CSE370 LAB 2

SQL (SUB-QUERIES & AGGREGATE FUNCTIONS)

Prepared By: M. Shafiul Alam [FUL/SFA]

MYSQL FUNCTIONS

WHAT ARE AGGREGATE FUNCTIONS?

Functions used in SQL that

- operates on **multiple** values
- returns one output on multiple values

Example: AVG(), COUNT(), MAX(), MIN(), SUM(), STD(), VARIANCE(), etc.

WHAT ARE SCALAR FUNCTIONS?

Functions used in SQL that

- operates on **single** value
- returns one output for each values

Example: UPPER(), LOWER(), ROUND(), REVERSE(), LENGTH(), etc.

AGGREGATE FUNCTIONS

Task 1:

a) Find the minimum CGPA

-> SELECT MIN(cgpa) FROM Lab_Grades;

std_id	name	major	section	days_present	project_ marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

MIN(cgpa)
3.25

MIN(cgpa) returns the minimum value (one output from multiple values)

std_id	name	major	section	days_present	project_ marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades

AGGREGATE FUNCTIONS

Task 1:

b) Find the total number of students and the average project marks

-> SELECT COUNT(*) as total_students, AVG(project_marks) as average_marks FROM Lab_Grades;

total_students	average_marks
7	18.5

Lab_Grades

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

COUNT(*) returns the total number of rows which is 7 (one output from multiple values)

AVG(project_marks) returns the average of all these values (one output from multiple values)

AGGREGATE FUNCTIONS

Task 1:

c) Find the sum of the number of days present

-> SELECT SUM(days_present) FROM Lab_Grades;

SUM(days_present)
70

Lab_Grades

std_id	name	major	section	days_present	project_ marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

SUM(days_present) returns the total sum of the mentioned column

AGGREGATE FUNCTIONS

Task 1:

d) How will you find the last submission date from the given table?

Lab_Grades

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

AGGREGATE FUNCTIONS

Task 1:

d) Find the last submission date

-> SELECT MAX(submission_date) AS last_submission FROM Lab_Grades;

last_submission
2018-09-18

Lab_Grades

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

MAX(submission_date) returns the maximum (latest) value

AGGREGATE FUNCTIONS - GROUP BY AND HAVING

GROUP BY

- Used for grouping rows that have same values
- Mostly used with aggregate functions
- Also used without aggregate functions

Example:

-> SELECT major, MIN(cgpa), MAX(cgpa) from Lab_Grades GROUP BY major;

-> SELECT major FROM Lab_Grades GROUP BY major;

HAVING

- Used for filtering after GROUP BY
- Mostly used with aggregate functions
- WHERE is used before grouping; HAVING is used afterwards

Example:

-> SELECT major, MIN(cgpa), MAX(cgpa) from Lab_Grades GROUP BY major HAVING COUNT(*)>=2;

-> SELECT name, cgpa from Lab_Grades HAVING cgpa>3.7;

ORDER OF EXECUTION

The keywords in a SQL statement are executed in the following order:

1. **FROM:** Specify tables
2. **WHERE:** Filter rows/records
3. **GROUP BY:** Group filtered rows/records
4. **HAVING:** Filter groups
5. **SELECT:** Select columns and expressions
6. **ORDER BY:** Sort the result
7. **LIMIT/OFFSET:** Limit the number of returned rows

i.e. -> SELECT major, AVG(cgpa) AS avg_cgpa FROM Lab_Grades WHERE days_present > 10 GROUP BY major HAVING AVG(cgpa) > 3.5 ORDER BY avg_cgpa DESC LIMIT 1;

Here, FROM is executed first. Afterwards, WHERE is used for filtering. Then, the WHERE filtered rows are GROUP'ed BY major. HAVING is then used for filtering the grouped rows. The mentioned rows are then SELECT'ed. Afterwards, the selected rows are ORDER'ed BY average cgpa in DESC'ending order. Then, the final result rows are LIMIT'ed to first 1 row.

AGGREGATE FUNCTIONS - GROUP BY AND HAVING

Task 1:

e) Find the minimum and maximum CGPA of each major

-> SELECT major, MIN(cgpa), MAX(cgpa)
FROM Lab_Grades GROUP BY major;

major	MIN(cgpa)	MAX(cgpa)
CS	3.57	4.00
CSE	3.70	3.86
ECE	3.25	3.67

Lab_Grades (GROUP BY major)

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

MIN(cgpa) returns the minimum (least) value

MAX(cgpa) returns the maximum value

AGGREGATE FUNCTIONS - GROUP BY AND HAVING

Task 1:

f) Find the total number of students for each major

-> SELECT major, COUNT(*) FROM Lab_Grades GROUP BY major;

major	COUNT(*)
CS	3
CSE	2
ECE	2

Lab_Grades (GROUP BY major)

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

COUNT(*) returns the total number of rows for each group

AGGREGATE FUNCTIONS - GROUP BY AND HAVING

Task 1:

g) Find the minimum and maximum CGPA for each major, but only if there are at least 3 students in the major

-> SELECT major, MIN(cgpa), MAX(cgpa)
FROM Lab_Grades GROUP BY major
HAVING COUNT(*)>=3;

major	MIN(cgpa)	MAX(cgpa)
CS	3.57	4.00

Lab_Grades (GROUP BY major)

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

COUNT(*) returns the total number of rows for each group

MIN(cgpa) returns the minimum (least) value

MAX(cgpa) returns the maximum value

AGGREGATE FUNCTIONS - GROUP BY AND HAVING

Task 1:

h) Find the minimum and maximum CGPA for each major, but consider students who submitted before or on 15th September 2018

-> SELECT major, MIN(cgpa), MAX(cgpa) FROM Lab_Grades WHERE submission_date <= '2018-09-15' GROUP BY major;

major	MIN(cgpa)	MAX(cgpa)
CS	3.91	4.00
CSE	3.70	3.86
ECE	3.25	3.25

i. Lab_Grades (after WHERE filter)

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15

ii. Lab_Grades (after GROUP BY major)

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20

SUB QUERIES/NESTED QUERIES

- A sub-query/nested query is a query within another query
- The inner query is executed first
- The result of the inner query is used by the outer query
- The result can be a single value/multiple values
- The sub-query can be placed after WHERE, FROM, SELECT and HAVING keywords

For example:

-> SELECT name from Lab_Grades WHERE project_marks = (SELECT MAX(project_marks) FROM Lab_Grades)

-> SELECT AVG(cgpa) FROM (SELECT cgpa FROM Lab_Grades WHERE major = 'CSE') AS CSE_CGPAs;

-> SELECT name, (SELECT MAX(cgpa) FROM Lab_Grades) AS MAX_UNIVERSITY_CGPA FROM Lab_Grades;

-> SELECT major, AVG(cgpa) AS avg_cgpa FROM Lab_Grades GROUP BY major HAVING AVG(cgpa) > (SELECT AVG(cgpa) FROM Lab_Grades);

SUB QUERIES/NESTED QUERIES

Task 2:

a) How will you find the names of students who got the highest project marks using a nested query?

Lab_Grades

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

SUB QUERIES/NESTED QUERIES

Task 2:

a) Find the names of students with the highest project marks

-> SELECT name FROM Lab_Grades WHERE project_marks = (SELECT MAX(project_marks) FROM Lab_Grades);

Lab_Grades

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

MAX(project_marks) returns the maximum (most) which is 20

SUB QUERIES/NESTED QUERIES

Task 2:

a) Find the names of students with the highest project marks

-> SELECT name FROM Lab_Grades WHERE project_marks = ~~(SELECT MAX(project_marks) FROM Lab_Grades)~~ 20;

name
Nafis
Arafat

Lab_Grades

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

MAX(project_marks) returns the maximum (most) which is 20

ANY & ALL

- ‘ANY’ and ‘ALL’ are used to compare a value to a set of values returned by the subquery
- Logical operators are used before ‘ANY’ and ‘ALL’ keywords i.e. =, <, >, <=, >=, <>

ANY

- Used to compare a value to **any** value in a list or subquery
- The condition returns true if the comparison is true for **at least one** value

ALL

- Used to compare a value to **all** values in a list or subquery
- The condition returns true if the comparison is true for **all** values

ANY & ALL

ANY

Task 2:

b) Find the CSE students whose CGPA is higher than at least 1 CS students

-> SELECT * FROM Lab_Grades WHERE major = 'CSE' AND cgpa > ANY (SELECT cgpa FROM Lab_Grades WHERE major = 'CS');

std_id	name	major	section	days_present	project_ marks	cgpa	submission_date
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15

> ANY

cgpa
3.91
3.57
4.00

Is 3.86 > 3.91 **or** 3.57 **or** 4.00? **Yes**

Is 3.7 > 3.91 **or** 3.57 **or** 4.00? **Yes**

Thus, the statement will return this table

std_id	name	major	section	days_present	project_ marks	cgpa	submission_date
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15

ALL

Task 2:

c) Find the CSE students whose CGPA is higher than all CS students

-> SELECT * FROM Lab_Grades WHERE major = 'CSE' AND cgpa > ALL (SELECT cgpa FROM Lab_Grades WHERE major = 'CS');

std_id	name	major	section	days_present	project_ marks	cgpa	submission_date
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15

> ALL

cgpa
3.91
3.57
4.00

Is 3.86 > 3.91 **and** 3.57 **and** 4.00? **No**

Is 3.7 > 3.91 **and** 3.57 **and** 4.00? **No**

Therefore, the statement will return an empty set or no table

CORRELATED SUB-QUERIES AND EXISTS

CORRELATED SUB-QUERY

- Executed multiple times, once for each row in the outer query.
- Often involve comparisons between rows of two tables or the same table under different names/aliases.

SUB-QUERY

- Executed once and are not dependent on the outer query.
- Returns a static result

EXISTS

- Used to test for the existence of any record in a sub-query.
- Mostly used in correlated sub-queries.
- Returns true if the sub-query returns one or more records/rows.

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

a) Find the majors for which at least one student has CGPA lower than 3.7

-> SELECT DISTINCT major FROM Lab_Grades L1 WHERE EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.major = L1.major AND L2.cgpa < 3.7);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

a) Find the majors for which at least one student has CGPA lower than 3.7

-> SELECT DISTINCT major FROM Lab_Grades L1 WHERE EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.major = L1.major AND L2.cgpa < 3.7);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS ✓	1	10	18.5	3.91 ✗	2018-09-15
s002	Nafis	CSE ✗	1	12	20	3.86	2018-08-15
s003	Tasneem	CS ✓	1	8	18	3.57 ✓	2018-09-18
s004	Nahid	ECE ✗	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS ✓	2	11	20	4.0 ✗	2018-09-13
s006	Tasneem	CSE ✗	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE ✗	1	10	19	3.67	2018-09-16

✓ will return true

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

a) Find the majors for which at least one student has CGPA lower than 3.7

-> SELECT DISTINCT major FROM Lab_Grades L1 WHERE EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.major = L1.major AND L2.cgpa < 3.7);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

will return false

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

a) Find the majors for which at least one student has CGPA lower than 3.7

-> SELECT DISTINCT major FROM Lab_Grades L1 WHERE EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.major = L1.major AND L2.cgpa < 3.7);

Lab_Grades L1

	std_id	name	major	section	days_present	project_marks	cgpa	submission_date
✓	s001	Abir	CS	1	10	18.5	3.91	2018-09-15
✗	s002	Nafis	CSE	1	12	20	3.86	2018-08-15
✓	s003	Tasneem	CS	1	8	18	3.57	2018-09-18
✓	s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
✓	s005	Arafat	CS	2	11	20	4.0	2018-09-13
✗	s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
✓	s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

a) Find the majors for which at least one student has CGPA lower than 3.7

-> SELECT DISTINCT major FROM Lab_Grades L1 WHERE EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.major = L1.major AND L2.cgpa < 3.7);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

DISTINCT returns the unique values.
Here the distinct majors are CS & ECE

major
CS
ECE

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

b) Find the student names with maximum project marks using EXISTS

-> SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id AND L2.project_marks > L1.project_marks);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

b) Find the student names with maximum project marks using EXISTS

-> SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id AND L2.project_marks > L1.project_marks);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001 ✗	Abir	CS	1	10	18.5	3.91	2018-09-15
s002 ✓	Nafis	CSE	1	12	20 ✓	3.86	2018-08-15 ✓
s003 ✓	Tasneem	CS	1	8	18 ✗	3.57	2018-09-18
s004 ✓	Nahid	ECE	2	7	16.5 ✗	3.25	2018-08-20
s005 ✓	Arafat	CS	2	11	20 ✓	4.0	2018-09-13 ✓
s006 ✓	Tasneem	CSE	1	12	17.5 ✗	3.7	2018-08-15
s007 ✓	Muhtadi	ECE	1	10	19 ✗	3.67	2018-09-16

will return NOT true which is false

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

b) Find the student names with maximum project marks using EXISTS

-> SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id AND L2.project_marks > L1.project_marks);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

will return NOT false which is true

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

b) Find the student names with maximum project marks using EXISTS

-> SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id AND L2.project_marks > L1.project_marks);

Lab_Grades L1

	std_id	name	major	section	days_present	project_marks	cgpa	submission_date
✗	s001	Abir	CS	1	10	18.5	3.91	2018-09-15
✓	s002	Nafis	CSE	1	12	20	3.86	2018-08-15
✗	s003	Tasneem	CS	1	8	18	3.57	2018-09-18
✗	s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
✓	s005	Arafat	CS	2	11	20	4.0	2018-09-13
✗	s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
✗	s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Lab_Grades L2

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

b) Find the student names with maximum project marks using EXISTS

-> SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id AND L2.project_marks > L1.project_marks);

Lab_Grades L1

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s005	Arafat	CS	2	11	20	4.0	2018-09-13

name
Nafis
Arafat

CORRELATED SUB-QUERIES AND EXISTS

Task 3:

b) Find the student names with maximum project marks using EXISTS

-> SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id AND L2.project_marks > L1.project_marks);

Task 3:

c) Find the student names with maximum project marks and who is unique using EXISTS

-> SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id AND L2.project_marks >= L1.project_marks);

What is the difference between the above two queries?

[Hint: One asks for unique and the other doesn't]

First query will return all students who have the highest project marks, even if multiple students have the same maximum project marks. But the second query will only return the student if no other student has marks equal to or greater than theirs. Thus, the second query will return an empty set because both Nafis and Arafat has highest project marks (there are no unique students with maximum project marks).

MANY WAYS TO SOLVE ONE TASK

Task: Find the total number of students who obtained the maximum marks

1. SELECT COUNT(*) FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id != L1.std_id and L2.project_marks > L1.project_marks);
2. SELECT COUNT(*) FROM Lab_Grades L1 WHERE project_marks = (SELECT MAX(project_marks) FROM Lab_Grades);
3. SELECT COUNT(*) FROM Lab_Grades L1 WHERE project_marks >= ALL (SELECT project_marks FROM Lab_Grades);

Bonus Task: Find the major which has the highest number of students enrolled

-> SELECT major FROM Lab_Grades GROUP BY major HAVING COUNT(*) >= ALL (SELECT COUNT(*) FROM Lab_Grades GROUP BY major);

THANK YOU

(GET YOUR ASSIGNMENT 2 DONE NOW)