INFO20003 Database Systems

Dr Renata Borovica-Gajic

Lecture 08
SQL

Week 4

What is SQL

- SQL – or SEQUEL is a language used in relational databases
- DBMS support CRUD
  - Create, Read, Update, Delete commands
- SQL supports CRUD
  - Create, Select, Update, Delete commands
- Other info
  - You can see the 2011 standard of SQL at
    - http://www.jtc1sc32.org/doc/N2151-2200/32N2153T-text_for_ballot-FDIS_9075-1.pdf
  - Wikipedia has several sections on SQL (good for generic syntax)
    - http://en.wikipedia.org/wiki/Category:SQL_keywords

SQL Language

- Provides the following capabilities:
  - Data Definition Language (DDL)
    - To define and set up the database
    - CREATE, ALTER, DROP
  - Data Manipulation Language (DML)
    - To maintain and use the database
    - SELECT, INSERT, DELETE, UPDATE
  - Data Control Language (DCL)
    - To control access to the database
    - GRANT, REVOKE
  - Other Commands
    - Administer the database
    - Transaction Control

How We Use SQL

- In **Implementation** of the database
  - Take the tables we design in physical design
  - Implement these tables in the database using create commands
- In **Use** of the database
  - Use Select commands to read the data from the tables, link the tables together etc
  - Use alter, drop commands to update the database
  - Use insert, update, delete commands to change data in the database

SQL Context in Development Process

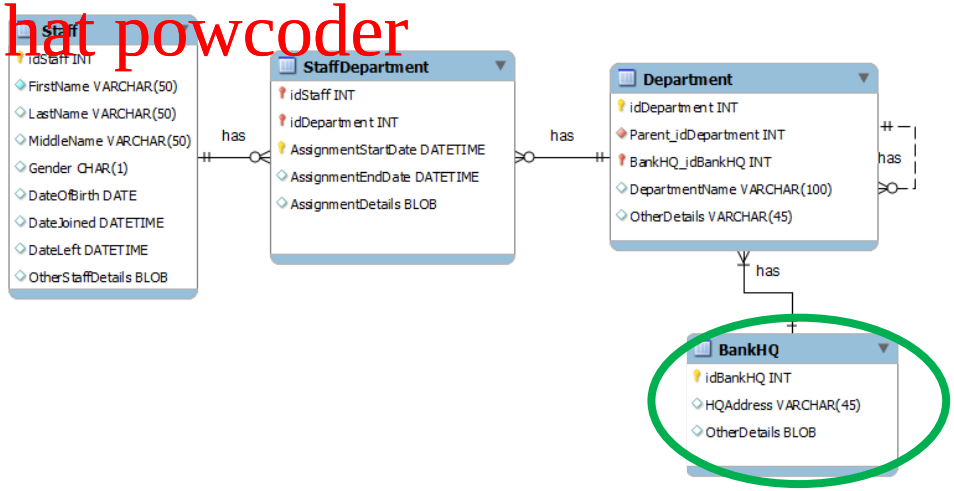
1.
```
CREATE TABLE BankHQ (
    BankHQID        INT(4)          AUTO_INCREMENT,
    HQAddress       VARCHAR(300)    NOT NULL,
    OtherHQDetails  VARCHAR(500),
PRIMARY KEY (BankHQID)
)
```


2.
```
INSERT INTO BankHQ VALUES
    (DEFAULT,"23 Charles St Peterson North 2022", 'Main Branch');
INSERT INTO BankHQ VALUES
    (DEFAULT,"213 Jones Rd Parkville North 2122", 'Sub Branch');
```

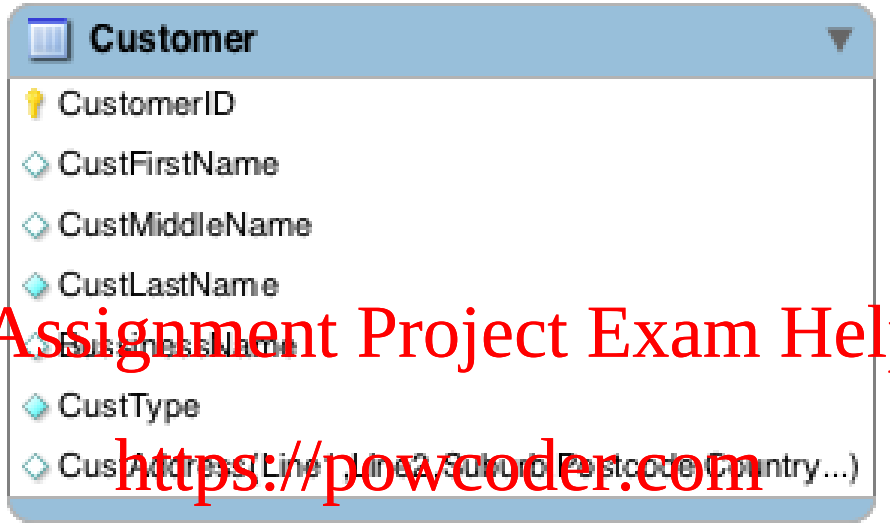

3.
```
select * from BANKHQ
```

Overview | Output | Snippets | Query 3 Result

BankHQID	HQAddress	OtherHQDetails
1	23 Charles St Peterson North 2022	Main Branch
2	213 Jones Rd Parkville North 2122	Sub Branch



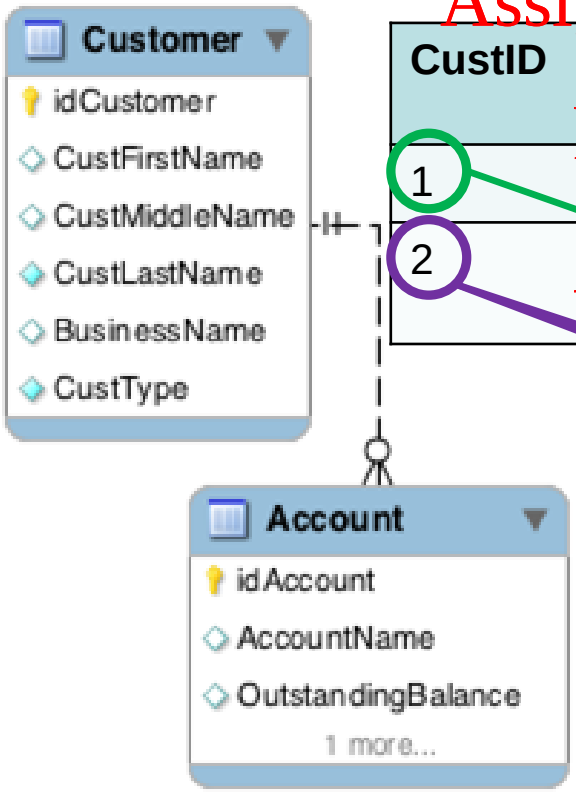
Create Table: Review




```
CREATE TABLE Customer (
  CustomerID          smallint                        auto_increment,
  CustFirstName       varchar(100),
  CustMiddleName      varchar(100),
  CustLastName        varchar(100)                    NOT NULL,
  BusinessName        varchar(200),
  CustType            enum('Personal','Company')      NOT NULL,
  PRIMARY KEY  (CustomerID)
);
```

Foreign keys: Review

- We looked at Customer
  - A customer can have a number of Accounts
  - The tables get linked through a foreign key



CustID	Customer FirstName	CustMiddle Name	CustLast Name	Business Name	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company

AccountID	AccountName	Outstanding Balance	CustID
01	Peter Smith	245.25	1
05	JJ Ent.	552.39	2
06	JJ Ent. Mgr	10.25	2

SQL CREATE Statement (With FK)

```
CREATE TABLE Account (
   AccountID              smallint        auto_increment,
   AccountName            varchar(100)    NOT NULL,
   OutstandingBalance     DECIMAL(10,2)   NOT NULL,
   CustomerID             smallint        NOT NULL,
   PRIMARY KEY (AccountID),
   FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
         ON DELETE RESTRICT
         ON UPDATE CASCADE
);
```

Insert Data

1)
```
INSERT INTO Customer
    (CustFirstName, CustLastName, CustType)
    VALUES ("Peter", "Smith", 'Personal');
```

Specifies which columns will be entered

2)
```
INSERT INTO Customer
    VALUES (DEFAULT, "James", NULL, "Jones",
            "JJ Enterprises", 'Company');
```

No column specification means ALL columns need to be entered

```
INSERT INTO Customer
    VALUES (DEFAULT, "", NULL, "Smythe",
            "", 'Company');
```


Customer

CustID	CustomerFirst Name	CustMiddle Name	CustLastName	BusinessName	CustType
1	Peter	NULL	Smith	NULL	Personal
2	James	NULL	Jones	JJ Enterprises	Company
3		NULL	Smythe		Company

What does NULL mean?

Null Island: The Busiest Place That Doesn't Exist:
https://www.youtube.com/watch?v=bjvIpI-1w84
by the channel MinuteEarth

Query Table with SELECT statement

- Select statement allows us to query table(s)
  - * (star): Allows us to obtain *all* columns from a table

All columns

```
1  select * from customer;
```

Output | Snippets | Query 3 Result | Query 4 Result

Fetched 3 records. Duration: 0.015 sec, f

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType
1	Peter	NULL	Smith	NULL	Personal
2	James	NULL	Jones	JJ Enterprises	Company
3		NULL	Smythe		Company

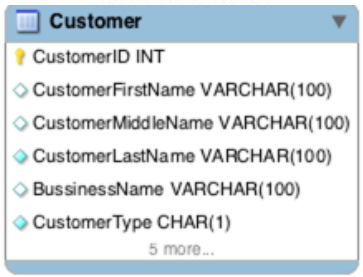
The SELECT Statement: Detail

- A cut down version of the SELECT statement – MySQL

- SELECT [ALL | DISTINCT] *select_expr* [, *select_expr* ...]
  - List the columns (and expressions) that are returned from the query
- [FROM *table_references*]
  - Indicate the table(s) or view(s) from where the data is obtained
- [WHERE *where_condition*]
  - Indicate the conditions on whether a particular row will be in the result
- [GROUP BY {*col_name* | *expr* } [ASC | DESC], ...]
  - Indicate categorisation of results
- [HAVING *where_condition*]
  - Indicate the conditions under which a particular category (group) is included in the result
- [ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ...]
  - Sort the result based on the criteria
- [LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}]
  - Limit which rows are returned by their return order (ie 5 rows, 5 rows from row 2)

Order is important! E.g. Limit cannot go before Group By or Having

Select Examples



SELECT * FROM Customer;
= Give me all information you have about customers

SQL

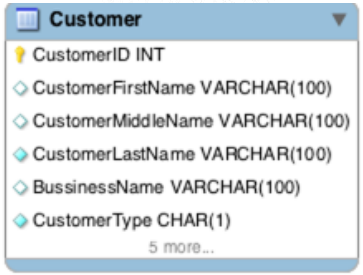
```
SELECT * FROM Customer;
```


RESULT

Edit: Export: Autosize:

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType
1	Peter	NULL	Smith	NULL	Personal
2	James	NULL	Jones	JJ Enterprises	Company
3	Akin	NULL	Smithies	Bay Wart	Company
4	Julie	Anne	Smythe	Konks	Company
5	Jen	NULL	Smart	BRU	Company
6	Lim	NULL	Lam	NULL	Personal
7	Kim	NULL	Unila	Saps	Company
8	James	Jay	Jones	JJ's	Company
9	Keith	NULL	Samson	NULL	Personal
NULL	NULL	NULL	NULL	NULL	NULL

Select Examples : Projection



In Relational Algebra:

$$\pi_{CustLastName}(Customer)$$

In SQL:
SELECT CustLastName
FROM Customer;

NOTE: MySQL doesn't discard duplicates. To remove them use DISTINCT in front of the projection list.

SQL

```
SELECT CustLastName FROM Customer;
```

Result

Export Autosize

CustLastName
Smith
Jones
Smythe
Smart
Lam
Unila
Jones
Samson

Select Examples: Selection

In Relational Algebra:

$$\sigma_{cond1 \wedge cond2 \vee cond3}(Rel)$$

In SQL:

```
WHERE cond1 AND cond2
      OR cond3
```

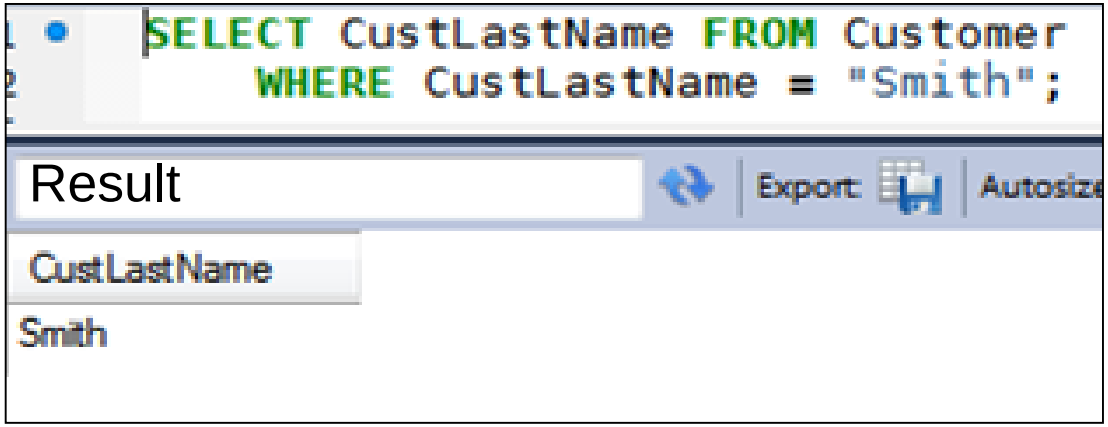
In Relational Algebra:

$$\pi_{CustLastName}(\sigma_{CustLastName="Smith"}(Customer))$$

In SQL:

```
SELECT CustLastName
FROM Customer
WHERE CustLastName = “Smith”;
```


SQL



Select Examples: LIKE clause

- In addition to arithmetic expressions, string conditions are specified with the LIKE clause

LIKE "REG_EXP"

% Represents zero, one, or multiple characters
_ Represents a single character

Examples:

WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and end with "o"

SQL:

```
SELECT CustLastName FROM Customer
  WHERE CustLastName LIKE "Sm%";
```

Export Autosize

CustLastName
Smith
Smithies
Smythe
Smart

Column renaming

We can rename the column name of the output by using the AS clause

```
SELECT CustType, Count(CustomerID)
    FROM Customer
    GROUP BY CustType;
```

Export Autosize

CustType	Count(CustomerID)
Personal	3
Company	6


```
SELECT CustType, Count(CustomerID) AS Count
    FROM Customer
    GROUP BY CustType;
```

Export Autosize

CustType	Count
Personal	3
Company	6

Aggregate Functions

Aggregate functions operate on the (sub)set of values in a column of a relation (table) and return a single value

- AVG()
  - Average value
- MIN()
  - Minimum value
- MAX()
  - Maximum value
- COUNT()
  - Number of values
- SUM()
  - Sum of values

- Plus others
  - http://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html

- All of these except for COUNT(*) ignore null values and return null if all values are null. COUNT(*) counts the number of records.

Aggregate Examples: Count/AVG

COUNT() - returns the number of records
AVG() - average of the values

Examples:

```
SELECT COUNT(CustomerID)
FROM Customer;
```

= How many customers do we have (cardinality)


```
SELECT AVG(OutstandingBalance)
FROM Account;
```

= What is the average balance of ALL ACCOUNTS


```
SELECT AVG(OutstandingBalance)
FROM Account
WHERE CustomerID= 1;
```

= What is the average balance of Accounts of Customer 1

```
SELECT AVG(OutstandingBalance)
FROM Account
GROUP BY CustomerID;
```

= What is the average balance **PER CUSTOMER**

GROUP BY clause

- **Group by** groups all records together over a set of attributes
- Frequently used with aggregate functions

- **Example:**

  *What is the average balance **PER CUSTOMER***


```
SELECT AVG(OutstandingBalance)
FROM Account
GROUP BY CustomerID;
```

Returns one record per each customer

HAVING Clause

- 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

ORDER BY Clause

- Orders records by particular column(s)

ORDER BY XXX ASC/DESC (ASC is default)

SQL

```
SELECT CustLastName, CustType
FROM Customer
ORDER BY CustLastName;
```

RESULT

CustLastName	CustType
Jones	Company
Jones	Company
Lam	Personal
Samson	Personal
Smart	Company
Smith	Personal
Smithies	Company
Smythe	Company
Unila	Company

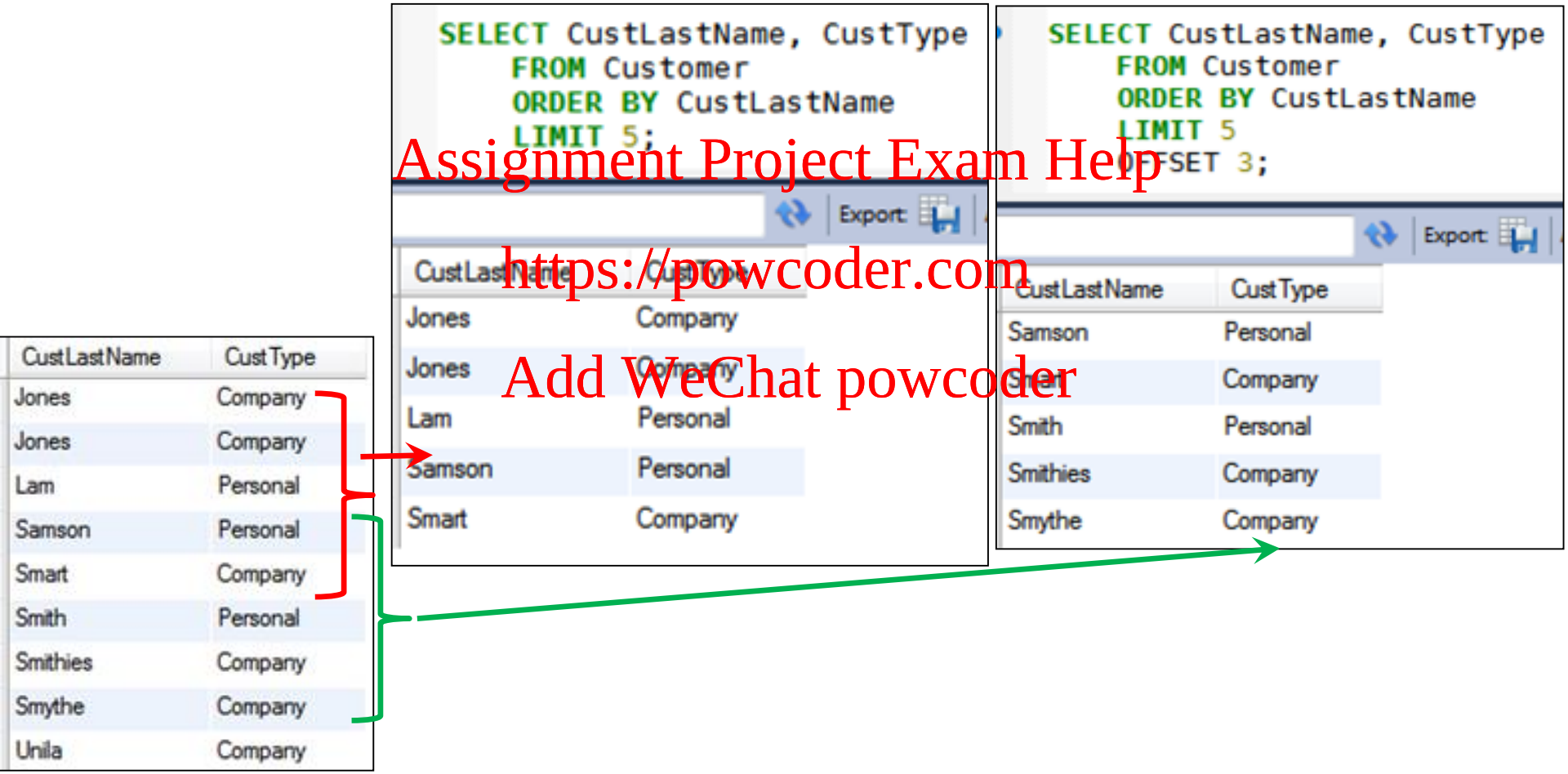
```
SELECT CustLastName, CustType
FROM Customer
ORDER BY CustLastName DESC;
```

Export:

CustLastName	CustType
Unila	Company
Smythe	Company
Smithies	Company
Smith	Personal
Smart	Company
Samson	Personal
Lam	Personal
Jones	Company
Jones	Company

Limit and Offset

- LIMIT number - limits the output size
- OFFSET number - skips first 'number' records



Joining tables together

- SELECT * FROM **Rel1, Rel2**; - this is a **cross product**

```
SELECT * FROM Customer, Account;
```

Export: Autosize:

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance	CustomerID
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25	1
2	James	NULL	Jones	JJ Enterprises	Company	1	Peter Smith	245.25	1
3	Akin	NULL	Smithies	Bay Wart	Company	1	Peter Smith	245.25	1
1	Peter	NULL	Smith	NULL	Personal	2	JJ ENt.	552.39	2
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ ENt.	552.39	2
3	Akin	NULL	Smithies	Bay Wart	Company	2	JJ ENt.	552.39	2
1	Peter	NULL	Smith	NULL	Personal	3	JJ ENt. Mgr	10.25	2
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ ENt. Mgr	10.25	2
3	Akin	NULL	Smithies	Bay Wart	Company	3	JJ ENt. Mgr	10.25	2

Not quite useful…

Typically we would like to find:

For every record in the Customer table list every record in the Account table

Joins: Different Types

- **Inner/Equi join**:
  - Joins the tables over keys

```
SELECT * FROM Customer INNER JOIN Account
  ON Customer.CustomerID = Account.CustomerID;
```

CONDITION

Export: Autosize:

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance	CustomerID
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25	1
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ ENt.	552.39	2
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ ENt. Mgr	10.25	2

- **Natural Join:**
  - Joins the tables over keys. The condition does not have to be specified (natural join does it automatically), but key attributes have to have the *same name.*


```
SELECT * FROM Customer NATURAL JOIN Account;
```

Export: Autosize:

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ ENt.	552.39
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ ENt. Mgr	10.25

Joins: Different Types

- **Outer join**:
  - Joins the tables over keys
  - Can be *left* or *right* (see difference below)
  - Includes records that **don't match** the join from the other table

```
SELECT * FROM Customer LEFT OUTER JOIN Account
    ON Customer.CustomerID = Account.CustomerID;
```

Export: Autosize:

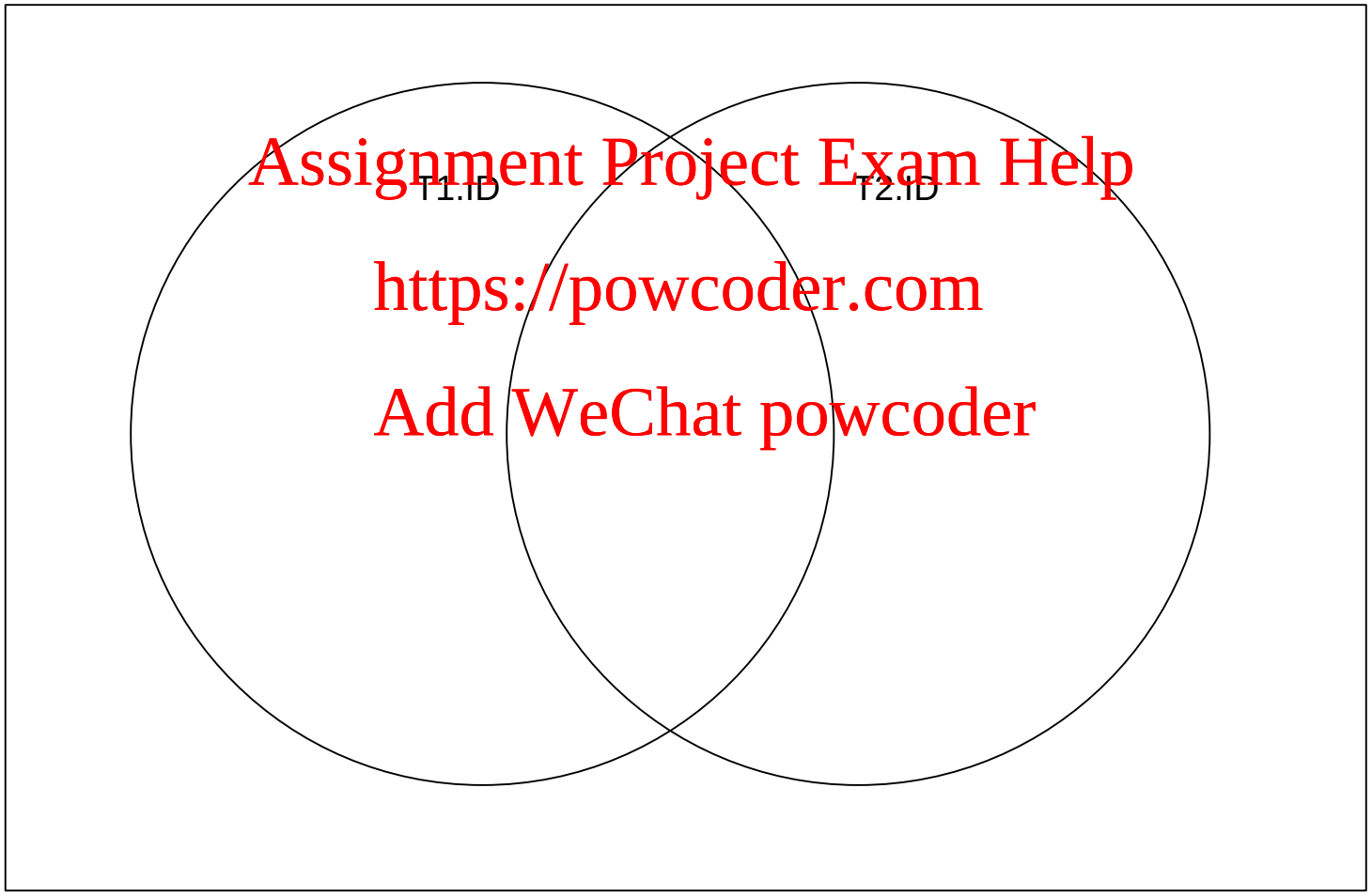
CustomerID	CustFirstName	CustMiddleNa	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance	CustomerID
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25	1
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ ENt.	552.39	2
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ ENt. Mgr	10.25	2
3	Akin	NULL	Smithies	Bay Wart	Company	NULL	NULL	NULL	NULL


```
SELECT * FROM Customer RIGHT OUTER JOIN Account
    ON Customer.CustomerID = Account.CustomerID;
```

Export: Autosize:

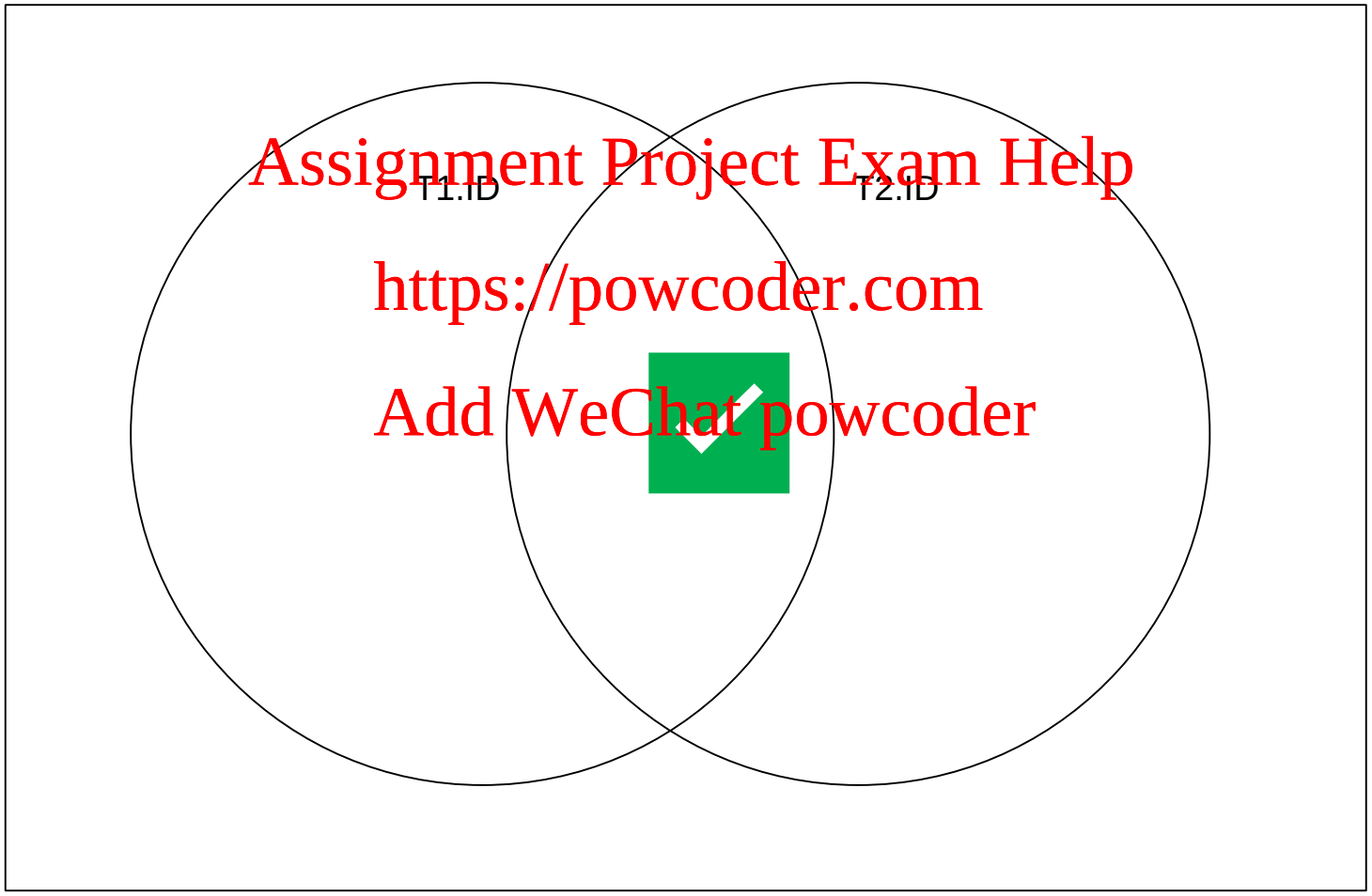
CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance	CustomerID
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25	1
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ ENt.	552.39	2
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ ENt. Mgr	10.25	2

JOINS depicted as Venn Diagrams



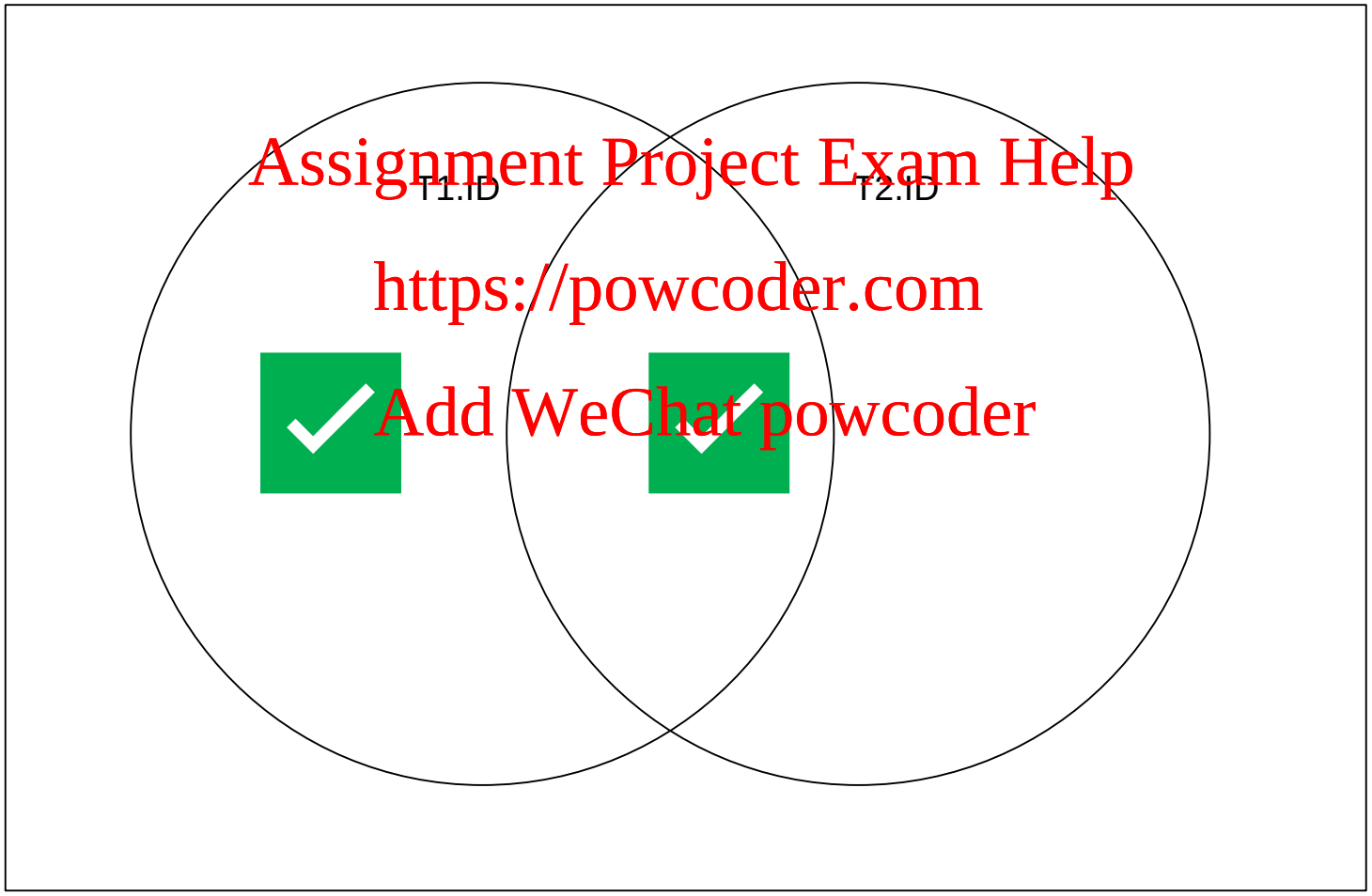
JOINS depicted as Venn Diagrams

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



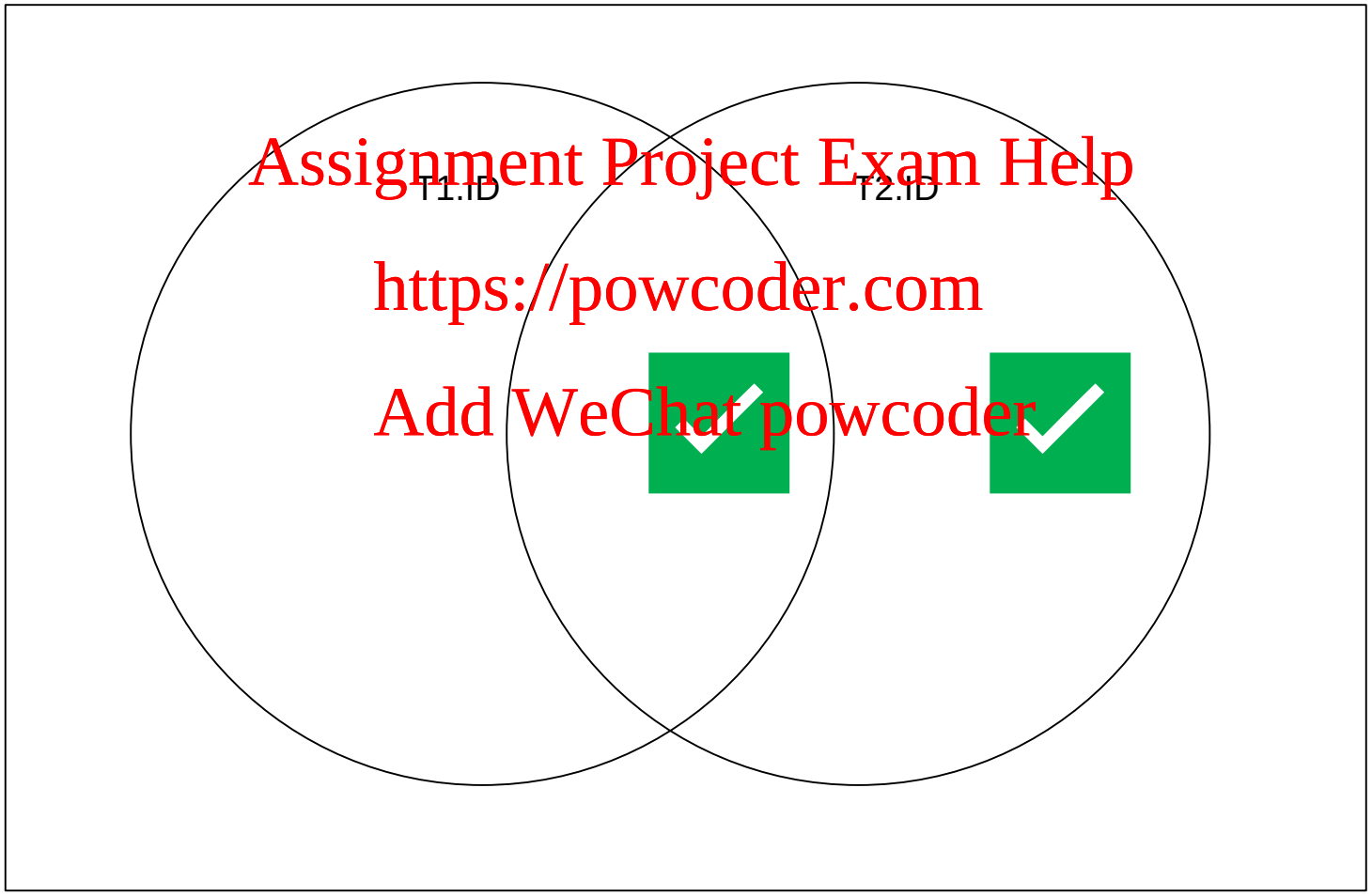
JOINS depicted as Venn Diagrams

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



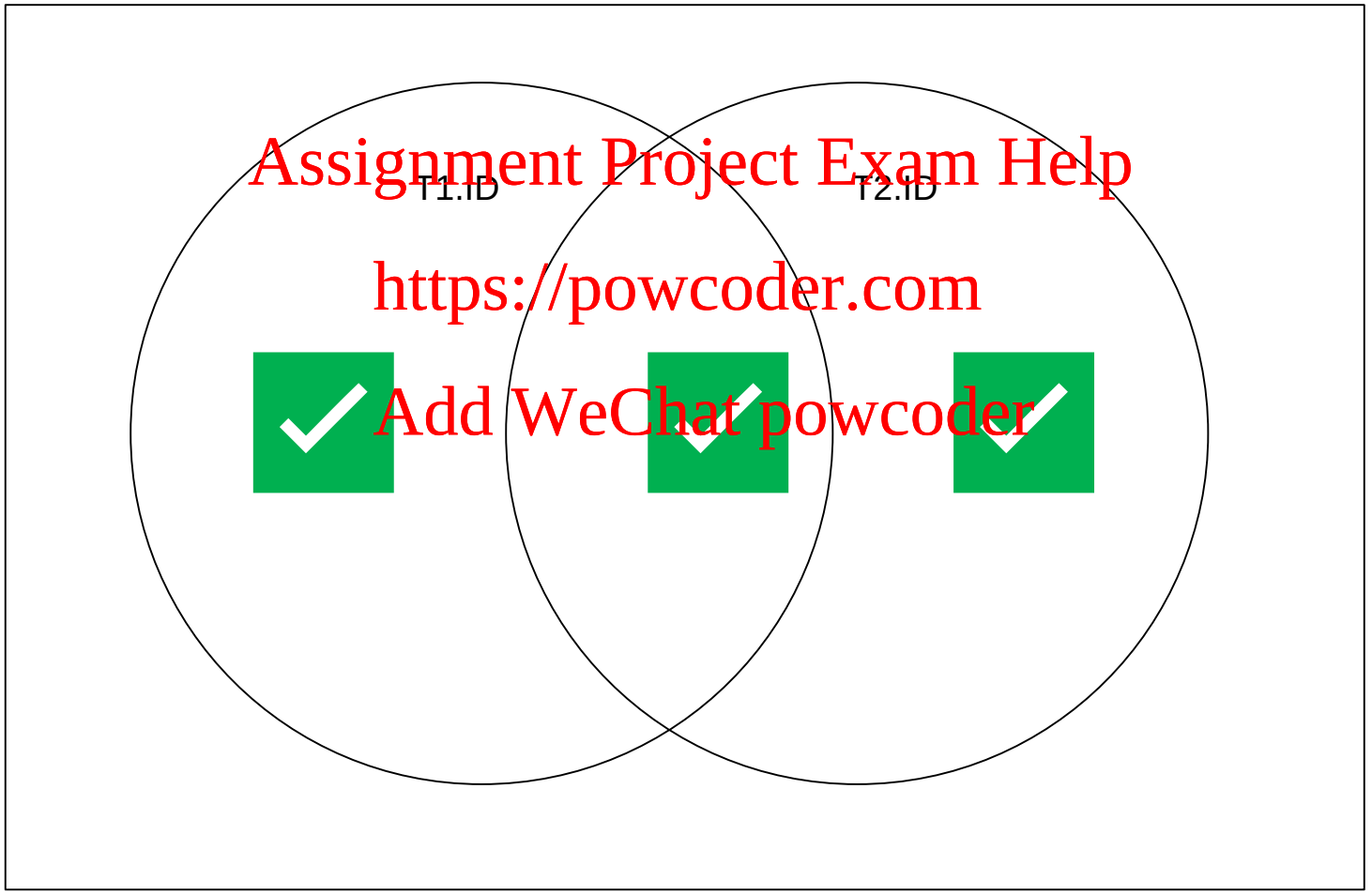
JOINS depicted as Venn Diagrams

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



JOINS depicted as Venn Diagrams

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



What's examinable

- You need to know how to write SQL
  - DDL
  - DML

Next Lecture

- SQL Summary
  - Overview of concepts, more examples