



MONASH University

Information Technology

FIT3176 Advanced Database Design

Week 3 – SQL Revision

Dr. Minh Le
Minh.Le@monash.edu

algorithm distributed systems **database**
systems **computation** knowledge ma
design e-business **model** data mining int
distributed systems **database** software
computation knowledge management an

Q1. Given the following code, how many SQL **STATEMENT(s)** is/are present?

```
SELECT empno, empname  
FROM employee  
WHERE salary > 5000;
```

- A. 1
- B. 2
- C. 3
- D. 4

mars.mu
Feed Code: **F1PEMT**

Q2. Given the following code, how many SQL **CLAUSE(S)** is/are present?

```
SELECT empno, empname  
FROM employee  
WHERE salary > 5000;
```

- A. 1
- B. 2
- C. 3
- D. 4

mars.mu
Feed Code: **F1PEMT**

Q3. Given the following code, what is the **name of the table** from which the set of rows will be retrieved?

```
SELECT empno, empname  
FROM employee  
WHERE salary > 5000;
```

- A. salary
- B. empno
- C. employee
- D. empname.

mars.mu
Feed Code: **F1PENT**

Q4. What component of a database is 'empno' in the code below?

```
SELECT empno, empname  
FROM employee  
WHERE salary > 5000;
```

- A. column
- B. row
- C. table
- D. constraint.

mars.mu
Feed Code: **F1PEMT**

Q5. Given the following code, what is the search condition or PREDICATE for this SQL statement?

```
SELECT empno, empname  
FROM employee  
WHERE salary > 5000;
```

- A. salary > 5000
- B. empno, empname
- C. employee
- D. There is no search condition.

mars.mu
Feed Code: **F1PENT**

SQL Predicates or Search Conditions

- A row will be returned if the search conditions are evaluated to be TRUE.
- Comparison
 - Compare the value of one expression to the value of another expression.
 - Operators:
 - =, < >, <, >, !=, <=, >=
 - 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

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 unitcode LIKE 'FIT100_'

SQL Predicates or Search Conditions

- NULL
 - To test whether a column has a NULL (unknown) value.
 - Example: WHERE grade IS NULL.
 - Beware: WHERE grade = NULL does not cause an error
- 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.

R#	SNO	R#	SEMESTER	R#	EYEAR	R#	UCODE	R#	MARK	R#	GRADE
1	1111		1	2012	FIT1001				78		D
2	1111		1	2013	FIT1002				(null)		(null)
3	1111		1	2013	FIT1004				(null)		(null)
4	1112		1	2012	FIT1001				35		N
5	1112		1	2013	FIT1001				(null)		(null)
6	1113		2	2012	FIT1001				65		C
7	1113		1	2013	FIT1004				(null)		(null)
8	1114		1	2013	FIT1004				(null)		(null)

Q6. Consider the predicate “mark > 50”, what row(s) will be evaluated to TRUE?

- A. 1, 4 and 6
- B. All rows
- C. 1 and 6
- D. 2, 3, 5, 7 and 8.

mars.mu
Feed Code: **F1PENT**

R#	SNO	R#	SEMESTER	R#	EYEAR	R#	UCODE	R#	MARK	R#	GRADE
1	1111		1	2012	FIT1001				78		D
2	1111		1	2013	FIT1002				(null)		(null)
3	1111		1	2013	FIT1004				(null)		(null)
4	1112		1	2012	FIT1001				35		N
5	1112		1	2013	FIT1001				(null)		(null)
6	1113		2	2012	FIT1001				65		C
7	1113		1	2013	FIT1004				(null)		(null)
8	1114		1	2013	FIT1004				(null)		(null)

Q7. What is the correct SQL predicate to retrieve a list of students who have passed (mark is greater than 50) together with students who have not been awarded any mark?

- A. mark > 50 AND mark IS NULL
- B. mark > 50 OR mark IS NULL
- C. mark > 50 AND mark IS NOT NULL
- D. mark > 50 OR mark IS NOT NULL
- E. None of the above

mars.mu
Feed Code: **F1PEMT**

R#	SNO	R#	SEMESTER	R#	EYEAR	R#	UCODE	R#	MARK	R#	GRADE
1	1111		1	2012	FIT1001		78	D			
2	1111		1	2013	FIT1002		(null)	(null)			
3	1111		1	2013	FIT1004		(null)	(null)			
4	1112		1	2012	FIT1001		35	N			
5	1112		1	2013	FIT1001		(null)	(null)			
6	1113		2	2012	FIT1001		65	C			
7	1113		1	2013	FIT1004		(null)	(null)			
8	1114		1	2013	FIT1004		(null)	(null)			

Q8. Which of the following SQL statements will retrieve ONLY the student number and grade for all students who passed (mark > 50) FIT1001 in the year 2012?

- | | | |
|------|----------------------|----------------------|
| A. A | a. SELECT * | b. SELECT sno, grade |
| B. B | FROM enrolment; | FROM enrolment; |
| C. C | c. SELECT sno, grade | d. SELECT sno, grade |
| D. D | FROM enrolment | FROM enrolment |
| | WHERE eyear=2012 | WHERE eyear=2012 |
| | AND | AND |
| | ucode='FIT1001' ; | ucode='FIT1001' |
| | | AND |
| | | mark > 50; |
- mars.mu
Feed Code: **F1PEMT**

	SNO	SEMESTER	EYEAR	UCODE	MARK	GRADE
1	1111	1	2012	FIT1001	78	D
2	1111	1	2013	FIT1002	(null)	(null)
3	1111	1	2013	FIT1004	(null)	(null)
4	1112	1	2012	FIT1001	35	N
5	1112	1	2013	FIT1001	(null)	(null)
6	1113	2	2012	FIT1001	65	C
7	1113	1	2013	FIT1004	(null)	(null)
8	1114	1	2013	FIT1004	(null)	(null)

Q9. What row number/s will be retrieved by the following SQL statement?

```
SELECT sno, mark/10
FROM enrolment
WHERE mark/10 > 5;
```

- A. All rows.
- B. 1, 4 and 6.
- C. 2, 3, 5, 7 and 8
- D. 1, 6

mars.mu
Feed Code: **F1PEMT**

Renaming Column

- Use the word "AS"
 - Especially useful for computed columns
 - If new name has a space must be included in *double* quotes
 - "New Mark"
 - If no spaces then quotes not needed
 - NewMark
- Example

```
SELECT sno, mark/10 as NewMark  
FROM enrolment;
```

Sorting Query Result

- "ORDER BY" clause.
 - ORDER BY **required** if more than one row may be returned
- Order can be ASCending or DESCending.
 - Default is ASCending.
- Sorting can be done for multiple columns.
 - order of the sorting is specified for each column.
- Example:

```
SELECT sno, mark  
FROM enrolment  
ORDER BY mark DESC
```

	R1	SNO	R2	MARK
1		1111		{null}
2		1111		{null}
3		1114		{null}
4		1112		{null}
5		1113		{null}
6		1111		78
7		1113		65
8		1112		35

	<small>R2</small> SNO	<small>R2</small> SEMESTER	<small>R2</small> EYEAR	<small>R2</small> UCODE	<small>R2</small> MARK	<small>R2</small> GRADE
1	1111	1	2012	FIT1001	78	D
2	1111	1	2013	FIT1002	(null)	(null)
3	1111	1	2013	FIT1004	(null)	(null)
4	1112	1	2012	FIT1001	35	N
5	1112	1	2013	FIT1001	(null)	(null)
6	1113	2	2012	FIT1001	65	C
7	1113	1	2013	FIT1004	(null)	(null)
8	1114	1	2013	FIT1004	(null)	(null)

Q10. What will be the output of the following SQL statement?

**SELECT sno, mark
FROM enrolment
WHERE mark IS NOT NULL
ORDER BY mark, sno DESC;**

- A. A
- B. B
- C. C

a.

1111	78
1113	65
1112	35

b.

1112	35
1113	65
1111	78

c.

1111	78
1112	35
1113	65

mars.mu

Feed Code: **F1PEMT**

Worksheet

Query Builder

1

2

3

4

SELECT

stu_nbr,

enrol_mark

FROM

enrolment

WHERE

enrol_mark

IS NOT NULL

ORDER BY

enrol_mark,

stu_nbr





desc;

ASC

▶

Query Result

×



SQL

All Rows Fetched: 6 in 0.003 seconds

	STU_NBR	ENROL_MARK
1	11111111	35
2	11111111	42
3	11111112	61
4	11111111	61
5	11111111	76
6	11111112	83

DESC

Worksheet

Query Builder

1

2

3

4

SELECT

stu_nbr, enrol_mark

FROM

enrolment

WHERE

enrol_mark IS NOT NULL





ORDER BY

enrol_mark desc, stu_nbr desc;

DESC

▶

Query Result x



SQL | All Rows Fetched: 6 in 0.002 seconds

	STU_NBR	ENROL_MARK
1	11111112	83
2	11111111	76
3	11111112	61
4	11111111	61
5	11111111	42
6	11111111	35

DESC

Removing Duplicate Rows in the Query Result

- Use "DISTINCT" as part of SELECT clause.
 - Be CAREFUL and clearly understand why you are using DISTINCT

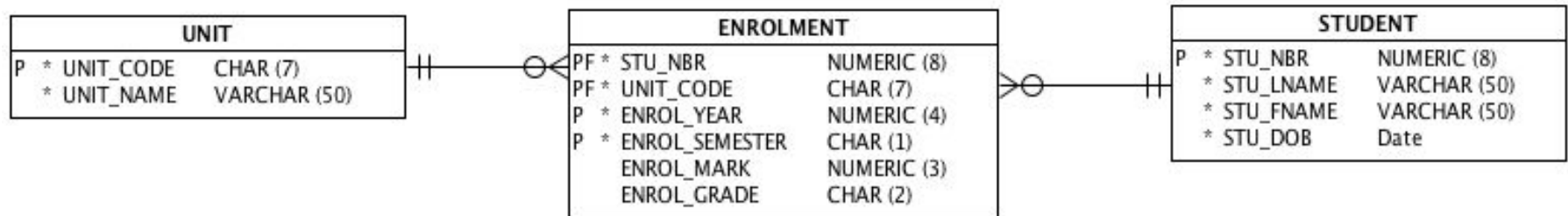
	SNO	SEMESTER	EYEAR	UCODE	MARK	GRADE
1	1111	1	2012	FIT1001	78	D
2	1111	1	2013	FIT1002	(null)	(null)
3	1111	1	2013	FIT1004	(null)	(null)
4	1112	1	2012	FIT1001	35	N
5	1112	1	2013	FIT1001	(null)	(null)
6	1113	2	2012	FIT1001	65	C
7	1113	1	2013	FIT1004	(null)	(null)
8	1114	1	2013	FIT1004	(null)	(null)

```
SELECT DISTINCT sno
FROM enrolment
WHERE mark IS NULL;
```

	SNO
1	1111
2	1114
3	1112
4	1113

JOIN-ing Multiple Tables - USING

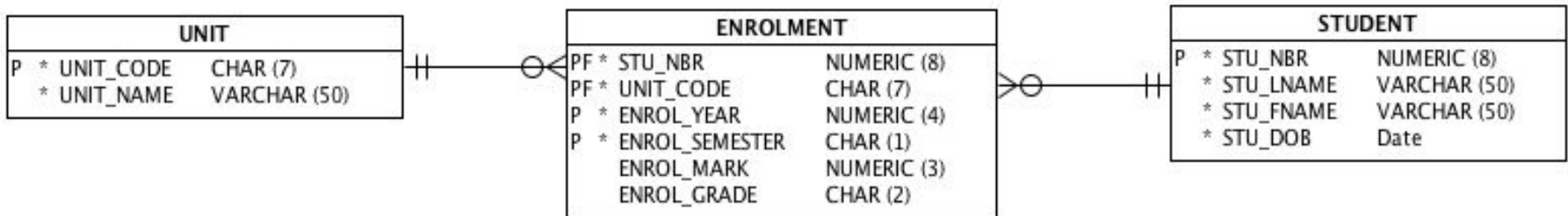
- Pair the PK and FK in the WHERE clause as a natural join condition.



```
SELECT stu_nbr, stu_fname, stu_lname, unit_name
FROM unit JOIN enrolment
      USING (unit_code)
      JOIN student
      USING (stu_nbr)
ORDER BY stu_nbr;
```

JOIN-ing Multiple Tables – ON predicate

- Pair the PK and FK in the WHERE clause as a natural join condition.



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

Aggregate Functions

- COUNT, MAX, MIN, SUM, AVG
- Example:

```
SELECT max(mark)
FROM enrolment;
```

```
SELECT min(mark)
FROM enrolment;
```

```
SELECT avg(mark)
FROM enrolment;
```

```
SELECT count(sno)
FROM enrolment
WHERE mark > 50;
```

	SNO	UCODE	EYEAR	SEMESTER	MARK	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)

Q11. Considering that "*" is a wildcard in SQL SELECT statement, what will be displayed the following SQL statement?

**SELECT count(*), count(mark)
FROM enrolment;**

- A. 8, 8
- B. 8, 3
- C. 3, 3
- D. 3, 8

mars.mu
Feed Code: **F1PEMT**

	SNO	UCODE	EYEAR	SEMESTER	MARK	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)

Q12. Considering that "*" is a wildcard in SQL SELECT statement, what will be displayed for the following SQL statement?

**SELECT count(*), count(sno), count(distinct sno)
FROM enrolment;**

- A. 8, 8, 4
- B. 8, 8, 8
- C. 8, 4, 8
- D. 8, 4, 4

mars.mu
Feed Code: **F1PEMT**

	SNO	UCODE	EYEAR	SEMESTER	MARK	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)

Q13. What will be displayed as the average mark based on the following SQL? Note: $(78+35+65)/3=59.333$, $(78+35+65)/8=22.25$

**SELECT avg(mark)
FROM enrolment;**

- A. 22.25
- B. 59.33
- C. Null
- D. None of the above.

mars.mu
Feed Code: **F1PEMT**

	SNO	UCODE	EYEAR	SEMESTER	MARK	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)

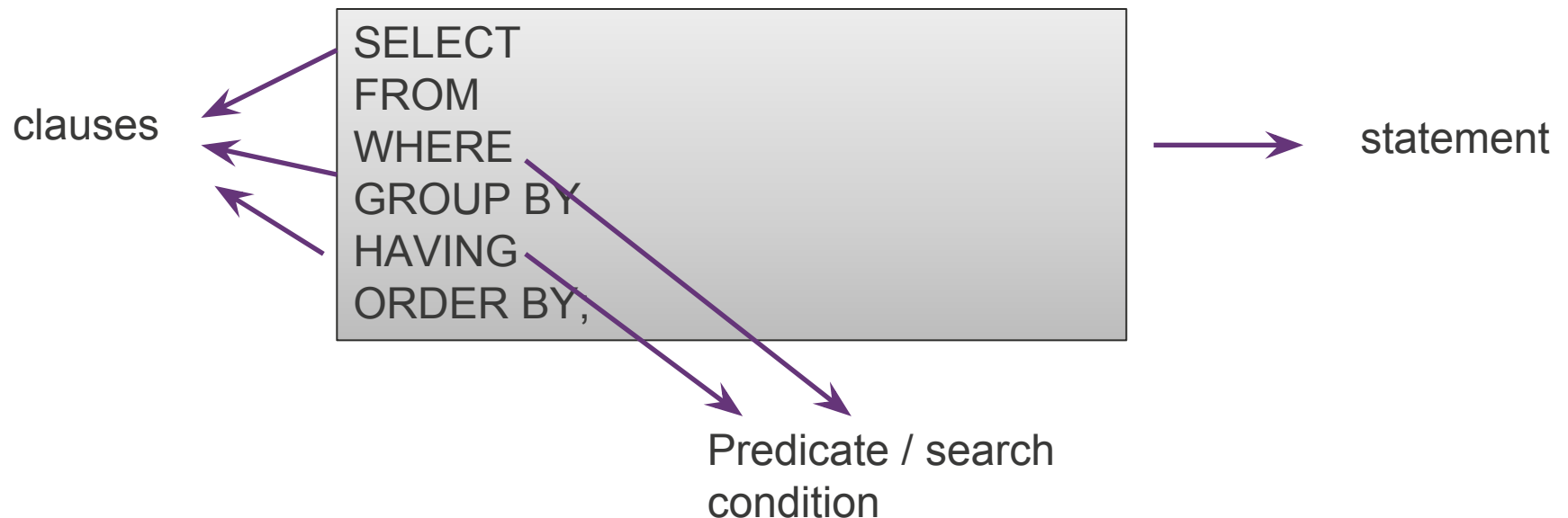
Q14. What will be displayed as the average mark based on the following SQL? Note: $(78+35+65)/3=59.333$, $(78+35+65)/8=22.25$

**SELECT sum(mark) / count(*)
FROM enrolment;**

- A. 22.25
- B. 59.33
- C. Null
- D. None of the above.

mars.mu
Feed Code: **F1PEMT**

Anatomy of an SQL Statement



GROUP BY

- If a GROUP BY clause is used with aggregate function, the DBMS will apply the aggregate function to the different groups defined in the clause rather than all rows.

```
SELECT avg(mark)  
FROM enrolment;
```

```
SELECT ucode, avg(mark)  
FROM enrolment  
GROUP BY ucode  
ORDER BY ucode;
```

	SNO	UCODE	EYEAR	SEMESTER	MARK	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)

Q15. What will be displayed as the average mark for FIT1001, FIT1002 and FIT1004 based on the following SQL? Note: $(78+35+65)/3=59.333$, $(78+35+65)/4=44.5$

**SELECT ucode, avg(mark)
FROM enrolment
GROUP BY ucode
ORDER BY ucode;**

- A. 59.333, null, null
- B. 59.333, 0, 0
- C. 44.5, null, null
- D. 44.5, 0, 0

mars.mu
Feed Code: **F1PEMT**

HAVING clause

- Is used to place a condition or conditions on the groups defined by the GROUP BY clause.

```
SELECT ucode, count(*)  
FROM enrolment  
GROUP BY ucode  
HAVING count(*) > 2  
ORDER BY ucode;
```

HAVING and WHERE clauses

```
SELECT ucode, count(*)  
FROM enrolment  
WHERE mark IS NULL  
GROUP BY ucode  
HAVING count(*) > 1  
ORDER BY ucode;
```

- The HAVING clause is applied to the groups defined by the GROUP BY clause.
- The WHERE clause is applied to ALL rows in the table.
- On the above example, the logic of the process will be:
 - All rows where mark is NULL are retrieved (due to the WHERE clause).
 - The retrieved rows then are grouped into different ucode values.
 - The total rows in each group (ucode) is computed.
 - If the total is greater than 1 the ucode and the total is displayed (due to the HAVING clause).

Subqueries

- Query within a query
 - Inner query is evaluated first

"Find all students whose mark is higher than the average mark of all enrolled students"

```
SELECT *  
FROM enrolment  
WHERE mark > (SELECT avg(mark)  
               FROM enrolment );
```

Types of Subqueries

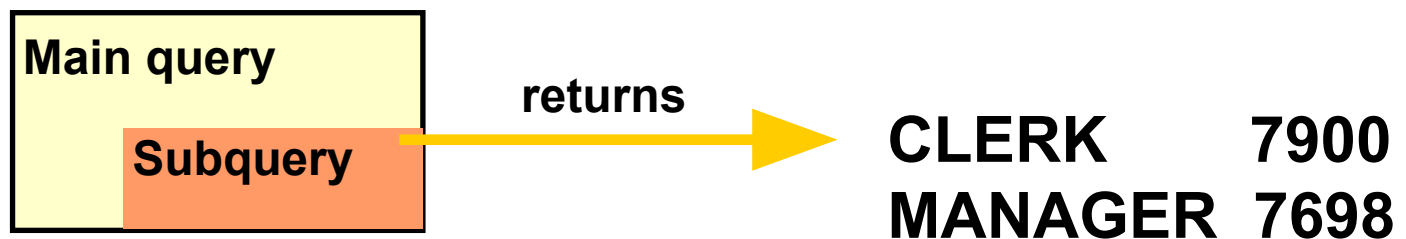
Single-row subquery (a single value – one row, may have multiple



Multiple-row subquery (a list of values – many rows, one column)



Multiple-column subquery (a virtual table – many rows, many columns)




```
SELECT *  
FROM enrolment  
WHERE mark > (SELECT avg(mark) FROM enrolment );
```

Q16. What will be returned by the inner query?

- A. A value (a single column, single row).
- B. A list of values.
- C. A table (multiple columns, multiple rows).
- D. None of the above.

mars.mu
Feed Code: **F1PENT**

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 upper('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 –"Useful Oracle Functions"

Q17. Given the following oracle syntax for round function:

ROUND(n [,integer]) where n is a number and integer determines the number of digits to the right of 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 2 digits after the decimal point?

_____ FROM enrolment;

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

mars.mu
Feed Code: **F1PEMT**

Date data type

- Dates are stored differently from the SQL standard
 - 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;`
» 01-Mar-2017
 - `select to_char(sysdate,'dd-Mon-yyyy hh:mi:ss PM') from dual;`
» 01-Mar-2017 10:27:10 AM
 - 10G introduced `TIMESTAMP` data type – finer granularity on secs
 - `select cast(sysdate as TIMESTAMP) from dual;`
– 01-MAR-17 10.27.10.000000000 AM

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

```
select studid,  
       studfname || ' ' || studlname as StudentName,  
       to_char(studdob, 'dd-Mon-yyyy') as StudentDOB  
from uni.student  
where studdob > to_date('01-Apr-1991', 'dd-Mon-yyyy')  
order by studdob;
```

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

```
> select sysdate from dual
SYSDATE
-----
19/MAR/15

> select to_char(sysdate,'dd-Mon-yyyy hh:mi AM') from dual
TO_CHAR(SYSDATE,'DD-MON-YYYYHH:MIAM')
-----
19-Mar-2015 08:28 PM

> select systimestamp from dual
SYSTIMESTAMP
-----
19/MAR/15 08:28:46.768000000 PM +11:00

> select user from dual
USER
-----
LSMI1
```

■ Subtraction

- Return days as a floating point, i.e will have fraction.

- Addition

39

STUDID	STUDENTNAME	STUDENTDOB
11111123	Tay Lee	01-Mar-1988
11111126	John Tse	03-Dec-1988
11111117	Jake Ryan	01-Jan-1990
11111127	Jake Brown	01-Jan-1990

Q18. The following SQL statement returns rows which include those show above. What part of the statement causes this error?

```
select studid,
       studfname || ' ' || studlname as StudentName,
       to_char(studdob,'dd-Mon-yyyy') as StudentDOB
from uni.student
where to_char(studdob,'dd-Mon-yyyy') > '01-Apr-1991'
order by studdob;
```

- A. The comparison operator should be "<".
- B. The to_char function causes the comparison in the condition to be character comparison rather than date.
- C. The date '01-Apr-1991' is not in a correct date format.
- D. More than one.

mars.mu
Feed Code: **F1PEMT**

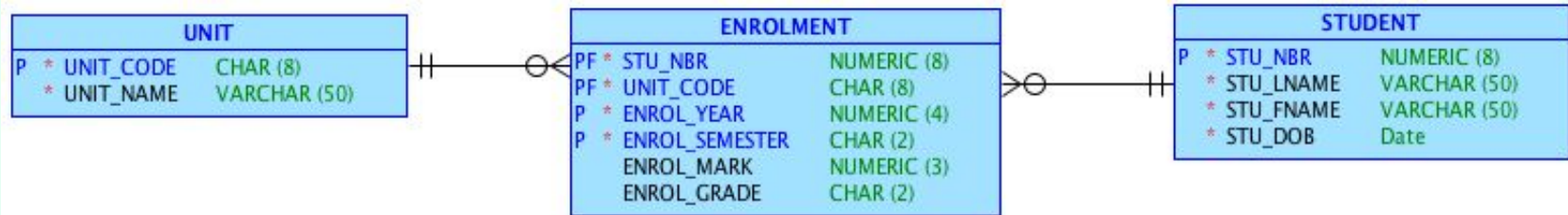
INSERT

- Adding data to a table in database.
- SYNTAX:

```
INSERT INTO table [(column [, column...])]  
VALUES (value [, value...]);
```

```
INSERT INTO unit VALUES ('FIT2077','Advanced Data  
Management');
```

```
INSERT INTO student VALUES (12345678,'Smith','Lindsay',  
to_date('01-JAN-1995','dd-MON-yyyy');
```



Q19. Assume the tables have been created and there is no data currently in the tables. In what order should we populate the table?

- A. UNIT- > STUDENT -> ENROLMENT
- B. ENROLMENT -> UNIT -> STUDENT
- C. STUDENT -> UNIT -> ENROLMENT
- D. More than one. (a,c)

mars.mu
Feed Code: **F1PEMT**

Using a SEQUENCE

- Oracle supports auto-increment of a numeric PRIMARY KEY
 - SEQUENCE
- Steps to use:
 - Create sequence (basic syntax) – for Data Modeller see Topic 01 lab

```
CREATE SEQUENCE stu_nbr_seq  
INCREMENT BY 1;
```

- Access the sequence using two built-in variables (pseudocolumns):
 - NEXTVAL and CURRVAL
 - INSERT INTO student
(stu_nbr_seq.nextval,'Bond','James','01-Jan-1994');
 - INSERT INTO enrolment
(stu_nbr_seq.currval,'FIT2077',.....);

```
-- Add two students
INSERT INTO student (stu_nbr_seq.nextval,'Bond','James','01-Jan-1994');
INSERT INTO student (stu_nbr_seq.nextval,'Lee','Bruce','01-Feb-1994');
-- Add the enrolments
INSERT INTO enrolment (sno_seq.currval,'FIT2077',2015,'S1',null,null);
INSERT INTO enrolment (sno_seq.currval, 'FIT2078',2014,'S1',null,null);
INSERT INTO enrolment (sno_seq.currval, 'FIT2077',2014,'S1',null,null);
COMMIT;
```

Q20. Two new students and their enrolment details need to be added, James Bond enrolls in FIT2077 and FIT2078, Bruce Lee only enrolls in FIT2077. What is/are the problem(s) with the use of SEQUENCE in the above SQL script?

- A. Duplication of PRIMARY KEY in the ENROLMENT table.
- B. Bruce Lee will be enrolled in FIT2078.
- C. There will be NO enrolment record for James Bond.
- D. More than one of the above

mars.mu
Feed Code: **F1PEMT**

UPDATE

- Change the values of existing data.
- For example, at the end of semester, change the mark and grade from null, null to the actual mark and grade.

```
UPDATE table  
SET column = (subquery) [, column = value, ...]  
[WHERE condition];
```

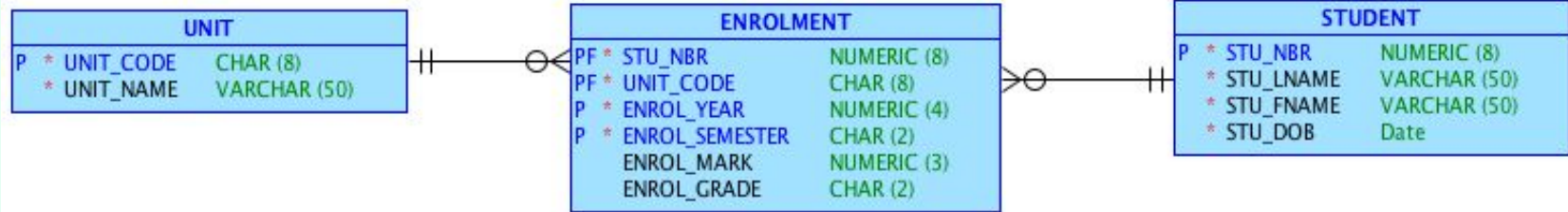
```
UPDATE enrolment  
SET enrol_mark = 80,  
    enrol_grade = 'HD'  
WHERE stu_nbr = 12345678  
    and unit_code = 'FIT2077'  
    and enrol_semester = 'S1'  
    and enrol_year = '2016';
```

DELETE

- Removing data from the database

```
DELETE FROM table  
[WHERE condition];
```

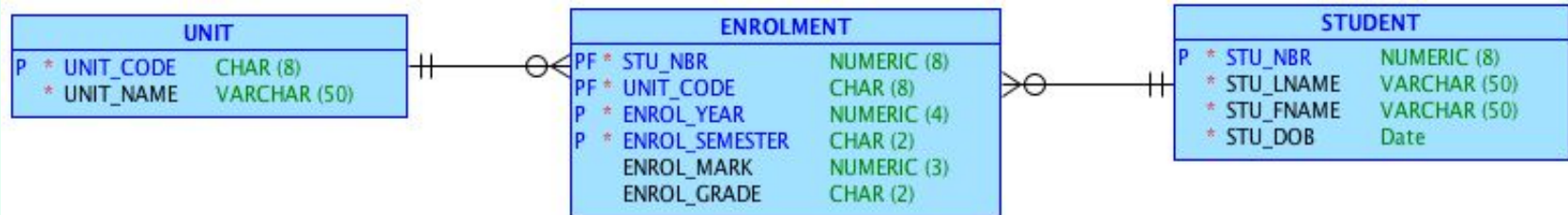
```
DELETE FROM enrolment  
WHERE stu_nbr =12345678  
    and  
        unit_code='FIT2077'  
    and  
        enrol_semester= 'S1'  
    and  
        enrol_year='2016';
```



Q21. Assume that the table ENROLMENT contains enrolment details for students in FIT2077 and FIT2078. The delete referential integrity constraint to UNIT is CASCADE. What would happen to tuples in ENROLMENT with the unit_code FIT2077 when we attempt to delete the FIT2077 record from UNIT?

- A. They will be deleted.
- B. The value of unit_code will be updated to NULL.
- C. The deletion is not possible, the DBMS will prevent the deletion.
- D. None of the above.

mars.mu
Feed Code: **F1PEMT**



Q22. What would happen to the student record with stu_nbr=12345678 in the STUDENT table when we delete all tuples with stu_nbr=12345678 in the ENROLMENT table? (Assume the referential integrity constraint to STUDENT is CASCADE)

- A. Student record with stu_nbr =12345678 in the STUDENT table will be deleted.
- B. Nothing will happen to the STUDENT table.
- C. The stu_nbr =12345678 in the STUDENT table will be updated to NULL.
- D. Deletion will not be permitted

mars.mu
Feed Code: **F1PEMT**