THE UNIVERSITY OF MELBOURNE

SQL Part 1

Database Systems & Information Modelling INFO90002.

Week 4 - SQL

Dr Tanya Linden
Dr Renata Borovica-Gajic
David Eccles

Changing your INFO90002db password

Student passwords are the username and then the year

E.g. jwu3 username would be jwu3_2024 password

To change your password:

- Connect to the INFO90002db server (you will need the VPN outside lab computers)

SYNTAX

```
SET PASSWORD = 'newpassw0rd';
```

Installing the Labs Script

rename 9999 → to studentID

BYOD (both workbench and MySQL Server are on the same machine)

The install script for the labs has two versions

Use labs2018byod.sql if you are using your own server, on your own computer

Use labs2018eng.sql if you are using the server on info90002db.eng.unimelb.edu.au

Note 1: students do not have permissions to run CREATE DATABASE command on the engineering server

Note 2: do not install assignment 2 scripts before reading instructions (you will regret it ;-)

What is SQL

- SQL – or "*sequel*" is a language used in relational databases
- ALL DBMS support CRUD
  - Create, Read, Update, Delete commands
- SQL supports CRUD
  - CREATE, SELECT, INSERT, UPDATE, DELETE, DROP 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
- W3Schools has a good self-paced tutorial and examples
  - https://www.w3schools.com/sql/default.asp

dangerous

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
    - START TRANSACTION
    - BEGIN, END

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

THE UNIVERSITY OF MELBOURNE

CREATE, INSERT, NULL

SQL DDL and DML

SQL in Development Process

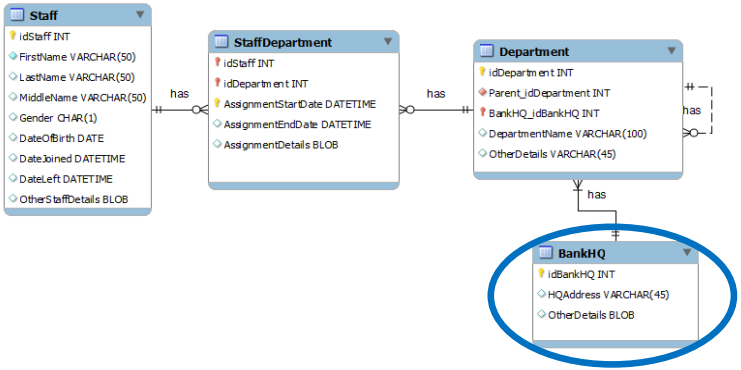
1.
```
CREATE TABLE BankHQ (
     BankHQID                  INT                  AUTO_INCREMENT,
     HQAddress                 VARCHAR(120)         NOT NULL,
     OtherHQDetails            VARCHAR(100),
PRIMARY KEY(BankHQID)
);
```

2.
```
INSERT INTO BankHQ VALUES
(1, "23 Charles St Peterson North 2022", "Main Branch");
INSERT INTO BankHQ VALUES
(2, "213 Jones Rd Parkville North 2122", "Sub Branch");
```

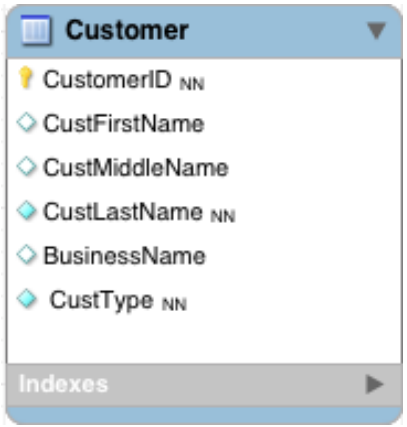
You have to input every column

3.
```
SELECT *
FROM BankHQ;
```

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



CREATE Table and Viewing Table Definition




```
CREATE TABLE Customer (
     CustomerID              INT                          AUTO_INCREMENT,
     CustFirstName           VARCHAR(12),
     CustMiddleName          VARCHAR(14),
     CustLastName            VARCHAR(20)                  NOT NULL,
     BusinessName            VARCHAR(100),
     CustType                ENUM("Personal","Company")   NOT NULL,
PRIMARY KEY(CustomerID)
);
```

To check what you created use one of the following:

```
DESCRIBE Customer;
DESC Customer;
SHOW COLUMNS FROM Customer;
```

see the describe

the result will be the same.

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	NULL	auto_increment
CustFirstName	varchar(12)	YES		NULL	
CustMiddleName	varchar(14)	YES		NULL	
CustLastName	varchar(20)	NO		NULL	
BusinessName	varchar(100)	YES		NULL	
CustType	enum('Personal','Company')	NO		NULL	

Insert Data

ID will automatic added

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

Specifies which columns will be entered; ID will be auto-increment

value not available

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

the system will check

no value

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

No column specification means ALL columns need to be entered

empty string is still text → is a value

Customer

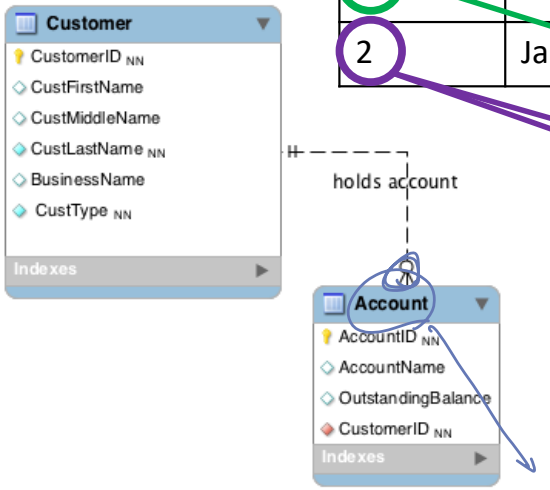
CustID	CustomerFirstName	CustMiddleName	CustLastName	BusinessName	CustType
1	Peter	NULL	Smith	NULL	Personal
2	James	NULL	Jones	JJ Enterprises	Company
3		NULL	Smythe		Company

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

Foreign keys: Review

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

CustID	CustomerFirst Name	CustMiddle Name	CustLastNa me	BusinessName	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company



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

is strong entity. but have FK(customer)

CREATE Statement (with FK)

In SQL, the `ON DELETE RESTRICT` constraint is used to prevent the deletion of a row from a parent table if there are corresponding rows in a child table that reference it. This constraint ensures referential integrity by enforcing that the parent row cannot be deleted as long as there are dependent rows in the child table.

```
CREATE TABLE Account (
     AccountID              INT                    AUTO_INCREMENT,
     AccountName            VARCHAR(12),
     OutstandingBalance     DECIMAL(10,2)          NOT NULL,
     CustomerID             INT                    NOT NULL,
PRIMARY KEY (AccountID),
FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
       ON DELETE RESTRICT
       ON UPDATE CASCADE
)
```

primary key in customer

THE UNIVERSITY OF MELBOURNE

SELECT

SQL DML

The SELECT Statement

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* }, ...]
- Indicate categorisation of results

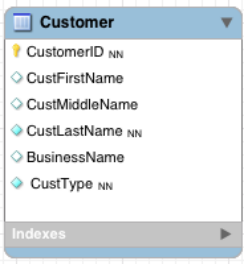
[HAVING *where_condition*]
- Indicate the conditions under which a particular category (group) is included in the result

[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 (i.e. 5 rows, 5 rows from row 2)

Order is important! E.g. HAVING **cannot** go before GROUP BY or WHERE

SELECT Examples



SELECT * FROM Customer;
= Extract all data about customers

SQL

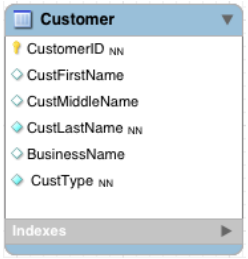
```
SELECT *
FROM Customer
```

RESULT SET

	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

SELECT Examples: Projection

Projection means specifying which attributes to show in the results set.



In SQL:

```
SELECT CustLastName
FROM Customer;
```

in the instruction seperate by comma

end of instruction

Result set

CustLastName
Smith
Jones
Smithies
Smythe
Smart
Lam
Unila
Jones
Samson

SELECT Examples: Selection

In SQL:

```
SELECT CustLastName
FROM Customer
WHERE CustLastName = "Smith";
```

Selection means specifying conditions that records in the output must meet.

Result

CustLastName
Smith

SELECT Examples: LIKE clause

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

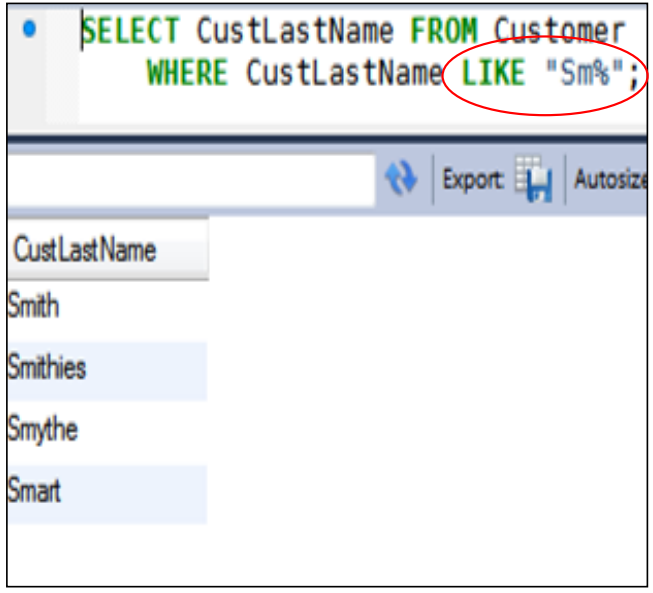
LIKE "REG_EXP"

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

Examples:

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

SQL:



Aggregate Functions

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

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

- N.B. All of these except for COUNT() ignore null values and return null if all values are null. COUNT() counts the rows not the values and thus even if the value is NULL. it is still counted
- COUNT(Column) versus COUNT(*)
- List of MySQL aggregate functions

  https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html

Aggregate Examples: COUNT and 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 with ID 1

where use in attribute

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

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

GROUP BY / HAVING

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;
```

The only way to put a selection condition over a group by statement is by using **having** clause

Example:

What is the exact average balance per customer for customers whose average balance is under 10000?

```
SELECT AVG(OutstandingBalance)
FROM Account
GROUP BY CustomerID
HAVING AVG(OutstandingBalance) < 10000
```

for aggregate function you cannot use where.

Changing Column Heading in Output - Alias

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

If it contains a gap, it must be in straight double quotes

```
SELECT custtype, COUNT(customerid)
FROM customer
GROUP BY custtype;
```

custtype	COUNT(customerid)
Personal	1
Company	2

more than two words use ""

alias to username

```
SELECT custtype AS "Customer Type", COUNT(customerid) AS CUST_TOTAL
FROM customer
GROUP BY custtype;
```

Customer Type	CUST_TOTAL
Personal	1
Company	2

ORDER BY

sort your result by particular order

- Orders records by particular column(s)

  ORDER BY colName ASC/DESC (ASC is default)

```
SELECT firstname, lastname
FROM employee
ORDER BY lastname;
```

by default ascend

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

```
SELECT firstname, lastname, departmentid
FROM employee
ORDER BY departmentid DESC, lastname ASC;
```

first order

second order

ASC

DESC

optional to write

firstname	lastname	departmentid
Andrew	Jackson	11
Ned	Kelly	11
Clare	Underwood	11
Sophie	Monk	10
Sarah	Fergusson	9
Brier	Patch	9
Todd	Beamer	8
Nancy	Cartwright	8
Mark	Zhang	7
Sanjay	Patel	6
Pat	Clarkson	5
Paul	Innit	4
James	Mason	4
Gigi	Montez	3
Maggie	Smith	3
Rita	Skeeter	2
Alice	Munro	1

LIMIT and OFFSET

LIMIT N - limits the output size

OFFSET N - skips first N records

```
SELECT firstname, lastname, departmentid
FROM employee
ORDER BY departmentid DESC, lastname ASC
LIMIT 6;
```

firstname	lastname	departmentid
Andrew	Jackson	11
Ned	Kelly	11
Clare	Underwood	11
Sophie	Monk	10
Sarah	Fergusson	9
Brier	Patch	9

LIMIT 6

firstname	lastname	departmentid
Andrew	Jackson	11
Ned	Kelly	11
Clare	Underwood	11
Sophie	Monk	10
Sarah	Fergusson	9
Brier	Patch	9
Todd	Beamer	8
Nancy	Cartwright	8
Mark	Zhang	7
Sanjay	Patel	6
Pat	Clarkson	5
Paul	Innit	4
James	Mason	4
Gigi	Montez	3
Maggie	Smith	3
Rita	Skeeter	2
Alice	Munro	1

```
SELECT firstname, lastname, departmentid
FROM employee
ORDER BY departmentid DESC, lastname ASC
LIMIT 6 OFFSET 3;
```

firstname	lastname	departmentid
Sophie	Monk	10
Sarah	Fergusson	9
Brier	Patch	9
Todd	Beamer	8
Nancy	Cartwright	8
Mark	Zhang	7

LIMIT 6 OFFSET 3

be careful if there are two highest you have to use max( ) function.

THE UNIVERSITY OF MELBOURNE

Joining Tables

Cross Product,
INNER JOIN,
NATURAL JOIN,
RIGHT OUTER JOIN,
LEFT OUTER JOIN

Table name qualification and aliases

A simple SQL query using the SELECT statement

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

Whenever we use a column name in a Query, we can qualify (prefix) with the appropriate table name

```
SELECT Customer.CustLastName, Customer.CustType
FROM   Customer;
```

We can use an **alias** for table name

```
SELECT c.CustLastName, c.CustType
FROM   Customer c;
```

The alias only applies to the current SQL statement (not remembered)

Once an alias is used within an SQL statement, you can cannot refer to the original tablename within that SQL statement

The usefulness of this feature becomes apparent when SQL statements refer to columns from two or more tables

Joining tables together

SELECT * FROM **Table1, Table2**; - the result set is a **cross product** or **Cartesian 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

Every row in the **first table** has been matched with **every row** in the **second table**

Not very useful…

Reason: an **SQL Select** statement does **not** 'know' how two tables are **related** (it doesn't consider any existing FK – PK constraints)

Joins: INNER JOIN

An **INNER JOIN** returns a result set that contains only data that satisfies a **Foreign Key – Primary Key condition**

Syntax:

```
SELECT   <column-names>
FROM     <table-name1>
INNER JOIN  <table-name2>
ON       <join-condition>
```

The **<join-condition>** is normally in the format

<foreign-key column-name> = <primary-key column name>

```
SELECT      c.CustomerFirstName, c.CustomerLastName, a.OutstandingBalance
FROM        Customer c
INNER JOIN  Account a
ON          c.CustomerID = a.CustomerID
```

customer with the account won't be shown in the joint 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*.

natural join works slower

```
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

Avoid **old** style Joins

Back in the **dark ages** (pre 2000), some DBMS products **did NOT use/have** the **INNER JOIN** keyword.

Instead the JOIN was expressed as part of the **where clause** within the Select statement

Many **old-time** developers, **old-time** web sites, **old-time** books / authors still use this old-style in their code and in their examples

```
SELECT    c.CustomerFirstName, c.CustomerLastName, a.OutstandingBalance
FROM Customer c, Account a
WHERE c.CustomerID = a.CustomerID
```

Joins: Different Types

inner join:
both have to be satisfy

left
customer is major

right
account is major

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

Includes Customers with associated accounts as well as Customers who do not have Account

Vs

```
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

Customers with associated accounts as well as Accounts without Customers

What's examinable

- **DDL**
- **DML**
- **SELECT**

THE UNIVERSITY OF MELBOURNE

Thank you