MONASH University

MONASH
INFORMATION
TECHNOLOGY

Week 10 - Structured Query Language (SQL) – Part 3 (Advanced)

FIT2094 - FIT3171 Databases

Clayton Campus S2 2019.

GROUP OF EIGHT AUSTRALIA

Overview

- **Hour 1**
  - **–Subquery – nested, inline, correlated**
  - –Views

… then COFFEE BREAK!

- **Hour 2**
  - –Joins - self join, outer join (Recall Rel'n Algebra?)
  - –Set Operators
  - –Oracle Functions

[Clayton] SQL Test Arrangements (reminder)

Your location will depend on your Student ID.
If your Student ID ends in an even number, you will attend the Thursday Test. Alternatively, if your Student ID ends in an odd number you will attend the Friday Test.

Thursday 6PM - 7PM: B25Exh/C1
Friday 6PM - 7PM: B25Exh/C1

- **Don't forget to bring your Student ID card**
- **If you can't make your allocated session, please contact the role account or your tutor to make alternative arrangements.**

[Clayton] SQL Test Arrangements (reminder)

[CLAYTON] Unit Test Scope/Things To Bring
by Marc Cheong - Tuesday, 1 October 2019, 11:32 AM

Just two kind reminders:

The SQL Unit Test in Week 10 covers content from Week 1 **up to and including Week 10's prereading and textbook chapters.**

Please bring your Student ID.
(NB: Also, if you are formally granted a swap, provide a printed copy of the email thread from unit management as evidence).

Reminder: there are extra consultations in MSB and Week 10 to help you out (this week and next week) - check schedule on Moodle.

-- Mgmt

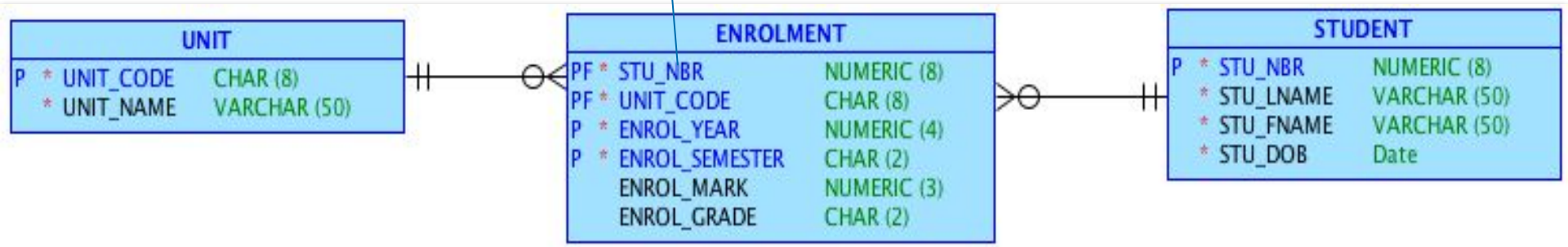
SELECT Subqueries...
(Advanced lessons)

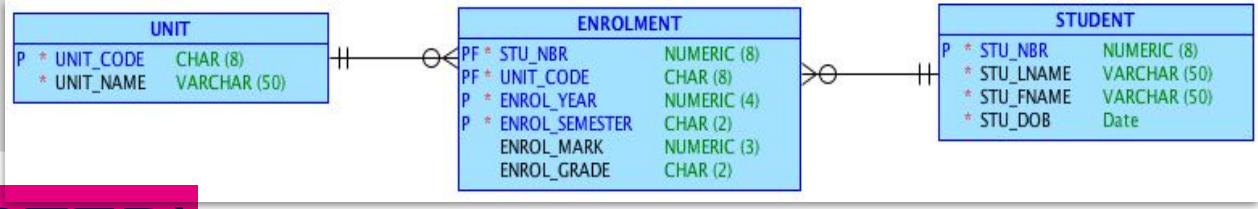
Img src: @samuelzeller at Unsplash / icons8

Case study - our usual (and simple) STUDENT-UNIT-ENROLMENT seen in Lecture 7…

… with more values added for this demo.

For each unit, find the students(**stu_nbr**) who obtained the maximum mark in the unit





Subquery (NESTED)

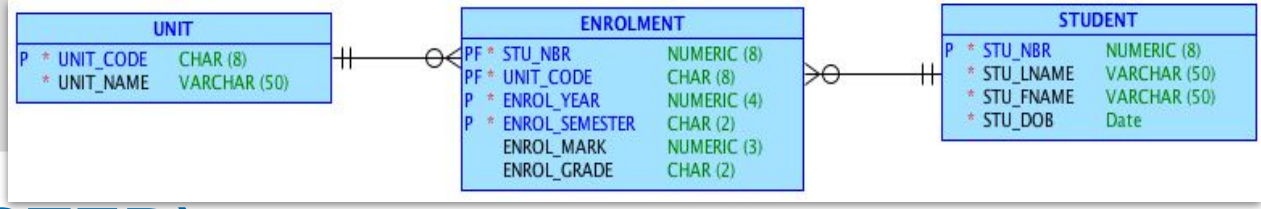
- For each unit, find the students who obtained the maximum mark in the unit

```
SELECT stu_nbr, unit_code, enrol_mark
FROM enrolment
WHERE (unit_code, enrol_mark)
IN (
```

UNIT_CODE	MAX(ENROL_MARK)
FIT1001	78
FIT1002	77
FIT1004	99

```
)
ORDER BY unit_code, stu_nbr;
```

- the subquery is independent of the outer query and is executed only once.

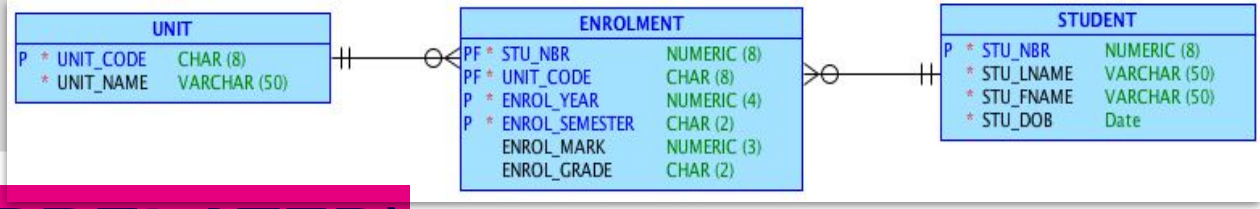


Subquery (NESTED)

- For each unit, find the students who obtained the maximum mark in the unit

```
SELECT stu_nbr, unit_code, enrol_mark
FROM enrolment
WHERE (unit_code, enrol_mark)
IN (SELECT unit_code, max(enrol_mark)
    FROM enrolment
    GROUP BY unit_code)
ORDER BY unit_code, stu_nbr;
```

- the subquery is independent of the outer query and is executed only once.

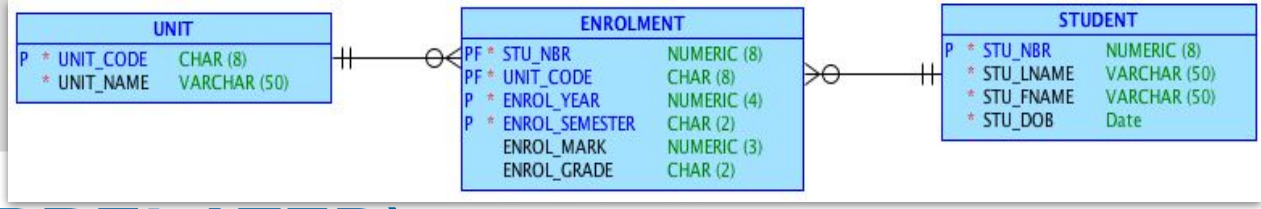


Subquery (CORRELATED)

- For each unit, find the students who obtained the maximum mark in the unit

```
SELECT stu_nbr, unit_code, enrol_mark
FROM enrolment e1
WHERE enrol_mark =
  ( the maximum mark corresponding
    to the above unit_code )
ORDER BY unit_code, stu_nbr;
```

- the subquery is related to the outer query and is considered to be evaluated once for each row of the outer query

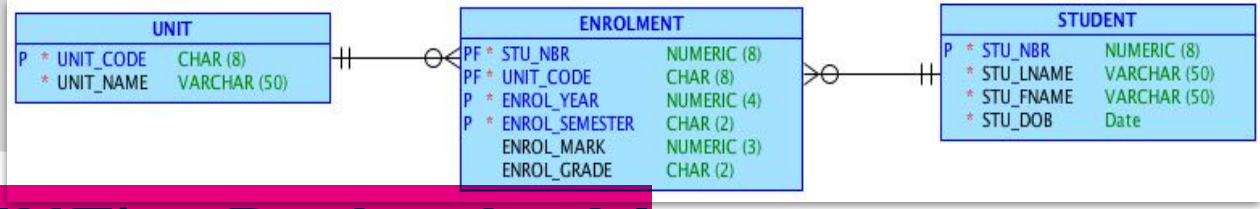


Subquery (CORRELATED)

- For each unit, find the students who obtained the maximum mark in the unit

```
SELECT stu_nbr, unit_code, enrol_mark
FROM enrolment e1
WHERE enrol_mark =
  ( SELECT max(enrol_mark)
    FROM enrolment e2
    WHERE e1.unit_code = e2.unit_code )
ORDER BY unit_code, stu_nbr;
```

- the subquery is related to the outer query and is considered to be evaluated once for each row of the outer query



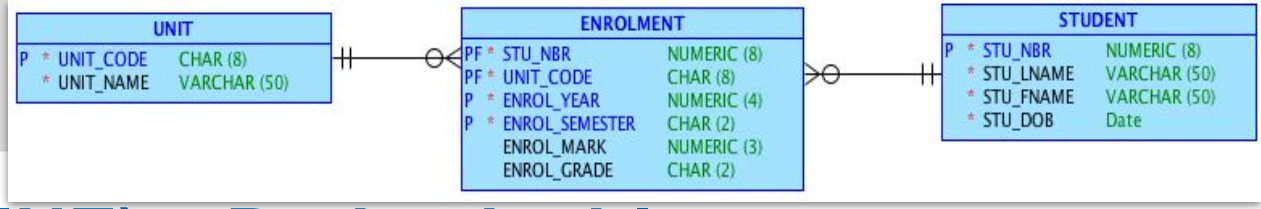
Subquery (INLINE) – Derived table

- For each unit, find the students who obtained the maximum mark in the unit

```
SELECT e.stu_nbr, e.unit_code, e.enrol_mark
FROM enrolment e
JOIN (
```

UNIT_CODE	MAX_MARK
FIT1001	78
FIT1002	77
FIT1004	99

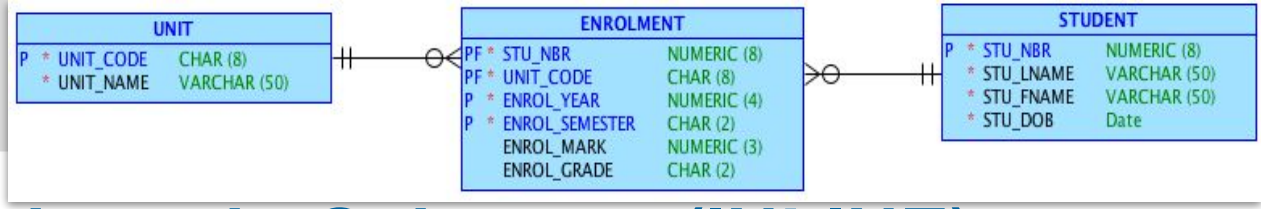
```
) max_table
ON e.unit_code = max_table.unit_code
AND e.enrol_mark = max_table.max_mark
ORDER BY unit_code, stud_nbr;
```



Subquery (INLINE) – Derived table

- For each unit, find the students who obtained the maximum mark in the unit

```
SELECT e.stu_nbr, e.unit_code, e.enrol_mark
FROM enrolment e
JOIN (SELECT unit_code, max(enrol_mark) as max_mark
      FROM enrolment
      GROUP BY unit_code) max_table
ON e.unit_code = max_table.unit_code
AND e.enrol_mark = max_table.max_mark
ORDER BY unit_code, stu_nbr;
```



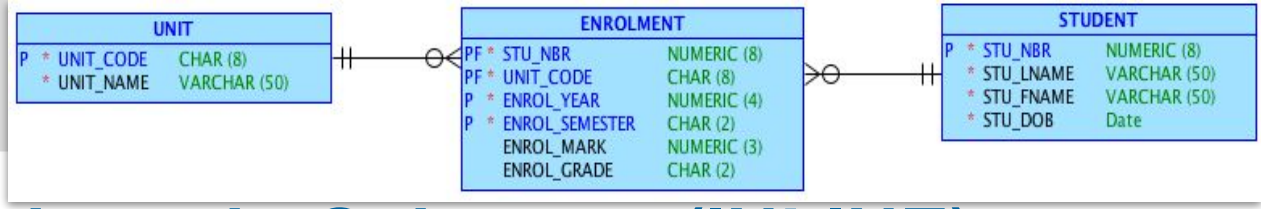
ADVANCED with math: Subquery (INLINE)

- For each grade, compute the percentage of the students who got that grade, out of ALL students (irrespective if they got a grade or not).

```
SELECT
    ???,
    ??? as grade_count,
    ??? as total_rows,
    ??? as percentage
FROM enrolment
WHERE enrol_grade is NOT NULL
GROUP BY enrol_grade
ORDER BY enrol_grade;
```

[Clayton Q&A] - fill in the blanks!

ENROL_GRADE	GRADE_COUNT	TOTAL_ROWS	PERCENTAGE
C	2	10	20
D	2	10	20
HD	1	10	10
N	2	10	20



ADVANCED with math: Subquery (INLINE)

- For each grade, compute the percentage of the students who got that grade, out of ALL students (irrespective if they got a grade or not).

```
SELECT
    enrol_grade,
    count(enrol_grade) as grade_count,
    (SELECT count(*) from enrolment) as total_rows,
    100*count(enrol_grade)/(SELECT count(*) FROM enrolment) as
percentage
FROM enrolment
WHERE enrol_grade is NOT NULL
GROUP BY enrol_grade
ORDER BY enrol_grade;
```

ENROL_GRADE	GRADE_COUNT	TOTAL_ROWS	PERCENTAGE
C	2	10	20
D	2	10	20
HD	1	10	10
N	2	10	20

VIEWs...
(Advanced lessons)

Img src: @samuelzeller at Unsplash / icons8

Views

- A virtual table derived from one or more base tables.
- Sometimes used as "Access Control" to the database. Syntax:

```
CREATE OR REPLACE VIEW [view_name] AS
SELECT … ;
-- example --
CREATE OR REPLACE VIEW max_view as
    SELECT unit_code, max(enrol_mark) as max_mark
    FROM enrolment
    GROUP BY unit_code;
```

- The view can be used as a ‘virtual table’ in SELECTs.

```
SELECT * from max_view
ORDER BY unit_code;
```

Views

- Oracle lets you amend values from some (simple) views…

```
CREATE OR REPLACE VIEW summary_grades as
SELECT unit_code, enrol_mark, enrol_grade
FROM enrolment
ORDER BY unit_code;


-- simple modification --
-- affects actual table --
UPDATE summary_grades
SET enrol_grade = 'ZZ'
WHERE enrol_mark > 77
```

```
1 SELECT * from enrolment
```

STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
11111111	FIT1001	2012	1	78	ZZ
11111111	FIT1002	2013	1	10	N
11111111	FIT1002	2014	1	65	C
11111111	FIT1004	2013	1		
11111112	FIT1001	2012	1	35	N
11111112	FIT1001	2013	1		
11111113	FIT1001	2012	2	65	C
11111113	FIT1004	2013	1		
11111114	FIT1004	2013	1	99	ZZ
11111114	FIT1002	2013	1	77	D

10 rows

- ...but of course some views (e.g. GROUP BY) can't be changed.

(Well, there is a rather complex workaround -- an advanced method using Oracle's "INSTEAD OF..." triggers. This complex trigger is strictly not examinable.)

```
1 -- modification fails --
2 UPDATE max_view
3 SET unit_code = 'AA'
4 WHERE max_mark > 88
```

```
ORA-01732: data manipulation operation not legal on this view
```

Views

- Oracle built in view: What objects do I own?

```
select * from user_objects;
```

OBJECT_NAME	SUBOBJECT_NAME	OBJECT_ID	DATA_OBJECT_ID	OBJECT_TYPE	CREATED	LAST_DDL_TIME	TIMESTAMP	STATUS	TEMPORARY
STUDENT		244610	244610	TABLE	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N
PK_STUDENT		244611	244611	INDEX	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N
UNIT		244612	244612	TABLE	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N
PK_UNIT		244613	244613	INDEX	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N
UQ_UNIT_NAME		244614	244614	INDEX	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N
ENROLMENT		244615	244615	TABLE	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N
PK_ENROLMENT		244616	244616	INDEX	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N
MAX_VIEW		244617		VIEW	06-MAY-19	06-MAY-19	2019-05-06:06:25:11	VALID	N

8 rows

Using Views - rewriting a subquery

- For each unit, find the students who obtained the maximum mark in the unit.
- Recall subquery

```
SELECT e.stu_nbr, e.unit_code, e.enrol_mark
FROM enrolment e
JOIN (SELECT unit_code, max(enrol_mark) as max_mark
      FROM enrolment
      GROUP BY unit_code) max_table
ON e.unit_code = max_table.unit_code
AND e.enrol_mark = max_table.max_mark
ORDER BY unit_code, stu_nbr;
```

Using Views - rewriting subquery

```
CREATE OR REPLACE VIEW max_view AS
 SELECT unit_code, max(enrol_mark) as max_mark
 FROM enrolment
 GROUP BY unit_code;
```

```
SELECT e.stu_nbr, e.unit_code, e.enrol_mark
FROM enrolment e JOIN max_view v
ON e.unit_code = v.unit_code
AND e.enrol_mark = v.max_mark
ORDER BY e.unit_code;
```

```
SELECT e.stu_nbr, e.unit_code, e.enrol_mark
FROM enrolment e
JOIN (SELECT unit_code, max(enrol_mark) as max_mark
      FROM enrolment
      GROUP BY unit_code) max_table
ON e.unit_code = max_table.unit_code
AND e.enrol_mark = max_table.max_mark
ORDER BY unit_code, stu_nbr;
```

JOINs

(Recall Relational Algebra?)

Img src: @samuelzeller at Unsplash / icons8

Self Join - Recall EMPLOYEE case study

- Show the number of the manager for each employee.
  How do we show the name?

```
SELECT
      empno,
      empname,
      empinit,
      mgrno
FROM
      emp.employee;
```

	EMPNO	EMPNAME	EMPINIT	MGRNO
1	7839	KING	CC	(null)
2	7566	JONES	JM	7839
3	7902	FORD	MG	7566
4	7369	SMITH	N	7902
5	7698	BLAKE	R	7839
6	7499	ALLEN	JAM	7698
7	7521	WARD	TF	7698
8	7654	MARTIN	P	7698
9	7782	CLARK	AB	7839
10	7788	SCOTT	SCJ	7566
11	7844	TURNER	JJ	7698
12	7876	ADAMS	AA	7788
13	7900	JONES	R	7698
14	7934	MILLER	TJA	7782

```
SELECT *
FROM emp.employee e1 JOIN emp.employee e2
ON e1.mgrno = e2.empno;
```

e1 e2

	EMPNO	EMPNAME	EMPINIT	MGRNO	EMPNO_1	EMPNAME_1	EMPINIT_1	MGRNO_1
1	7902	FORD	MG	7566	7566	JONES	JM	7839
2	7788	SCOTT	SCJ	7566	7566	JONES	JM	7839
3	7900	JONES	R	7698	7698	BLAKE	R	7839
4	7499	ALLEN	JAM	7698	7698	BLAKE	R	7839
5	7521	WARD	TF	7698	7698	BLAKE	R	7839
6	7654	MARTIN	P	7698	7698	BLAKE	R	7839
7	7844	TURNER	JJ	7698	7698	BLAKE	R	7839
8	7934	MILLER	TJA	7782	7782	CLARK	AB	7839
9	7876	ADAMS	AA	7788	7788	SCOTT	SCJ	7566
10	7782	CLARK	AB	7839	7839	KING	CC	(null)
11	7698	BLAKE	R	7839	7839	KING	CC	(null)
12	7566	JONES	JM	7839	7839	KING	CC	(null)
13	7369	SMITH	N	7902	7902	FORD	MG	7566

Joined rows
1,12
2,12
3,11
etc
etc

Note some columns have been hidden

Q: Why now only 13 rows?
Hint: who is the Big Boss?

```
SELECT e1.empno, e1.empname, e1.empinit, e1.mgrno,
    e2.empname AS MANAGER
FROM emp.employee e1 JOIN emp.employee e2
    ON e1.mgrno = e2.empno
ORDER BY e1.empname;
```

	EMPNO	EMPNAME	EMPINIT	MGRNO	MANAGER
1	7876	ADAMS	AA	7788	SCOTT
2	7499	ALLEN	JAM	7698	BLAKE
3	7698	BLAKE	R	7839	KING
4	7782	CLARK	AB	7839	KING
5	7902	FORD	MG	7566	JONES
6	7900	JONES	R	7698	BLAKE
7	7566	JONES	JM	7839	KING
8	7654	MARTIN	P	7698	BLAKE
9	7934	MILLER	TJA	7782	CLARK
10	7788	SCOTT	SCJ	7566	JONES
11	7369	SMITH	N	7902	FORD
12	7844	TURNER	JJ	7698	BLAKE
13	7521	WARD	TF	7698	BLAKE

Coffee break - see you in 10 minutes.

Img src: Jazmin Quaynor @jazminantoinette at Unsplash

NATURAL JOIN - Recall Rel. Alg. case study

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Natural Join gives no information for Chris (no enrolment, e.g. just enrolled) and the student with ID 4 (student without info, e.g. quit uni)

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90

```
select * from student s JOIN mark m on s.id = m.id;
```

FULL OUTER JOIN - Recall Rel. Alg. case study

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of both Chris and student with ID 4 i.e. we want NULLS both sides too.

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90
(null)	(null)	4	1004	100
3	Chris	(null)	(null)	(null)

```
select * from
student s FULL OUTER JOIN mark m
on s.id = m.id;
```

LEFT OUTER JOIN - Recall Rel. Alg. case study

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of only Chris - i.e. make sure all LEFT values present...

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90
3	Chris	(null)	(null)	(null)

```
select * from
student s LEFT OUTER JOIN mark m
on s.id = m.id;
```

RIGHT OUTER JOIN - Recall Rel. Alg. case study

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of the student with ID 4 - i.e. make sure all RIGHT values present...

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1045	90
1	Alice	1	1004	95
2	Bob	2	1045	55
(null)	(null)	4	1004	100

```
select * from
student s RIGHT OUTER JOIN mark m
on s.id = m.id;
```

Q1. What is the output from the following SQL?

ID	Name	ManagerID
1	Alice	2
2	Bob	3
3	Chris	

```
SELECT e1.name as emp_name,
e2.name as manager_name
FROM employee e1 RIGHT OUTER JOIN employee e2
ON e1.managerid = e2.id;
```

(A)

EMP_NAME	MANAGER_NAME
Alice	Bob
Bob	Chris
(null)	Alice

(B)

EMP_NAME	MANAGER_NAME
Alice	Bob
Bob	Chris
Chris	(null)

Employee

ID	Name	Salary
1	Alice	100,000
2	Bob	150,000
3	Chris	200,000

Project

Project	Cost	EmpID
Alpha	4000	1
Beta	3000	2
Gamma	5000	2

Q2. Which of the following shows, for *each* employee, the total amount of projects they are assigned to? (E.g., Alice is assigned to Alpha with total cost 4000, Bob is assigned to Beta and Gamma with total cost 8000, Chris has $0)

A.
```
SELECT e.name, sum(cost) as total
FROM employee e LEFT OUTER JOIN project p
ON e.id = p.empid GROUP BY e.name;
```

B.
```
SELECT e.name, sum(cost) as total
FROM employee e RIGHT OUTER JOIN project p
ON e.id = p.empid GROUP BY e.name;
```

C.
```
SELECT e.name, count(sum(cost)) as total
FROM employee e LEFT OUTER JOIN project p
ON e.id = p.empid GROUP BY e.name;
```

D. None of the answers given!

Set Operators

(Recall Relational Algebra?)

Img src: @samuelzeller at Unsplash / icons8

Q3. Two or more queries that are connected using a set operator have to be union compatible. When would two relations be union compatible? It is when the two relations have:

A. the same degree and similar domain for the attributes
B. the same degree and attributes' name
C. the same degree and cardinality.
D. the same cardinality.

Relational Set Operators

- Using the set operators you can combine two or more sets to create new sets (relations)
- Union All: All rows selected by either query, including all duplicates
- Union: All rows selected by either query, removing duplicates (e.g. DISTINCT on Union All)
- Intersect: All distinct rows selected by both queries
- Minus: All distinct rows selected by the first query but not by the second

- All set operators have equal precedence. **If a SQL statement contains multiple set operators, Oracle evaluates them from the left to right if no parentheses explicitly specify another order.**
- The two sets must be UNION COMPATIBLE (i.e., same number of attributes and similar data types)

Basic Examples

- The next few slides show a very simple example with the following data.
  - NB: this is not normalised nor efficient and used purely for illustrative purposes!

ID	NAME	ARTIST	YEAR	GENRE
1	Thrift Shop	Macklemore & Ryan Lewis	2012	Hip hop
2	Imagine	John Lennon	1971	Pop
3	Numb	Linkin Park	2003	Rock
4	Lose Yourself	Eminem	2002	Rap
5	Sandstorm	Darude	1999	Trance
6	Help	Beatles	1965	Pop
7	Teardrops on My Guitar	Taylor Swift	2006	Country
8	Shake It Off	Taylor Swift	2014	Pop
9	Youve Got a Friend	James Taylor	1971	Folk

- There are complex examples shown afterwards, with relation to the complex UNIVERSITY ENROLMENT database.
  - **Advanced: left as homework.**
  - Remember to add the prefix to the tables as per Tute 8!

Basic Examples: UNION ALL

- All rows selected by either query, including all duplicates

```
-- query A in black ink, query B in blue ink --
SELECT * FROM song WHERE year > 2012
UNION ALL
SELECT * FROM song WHERE artist LIKE '%Taylor%';
```

ID	NAME	ARTIST	YEAR	GENRE
8	Shake It Off	Taylor Swift	2014	Pop
7	Teardrops on My Guitar	Taylor Swift	2006	Country
8	Shake It Off	Taylor Swift	2014	Pop
9	Youve Got a Friend	James Taylor	1971	Folk

Note: ID 8 (Shake It Off by Taytay) appears twice, as it matches both query A (released in 2014) and query B (artist name has 'Taylor').

Basic Examples: UNION

- All rows selected by either query, removing duplicates

```
-- query A in black ink, query B in blue ink --
SELECT * FROM song WHERE year > 2012
UNION
SELECT * FROM song WHERE artist LIKE '%Taylor%';
```

ID	NAME	ARTIST	YEAR	GENRE
7	Teardrops on My Guitar	Taylor Swift	2006	Country
8	Shake It Off	Taylor Swift	2014	Pop
9	Youve Got a Friend	James Taylor	1971	Folk

Note 1: ID 8 shown only once, cf. prev. slide.

Note 2: DISTINCT on Union All -- similar result

```
SELECT DISTINCT * FROM
(
SELECT * FROM song WHERE year > 2010
UNION ALL
SELECT * FROM song WHERE artist LIKE '%Taylor%'
);
```

Basic Examples: INTERSECT

- All distinct rows selected by **both** queries

```
-- query A in black ink, query B in blue ink --
SELECT * FROM song WHERE year > 2010
INTERSECT
SELECT * FROM song WHERE artist LIKE '%Taylor%';
```

ID	NAME	ARTIST	YEAR	GENRE
8	Shake It Off	Taylor Swift	2014	Pop

Note: see how the intersection works if the queries were done separately

ID	NAME	ARTIST	YEAR	GENRE
1	Thrift Shop	Macklemore & Ryan Lewis	2012	Hip hop
8	Shake It Off	Taylor Swift	2014	Pop

ID	NAME	ARTIST	YEAR	GENRE
7	Teardrops on My Guitar	Taylor Swift	2006	Country
8	Shake It Off	Taylor Swift	2014	Pop
9	Youve Got a Friend	James Taylor	1971	Folk

Basic Examples: MINUS

- All distinct rows selected by the first query but not by the second

```
-- query A in black ink, query B in blue ink --
SELECT * FROM song WHERE year > 2010
MINUS
SELECT * FROM song WHERE artist LIKE '%Taylor%';
```

ID	NAME	ARTIST	YEAR	GENRE
1	Thrift Shop	Macklemore & Ryan Lewis	2012	Hip hop

Note: see how the minus works if the queries were done separately

ID	NAME	ARTIST	YEAR	GENRE
1	Thrift Shop	Macklemore & Ryan Lewis	2012	Hip hop
8	Shake It Off	Taylor Swift	2014	Pop

ID	NAME	ARTIST	YEAR	GENRE
7	Teardrops on My Guitar	Taylor Swift	2006	Country
8	Shake It Off	Taylor Swift	2014	Pop
9	Youve Got a Friend	James Taylor	1971	Folk

Advanced example: UNION
(on ENROLMENT case study, Tute 8)

- Create a list of units with its average mark. Give the label “Below distinction” to all units with the average less than 70 and “Distinction and Above” for those units with average greater or equal to 70.

UNITCODE	AVERAGE	AVERAGE_STATUS
FIT5131	77.5	Distinction and Above
FIT5136	75.5	Distinction and Above
FIT5132	74	Distinction and Above
FIT1004	71.7	Distinction and Above
FIT1040	70	Distinction and Above
FIT2077	64.5	Below Distinction

1. Select units with average marks less than 70 and set status
2. Select units with average marks greater or equal to 70 and set status
3. Take a union of 1 and 2

Advanced example: UNION (on ENROLMENT case study, Tute 8)

```
SELECT unitcode, AVG(mark) AS Average, 'Below Distinction' AS Average_Status
FROM
  enrolment
GROUP BY
  unitcode
HAVING
  AVG(mark) < 70
UNION
SELECT unitcode, AVG(mark) AS Average, 'Distinction and Above' AS
Average_Status
FROM
  enrolment
GROUP BY
  unitcode
HAVING
  AVG(mark) >= 70
ORDER BY
  Average DESC ;
```

Advanced example: INTERSECTION (on ENROLMENT case study, Tute 8)

```
SELECT studid, studfname, studlname
FROM
  student
WHERE
  studlname IN
  ( SELECT DISTINCT studlname
    FROM
      student
    INTERSECT
    SELECT DISTINCT stafflname
    FROM
      staff)
ORDER BY studid;
```

Find students who have the same surname as a staff member's surname.

1. (Find the common surnames in staff and student table.)
2. Find students with the surname present in step 1

Advanced example: MINUS (on ENROLMENT case study, Tute 8)

- List the name of staff who are not a chief examiner in an offering.

```
select staffid, stafflname, stafffname
from staff
where staffid IN
    (select staffid from staff
    minus
    select chiefexam from offering);
```

STAFFID
1
2
3
4
5
6
7

CHIEFEXAM
1
1
1
1
3
4
5
5
7
7
7
7
7
7

STAFFID	STAFFLNAME	STAFFFNAME
2	Burbage	Charity
6	Umbridge	Dolores

Oracle Functions

(refer the PDF reference on Moodle!)

Img src: @samuelzeller at Unsplash / icons8

Function Type	Applicable to	Example
Arithmetic	Numerical data	SELECT ucode, round(avg(mark)) FROM enrolment GROUP BY ucode;
Text	Alpha numeric data	SELECT studsurname FROM enrolment WHERE upper(studsurname) LIKE 'B%';
Date	Date/Time-related data	
General	Any data type	NVL function
Conversion	Data Type conversion	SELECT to_char(empmsal,'$0999.99') FROM employee;
Group	Sets of Values	avg(), count(), etc

See document on Moodle

Q4. Given the following Oracle syntax for the ROUND function:

`ROUND(n [,integer])` where n is a number and integer determines the decimal point;

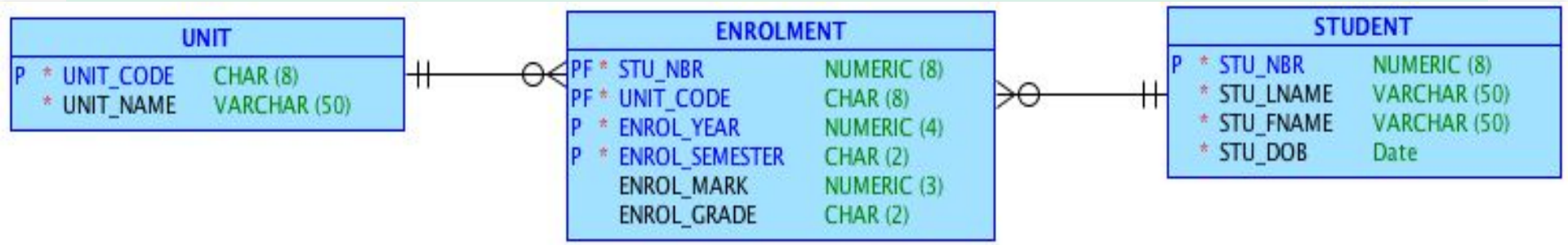
what would be the right SELECT clause for rounding the average mark of all marks in the enrolment (not including the NULL values) to the nearest 2 decimal points?

A. SELECT avg(round(mark,2))
B. SELECT round(avg(mark,2))
C. SELECT round(avg(mark),2)
D. SELECT avg(mark(round(2)))

Last-minute revision?

Print student last names, enrolment years, and the count (total) grades recorded for each.

```
------
Bloggs 2019    4
Bloggs 2018    7
Huynh  2018    12
Huynh  2017    7
Taylor 2015    4
-------
e.g.
Fred Bloggs:  N N P N (2019) + D D D D D D D (2018)
Joe Huynh:    C C D P P P C C (2018) + P P P P (2017)
Elvis Huynh:  HD HD HD HD (2018) + HD HD HD (2017)
Bravo Taylor: HD HD HD HD (2015)
```



Last-minute revision?

Print last names & first names, total (count) of grades received, and corresponding unit names; such that I only get students who are (or have been) attempting a unit 2 or more times. Assume we ignore units still pending a grade (null)

```
Bloggs Jim      3  Databases
Huynh  Brendon  5  Database Advanced
Huynh  Brendon  3  Emojis for Computer Science
Huynh  Brendon  2  Database Basics
Taylor Marc     2  Data Science
Taylor Fred     2  Economics
Taylor Arif     9  Quantum Politics
```