SQL Intermediate

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(stu_nbr)
FROM enrolment
WHERE mark >= 50;
```

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_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)

Q1. What will be displayed by the following SQL statement?

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

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

MONASH University

3

STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_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)

Q2. What will be displayed by the following SQL statement?

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

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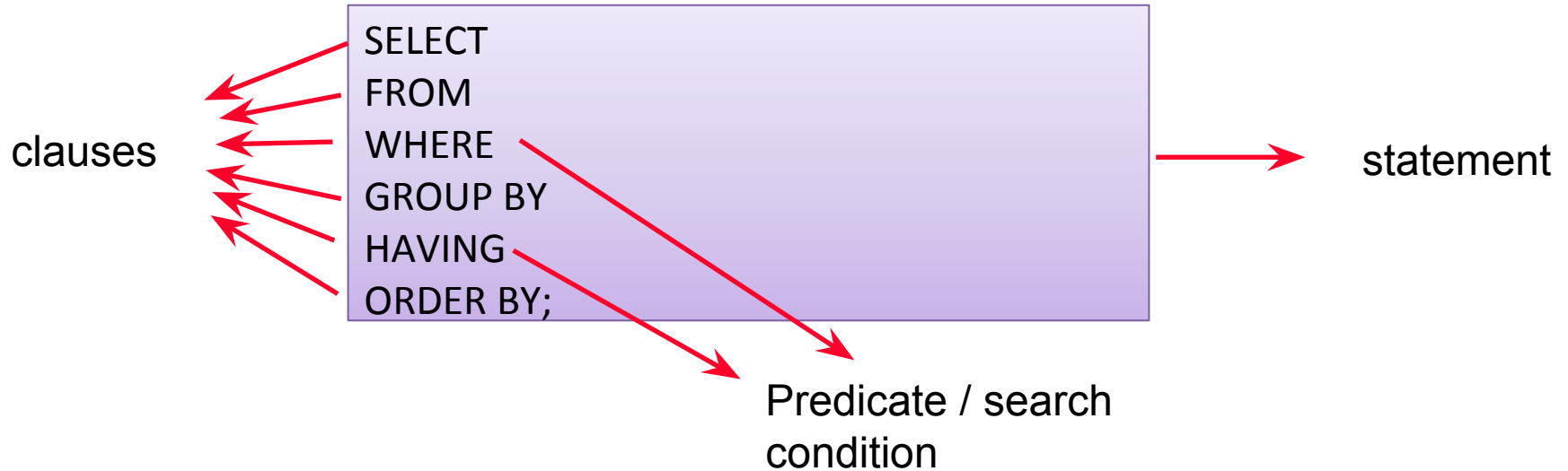- B. 8, 8, 8
- C. 8, 4, 8
- D. 8, 4, 4

STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_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)

Q3. We want to calculate the *average mark of the 8 rows* in the above table. What SQL statement should we use?
Note: We want to calculate (78+35+65)/8=22.25

A. SELECT avg(mark) FROM enrolment;

B. SELECT sum(mark)/count(mark) FROM enrolment;

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

D. SELECT avg(NVL(mark,0)) FROM enrolment;

E. None of the above.

F. More than one option is correct.

Anatomy of an SQL Statement - Revisited

clauses

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY;

statement

Predicate / search condition

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 unit_code, avg(mark)
FROM enrolment
GROUP BY unit_code
ORDER BY unit_code;

```
SQL>
SQL> SELECT avg(mark)
  2  FROM enrolment;

 AVG(MARK)
----------
59.3333333

SQL>
SQL> SELECT unit_code, avg(mark)
  2  FROM enrolment
  3  GROUP BY unit_code
  4  ORDER BY unit_code;

UNIT_CO  AVG(MARK)
------- ----------
FIT1001 59.3333333
FIT1002
FIT1004
```

What output is produced?

SELECT avg(mark)
FROM enrolmentA;

SELECT unit_code, avg(mark)
FROM enrolmentA
GROUP BY unit_code
ORDER BY unit_code;

SELECT unit_code, avg(mark), count(*)
FROM enrolmentA
GROUP BY unit_code
ORDER BY unit_code;

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

```
SQL> SELECT avg(mark)
  2  FROM enrolmentA;


 AVG(MARK)
----------
        56


SQL>
SQL> SELECT unit_code, avg(mark)
  2  FROM enrolmentA
  3  GROUP BY unit_code
  4  ORDER BY unit_code;


UNIT_CO  AVG(MARK)
-------  ----------
FIT2004        60
FIT2094        50


SQL>
SQL> SELECT unit_code, avg(mark), count(*)
  2  FROM enrolmentA
  3  GROUP BY unit_code
  4  ORDER BY unit_code;


UNIT_CO  AVG(MARK)  COUNT(*)
-------  ---------- ----------
FIT2004        60         3
FIT2094        50         2
```

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

MONASH University

What output is produced?

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

SELECT unit_code, avg(mark), count(*)
FROM enrolmentA
GROUP BY unit_code, year
ORDER BY unit_code, year;

```
SQL> SELECT unit_code, avg(mark), count(*)
  2  FROM enrolmentA
  3  GROUP BY unit_code, year
  4  ORDER BY unit_code, year;
```

Note: attributes in the GROUP BY clause do not have to appear in the select list

```
UNIT_CO  AVG(MARK)   COUNT(*)
-------  ----------  ----------
FIT2004         40           2
FIT2004        100           1
FIT2094         20           1
FIT2094         80           1
```

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

```
SQL> SELECT unit_code, year, avg(mark), count(*)
  2  FROM enrolmentA
  3  GROUP BY unit_code, year
  4  ORDER BY unit_code, year;
```

```
UNIT_CO     YEAR  AVG(MARK)   COUNT(*)
-------  ----------  ----------  ----------
FIT2004     2015         40           2
FIT2004     2016        100           1
FIT2094     2015         20           1
FIT2094     2016         80           1
```

HAVING clause

- It is used to put a condition or conditions on the groups defined by GROUP BY clause.

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

What output is produced?

SELECT unit_code, avg(mark), count(*)
FROM enrolmentA
GROUP BY unit_code
HAVING count(*) > 2
ORDER BY unit_code;

SELECT unit_code, avg(mark), count(*)
FROM enrolmentA
GROUP BY unit_code
HAVING avg(mark) > 55
ORDER BY unit_code;

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

```
SQL> SELECT unit_code, avg(mark), count(*)
  2  FROM enrolmentA
  3  GROUP BY unit_code
  4  HAVING count(*) > 2
  5  ORDER BY unit_code;


UNIT_CO  AVG(MARK)   COUNT(*)
-------  ----------  ----------
FIT2004         60           3


SQL>
SQL> SELECT unit_code, avg(mark), count(*)
  2  FROM enrolmentA
  3  GROUP BY unit_code
  4  HAVING avg(mark) > 55
  5  ORDER BY unit_code;


UNIT_CO  AVG(MARK)   COUNT(*)
-------  ----------  ----------
FIT2004         60           3
```

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

MONASH University

HAVING and WHERE clauses

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

- The WHERE clause is applied to ALL rows in the table.

- The HAVING clause is applied to the groups defined by the GROUP BY clause.

- The order of operations performed is FROM, WHERE, GROUP BY, HAVING and then ORDER BY.

- 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 unit_code.

  - If the number of rows in a group is greater than 1, the unit_code and the total is displayed. (due to the HAVING clause)

What output is produced?

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

SELECT unit_code, avg(mark), count(*)
FROM enrolmentA
WHERE year = 2015
GROUP BY unit_code
HAVING avg(mark) > 30
ORDER BY avg(mark) DESC;

```
SQL> SELECT unit_code, avg(mark), count(*)
  2   FROM enrolmentA
  3   WHERE year = 2015
  4   GROUP BY unit_code
  5   HAVING avg(mark) > 30
  6   ORDER BY avg(mark) DESC;


UNIT_CO  AVG(MARK)   COUNT(*)
-------  ----------  ----------
FIT2004          40           2
```

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

MONASH University

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

Q4. What is the output for:

SELECT unit_code, studid, avg(mark)
FROM enrolmentA
GROUP BY unit_code
HAVING avg(mark) > 55
ORDER BY unit_code, studid;

A.   FIT2094, 50, 111

B.   FIT2004, 60, 111

C.   FIT2004, 60, 111, 222, 333

D.   FIT2004, 100, 111

E.   Will print three rows

F.   Error

```
SQL> SELECT unit_code, studid, avg(mark)
  2  FROM enrolmentA
  3  GROUP BY unit_code
  4  HAVING avg(mark) > 55
  5  ORDER BY unit_code, studid;

Error starting at line : 1 in command -
SELECT unit_code, studid, avg(mark)
FROM enrolmentA
GROUP BY unit_code
HAVING avg(mark) > 55
ORDER BY unit_code, studid
Error at Command Line : 1 Column : 19
Error report -
SQL Error: ORA-00979: not a GROUP BY expression
00979. 00000 -  "not a GROUP BY expression"
*Cause:
*Action:
```

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

MONASH University

```
SELECT stu_lname, stu_fname, avg(mark)
FROM enrolment e JOIN student s
        ON s.stu_nbr = e.stu_nbr
GROUP BY s.stu_nbr;
```

The above SQL generates error message

```
SQL Error: ORA-00979: not a GROUP BY expression
00979. 00000 -  "not a GROUP BY expression"
```

Why and how to fix this?
- Why? Because the grouping is based on the stu_nbr, whereas the display is based on stu_lname and stu_fname. The two groups may not have the same members.
- How to fix this?
  - Include the stu_lname,stu_fname as part of the GROUP BY condition.
- Attributes that are used in the SELECT, HAVING and ORDER BY must be included in the GROUP BY clause.

Subqueries

- Query within a query.

"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

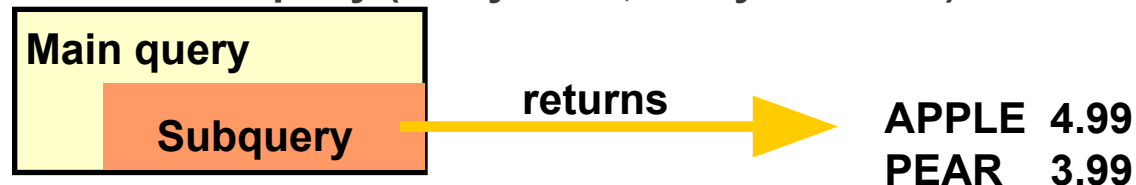**Single-value**

Main query

Subquery → returns → **APPLE**

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

Main query

Subquery → returns → **APPLE PEAR**

Multiple-column subquery (many rows, many columns)

Main query

Subquery → returns → **APPLE 4.99 PEAR 3.99**

Q5. What will be returned by the *inner query*?

SELECT *
FROM enrolment
WHERE mark > (SELECT avg(mark)
          FROM enrolment
          GROUP BY unit_code);

A. A value (a single column, single row).

B. A list of values.

C. Multiple columns, multiple rows.

D. None of the above.

```
SQL> SELECT *
  2  FROM enrolment
  3  WHERE mark > (SELECT avg(mark)
  4                FROM enrolment
  5                GROUP BY unit_code);

Error starting at line : 1 in command -
SELECT *
FROM enrolment
WHERE mark > (SELECT avg(mark)
              FROM enrolment
              GROUP BY unit_code)
Error report -
ORA-01427: single-row subquery returns more than one
row
```

Q6. What will be returned by the *inner query*?

SELECT unit_code, stu_lname, stu_fname, mark
FROM enrolment e join student s
            on e.stu_nbr = s.stu_nbr
WHERE (unit_code, mark) IN (SELECT unit_code, max(mark)
                              FROM enrolment
                              GROUP BY unit_code);

A.   A value (a single column, single row).

B.   A list of values.

C.   Multiple columns, multiple rows.

D.   None of the above.

Comparison Operators for Subquery

- Operator for single value comparison.
  =, <, >

- Operator for multiple rows or a list comparison.
  - equality
    - IN
  - inequality
    - ALL, ANY combined with <, >

STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	MARK	GRADE
1 11111111	FIT1001	2012	1	78	D
2 11111111	FIT1002	2013	1	80	HD
3 11111111	FIT1004	2013	1	85	HD
4 11111112	FIT1001	2012	1	35	N
5 11111112	FIT1001	2013	1	50	P
6 11111113	FIT1001	2012	2	65	C
7 11111113	FIT1004	2013	1	89	HD
8 11111114	FIT1004	2013	1	50	P

Q7. Which row(s) in ENROL2 table will be retrieved by the following SQL statement?

```
SELECT * FROM enrol2
WHERE mark IN (SELECT max(mark)
                      FROM enrol2
                      GROUP BY unit_code);
```

   A.   1, 2, 7

   B.   7

   C.   2, 3, 7

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	MARK	GRADE
1	11111111	FIT1001	2012 1		78	D
2	11111111	FIT1002	2013 1		80	HD
3	11111111	FIT1004	2013 1		85	HD
4	11111112	FIT1001	2012 1		35	N
5	11111112	FIT1001	2013 1		50	P
6	11111113	FIT1001	2012 2		65	C
7	11111113	FIT1004	2013 1		89	HD
8	11111114	FIT1004	2013 1		50	P

```
SQL> SELECT * FROM enrol2
  2  WHERE mark IN (SELECT max(mark)
  3                 FROM enrol2
  4                 GROUP BY unit_code)
  5  ORDER BY stu_nbr, unit_code, enrol_year;

  STU_NBR UNIT_CO ENROL_YEAR E       MARK GRA
---------- ------- ---------- - ---------- ---
  11111111 FIT1001       2012 1         78 D
  11111111 FIT1002       2013 1         80 HD
  11111113 FIT1004       2013 1         89 HD
```

MONASH University

STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	MARK	GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	80	HD
3	11111111	FIT1004	2013	1	85	HD
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	50	P
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	89	HD
8	11111114	FIT1004	2013	1	50	P

UCODE	ROUND(AVG(MARK))
FIT1001	57
FIT1002	80
FIT1004	75

Q8. Which row/s in ENROL2 will be retrieved by the following SQL statement?

SELECT * FROM enrol2
WHERE mark > **ANY** (SELECT avg(mark)
        FROM enrol2
            GROUP BY unit_code);

A.   1, 2, 3, 6, 7

B.   2, 3, 7

C.   3, 7

D.   No rows will be returned

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	MARK	GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	80	HD
3	11111111	FIT1004	2013	1	85	HD
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	50	P
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	89	HD
8	11111114	FIT1004	2013	1	50	P

UCODE	ROUND(AVG(MARK))
FIT1001	57
FIT1002	80
FIT1004	75

```
SQL> SELECT * FROM enrol2
  2  WHERE mark > ANY (SELECT avg(mark)
  3                           FROM enrol2
  4                           GROUP BY unit_code)
  5  ORDER BY stu_nbr, unit_code, enrol_year, enrol_semester;

  STU_NBR UNIT_CO ENROL_YEAR E       MARK GRA
---------- ------- ---------- - ---------- ---
  11111111 FIT1001       2012 1         78 D
  11111111 FIT1002       2013 1         80 HD
  11111111 FIT1004       2013 1         85 HD
  11111113 FIT1001       2012 2         65 C
  11111113 FIT1004       2013 1         89 HD
```

MONASH University

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	MARK	GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	80	HD
3	11111111	FIT1004	2013	1	85	HD
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	50	P
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	89	HD
8	11111114	FIT1004	2013	1	50	P

UCODE	ROUND(AVG(MARK))
FIT1001	57
FIT1002	80
FIT1004	75

Q9. Which row/s in ENROL2 will be retrieved by the following SQL statement?
SELECT * FROM enrol2
WHERE mark > **ALL** (SELECT avg(mark)
        FROM enrol2
            GROUP BY unit_code);

A. 1, 2, 3, 6, 7

B. 2, 3, 7

C. 3, 7

D. No rows will be returned

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	MARK	GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	80	HD
3	11111111	FIT1004	2013	1	85	HD
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	50	P
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	89	HD
8	11111114	FIT1004	2013	1	50	P

UCODE	ROUND(AVG(MARK))
FIT1001	57
FIT1002	80
FIT1004	75

```
SQL> SELECT * FROM enrol2
  2  WHERE mark > ALL (SELECT avg(mark)
  3                         FROM enrol2
  4                         GROUP BY unit_code)
  5  ORDER BY stu_nbr, unit_code, enrol_year, enrol_semester;

  STU_NBR UNIT_CO ENROL_YEAR E       MARK GRA
---------- ------- ---------- - ---------- ---
  11111111 FIT1004       2013 1         85 HD
  11111113 FIT1004       2013 1         89 HD
```

Q10. Find all students whose mark in any enrolled unit is lower than Wendy Wheat's lowest mark for all units she is enrolled in. What would be a possible inner query statement for the above query (assume Wendy Wheat's name is unique)?

A. SELECT min(mark)
   FROM enrol2
   WHERE stu_lname='Wheat' AND stu_fname='Wendy';

B. SELECT min(mark)
   FROM enrol2 e JOIN student s on e.studid = s.studid
   WHERE stu_lname='Wheat' AND stu_fname='Wendy';

C. SELECT min(mark) FROM enrol2;

D. SELECT mark
   FROM enrol2 e JOIN student s on e.studid = s.studid
   WHERE stu_lname='Wheat' AND stu_fname='Wendy';

Summary

- Aggregate Functions
    - count, min, max, avg, sum
- GROUP BY and HAVING clauses.
- Subquery
    - Inner vs outer query
    - comparison operators (IN, ANY, ALL)

STOP HERE for WORKSHOP - practice examples in Workshop Q&A

Practice - Uni Data Model
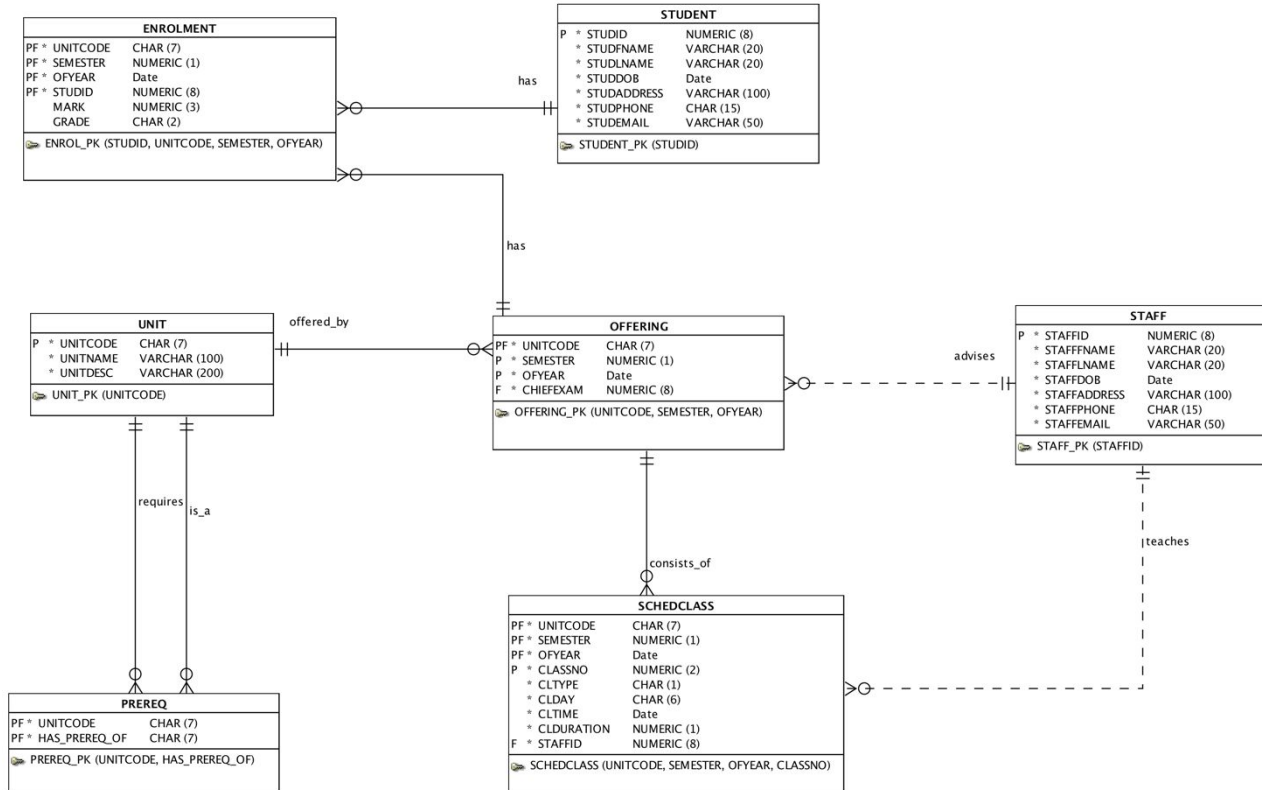
Q1. Display the number of enrolments with a mark assigned, and the average mark for all enrolments

Q2. Select the highest mark ever in any unit

Q3. Select the highest mark ever for each unit (show the unit code only)

Q4. For each student (show the id only), select the highest mark he/she ever received

Q5. For each unit, print unit code, unit name and the highest mark ever in that unit. Print the results in descending order of highest marks.

Q6. For each offering of a unit with marks show the unit code, unitname, offering details and and average mark

Q7. For each student that is enrolled in at least 3 different units, print his/her name and average mark. Also, display the number of units he/she is enrolled in.

Q8. For each unit, count the total number of HDs

Q9. For each unit, print the total number of HDs, Ds, and Cs. The output should contain three columns named unitcode, grade_type, num where grade_type is HD, D or C and num is the number of students that obtained the grade.

Q10. For each unit, print the student ids of the students who obtained maximum marks in that unit