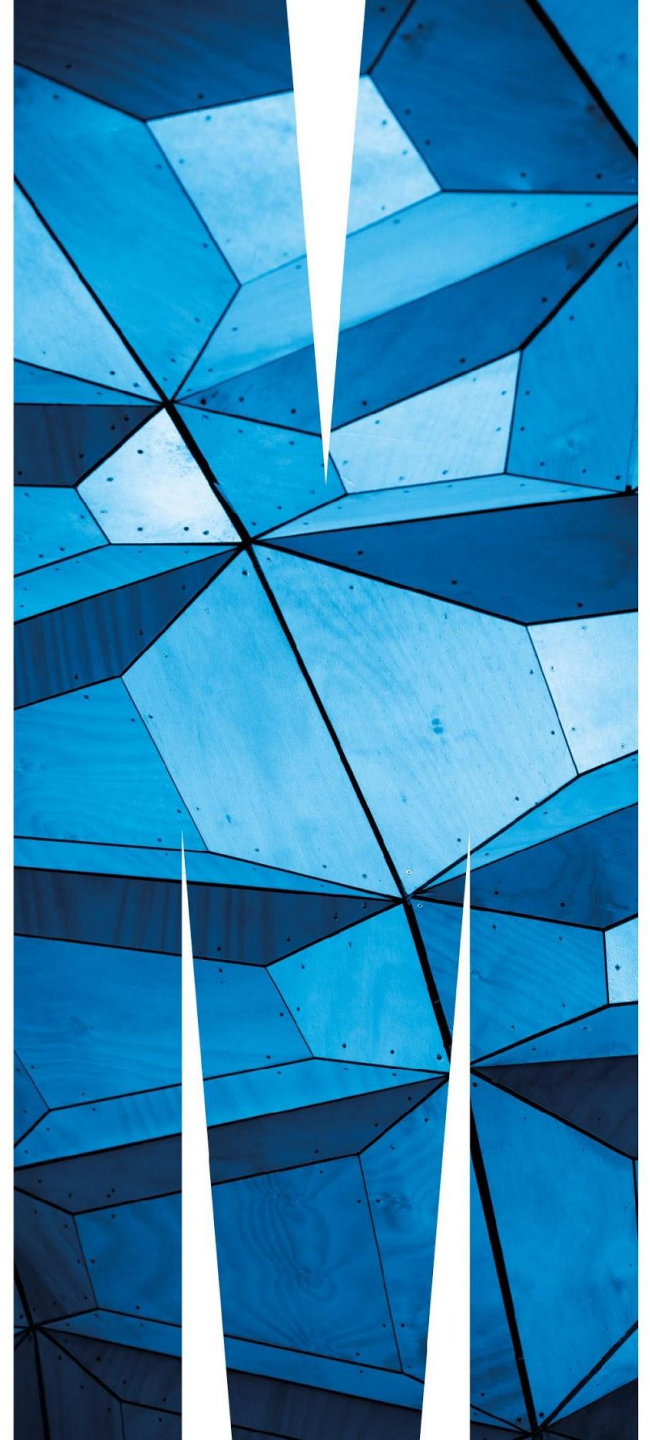


Week 7 - Structured Query Language (SQL) – Part 1

FIT2094 - FIT3171 Databases
Clayton Campus S2 2019.



Overview

▪ Hour 1

–SQL SELECT

- Basics
- Predicate
- Math
- Refining the query results...

... then COFFEE BREAK!

▪ Hour 2

–SQL SELECT cont'd

- Joining
- Oracle Dates

–Actual practice

[Clayton] Marc's anecdote



[Sign in or Register](#) | [Employer site](#)

Job Search

\$150k+ Jobs

Profile

Company Reviews

Career Advice

What

database



Any Classification



Where

Enter suburb, city, or region

SEEK

All work types

paying \$0

to \$200k+

listed any time

7,356 jobs found



[Sign in or Register](#) | [Employer site](#)

Job Search

\$150k+ Jobs

Profile

Company Reviews

Career Advice

What

SQL



Any Classification



Where

Enter suburb, city, or region

SEEK

All work types

paying \$0

to \$200k+

listed any time

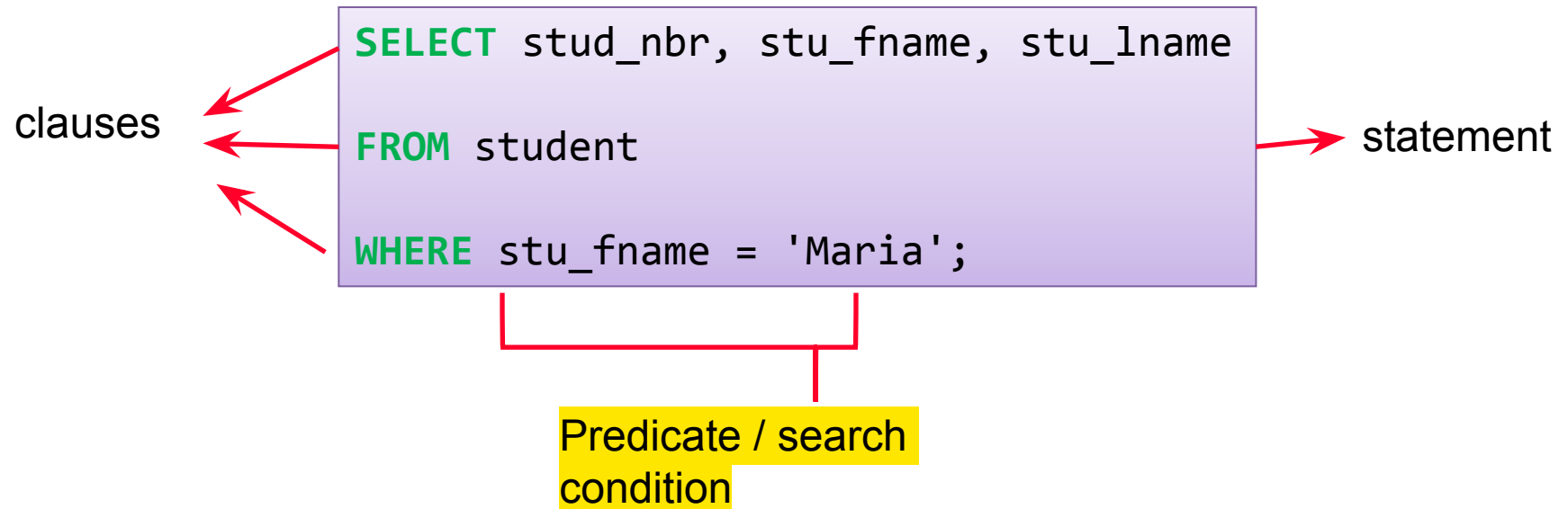
3,841 jobs found

Sorted by **relevance**

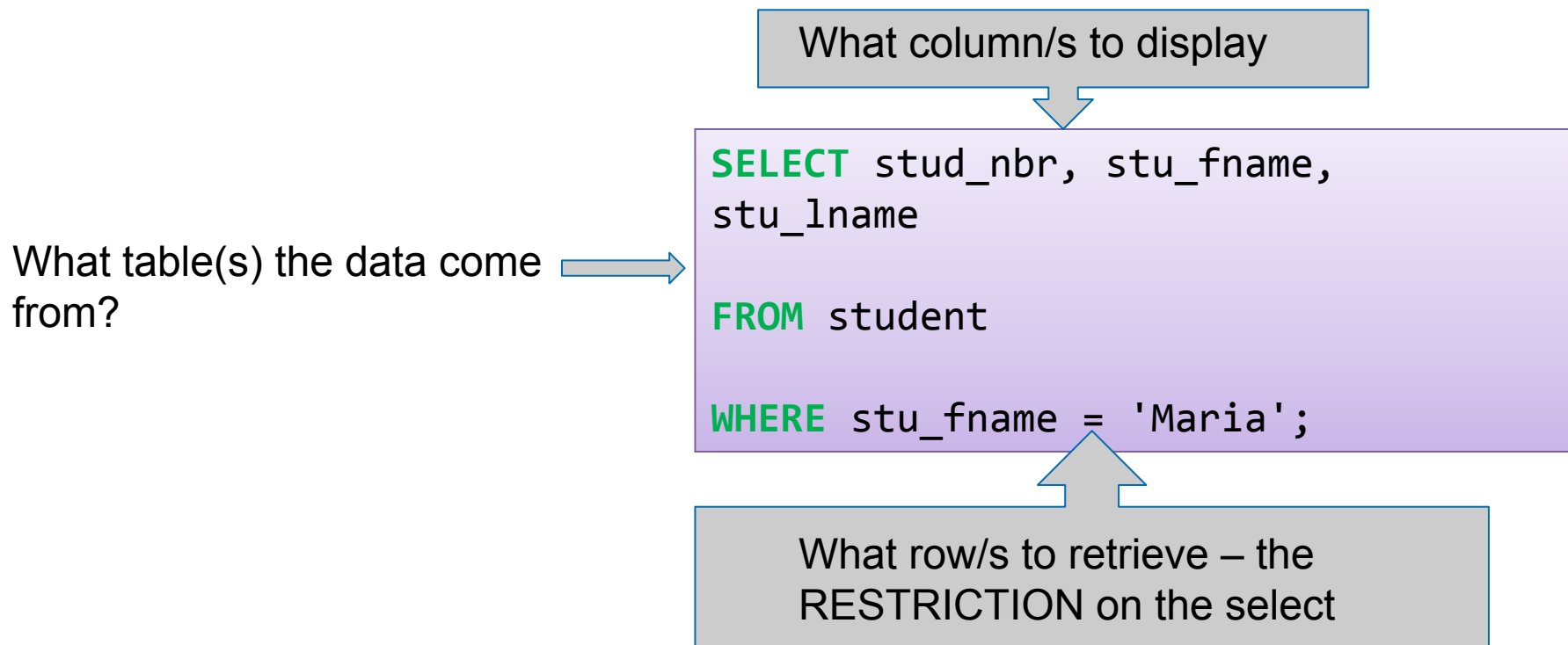


SELECT... **(the DQL part of SQL)**

Anatomy of an SQL SELECT Statement



SQL SELECT Statement - Usage



Remember: some of the SQL statement syntaxes in the unit are ORACLE SPECIFIC

**These are transferable skills to other DBMS 'brands'...
... however code might not work 'out of the box' for other 'brands'.**

Refer to Coronel & Morris for examples of how certain syntax differs from 'brand' to 'brand' e.g. Oracle vs MySQL

SQL Predicates or Search Conditions

- The search conditions are applied on each row, and the row is returned if the search conditions are evaluated to be TRUE for that row.
- **Comparison**
 - Compare the value of one expression to the value of another expression.
 - Operators (note: = not the same as prog. lang.'s double ==)
 - =, <, >, <=, >=, !=
 - Example: salary > 5000
- **Range**
 - Test whether the value of an expression falls within a specified range of values.
 - Operators:
 - BETWEEN
 - Example: salary BETWEEN 1000 AND 3000 (both are inclusive)

SQL Predicates or Search Conditions

- **Set Membership**

- To test whether the value of expression equals one of a set of values.
- Operator:
 - **IN**
- Example: `city IN ('Melbourne', 'Sydney')`

- **Pattern Match**

- To test whether a string (text) matches a specified pattern.
- Operator:
 - **LIKE**
- Patterns:
 - % character represents any sequence of zero or more character.
 - _ character represents any single character.
- Example:
 - `WHERE city LIKE 'M%'`
 - `WHERE unit_code LIKE 'FIT20__'`

SQL Predicates or Search Conditions

- **NULL**

- To test whether a column has a NULL (unknown) value.

- Example: `WHERE grade IS NULL`

- Use in subquery (**to be discussed in the future**)

- ANY, ALL (... subquery ...)

- EXISTS (... subquery ...)

What row will be retrieved?

- Predicate evaluation is done using three-valued logic.
 - **TRUE**, **FALSE** and **UNKNOWN**
- DBMS will evaluate the predicate against each row.
- Row that is evaluated to be **TRUE** will be retrieved.
- **NULL** is considered to be **UNKNOWN**.

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(null)
8	11111114	FIT1004	2013	1	(null)	(null)

Q1. Consider the predicate "enrol_mark >= 50", what row(s) will be selected for this predicate by the DBMS?

- a. 1, 4 and 6
- b. All rows
- c. 1 and 6
- d. All rows except row 4

Combining Predicates

- Logical operators
 - AND, OR, NOT
- Rules:
 - An expression is evaluated LEFT to RIGHT.
 - Sub-expression in brackets are evaluated first.
 - NOTs are evaluated before AND and OR
 - ANDs are evaluated before OR.
 - NB: similar to many prog. languages.

Truth Table

- **AND** is evaluated to be TRUE if and only if **both conditions** are TRUE
- **OR** is evaluated to be TRUE if and only if **at least one** of the conditions is TRUE

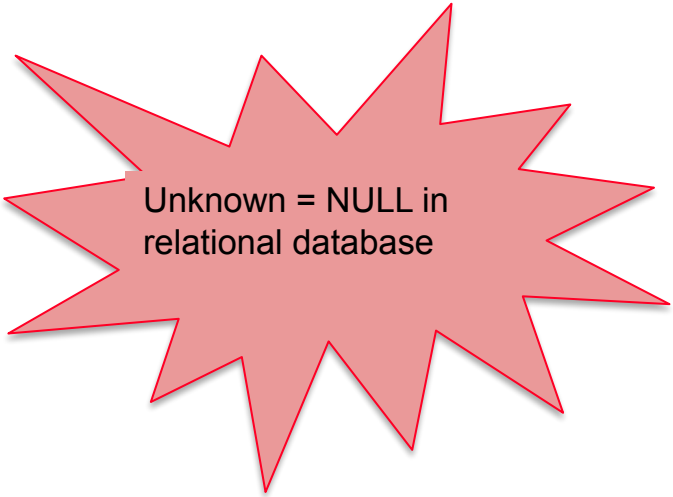
AND

A \ B	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

T = TRUE
F = FALSE
U = Unknown

OR

A \ B	T	U	F
T	T	T	T
U	T	U	U
F	T	U	F



Unknown = NULL in relational database

Q2. What row will be retrieved when the WHERE clause predicate is written as

V_CODE = 21344 AND V_CODE = 24288

	V_CODE
1	21344
2	20001
3	24288
4	20001
5	24288

a. 1,3,5

b. 1

c. 3,5

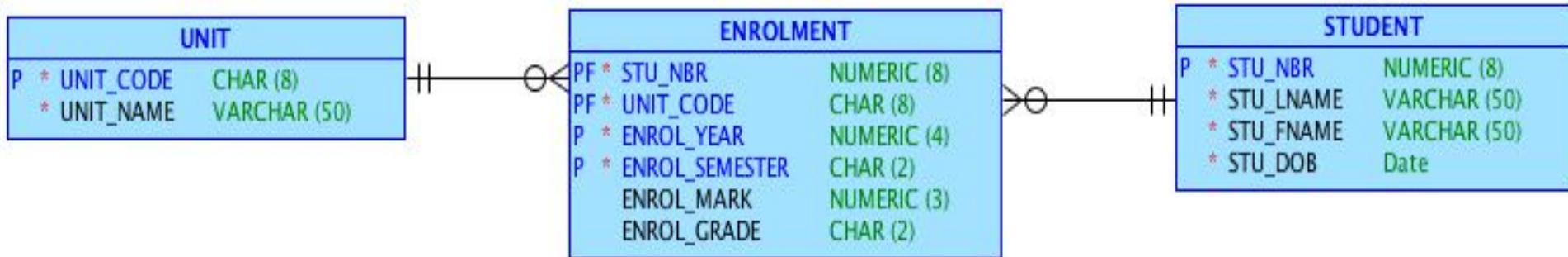
d. No rows will be retrieved

Q3. What row will be retrieved when the WHERE clause predicate is written as

V_CODE <> 21344 OR V_CODE <> 24288

	V_CODE
1	21344
2	20001
3	24288
4	20001
5	24288

- a. 1,3,5
- b. 2,4
- c. 3,5
- d. 1,2,3,4,5



SELECT * FROM UNIT;

Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0.015 seconds

	UNIT_CODE	UNIT_NAME
1	FIT1001	Computer Systems
2	FIT1002	Computer Programming
3	FIT1004	Database

SELECT * FROM STUDENT;

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.022 seconds

	STU_NBR	STU_LNAME	STU_FNAME	STU_DOB
1	11111111	Bloggs	Fred	01/JAN/90
2	11111112	Nice	Nick	10/OCT/94
3	11111113	Wheat	Wendy	05/MAY/90
4	11111114	Sheen	Cindy	25/DEC/96

SELECT * FROM ENROLMENT;

Script Output x Query Result x

SQL | All Rows Fetched: 8 in 0.016 seconds

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(null)
8	11111114	FIT1004	2013	1	(null)	(null)

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(null)
8	11111114	FIT1004	2013	1	(null)	(null)

Q4. What is the correct SQL predicate to retrieve those students who have passed **and also those students who have not been awarded any mark?**

- a. enrol_mark >= 50 AND enrol_mark IS NULL
- b. enrol_mark >= 50 OR enrol_mark IS NULL**
- c. enrol_mark >= 50 AND enrol_mark IS NOT NULL
- d. enrol_mark >= 50 OR enrol_mark IS NOT NULL
- e. None of the above

Arithmetic Operations

- Can be performed in SQL.
- For example:

```
SELECT stu_nbr, enrol_mark/10  
FROM enrolment;
```

	STU_NBR	ENROL_MARK/10
1	11111111	7.8
2	11111111	(null)
3	11111111	(null)
4	11111112	3.5
5	11111112	(null)
6	11111113	6.5
7	11111113	(null)
8	11111114	(null)

Oracle NVL function

- It is used to replace a NULL with a value.

```
SELECT stu_nbr,  
       NVL(enrol_mark,0),  
       NVL(enrol_grade,'WH')  
FROM enrolment;
```

	STU_NBR	NVL(ENROL_MARK,0)	NVL(ENROL_GRADE,'WH')
1	11111111	78	D
2	11111111	0	WH
3	11111111	0	WH
4	11111112	35	N
5	11111112	0	WH
6	11111113	65	C
7	11111113	0	WH
8	11111114	0	WH

ADVANCED question...

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(null)
8	11111114	FIT1004	2013	1	(null)	(null)

Time to put it all together.

In a single **SELECT** statement, how do you produce -- the student number, unit code, and mark MINUS 5 -- for all **FAILED** students, and all students who don't have marks should get exactly 1 mark.

Sample output →

11111111	FIT1002	1
11111111	FIT1004	1
11111112	FIT1001	30
11111112	FIT1001	1
11111113	FIT1004	1
11111114	FIT1004	1

AS - Column Aliases

	STU_NBR	NVL(ENROL_MARK,0)	NVL(ENROL_GRADE,'WH')
1	11111111	780	

- Note column headings with functions - not intuitive!
- Use the word "AS" to specify a **column alias**
 - New column name in " " to maintain case or spacing
 - One limitation in the next few slides.
- Example...

```
SELECT stu_nbr, enrol_mark/10 AS new_mark  
FROM enrolment;
```

```
SELECT stu_nbr, enrol_mark/10 AS "New Mark"  
FROM enrolment;
```


Sorting Query Result

- "ORDER BY" clause – *tuples have no order*
 - Must be used if more than one row may be returned
- Order can be ASCending or DESCending. The default is ASCending.
 - NULL values can be explicitly placed first/last using "NULLS LAST" or "NULLS FIRST" command
- Sorting can be done for multiple columns.
 - order of the sorting is specified for each column.
- Example:

```
SELECT stu_nbr, enrol_mark  
FROM enrolment  
ORDER BY enrol_mark DESC
```

	STU_NBR	ENROL_MARK
1	11111111	(null)
2	11111111	(null)
3	11111114	(null)
4	11111112	(null)
5	11111113	(null)
6	11111111	78
7	11111113	65
8	11111112	35

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(null)
8	11111114	FIT1004	2013	1	(null)	(null)

Q5. What will be the output of the following SQL statement?

```
SELECT stu_nbr
FROM enrolment
WHERE enrol_mark IS NULL
ORDER BY stu_nbr ASC NULLS FIRST;
```

- a.

11111111
11111112
11111113
11111114
- b.

11111111
11111111
11111112
11111113
11111114
- c.

11111111
11111112
11111113
- d.

(null)
(null)
(null)
(null)
(null)

COMMON PROBLEM - take note!

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(null)
8	11111114	FIT1004	2013	1	(null)	(null)

```
SELECT stu_nbr AS alias_id, (year-2000) AS alias_short_year
FROM enrolment
WHERE alias_short_year > 09 AND alias_short_year < 13
ORDER BY alias_id DESC;
-- assume we want students from the graduate class of '10, '11, '12 only.
```

This SQL code will FAIL!
Why?

**A column alias (AS) can be used in ORDER BY...,
but cannot be used in WHERE...**

Quick fix: subqueries (seen in future)

Removing Duplicate Rows in the Query Result

- Use "DISTINCT" as part of SELECT clause.

```
SELECT DISTINCT stu_nbr  
FROM enrolment  
WHERE enrol_mark IS NULL;
```

	STU_NBR
1	11111114
2	11111111
3	11111112
4	11111113



Coffee break - see you in 10 minutes.

SQL JOIN: simplified example

STUDENT

sno	name
1	alex
2	maria
3	bob

QUALIFICATION

sno	degree	year
1	bachelor	1990
1	master	2000
2	PhD	2001

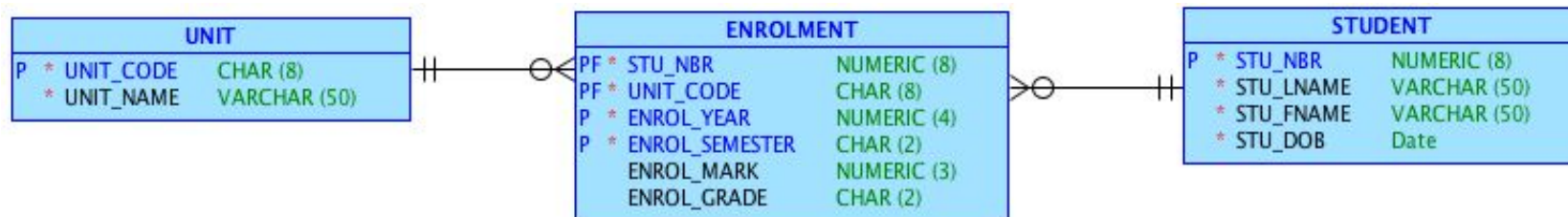
```
SELECT *  
FROM student JOIN qualification  
          ON student.sno = qualification.sno  
ORDER BY student.sno
```

sno	name	degree	year
1	alex	bachelor	1990
1	alex	master	2000
2	maria	PhD	2001

JOIN-ing Multiple Tables

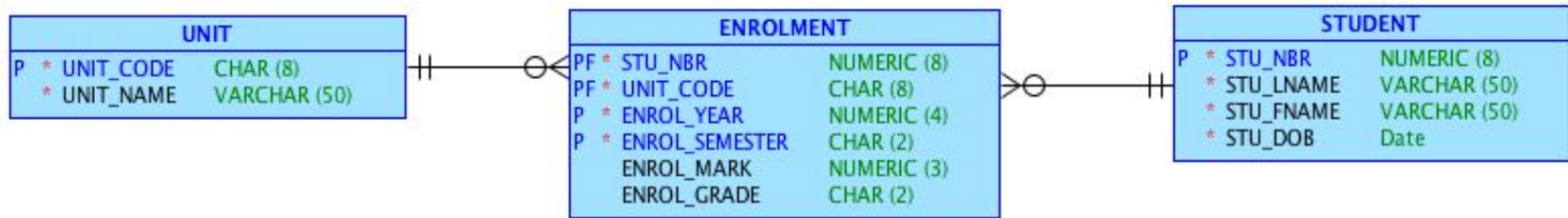
Pair the PK and FK in the JOIN condition

Note table aliasing e.g. `unit u` in FROM clause



```
SELECT s.stu_nbr, s.stu_lname, u.unit_name
FROM ((unit u JOIN enrolment e ON u.unit_code=e.unit_code)
      JOIN student s ON e.stu_nbr=s.stu_nbr)
ORDER BY s.stu_nbr, u.unit_name;
```


JOIN-ing Multiple Tables - Step by Step



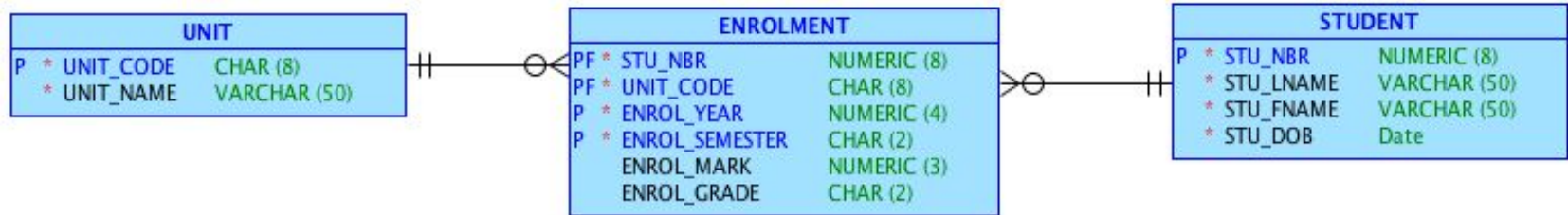
```
SELECT s.stu_nbr, s.stu_lname, u.unit_name
```

unit is now called 'u' as a table alias same goes for enrolment 'e'

```
FROM ((unit u JOIN enrolment e ON u.unit_code=e.unit_code)
      JOIN student s ON e.stu_nbr=s.stu_nbr)
```

```
ORDER BY s.stu_nbr, u.unit_name;
```

[Clayton] Audience Q&A



```
SELECT s.stu_nbr, s.stu_lname, u.unit_name
FROM ((unit u JOIN enrolment e ON u.unit_code=e.unit_code)
      JOIN student s ON e.stu_nbr=s.stu_nbr)
ORDER BY s.stu_nbr, u.unit_name;
```

Time to put it all together.

1. What happens to a new unit (without any enrolments)?
2. What happens to a newly-registered student but who hasn't enrolled in a unit (e.g. pending fees paid)?
3. Any NULLs anywhere?
4. What happens if we do **ONLY the first join** (unit u JOIN enrolment e) without the second join (student s...)



Oracle Reference: Date Data Type

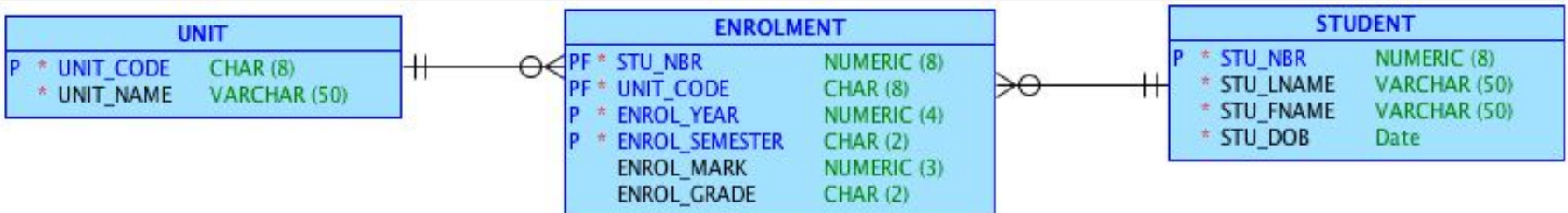
SQL Standard vs Oracle

- Dates are stored differently from the SQL standard
- SQL's standard uses two different types: **date** and **time**
- Oracle uses one type: DATE
 - Stored in internal format contains date and time
 - Output is controlled by formatting
 - select **to_char**(sysdate, 'dd-Mon-yyyy')
from dual;
Result: 14-Apr-2018
 - select **to_char** (sysdate, 'dd-Mon-yyyy
hh:mi:ss PM') **from dual**;
Result: 14-Apr-2018 02:51:24 PM

Oracle: DATE data type

- DATE data type should be formatted with **TO_CHAR** when selecting for **display**.
- Text representing date **must be formatted** with **TO_DATE** when **comparing** or **inserting/updating**.
- Example:

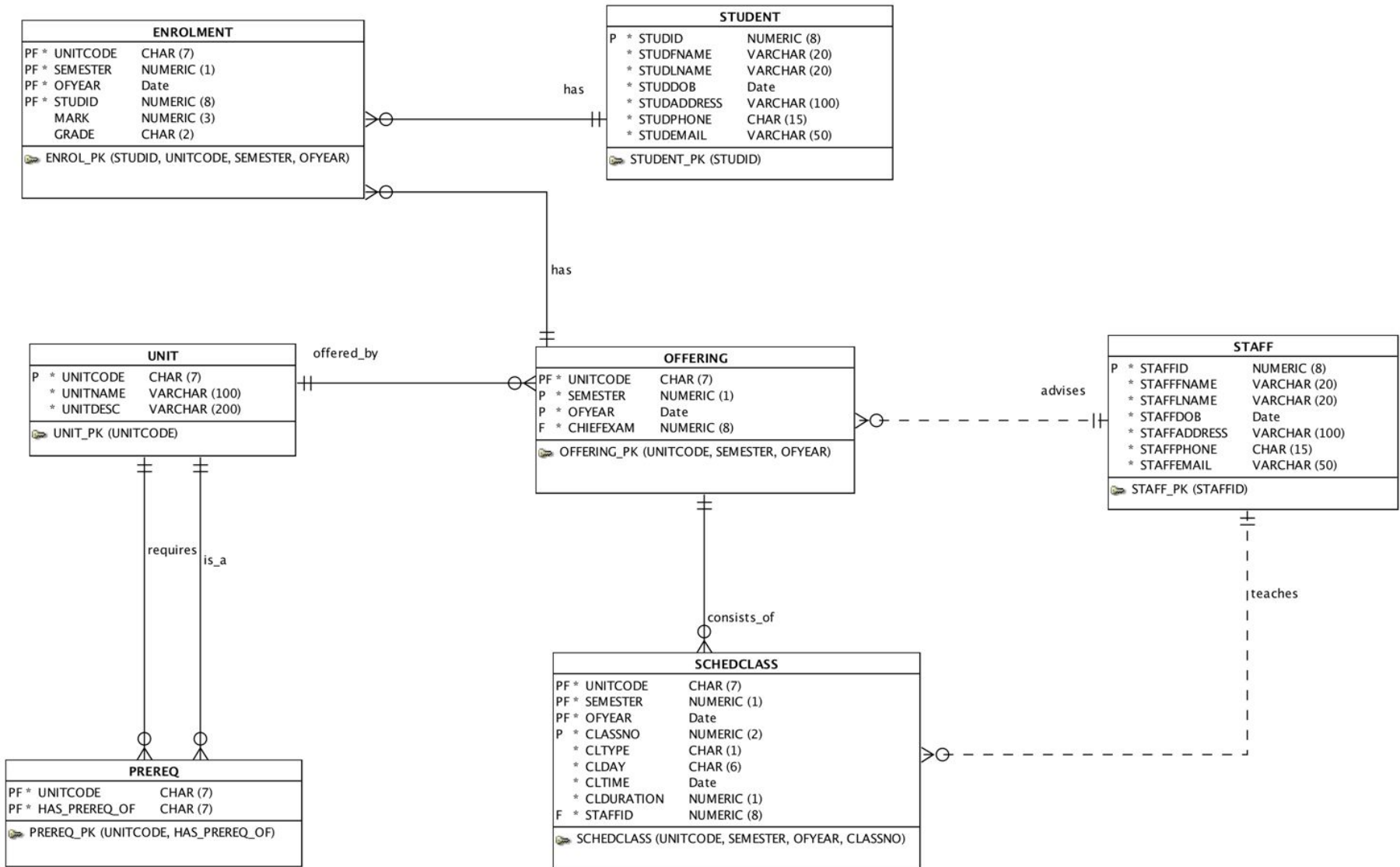
```
SELECT stu_nbr, stu_fname, stu_lname,  
       to_char(stu_dob, 'dd-Mon-yyyy') as text_dob  
FROM   student  
WHERE  stu_dob > to_date('01-Apr-1991', 'dd-Mon-yyyy')  
ORDER BY stu_dob;
```



Oracle: Current Date

- Current date can be queried from the DUAL table using the **SYSDATE** attribute.
 - SELECT **sysdate** FROM dual;
- Oracle internal attributes include:
 - **sysdate**: current date/time
 - **systimestamp**: current date/time as a timestamp
 - **user**: current logged in user

Practice



Practice

- Show the unit codes that have lectures (type = L) scheduled on Mondays (Mon)
- Show names of students and their DOBs where DOB is displayed as something like "01-JAN-1999"
- Show the first name and last name of the students who got HD in FIT1004
- Show unit name, and the names of the students who got HD in any unit that contains the word 'Data' in its name
- Show the names of the unit that have lectures scheduled on Mondays
- Show the names of all students who come to university to attend a lecture on Mondays. We assume an ideal world where a student never misses the scheduled lectures of any unit he/she is enrolled in :P

Summary

- SQL statement, clause, predicate.
- Writing SQL predicates.
 - Comparison, range, set membership, pattern matching, is NULL
 - Combining predicates using logic operators (AND, OR, NOT)
- Arithmetic operation.
 - NVL function
- Column alias.
- Ordering (Sorting) result.
- Removing duplicate rows.
- JOIN-ing tables
- Oracle Dates