# Intro to Databases (for Astronomy)

Noah Franz

# Learning Goals

1. Databases
    a. What they are
    b. When and why to use them
    c. Minimizing data storage
2. SQL
    a. Write basic SELECT statements
    b. Write basic JOIN statements
    c. INSERT data

# Workshop Outline

1. Intro to relational databases (with a brief mention of other types; 10 min)
2. Intro to SQL (10 min)
3. Activity
    a. Get everyone setup (5-10 min)
    b. Interactive Queries Together (15 min)
    c. Activity with your table (45 min)

# Database Basics

# Why use a database over a spreadsheet?

- Efficient data storage for large datasets
  - No dataset redundancy = less data to store
- Easily and efficiently backup dataset
- Very fast querying (much faster than parsing a text file)
- Security Functionality
- Access the dataset anywhere but only store it in one place
- Reinforce dataset structure

# Definitions

| | |
|---|---|
| Database | A collection of documents (or tables) storing data |
| Document | The table storing a collection of data |
| Query | Filtering a database to access specific information |
| Data | A value corresponding to a Column and Row (a “cell” in Excel lingo) |
| Key | An identifier for a row |
| Unique | No other data in that column have the same value |
| Null | An empty data value |
| Primary Key | A UNIQUE and NOT NULL identifier for a row |
| Foreign Key | A pointer to a primary key in another table |
| Join | Merging two tables by matching on a foreign key |

# Contents of a database table

REMINDER: A table is basically just an individual spreadsheet!

1. Columns and Rows
2. Column data types (one per column)

   Ex. → TEXT, REAL, INTEGER, etc.

3. Column constraints

   NOT NULL (NN) → Data in this column can not be empty

   UNIQUE → Data in this column can not be repeated

   PRIMARY KEY (PK) → The primary key of the table (every table *must* have this)

Example Database Table: Astronomical Telescopes

| **Telescope Name**<br><br>*Type*: TEXT<br>*Constraints*: PK | **Latitude**<br><br>*Type*: REAL<br>*Constraints*: NN | **Longitude**<br><br>*Type*: REAL<br>*Constraints*: NN |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Relationships between database tables

Database tables are related by foreign keys.

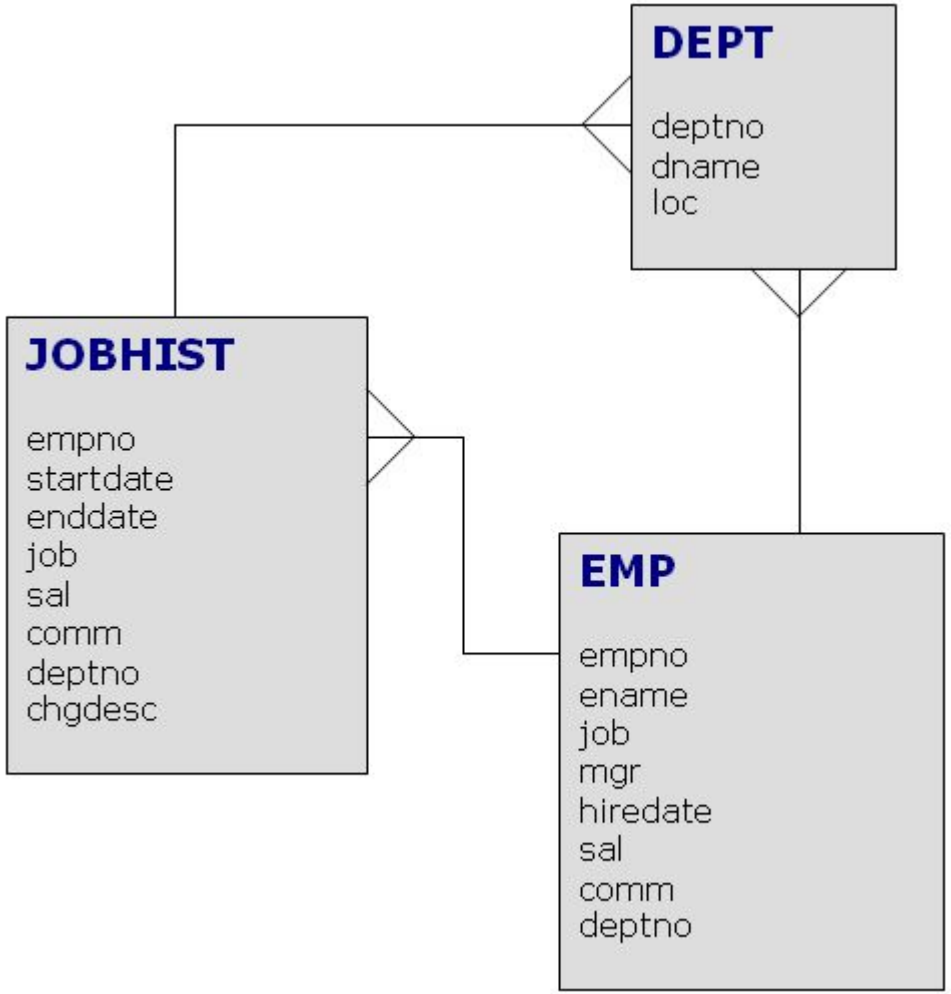
Types of relationships:

One-to-one (1:1) → One row in table 1 corresponds to one, and only one, row in table 2

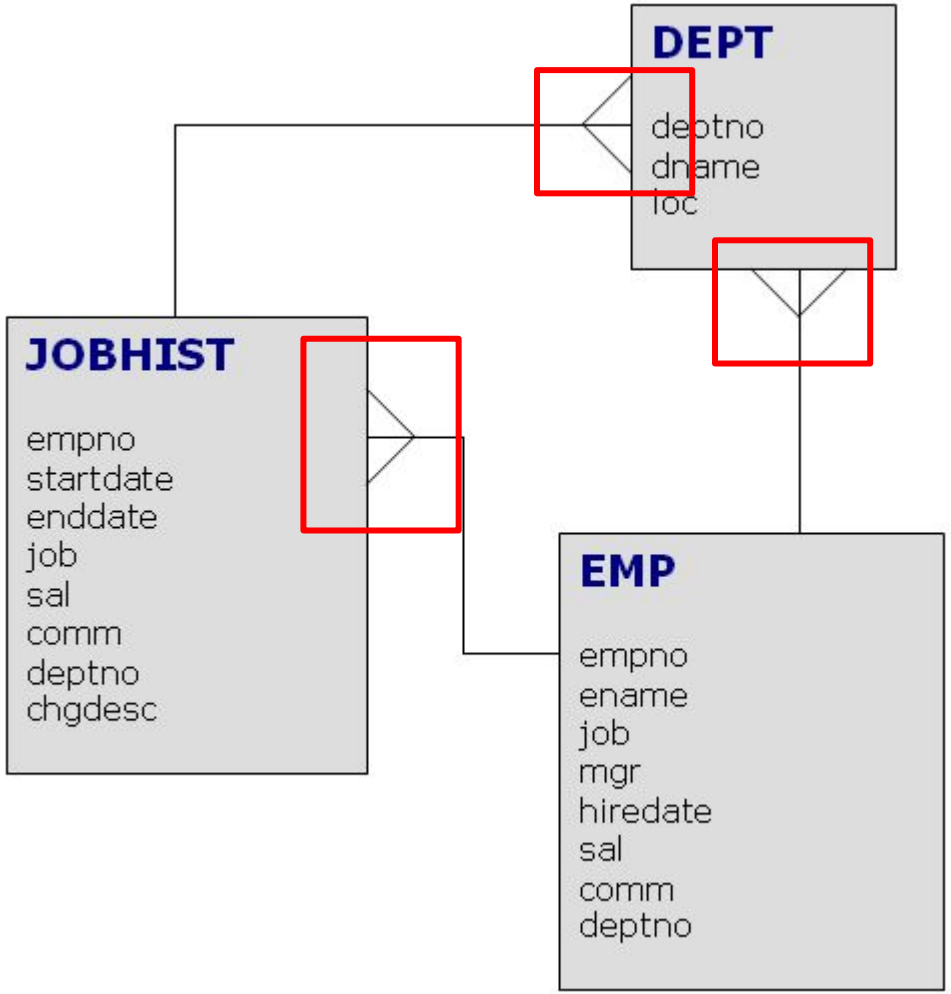
One-to-many (1:N) → One row in table 1 corresponds to many rows in table 2

Many-to-many (N:M) → Many rows in table 1 can correspond to many rows in table 2

Example Employee Database



Example Employee Database



# SQL Basics

# Creating a table*

Syntax

```
CREATE TABLE `TableName` (
  `Col1` TYPE [constraints],
  `Col2` TYPE [constraints],
  ....
);
```

Example

```
CREATE TABLE `Telescope` (
  `name` TEXT [PRIMARY KEY],
  `latitude` REAL [NOT NULL],
  `longitude` REAL [NOT NULL]
);
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|

*We won't practice this today, and it does differ depending on the database

# Inserting a row into a table

Syntax

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Example

```
INSERT INTO telescope (
        'MMT',
        31.6883,
        -110.8850
);
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |

# Inserting a row into a table

Syntax

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Example

```
INSERT INTO telescope (
        'MMT',
        31.6883,
        -110.8850
);
```

Result (after two more inserts)

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Getting data from a table

Syntax

```
SELECT col1, col2
FROM table_name;
```

Example

```
SELECT latitude, longitude
FROM telescope;
```

Input

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Getting data from a table

Syntax

```
SELECT col1, col2
FROM table_name;
```

Example

```
SELECT latitude, longitude
FROM telescope;
```

Result

| Latitude | Longitude |
|---|---|
| 31.6883 | -110.885 |
| 31.9648 | -111.5995 |
| -30.24464 | -70.74942 |

# Getting *all* data from a table

Syntax

```
SELECT *
FROM table_name;
```

Example

```
SELECT *
FROM telescope;
```

Input

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Getting *all* data from a table

Syntax

```
SELECT *
FROM table_name;
```

Example

```
SELECT *
FROM telescope;
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1;
```

Example

```
SELECT *
FROM telescope
WHERE name = 'MMT';
```

Input

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1;
```

Example

```
SELECT *
FROM telescope
WHERE name = 'MMT';
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1;
```

Example

```
SELECT *
FROM telescope
WHERE name != 'MMT';
```

Input

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1;
```

Example

```
SELECT *
FROM telescope
WHERE name != 'MMT';
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1;
```

Example

```
SELECT *
FROM telescope
WHERE name LIKE 'M%';
```

Input

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1;
```

Example

```
SELECT *
FROM telescope
WHERE name LIKE 'M%';
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1
AND condition2 AND …;
```

Example

```
SELECT *
FROM telescope
WHERE name LIKE ‘M%’
AND longitude > -111;
```

Input

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1
AND condition2 AND …;
```

Example

```
SELECT *
FROM telescope
WHERE name LIKE ‘M%’
AND longitude > -111;
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1
OR condition2 OR …;
```

Example

```
SELECT *
FROM telescope
WHERE name LIKE ‘M%’
OR latitude < 0;
```

Input

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Filtering data

Syntax

```
SELECT *
FROM table_name
WHERE condition1
OR condition2 OR …;
```

Example

```
SELECT *
FROM telescope
WHERE name LIKE ‘M%’
OR latitude < 0;
```

Result

| Telescope Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

# Joining two tables

Syntax

```
SELECT *
FROM table1
JOIN table2
WHERE table1.foreign_key =
table2.primary_key
```

Example

```
SELECT *
FROM telescope
JOIN observation
WHERE observation.telescope_name =
telescope.name
```

Input: `telescope`

| Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

Input: `observation`

| ID | telescope_name | date |
|---|---|---|
| 0 | MMT | Jan 31, 2026 |
| 1 | Mayall | Feb 1, 2026 |
| 2 | Rubin | Feb 1, 2026 |

# Joining two tables

Syntax

```
SELECT *
FROM table1
JOIN table2
WHERE table1.foreign_key =
table2.primary_key
```

Example

```
SELECT *
FROM telescope
JOIN observation
WHERE observation.telescope_name =
telescope.name
```

Ouput

| Name | Latitude | Longitude | ID | telescope_name | date |
|---|---|---|---|---|---|
| MMT | 31.6883 | -110.885 | 1 | MMT | Feb 1, 2026 |
| Mayall | 31.9648 | -111.5995 | 0 | Mayall | Jan 31, 2026 |
| Rubin | -30.24464 | -70.74942 | 2 | Rubin | Feb 1, 2026 |

# Joining two tables

Syntax

```
SELECT *
FROM table1
JOIN table2
WHERE table1.foreign_key =
table2.primary_key
AND table2.column condition;
```

Example

```
SELECT *
FROM telescope
JOIN observation
WHERE observation.telescope_name =
telescope.name
AND telescope.latitude < 0;
```

Input: `telescope`

| Name | Latitude | Longitude |
|---|---|---|
| MMT | 31.6883 | -110.885 |
| Mayall | 31.9648 | -111.5995 |
| Rubin | -30.24464 | -70.74942 |

Input: `observation`

| ID | telescope_name | date |
|---|---|---|
| 0 | MMT | Jan 31, 2026 |
| 1 | MMT | Feb 1, 2026 |
| 2 | Rubin | Feb 1, 2026 |

# Joining two tables

Syntax

```
SELECT *
FROM table1
JOIN table2
WHERE table1.foreign_key =
table2.primary_key
AND table2.column condition;
```

Example

```
SELECT *
FROM telescope
JOIN observation
WHERE observation.telescope_name =
telescope.name
AND telescope.latitude < 0;
```

Ouput

| Name | Latitude | Longitude | ID | telescope_name | date |
|---|---|---|---|---|---|
| Rubin | -30.24464 | -70.74942 | 2 | Rubin | Feb 1, 2026 |

Recap before the activity

datacamp

# MySQL Cheat Sheet

Learn SQL online at www.DataCamp.com

## What is MySQL?

MySQL is an open-source relational database management system (RDBMS) known for its fast performance and reliability. Developed by Oracle Corporation, it's widely used for web applications and online publishing.

## > Sample Data

The dataset contains details of the world's highest valued media franchises by gross revenue. Each row contains one franchise, and the table is named franchises.

| franchise | inception_year | total_revenue_busd | original_medium | owner | n_movies |
|---|---|---|---|---|---|
| Star Wars | 1977 | 46.7 | movie | The Walt Disney Company | 12 |
| Mickey Mouse and Friends | 1928 | 52.2 | cartoon | The Walt Disney Company | |
| Anpanman | 1973 | 38.4 | book | Froebel-kan | 33 |
| Winnie the Pooh | 1924 | 48.5 | book | The Walt Disney Company | 6 |
| Pokémon | 1996 | 88 | video game | The Pokémon Company | 24 |
| Disney Princess | 2000 | 45.4 | movie | The Walt Disney Company | |

## > Querying tables

**Get all the columns from a table using** `SELECT *`

```
SELECT *
  FROM franchises
```

**Get a column from a table by name using** `SELECT col`

```
SELECT franchise
  FROM franchises
```

**Get multiple columns from a table by name using** `SELECT col1, col2`

```
SELECT franchise, inception_year
  FROM franchises
```

**Override column names with** `SELECT col AS new_name`

```
SELECT franchise, inception_year AS creation_year
  FROM franchises
```

**Arrange the rows in ascending order of values in a column with** `ORDER BY col`

```
SELECT franchise, inception_year
  FROM franchises
  ORDER BY inception_year
```

**Arrange the rows in descending order of values in a column with** `ORDER BY col DESC`

```
SELECT franchise, total_revenue_busd
  FROM franchises
  ORDER BY total_revenue_busd DESC
```

**Limit the number of rows returned with** `LIMIT n`

```
SELECT *
  FROM franchises
  LIMIT 2
```

**Get unique values with** `SELECT DISTINCT`

```
SELECT DISTINCT owner
  FROM franchises
```

## > Filtering Data

### Filtering on numeric columns

**Get rows where a number is greater than a value with** `WHERE col > n`

```
SELECT franchise, inception_year
  FROM franchises
  WHERE inception_year > 1928
```

**Get rows where a number is greater than or equal to a value with** `WHERE col >= n`

```
SELECT franchise, inception_year
  FROM franchises
  WHERE inception_year >= 1928
```

**Get rows where a number is less than a value with** `WHERE col < n`

```
SELECT franchise, inception_year
  FROM franchises
  WHERE inception_year <= 1977
```

**Get rows where a number is equal to a value with** `WHERE col = n`

```
SELECT franchise, inception_year
  FROM franchises
  WHERE inception_year = 1996
```

**Get rows where a number is not equal to a value with** `WHERE col <> n` **or** `WHERE col != n`

```
SELECT franchise, inception_year
  FROM franchises
  WHERE inception_year <> 1996
```

**Get rows where a number is between two values (inclusive) with** `WHERE col BETWEEN m AND n`

```
SELECT franchise, inception_year
  FROM franchises
  WHERE inception_year BETWEEN 1928 AND 1977
```

### Filtering on text columns

**Get rows where text is equal to a value with** `WHERE col = 'x'`

```
SELECT franchise, original_medium
  FROM franchises
  WHERE original_medium = 'book'
```

**Get rows where text is one of several values with** `WHERE col IN ('x', 'y')`

```
SELECT franchise, original_medium
  FROM franchises
  WHERE original_medium IN ('movie', 'video game')
```

**Get rows where text contains specific letters with** `WHERE col LIKE '%abc%'`
(`%` represents any characters)

```
SELECT franchise, original_medium
  FROM franchises
  WHERE original_medium LIKE '%oo%'
```

### Filtering on multiple columns

**Get the rows where one condition and another condition holds with** `WHERE condn1 AND condn2`

```
SELECT franchise, inception_year, total_revenue_busd
  FROM franchises
  WHERE inception_year < 1950 AND total_revenue_busd > 50
```

**Get the rows where one condition or another condition holds with** `WHERE condn1 OR condn2`

```
SELECT franchise, inception_year, total_revenue_busd
  FROM franchises
  WHERE inception_year < 1950 OR total_revenue_busd > 50
```

### Filtering on missing data

**Get rows where values are missing with** `WHERE col IS NULL`

```
SELECT franchise, n_movies
  FROM franchises
  WHERE n_movies IS NULL
```

**Get rows where values are not missing with** `WHERE col IS NOT NULL`

```
SELECT franchise, n_movies
  FROM franchises
  WHERE n_movies IS NOT NULL
```

## > Aggregating Data

### Simple aggregations

**Get the total number of rows** `SELECT COUNT(*)`

```
SELECT COUNT(*)
  FROM franchises
```

**Get the total value of a column with** `SELECT SUM(col)`

```
SELECT SUM(total_revenue_busd)
  FROM franchises
```

**Get the mean value of a column with** `SELECT AVG(col)`

```
SELECT AVG(total_revenue_busd)
  FROM franchises
```

**Get the minimum value of a column with** `SELECT MIN(col)`

```
SELECT MIN(total_revenue_busd)
  FROM franchises
```

**Get the maximum value of a column with** `SELECT MAX(col)`

```
SELECT MAX(total_revenue_busd)
  FROM franchises
```

### Grouping, filtering, and sorting

**Get summaries grouped by values with** `GROUP BY col`

```
SELECT owner, COUNT(*)
  FROM franchises
  GROUP BY owner
```

**Get summaries grouped by values, in order of summaries with** `GROUP BY col ORDER BY smmry DESC`

```
SELECT original_medium, SUM(n_movies) AS total_movies
  FROM franchises
  GROUP BY original_medium
  ORDER BY total_movies DESC
```

**Get rows where values in a group meet a criterion with** `GROUP BY col HAVING condn`

```
SELECT original_medium, SUM(n_movies) AS total_movies
  FROM franchises
  GROUP BY original_medium
  ORDER BY total_movies DESC
  HAVING total_movies > 10
```

**Filter before and after grouping with** `WHERE condn_before GROUP BY col HAVING condn_after`

```
SELECT original_medium, SUM(n_movies) AS total_movies
  FROM franchises
  WHERE owner = 'The Walt Disney Company'
  GROUP BY original_medium
  ORDER BY total_movies DESC
  HAVING total_movies > 10
```

## > MySQL-Specific Syntax

Not all code works in every dialect of SQL. The following examples work in MySQL, but are not guaranteed to work in other dialects.

**Limit the number of rows returned, offset from the top with** `LIMIT m, n`

```
SELECT *
  FROM franchises
  LIMIT 2, 3
```

**By default, MySQL uses case insensitive matching in** `WHERE` **clauses.**

```
SELECT *
  FROM franchises
  WHERE owner = 'THE WALT DISNEY COMPANY'
```

**To get case sensitive matching, use** `WHERE BINARY condn`

```
SELECT *
  FROM franchises
  WHERE BINARY owner = 'THE WALT DISNEY COMPANY'
```

**Get the current date with** `CURDATE()` **and the current** `datetime` **with** `NOW()` **or** `CURTIME()`

```
SELECT CURDATE(), NOW(), CURTIME()
```

**List available tables with** `show tables`

```
show tables
```

Recap before the activity

DON’T FORGET YOUR SEMICOLONS!

# Your Task: Plan and track observations of targets

Imagine you are helping manage a large team of observers working at multiple telescopes, and you want to make sure observations aren’t being duplicated.

What types of information do we need to store?

# Your Task: Plan and track observations of targets

Imagine you are helping manage a large team of observers working at multiple telescopes, and you want to make sure observations aren’t being duplicated.

What types of information do we need to store?

- Target name, coordinates, and brightness

# Your Task: Plan and track observations of targets

Imagine you are helping manage a large team of observers working at multiple telescopes, and you want to make sure observations aren’t being duplicated.

What types of information do we need to store?

- Target name, coordinates, and brightness
- Telescope name and coordinates

# Your Task: Plan and track observations of targets

Imagine you are helping manage a large team of observers working at multiple telescopes, and you want to make sure observations aren’t being duplicated.

What types of information do we need to store?

- Target name, coordinates, and brightness
- Telescope name and coordinates
- Observation details: target, telescope, exposure time, filter used

# So, what database tables do we need?

Discuss with your tables for ~5 minutes

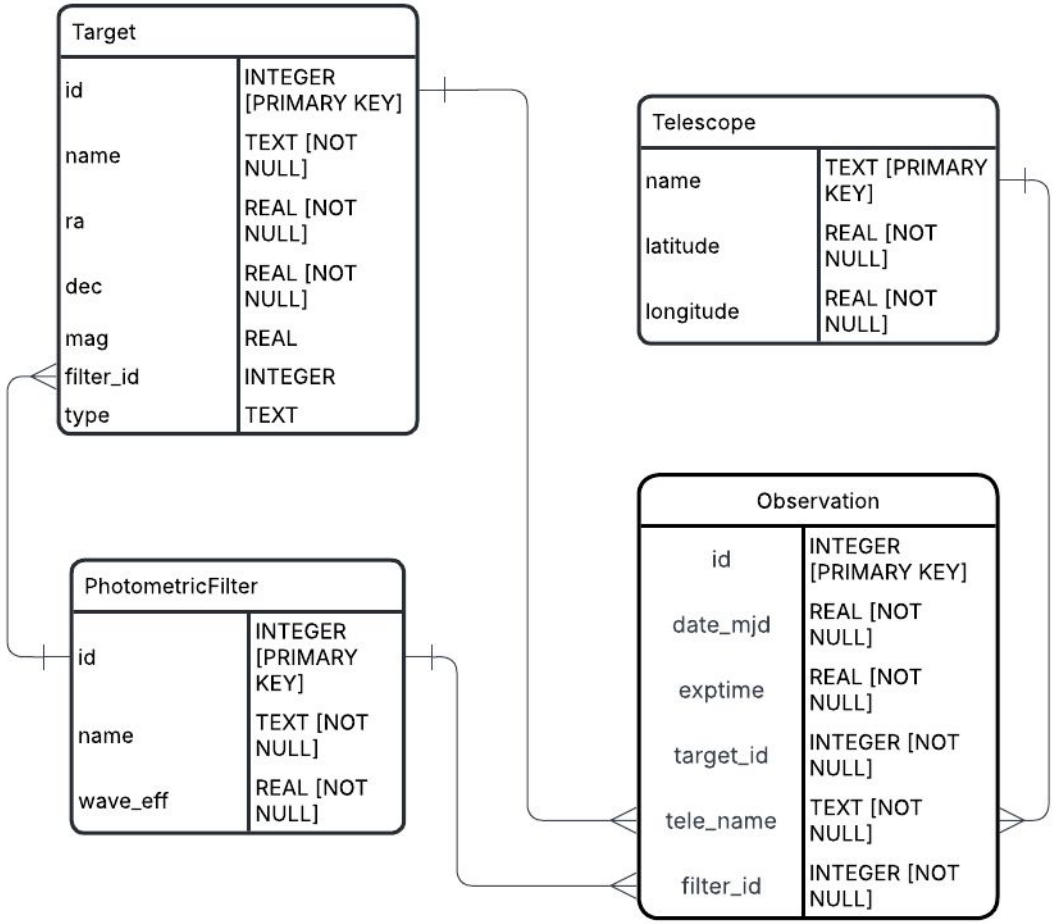
REMINDER: We want to limit the duplication of data!

# So, what tables do we need?

Here's what I did

- Target
- PhotometryFilter
- Telescope
- Observation

Tip: This is called an "Entity Relationship" Diagram. Look it up if you want to learn more!



# Briefly show off a real database

# Let’s look at this basic database

https://sqliteviewer.app/

# And we can play around with querying this database!

A SQLite playground: https://sqlime.org/

You will find a worksheet with activity details on D2L.