wbs
WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

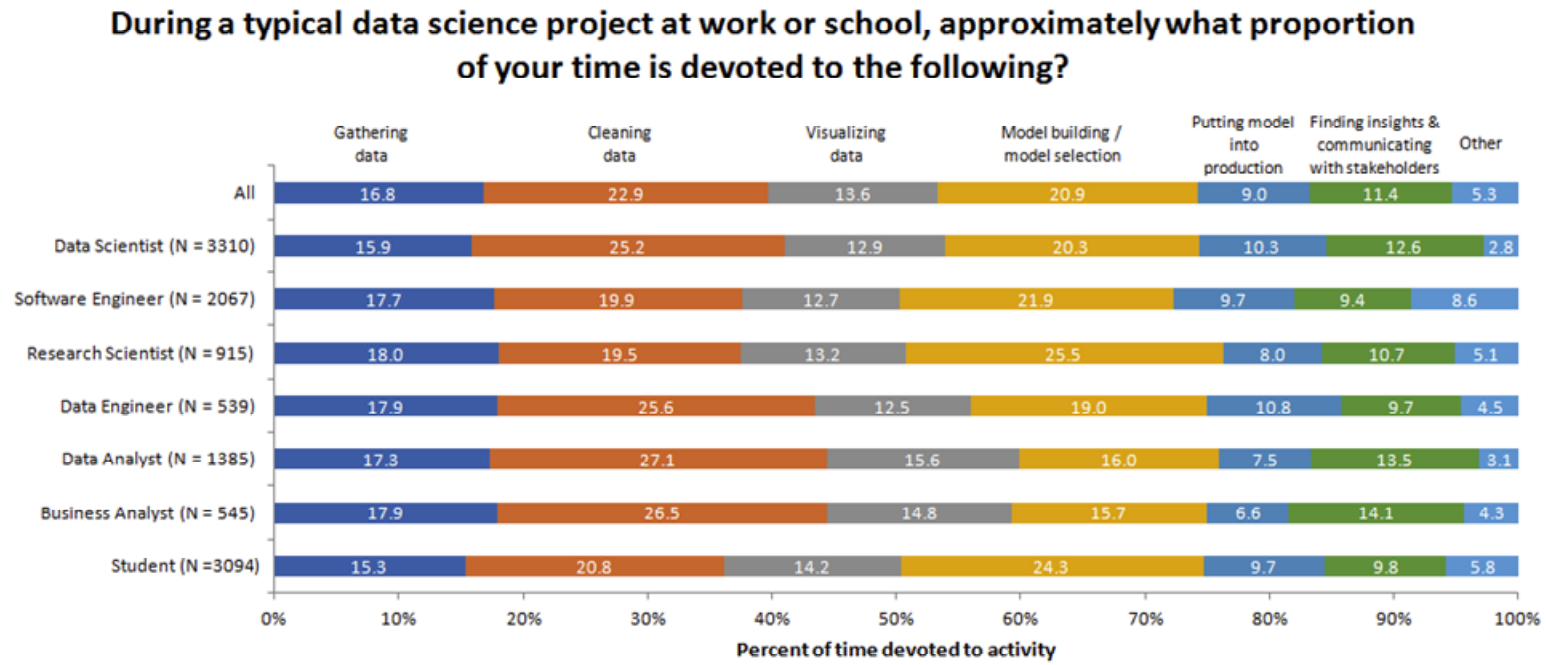
For the Change Makers

Programming for Data Analytics

Week 1: Introduction and Data Collection

Information Systems and Management
Warwick Business School

Data analytics: Things to do and time to spend



Note: Data are from the 2018 Kaggle ML and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018.
A total of 23859 respondents completed the survey; the percentages in the graph are based on a total of 15937 respondents who provided an answer to this question. Only selected job titles are presented.

BUSINESS BROADWAY
DATA SCIENCE | CUSTOMER ANALYTICS | MACHINE LEARNING

Copyright 2019 Business Over Broadway

What is data?

- Collection of **data objects** and their **attributes**
- An attribute is a *property* or *characteristic* of an object
  - Examples: eye colour of a person, temperature, etc.
  - Attribute is also known as *variable, field, characteristic, or feature.*
- A collection of attributes describe an object
  - Object is also known as *record, point, case, sample, entity, or instance*

Attributes

Objects

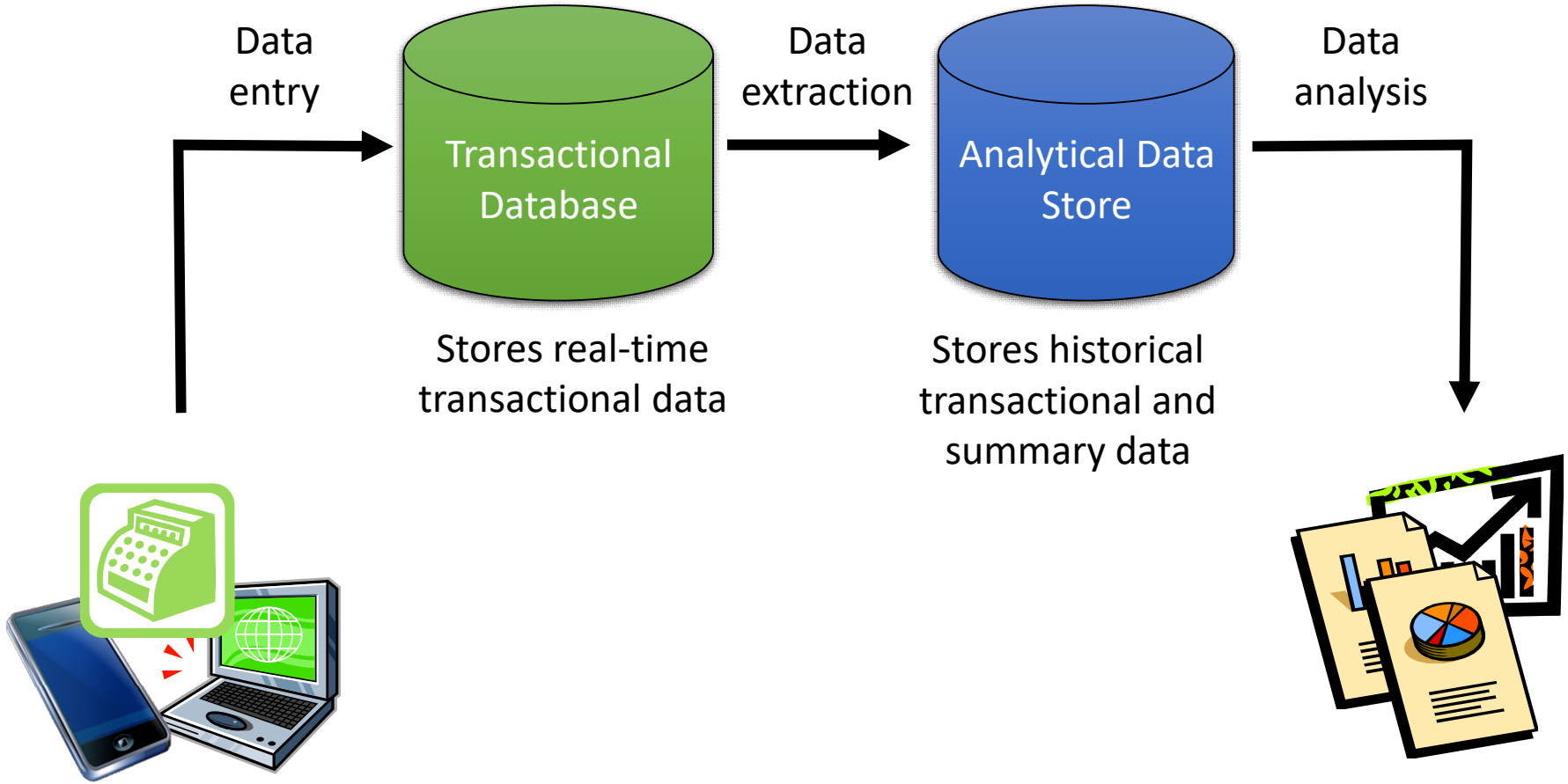
rid	user_name	user_numberoffriends	user_numberofreviews	rating
1	Daphne G.	0	6	5
1	Sarah H.	276	15	5
1	Khalia F.	0	12	5
1	Foodie M.	3	128	4
1	Vee N.	19	100	4
1	Eli N.	7	111	4
1	Deanna D.	29	58	5
1	Quijana B.	361	436	4
1	TK B.	48	122	3

Collecting publicly available data

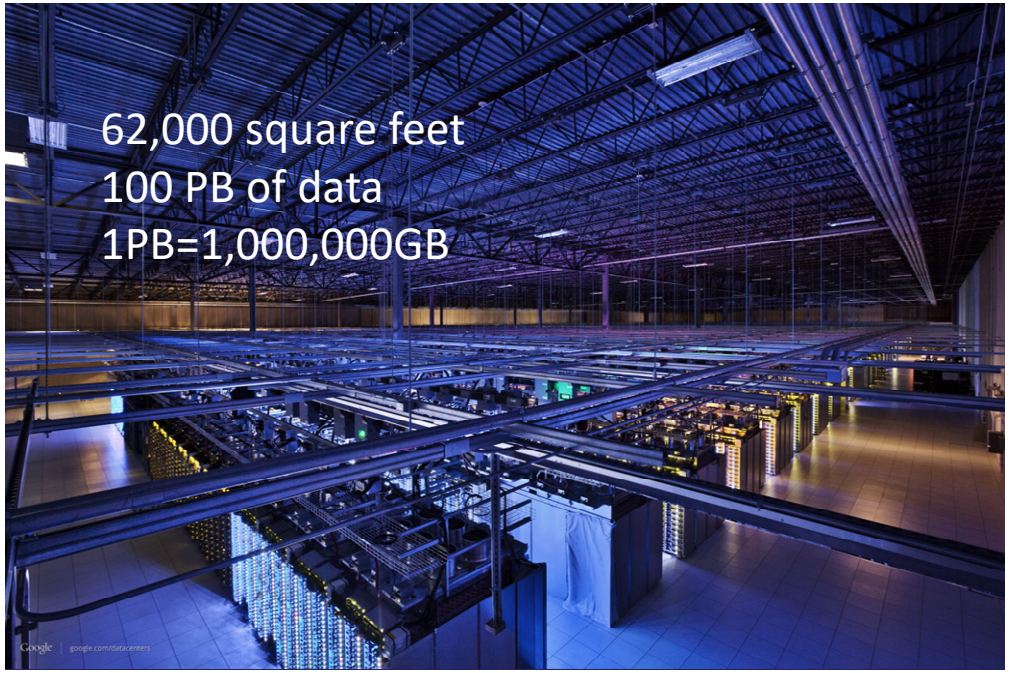
- Increasing number of digital trace data that are publicly available online.
- Three major approaches to collect those data:
    1. Data repositories (public and private)
    2. Web scraping (standard and non-standard)
    3. Platform APIs (official and non-official)

Access data repository with SQL queries

The Information Architecture of an Organization



The Relational Paradigm



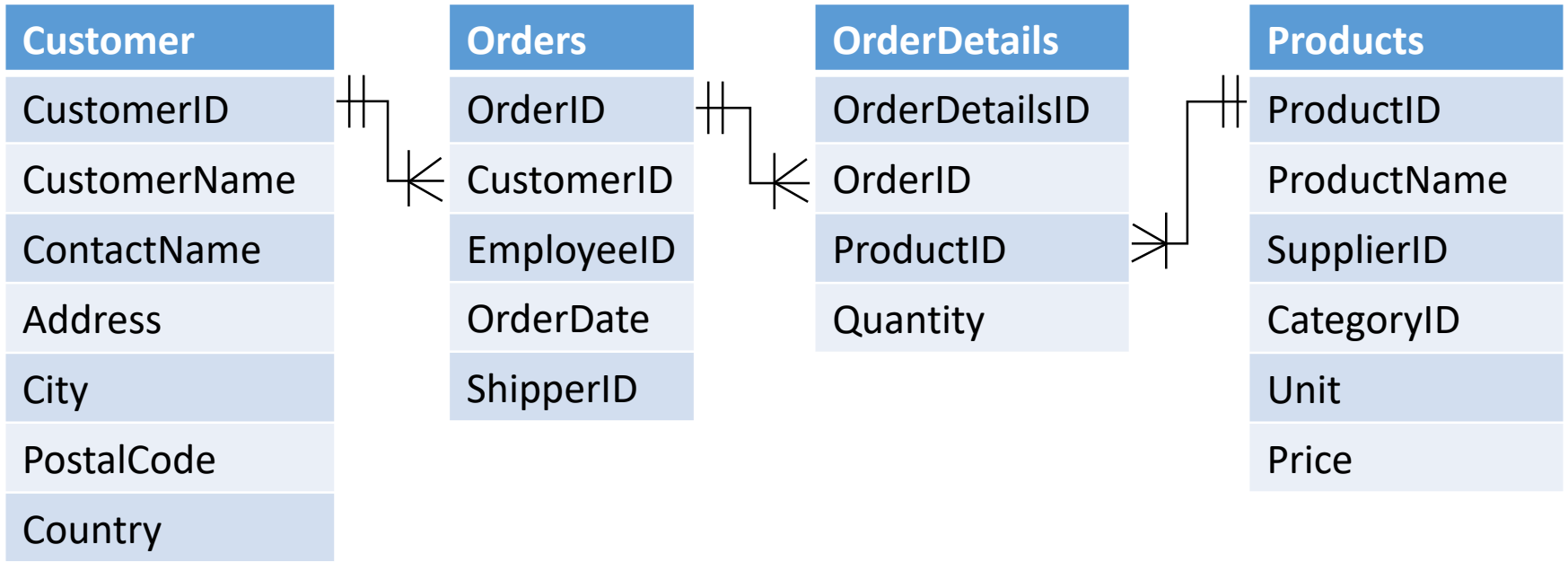
- How data is collected and stored
- Primary Goal: Minimize redundancy
  - Reduce errors
  - Less space required

Which of these do you think is more important today ?

- Most database management systems are based on the relational paradigm
  - Oracle, MySQL, SQL Server

The relational database

- A series of tables
- Linked together through primary/foreign key relationships



Why more than one table?

- Split the details off into separate tables
- Information is entered and stored once
- Minimizes redundancy

Customer	Orders	OrderDetails	Products
CustomerID	OrderID	OrderDetailsID	ProductID
CustomerName	CustomerID	OrderID	ProductName
ContactName	EmployeeID	ProductID	SupplierID
Address	OrderDate	Quantity	CategoryID
City	ShipperID		Unit
PostalCode			Price
Country			

How data is organized: key terms

- **Table**: a list of data on a(n) (business) entity
- **Column** (or **field**): defines key attributes of the kind of data held in a table
- **Row** (or **record**): a single instance of whatever data a particular table holds
- **Database**: a single table or a collection of tables
  - **Relational database**: multiple tables related (or linked) based on primary and foreign key relationships

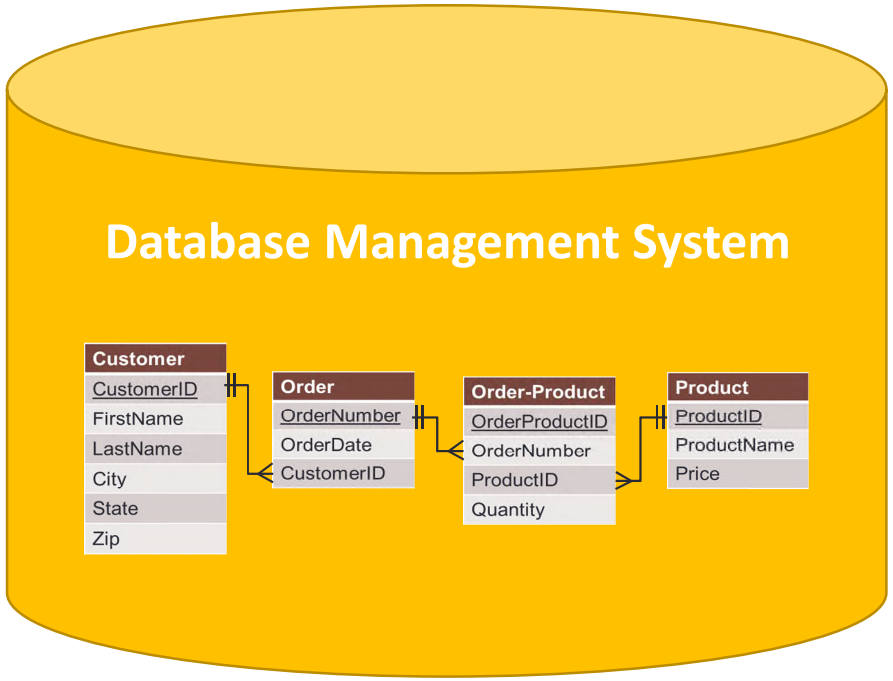
How data is organized: key terms

- **Primary Key:** an attribute that can uniquely identify a record.
- **Foreign key:** an attribute to specify the dependency between two tables, linking two tables via primary-foreign key relationship.

What do we want to do?

Get information out of the database (retrieve)

Put information into the database (change)



To do this we use SQL

- Structured Query Language
- A high-level set of commands that let you communicate with the database
- With SQL, you can
  - **Retrieve records**
  - **Join (combine) tables**
  - Insert records
  - Delete records
  - Update records
  - Add and delete tables

We will be doing this.

A **statement** is any SQL command that interacts with a database.

A SQL statement that **retrieves** information is referred to as a **query**.

Some points about SQL

It’s not a true programming language

- It is used by programming languages to interact with databases

There is no standard syntax

- MySQL, Oracle, SQL Server, and BigQuery all have slight differences

There are a lot of statements and variations among them

- We will be covering the basics, and the most important ones

SELECT statement

https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_in

SELECT column_name(s) FROM *schema_name.*table_name

Example:

Customer

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

SELECT postalcode FROM Customers

postalcode
12209
05021
05023
WA1 1DP

This returns the PostalCode column for every row in the Customers table.
Called a "View."

Retrieving multiple columns

SELECT CustomerID, PostalCode FROM Customers

CustomerID	PostalCode
1	12209
2	05021
3	05023
4	WA1 1DP

SELECT * FROM Customer

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mex
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mex
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

The * is called a **wildcard**.
It means "return every column."

Counting records

SELECT COUNT(CustomerName) FROM Customers

91

Total number of records in the table
where the field is not empty.
(don't forget the parentheses!)

SELECT COUNT(CustomerID) FROM Customers

91

Why is this the same number as the previous query?

SELECT COUNT(*) FROM Customers

?

What number would be returned?

Fancier counting of records

Asks: How many customers from each country are there in the Customers table?

```
SELECT Country, COUNT(CustomerID)
FROM Customers
GROUP BY Country
```

Country	Value
Argentina	3
Austria	2
…	…

GROUP BY organizes the results by specified column values.

So it looks for unique Country values and then counts the number of records for each of those values.

Counting and sorting

You may want to have your result sorted:

```
SELECT Country, COUNT(CustomerID)
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC
```

Country	Value
USA	13
France	11
…	…

GROUP BY organizes the results by column values.

ORDER BY sorts results from lowest to highest based on a field (in this case, COUNT(CustomerID))

Nested statements

- We want to get a COUNT of how many DISTINCT states there are in the table

```
SELECT COUNT(*)
FROM (SELECT DISTINCT Country FROM Customers) AS tmp1
```

- To see how this works:
  - Start with the SELECT DISTINCT…

Country
Argentina
Austria
…

  - …then COUNT those values

21

But wait a minute…

SELECT COUNT(*) FROM (SELECT DISTINCT Country FROM Customer) AS tmp1

- We see this works, but

  SELECT DISTINCT Country FROM Customers

  isn't a table – it's a **view**!

- You also can SELECT data FROM a view
  - Look back at the query results – they are all basically tables anyway!
  - *But you may have to give it a name (i.e., tmp1)*

Functions: Retrieving highest, lowest, average, and sum

SELECT MAX(Price) FROM Products

Price
263.5

SELECT MIN(Price) FROM Products

Price
2.5

SELECT AVG(Price) FROM Products

Price
28.8664

SELECT SUM(Price) FROM Products

Price
2222.71

Returning only certain records

- We don’t always want every record in the table

Customer

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

Let’s retrieve only those customers who live in Denmark.

use: SELECT * FROM schema_name.table_name WHERE *condition*

so SELECT * FROM Customers WHERE Country= 'denmark'

returns this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
73	Simons bistro	Jytte Petersen	Vinbæltet 34	København	1734	Denmark
83	Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus	8200	Denmark

More conditional statements

SELECT * FROM Customers WHERE Country <> 'denmark'

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
…	…	…	…	…	…	…

> means “greater than”
< means “less than”
= means “equal to”
<> means “not equal to”

SELECT ProductName, Price FROM Products WHERE Price > 100

ProductName	Price
Thüringer Rostbratwurst	123.79
Côte de Blaye	263.5

String condition

- In addition to =,<,>,<> comparison, you can also use LIKE and *CONTAINS* to set string conditions.

SELECT * FROM Customers WHERE City LIKE '%Lon%'

% The percent sign represents zero, one, or multiple characters

_ The underscore represents a single character

*SELECT * FROM Customers WHERE City CONTAINS 'Lon'*

Multiple conditions

- You may combine multiple conditions in SELECT statement using AND or OR.

SELECT * FROM schema_name.table_name

WHERE *condition1* AND/OR *condition2;*

AND means BOTH condition 1 AND condition 2 need be satisfied.

OR means EITHER condition 1 OR condition 2 needs be satisfied.

Querying multiple tables

- Right now, you can answer
  - How many customers live in the UK?
  - What is the most expensive product sold?

- Because those two questions can be answered by looking at only a single table.

- But what if we want to find out the orders a customer placed?

- You need a construct a query that combines two (or more) tables.

The (Inner) Join

- We often need to retrieve data from multiple tables.

OrderID	OrderDate	CustomerID	Customer ID	CustomerName	City	Country	PostalCode
101	2011-3-2	1001	1001	Maria Anders	Berlin	Germany	12209
102	2011-3-3	1002	1002	Ana Trujillo	Mexico D.F.	Mexico	05021
103	2011-3-4	1001	1001	Antoni Moreno	Mexico D.F.	Mexico	05023
104	2011-3-6	1004	1004	Thomas Hardy	London	UK	WA1 1DP

- We matched the Order and Customer tables based on the common field (CustomerID)
- We can construct a SQL query to do this

Joining tables using WHERE

SELECT columns FROM tableA, tableB

WHERE tableA.key1 = tableB. key2

SELECT * FROM Customers, Orders

WHERE Customers.CustomerID=Orders.CustomerID

Returns this:

Customers.CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country	OrderID	Orders.CustomerID	EmployeeID	OrderDate	ShipperID
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico	10308	2	7	9/18/1996	3
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico	10365	3	3	11/27/1996	2
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	10383	4	8	12/16/1996	3
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	10355	4	6	11/15/1996	1

Joining tables using JOIN…ON

SELECT columns FROM tableA

(INNER) JOIN tableB ON tableA.key1 = tableB.key2

SELECT * FROM Customers

(INNER) JOIN Orders ON Customers.CustomerID=Orders.CustomerID

Returns this:

Customers.CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country	OrderID	Orders.CustomerID	EmployeeID	OrderDate	ShipperID
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico	10308	2	7	9/18/1996	3
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico	10365	3	3	11/27/1996	2
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	10383	4	8	12/16/1996	3
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	10355	4	6	11/15/1996	1

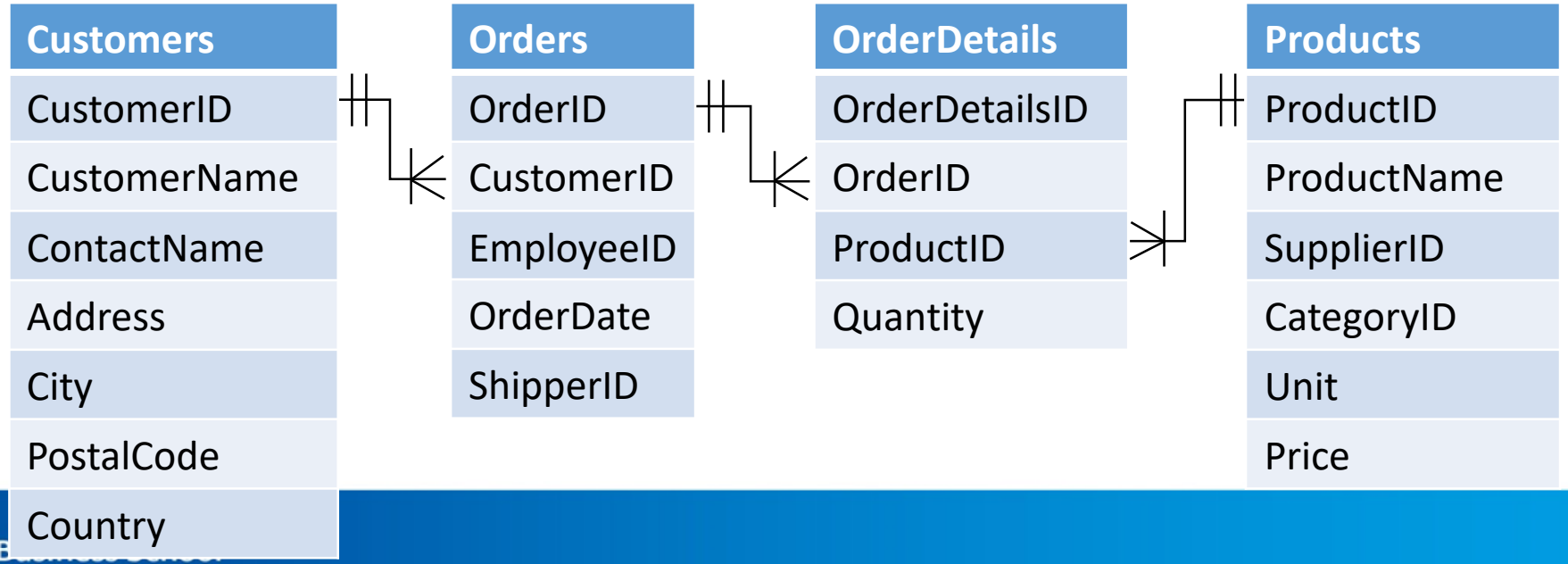
A more complex join

- Let’s say we want to find out what each customer ordered
- We want to wind up with this view of the database

OrderID	CustomerName	ProductName	Quantity	Price
10248	Wilman Kala	Mozzarella di Giovanni	5	34.8
10248	Wilman Kala	Queso Cabrales	12	21
10248	Wilman Kala	Singaporean Hokkien Fried Mee	10	14
10249	Tradição Hipermercados	Manjimup Dried Apples	40	53
10249	Tradição Hipermercados	Tofu	9	23.25
10250	Hanari Carnes	Louisiana Fiery Hot Pepper Sauce	15	21.05

How to do it?

- We need information from Customers and Products
- So we need to link all of the tables together
  - To associate Customers with Products we need to follow the path from **Customers** to **Products**

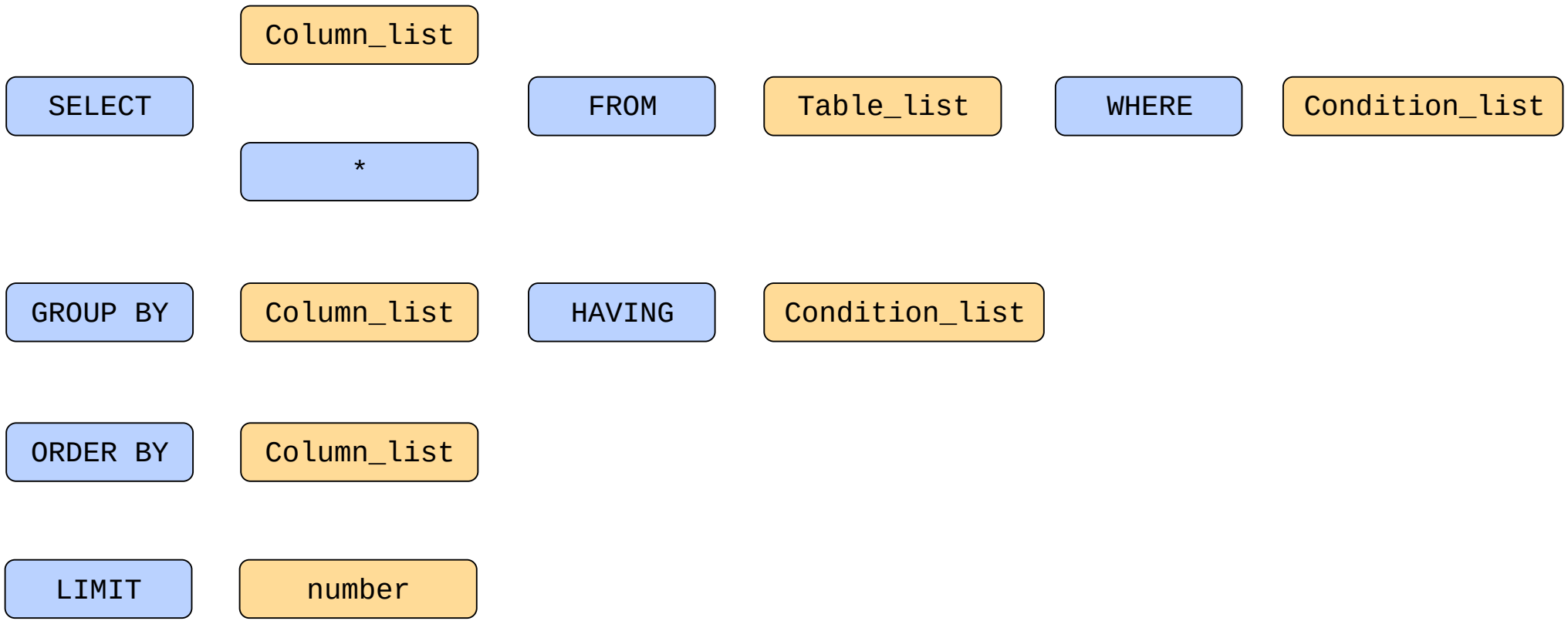


Here's the query

```
SELECT Orders.OrderID, Customers.CustomerName,
Products.ProductName, OrderDetails.Quantity,
Products.Price

FROM Customers, Orders, Products, OrderDetails

WHERE Customers.CustomerID=Orders.CustomerID

AND Orders.OrderID=OrderDetails.OrderID

AND Products.ProductID=OrderDetails.ProductID
```

It looks more complicated than it is!
Note that we have three conditions in the WHERE clause, and we have three relationships in our schema.

Structure of SQL statements



Further readings and exercises

- Please follow the instruction posted along with this video to read the required book chapters and work out the exercises.

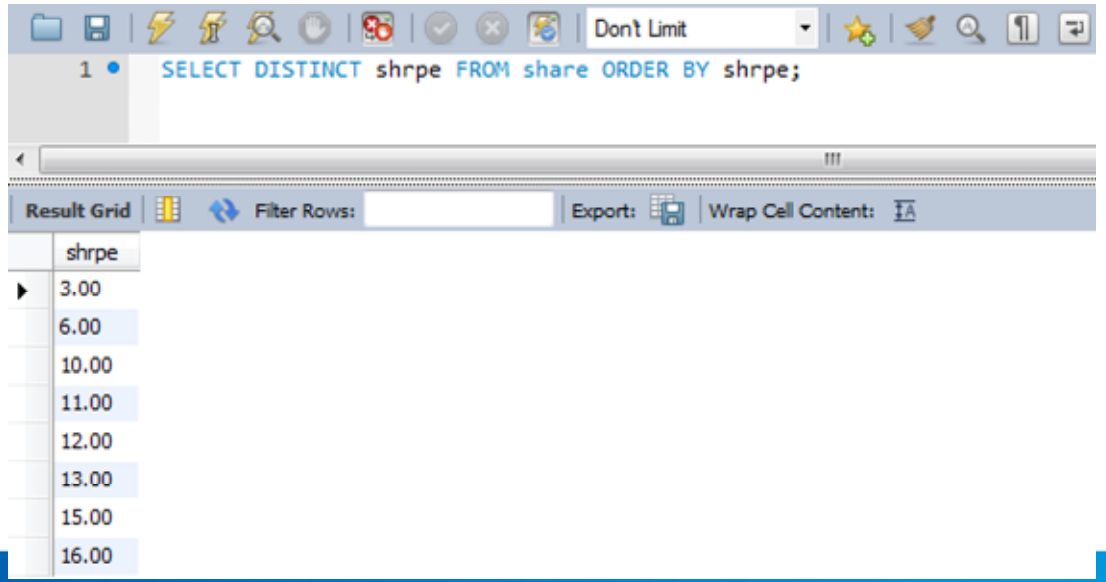
Exercise

- Please write SQL queries to answer the following questions based the dataset "bigquery-public-data.Austin_bikeshare".

1. Find the top 10 most used bikes based on the frequency.
2. Find the top 10 most used bikes based on total trip durations.
3. Find and rank the average trip durations for each subscriber type.
4. Find the top 5 most popular destination stations for single trip subscriber.
5. Find the top 5 most popular destination stations for annual subscriber that are now closed.

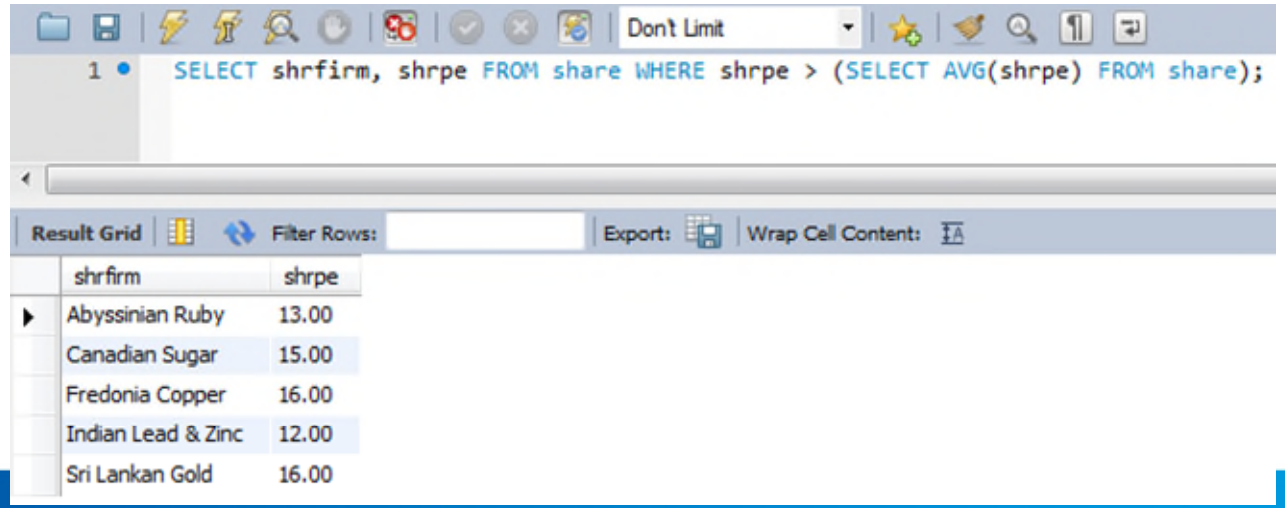
DISTINCT

- Eliminate duplicate rows when reporting i.e., report the different values of the PE ratio.
  - `SELECT DISTINCT shrpe FROM share ORDER BY shrpe;`



Subqueries

- A query within a query i.e., report all firms with a PE ratio greater than the average for the portfolio.
  - `SELECT shrfirm, shrpe FROM share WHERE shrpe > (SELECT AVG(shrpe) FROM share);`



Lists: the IN operator

- Used with a list of values i.e., report data on firms with share codes of FC, AR, or SLG.
  - `SELECT * FROM share WHERE shrcode IN ('FC','AR','SLG');`

is equivalent to…

  - `SELECT * FROM share WHERE shrcode = 'FC' OR shrcode = 'AR' OR shrcode = 'SLG';`

Lists: the NOT IN operator

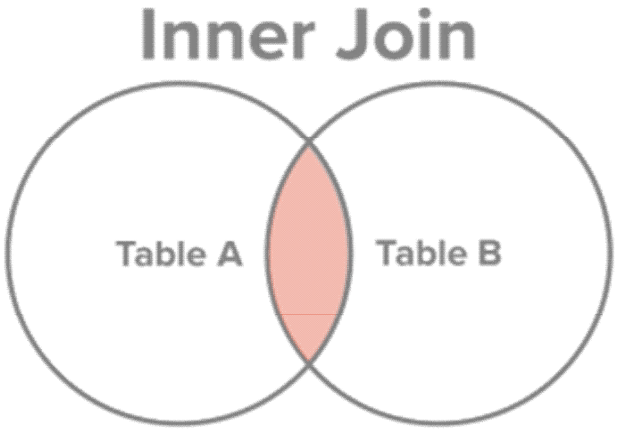
- Not in a list of values i.e., report all firms other than those with the code CS or PT.
  - `SELECT * FROM share WHERE shrcode NOT IN ('CS', 'PT');`

    is equivalent to…

  - `SELECT * FROM share WHERE shrcode <> 'CS' AND shrcode <> 'PT';`

Inner Join

[INNER] JOIN: Returns records that have matching values in both tables.




```
SELECT columns
FROM tableA
INNER JOIN tableB
ON tableA.key1 = tableB.key2
```

	shrcode	shrfirm	shrprice	shrqty	shrdiv	shrpe	shrnat
▶	AR	Abyssinian Ruby	31.82	22010	1.32	13.00	NULL
	BE	Burmese Elephant	0.07	154713	0.01	3.00	UK
	BS	Bolivian Sheep	12.75	231678	1.78	11.00	US
	CS	Canadian Sugar	52.78	4716	2.50	15.00	NULL
	FC	Freedonia Copper	27.50	10529	1.84	16.00	US
	ILZ	Indian Lead & Zinc	37.75	6390	3.00	12.00	UK
	NG	Nigerian Geese	35.00	12323	1.68	10.00	NULL
	PT	Patagonian Tea	55.25	12635	2.50	10.00	US
	ROF	Royal Ostrich Farms	33.75	1234923	3.00	6.00	UK
	SLG	Sri Lankan Gold	50.37	32868	2.68	16.00	UK

	natcode	natname	natexrate
▶	CA	Canada	1.64
	FR	France	1.18
	UK	United Kingdom	1.00
	US	United States	1.25

Example

shrcode	shrfirm	shrprice	shrqty	shrdiv	shrpe	shrnat
AR	Abyssinian Ruby	31.82	22010	1.32	13.00	NULL
BE	Burmese Elephant	0.07	154713	0.01	3.00	UK
BS	Bolivian Sheep	12.75	231678	1.78	11.00	US
CS	Canadian Sugar	52.78	4716	2.50	15.00	NULL
FC	Freedonia Copper	27.50	10529	1.84	16.00	US
ILZ	Indian Lead & Zinc	37.75	6390	3.00	12.00	UK
NG	Nigerian Geese	35.00	12323	1.68	10.00	NULL
PT	Patagonian Tea	55.25	12635	2.50	10.00	US
ROF	Royal Ostrich Farms	33.75	1234923	3.00	6.00	UK
SLG	Sri Lankan Gold	50.37	32868	2.68	16.00	UK

natcode	natname	natexrate
CA	Canada	1.64
FR	France	1.18
UK	United Kingdom	1.00
US	United States	1.25

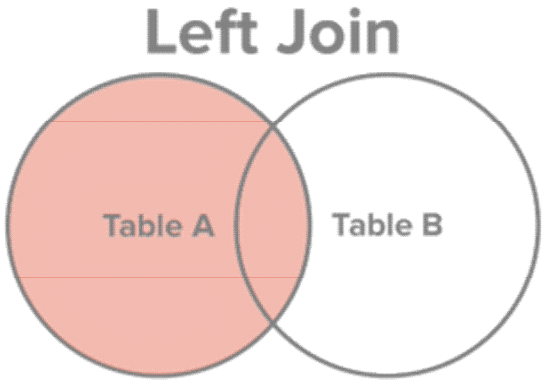
```
SELECT shrfirm, shrprice, natname
FROM share, nation
WHERE share.shrnat = nation.natcode
```

shrfirm	shrprice	natname
Burmese Elephant	0.07	United Kingdom
Indian Lead & Zinc	37.75	United Kingdom
Royal Ostrich Farms	33.75	United Kingdom
Sri Lankan Gold	50.37	United Kingdom
Bolivian Sheep	12.75	United States
Freedonia Copper	27.50	United States
Patagonian Tea	55.25	United States

```
SELECT shrfirm, shrprice, natname
FROM share
JOIN nation
ON share.shrnat = nation.natcode
```

Left Join

LEFT JOIN: Return all records from the left table, and the matched records from the right table.




```
SELECT columns
FROM tableA
LEFT JOIN tableB
ON tableA.key1 = tableB.key2
```

Example

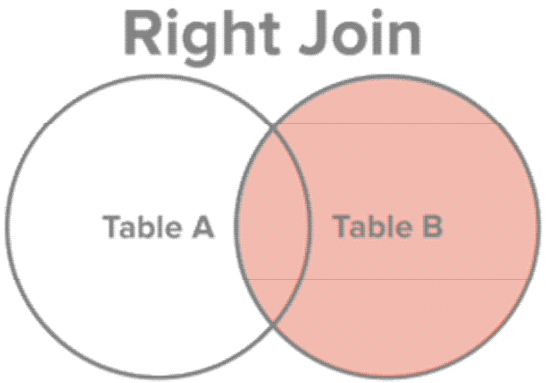
```
SELECT shrfirm, shrprice, natname
FROM share
LEFT JOIN nation
ON share.shrnat = nation.natcode
```

If a record in table A do not have matching data in table b, then it will return data from table A with corresponding columns from table B showing NULL.

shrfirm	shrprice	natname
Abyssinian Ruby	31.82	NULL
Burmese Elephant	0.07	United Kingdom
Bolivian Sheep	12.75	United States
Canadian Sugar	52.78	NULL
Freedonia Copper	27.50	United States
Indian Lead & Zinc	37.75	United Kingdom
Nigerian Geese	35.00	NULL
Patagonian Tea	55.25	United States
Royal Ostrich Farms	33.75	United Kingdom
Sri Lankan Gold	50.37	United Kingdom

Right Join

RIGHT JOIN: Return all records from the right table, and the matched records from the left table.




```
SELECT columns
FROM tableA
RIGHT JOIN tableB
ON tableA.key1 = tableB.key2
```

Example

```
SELECT shrfirm, shrprice, natname
FROM share
RIGHT JOIN nation
ON share.shrnat = nation.natcode
```

If a record in table B do not have matching data in table b, then it will return data from table B with corresponding columns from table A showing NULL.

shrfirm	shrprice	natname
NULL	NULL	Canada
NULL	NULL	France
Burmese Elephant	0.07	United Kingdom
Indian Lead & Zinc	37.75	United Kingdom
Royal Ostrich Farms	33.75	United Kingdom
Sri Lankan Gold	50.37	United Kingdom
Bolivian Sheep	12.75	United States
Freedonia Copper	27.50	United States
Patagonian Tea	55.25	United States