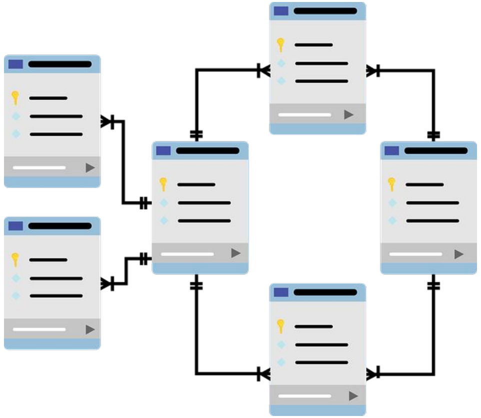
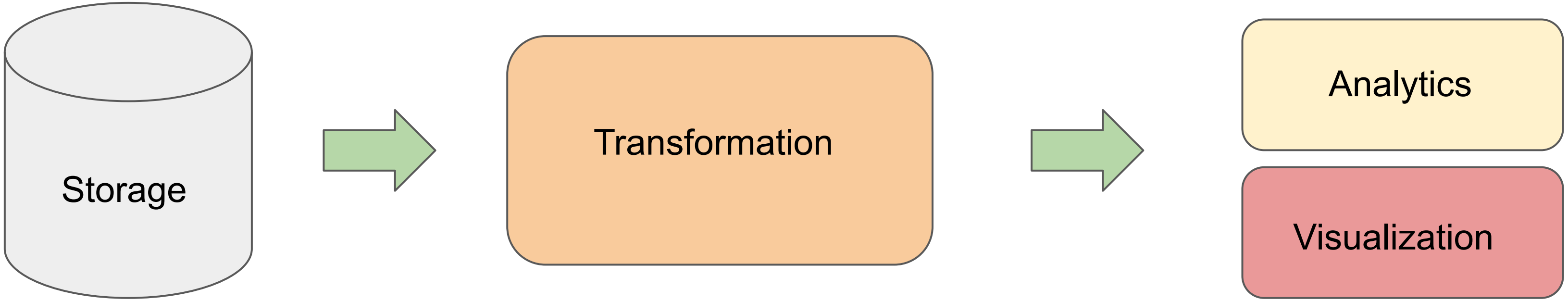
Data Transformations



Data Transformations for Data Science

- Data in source vs Data how you want it
- Data storage and analytics should be “de-coupled”
  - Same data can answer multiple questions
  - What queries should be sped up?
  - What information is needed ASAP vs at set intervals?

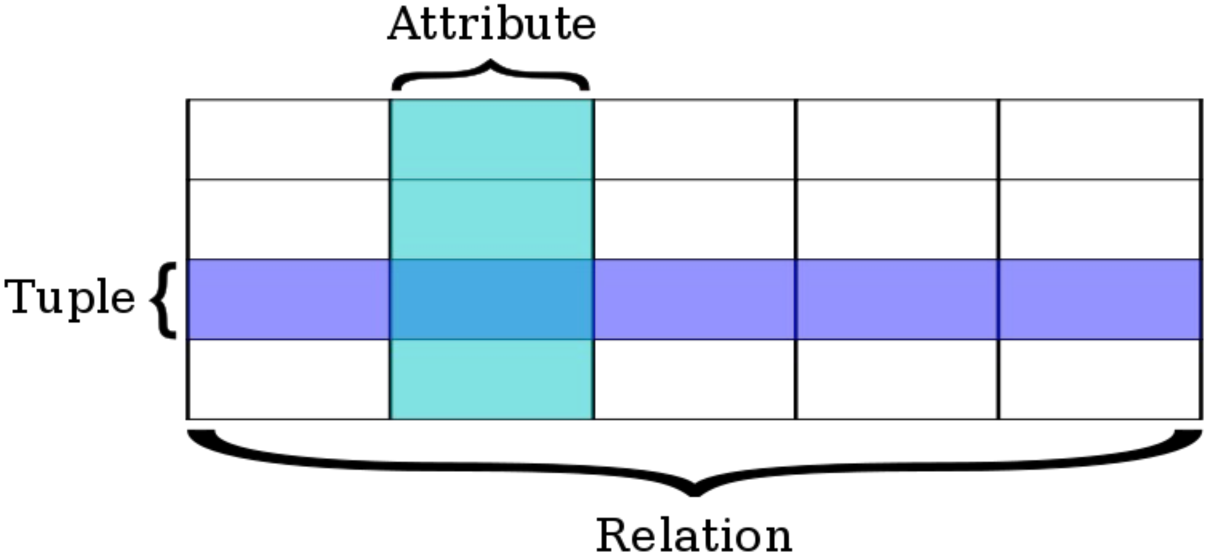


Data Storage

- How can data be stored?
    - Collection of files
        - Versatile
        - Slow for querying
    - Database - Organized by an index
        - Fulfils a more specific purpose
        - Fast querying
        - Requires planning at time of data collection
    - Database - No Index
        - More versatile than with index, still needs up-front planning
        - Quick if in-memory
        - Slower than with an index

Relational Databases

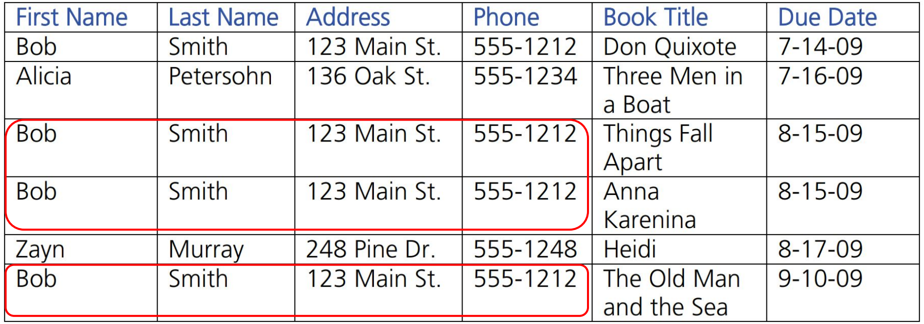
- Relational Model
  - Relations = Tables
  - Data stored as a set of related tables



Relational Databases

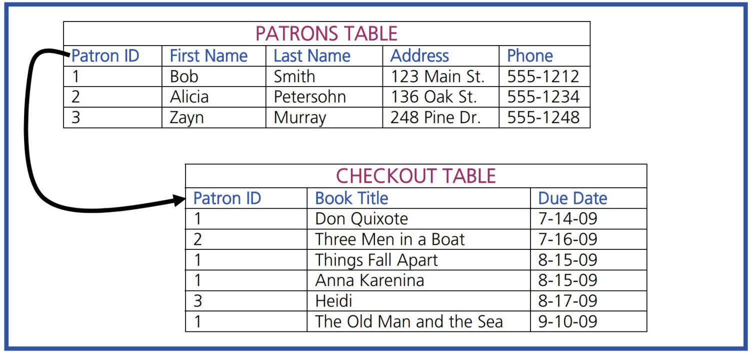
Ex: Books being checked out of library

1. Flat table



First Name	Last Name	Address	Phone	Book Title	Due Date
Bob	Smith	123 Main St.	555-1212	Don Quixote	7-14-09
Alicia	Petersohn	136 Oak St.	555-1234	Three Men in a Boat	7-16-09
Bob	Smith	123 Main St.	555-1212	Things Fall Apart	8-15-09
Bob	Smith	123 Main St.	555-1212	Anna Karenina	8-15-09
Zayn	Murray	248 Pine Dr.	555-1248	Heidi	8-17-09
Bob	Smith	123 Main St.	555-1212	The Old Man and the Sea	9-10-09

2. Relational database



PATRONS TABLE

Patron ID	First Name	Last Name	Address	Phone
1	Bob	Smith	123 Main St.	555-1212
2	Alicia	Petersohn	136 Oak St.	555-1234
3	Zayn	Murray	248 Pine Dr.	555-1248

CHECKOUT TABLE

Patron ID	Book Title	Due Date
1	Don Quixote	7-14-09
2	Three Men in a Boat	7-16-09
1	Things Fall Apart	8-15-09
1	Anna Karenina	8-15-09
3	Heidi	8-17-09
1	The Old Man and the Sea	9-10-09

Courtesy: https://webs.wofford.edu/whisnantdm/Courses/CS101/PDF/Database/Relational_database_concepts.pdf

SQL - Structured Query Language

- This is not an SQL course!
- SQL is a standard
  - Implementations can differ - the language is fairly consistent
  - Will encounter it in jobs often
  - Can transform data
  - Can pull data (query)
- Mostly meant for relational databases
  - Some systems have adapted it for types of storage
- https://www.w3schools.com/sql/ https://www.w3schools.com/sqL/trysql.asp?filename=trysql_select_top&ss=-1

Select

```
SELECT column_1, column_2, ...
FROM table_name;
```

```
SELECT *
FROM table_name;
```

- Stored in “Result Set”
- Returns all rows

Distinct

```
SELECT DISTINCT column_1, column_2, ...
FROM table_name;
```

```
SELECT DISTINCT *
FROM table_name;
```

- Shows distinct values

Conditions - Where Statement

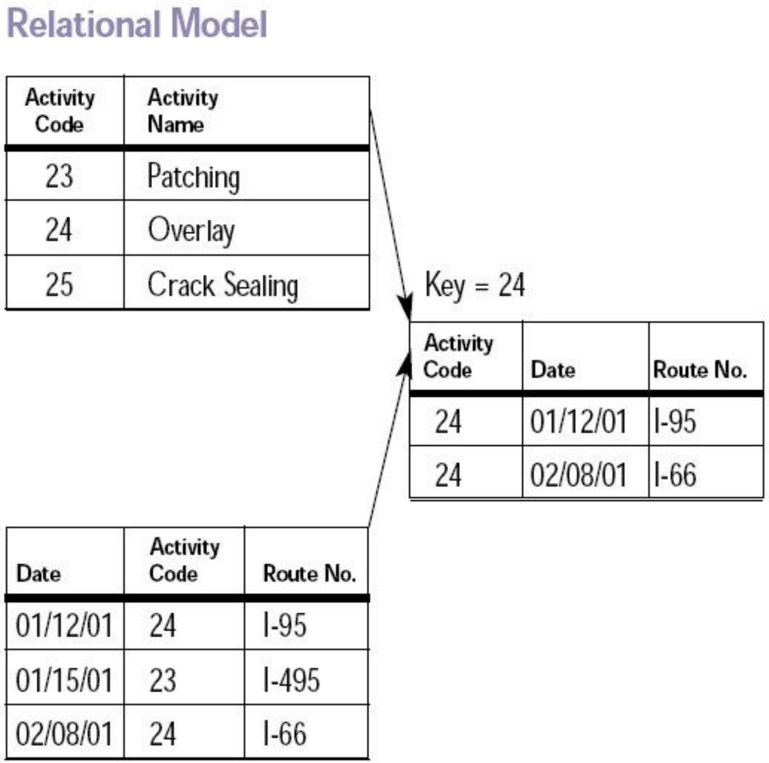
```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Introduces a constraint to a query
- “Filters” results
- E.g. SELECT name from student_table WHERE class_enrolled=DS-CERT.

Where - Conditions

=	Equal
<> or !=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between a range (inclusive)
LIKE	Patterns - https://www.w3schools.com/sql/sql_like.asp
IN	List of possible values

Integrating between tables



Relational Model

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

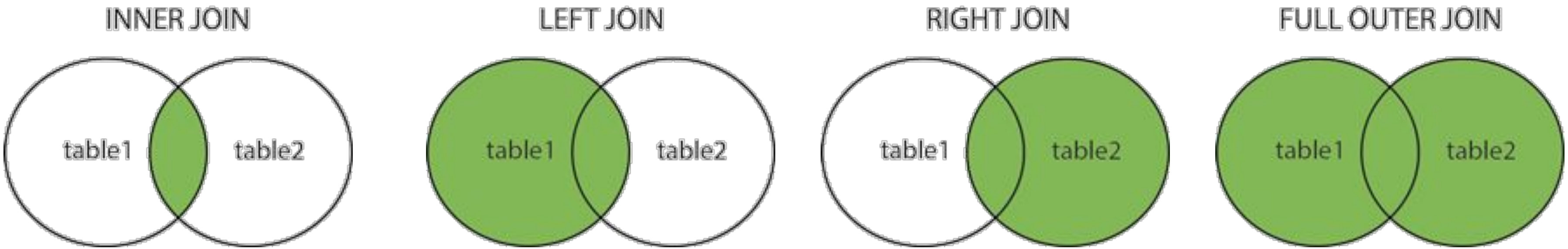
Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Joins

Images from: https://www.w3schools.com/sql/sql_join.asp

Table_1
<TYPE> JOIN Table_2 ON Table_1.column=Table2.column;



Joins

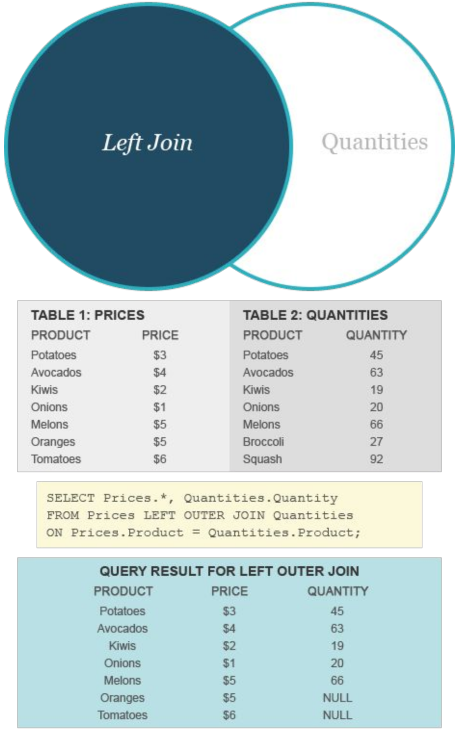


TABLE 1: PRICES		TABLE 2: QUANTITIES	
PRODUCT	PRICE	PRODUCT	QUANTITY
Potatoes	$3	Potatoes	45
Avocados	$4	Avocados	63
Kiwis	$2	Kiwis	19
Onions	$1	Onions	20
Melons	$5	Melons	66
Oranges	$5	Broccoli	27
Tomatoes	$6	Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices LEFT OUTER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR LEFT OUTER JOIN

PRODUCT	PRICE	QUANTITY
Potatoes	$3	45
Avocados	$4	63
Kiwis	$2	19
Onions	$1	20
Melons	$5	66
Oranges	$5	NULL
Tomatoes	$6	NULL

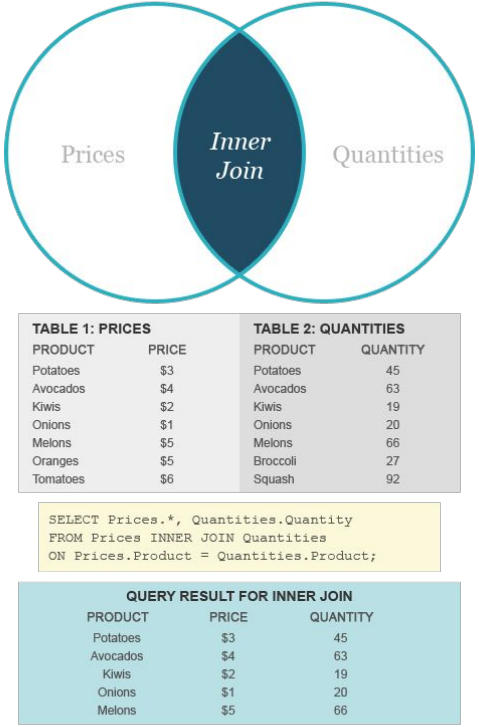


TABLE 1: PRICES		TABLE 2: QUANTITIES	
PRODUCT	PRICE	PRODUCT	QUANTITY
Potatoes	$3	Potatoes	45
Avocados	$4	Avocados	63
Kiwis	$2	Kiwis	19
Onions	$1	Onions	20
Melons	$5	Melons	66
Oranges	$5	Broccoli	27
Tomatoes	$6	Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices INNER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR INNER JOIN

PRODUCT	PRICE	QUANTITY
Potatoes	$3	45
Avocados	$4	63
Kiwis	$2	19
Onions	$1	20
Melons	$5	66

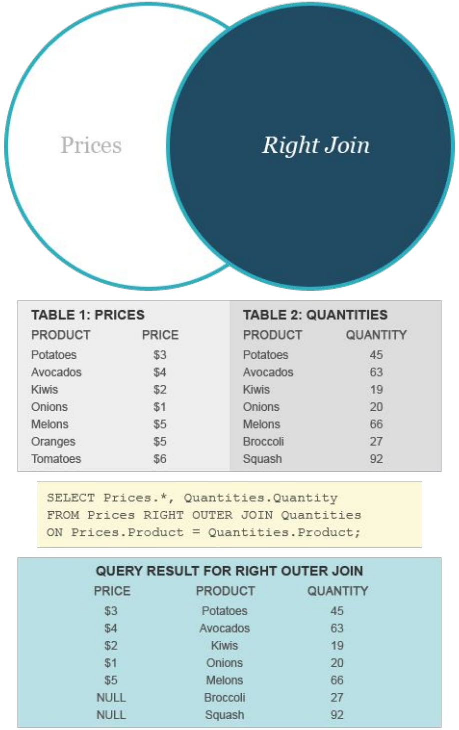


TABLE 1: PRICES		TABLE 2: QUANTITIES	
PRODUCT	PRICE	PRODUCT	QUANTITY
Potatoes	$3	Potatoes	45
Avocados	$4	Avocados	63
Kiwis	$2	Kiwis	19
Onions	$1	Onions	20
Melons	$5	Melons	66
Oranges	$5	Broccoli	27
Tomatoes	$6	Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices RIGHT OUTER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

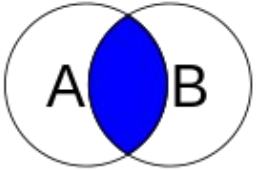
QUERY RESULT FOR RIGHT OUTER JOIN

PRICE	PRODUCT	QUANTITY
$3	Potatoes	45
$4	Avocados	63
$2	Kiwis	19
$1	Onions	20
$5	Melons	66
NULL	Broccoli	27
NULL	Squash	92

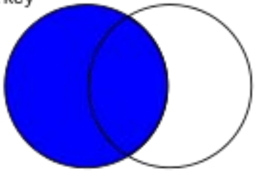
https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

SQL JOINS

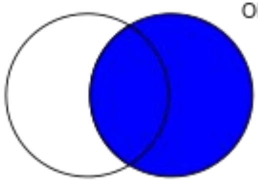
```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

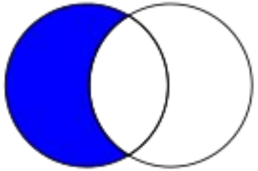


```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```

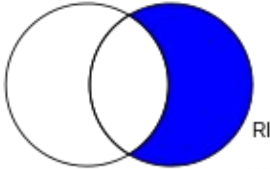


```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

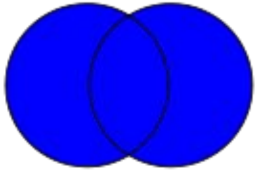




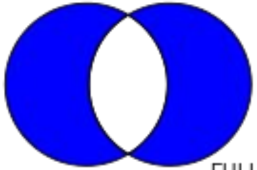
```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```



```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
```



```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```



```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

Group and Aggregate

- Collect data into groups and then perform an operation
- E.g. Get average grade per student
  - Student is the group
  - Average is the operation

Group By Statement

```
SELECT column_1, column_2, ….
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

https://www.w3schools.com/sql/sql_groupby.asp

- Columns in select must be used in GROUP BY
- Alternatively you can choose aggregation operations

Aggregation Operations

- Get average grade per student?

```
SELECT student_id, avg(grade)
FROM students
GROUP BY student_id
```

- (COUNT, MAX, MIN, SUM, AVG)

SQL for data science

- Often we query to get data into one form and then transform in another
- Often use SQL to get data into a “flat” representation
- We will move on to Pandas in Python3 for exploring, analyzing and transforming data

Notebooks

http://jupyter.org/

https://colab.research.google.com

Integrates with Pandas, matplotlib, sklearn, etc.