upGrad

PLACEMENT REFRESHER PROGRAM

Session 3 - SQL 1 : Basics of SQL

Introduction, Clauses, Aggregate Functions, CRUD Operations,

By
Ritesh Kumar Pandey

Agenda

- Introduction
- SQL Components
- SQL Constraints
- SQL Syntax
- CRUD Operations
- Aggregate Functions

Introduction

upGrad

SQL - **'Structured Query Language'**

SQL is _____

1. Programming Language
2. General Purpose Programming Language
3. Domain Specific Language
4. Turing Complete

SQL - **'Structured Query Language'**

SQL is _____

1. Programming Language
2. General Purpose Programming Language
3. Domain Specific Language
4. Turing Complete

Introduction

upGrad

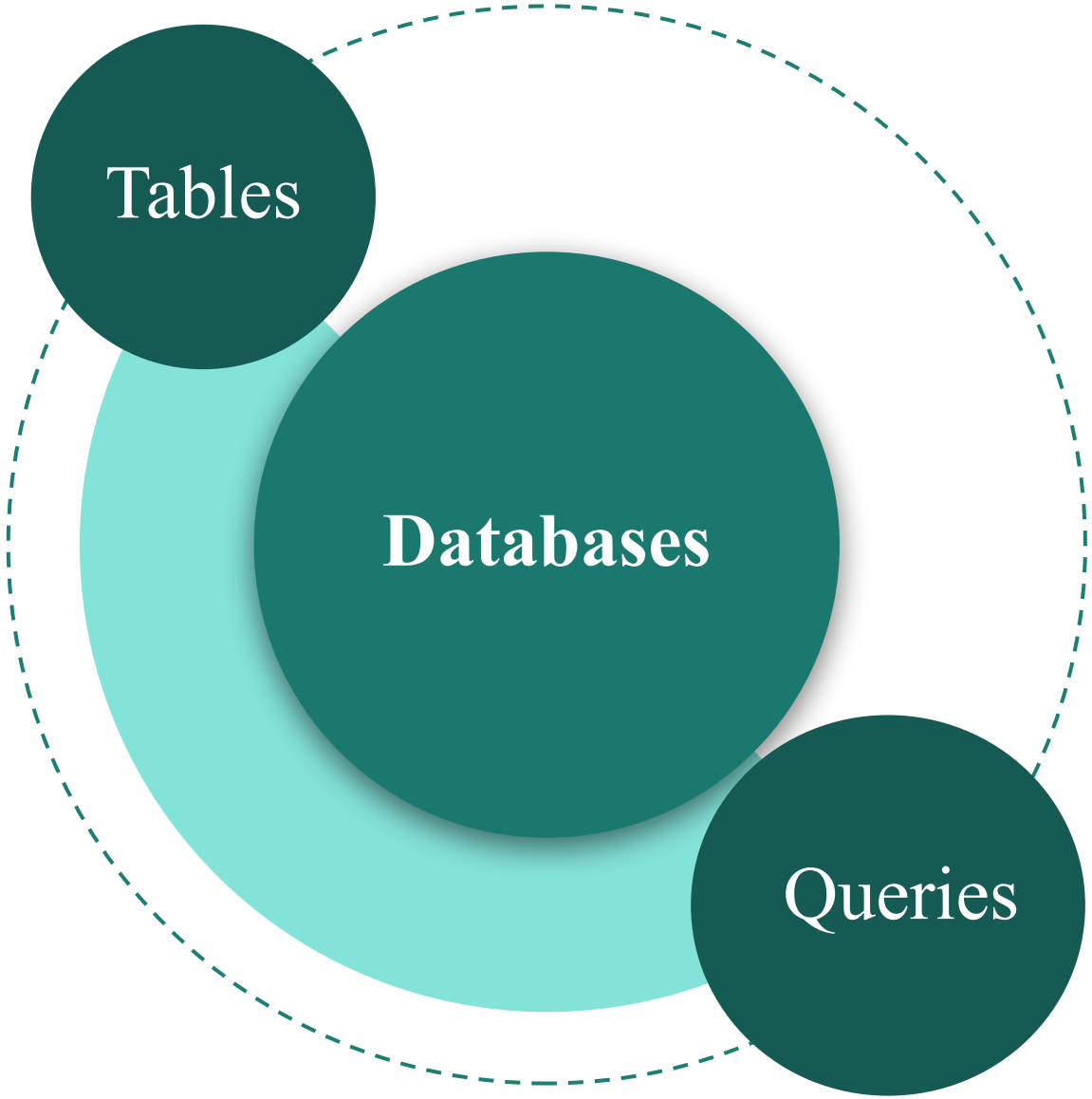
- SQL is used to interact with databases to perform various operations, including data retrieval, data insertion, data modification, and data deletion.

- It is the standard language for relational database management systems (RDBMS) such as MySQL, PostgreSQL, Oracle, SQL Server, and SQLite.

SQL Components

upGrad

Structures within database which hold data



SQL statements used to interact with data

SQL Components

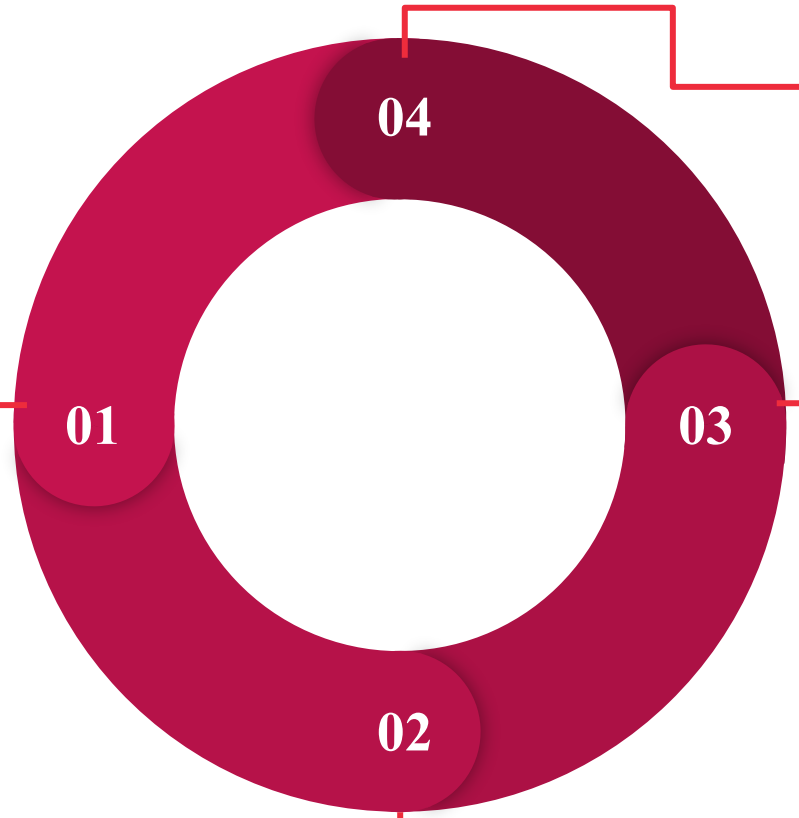
upGrad

DDL

- Data Definition Language
- Create, Alter, Drop

DQL

- Data Query Language
- Select



DML

- Data Manipulation Language
- Insert, Update, Delete

DCL

- Data Control Language
- Grant, Revoke

What is the difference between a Primary Key and a Unique Key ?

What is the difference between a Primary Key and a Unique Key ?

Primary Key	Unique Key
Cannot contain Null Value	Can contain only one Null Value
Only one Primary Key per table	Multiple Unique Keys per table

SQL Constraints

upGrad

- **NOT NULL** - Surely field gets value for every row
- **DEFAULT** - If u didnt give a value it default value will be given to field.
- **PRIMARY KEY** - not null+unique
- **FOREIGN KEY** - references a column of another table (mostly primary key)
- **UNIQUE** - All the field values must be different, but it allow one null value.
- **CHECK CONSTRAINT** - specifies a requirement that must be met by each row in database

SQL Syntax

upGrad

- A typical SQL query has the form:

```
select column1, column2, …., columnN
from table1, table2, …., tableM
where P
```

  - column represents an attribute
  - table represents a relation
  - P is a predicate / condition.

- The result of an SQL query is a relation

students

ID	NAME	GRADE
1	John	B
13	Chris	B
28	Shawn	A

select NAME from students;

select name from students;

students

ID	NAME	GRADE
1	John	B
13	Chris	B
28	Shawn	A

```
select NAME from students;
```

```
select name from students;
```

SQL is case **insensitive**

CRUD Operations

upGrad

- Create- CREATE Database, CREATE Table, INSERT Into
- Retrieve - SELECT
- Update - UPDATE, SET
- Delete - DELETE FROM

CRUD Operations - Create

upGrad

Create Database:

```
create database students;


use students;
```

Create Table:

```
create table student_info (
    student_id int primary key not null,
    student_name varchar(255) not null,
    department varchar(255),
    marks int
);
```

Insert into:

```
insert into student_info (student_id, student_name, department, marks)
values (12, 'Emily', 'IT', 85);
```

Which of the following is correct ?

1. select * from students;
2. select distinct name from students;
3. select 28;
4. select 28 as ‘No’;

Which of the following is correct ?

1. select * from students;
2. select distinct name from students;
3. select 28;
4. select 28 as ‘No’;

ALL OF THE ABOVE

CRUD Operations - Retrieve

upGrad

select * from students;

ID	NAME	GRADE
1	John	B
13	Chris	B
28	Shawn	A
39	Chris	C

select 28;

28
28

select distinct name from students;

NAME
John
Chris
Shawn

select 28 as No;

No
28

An attribute can be a literal with no **from** clause

CRUD Operations - Update

upGrad

Update the name of the student to **Austin** whose ID is 10

Update the name of the student to **Jenny** and **marks** to 72 whose ID is 10

CRUD Operations - Update

upGrad

Update the name of the student to **Austin** whose ID is 10

```
update student_info set student_name = 'Austin' where student_id = 10;
```

Update the name of the student to **Jenny** and **marks** to 72 whose ID is 10

```
update student_info set student_name = 'Jenny', marks = 72 where student_id = 10;
```

CRUD Operations - Delete

upGrad

Update the student_info whose department is **History**

CRUD Operations - Delete

upGrad

Update the student_info whose department is **History**

```
delete from student_info where department = 'History';
```

Clauses - WHERE Clause

upGrad

- The **where** clause specifies conditions that the result must satisfy
- To find all instructors in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.'

- Comparison results can be combined using the logical connectives **and, or,** and **not**
- To find all instructors in Comp. Sci. dept with salary > 80000

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000

- Comparisons can be applied to results of arithmetic expressions.

Aggregate Functions & Group By Clause

upGrad

These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value

min: minimum value

max: maximum value

sum: sum of values

count: number of values

Aggregate Functions & Group By Clause

upGrad

Find the average salary of instructors in each department

```
select dept_name, avg (salary) as salary
from instructor
group by dept_name;
```

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

Aggregate Functions & Group By Clause

upGrad

Find the average salary of instructors in each department

```
select dept_name, avg (salary) as salary
from instructor
group by dept_name;
```

dept_name	salary
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

Aggregate Functions & Group By Clause

upGrad

Find out the maximum marks in each department

Aggregate Functions & Group By Clause

upGrad

Find out the maximum marks in each department

```
select department, max(marks) as marks from student_info group by
department;
```

Common Interview Questions

upGrad

What is the difference between the SQL COUNT(*) and COUNT(column_name) functions?

Common Interview Questions

upGrad

What is the difference between the SQL COUNT(*) and COUNT(column_name) functions?

```
select count(*) from student_info;

#Answer: 8
```

```
select count(department) from student_info;

#Answer: 7
```

Common Interview Questions

upGrad

What is the difference between the SQL COUNT(*) and COUNT(column_name) functions?

- **COUNT(*):**
  - Counts all rows in a table, including rows with NULL values.
  - Returns the total number of rows in the result set.

- **COUNT(column_name):**
  - Counts the number of non-null values in a specific column.
  - Ignores NULL values in the specified column.

Common Interview Questions

upGrad

What is the HAVING clause, & how does it differ from the WHERE clause?

Common Interview Questions

upGrad

What is the HAVING clause, & how does it differ from the WHERE clause?

Find out the department whose maximum marks is less than 80

What is the HAVING clause, & how does it differ from the WHERE clause?

Find out the department whose maximum marks is less than 80

```
select department from student_info group by department having max(marks) < 80;
```

Common Interview Questions

upGrad

What is the HAVING clause, & how does it differ from the WHERE clause?

- **WHERE Clause:** The WHERE clause is used to filter rows before they are grouped, and it filters individual rows based on column values.
- **HAVING Clause:** The HAVING clause is used to filter groups of rows after they are grouped using the GROUP BY clause, and it filters based on aggregated values (e.g., the result of an aggregate function like SUM or AVG).

- **WHERE Clause:** Typically placed before the GROUP BY clause.
- **HAVING Clause:** Placed after the GROUP BY clause.

QUESTIONS

upGrad

Retrieve the top 3 students from the "student_info" table, ordered by marks in descending order.

QUESTIONS

upGrad

Retrieve the top 3 students from the "student_info" table, ordered by marks in descending order.

```
select * from student_info order by marks desc limit 3;
```

QUESTIONS

upGrad

Retrieve the average marks of students in the "IT" department, but only consider students having marks more than 70.

QUESTIONS

upGrad

Retrieve the average marks of students in the "IT" department, but only consider students having marks more than 70.

```
SELECT AVG(marks) AS average_marks
FROM student_info
WHERE department = 'IT'
AND marks > 70;
```

QUESTIONS

upGrad

Write an SQL query to find all students where the student name contains the letter "a," but the department does not contain the letter "t."

QUESTIONS

upGrad

Write an SQL query to find all students where the student name contains the letter "a," but the department does not contain the letter "t."

```
SELECT *
FROM student_info
WHERE student_name LIKE '%a%'
AND department NOT LIKE '%t%';
```

Common Interview Questions

upGrad

Explain the difference between SQL and NoSQL databases.

Explain the difference between SQL and NoSQL databases.

Aspect	SQL Databases	NoSQL Databases
Data Structure	Structured Data with tables and fixed schemas	Unstructured, Semi-structured or Structured Data without strict schemas
Scalability	Scales Vertically	Scales Horizontally
Schema	Rigid	Flexible
Use Cases	Financial Systems	Social Media, IoT

QUESTIONS

upGrad

You have a table named "orders" with the following columns: "order_id" (primary key), "order_date," "customer_id," and "total_amount." You need to Update the order date for all orders placed by customers to ‘2022-10-15’ whose email addresses contain the word "gmail" and average total_amount is more than 5000.

QUESTIONS

upGrad

You have a table named "orders" with the following columns: "order_id" (primary key), "order_date," "customer_id," and "total_amount." You need to Update the order date for all orders placed by customers to ‘2022-10-15’ whose email addresses contain the word "gmail" and average total_amount is more than 5000.

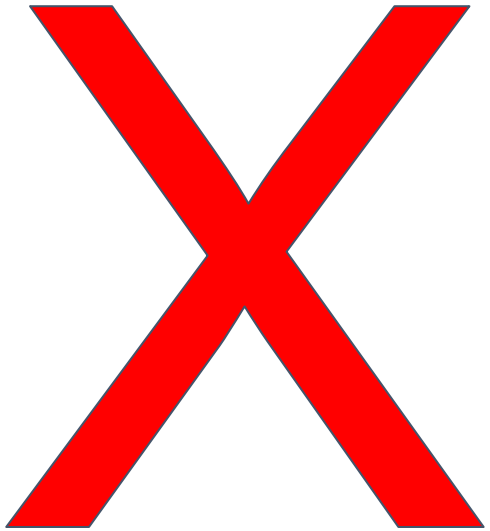
```
UPDATE orders
SET order_date = '2023-10-15'
WHERE email_address LIKE '%gmail%'
GROUP BY customer_id
HAVING AVG(total_amount) > 5000
```

QUESTIONS

upGrad

You have a table named "orders" with the following columns: "order_id" (primary key), "order_date," "customer_id," and "total_amount." You need to Update the order date for all orders placed by customers to ‘2022-10-15’ whose email addresses contain the word "gmail" and average total_amount is more than 5000.

```
UPDATE orders
SET order_date = '2023-10-15'
WHERE email_address LIKE '%gmail%'
GROUP BY customer_id
HAVING AVG(total_amount) > 5000
```



QUESTIONS

upGrad

You cannot use the GROUP BY and HAVING clauses directly in an UPDATE statement. However, you can achieve the desired result by first identifying the customer IDs that meet the criteria and then using an UPDATE statement to modify the orders associated with those customers.

QUESTIONS

upGrad

You cannot use the GROUP BY and HAVING clauses directly in an UPDATE statement. However, you can achieve the desired result by first identifying the customer IDs that meet the criteria and then using an UPDATE statement to modify the orders associated with those customers.

```
UPDATE orders SET order_date = '2023-10-15'
WHERE customer_id IN (
    SELECT customer_id FROM customers
    WHERE email_address LIKE '%gmail%'
    GROUP BY customer_id HAVING AVG(total_amount) > 5000
);
```

THANK YOU