GENERAL ASSEMBLY

CREATING A LOCAL DATABASE

POSTGRES AND PGADMIN INSTALL

BRIEF OVERVIEW DATA MODELING

Matthew Morris

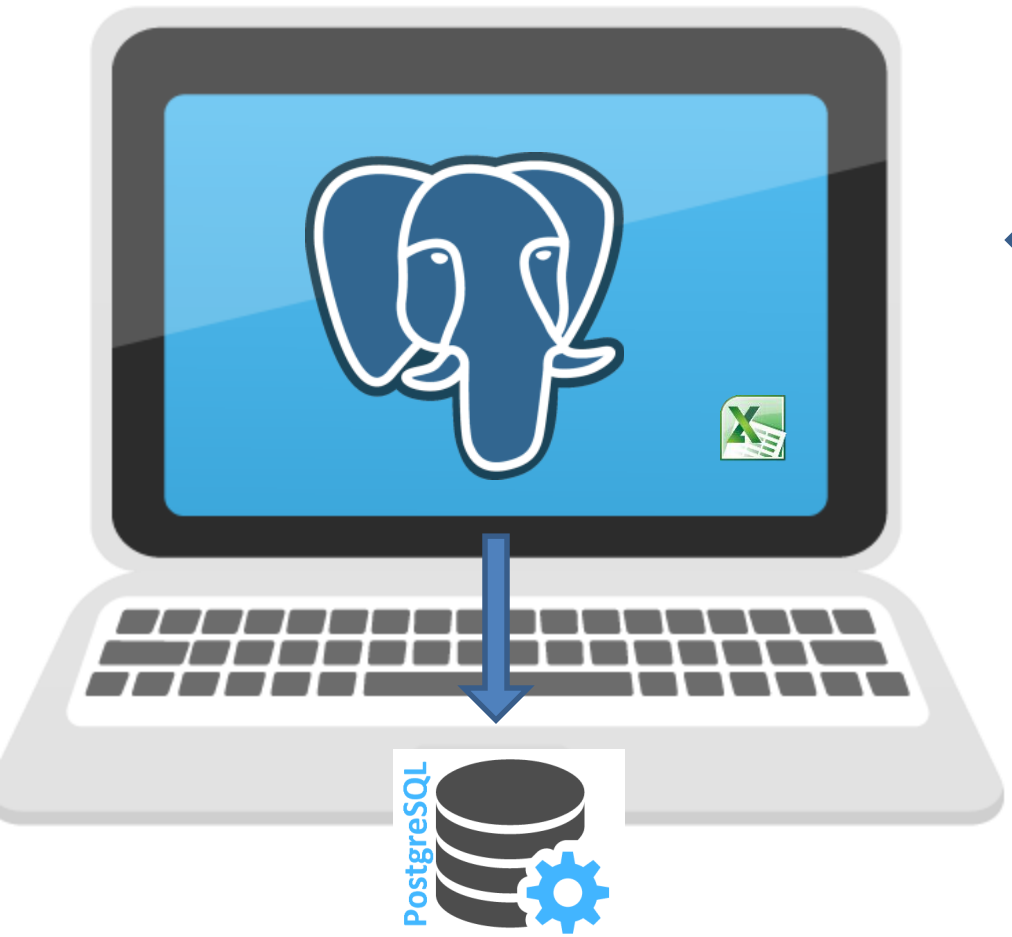
Git: Morrisdata

MatthewMorris.DA@gmail.com

CREATING LOCAL DATABASES

OBTAINING AND MODELING DATA

Obtaining Data



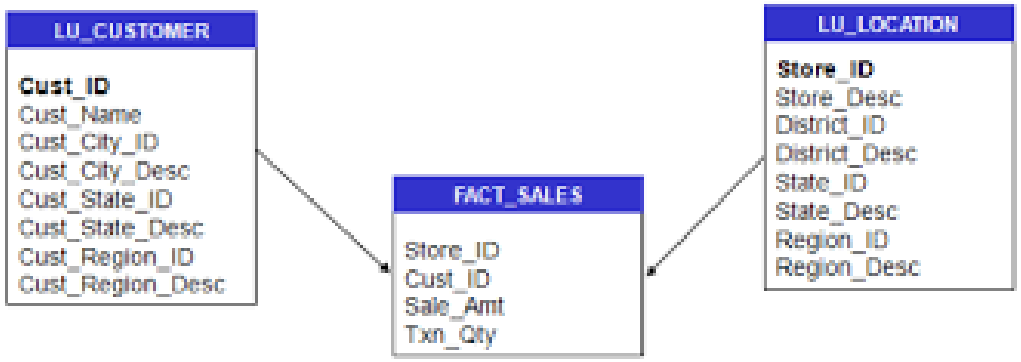
Obtaining Data

Id	Release date	Record Label	Artist	Song	Album	sales
1						
2						
3						
4						
5						

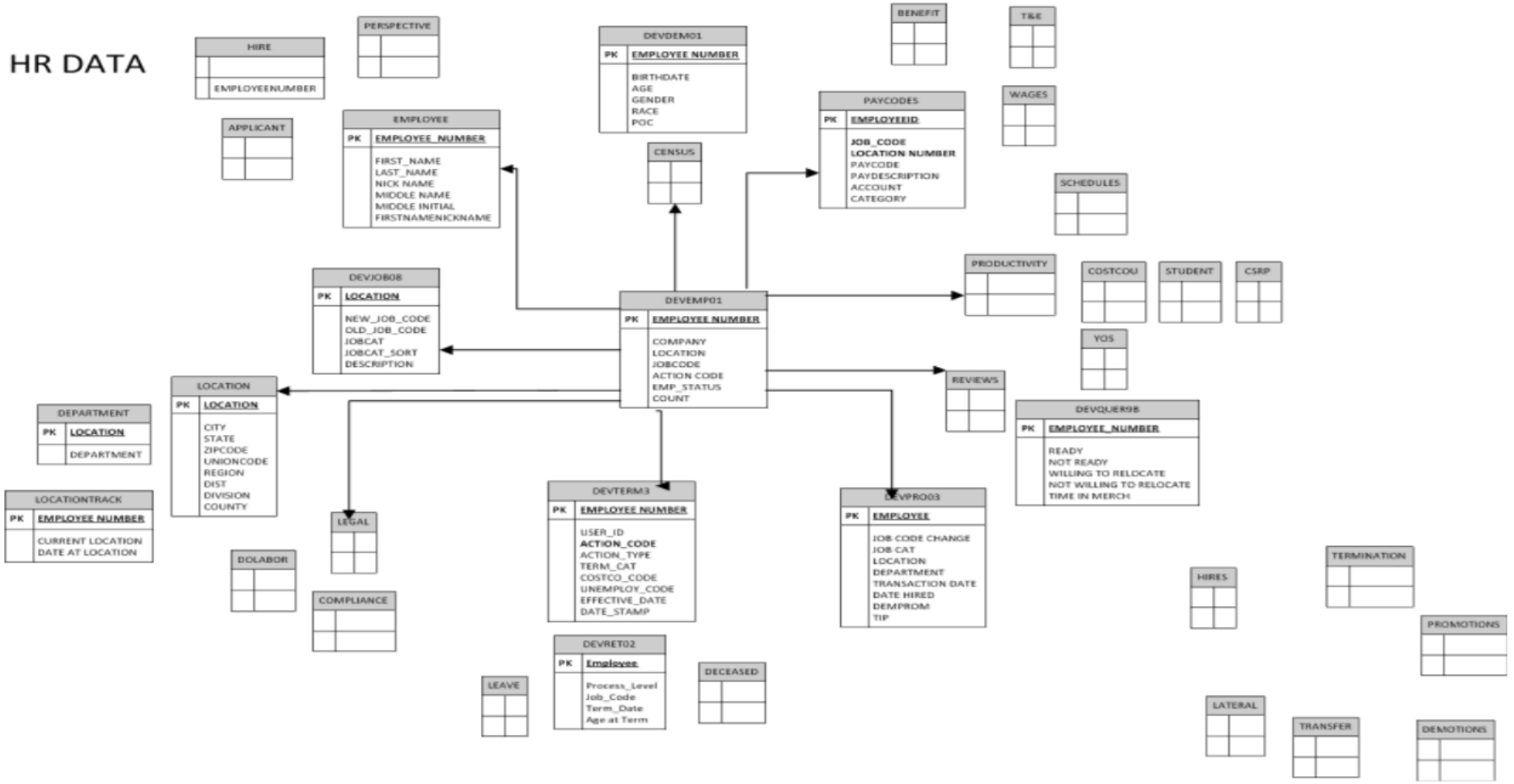
Id	Artist	Song	Album	sales
1				
2				
3				
4				
5				

Obtaining Data

	SALES		
	FIELD	TYPE	LENGTH
	ID	PK	1
	ARTIST	Char	25
	SONG	Char	225
	ALBUM	Char	225



Obtaining Data



Creating Table and loading Data

```
CREATE TABLE
schema.tablename
(
        field1     integer,
        field2     numeric,
        field3     character(30),
       field4      money
);
```

Creating Table and Loading Data

COPY table FROM ‘datasource’
Delimiter ‘,’ filetype Header;

GENERAL ASSEMBLY

FUNDAMENTALS OF DATAFLOW AND SQL

Explain where SQL fits in the dataflow
Retrieve and filter data with basic SQL
Navigate a Relational Database

SELECT
FROM
WHERE
ORDERBY
LIMIT

Matthew Morris
Git: Morrisdata
MatthewMorris.DA@gmail.com

GATHERING REQUIREMENTS

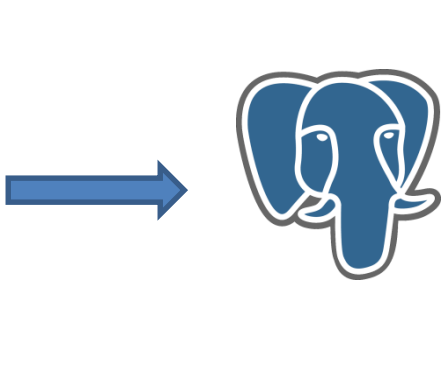
GATHERING REQUIREMENTS

GATHERING REQUIREMENTS

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itoltaw	28

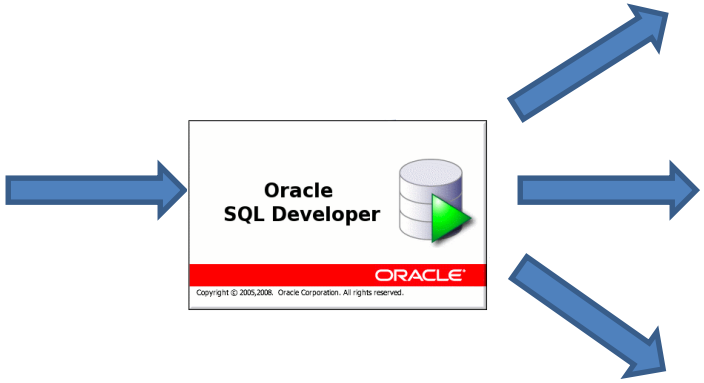
GATHERING REQUIREMENTS

Basic Ecosystem

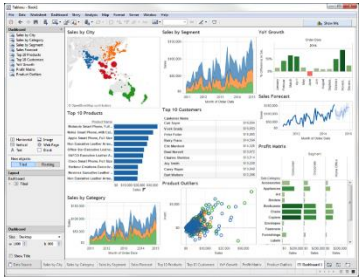


COGNOS

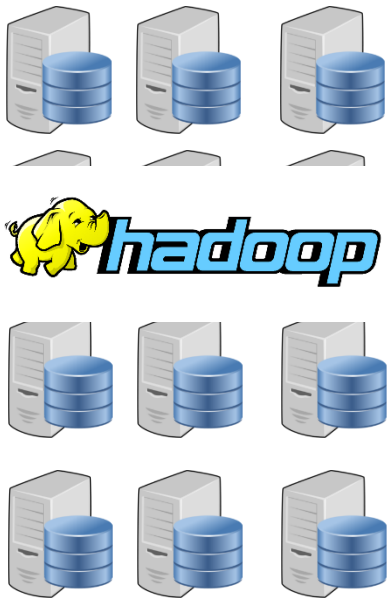
Basic Ecosystem

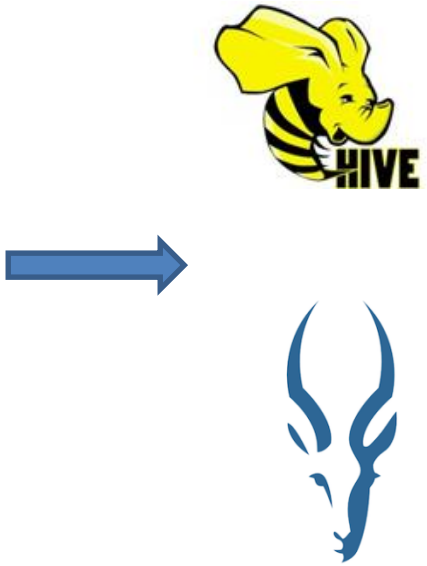


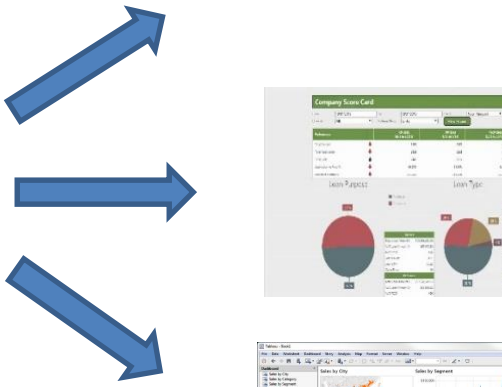
COGNOS

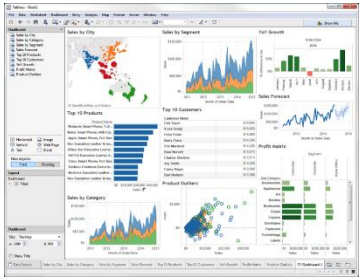


Basic Ecosystem

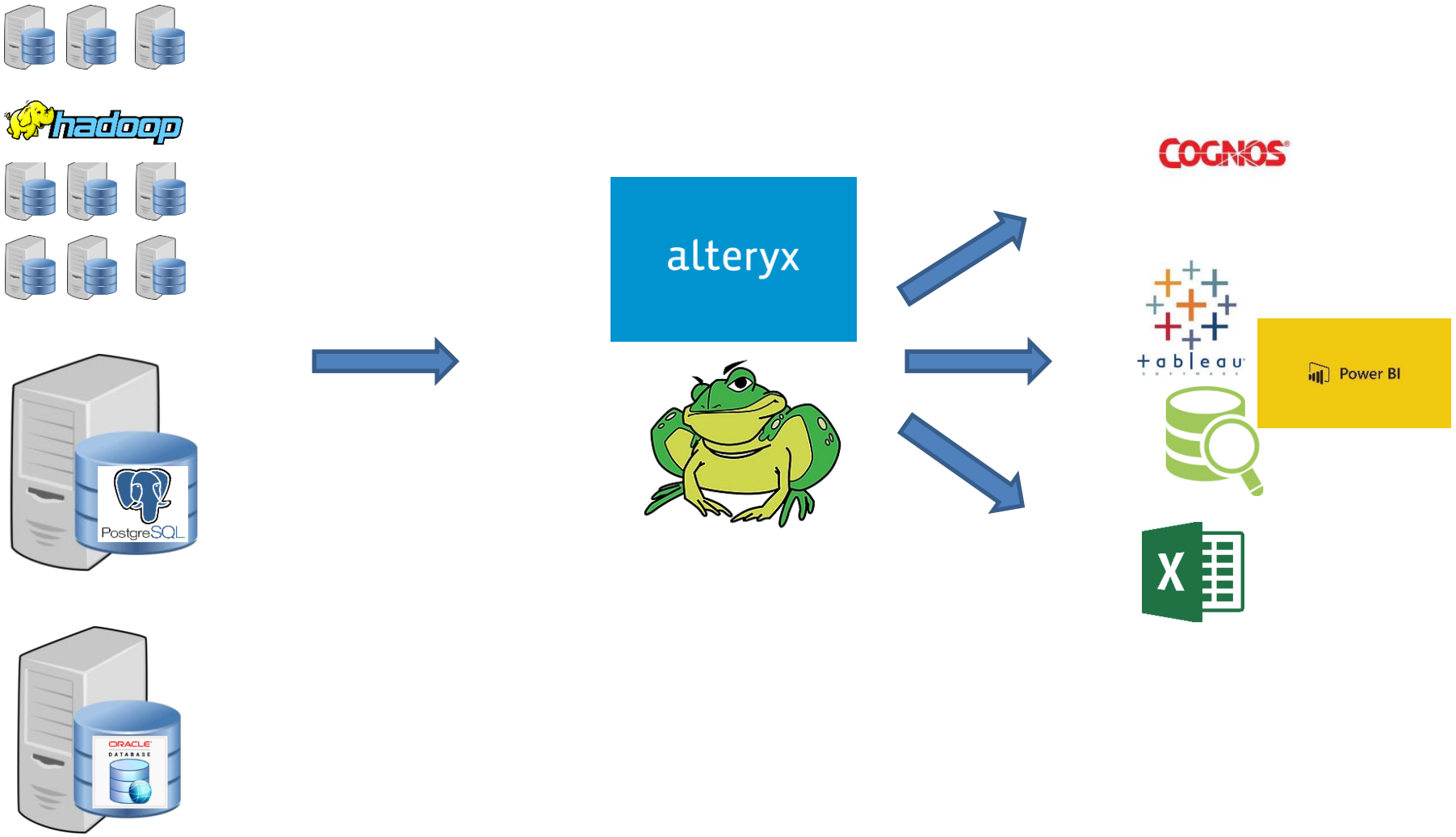




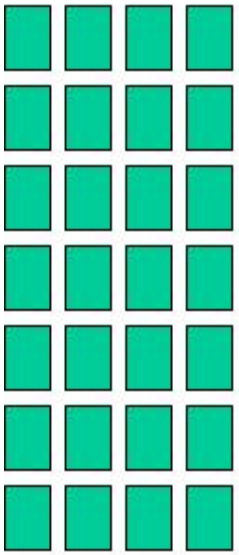




Basic Ecosystem



Relational DB



Libraries/Collections

Schema

Tables/Files/Objects

Members/Partitions in files

Review

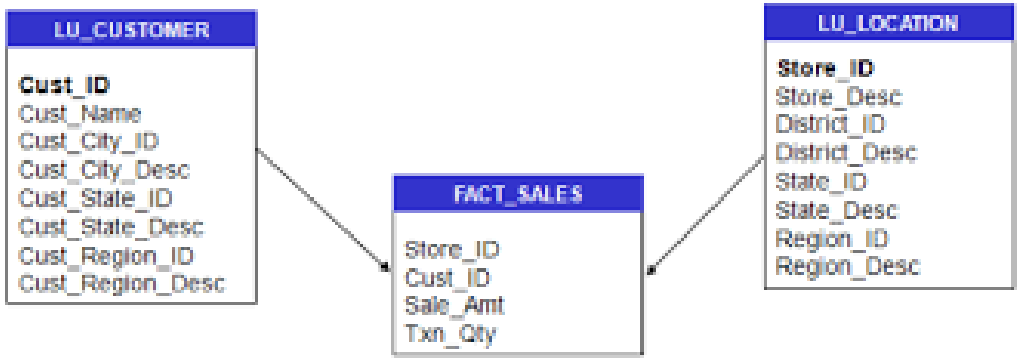
Id	Release date	Record Label	Artist	Song	Album	sales
1						
2						
3						
4						
5						

Id	Artist	Song	Album	sales
1				
2				
3				
4				
5				

Review

SALES

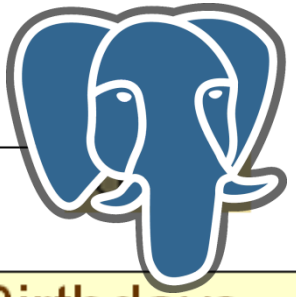
FIELD	TYPE	LENGTH
ID	PK	1
ARTIST	Char	25
SONG	Char	225
ALBUM	Char	225



Relational DB

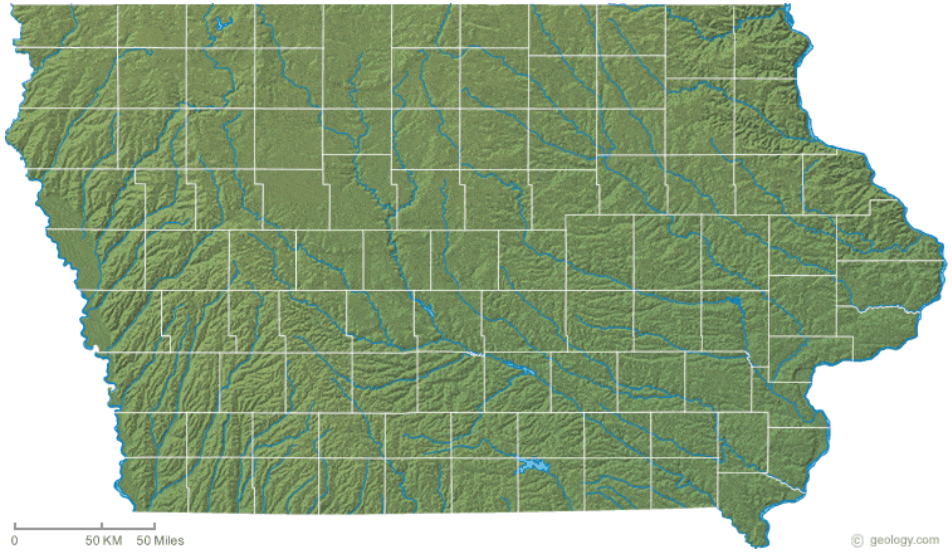
NETWORK

Contacts	Gift Ideas	Party
Name	Name	Name
Friends	Gift ideas	Holidays
Family	Gifts received	Birthday
Business Name	Gifts given	Anniversaries
Address		
Phone number		
Occupation		



Birthdays
Name
Birthday
Gift Ideas
Gifts received
Gifts given
Family

Relational DB



Understanding your data.
What are the tables?
What are the fields?
How might you link the tables?

Basic SQL

SELECT
FROM
WHERE
ORDER BY
LIMIT

Basic SQL

```
SELECT *

SELECT FIELD1, FIELD2 …

SELECT (FIELD1+FIELD2), FIELD 3…

SELECT SUM(FIELD1), FIELD2
```

```
SELECT DISTINCT  Location, NumberOfSales
```

Location	NumberOfSales
Seattle	101
Seattle	40
Tacoma	72

SELECT DISTINCT Location, NumberOfSales, Date

Location	NumberOfSales	Date
Seattle	101	10/28/17
Seattle	101	10/27/17
Seattle	40	10/26/17
Tacoma	72	10/28/17
Tacoma	72	10/27/17

Basic SQL

```
WHERE COUNTRY = US
```

```
WHERE COUNTRY = US
AND STATE = WA
```

```
WHERE COUNTRY = US
AND STATE = WA
AND SALES > 100
```

ORDER BY 1

ORDER BY 1,2 DESC

LIMIT 1000

ROWNUM <= 1000

CAST(field AS type)

A helpful String function

Basic SQL

1. Select various fields from the SALES table that interest you.
   *BE sure to use LIMIT 1000
2. Practice using filters.
3. Use AND to apply multiple filters

Change the sort.

4. Save your Query

Use your new skills to review Iowa Liquor Sales

GENERAL ASSEMBLY

Filters and Aggregations

FILTERS = , !=, >, <
IN, NOT IN, BETWEEN, LIKE, NOT LIKE
SUM, MIN, MAX, COUNT
GROUP BY, HAVING
COMMENTING

Matthew Morris
Git: Morrisdata
MatthewMorris.DA@gmail.com

WHERE

- =, !=, >,<
- IS NULL, IS NOT NULL
- IN, NOT IN
- BETWEEN
- LIKE
- OR

Aggregations

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT

Aggregate functions

- MIN
- MAX
- SUM
- COUNT

Aggregations

GROUP BY store, item

GROUP BY 1,2

```
HAVING AVG(sales)>100
AND COUNT(customers)>20
```

Aggregations

WHERE – filter for dimensions and measures

GROUP BY –groups dimensions when a measure is aggregated

HAVING – filter for aggregated measure

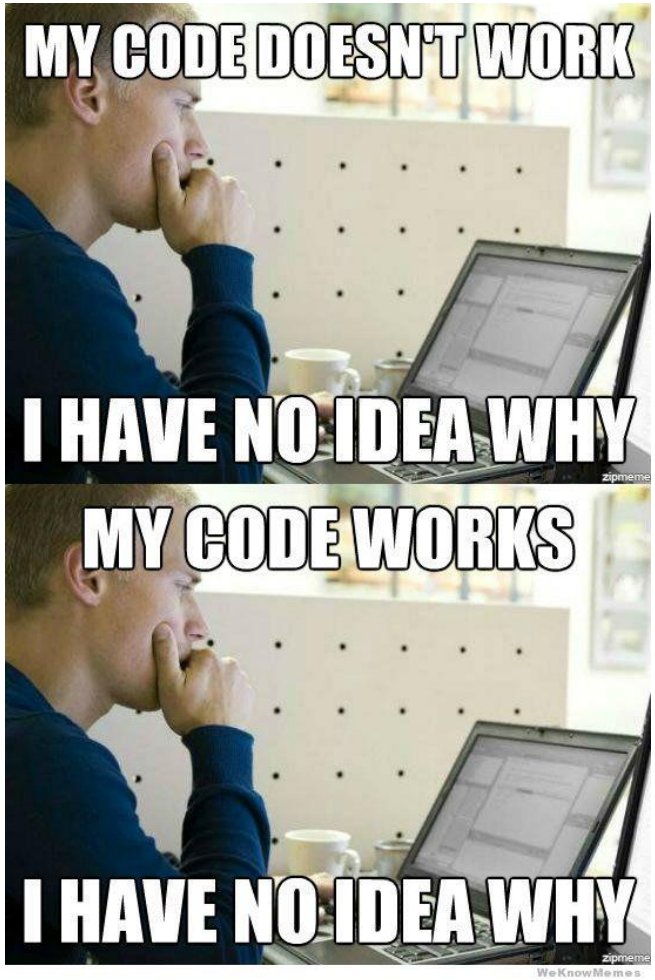
Aggregations

SELECT	- Fields you want to see in your results
FROM	- Table where fields come from
WHERE	- Filters for your results
GROUP BY	- Groups dimensions when using an aggregate
HAVING	- Filters aggregations
ORDER BY	- How you can sort your results
LIMIT	- Limits number of records returned

Fix the Code

```
SELECT Store, (cost –sell price),   SUM(sales),
FROM sales
WHERE Category = ‘Tequila’
AND units purchased >2
GROUP BY Store
HAVING SUM(sales) > 30.00
ORDER BY 3
```

Commenting




```
-- Basic commenting

/* Multiple line
comment

*inside a line comment*

End of Multiple line
comment*/
```

Workshop

Which products have a case cost of more than $100?

Which tequilas have a case cost of more than $100?

Which tequilas or scotch whiskies have a case cost of more than $100?

Which tequilas or scotch whiskies have a cast cost between $100 and $120?

Which whiskies of any kind cost more than $100?'

Which whiskies of any kind cost between $100 and $150?

Which products except tequilas cost between $100 and $120?

GENERAL ASSEMBLY

Querying Relational Database

UNION
JOIN 2 Tables
JOIN Multiple Tables

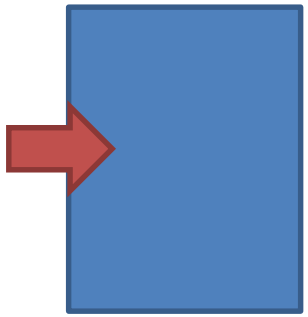
Matthew Morris
Git: Morrisdata
MatthewMorris.DA@gmail.com

Query a Relational Database

SELECT
FROM
JOIN
ON
WHERE
GROUP BY
HAVING
UNION
ORDER BY
LIMIT

Unions

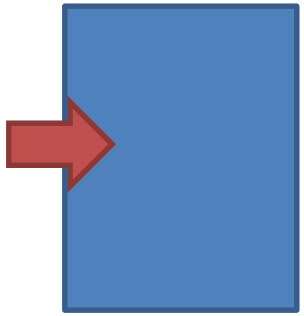
FY17



```
SELECT fy, pd, store_name, week1,
week2, week3 week4
FROM  FY17
```

UNION

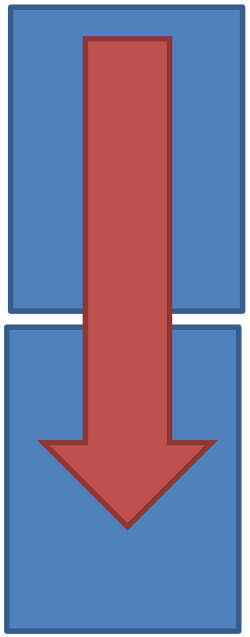
FY18



```
SELECT fy, pd, store_name, week1,
week2, week3 week4
FROM  FY18
```

Unions

FY17



FY18

```
SELECT fy, pd, store_name, week1,
week2, week3 week4
FROM  FY17
UNION
SELECT fy, pd, store_name, week1,
week2, week3 week4
FROM  FY18
```

COLUMNS
CONDITIONS
UNION and UNION ALL
ORDER BY

JOIN 1 Table

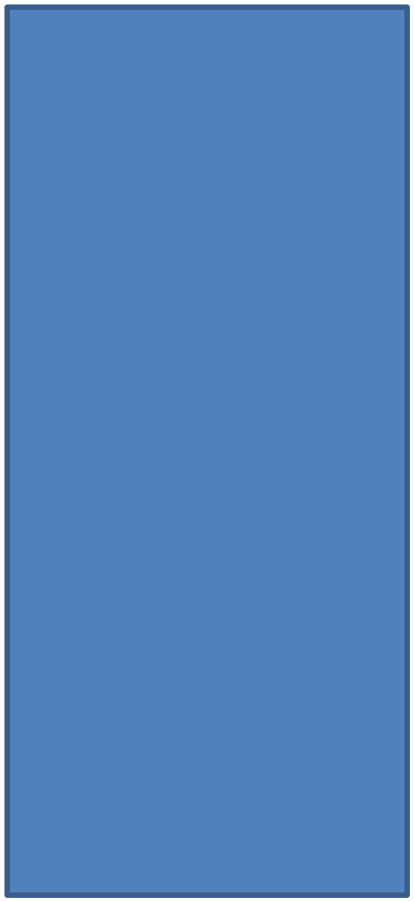
Joins

Querying a Relational Database

Joins

Query a Relational Database

LEFT/PRIMARY



RIGHT/SECONDARY

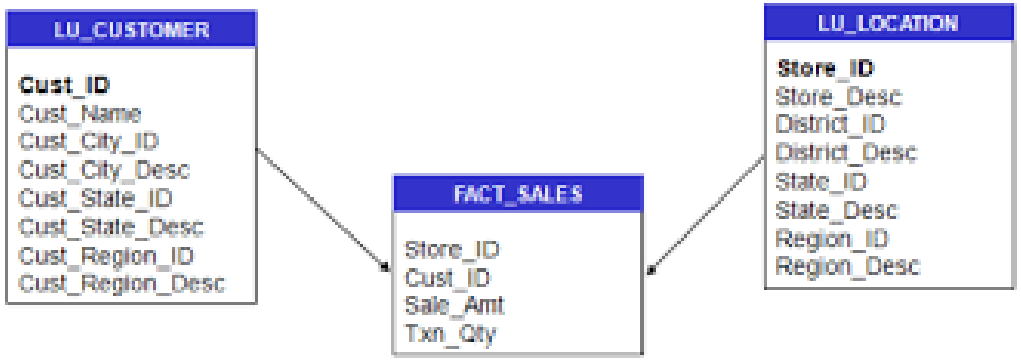
What table is the transaction table?

If you wanted to link on the lowest level of detail to the other tables what fields would you use?

Joins

	SALES		
	FIELD	TYPE	LENGTH
	ID	PK	1
	ARTIST	Char	25
	SONG	Char	225
	ALBUM	Char	225

Create a rough sketch with how The Iowa Liquor Sales Database Would JOIN



Joins

```
SELECT a.item, b.description, a.sales

FROM sales a
JOIN products b
ON a.item=b.item
```

Joins

1. Create separate queries to join each table to Sales
    a. Products to Sales
    b. County to Sales
    c. Stores to Sales
2. Use this as an opportunity to bring fields in from both tables.
3. Try out some aggregations or Wild card searches. Stretch with an Aggregate and a Group by

Joins

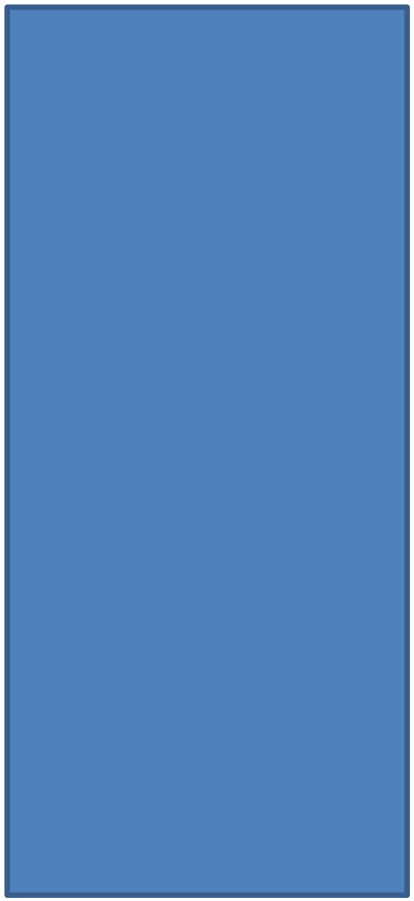
```
SELECT c.field, a.field, b.field, a.field,c.field

FROM table1 a

JOIN table2 b
ON a.field=b.field

JOIN table3 c
ON a.field=c.field
```

LEFT/PRIMARY



RIGHT/SECONDARY

Joins

Types of Joins

nner Join	Match in both tables
Left-Outer Join	Includes data from the primary table that may not have matches
Right-Outer Join	Includes data from the secondary table that may not have matches
Exception Join	Returns Primary table data that does not match with the secondary table
Right-Exception Join	Returns Secondary table data that does not match with the Primary table
Cross Join	Returns all data whether a match exists or not

Example

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

Inner Join

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

```
FROM Employees a
JOIN Salaries b
ON a.id=b.id
```

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	7000

Left Outer Join

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

```
FROM Employees a
LEFT JOIN Salaries b
ON a.id=b.id
```

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
5	Simone	MacDonald	NULL	NULL
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	120000

Right-Outer Join

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

```
FROM Employees a
RIGHT JOIN Salaries b
ON a.id=b.id
```

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	120000
NULL	NULL	NULL	17	70000

Left Exception Join

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

```
FROM Employees a
LEFT JOIN Salaries b
ON a.id=b.id
WHERE b.id IS NULL
```

id	first_name	last_name	id	current_salary
5	Simone	MacDonald	NULL	NULL

Right Exception Join

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

```
FROM Employees a
RIGHT JOIN Salaries b
ON a.id=b.id
WHERE a.id IS NULL
```

id	first_name	last_name	id	current_salary
NULL	NULL	NULL	17	70000

Cross Join

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

FROM Employees a
CROSS JOIN Salaries b
ON a.id=b.id

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
5	Simone	MacDonald	NULL	NULL
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	120000
NULL	NULL	NULL	17	70000

GENERAL ASSEMBLY

Join types

LEFT OUTER
RIGHT OUTER
LEFT EXCEPTION
RIGHT EXCEPTION
CROSS
COALESCE

Matthew Morris

Git: Morrisdata

MatthewMorris.DA@gmail.com

Join Types

SELECT
FROM
JOIN
ON
WHERE
GROUP BY
HAVING
UNION
ORDER BY
LIMIT

Join Types

OUTER, EXCEPTION, AND CARTESIAN JOINS

Employees

id	first_name	last_name
2	Gabe	Moore
3	Doreen	Mandeville
5	Simone	MacDonald
7	Madisen	Flateman
11	Ian	Paasche
13	Mimi	St. Felix

Salaries

id	current_salary
2	50000
3	60000
7	55000
11	75000
13	120000
17	70000

Join Types

OUTER, EXCEPTION, AND CARTESIAN JOINS

- An INNER JOIN (also called a direct join) displays only the rows that have a match in both joined tables.

- An INNER JOIN would yield this table:

SELECT * FROM employees INNER JOIN salaries ON employees.ID = salaries.ID;

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	7000

Join Types

OUTER, EXCEPTION, AND CARTESIAN JOINS

A LEFT OUTER JOIN returns both:

- Data that both tables have in common.
- Data from the **primary** table selected, which does not have matching data to join to in the secondary table.

- A LEFT OUTER JOIN would yield this table:

SELECT * FROM employees LEFT OUTER JOIN salaries ON employees.ID = salaries.ID;

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
5	Simone	MacDonald	NULL	NULL
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	120000

Join Types

OUTER, EXCEPTION, AND CARTESIAN JOINS

- A RIGHT OUTER JOIN returns both:
  - Data that two tables have in common.
  - Data from the **secondary** table selected, which does not have matching data to join to in the primary table.
- A RIGHT OUTER JOIN would yield this table:

SELECT * FROM employees RIGHT OUTER JOIN salaries ON employees.ID = salaries.ID;

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	120000
NULL	NULL	NULL	17	70000

Join Types

OUTER, EXCEPTION, AND CARTESIAN JOINS

- A FULL OUTER JOIN returns all data from each table, regardless of whether it has matching data in the other table.
- A FULL OUTER JOIN would yield this table:

SELECT * FROM employees FULL OUTER JOIN salaries ON employees.ID = salaries.ID;

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	3	60000
5	Simone	MacDonald	NULL	NULL
7	Madisen	Flateman	7	55000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	13	120000
NULL	NULL	NULL	17	70000

Join Types

OUTER, EXCEPTION, AND CARTESIAN JOINS

- An EXCEPTION JOIN returns only the data from the primary, or first table selected, which does not have matching data to join to in the secondary table.
- An EXCEPTION JOIN would yield this table:

```
SELECT * FROM employees LEFT OUTER
JOIN salaries ON employees.ID = salaries.ID
WHERE salaries.ID IS NULL;
```

id	first_name	last_name	id	current_salary
5	Simone	MacDonald	NULL	NULL

Join Types

OUTER, EXCEPTION, AND CARTESIAN JOINS

- A RIGHT EXCEPTION JOIN returns only the data from the secondary, or second table selected, which does not have matching data to join to in the primary table.
- A RIGHT EXCEPTION JOIN would yield this table:

```
SELECT * FROM employees RIGHT OUTER
JOIN salaries ON employees.ID = salaries.ID
WHERE employees.ID IS NULL;
```

id	first_name	last_name	id	current_salary
NULL	NULL	NULL	17	70000

Join Types

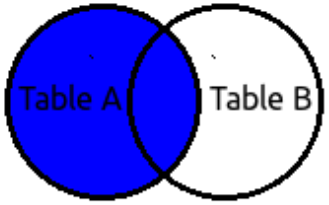
OUTER, EXCEPTION, AND CARTESIAN JOINS

- A CROSS JOIN matches every row of the primary table with every row of the secondary table.
- This type of join results in a Cartesian product of the tables, is generally detrimental to slow performance, and is not desired.
- A CROSS JOIN would yield this table:

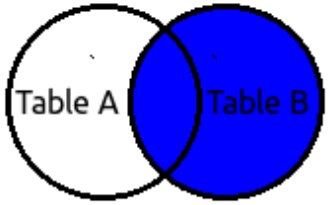
SELECT * FROM employees CROSS JOIN salaries;

id	first_name	last_name	id	current_salary
2	Gabe	Moore	2	50000
3	Doreen	Mandeville	2	50000
5	Simone	MacDonald	2	50000
7	Madisen	Flateman	2	50000
11	Ian	Paasche	2	50000
13	Mimi	St. Felix	2	50000
2	Gabe	Moore	3	60000
3	Doreen	Mandeville	3	60000
5	Simone	MacDonald	3	60000
7	Madisen	Flateman	3	60000
11	Ian	Paasche	3	60000
13	Mimi	St. Felix	3	60000
2	Gabe	Moore	7	55000
3	Doreen	Mandeville	7	55000
5	Simone	MacDonald	7	55000
7	Madisen	Flateman	7	55000
11	Ian	Paasche	7	55000
13	Mimi	St. Felix	7	55000
2	Gabe	Moore	11	75000
3	Doreen	Mandeville	11	75000
5	Simone	MacDonald	11	75000
7	Madisen	Flateman	11	75000
11	Ian	Paasche	11	75000
13	Mimi	St. Felix	11	75000
2	Gabe	Moore	13	120000
3	Doreen	Mandeville	13	120000
5	Simone	MacDonald	13	120000
7	Madisen	Flateman	13	120000
11	Ian	Paasche	13	120000
13	Mimi	St. Felix	13	120000
2	Gabe	Moore	17	70000
3	Doreen	Mandeville	17	70000
5	Simone	MacDonald	17	70000
7	Madisen	Flateman	17	70000
11	Ian	Paasche	17	70000
13	Mimi	St. Felix	17	70000

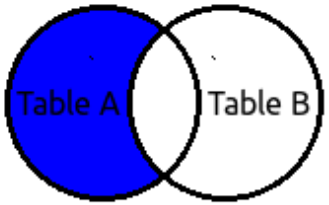
Join Types



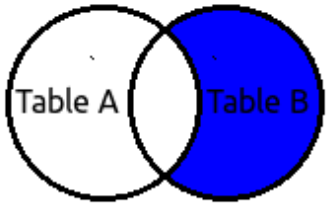

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
```



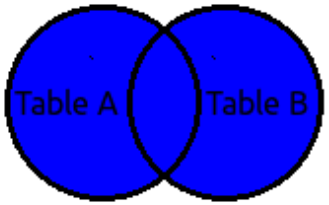

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
```



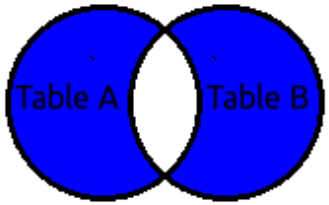

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE B.Value IS NULL
```



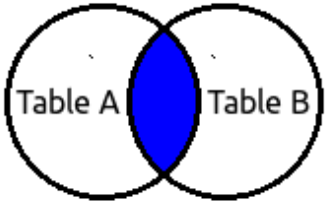

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
OR B.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
INNER JOIN
    [Table B] B
ON A.Value = B.Value
```

Join Types

"I want to see all of the information we can get on inactive stores for sales, if there are any, and their addresses."

Process to pick the right join (be methodical)

GENERAL ASSEMBLY

Common Table Expressions/Subselect

WITH
SELECT INTO

Matthew Morris

Git: Morrisdata

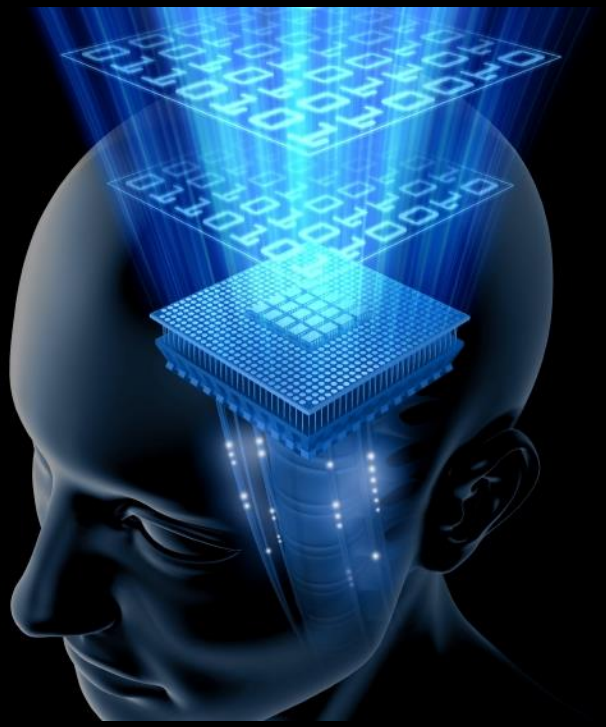
MatthewMorris.DA@gmail.com

GENERAL ASSEMBLY

3 Value logic and Case statements

```
CASE
      WHEN  expression THEN value
       ELSE
END
```

CREATING LABELS
LOGIC
SUBSELECT TRICK WITH COUNT



Matthew Morris
Git: Morrisdata
MatthewMorris.DA@gmail.com

Case Statement

TRUE

FALSE

UNKNOWN

QUERY

Compounded computed columns

Week 1 Units	Week 2 Units	Week 3Units	Week 4 Units	Week 5Units	Week 6 Units	Week 7 Units	Week 8 Units

Period 1 Units = ((Week 5 Units + Week 6 Units + Week 7 Units + Week 8 Units) – (Week 1 Units + Week 2 Units + Week 3 Units + Week 4 Units)) / (Week 1 Units + Week 2 Units + Week 3 Units + Week 4 Units) * 100

Period difference = Period 2 Units – 1

Period 1 to Period 2 % of Difference

QUERY

IMU% AND COMPSHOP

QUERY

CASE exercise Building the basic query and the IMU%

Sell Price to Wholesaler | Net Landed Cost

$$\frac{\text{Sell Price to Wholesaler} - \text{Net Landed Cost}}{\text{IMU\%}} \times 100$$

QUERY

CASE exercise Create the lowest competitive shop column.

IMU%	Comp Shop1	Comp Shop2	Comp Shop3

New CompShop1

If Comp Shop1 = .00 Then show 999999

Otherwise show Comp Shop1

New CompShop2

If Comp Shop2 = .00 Then show 999999

Otherwise show Comp Shop2

New CompShop3

If Comp Shop3 = .00 Then show 999999

Otherwise show Comp Shop3

QUERY

CASE exercise Lowest comp shop between CompShop1 and CompShop2

IMU%	New CompShop1	New CompShop2	New CompShop3

If **New CompShop1** < **New CompShop2**

Then

New CompShop1

Otherwise show

New CompShop2

Lowest Comp between 1 and 2

QUERY

CASE exercise Lowest comp shop between CompShop1 and CompShop2

IMU%	Lowest Comp between 1 and 2	New CompShop3

If **Lowest Comp between 1 and 2** < **New CompShop3**

Then

Lowest Comp between 1 and 2

Otherwise show

New CompShop3

Lowest CompShop

QUERY

CASE exercise Building the basic query and the IMU%

Sell Price to Wholesaler	**IMU%**	**Lowest CompShop**

$$\frac{\text{Sell Price to Wholesaler} - \textbf{Lowest CompShop}}{\text{Sell Price to Wholesaler}} \times 100$$

Sell Price to Comp % of Diff

QUERY

CASE exercise Building the basic query and the IMU%

IMU%	% of Diff High

If **IMU%** < 2 **AND** **Sell Price to Comp % of Diff** < - 5

Then **ACTION REQUIRED**

QUERY

ACTION REQUIRED	ACTION REQUIRED	ACTION REQUIRED
Order 5 of X	Research Vendor A	Retire Asset
Order 7 of Y	Pay Vendor X	Move Asset
Hold Order on B	Pay Vendor Y	Research Asset
Research Order C	Deny Vendor B	Retire Asset

Expert

Case Statements

```
SELECT field1, field2,
CASE
        WHEN field1 >= A THEN ‘field or expression'
END AS field3,
field4

FROM  Table1
```

Lets look at some code

Case Statement drills