

Dr Renata Borovica-Gajic
David Eccles



INFO90002 Database Systems & Information Modelling

Lecture 9 SQL Part 2

SELECT, HAVING, SET OPERATORS, VENN DIAGRAMS, DML, CASE



Lecture 9 SQL – Part 2

SELECT, HAVING, SET OPERATORS, VENN
DIAGRAMS, DML, CASE



www.csse.monash.edu.au/~hlc

- Extending your knowledge
 - DML
 - Comparison & Logic Operators
 - Set Operations
 - Subquery
 - Multiple record INSERTs
 - INSERT from a table, UPDATE, DELETE, REPLACE
 - Views
 - DDL
 - ALTER and DROP, TRUNCATE, RENAME
 - CTAS
- How to think about SQL
 - Problem Solving



- SQL keywords are case insensitive
 - We try to CAPITALISE them to make them clear
 - Improve readability of your statements
- Table names are Operating System Sensitive
 - If case sensitivity exists in the operating system, then the table names are case sensitive! (i.e. Linux)
 - Account <> ACCOUNT
- Field names are case insensitive
 - ACCOUNTID == AccountID == AcCoUnTID
- You can do maths in SQL...
 - `SELECT 1*1+1/1-1;`



- The select statement's job is just to return rows of data, it doesn't care about the order of these rows unless you specify the ORDER BY clause
- So what order do rows come out in if you don't specify the ORDER BY clause?
 - Any order
 - Possibly the order the records were created in
 - It is undefined
 - Because SQL may optimise the query which may change the order of results...
- So make sure you get into the habit of using the ORDER BY clause *if* you need a particular order
 - If you don't need order, don't use it – it's going to be slower



- The HAVING clause was added to SQL because the WHERE keyword **cannot be used** with **aggregate** functions.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

- Example:**

List the number of customers of each country, but ONLY include countries with more than 5 customers

```
SELECT COUNT(CustomerID), CountryName
FROM Customers
GROUP BY CountryName
HAVING COUNT(CustomerID) > 5;
```

Condition over the aggregate



- Comparison:

Operator	Description
=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<> OR !=	Not equal to (depends on DBMS as to which is used)

- Logic:

- AND, NOT, OR
- Example:

```
SELECT *
FROM Furniture
WHERE ((Type = 'Chair' AND Colour = 'Black')
       OR
       (Type = 'Lamp' AND Colour = 'Black'))
);
```



- **UPPER()**
 - Change to upper case
- **LOWER()**
 - Change to lower case
- **LEFT()**
 - Take the left X characters from a string
- **RIGHT()**
 - Take the X right characters from a string
- Many more examples are in the labs!



- UNION
 - Shows all rows returned from the queries (or tables)
- INTERSECT
 - Shows only rows that are common in the queries (or the tables)
- [UNION/INTERSECT] ALL
 - If you want duplicate rows shown in the results you need to use the ALL keyword.. UNION ALL etc.
- In MySQL only UNION and UNION ALL are supported



www.csse.unimelb.edu.au/~dbcsse/

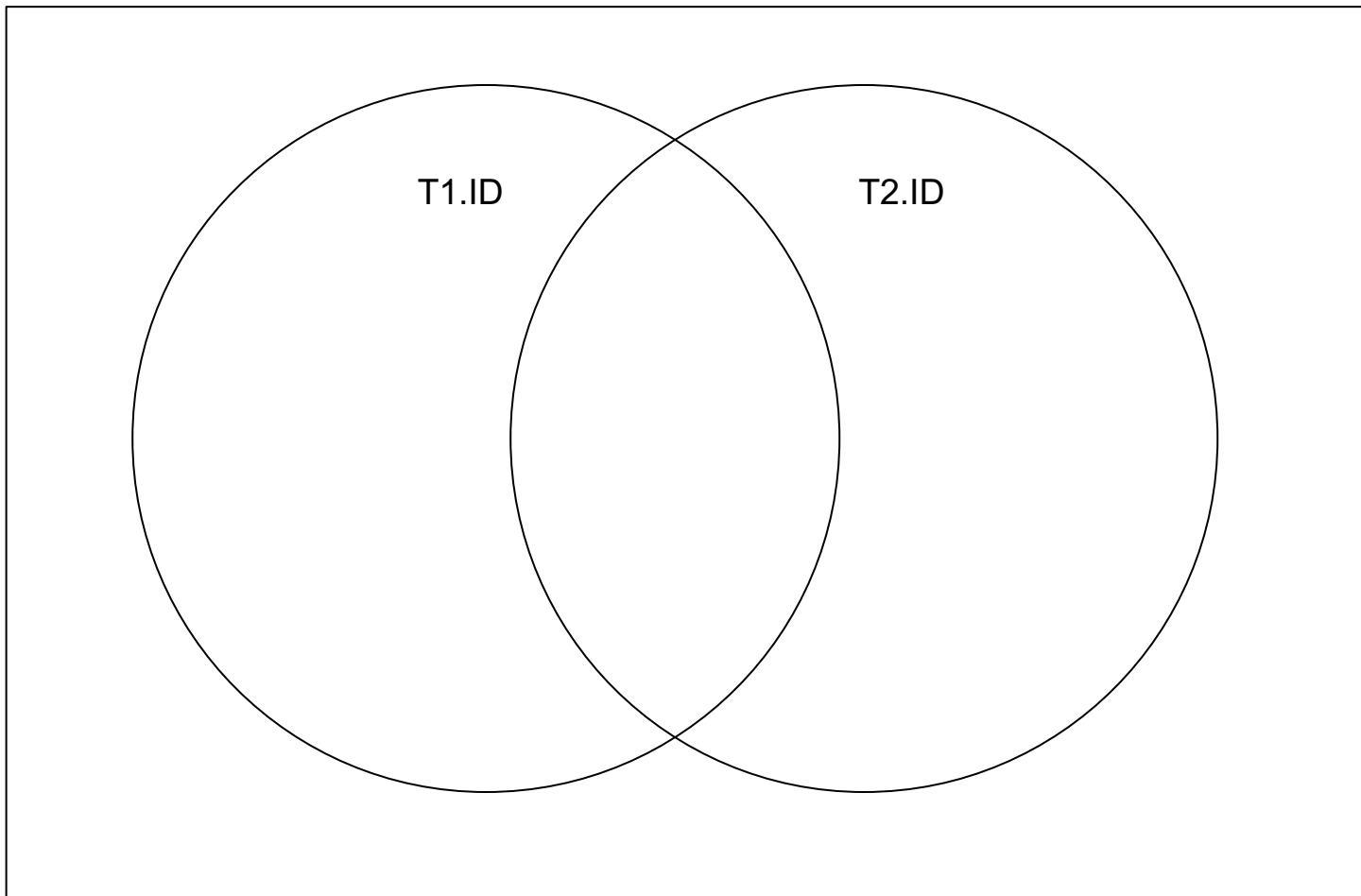
```
SELECT Employee.Name, EmployeeType
      FROM Employee INNER JOIN Hourly
        ON Employee.ID = Hourly.ID
UNION
SELECT Employee.Name, EmployeeType
      FROM Employee INNER JOIN Salaried
        ON Employee.ID = Salaried.ID;
```

Output Snippets Query 1 Result Lecture7.sql Result × Fetched 4 records. Duration: 00:00:00.000000

Name	EmployeeType
Alice	H
Alan	H
Sean	S
Linda	S



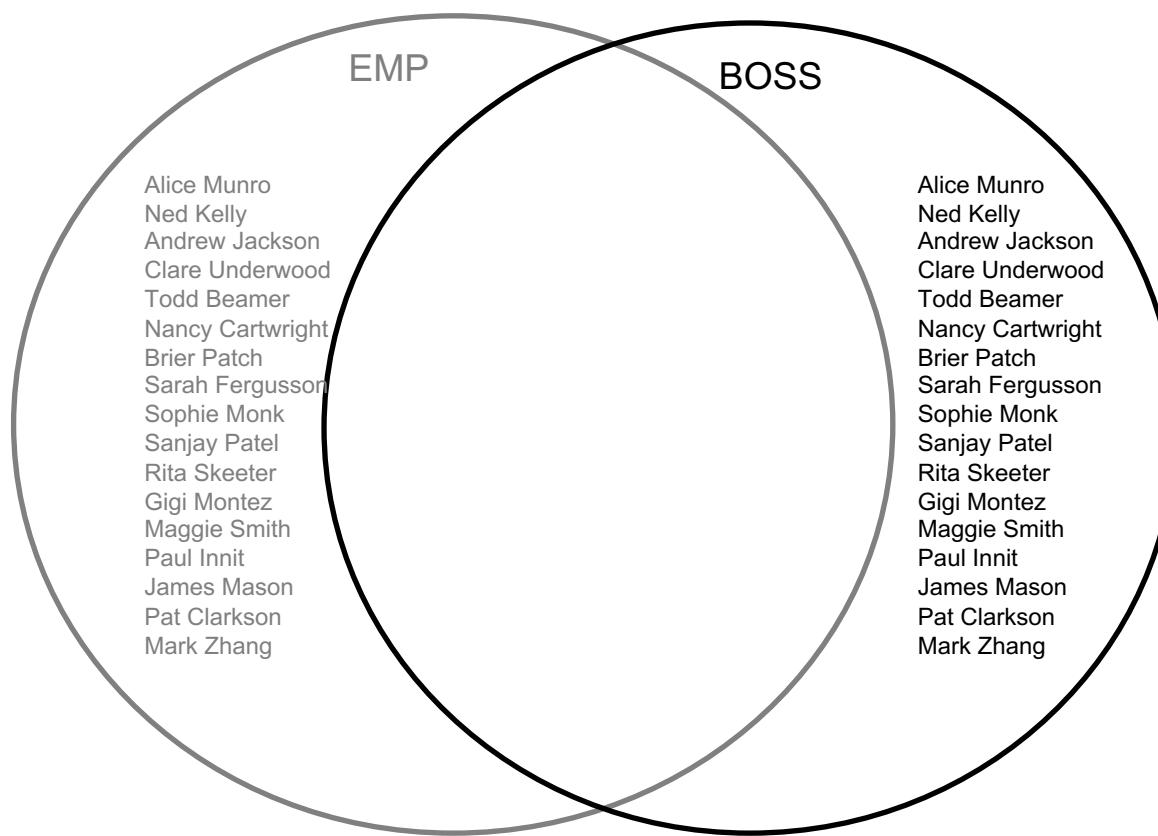
DATA MANIPULATION LANGUAGE





DATA MANUFACTURING

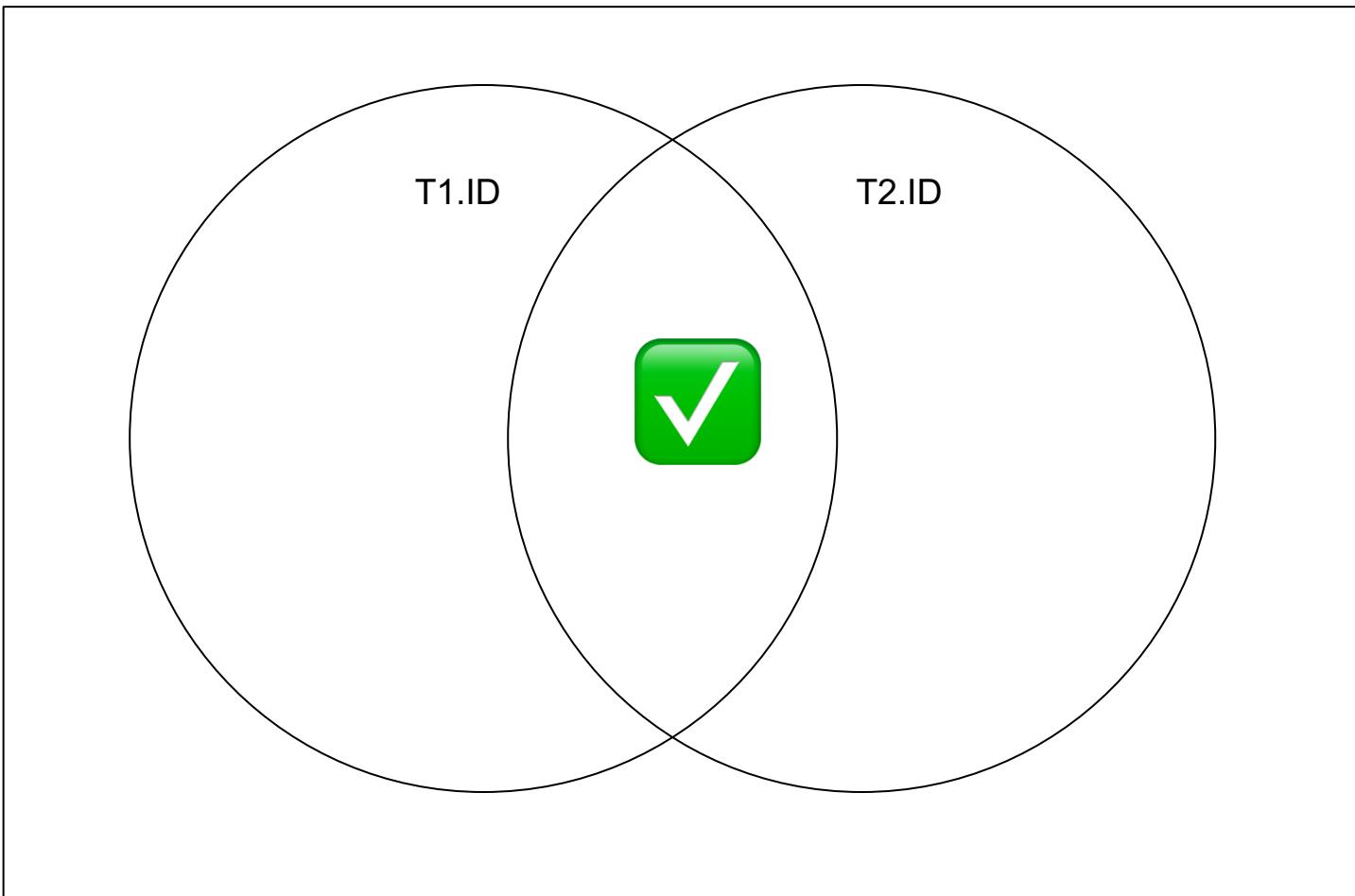
Unary join of the Employee table aliased as EMP, BOSS





DATA MANUFACTURE

- T1 INNER JOIN T2 ON T1.ID = T2.ID
- T1 NATURAL JOIN T2

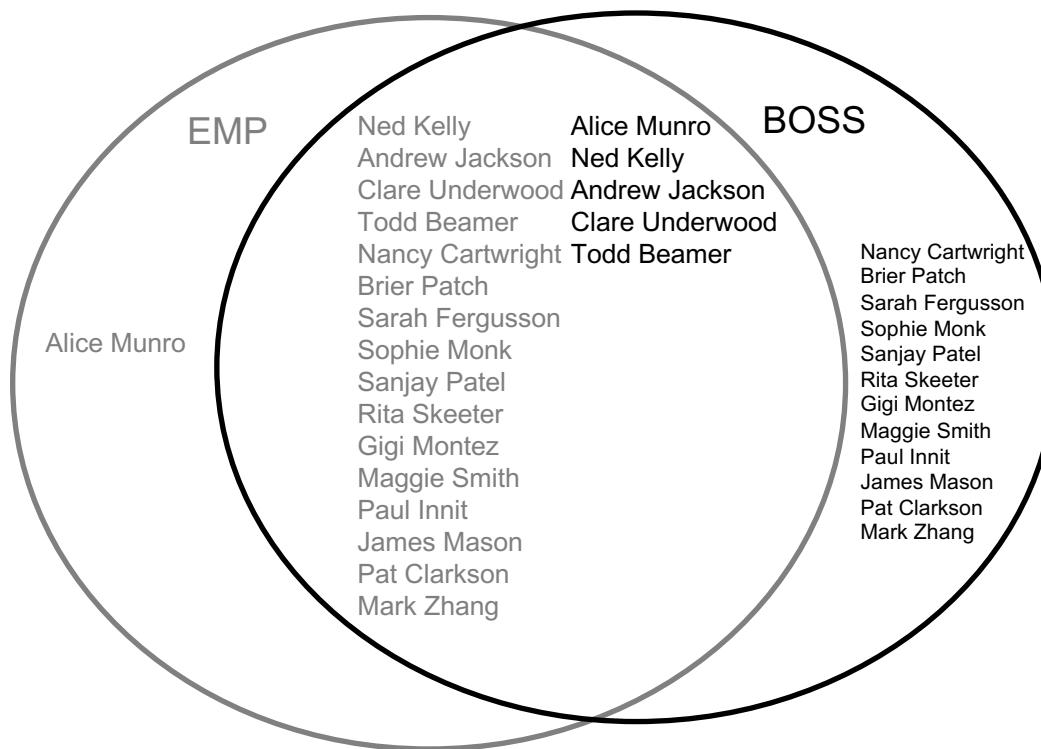




www.csse.unimelb.edu.au/~hdc/DB

```
SELECT emp.firstname as efirst, emp.lastname as elast,
       boss.firstname as bfirst, boss.lastname as blast
  FROM employee emp INNER JOIN employee boss
 WHERE emp.bossid = boss.employeeid;
```

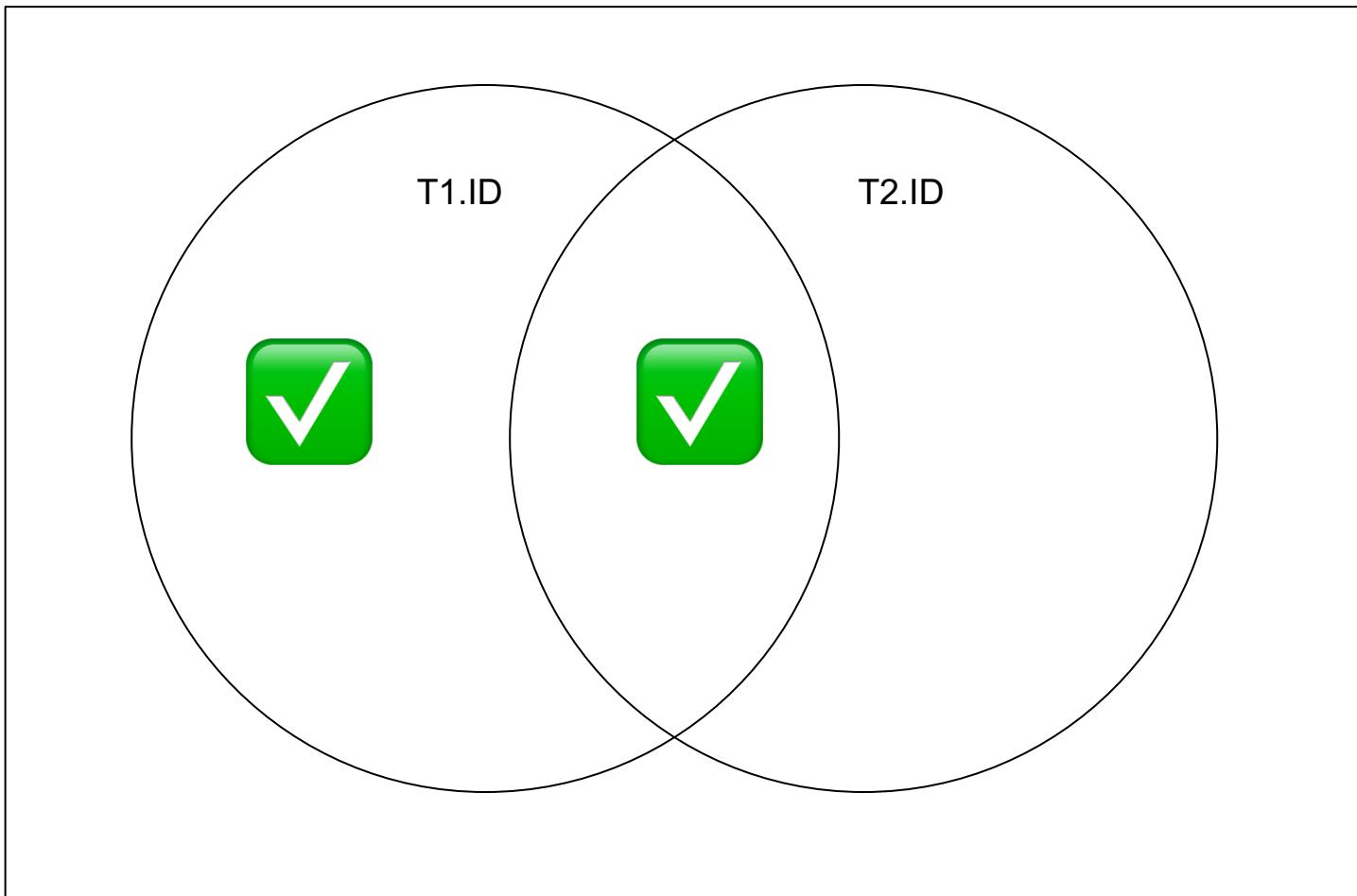
	efirst	elast	bfirst	blast
▶	Ned	Kelly	Alice	Munro
	Andrew	Jackson	Ned	Kelly
	Clare	Underw...	Ned	Kelly
	Todd	Beamer	Alice	Munro
	Nancy	Cartwright	Todd	Beamer
	Brier	Patch	Alice	Munro
	Sarah	Fergusson	Brier	Patch
	Sophie	Monk	Alice	Munro
	Sanjay	Patel	Andrew	Jackson
	Rita	Skeeter	Clare	Underw...
	Gigi	Montez	Clare	Underw...
	Maggie	Smith	Clare	Underw...
	Paul	Innit	Andrew	Jackson
	James	Mason	Andrew	Jackson
	Pat	Clarkson	Andrew	Jackson
	Mark	Zhang	Andrew	Jackson





DATA MANUFACTURE

- T1 LEFT OUTER JOIN T2 ON T1.ID = T2.ID

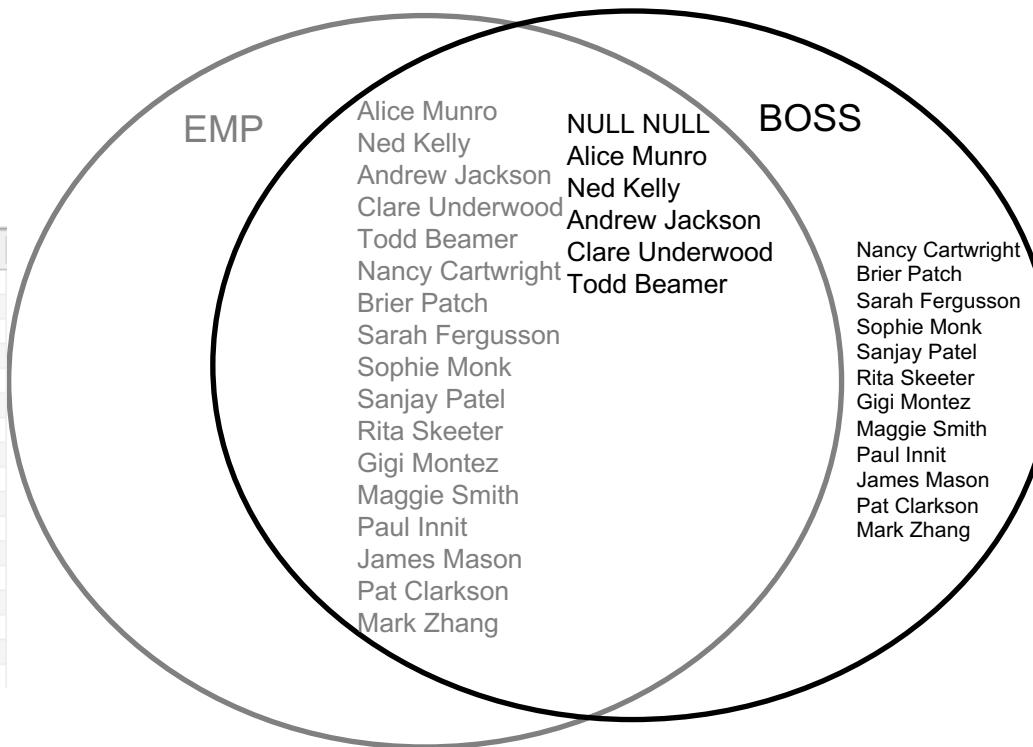




AUGUST 2018 - 2019 SEMESTER

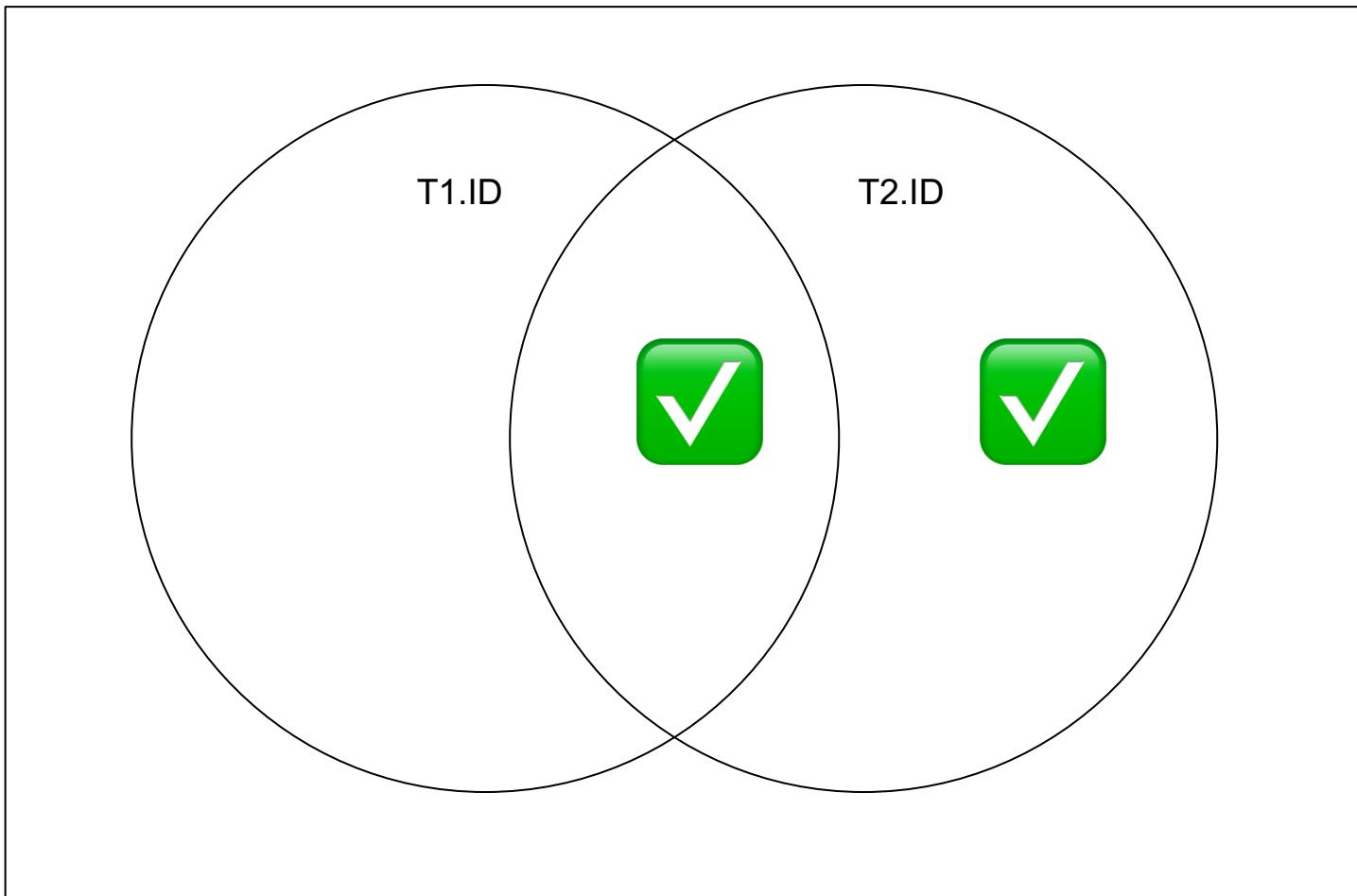
```
SELECT emp.firstname as efirst, emp.lastname as elast,
       boss.firstname as bfirst, boss.lastname as blast
  FROM employee emp LEFT OUTER JOIN employee boss
 WHERE emp.bossid = boss.employeeid;
```

efirst	elast	bfirst	blast
Alice	Munro	NULL	NULL
Ned	Kelly	Alice	Munro
Andrew	Jackson	Ned	Kelly
Clare	Underwood	Ned	Kelly
Todd	Beamer	Alice	Munro
Nancy	Cartwright	Todd	Beamer
Brier	Patch	Alice	Munro
Sarah	Fergusson	Brier	Patch
Sophie	Monk	Alice	Munro
Sanjay	Patel	Andrew	Jackson
Rita	Skeeter	Clare	Underwood
Gigi	Montez	Clare	Underwood
Maggie	Smith	Clare	Underwood
Paul	Innit	Andrew	Jackson
James	Mason	Andrew	Jackson
Pat	Clarkson	Andrew	Jackson
Mark	Zhang	Andrew	Jackson





- T1 **RIGHT OUTER JOIN** T2 ON T1.ID = T2.ID

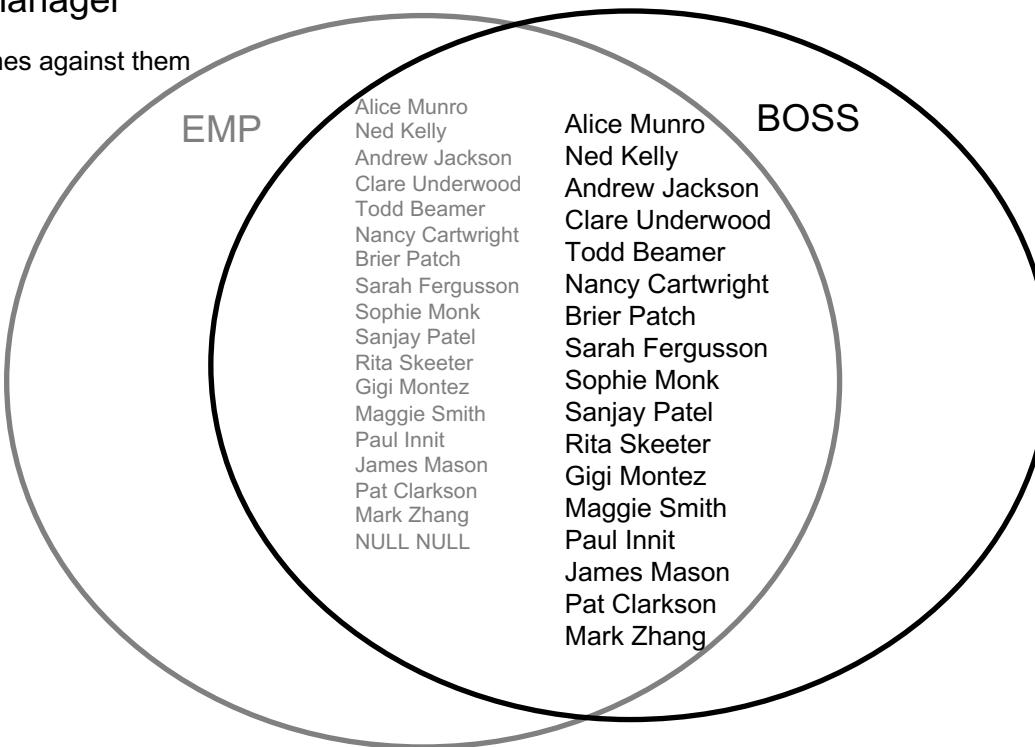




AUGUST 2018 - 2019 SEMESTER

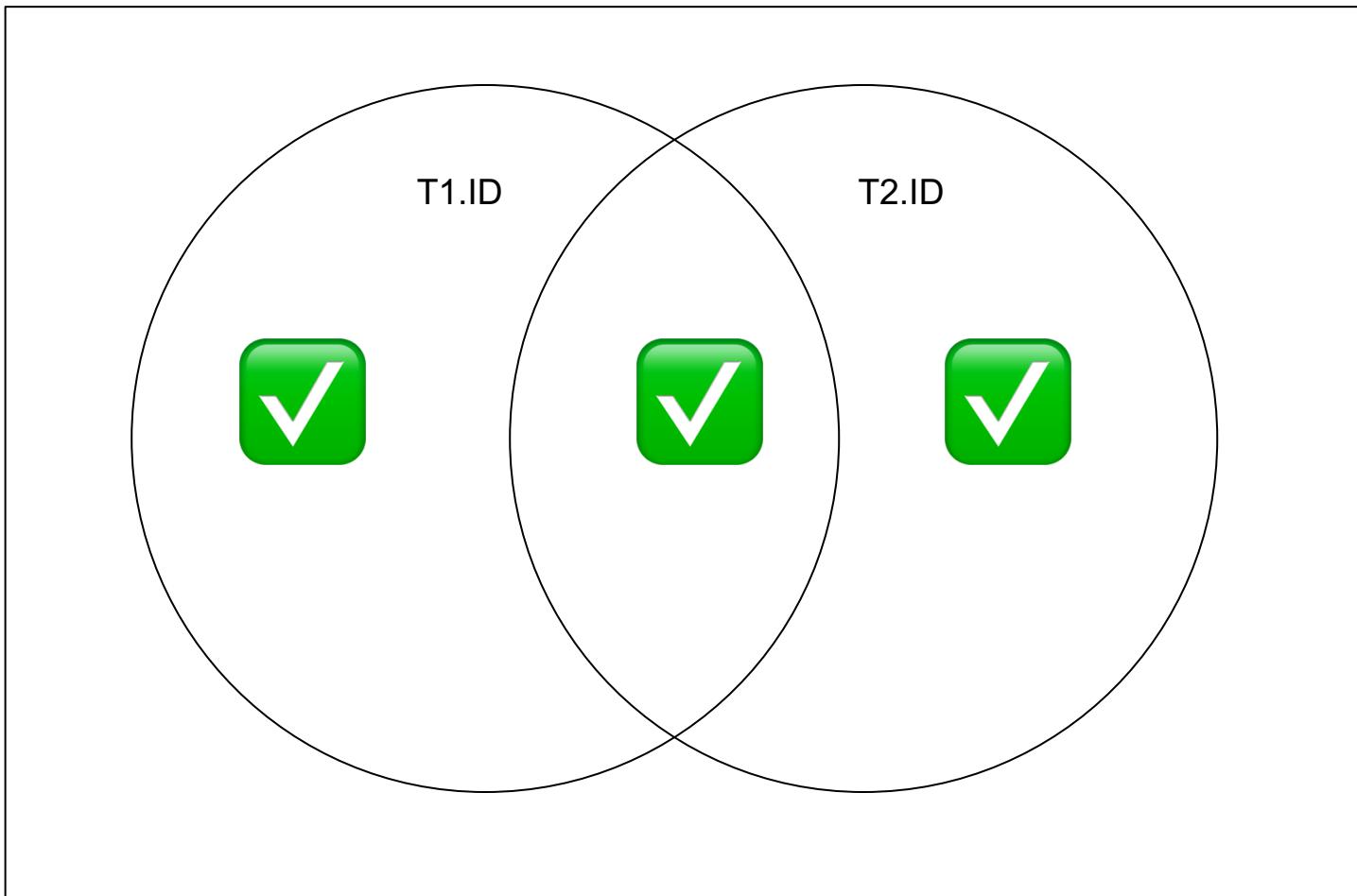
```
SELECT emp.firstname as efirst, emp.lastname as elast,
       boss.firstname as bfirst, boss.lastname as blast
  FROM employee emp RIGHT OUTER JOIN employee boss
 WHERE emp.bossid = boss.employeeid;
-- Asks list every employee of every manager
-- Hence Sarah, Sophie, Rita, Mark etc have no names against them
```

efirst	elast	bfirst	blast
Ned	Kelly	Alice	Munro
Todd	Beamer	Alice	Munro
Brier	Patch	Alice	Munro
Sophie	Monk	Alice	Munro
Andrew	Jackson	Ned	Kelly
Clare	Underwood	Ned	Kelly
Sanjay	Patel	Andrew	Jackson
Paul	Innit	Andrew	Jackson
James	Mason	Andrew	Jackson
Pat	Clarkson	Andrew	Jackson
Mark	Zhang	Andrew	Jackson
Rita	Skeeter	Clare	Underwood
Gigi	Montez	Clare	Underwood
Maggie	Smith	Clare	Underwood
Nancy	Cartwright	Todd	Beamer
NULL	NULL	Nancy	Cartwright
Sarah	Fergusson	Brier	Patch
NULL	NULL	Sarah	Fergusson
NULL	NULL	Sophie	Monk
NULL	NULL	Sanjay	Patel
NULL	NULL	Rita	Skeeter
NULL	NULL	Gigi	Montez
NULL	NULL	Maggie	Smith
NULL	NULL	Paul	Innit
NULL	NULL	James	Mason
NULL	NULL	Pat	Clarkson
NULL	NULL	Mark	Zhang





- T1 **FULL OUTER JOIN** T2 ON T1.ID = T2.ID



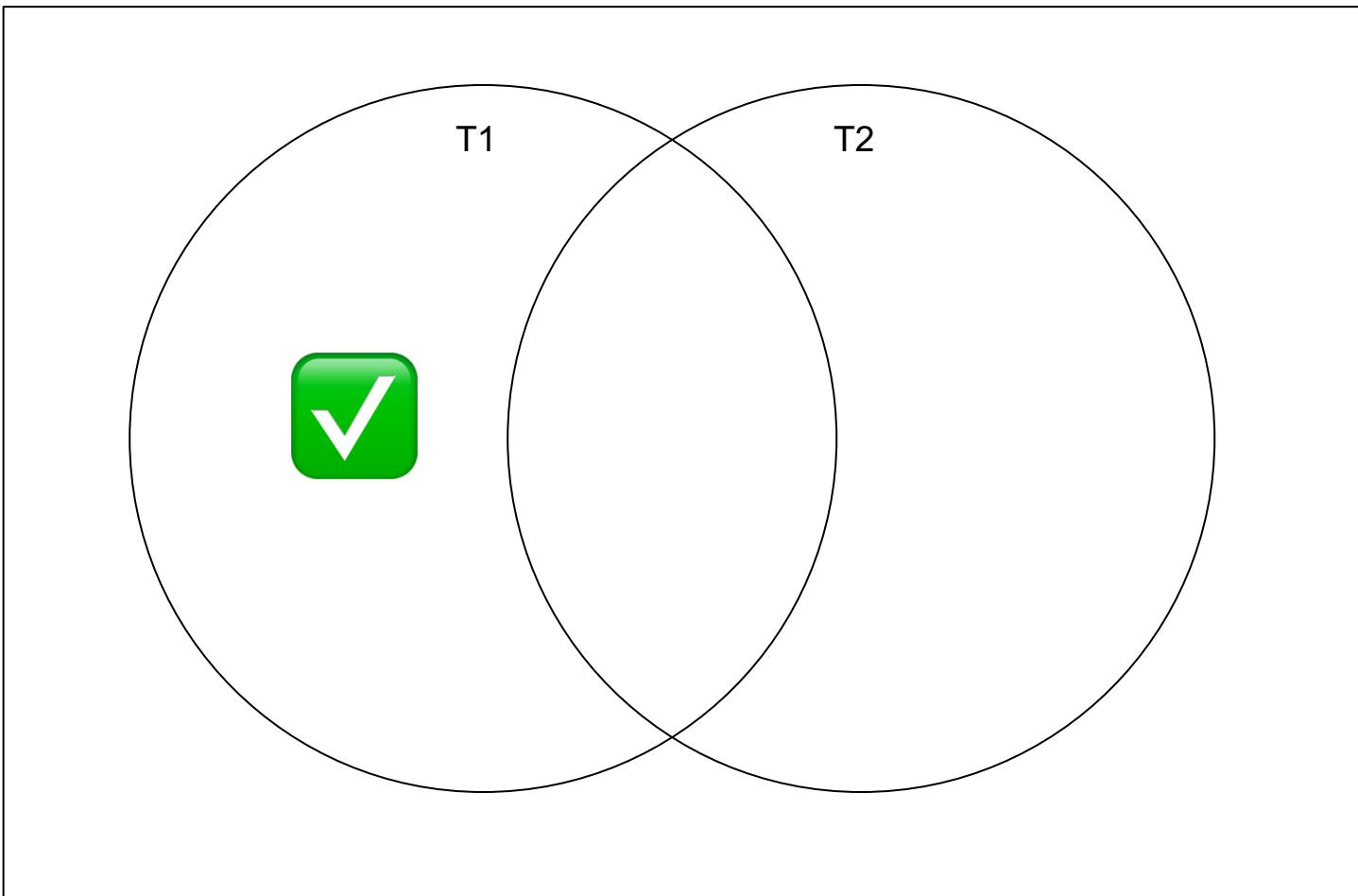


DATA MANAGERS FOR THE 21ST CENTURY

- MySQL DBMS server does not support full outer joins



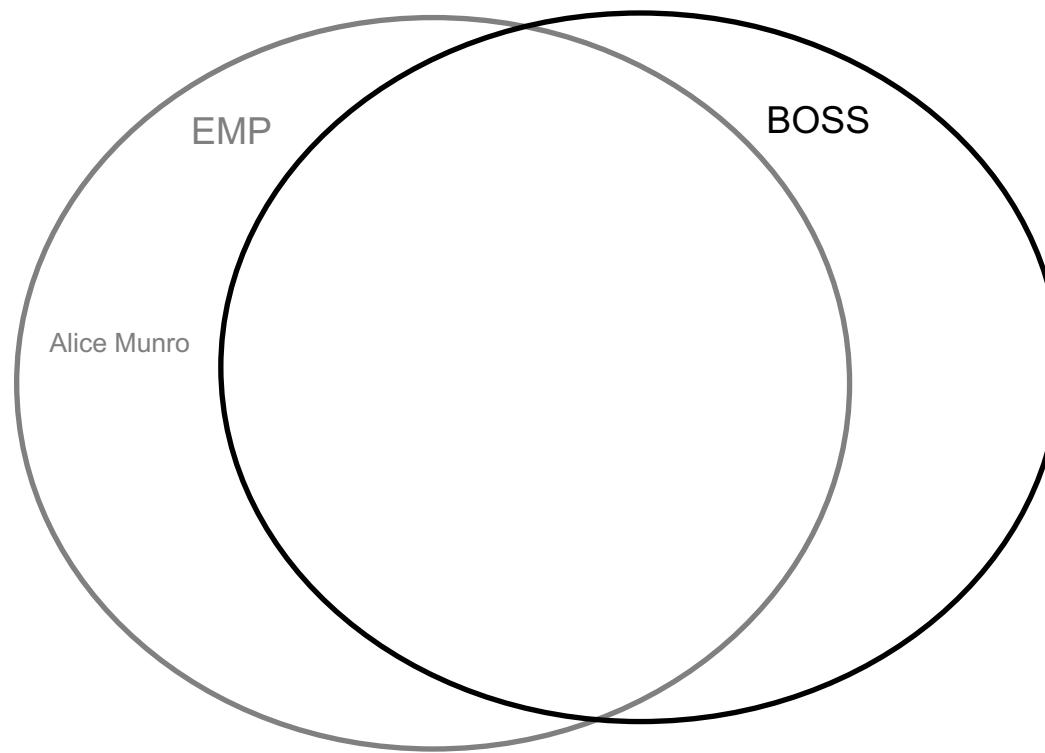
- T1 LEFT OUTER JOIN T2 ON T1.ID = T2.ID
WHERE T2.ID IS NULL





DATA MANIPULATION LANGUAGE

```
SELECT emp.firstname as efirst, emp.lastname as elast,  
boss.firstname as bfirst, boss.lastname as blast  
FROM employee emp LEFT OUTER JOIN employee boss  
ON emp.bossid = boss.employeeid  
WHERE boss.employeeid IS NULL;
```





- Inserting records from a table:
 - Note: table must already exist

```
INSERT INTO NewEmployee
SELECT * FROM Employee;
```

- Multiple record inserts: All columns must be inserted

```
INSERT INTO Employee VALUES
(DEFAULT, "A", "A's Addr", "2012-02-02", NULL, "S"),
(DEFAULT, "B", "B's Addr", "2012-02-02", NULL, "S"),
(DEFAULT, "C", "C's Addr", "2012-02-02", NULL, "S");
```

Specific columns will be inserted

```
INSERT INTO Employee
(Name, Address, DateHired, EmployeeType)
VALUES
("D", "D's Addr", "2012-02-02", "C"),
("E", "E's Addr", "2012-02-02", "C"),
("F", "F's Addr", "2012-02-02", "C");
```

- Changes *existing* data in tables
 - Order of statements is important
 - Specifying a WHERE clause is important
 - Unless you want it to operate on the whole table

```
UPDATE Hourly
      SET HourlyRate = HourlyRate * 1.10;
```

- **Example:** Increase all salaries greater than \$100000 by 10% and all other salaries by 5%

```
UPDATE Salaried
      SET AnnualSalary = AnnualSalary * 1.05
      WHERE AnnualSalary <= 100000;
UPDATE Salaried
      SET AnnualSalary = AnnualSalary * 1.10
      WHERE AnnualSalary > 100000;
```

Any problems with this?



- A better solution in this case is to use the **CASE** command

```
UPDATE Salaried
SET AnnualSalary =
CASE
    WHEN AnnualSalary <= 100000
        THEN AnnualSalary * 1.05
    ELSE AnnualSalary * 1.10
END;
```

If salary is lower than 100000 increase it by 5%,
otherwise increase it by 10%



- REPLACE
 - REPLACE works identically as INSERT
 - Except if an old row in a table has a key value the same as the new row then it is overwritten...
- DELETE
 - The DANGEROUS command – deletes *ALL* records
 - **DELETE FROM Employee;**
 - The better version (unless you are really, really sure)
**DELETE FROM Employee
WHERE Name = "Grace";**
 - Be aware of the foreign key constraints
 - ON DELETE CASCADE or ON DELETE RESTRICT (lab practice)



DATA MANAGEMENT

- You need to be able to write SQL
 - SELECT
 - DML (INSERT, UPDATE, DELETE)
 - DDL (CREATE, ALTER, DROP)



ANSWER QUESTIONS

- More SQL (SQL 3)