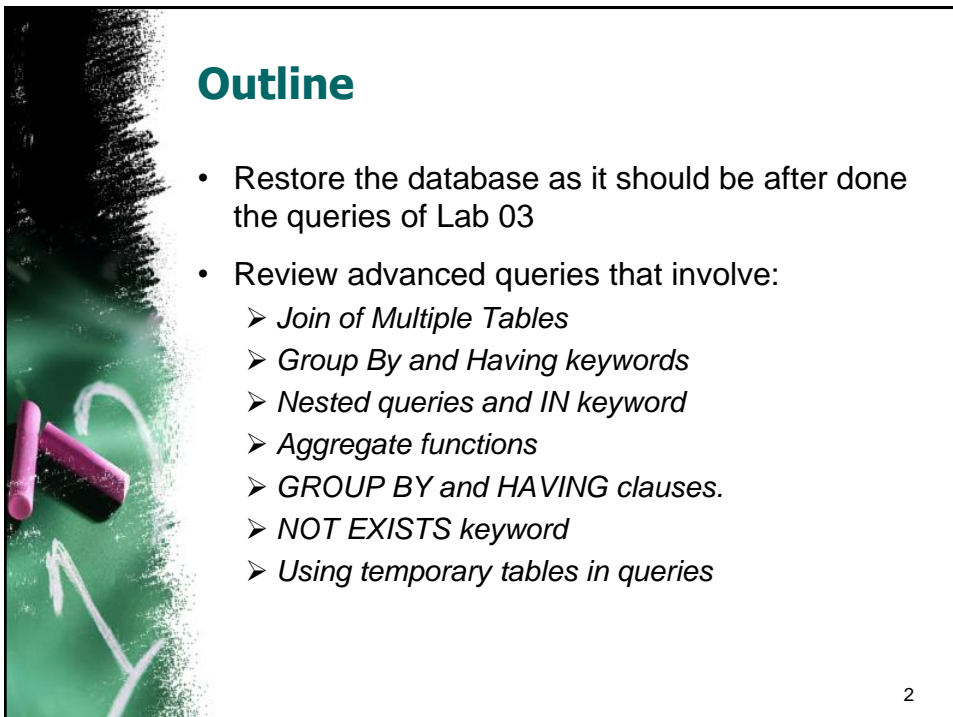




CSI 2132 Lab4

Advanced SQL queries

1



Outline

- Restore the database as it should be after done the queries of Lab 03
- Review advanced queries that involve:
 - *Join of Multiple Tables*
 - *Group By and Having keywords*
 - *Nested queries and IN keyword*
 - *Aggregate functions*
 - *GROUP BY and HAVING clauses.*
 - *NOT EXISTS keyword*
 - *Using temporary tables in queries*

2



Restore the database

- Delete all the tables within “laboratories” schema.
- Download from the course website the following file:

➤ *EndLab03.backup*

This file contains the backup of the database as it should be after done all the queries presented in Lab 03

- Restore the database (explained during previous lab).

3



Write SQL queries for the following

- Note that these queries involve more than one table.
1. List the names and customer ids of all customers who like Picasso.
 2. List the names of all customers who like Artists from the Cubism style and having an amount larger than 30000.

4

Sailors DB

Sailors	
<u>sid</u>	integer
sname	Varchar(50)

Reserves	
<u>sid</u>	integer
<u>bid</u>	integer

Boats	
<u>bid</u>	integer
color	Varchar(20)

5

A nested query using IN keyword in Sailors DB

- Find the names of sailors who have reserved both a red and a green boat.

Sailors	
<u>sid</u>	sname
1	Salvador
2	Rafael

Reserves	
<u>sid</u>	<u>bid</u>
1	1
1	2
2	1

Boats	
<u>bid</u>	color
1	red
2	green

SELECT * FROM Sailors S, Reserves R, Boats B

JOIN					
S.sid	S.sname	R.sid	R.bid	B.bid	B.color
1	Salvador	1	1	1	Red
1	Salvador	1	1	2	green
1	Salvador	1	2	1	red
1	Salvador	1	2	2	green
1	Salvador	2	1	1	red
1	Salvador	2	1	2	green
2	Rafael	1	1	1	red
2	Rafael	1	1	2	green
2	Rafael	1	2	1	red
2	Rafael	1	2	2	green
2	Rafael	2	1	1	red
2	Rafael	2	1	2	green

A nested query using IN keyword in Sailors DB

- Find the names of sailors who have reserved both a red and a green boat.

Sailors	
sid	sname
1	Salvador
2	Rafael

Reserves	
sid	bid
1	1
1	2
2	1

Boats	
bid	color
1	red
2	green

SELECT * **FROM** Sailors, Reserves, Boats

WHERE S.sid = R.sid **AND** R.bid = B.bid

JOIN					
S.sid	S.sname	R.sid	R.bid	B.bid	B.color
1	Salvador	1	1	1	Red
1	Salvador	1	2	2	green
1	Salvador	1	2	1	red
1	Salvador	1	2	2	green
1	Salvador	2	1	1	red
1	Salvador	2	1	2	green
2	Rafael	1	1	1	red
2	Rafael	1	1	2	green
2	Rafael	1	2	1	red
2	Rafael	1	2	2	green
2	Rafael	2	1	1	red
2	Rafael	2	1	2	green

A nested query using IN keyword in Sailors DB

- Find the names of sailors who have reserved **both a red and a green boat**.

Sailors	
sid	sname
1	Salvador
2	Rafael

Reserves	
sid	bid
1	1
1	2
2	1

Boats	
bid	color
1	red
2	green

SELECT * **FROM** Sailors, Reserves, Boats

WHERE Sailors.sid = Reserves.sid **AND** Reserves.bid = Boats.bid

AND

JOIN					
Sailors.sid	Sailors.sname	Reserves.sid	Reserves.bid	Boats.bid	Boats.color
1	Salvador	1	1	1	Red
1	Salvador	1	2	2	green
2	Rafael	2	1	1	red

A nested query using IN keyword in Sailors DB

- Find the names of sailors who have reserved both a red and a green boat.

```
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
      AND S.sid IN (SELECT S2.sid
                    FROM Sailors S2,Boats B2, Reserves R2
                    WHERE S2.sid = R2.sid AND
                        R2.bid = B2.bid AND
                        B2.color = 'green' );
```

- The query between `()` will return the sailor IDs who have reserved a green boat.
- First three lines of the query will find sailors who reserved a red boat.
- Thus, in the 4th line, we will have the ID of a sailor who reserved a red boat, and check if this same sailor also reserved a green boat by `IN` keyword.

Your turn

- Using our Artist database, find names of Customers who likes both an artist born in Malaga and an artist born in Florence.
- You need the data that we inserted in Lab2 and Lab3.
- Answer this question by writing a similar query that is described in previous slide.

Sailors DB (2)

Sailors	
<u>sid</u>	integer
sname	Varchar(50)
age	integer
rating	Varchar(20) {good,fair,poor}

Reserves	
<u>sid</u>	integer
<u>bid</u>	integer

Boats	
<u>bid</u>	integer
color	Varchar(20)

11

More on SELECT statements

Remember (previous lab) the order and the syntax for **GROUP BY** and **HAVING** clauses

SELECT select-list
FROM from-list
WHERE record-qualification
GROUP BY grouping-list
HAVING group-qualification

12

A query using aggregate function AVG, and GROUP BY and HAVING clauses in Sailors DB

- Find the **average age** of sailors who are at least 18 years old, for each **rating level** that has at least two such sailors.

Sailors			
sid	sname	age	rating
1	Salvador	26	Good
2	Rafael	28	Good
3	John	10	Good
4	Bruce	18	Fair
5	James	17	Fair
6	Smith	18	Poor
7	Peter	22	Poor

```
SELECT *
FROM Sailors S
WHERE S.age >= 18
```

- WHERE clause eliminates all the sailors whose age is lesser than 18.

Sailors			
sid	sname	age	rating
1	Salvador	26	Good
2	Rafael	28	Good
4	Bruce	18	Fair
6	Smith	18	Poor
7	Peter	22	Poor

13

A query using aggregate function AVG, and GROUP BY and HAVING clauses in Sailors DB

- Find the **average age** of sailors who are at least 18 years old, for each **rating level** that has at least two such sailors.

Sailors			
sid	sname	age	rating
1	Salvador	26	Good
2	Rafael	28	Good
4	Bruce	18	Fair
6	Smith	18	Poor
7	Peter	22	Poor

```
SELECT S.rating,
       AVG(S.age) AS avgage,
       COUNT(*) numssailors
FROM   Sailors S
WHERE  S.age >= 18
GROUP BY S.rating
```

- The remaining rows will be grouped by their rating using GROUP BY clause and we also obtain the average age and the number of sailors in each group.
- Up to now, we have sailors who are older than 17 grouped by their rating.

Sailors		
rating	avgage	numssailors
Good	27	2
Fair	18	1
Poor	20	2

14

A query using aggregate function AVG, and GROUP BY and HAVING clauses in Sailors DB

- Find the **average age** of sailors who are at least 18 years old, for each rating level that has at least two such sailors.

Sailors		
rating	avgage	numsailors
Goog	27	2
Fair	18	1
Poor	20	2

```
SELECT S.rating,
       AVG(S.age) AS avgage
FROM   Sailors S
WHERE  S.age >= 18
GROUP BY S.rating
HAVING COUNT(*) > 1
```

- HAVING clause allows us to specify a qualification (filter) for each group.
- COUNT(*) > 1 in the HAVING clause eliminates all the rating groups which do not have at least two sailors.

Sailors	
rating	avgage
Goog	27
Poor	20

- TIP: WHERE → SELECT
HAVING → GROUP BY

15

Another query similar to previous one

- Find the age of the **youngest sailor with age > 18**, for each rating level **with at least 2 sailors (of any age)**

```
SELECT S.rating, MIN(S.age) AS minage
FROM   Sailors S
WHERE  S.age >= 18
GROUP BY S.rating
HAVING (
        SELECT COUNT (*)
        FROM   Sailors S2
        WHERE  S.rating=S2.rating
    ) >= 2
```

- All clauses except HAVING clause are similar to the previous query.
- This time, since group qualification is having 2 sailors of any age, we get the total number of rows for that rating group with the query inside HAVING clause, and check if this number is at least 2.

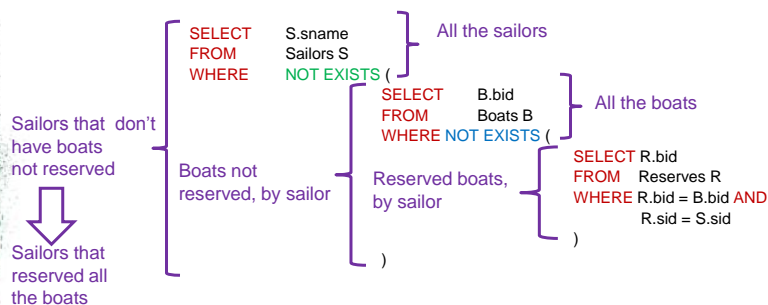
Your turn

- You need to insert following rows to Artwork table.
 - ('Saints', 1470, 'Renaissance', 30000.00, 'Leonardo')
 - ('Hand of god', 1510, 'Renaissance', 52000.00, 'Michelangelo')
 - ('Murder', 1600, 'Baroque', 15000.00, 'Caravaggio')
 - ('Green', 1950, 'Modern', 5000.00, 'John')
- Find the **average price** of artworks which are **painted after 1490**, for each **artwork type** that has **at least two** such artworks.
- And find the **average price** of artworks which are **painted after 1490**, for each **artwork type** that has **at least two** artworks (**painted in any year**).
- Write a query similar to what is described in previous slides to answer the question.

17

A nested query using NOT EXISTS

- Find name of the sailors who **reserved all the boats**.



- The intuition behind this query is:
 - Find name of the sailors such that; there is **no boat** that he/she **did not reserve**.
 - Logically equivalent to 'Find name of the sailors who reserved all the boats'.

18

Your turn

- You need the following values inserted into LikeArtist table first.
 - (2,'Caravaggio')
 - (2,'Hans Hofmann')
 - (2,'John')
 - (2,'Josefa')
 - (2,'Michelangelo')
- Find names of **customers** who **like all the artists**. You can answer this query using what you have learned in previous slide.

19

Using temporary tables in queries.

- We can generate temporary tables and refer to their rows in our queries.
- Find those **ratings** for which the **average age is minimum over all ratings**.

Average age per rating where the average age is equals to the minimum average age per rating

```

SELECT Tmp.rating, Tmp.avgage
FROM (
  SELECT S.rating, AVG(S.age) AS avgage
  FROM Sailors S
  GROUP BY S.rating
) AS Tmp
WHERE Tmp.avgage = (
  SELECT MIN(T.avgage)
  FROM (
    SELECT S.rating, AVG(S.age) AS avgage
    FROM Sailors S
    GROUP BY S.rating
  ) AS T
)

```

Average ages per rating

Minimum average age

- Table Tmp and T will store average ages for all the ratings.
- Query in WHERE clause will return a single value, which is the minimum of all the average ages.
- WHERE clause selects the rows where avgage equals to minimum average age.

20



Your turn

- First, delete a row that we have inserted in previous lab.

➤ *DELETE FROM Artwork WHERE price = 4000.00;*

- Find those **painting types** for which the **average price** is the **minimum over all types**.