

## ANSI SQL

## SQL Operators

LEVEL – LEARNER



# Icons Used



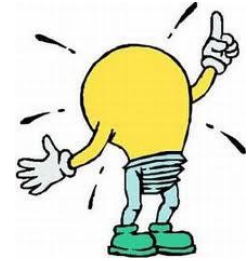
Hands-on Exercise



Reference



Questions



Points To Ponder



Coding Standards



Lend A Hand



Summary



Test Your Understanding

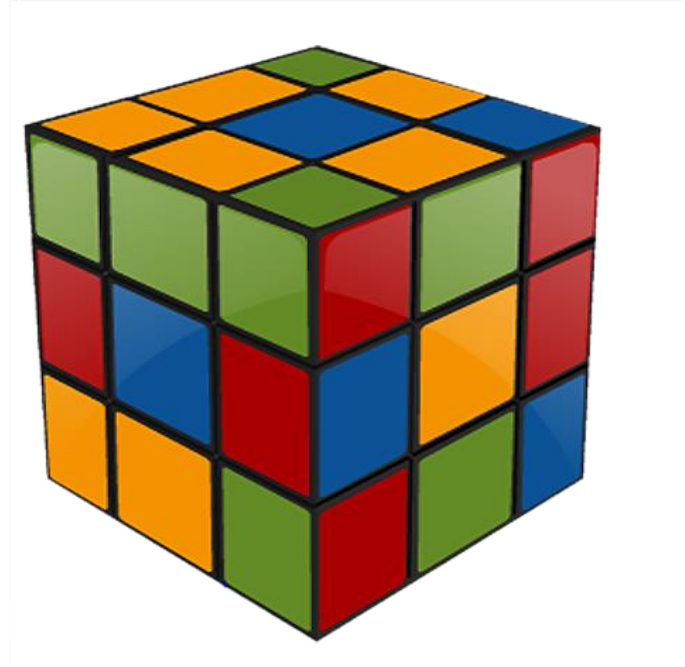
# Overview



This session on SQL operators, provides knowledge and understanding on the use of operators available in ANSI. It also demonstrates the application of the syntax learned as part of this session in a case study provided.

# Objectives

- After completing this session, you will be able to:
  - Describe the Arithmetic operators.
  - Describe the Comparison operators.
  - Describe the Logical operators.
  - Describe the Set operators.





# Scenario

- To understand ANSI SQL in detail, we are going to make use of **Product Management System (PMS)**, for ABC Traders.
- ABC Traders is a company that buys collectible model cars, trains, trucks, buses, trains, and ships directly from manufacturers and sells them to distributors across the globe. In order to manage the stocking, supply and payment transactions, this software is developed.
- As per the requirement of the trading company, an inventory system is developed to collect the information of products, customers, and their payment processing.



# Database Tables

- There are many entities involved in **Product Management System**.
- Here are the entities, which we will be dealing with throughout this course:

## Offices

To maintain information of Offices.

For instance: Office code, address, city, and so on.

## Customer

To maintain customer details.

For instance: Customer Name, address, and so on.

## Employees

To maintain employee details.

For instance: ID, Name, and so on.

## Products

To maintain information of products.

For instance: Product ID, name, and so on.

## Payments

To maintain information of payments done.

For instance: Payment date, amount, and so on.

## Orders

To maintain Orders done by customers.

For instance: Order no., date, and so on.

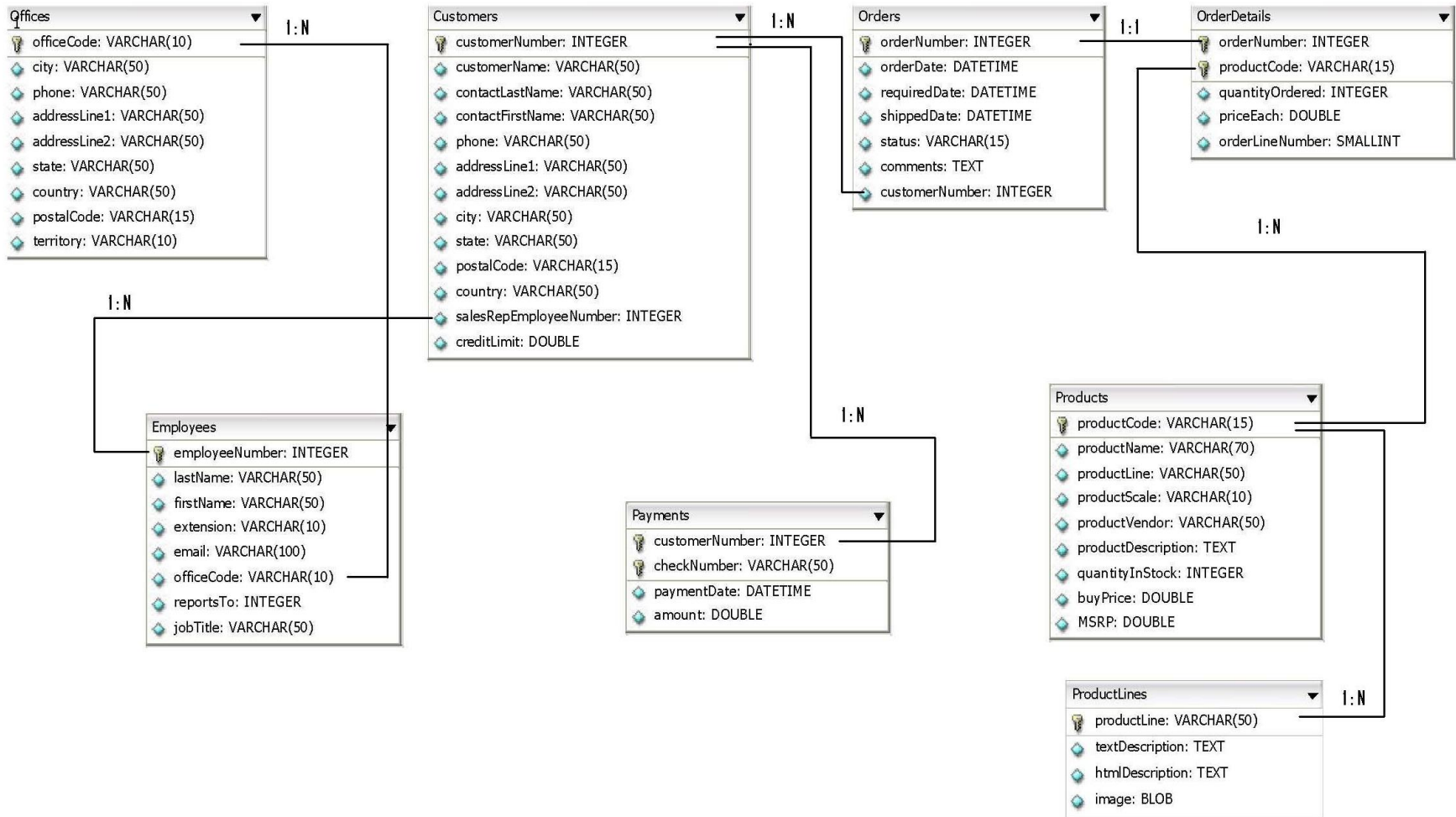
## Order Details

To maintain Orders done by customers.

For instance: Order no., date, and so on.



# Schema Diagram



# SQL Operators



Hi!

Now that you have created tables and applied constraints to tables, I would want you to take care of some requirements, which involve adding data from two columns and display it in single column.

To help us meet Tim's requirements, let us learn about arithmetic operators.



# SQL Operators



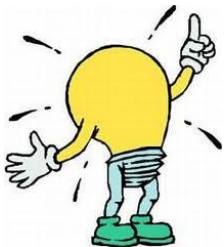
- What are SQL Operators?

- An SQL Operator is used for processing data values (stored in columns of tables) after which it returns a result. The data values are called operands.
- SQL Operators are represented by special characters or by keywords.
- The operators supported by ANSI SQL are listed below:
  - Arithmetic operators
  - Comparison operators
  - Logical operators
  - Set, Union, Intersect & Minus Operators



# Arithmetic Operators

- Arithmetic Operators:
  - Arithmetic operators are used to manipulate numeric operands, which are columns storing numeric values.
- Types of arithmetic operators:
  - **Monadic Arithmetic Operators**, which are namely,
    - $+$  and  $-$
  - **Dyadic Arithmetic Operators**, which are namely,
    - $/$ ,  $*$ ,  $+$ , and  $-$ .

**Rule:**

If the value of any operand in a numeric value expression is null value, then the result of that numeric value expression is the null value.

# Arithmetic Operators

- Here are some examples of the arithmetic operators:

Operator	Description	Example
+(monadic)	Makes operand positive	SELECT + Creditlimit FROM Customers;
-(monadic)	Makes operand negative	SELECT - Creditlimit FROM Customers;
/	Division(Used with Number and Date)	SELECT Creditlimit / 10 FROM Customers;
*	Multiplication	SELECT Creditlimit * 10 FROM Customers;
+	Addition (numbers and dates)	SELECT Creditlimit + 1000 FROM Customers;
-	Subtraction (numbers and dates)	SELECT Creditlimit - 500 FROM Customers;

# Scenario



Hurray!  
You have understood my requirement  
and provided the solution!  
My next requirement is to find the  
customers whose credit limit is in the  
range between 10000 and 15000.

Let's learn about comparison operators which will help us meet Tim's requirements..



# Comparison Operators

- Comparison Operators:
  - Comparison operators are used in conditions that compare one operand with another. The result of a comparison can be TRUE (or) FALSE (or) NULL.
- Types of comparison operators:

Comparison operator	Name
=	equals operator
!=, <>	not equals operator
<	less than operator
>	greater than operator
<=	less than or equals operator
>=	greater than or equals operator

# Comparison Operators

Operators	Syntax
between predicate	[ NOT ] BETWEEN <row value predicand> AND <row value predicand>
in predicate	[ NOT ] IN <in predicate value>
character like predicate	[ NOT ] LIKE <character pattern> [ ESCAPE <escape character> ]
null predicate	IS [ NOT ] NULL
exists predicate	EXISTS <table sub query>
quantifier	<all>   <some>
all	ALL
some	SOME   ANY



# Comparison Operators

- The comparison operators displayed below are used in conditions that compare one operand with another. The result of a comparison can be TRUE (or) FALSE.

Operator	Description	Example
=	Equality Test	SELECT CustomerName FROM Customers WHERE Country ='USA';
!=, <>	Inequality Test	SELECT CustomerName FROM Customers WHERE Country !='USA';
>	Greater than test	SELECT CustomerName FROM Customers WHERE creditLimit > 5000;
<	Less than test	SELECT CustomerName FROM Customers WHERE creditLimit < 10000;
>=	Greater than or equal to test.	SELECT CustomerName FROM Customers WHERE CreditLimit >= 5000;
<=	Less than or equal to test.	SELECT CustomerName FROM Customers WHERE CreditLimit <= 10000;

# Comparison Operators

- The comparison operators displayed below are used in conditions that compare a particular value to each value in a list.

Operator	Description	Example
IN	Equivalent to comparing the operand value with a list of values and if any match happens it returns true.	SELECT CustomerName FROM Customers WHERE Country IN ('USA', 'Norway');
NOT IN	Equivalent to comparing the operand value with a list of values and if any match happens it returns true.	SELECT CustomerName FROM Customers WHERE Country NOT IN ('USA', 'Norway');

# Comparison Operators



Operator	Description	Example
BETWEEN AND	Checks whether the operand value falls within a range. A range can be defined with lower and upper limits	<pre>SELECT CustomerName FROM Customers WHERE CreditLimit BETWEEN 10,000 AND 15000; //</pre> selects customer name whose credit limit between 10000 and 15000. it also includes the value 10000 and 15000.
NOT BETWEEN AND	Checks whether the operand value does not falls within a range. A range can be defined with lower and upper limits	<pre>SELECT CustomerName FROM Customers WHERE CreditLimit NOT BETWEEN 10000 AND 15000; //</pre> selects customer name whose duration NOT between 10000 and 15000. it also includes the value 10000 and 15000.

# Comparison Operators

Operator	Description	Example
LIKE/NOT LIKE	The LIKE operator is used for wild card matching. % used for multiple or no character.	<pre>SELECT CustomerName FROM Customers WHERE CustomerName LIKE '%Gift Stores'; //Select Customers whose name ends with 'Gift Stores' Example: SIGNAL GIFT STORES.</pre>
LIKE /NOT LIKE	The LIKE operator is used for wild card matching. _ is used for single character.	<pre>SELECT CustomerName FROM Customers WHERE CustomerName LIKE 'Herkku Gift_'; //Select customers whose name starts with Herkku Gift and ends with one character after it.</pre>
IS NULL/ IS NOT NULL	Tests for nulls. This is the only operator that should be used to test for nulls.	<pre>SELECT CustomerName FROM Customers WHERE CustomerName IS NOT NULL AND Creditlimit &lt;= 10000; // returns all records which has credit limit &lt;= 10000 and customer name is not null</pre>

# Scenario



Hurray!  
The requirement is  
completed. Thanks for  
solving this problem!

# Check Your Understanding



- Now let us answer these questions in order to test our learning.
- Requesting all associates to reflect the following before proceeding.

What is the operator used for checking whether an age falls in the range between 10 and 60?

What is the operator used, to check if a name starts with "An"?

What is the operator used, to check if a column values meets all the values in a list or a sub-query?

How does one check if a column is null?





# Problem Scenario



Hi!  
Can you provide me with a query which will get the customer details of clients who are not just from London but also from UK?

Let's learn about logical operators which will help us meet Tim's requirements.

# Logical Operators

- **Logical operator**

**Logical operators** are used for manipulating the results of one or more conditions. In SQL, all logical operators evaluate to TRUE, FALSE, or NULL (UNKNOWN).

- Types of comparison operators:

Operator	Description
NOT	Returns TRUE if the condition returns FALSE. Returns FALSE if the return values is TRUE.
AND	Used to combine two conditions. Returns TRUE if both condition are met. Returns FALSE if either of it is FALSE.
OR	Returns TRUE if one of the condition returns TRUE. Returns FALSE if both are FALSE.

# Logical Operators

- **Example:** If age > 45 AND salary < 4000.

Here, **And** is the operator used to combine the results of the both the conditions and returns a result.

Operator	Example	Result
NOT	SELECT CustomerName FROM Customers WHERE NOT (Creditlimit IS NULL); // Retrieves the customer names who has a creditlimit assigned.	Atelier graphique ....
AND	SELECT CustomerName FROM Customers WHERE Country = 'UK' and City = 'London'; // Retrieves the customer names who has country UK and their city is London.	Stylish Desk Decors, Co. Double Decker Gift Stores, Ltd
OR	SELECT CustomerName FROM Customers WHERE Country = 'UK' OR City = 'London'; // Retrieves the customer names who has country UK (OR) their location is London.	AV Stores, Co. UK Collectables, Ltd. giftsbymail.co.uk Stylish Desk Decors, Co. Double Decker Gift Stores, Ltd

# Scenario



Hi!

Can you provide a solution to how we can get the unique country, and state, from two tables.

Let's learn about set operators which will help us meet Tim's requirements..

# Set Operators

- **Set operators** combine the results of two queries into a single result.
- The two queries can be a select query from a same table or from different tables.
- The different types of Set Operators are given below.

Operators	Description
UNION	Returns all distinct rows selected by both the queries
UNION ALL	Returns all rows selected by either query, including all duplicates
INTERSECT	Returns all distinct rows selected by both queries
MINUS	Returns all distinct rows selected by the first query but not the second



# Rules of Set Operators

- Some rules of set operators are:

1. Both queries should select the same number of columns.
2. The columns must be of the same data type. However the length and name of the columns may be different.
3. Column names of first query will be column headings of the retrieved records.

```
SELECT Country, State  
FROM Customers  
<Set Operator>  
SELECT Country, State  
FROM Offices
```

The records retrieved will have the columns for the first table.

Country	State
Japan	Tokyo



# Union Operators

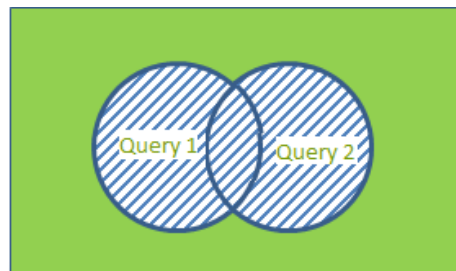
- UNION

The UNION operator combines the output of two query expressions into a single result set. Query expressions are executed independently, and their output is combined into a single result table.

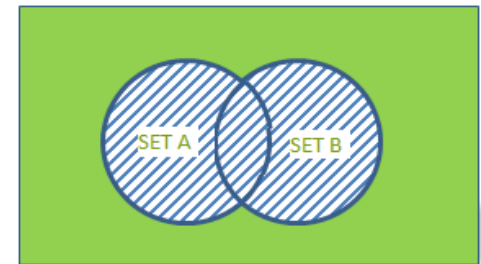
- Syntax

```
{ <query_specification> | ( <query_expression> ) }
UNION <query_specification> | ( <query_expression> )
[ UNION <query_specification> | ( <query_expression> )
[ ...n ] ]
```

```
SELECT coulm1 FROM
table1
UNION
SELECT coulm1 FROM
table2;
```



$A \cup B$



# Example: Union Operator

**Customers**

Country	State
Japan	Tokyo
USA	MA
USA	NY

Duplicate records across the table.

**Offices**

Country	State
Japan	Tokyo
UK	London
USA	NA
UK	London

Duplicate records within the table.

**Output**

Country	State
Japan	Tokyo
USA	MA
USA	NY
UK	London
USA	NA

```
SELECT Country, State
FROM Customers
UNION
SELECT Country, State
FROM Offices;
```

All the unique records from both the tables will be fetched.

# Union All Operators

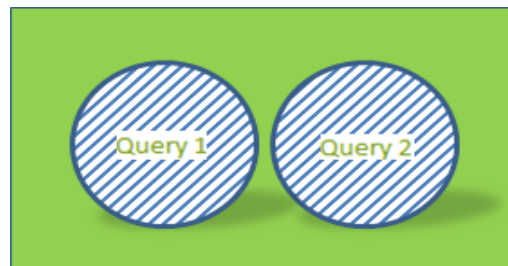
- UNION ALL

If UNION ALL is specified, duplicate rows returned by union expression are retained. If two query expressions return the same row, two copies of the row are returned in the final result. If ALL is not specified, duplicate rows are eliminated from the result set.

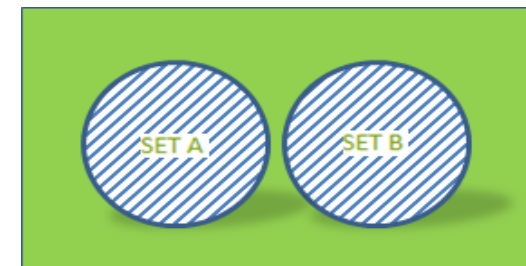
- Syntax

```
{ <query_specification> | ( <query_expression> ) }
UNION ALL <query_specification> | ( <query_expression> )
[ UNION ALL <query_specification> | ( <query_expression> )
[ ...n ] ]
```

```
SELECT column1 FROM
table1
UNION ALL
SELECT column1 FROM
table2;
```



A & B  
where  
 $A \cap B = \emptyset$



# Example: Union All Operator

## Customers

Country	State
Japan	Tokyo
USA	MA
USA	NY

## Offices

Country	State
Japan	Tokyo
USA	NA
UK	London

## Output

Country	State
Japan	Tokyo
USA	MA
USA	NY
UK	London
USA	NA
Japan	Tokyo

```
SELECT Country, State
FROM Customers
UNION ALL
SELECT Country, State
FROM Offices;
```

This also retrieves the duplicate records

# Scenario



I am happy that you  
guys have completed all  
the given requirements!  
Kudos to you!



# Recap of the Case Study

We will use the same CMS case study for learning how to use operators in DQL and DML statements.

- Case Study Scenario
  - This case study will demonstrate how to develop a Course Management System (CMS) for ABC University. The following are the two use cases for which the database needs to be designed.
    - Add Course:
      - This database would add the course details into the course management system.
    - Retrieve Course:
      - This database would retrieve the courses stored in the system and display it. The courses to be added will have the following attributes: Course Code, Course Name, Number of participants, Course Description, Course Duration, Course start date and Course Type.






# Lend a Hand – Prerequisites

- For the next exercise, let us have a look at the prerequisites.
- Prerequisite 1:
  - Associates should ensure that the tables specified in the document are available in the My SQL database, with each table followed by the employee ID.
- Pre-requisite 2:
  - Load the table with necessary data using the DML statements.



Course Mgmt  
System - DDL



# Lend a Hand – Case study

- Develop the queries for the problems stated below.
  - Problem # 1:
    - Calculate the total fees (base fees + Special fees) for the all the courses and display the course code along with the total fees.
  - Problem # 2:
    - Calculate the discount fees for all the courses and display the course code and discount fees.
    - $\text{Discount fees} = \text{discount} * (\text{base fees} + \text{Special fees}) / 100$
    - [Hint: Use the `course_fees` table for this.]



# Solutions

- Solution #1:

```
SELECT COURSE_CODE, BASE_FEES+SPECIAL_FEES  
AS TOTAL_FEES  
FROM COURSE_FEES
```

- Solution #2:

```
SELECT COURSE_CODE, DISCOUNT* (BASE_FEES+SPECIAL_FEES) /100  
AS DISCOUNTFEES  
FROM COURSE_FEES
```



# Lend a Hand

- Develop the queries for the problems stated below.
  - Problem # 3:
    - Display the names of all the courses, the course duration of which is greater than 10 and number of participants is less than 20.
    - [Hint: Use the courses\_info table for this.]
  - Problem # 4:
    - Display the course code whose base fees are greater than 100 or special fees are less than 1000.
    - [Hint: Use the course\_fees table for this.]



# Solutions

- Solution #3:

```
SELECT course_name
FROM course_info
WHERE course_duration >10
and no_of_participants <20
```

- Solution #4:

```
SELECT course_name
FROM course_info
WHERE course_duration >10
and no_of_participants <20
```



# Lend a Hand

- Develop the queries for the problems stated below.
  - Problem # 5:
    - Select all the courses whose base fee  $> 200$ .
    - [Hint: Use the `course_fees` table for this.]
  - Problem # 6:
    - Display the students' ID, first name whose first name is different from their last name.
    - [Hint: Use the `student_info` table for this.]
  - Problem # 7:
    - Select all the courses whose base fee is in the range 100 and 3000.
    - [Hint: Use the `course_fees` table for this.]



# Lend a Hand

— Problem # 8:

- Display the students ID, and first name, whose first name starts with 'A'
- [Hint: Use the student\_info table for this.]

— Problem # 9:

- Display the students ID, first name whose first name has a character 'o'
- [Hint: Use the student\_info table for this.]

— Problem # 10:

- Display the names of all the courses where the course description is Null.
- [Hint: Use the courses\_info table for this.]





# Solutions

- Solution #5:  
**SELECT** **COURSE\_CODE**  
**FROM** **COURSE\_FEES**  
**WHERE** **BASE\_FEES>200**
- Solution #6:  
**SELECT** **STUDENT\_ID,FIRST\_NAME**  
**FROM** **STUDENT\_INFO**  
**WHERE** **FIRST\_NAME!=LAST\_NAME**
- Solution #7:  
**SELECT** **COURSE\_CODE**  
**FROM** **COURSE\_FEES**  
**WHERE** **BASE\_FEES**  
**BETWEEN** **100**  
**AND** **3000**



# Solutions

Solution #8:

```
SELECT STUDENT_ID, FIRST_NAME  
FROM STUDENT_INFO  
WHERE FIRST_NAME  
LIKE 'A%'
```

Solution #9:

```
SELECT STUDENT_ID, FIRST_NAME  
FROM STUDENT_INFO  
WHERE FIRST_NAME  
LIKE '%O%'
```

Solution #10:

```
SELECT course_name  
FROM COURSE_INFO  
WHERE COURSE_DESCRIPTION  
IS NULL
```

# Lend a Hand

- The prerequisite for the given activity is to create the following tables.

Column Name	Data Type
Course_Code	Varchar2
Base_fees	Number
Special_fees	Number
Created_By	Varchar2
Updated_By	Varchar2

COURSE_CODE	BASE_FEES	SPECIAL_FEES	DISCOUNT
1	180	100	10
2	150	110	10
3	160	170	5
4	150	100	10
6	190	100	40

*COURSE\_FEES*

COURSE_CODE	BASE_FEES	SPECIAL_FEES	CREATED_BY	Updated_By
1	120	123	Ram	Ramesh
2	150	110	Bala	Ram
3	160	170	Bala	Vinu
4	170	235	Ram	Ram
6	190	100	Vinod	Vinod

*COURSE\_FEES\_HISTORY*



# Lend a Hand

## – Problem:

- Display all the unique courses between course fees and course fees\_history.
- Use the following columns to check for uniqueness of Course\_Code, BASE\_FEES and SPECIAL\_FEES of the courses in both the COURSE\_FEES and COURSE\_FEES\_HISTORY.

## – Sample Output:

COURSE_FEES	BASE_FEES	SPECIAL_FEES
1	120	123
1	180	100
2	150	110
3	160	170
4	150	100
4	170	235
6	190	100

# Check Your Understanding



What is the operator used for retrieving the common records between two tables?

How can one retrieve all the unique records from both the tables?

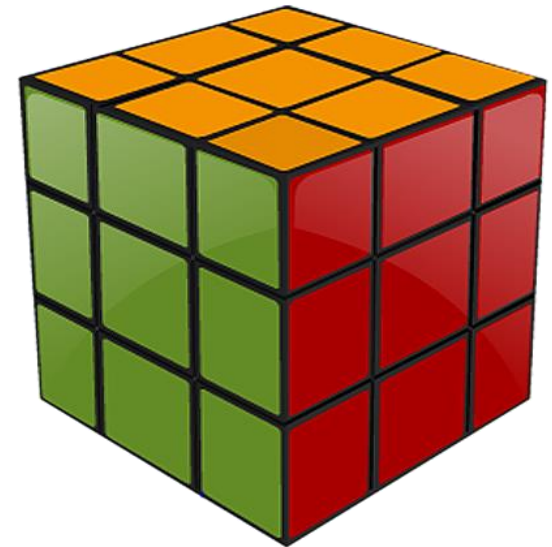
How can one retrieve all the records including the duplicate values from both the tables?



# Summary



- The key points covered in this session are:
  - An SQL Operator is used for processing data values (stored in columns of tables) after which it returns a result. The data values are called operands.
  - Arithmetic operators are used to manipulate numeric operands, which are columns storing numeric values.
  - Comparison operators are used in conditions that compare one operand with another. The result of a comparison can be TRUE (or) FALSE (or) NULL.
  - Logical operators are used for manipulating the results of one or more conditions. In SQL, all logical operators evaluate to TRUE, FALSE, or NULL (UNKNOWN).
  - Set operators combine the results of two queries into a single result.





# Source

- <http://en.wikipedia.org/wiki/SQL>

**Disclaimer:** Parts of the content of this course is based on the materials available from the Web sites and books listed above. The materials that can be accessed from linked sites are not maintained by Cognizant Academy and we are not responsible for the contents thereof. All trademarks, service marks, and trade names in this course are the marks of the respective owner(s).





# Change Log

Version Number	Changes made			
V1.0	Initial Version			
V1.1	Slide No.	Changed By	Effective Date	Changes Effected
	1-48	Learning Content Team CI Team CATP Technical Team	17-05-2013	Base-lining content

## ANSI SQL

You have successfully completed -  
SQL Operators

