 tables containing all of their user data and revenue data.

This database is publicly accessible. In most organizations, however, access is only possible when at specific IP addresses.

The screenshot shows the MySQL Workbench interface for the 'intendo' database. The 'daily\_users' table is selected in the 'STRUCTURE' tab. The table has 15 columns:

Field	Type	Length	Unsigned	Zerofill	Binary	Allow Null	Key	Default	Extra	Encoding	Collation	Comment
user_id	VARCHAR	16				✓	M...	None	cp1252	latin1_swedi		
session_id	VARCHAR	16				✓	M...	None	cp1252	latin1_swedi		
time	DATETIME					✓	M... NULL	None				
total_sessions	BIGINT	20				✓	M... NULL	None				
total_time	DOUBLE					✓	M... NULL	None				
level_reached	BIGINT	20				✓	M... NULL	None				
at_eoc	TINYINT	1				✓	M... NULL	None				
in_ftue	TINYINT	1				✓	M... NULL	None				
is_customer	TINYINT	1				✓	M... NULL	None				
iap_revenue	DOUBLE					✓	M... NULL	None				
ad_revenue	DOUBLE					✓	M... NULL	None				
total_revenue	DOUBLE					✓	M... NULL	None				
iap_count	BIGINT	20				✓	M... NULL	None				
ad_count	BIGINT	20				✓	M... NULL	None				

Below the table structure, the 'INDEXES' section shows 8 indexes:

Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Comment
1	user_id	1	user_id	A	742389	NULL	NULL	
1	session_id	1	session_id	A	1428330	NULL	NULL	
1	time	1	time	A	1067366	NULL	NULL	
1	total_sessions	1	total_sessions	A	18	NULL	NULL	
1	total_time	1	total_time	A	643	NULL	NULL	
1	level_reached	1	level_reached	A	10	NULL	NULL	
1	at_eoc	1	at_eoc	A	1	NULL	NULL	
1	in_ftue	1	in_ftue	A	1	NULL	NULL	



There are three tables: `daily_users`, `revenue`, and `users`.

---

### `daily_users`

- updated daily, has a record per user, per session (could be multiple logins per day)
- each record contains info on user's past activity and revenue totals

---

### TABLE COLUMNS

---

<code>user_id</code>	<code>total_sessions</code>	<code>level_reached</code>	<code>is_customer</code>	
<code>session_id</code>	<code>total_time</code>	<code>at_eoc</code>	<code>iap_revenue</code>	<code>iap_count</code>
<code>time</code>		<code>in_ftue</code>	<code>ad_revenue</code>	<code>ad_count</code>
IDs for the user and for the session; login date-time.	summary totals on number of sessions and total time active (in minutes).	Progression info: What level? At the end of content? In the first-time user experience?	<code>total_revenue</code> Has user spent <i>any</i> money up to now? USD amounts for IAPs, ad views, and sum of both.	Count of IAPs bought, count of ads viewed.



There are three tables: `daily_users`, `revenue`, and `users`.

---

### revenue

- updated daily, has a record IAP bought or ad viewed by a user
- contains specifics on the name of IAP, ad type, and price for IAP, and actual revenue

---

### TABLE COLUMNS

user_id	name	type	price	revenue
session_id	size			
user_id	name	type	price	revenue
session_id	size			
time				
IDs for the user and for the session; login date-time.	Name of the IAP or ad type. The <code>size</code> is relative worth of the IAP.	The type of IAP or "ad" for advertising revenue. IAP types are: "currency", "season_pass", or "offer_agent".	The price in USD for the IAP ( <code>NULL</code> if type is "ad").	The actual revenue received for the ad view or IAP purchase.

## WORKING WITH DATA IN A DATABASE

### Introducing our database and our data tables.



There are three tables: `daily_users`, `revenue`, and `users`.

#### users

- updated daily, a user summary table with one row per user
- contains specifics on user first login, revenue earned, primary device, and country

#### TABLE COLUMNS

user_id	iap_count	customer	platform	acquired
first_login	iap_revenue	subscriber	device	country
ID for the user and first login date-time.	ad_count	first_iap	What was the login device and its platform?	How was the user acquired, through a campaign or "organic"? What country is the user from?
	ad_revenue	Has user ever bought an IAP? Ever bought a "season_pass"?		
	total_revenue	What was the first IAP?		
	Revenue from and counts of IAPs and ad views.			

## WORKING WITH DATA IN A DATABASE

CODE DEMO: 01\_02\_databases.Rmd

---



We are going to move over to the .Rmd file named:

**exercises/01\_02\_databases.Rmd**

We'll learn how to connect to the *Intendo's intendo* database.  
It involves making a *connection* object.

We'll also learn about how to list the database tables.  
That involves using the **DBI :: dbListTables()** function.

---

### YOUR TURN (WORKING ON THE `intendo` PACKAGE)

---



**30 minutes**

**Open up the package-in-progress at:**

`p_01_intendo/01_intendo.Rproj`

---

Take a look around the package and see which components are already in place.

There are no functions yet (No R folder!) but there is a scripts folder with .R files that contain functions. Test them out and then make them part of the package.

**Use some of these `usethis` functions:**

**ESSENTIAL** `use_r()` `use_pipe()` `use_build_ignore()` `use_package()`

**BONUS** `use_package_doc()` `use_news_md()` `use_readme_rmd()` `use_roxygen_md()`

---

Discuss things with your neighbors, talk to the instructors and TAs... we'll help!

---