

SQL Intermediate

Aamir Cheema



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

	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

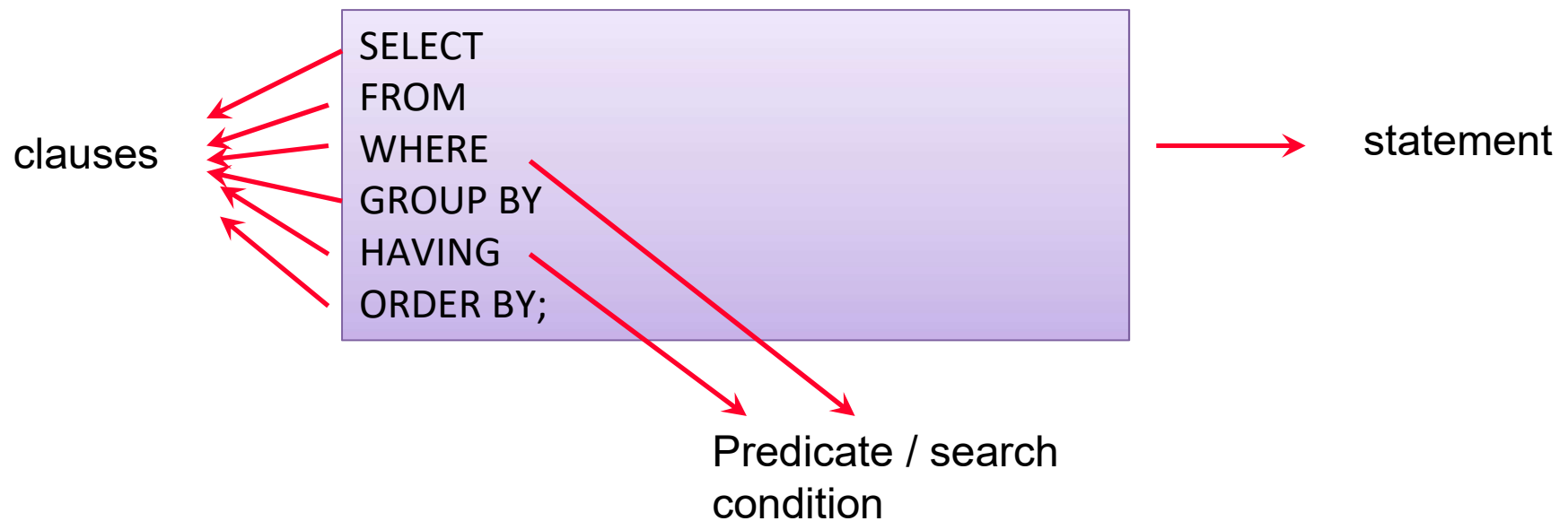
	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	MARK	GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(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 are correct.

Anatomy of an SQL Statement - Revisited



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;
```

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 avg(mark)
FROM enrolment;
```

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

```
SELECT Unit_code, avg(mark), count(*)
FROM enrolment
GROUP 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

```
SELECT Unit_code, avg(mark), count(*)
FROM enrolment
GROUP BY Unit_code, Year;
```

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;
```

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 enrolment
GROUP BY Unit_code
HAVING count(*) > 2;
```

```
SELECT Unit_code, avg(mark), count(*)
FROM enrolment
GROUP BY Unit_code
HAVING avg(mark) > 55;
```

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)

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 enrolment
WHERE year = 2015
GROUP BY Unit_code
HAVING avg(mark) > 30
ORDER BY avg(mark) DESC;

```

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

What is the output for:

```
SELECT Unit_code, avg(mark), Studid
FROM enrolment
GROUP BY Unit_code
HAVING avg(mark) > 55;
```

- 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

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

The above SQL generates error message "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

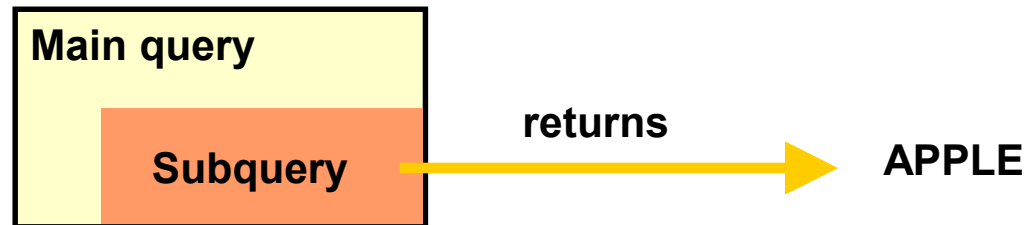
- Query within a query.

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

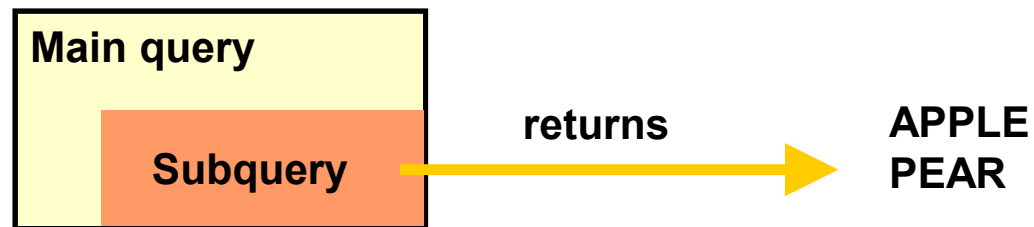
```
SELECT *  
FROM enrolment  
WHERE mark > (SELECT avg (mark)  
               FROM enrolment );
```


Types of Subqueries

Single-value



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



Multiple-column subquery (many rows, many columns)



Q4. 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.

Q5. 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

Q6. 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

Q7. 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);
```

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

- 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

Q8. 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);
```

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

- A. 1,2,3,6,7
- B. 2,3,7
- C. 3,7
- D. No rows will be returned

Q9. Find all students whose mark in any enrolled unit is lower than 'Wendy Wheat's lowest mark of all units she is enrolled.

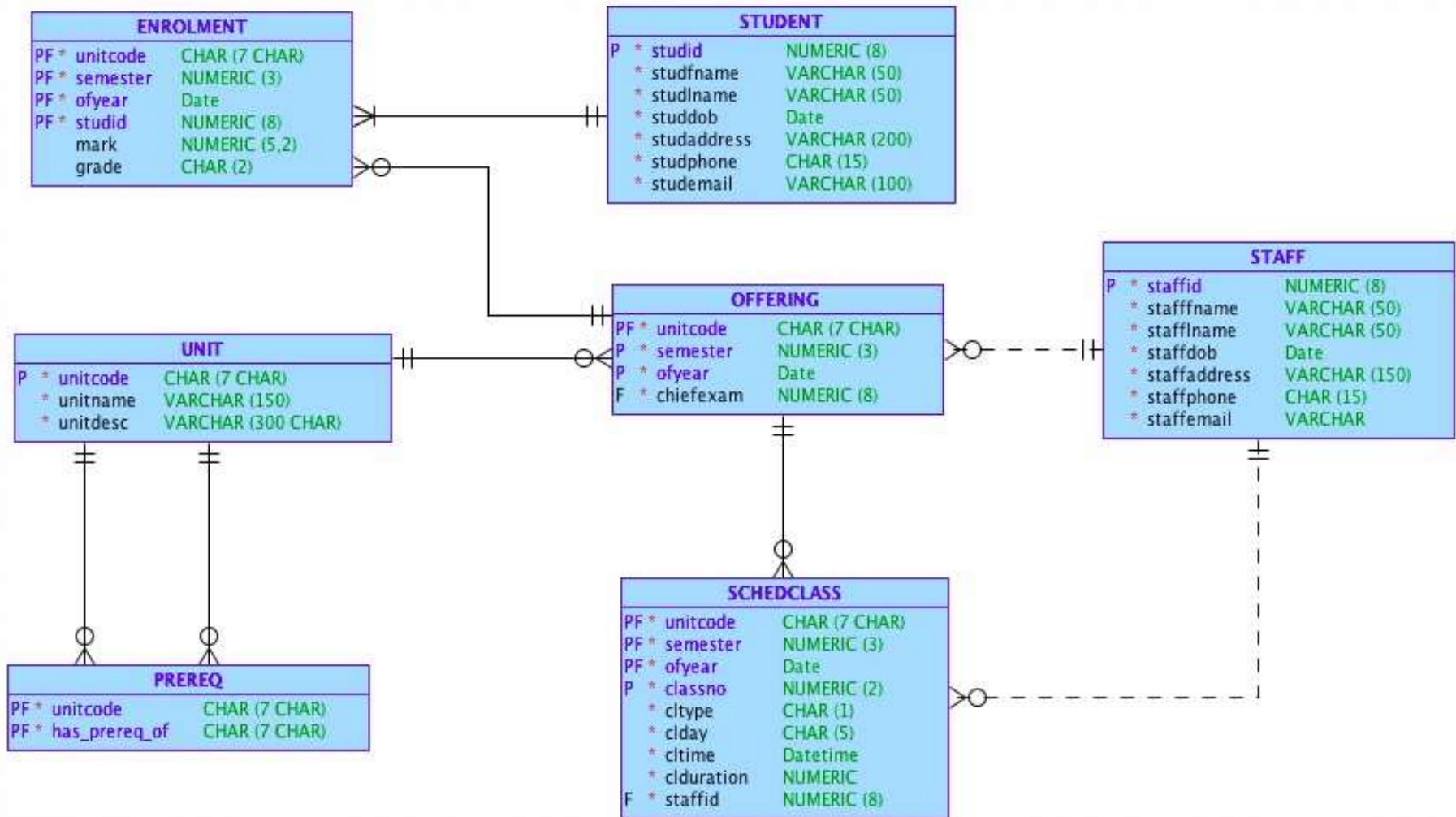
What would be a possible inner query statement for the above query?

- 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)

Practice



University data model