GENERAL ASSEMBLY

DATABASES

Angelo Klin

Katra Analytics

LEARNING OBJECTIVES

- Understanding of the uses and differences of databases
- Accessing databases from Pandas

PRE-WORK

PRE-WORK REVIEW

- There will be multiple ways to run the exercises:
  - Using Postgres Exercises
    - Install Postgres
      - macOS: If brew is installed, this should be as simple as

```
brew install postgres
```

  - Use Wagon
    - Create an account at https://www.wagonhq.com and download the software

OPENING

DATABASES

DATABASES

- Today’s lesson will be on databases and SQL
- Databases are the standard solution for data storage
  - They are far more robust than text and CSV files
- They come in many flavours, but we will explore the most common
  - Relational Databases

DATABASES

- Relational Databases also come in different varieties, but almost all use SQL (Structured Query Language) as a basis for querying (i.e. retrieving) data

- Most analyses typically involve pulling data from a database

INTRODUCTION

DATABASES

DATABASES

- Databases are computer systems that manage the storage and querying of data sets
- They provide a way to organise the data on disk (i.e. hard drive) and efficient methods to retrieve information
  - Databases allow a user to create rules that ensure proper data management and verification
- Typically, retrieval is performed using a query language with a few operators for data transformation
- The most common query language is SQL (Structured Query Language

DATABASES

- A relational database is based on links between data entities or concepts

- Typically, relational databases are organised into tables

- Each table should correspond to one entity or concept
  - Each table is similar to a single CSV file or Pandas dataframe

- For example, consider an application like Twitter
  - Our two main entities are Users and Tweets and for each of these we would have a separate table

DATABASES

- A table is made up of rows and columns, similar to a Pandas dataframe or a spreadsheet
- Each table has a specific schema, a set of rules for what goes in each table
    - These specify which columns are contained in the table and what type of data is in each column (e.g. text, integers, decimals, etc)

Users Table Schema	
Field	**Type**
user_id	char
user_sign_up_date	date
user_follower_count	int

DATABASES

- This means you can not add text data to an integer column in that database

- The additional type information make this constraint stronger than the header of a CSV file

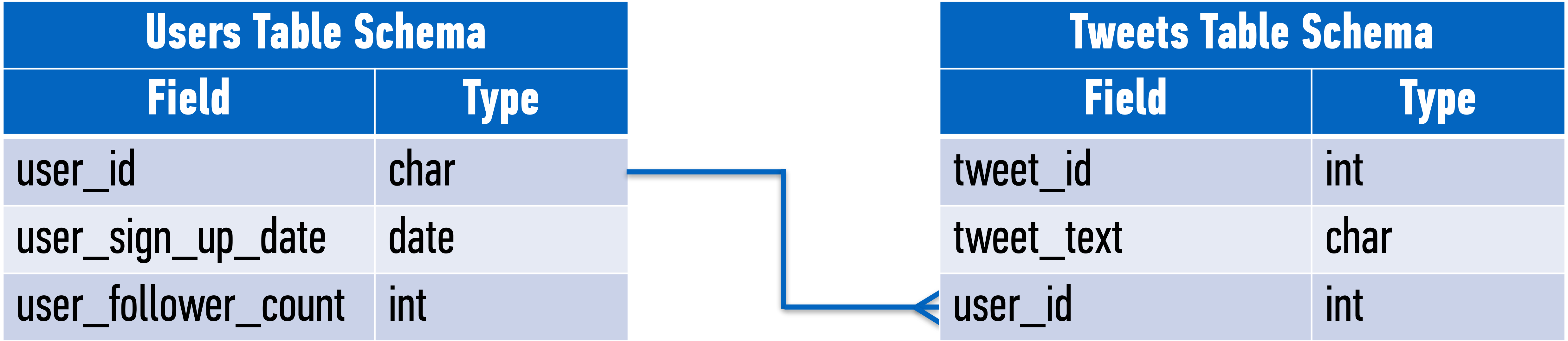
- For this reason and many others, databases allow for stronger consistency of the data and are often a better solution for data storage

DATABASES

- Each table typically has a **primary key** column
  - This column has a unique value per row and serves as the identifier for the row
- A table can have many **foreign keys** as well
  - A foreign key is a column that contains values to link the table to the other tables
- These keys that link the table together define the relational database

DATABASES

- For example, the tweets table may have as columns:
  - tweet_id the primary key tweet identifier
  - tweet_text
  - user_id a foreign key to the users table



Users Table Schema	
Field	**Type**
user_id	char
user_sign_up_date	date
user_follower_count	int

Tweets Table Schema	
Field	**Type**
tweet_id	int
tweet_text	char
user_id	int

DATABASES

- MySQL and Postgres are popular variants of relational databases and are widely used
  - Both are open-source and available for free

- Alternatively, many companies use proprietary software such as IBM DB2, Oracle or Microsoft databases

- While these databases offer many of the same features and use the same SQL language, the latter three offer some maintenance features and support that large companies find useful

NORMALISED AND DENORMALISED DATA

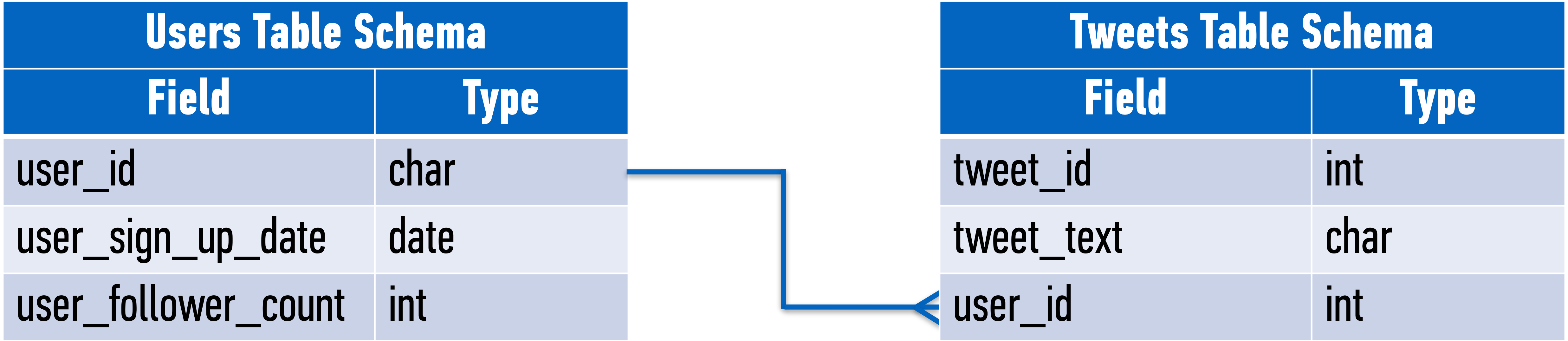
- Once we start organising our data into tables, we start to separate it into normalised and denormalised setups

- **Normalised** structures have a single table per entity and use many foreign keys or link tables to connect the entities

- **Denormalised** structures have fewer tables that combine different entities

NORMALISED AND DENORMALISED DATA

- With our Twitter example, a **normalised** structure would place users and tweets in **different** tables



Users Table Schema	
Field	Type
user_id	char
user_sign_up_date	date
user_follower_count	int

Tweets Table Schema	
Field	Type
tweet_id	int
tweet_text	char
user_id	int

NORMALISED AND DENORMALISED DATA

- A **denormalised** structure would put them **both** in one table

Tweets Table Schema	
Field	**Type**
tweet_id	int
tweet_text	char
user_id	int
user_sign_up_date	date
user_follower_count	int

NORMALISED AND DENORMALISED DATA

Normalised structures	Denormalised structures
Save storage space by separating information	Duplicates a lot of information
Requires joining of table to access information about two different entities, a slow operation	Makes data easy to access since it is all in one table

ALTERNATIVE DATABASES

- While relational databases are the most popular and broadly used, specific applications may require different data organisation

- You do not need to know every variety, but it is good to know some overall themes

KEY-VALUE STORES

- **Key-Value** databases are nothing more than very large and very fast hash maps or dictionaries
- These are useful for storing key based data, e.g. a count of things per user or customer, a last visit per customer
- Every entry in these databases has two values, a key and a value
  - We can retrieve any value based upon its key

KEY-VALUE STORES

- This is exactly like a Python dictionary, but it can be larger than your memory (i.e. RAM)
  - So these systems use smart caching algorithms to ensure frequently or recently accessed items are quickly accessible

- Popular key-value stores include
  - Cassandra and
  - MemcacheDB (pronounced mem-cash-dee-bee)

NO-SQL OR DOCUMENT DATABASES

- NoSQL databases are those that do not rely on a traditional relational table setup and more flexible in their data organisation
  - Definitions vary, but one is “Not Only SQL”

- Typically they actually do have SQL querying abilities but model their data differently

NO-SQL OR DOCUMENT DATABASES

- Relational Structure

user_id	user_name	age
13123	robby_g	25
18423	jt1235	31

user_id	user_hobby
13123	guitar
13123	cars
18423	football

- NoSQL Data Structure

```
{
  "user_id": 13123,
  "user_name": "robby_g",
  "user_hobbies": ["guitar", "cars"],
  "user_age": 25
}
```

```
{
  "user_id": 19423,
  "user_name": "jt1235",
  "user_hobbies": ["football"],
  "user_age": 31
}
```

NO-SQL OR DOCUMENT DATABASES

- They may organise data on an entity level, but often have denormalised and nested data setups

- This nested data layout is often similar to that in JSON documents

- Popular databases include
  - MongoDB and
  - CouchDB

NO-SQL OR DOCUMENT DATABASES

Relational data model

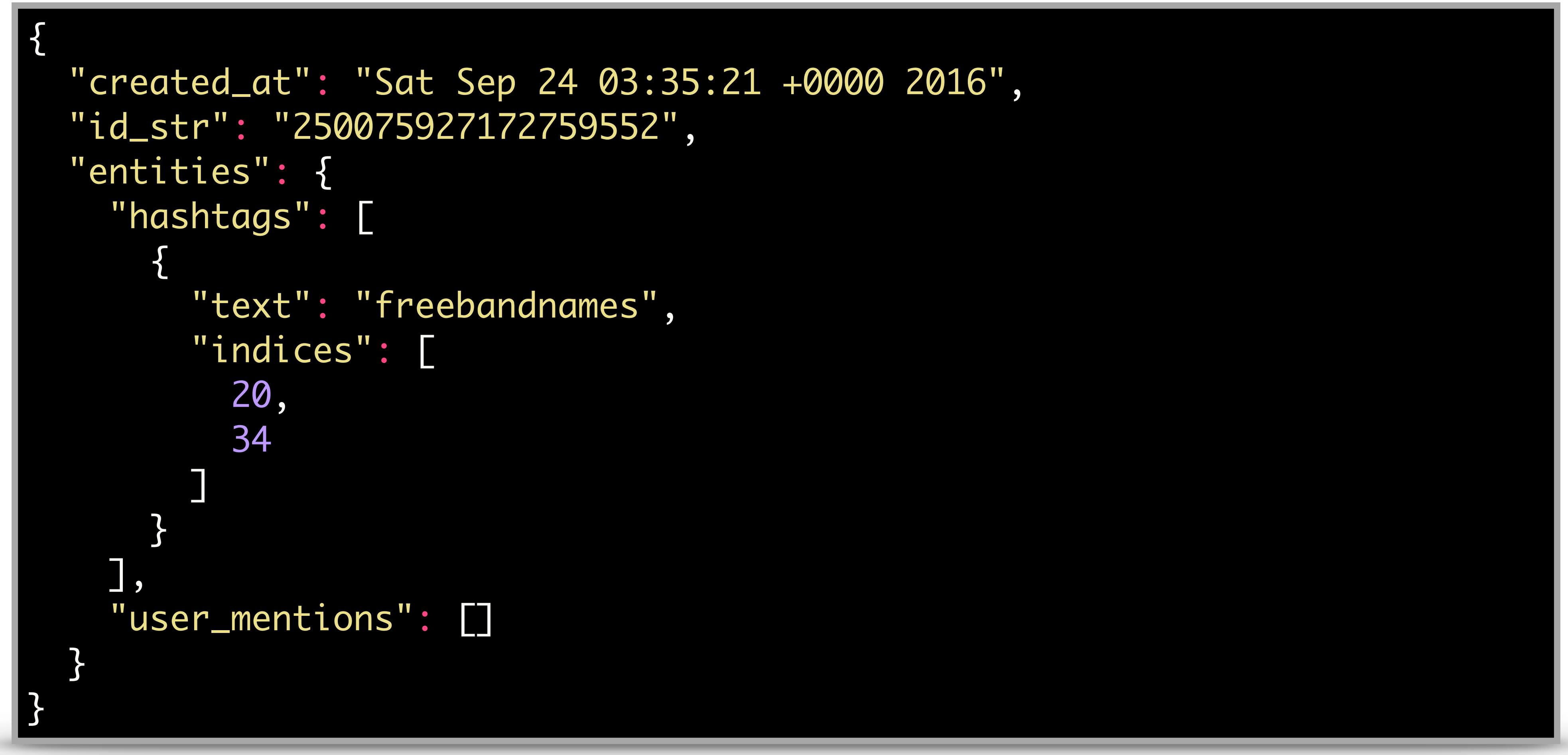
Highly-structured table organization with rigidly-defined data formats and record structure.

Document data model

Collection of complex documents with arbitrary, nested data formats and varying "record" format.

NO-SQL OR DOCUMENT DATABASES

- The following is an example of the storage document for a tweet



```
{
  "created_at": "Sat Sep 24 03:35:21 +0000 2016",
  "id_str": "250075927172759552",
  "entities": {
    "hashtags": [
      {
        "text": "freebandnames",
        "indices": [
          20,
          34
        ]
      }
    ],
    "user_mentions": []
  }
}
```

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

DIRECTIONS: ANSWER THE FOLLOWING QUESTIONS

1. In the following examples, which might be the best storage or database solution and why?
   a. An application where a user can create a profile
   b. An online store
   c. Storing the last visit date of a user

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

DIRECTIONS: ANSWER THE FOLLOWING QUESTIONS

1. Consider a data set from Uber with the following fields
   - User ID
   - User Name
   - Driver ID
   - Drive Name
   - Ride ID
   - Ride Time
   - Pickup Latitude
   - Pickup Longitude
   - Pickup Location Entity
   - Dropoff Longitude
   - Dropoff Latitude
   - Dropoff Location Entity
   - Miles
   - Travel Time
   - Fare
   - CC Number
2. In a group, discuss how you would design a relational database to support this data
3. List the tables you would create, the fields they would contain and how they would link to other tables

DEMONSTRATION

ACCESSING DATABASES FROM PANDAS

ACCESSING DATABASES FROM PANDAS

- While databases provide many analytical capabilities, often it is useful to pull the data back into Python for more flexible programming

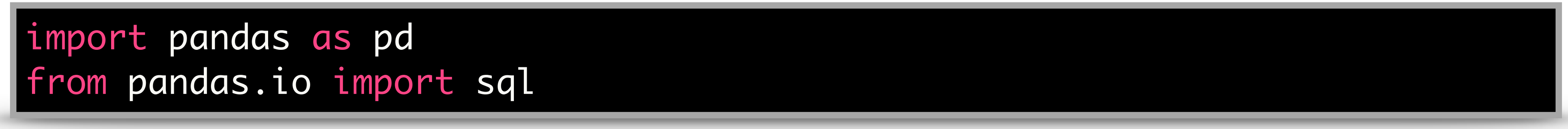
- Large, fixed operations would be more efficient in a database, but Pandas allows for interactive processing

- For example, if you just want to aggregate login or sales data to present a report or dashboard, this operation is operating on a large data set and not often changing

- This would run very efficiently in a database vs connecting to Python

ACCESSING DATABASES FROM PANDAS

- However, if we want to investigate the login or sales data further and ask more interactive questions, then using Python would come in very handy



```python
import pandas as pd
from pandas.io import sql
```

- Pandas can be used to connect to most relational databases

ACCESSING DATABASES FROM PANDAS

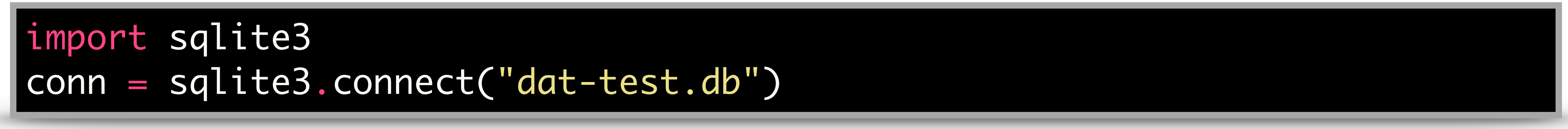
- In this demonstration, we will create and connect to a SQLite database
  - SQLite creates portable relational databases saved in a single file

- These databases are stored in a very efficient manner and allow fast querying, making them ideal for small databases or databases that need to be moved across machines

- Additionally, SQLite databases can be created with the setup of MySQL or Postgres databases

ACCESSING DATABASES FROM PANDAS

- We can create a SQLite databases as follows



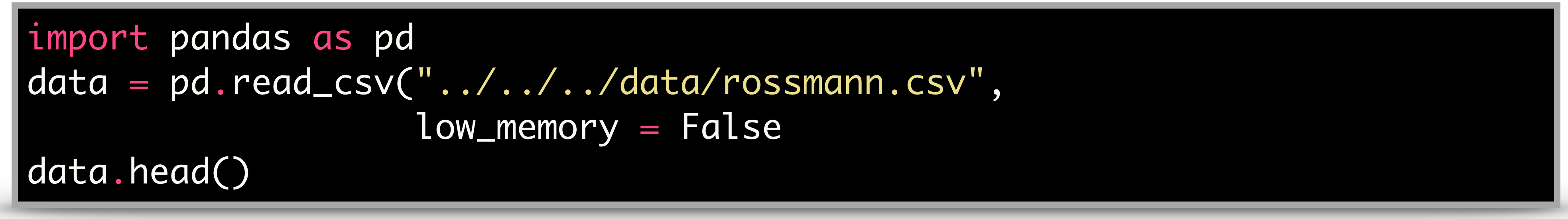
```
import sqlite3
conn = sqlite3.connect("dat-test.db")
```

- This creates a file, dat-test.db, which will act as a relational/SQL database

WRITING DATA INTO A DATABASE

- Data in Pandas can be loaded into a relational database
  - For the most part, Pandas can use the databases column information to infer the schema for the table it creates

- Let's return to the Rossmann sales data and load it into our database



```python
import pandas as pd
data = pd.read_csv("../../../data/rossmann.csv",
                   low_memory = False
data.head()
```

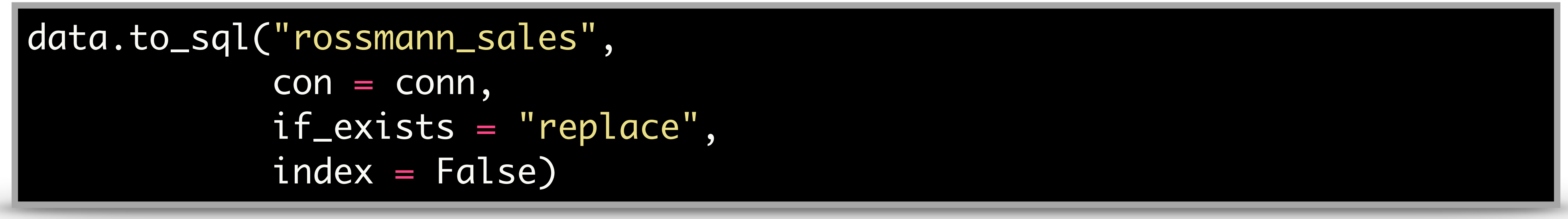
ACCESSING DATABASES FROM PANDAS

- Data is moved to the database with the `to_sql` command, similar to the `to_csv` command

- `to_sql` takes several arguments
  - `name` the table name to create
  - `con` a connection to a database
  - `index` whether to input the index column
  - `schema` if we want to write a custom schema for the new table
  - `if_exists` what to do if the table already exists: overwrite, add or fail

WRITING DATA INTO A DATABASE

- The following code loads the Rossmann sales data to our database

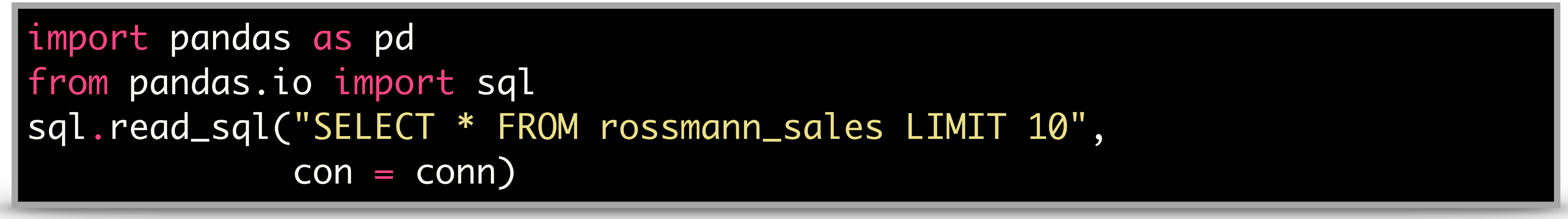


```
data.to_sql("rossmann_sales",
            con = conn,
            if_exists = "replace",
            index = False)
```

READING DATA FROM A DATABASE

- If we already have data in the database, we can use Pandas to query our database

- Querying is done through the read_sql command in the sql module



```
import pandas as pd
from pandas.io import sql
sql.read_sql("SELECT * FROM rossmann_sales LIMIT 10",
             con = conn)
```

- This runs the query passed in and returns a dataframe with the results

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

DIRECTIONS: (5 MINUTES)

1. Load the Rossmann Store metadata in rossmann-stores.csv and create a table in the database with it

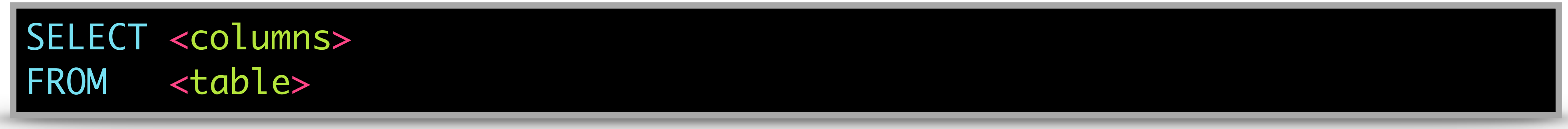
DEMONSTRATION

EXPLORING ROSSMANN DRUGSTORE SALES DATA

SQL OPERATORS: SELECT

- Every query should start with SELECT
  - SELECT is followed by the names of the columns in the output

- SELECT is always paired with FROM, which identifies the table to retrieve data from

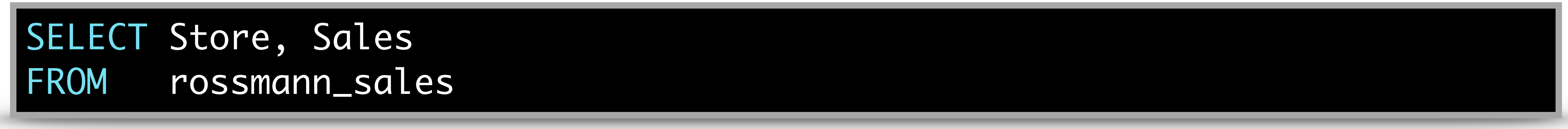


```
SELECT <columns>
FROM   <table>
```

- SELECT * denotes returning all of the columns

SQL OPERATORS: SELECT

- Rossmann Stores example



```
SELECT Store, Sales
FROM   rossmann_sales
```

ACTIVITY: KNOWLEDGE CHECK

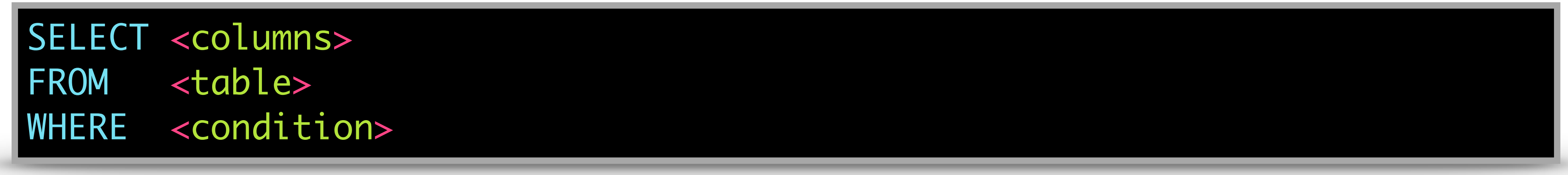
EXERCISE

DIRECTIONS: ANSWER THE FOLLOWING QUESTIONS

1. Write a query for the Rossmann Sales data that returns Store, Date and Customers

SQL OPERATORS: WHERE

- WHERE is used to filter a table using a specific criteria
  - The WHERE clause follows the FROM clause

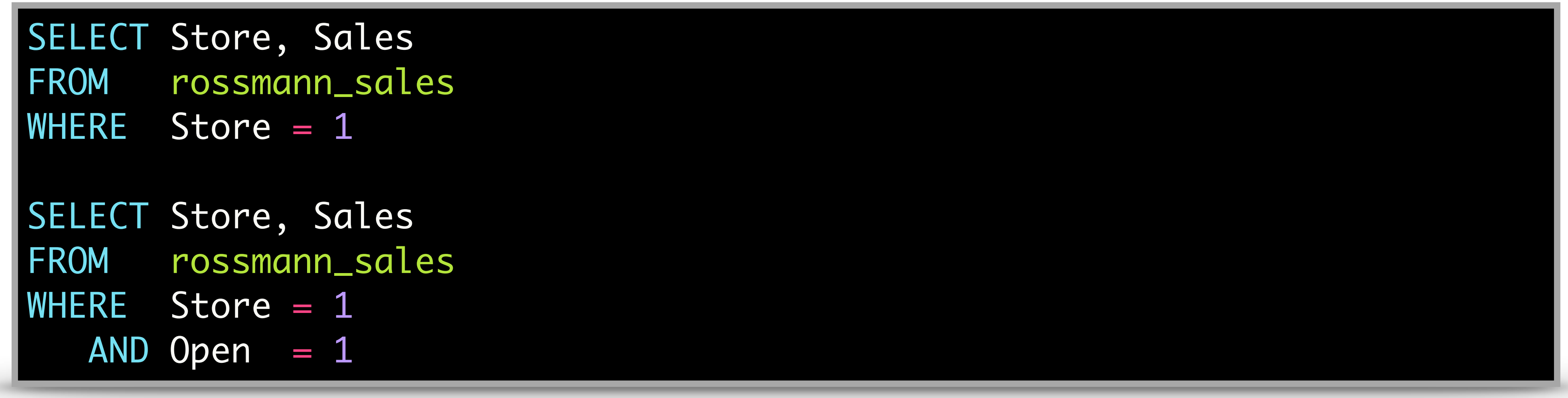


```
SELECT <columns>
FROM   <table>
WHERE  <condition>
```

- The condition is some filter applied to the rows, where rows that match the condition will be output

SQL OPERATORS: WHERE

- Rossmann Stores example



```
SELECT Store, Sales
FROM   rossmann_sales
WHERE  Store = 1

SELECT Store, Sales
FROM   rossmann_sales
WHERE  Store = 1
   AND Open  = 1
```

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

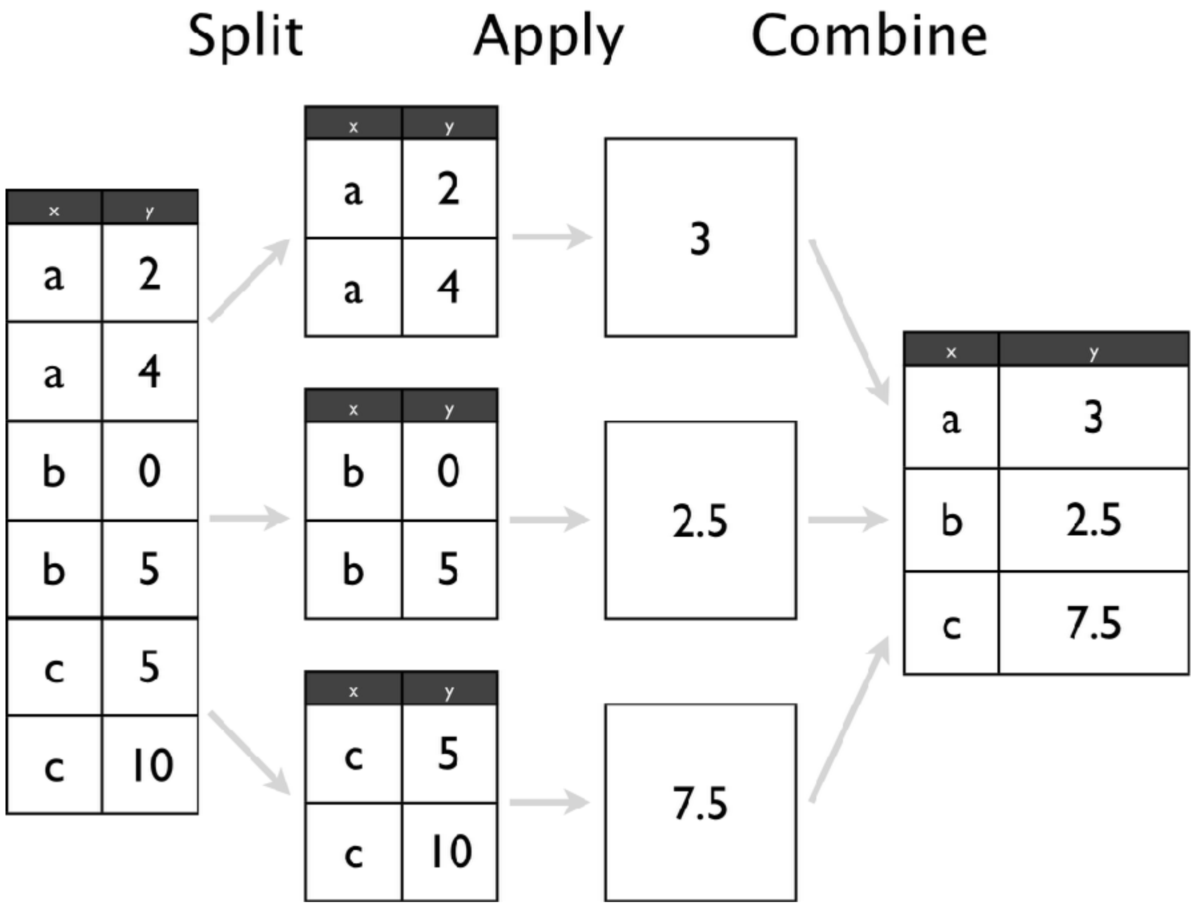
DIRECTIONS: ANSWER THE FOLLOWING QUESTIONS

1. Write a query for the Rossmann Sales data that returns Store, Date and Customers for stores that were open and running a promotion

SQL OPERATORS: GROUP BY

- GROUP BY allows us to aggregate over any field in the table by applying the concept of **Split**, **Apply**, **Combine**

- We identify some key with which we want to segment the rows, then we roll up or compute some statistics over all of the rows that match that key



x	y
a	2
a	4
b	0
b	5
c	5
c	10

x	y
a	2
a	4

x	y
b	0
b	5

x	y
c	5
c	10

x	y
a	3
b	2.5
c	7.5

SQL OPERATORS: GROUP BY

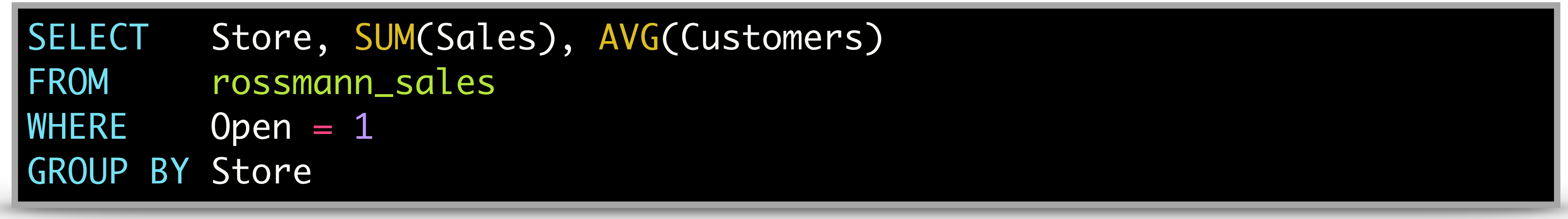
- GROUP BY must be paired with an aggregate function, the statistic we want to compute in the rows, in the SELECT statement

- COUNT(*) denotes counting up all of the rows
  - Other aggregate functions commonly available are AVG (average), MAX, MIN and SUM

- If we want to aggregate over the entire table, without results specific to any key, we can use an aggregate function in the SELECT clause and ignore the GROUP BY clause

SQL OPERATORS: GROUP BY

- Rossmann Stores example



```
SELECT   Store, SUM(Sales), AVG(Customers)
FROM     rossmann_sales
WHERE    Open = 1
GROUP BY Store
```

ACTIVITY: KNOWLEDGE CHECK

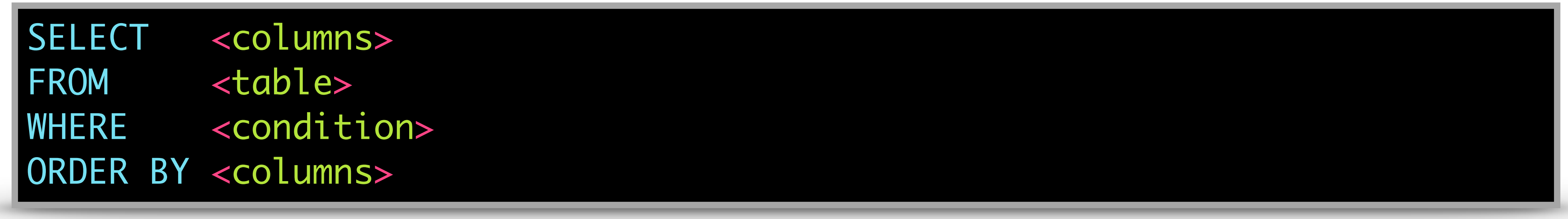
EXERCISE

DIRECTIONS: ANSWER THE FOLLOWING QUESTIONS

1. Write a query that returns the total sales on the promotion and non-promotion days

SQL OPERATORS: ORDER BY

- ORDER BY is used to sort the results of a query

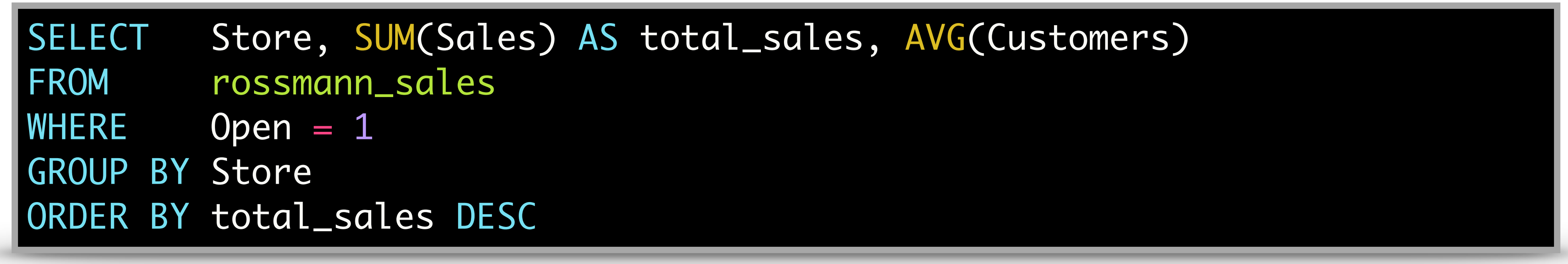


```
SELECT   <columns>
FROM     <table>
WHERE    <condition>
ORDER BY <columns>
```

- You can order by multiple columns in ascending (ASC) or descending (DESC) order

SQL OPERATORS: ORDER BY

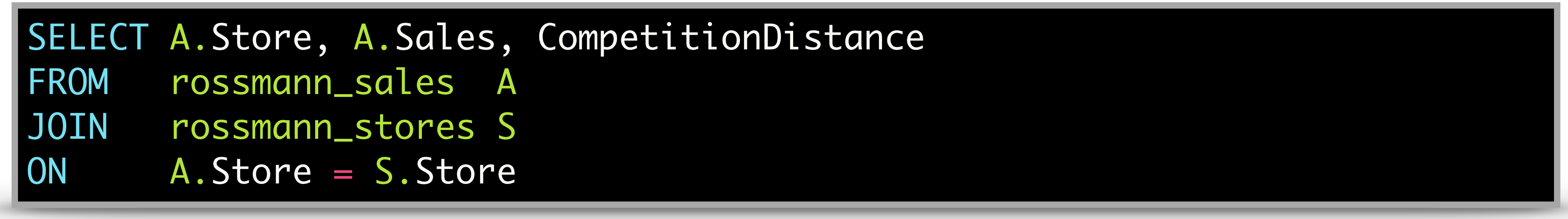
- Rossmann Stores example



```
SELECT   Store, SUM(Sales) AS total_sales, AVG(Customers)
FROM     rossmann_sales
WHERE    Open = 1
GROUP BY Store
ORDER BY total_sales DESC
```

SQL OPERATORS: JOIN

- JOIN allows us to access data across many tables
  - We specify how a row in one table links to another



```
SELECT A.Store, A.Sales, CompetitionDistance
FROM   rossmann_sales  A
JOIN   rossmann_stores S
ON     A.Store = S.Store
```

- Here, ON denotes an **inner** join

SQL OPERATORS: JOIN

- By default, most joins are an **Inner Join**, which means only when there is a match in both tables does a row appear in the results

- If we want to keep the rows of one table **even if there is no matching counterpart**, we can perform an **Outer Join**

- Outer joins can be LEFT, RIGHT or FULL, meaning keep all of the left rows, all the right rows or all the rows, respectively

INDEPENDENT PRACTICE

PANDAS AND SQL

ACTIVITY: PANDAS AND SQL

EXERCISE

DIRECTIONS: (40 MINUTES)

1. Load the Walmart sales and store features data
2. Create a table for each of those data sets
3. Select the store, date and fuel price on days it was over 90 degrees
4. Select the store, date and weekly sales and temperature
5. What were average sales on holiday vs. non-holiday sales?
6. What were average sales on holiday vs. non-holiday sales when the temperature was below 32 degrees?

INDEPENDENT PRACTICE

EXTRA SQL PRACTICE

ACTIVITY: EXTRA SQL PRACTICE

EXERCISE

DIRECTIONS: ANSWER THE FOLLOWING QUESTIONS

PG-Exercises

1. The website pgexercises.com is a very good site for Postgres exercises
2. Go to PG Exercises-Getting Started to get started
3. Complete 3-5 questions in each of the following
   a. Simple SQL Queries
   b. Aggregation
   c. Joins and Subqueries

ACTIVITY: EXTRA SQL PRACTICE

EXERCISE

DIRECTIONS: ANSWER THE FOLLOWING QUESTIONS

Wagon

1. This requires signing up for the Wagon service and downloading their application, it gives access to some sample databases
   a. Display all tracks on which Jimmy Page was the composer
   b. Who were the top five composers by number of tracks?
   c. Who were the top five composers by length of tracks?
   d. Select all of the albums from Led Zeppelin
   e. Count the number of albums per artist and display the top 10
   f. Display the track name and album name from all Led Zeppelin albums
   g. Compute how many songs and how long (in minutes) each Led Zeppelin album was

CONCLUSION

TOPIC REVIEW

TOPIC REVIEW

- While this was a brief introduction, databases are often at the core of any data analysis
  - Most analysis starts with retrieving data from a database

- SQL is a key language that any data scientist should understand
  - `SELECT` : Used in every query to define the resulting columns
  - `WHERE` : Filters rows based on a given condition
  - `GROUP BY` : Groups rows for aggregation
  - `JOIN` : Combines two tables based upon a given condition

TOPIC REVIEW

- Pandas can be used to access data from databases as well
  - The result of the queries will end up in a Pandas dataframe

- There is much more to learn about query optimisation if one dives further!

BEFORE NEXT CLASS

BEFORE NEXT CLASS

DUE DATE

- Project
  - Final Project, part 4

DATABASES

Q & A

CREDITS AND REFERENCES

CREDITS AND REFERENCES

- Pandas: Comparison with SQL
- PostgresSQL Exercises
- SQL Zoo