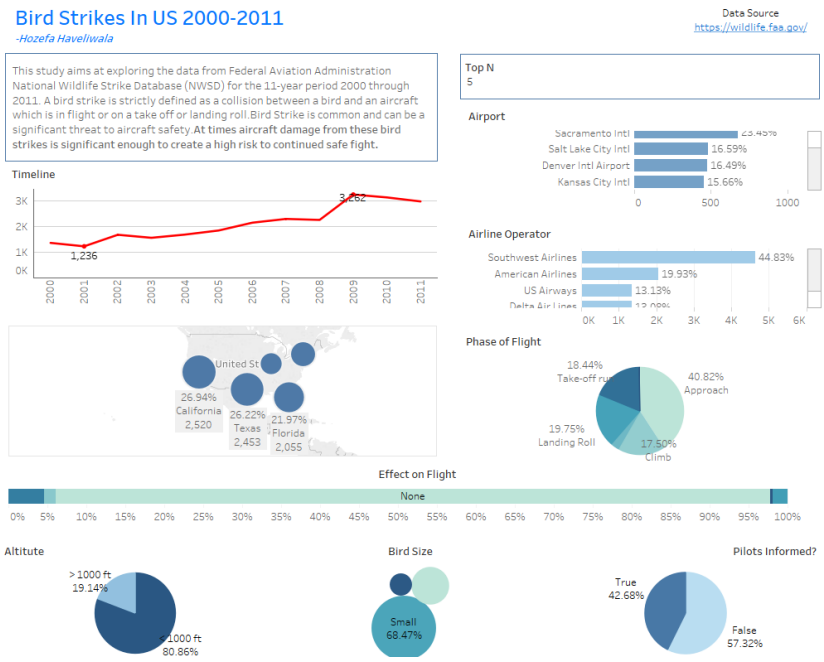
[20 hrs] BUILD: Practicum I / Design & Implement a Relational Database

Overview

In this practicum you will build a database that can be used to analyze bird strikes on aircraft. For an existing data set from the FAA [1], you will build a logical data model, a relational schema, realize the relational schema in MySQL/MariaDB, load data into the database, execute SQL queries, a finally perform some simple analysis of the data.

The graphic below shows some statistics regarding bird strikes and helps frame the data in the data file.



Use the provided time estimates for each tasks to time-box your time. Seek assistance if you spend more time than specified on a task -- you are likely not solving the problem correctly. A key objective is to learn how to look things up, how to navigate complex problems, and how to identify and resolve programming errors.

Learning Objectives

In this practicum you will learn how to:

- ☐ install/procure MySQL or MariaDB
- ☐ connect to MySQL/MariaDB from R in an R Notebook
- ☐ implement a relational schema for an existing data set
- ☐ load data from CSV files into a relational database through R
- ☐ execute SQL queries against a MySQL/MariaDB database through R
- ☐ perform simple analytics in R
- ☐ identify and resolve programming errors
- ☐ look up details for R, SQL, and MySQL/MariaDB
- ☐ time-box work

Format

- Complete in pairs or individually; working in pairs is not required. We do not pair up students; finding a partner is the student's responsibility.
- Add both team members as authors to the submission in the R Notebook and in the submission comments, but only one person needs to make a submission. Both team members will receive the same grade.
- If a team member is not contributing, the other team member may simply quit and continue individually using all work done up to that point. It is expected that both team members do an equal amount of work of similar complexity. You must write a submission comment explaining who you worked with, when you started working together, when you split, and why.
- A pair = two members in a team, that means two and no more than two; working in a group of three is not a pair and is not allowed.

Prerequisite Tasks

1. Read the **Hints and Tips** section below and go back to them often when you encounter problem. Most problems are addressed in that section. Consult the list before contacting us for help as you'll be able to resolve the issue more quickly.
2. Decide on whether you will use MySQL (or MariaDB) locally or whether you will use a cloud installation: READ FIRST: Which Database to Use? If you use *db4free* then you cannot use *dbWriteTable()* and need to insert data row by row. See the tutorials above.
3. Create an R Project and within that project create an R Notebook in which to do all of your work. Place all files within the project folder and do not use absolute file paths. Embed any diagrams in the R Notebook and be sure to submit them as files in your submission. Keep updating this notebook as you complete tasks. Learn how to use journaling in data analysis work. Use this R Markdown cheat sheet for reference. Of course, in practice, you will likely build different programs (perhaps even using different programming language) for database creation, data loading, and the dashboard. But here we will do it all in one R Notebook.
4. Become familiar with the problem of bird strikes on aircraft. The report in [1] provides an overview. You should scan it to become familiar with the domain but you do not need to read it deeply.
5. Download the file BirdStrikesData.csv (Links to an external site.) and save it locally to your R Project folder (the same folder that contains you *.Rmd* and your *.Rproj* files) or reference it from the URL. You may wish to create a new data file that is a subset of the full data that you use for development so loading takes less time. This is a common strategy in practice. To download the file, use the right mouse button and choose "*Save Link As...*" or a similar menu choice on your browser. Do not click on the link as that will cause the browser to open and display the file.

Tasks

Before you start, read all of the questions. Inspect the CSV data file that you downloaded so you are familiar with its columns, data types, and overall structure. Assume that this database will eventually be used for an app that can be used by pilots (of any kind of aircraft) to report wildlife incidents.

All R and SQL code blocks must be named, as shown in the example below. This is necessary so that you can reference your code blocks in the self-evaluation rubric to be filled out at the end of the practicum. The names of code blocks must be unique.

```{r *nameOfRCodeBlock*, eval = T, warning = F}

```{sql *nameOfSQLCodeBlock*, connection = xDB}
```

You may add any additional block parameters as needed.

Use functions to structure your code so that it becomes more readable and easier to develop and debug. Use headers to segment your notebook and add explanations as to what each code block means. Follow common coding practices and format your code so it is readable, and use functions to break down complex code.

1. (5 *pts / 2.0 hrs*) Set up, configure a MySQL Server locally or on the cloud using the resources provided here and on Canvas. Once configured, create a new MySQL database. Note that you may use SQLite instead of MySQL but you will not get credit for this question nor credit for the question below that asks you to create a stored procedure as those are not supported in SQLite.

2. (1 *pts / 0.1 hrs*) Create an R Notebook named "*LastNameFirstInitial*.CS5200.PractI-S23.Rmd" where *LastName* is your last name and *FirstInitial* is the first letter of your first name, *e.g.*, *SchedlbauerM*. Set the title of the R Notebook to "Practicum I CS5200", add an author field, and set the date to "Spring 2023".

3. (4 *pts / 0.1 hrs*) Add a second level header (##) to the R Notebook called "Connect to Database" and then add an R code chunk that connects to your MySQL database. Use headers for all other questions with appropriate titles so you (and we) can navigate the notebook more easily. If you have difficulty connecting to or setting up MySQL, then use SQLite and proceed. You can always come back to this question and change your configuration so that you connect to MySQL. This is the benefit of relational databases: you can easily switch between databases without changing your code.

4. (30 *pts / 3.5 hrs*) Add a second level header to the R Notebook called "Create Database" and then add appropriate R and/or SQL code chunks to create the database schema described below. Add appropriate constraints, primary key and foreign key definitions. In the schema definitions below, primary keys are underlined and foreign keys are bolded.

    A. (5 *pts / 0.5 hrs*) Create a table *incidents* that stores wildlife strike incidents with this schema:

    incidents (<u>rid : integer,</u>
    dep.date : date, ***origin*** *: integer,* ***airline*** *: integer,*
    aircraft : text, flight.phase : {takeoff, landing, inflight, unknown},
    altitude : integer ≥ 0, conditions : {...},
    warned: boolean)

    The column *altitude* should be restricted to positive integers. The column *flightphase* should be restricted to those values (use a value set definition and not a lookup table). For now, make the column *conditions* of type 'text' or 'varchar'; later we will refine that to using a lookup table. Make *warned* a Boolean flag and use TRUE if the pilot was warned, FALSE otherwise. Use appropriate data types for the columns and store any date as a date type not as text (subject to the data types your chosen database supports). If date or boolean are not supported, choose another data type that will work or split the dates into month, day, and year columns. Note that some columns contain periods so that will require special treatment in SQL -- investigate how to deal with this common issue.
```

B. (5 *pts / 0.5 hrs*) Create a table that stores airports and states called *airports* and that follows this schema:
airports (aid : integer, airportName : text, airportCode : text, state : text)

aid is a synthetic primary key, *airportName* and *state* correspond to the airport name and state from the data file. The *airport* code should be the airport's international code, *e.g.*, BOS for Boston or LGA for LaGuardia. However, you may leave it empty for this database -- it is for future expansion.

C. (4 *pts / 0.3 hrs*) Link the *incidents* and *airports* tables via the *origin* foreign key in *incidents* to the primary key *aid* in *airports*. The *origin* is an FK to the airport in the *airports* table. Update the above table definitions for *airports* and *incidents* as necessary.

D. (4 *pts / 0.5 hrs*) Create a lookup table called conditions defined as follows:
conditions (cid, condition, explanation)

Link this lookup table to the *incidents* table through the *conditions* foreign key. This table contains the values of all conditions, *e.g.*, 'Overcast'. Leave the *explanation* column empty (future expansion).

E. (4 *pts / 0.5 hrs*) Create a table that stores airlines called *airlines* and that follows this schema:
airlines (eid : integer, airlineName : text, airlineCode : text, flag : text)

eid is a synthetic primary key, *airlineName* corresponds to the airline column from the data file. The *airlineCode* should be the airlinet's abbreviation code, *e.g.*, J6 for JetBlue or AA for American Airlines. However, you may leave it empty for this database -- it is for future expansion. The *flag* column is the flag under which this airline flies, *e.g.*, Lufthansa flies under the flag of "Germany". However, you may leave it empty for this database -- it is for future expansion.

F. (3 *pts / 0.3 hrs*) Link the *incidents* and *airlines* tables via the *airline* foreign key in *incidents* to the primary key e*id* in *airlines*.

G. (2 *pts / 0.3 hrs*) Add one or more code chunks that are not evaluated *(eval = F*) when the Notebook is knitted but can be used to test your table definitions. Add whatever test code you need to assure yourself that your table definitions are correct.

5. (1 *pts / 0.1 hrs*) If you haven't yet, download the bird strikes CSV data file from the link provided before. Place the Bird Strikes CSV file into the same folder as your R Notebook and the load it into a dataframe called *bds.raw*. Do not use a path name when loading. The default path is the local folder that contains the R Notebook when you have the R Notebook in an R Project.

6. (20 *pts / 8 hrs*) Using the table definitions and the data in the dataframe *bds.raw* from above, populate the tables with the data from the appropriate columns. Omit the columns from the CSV that are not referenced in the tables. You do not need to create any additional tables. Because we are not adding additional tables there will be (unnormalized) data and repetitions, for example for aircraft -- that is acceptable for this practicum due to time constraints but would not be if this were an actual analytics database project. Assume "Business" to be an airline name (it is actually a private flight but we assume it is the airline called "Business") and store those incidents.

Use default values where the data file does not contain values or leave empty. If there is no airport or airline, then link to a "sentinel" airline or airport, *i.e.*, add an "*unknown*" airline and airport to the tables rather than leaving the value NULL. Assign synthetic key values as and where needed and use them as primary keys. Whether you generate them in R code or the database is up to you -- each has pros and cons and part of the objective of this practicum is for you to think through such decisions.

Map the values in the data file to appropriate values in any lookup tables using reasonable rules that you may define as necessary.

See the **Hints** below for information on *db4free*. **All data manipulation and importing work must occur in R**. **You may not modify the original data outside of R -- that would not be reproducible work**. It may be helpful to create a subset of the data for development and testing as the full file is quite large and takes time to load.

7. (1 *pts / 0.5 hr*) Show that the loading of the data worked by displaying parts of each table (do not show the entire tables).

8. (3 *pts / 1 hr*) Create a SQL query against your database to find the 10 states with the greatest number of incidents. You may either use a *{sql}* code chunk or an R function to execute the query. It must be a single query. Display the state and the number of incidents. Note that every row in the *incidents* table constitutes one "incident".

9. (10 *pts / 1 hr*) Create a SQL query against your database to find the airlines that had an above average number bird strike incidents. You may either use a *{sql}* code chunk or an R function to execute the query. It must be a single query. To do this, find the number of bird strike incidents for each airline (remember that each row in the *incidents* table is a single bird strike incident). Then calculate the average across all airlines and from there find those airlines which had an above average number of bird strike incidents. List the names of the airlines and the number of incidents for each.

10. (8 *pts / 1 hr*) Create a SQL query against your database to find the number of bird strike incidents by month and by flight phase (across all years). Save the result of the query in a dataframe. Include all airlines and all flights. You may either use a *{sql}* code chunk or an R function to execute the query. It must be a single query. This query can help answer the question which month and flight phase, historically, are the most dangerous for bird strikes. Display the first six rows of the dataframe.

11. (5 *pts / 2 hrs*) Using the dataframe from Question 10 above, build a scatter plot that plots month along the x-axis versus number of incidents (across all airlines and flight phases). Adorn the graph with appropriate axis labels, titles, legend, data labels, etc. You should use the standard R `plot()` function; you do not need to use packages such as **ggplot**, **ggplot2,** or **plotly** -- although you may, of course. This tutorialLinks to an external site. may help you get started.

12. (10 *pts / 3 hrs*) Create a stored procedure in MySQL (note that if you used SQLite, then you cannot complete this step) that adds a new incident to the database. You may decide what you need to pass to the stored procedure to add a bird strike incident and you must account for there being potentially a new airport and/or airline. After insertion, show (in R) that your procedure worked. Note that if you used SQLite rather than the required MySQL for the practicum, then you cannot complete this question as SQLite does not support stored procedures.

13. (5 *pts*) Professionally developed code that is well documented and all chunks are labeled.

Resources

[1] Data Visualization - Bird Strike Dataset by H. Haveliwala | data.worldLinks to an external site.
[2] R Markdown Cheat Sheet

[3] Using MySQL and MariaDB with R (jagg19.github.io)Links to an external site.
[4] RMariaDB: MariaDB Driver for R - MariaDB Knowledge BaseLinks to an external site.