Java Annotations and Reflections

@Override

@Test

@Before

@After

Annotations tells the compiler what to do.

Annotations is like comment or explanation or description

Comments are ignored by the compiler. Whereas Annotations can be present in the compiled code.

Annotations can be various types: They are decided by their retention policy.

COMPILER

RUNTIME

SOURCE

These annotations serve the purpose of metadata.

Metadata is like bio data. It is self descriptive.

Reading metadata is done by reflection.

A java class has lot of properties and methods

If another class can read the annotations of these properties and methods and then it is called as reading metadata. This is one of the advantages of Reflection.

Demo:

A java program to list down the methods present in a class.

**package** com.ust.demo;

**public** **class** Employee {

**private** Integer id;

**private** String firstName;

**private** String lastName;

**private** Double salary;

**public** Employee() {}

**public** Employee(Integer id, String firstName, String lastName, Double salary) {

**super**();

**this**.id = id;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.salary = salary;

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** Double getSalary() {

**return** salary;

}

**public** **void** setSalary(Double salary) {

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", salary=" + salary

+ "]";

}

**public** **void** attend()

{

System.***out***.println("Employee attends");

}

}

**package** com.ust.demo;

**import** java.lang.reflect.Method;

**public** **class** App {

**public** **static** **void** main(String[] args) {

Employee raja=**new** Employee();

Class<? **extends** Employee> c = raja.getClass();

Method[] methods = c.getDeclaredMethods(); //c.getMethods();

**for**(Method method:methods)

{

System.***out***.println(method.getName());

}

}

}

Activity:

Identify the different ways the methods of a java class can be obtained using java reflection.

Task:

Write a java program to display only the public methods of a class.

**package** com.ust.demo;

**import** java.lang.reflect.Method;

**import** java.lang.reflect.Modifier;

**public** **class** App {

**public** **static** **void** main(String[] args) {

Employee raja=**new** Employee();

Class c = raja.getClass();

Method[] methods = c.getDeclaredMethods(); //c.getMethods();

**for**(Method method:methods)

{

**if**(!Modifier.*isPublic*(method.getModifiers()))

System.***out***.println(method.getName());

}

}

}

Annotations:

We know that there are lot of annotations. But how can we create an annotation?

When you create an annotation, you need to decide:

1. What is the target of annotation? Will this annotation be used on a class? Or method? Or field? Or constructor? Or parameter
2. What is the retention policy? Will it be available in Source? Or Compilation? Or Runtime?

Syntax to create an annotation:

public @interface MyAnnotation

{

}

@MyAnnotation

**package** com.ust.demo;

**import** **static** java.lang.annotation.RetentionPolicy.***RUNTIME***;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.Target;

@Retention(***RUNTIME***)

//@Target({ElementType.TYPE,ElementType.FIELD, ElementType.CONSTRUCTOR})

**public** **@interface** Jag {

}

Task:

Create an annotation and mention the target.

Use the annotation.

Next:

Using reflection, can we read the annotations?

SELECT statement:

PROJECTION

Using specific columns

SELECTION

Using WHERE condition

SORTING

ORDER BY ASC | DESC

Sub Queries:

Allow us to use the result of a query to be used by another query.

Task:

Write a SELECT statement on HR.EMPLOYEES to display the

EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, JOB\_ID

Of all employees whose JOB\_ID is same as employee whose id is 119

Ans:

SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, JOB\_ID

FROM HR.EMPLOYEES

WHERE JOB\_ID=(SELECT JOB\_ID FROM HR.EMPLOYEES WHERE EMPLOYEE\_ID=119)

Rules for subquery:

1. When we use sub query in a condition that use operators like

=

!=

>

<

>=

<=

Then the sub query should return only 1 row.

In case if you use sub query that returns more than 1 row, then use modifier like

ANY, ALL

1. It is good to use subquery always in the right side of the comparison or any operator.
2. Sub query need not use ORDER BY except when we use top N analysis
3. Sub query columns list, usually 1 column is used in sub query.

Sub query also sometimes can use SELECT \*

Sub query also sometimes can use MULTIPLE COLUMNS.

Sub query also sometimes can use SELECT \*

How????

When the condition is

SELECT \* FROM ORDERS O

WHERE EXISTS (SELECT \* FROM PRODUCTS P WHERE P.ID=O.ID)

Note:

Only when I use EXISTS, I am able to use SELECT \* in a sub query.

Reminder:

Lets discuss during correlated sub query.

Sub query also sometimes can use MULTIPLE COLUMNS.

The outer query condition and sub query columns must be pair wise comparison.

SELECT \*

FROM employees

WHERE (manager\_id, department\_id) IN (SELECT manager\_id,

  department\_id

  FROM employees...

FUNCTIONS:

There are lot of pre-defined functions in Oracle.

Categories of functions:

In one way, we can categorize them into

Scalar functions (single row function)

In another way,

Aggregate functions (that produce 1 result per relation)

Also, we can categorize the functions as:

Char

Numeric

Date

Conversion

Example

UPPER

-- SELECT FIRST\_NAME, UPPER(FIRST\_NAME) FROM HR.EMPLOYEES; --Example for scalar/single row function

-- SELECT SUM(SALARY) FROM HR.EMPLOYEES; --aggregate function

Activity:

Find what are all functions available in

CHARACTER

NUMERIC

DATE

CONVERSION

-- SELECT FIRST\_NAME, UPPER(FIRST\_NAME) FROM HR.EMPLOYEES;

-- SELECT SUM(SALARY) FROM HR.EMPLOYEES;

-- SELECT UPPER('jag') FROM DUAL; --JAG

-- SELECT LOWER('JAG INDIA') FROM DUAL; --jag india

-- SELECT INITCAP('jag india') FROM DUAL; --Jag India

UPPER

LOWER

INITCAP

SUBSTR to retrieve a portion of a string

INSTR SELECT \* FROM HR.EMPLOYEES WHERE INSTR(JOB\_ID,'CLERK')!=0;

TRIM select \* from hr.employees where trim(JOB\_ID)=trim('ST\_CLERK ');

LPAD we specify the size of the string. If the string is less than that length, then those remaining length is filled with specified character.

RPAD SELECT RPAD(FIRST\_NAME,15,'#') FROM HR.EMPLOYEES;

REPLACE SELECT REPLACE('RAM','A','O') FROM DUAL;

NUMERIC functions:

ABS ABS(-20) = 20

MAX is aggregate function like SUM, MIN, AVG, COUNT

CEIL

FLOOR

ROUND

POW

PI

LOG

SQRT

ROUND vs CEILING vs FLOOR

ROUND(12.4545,1) - 12.5

ROUND(12.4545,2) = 12.45

ROUND(12.4545,3) = 12.455

CEILING CEIL(5.1

FLOOR

True/False

FLOOR returns the highest integer lesser than the given number Ans: true

CEIL returns the smallest integer bigger than the given number Ans: true

TRUNC(12.4545,2) = 12.45

DATE FUNCTIONS:-

ADD\_MONTHS SELECT ADD\_MONTHS(SYSDATE,3) FROM DUAL;

NEXT\_DAY SELECT NEXT\_DAY(SYSDATE,'FRIDAY') FROM DUAL;

LAST\_DAY SELECT LAST\_DAY(SYSDATE) FROM DUAL;

SYSDATE

MONTHS\_BETWEEN SELECT MONTHS\_BETWEEN(SYSDATE,'01-Jan-23') FROM DUAL;

ROUND SELECT round(sysdate,'YEAR') FROM DUAL;

CONVERSION FUNCTIONS

Convert from CHAR to NUMBER and vice versa

Convert from CHAR TO DATE and vice versa

TO\_CHAR

TO\_DATE

TO\_NUMBER

select TO\_CHAR(sysdate,'dd/MM/yyyy') from dual;

TO\_DATE converts characters to date when we specify the format

|  |
| --- |
| **MONTHS\_BETWEEN(TO\_DATE('31/12/22','DD/MM/YY'),TO\_DATE('30/06/23','DD/MM/YY'))** |
| -6 |

TO\_NUMBER

SELECT TO\_NUMBER('20') FROM DUAL;

GROUP BY

--------------

Example:

Display the sum of salary

SELECT SUM(SALARY) FROM HR.EMPLOYEES;

This query returns the sum of all the salary present in the table

SELECT DEPARTMENT\_ID, SUM(SALARY) FROM HR.EMPLOYEES

GROUP BY DEPARTMENT\_ID;

Observation:

Whenever we use an aggregate function, we get one result per relation.

But when we use group by, then we get one result per group.

You do not want all groups to be retrieved. Only groups which are significant must be retrieved.

What is significant group? For example, wherever departments have sum of salary > 20000 only needs to be displayed.

How to do that?

SELECT DEPARTMENT\_ID, SUM(SALARY)

FROM HR.EMPLOYEES

GROUP BY DEPARTMENT\_ID

WHERE SUM(SALARY)>20000;

Error: WHERE clause does not allow us to use AGGREGATE functions.

So, instead of WHERE, we use HAVING

SELECT DEPARTMENT\_ID, SUM(SALARY)

FROM HR.EMPLOYEES

GROUP BY DEPARTMENT\_ID

HAVING SUM(SALARY)>20000;

What is the difference between WHERE and HAVING?

WHERE is used to restrict the rows.

HAVING is used to restrict the groups.

JOIN

------

What is the need for JOIN?

When we need to retrieve columns from more than one table, we use JOIN.

Demo:

Display the Employee id, first name, last name and their department name.

JOINs are implemented using 2 different syntax:

1. JOIN we use JOIN between table names
2. Theta we use , between table names

LEFT OUTER JOIN

SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, DEPARTMENT\_NAME

FROM HR.EMPLOYEES, HR.DEPARTMENTS

WHERE HR.EMPLOYEES.DEPARTMENT\_ID=HR.DEPARTMENTS.DEPARTMENT\_ID(+)

ORDER BY EMPLOYEE\_ID;

We can also use table name alias to simplify the code:

SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, DEPARTMENT\_NAME

FROM HR.EMPLOYEES E, HR.DEPARTMENTS D

WHERE E.DEPARTMENT\_ID=D.DEPARTMENT\_ID(+)

ORDER BY E.EMPLOYEE\_ID;

In the above query, we have sued LEFT OUTER JOIN, so all records from left table (EMPLOYEES) will be retrieved and if there is no matching records in right side table, then NULL is retrieved.

JOIN syntax

---------------

SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, DEPARTMENT\_NAME

FROM HR.EMPLOYEES E JOIN HR.DEPARTMENTS D

ON E.DEPARTMENT\_ID=D.DEPARTMENT\_ID;

BANK Schema SCRIPT For postgresql:

CREATE TABLE customer

(

custid VARCHAR(6),

fname VARCHAR(30),

mname VARCHAR(30),

ltname VARCHAR(30),

city VARCHAR(15),

mobileno VARCHAR(10),

occupation VARCHAR(10),

dob DATE,

PRIMARY KEY(custid)

);

CREATE TABLE branch

(

bid VARCHAR(6),

bname VARCHAR(30),

bcity VARCHAR(30),

CONSTRAINT branch\_bid\_pk1 PRIMARY KEY(bid)

);

CREATE TABLE account

(

acnumber VARCHAR(6),

custid VARCHAR(6),

bid VARCHAR(6),

opening\_balance INT,

aod DATE,

atype VARCHAR(10),

astatus VARCHAR(10),

CONSTRAINT account\_acnumber\_pk1 PRIMARY KEY(acnumber),

CONSTRAINT account\_custid\_fk1 FOREIGN KEY(custid) REFERENCES customer(custid),

CONSTRAINT account\_bid\_fk1 FOREIGN KEY(bid) REFERENCES branch(bid)

);

CREATE TABLE trandetails

(

tnumber VARCHAR(6),

acnumber VARCHAR(6),

dot DATE,

medium\_of\_transaction VARCHAR(20),

transaction\_type VARCHAR(20),

transaction\_amount INT,

CONSTRAINT trandetails\_tnumber\_pk1 PRIMARY KEY(tnumber),

CONSTRAINT trandetails\_acnumber\_fk1 FOREIGN KEY(acnumber) REFERENCES account(acnumber)

);

CREATE TABLE loan

(

custid VARCHAR(6),

bid VARCHAR(6),

loan\_amount INT,

CONSTRAINT loan\_customer\_custid\_bid\_pk1 PRIMARY KEY(custid,bid),

CONSTRAINT loan\_custid\_fk1 FOREIGN KEY(custid) REFERENCES customer(custid),

CONSTRAINT loan\_bid\_fk1 FOREIGN KEY(bid) REFERENCES branch(bid)

);

INSERT INTO customer VALUES('C00001','Ramesh','Chandra','Sharma','Delhi','9543198345','Service','1976-12-06');

INSERT INTO customer VALUES('C00002','Avinash','Sunder','Minha','Delhi','9876532109','Service','1974-10-16');

INSERT INTO customer VALUES('C00003','Rahul',null,'Rastogi','Delhi','9765178901','Student','1981-09-26');

INSERT INTO customer VALUES('C00004','Parul',null,'Gandhi','Delhi','9876532109','Housewife','1976-11-03');

INSERT INTO customer VALUES('C00005','Naveen','Chandra','Aedekar','Mumbai','8976523190','Service','1976-09-19');

INSERT INTO customer VALUES('C00006','Chitresh',null,'Barwe','Mumbai','7651298321','Student','1992-11-06');

INSERT INTO customer VALUES('C00007','Amit','Kumar','Borkar','Mumbai','9875189761','Student','1981-09-06');

INSERT INTO customer VALUES('C00008','Nisha',null,'Damle','Mumbai','7954198761','Service','1975-12-03');

INSERT INTO customer VALUES('C00009','Abhishek',null,'Dutta','Kolkata','9856198761','Service','1973-05-22');

INSERT INTO customer VALUES('C00010','Shankar',null,'Nair','Chennai','8765489076','Service','1976-07-12');

INSERT INTO branch VALUES('B00001','Asaf ali road','Delhi');

INSERT INTO branch VALUES('B00002','New delhi main branch','Delhi');

INSERT INTO branch VALUES('B00003','Delhi cantt','Delhi');

INSERT INTO branch VALUES('B00004','Jasola','Delhi');

INSERT INTO branch VALUES('B00005','Mahim','Mumbai');

INSERT INTO branch VALUES('B00006','Vile parle','Mumbai');

INSERT INTO branch VALUES('B00007','Mandvi','Mumbai');

INSERT INTO branch VALUES('B00008','Jadavpur','Kolkata');

INSERT INTO branch VALUES('B00009','Kodambakkam','Chennai');

INSERT INTO account VALUES('A00001','C00001','B00001',1000,'2012-12-15','Saving','Active');

INSERT INTO account VALUES('A00002','C00002','B00001',1000,'2012-06-12','Saving','Active');

INSERT INTO account VALUES('A00003','C00003','B00002',1000,'2012-05-17','Saving','Active');

INSERT INTO account VALUES('A00004','C00002','B00005',1000,'2013-01-27','Saving','Active');

INSERT INTO account VALUES('A00005','C00006','B00006',1000,'2012-12-17','Saving','Active');

INSERT INTO account VALUES('A00006','C00007','B00007',1000,'2010-08-12','Saving','Suspended');

INSERT INTO account VALUES('A00007','C00007','B00001',1000,'2012-10-02','Saving','Active');

INSERT INTO account VALUES('A00008','C00001','B00003',1000,'2009-11-09','Saving','Terminated');

INSERT INTO account VALUES('A00009','C00003','B00007',1000,'2008-11-30','Saving','Terminated');

INSERT INTO account VALUES('A00010','C00004','B00002',1000,'2013-03-01','Saving','Active');

INSERT INTO trandetails VALUES('T00001','A00001','2013-01-01','Cheque','Deposit',2000);

INSERT INTO trandetails VALUES('T00002','A00001','2013-02-01','Cash','Withdrawal',1000);

INSERT INTO trandetails VALUES('T00003','A00002','2013-01-01','Cash','Deposit',2000);

INSERT INTO trandetails VALUES('T00004','A00002','2013-02-01','Cash','Deposit',3000);

INSERT INTO trandetails VALUES('T00005','A00007','2013-01-11','Cash','Deposit',7000);

INSERT INTO trandetails VALUES('T00006','A00007','2013-01-13','Cash','Deposit',9000);

INSERT INTO trandetails VALUES('T00007','A00001','2013-03-13','Cash','Deposit',4000);

INSERT INTO trandetails VALUES('T00008','A00001','2013-03-14','Cheque','Deposit',3000);

INSERT INTO trandetails VALUES('T00009','A00001','2013-03-21','Cash','Withdrawal',9000);

INSERT INTO trandetails VALUES('T00010','A00001','2013-03-22','Cash','Withdrawal',2000);

INSERT INTO trandetails VALUES('T00011','A00002','2013-03-25','Cash','Withdrawal',7000);

INSERT INTO trandetails VALUES('T00012','A00007','2013-03-26','Cash','Withdrawal',2000);

INSERT INTO loan VALUES('C00001','B00001',100000);

INSERT INTO loan VALUES('C00002','B00002',200000);

INSERT INTO loan VALUES('C00009','B00008',400000);

INSERT INTO loan VALUES('C00010','B00009',500000);

INSERT INTO loan VALUES('C00001','B00003',600000);

INSERT INTO loan VALUES('C00002','B00001',600000);

Problem #1:

Write a query to display the customer number , firstname, customer’s date of birth . Display in a sorted order of date of birth year and within that sort by firstname.

+--------+----------+------------+

| custid | fname | dob |

+--------+----------+------------+

| C00009 | Abhishek | 1973-05-22 |

| C00002 | Avinash | 1974-10-16 |

| C00008 | Nisha | 1975-12-03 |

| C00005 | Naveen | 1976-09-19 |

| C00004 | Parul | 1976-11-03 |

| C00001 | Ramesh | 1976-12-06 |

| C00010 | Shankar | 1976-07-12 |

| C00007 | Amit | 1981-09-06 |

| C00003 | Rahul | 1981-09-26 |

| C00006 | Chitresh | 1992-11-06 |

+--------+----------+------------+

10 rows in set (0.01 sec)

select custid, fname, dob from customer order by extract(year from dob),fname;

-----------------------------------------------------------------

Problem #2:

Write a query to display customer’s number, first name and middle name. The customer’s who don’t have middle name, for them display the last name. Give the alias name as Cust\_Name.

+----------------+

| CUST\_NAME |

+----------------+

| Ramesh Chandra |

| Avinash Sunder |

| Rahul Rastogi |

| Parul Gandhi |

| Naveen Chandra |

| Chitresh Barwe |

| Amit Kumar |

| Nisha Damle |

| Abhishek Dutta |

| Shankar Nair |

+----------------+

10 rows in set (0.01 sec)

select custid, concat(fname, coalesce(mname,ltname)) as cust\_name from customer;

-----------------------------------------------------------------Problem#3:

Write a query to display account number, customer’s number, customer’s firstname,lastname,account opening date.

+----------+--------+----------+---------+------------+

| acnumber | custid | fname | ltname | aod |

+----------+--------+----------+---------+------------+

| A00001 | C00001 | Ramesh | Sharma | 2012-12-15 |

| A00002 | C00002 | Avinash | Minha | 2012-06-12 |

| A00003 | C00003 | Rahul | Rastogi | 2012-05-17 |

| A00004 | C00002 | Avinash | Minha | 2013-01-27 |

| A00005 | C00006 | Chitresh | Barwe | 2012-12-17 |

| A00006 | C00007 | Amit | Borkar | 2010-08-12 |

| A00007 | C00007 | Amit | Borkar | 2012-10-02 |

| A00008 | C00001 | Ramesh | Sharma | 2009-11-09 |

| A00009 | C00003 | Rahul | Rastogi | 2008-11-30 |

| A00010 | C00004 | Parul | Gandhi | 2013-03-01 |

+----------+--------+----------+---------+------------+

10 rows in set (0.01 sec)

select a.acnumber, c.custid, c.fname, c.ltname, a.aod from account a join customer c on a.custid=c.custid;

-----------------------------------------------------------------

Problem # 4:

Write a query to display the number of customer’s from Delhi. Give the count an alias name of Cust\_Count.

+------------+

| CUST\_COUNT |

+------------+

| 4 |

+------------+

1 row in set (0.00 sec)

select count(\*) as Cust\_Count from customer where trim(city)='Delhi';

-----------------------------------------------------------------Problem # 5:

Write a query to display the customer number, customer firstname,account number for the customer’s whose accounts were created after 15th of any month.

+--------+----------+----------+

| custid | fname | acnumber |

+--------+----------+----------+

| C00002 | Avinash | A00004 |

| C00003 | Rahul | A00003 |

| C00003 | Rahul | A00009 |

| C00006 | Chitresh | A00005 |

+--------+----------+----------+

4 rows in set (0.00 sec)

select c.custid, c.fname, a.acnumber from customer c join account a on c.custid=a.custid WHERE extract(day from aod)>15;

-----------------------------------------------------------------Problem # 6:

Write a query to display the customers firstname, city and account number who are into business, service or studies.

+----------+--------+----------+

| fname | city | acnumber |

+----------+--------+----------+

| Ramesh | Delhi | A00001 |

| Ramesh | Delhi | A00008 |

| Avinash | Delhi | A00002 |

| Avinash | Delhi | A00004 |

| Rahul | Delhi | A00003 |

| Rahul | Delhi | A00009 |

| Chitresh | Mumbai | A00005 |

| Amit | Mumbai | A00006 |

| Amit | Mumbai | A00007 |

+----------+--------+----------+

9 rows in set (0.00 sec)

select c.fname, c.city, a.acnumber from customer c join account a on c.custid=a.custid where occupation in ('Business','Service','Student');

-----------------------------------------------------------------Problem # 7:

Write a query to display city name and count of branches in that city. Give the count of branches an alias name of Count\_Branch.

+---------+--------------+

| bcity | COUNT\_BRANCH |

+---------+--------------+

| Chennai | 1 |

| Delhi | 4 |

| Kolkata | 1 |

| Mumbai | 3 |

+---------+--------------+

4 rows in set (0.00 sec)

select bcity, count(bid) as Count\_Branch from branch group by bcity;

-----------------------------------------------------------------

Problem # 8:

Write a query to display account id, customer’s firstname, customer’s lastname for the customer’s whose account is Active.

+----------+----------+---------+

| acnumber | fname | ltname |

+----------+----------+---------+

| A00001 | Ramesh | Sharma |

| A00002 | Avinash | Minha |

| A00004 | Avinash | Minha |

| A00003 | Rahul | Rastogi |

| A00010 | Parul | Gandhi |

| A00005 | Chitresh | Barwe |

| A00007 | Amit | Borkar |

+----------+----------+---------+

7 rows in set (0.00 sec)

select a.acnumber, c.fname, c.ltname from customer c join account a on c.custid=a.custid where astatus='Active';

-----------------------------------------------------------------Problem # 9:

Write a query to display the customer’s number, customer’s firstname, branch id and loan amount for people who have taken loans.

+--------+----------+--------+-------------+

| custid | fname | bid | loan\_amount |

+--------+----------+--------+-------------+

| C00001 | Ramesh | B00001 | 100000 |

| C00001 | Ramesh | B00003 | 600000 |

| C00002 | Avinash | B00001 | 600000 |

| C00002 | Avinash | B00002 | 200000 |

| C00009 | Abhishek | B00008 | 400000 |

| C00010 | Shankar | B00009 | 500000 |

+--------+----------+--------+-------------+

6 rows in set (0.00 sec)

select c.custid, c.fname, l.bid, l.loan\_amount from customer c join loan l on c.custid=l.custid;

-----------------------------------------------------------------Problem # 10:

Write a query to display customer number, customer name, account number where the account status is terminated.

+--------+--------+----------+

| custid | fname | acnumber |

+--------+--------+----------+

| C00001 | Ramesh | A00008 |

| C00003 | Rahul | A00009 |

+--------+--------+----------+

2 rows in set (0.00 sec)

select c.custid, c.fname, a.acnumber from customer c join account a on c.custid=a.custid where a.astatus='Terminated';

Coalesce

Function accepts any number of parameters

Coalesce(a,b,c,d,e,f….)

If a is null, then it checks b. If b is null, then it check c, and so on.

The first not null found, will be returned. If all are null, then it returns null.

If you see the queries we have done today, mostly there was a common column between 2 tables. We used that as a condition.

It is not always, you will get such common columns. Column names may be same or different.

In all these cases, column names are same among tables or different also, common values are found.

These kind of joins are also called as EQUI JOIN.

You join condition is either = or <> !=

What is NON EQUI JOIN?

There is no common column.

The value in table1 is not equal to or not equal to a value in table2.

select E.EMPLOYEE\_ID, E.FIRST\_NAME, E.SALARY, J.JOB\_TITLE

FROM HR.EMPLOYEES E

JOIN HR.JOBS J

ON E.SALARY BETWEEN J.MIN\_SALARY AND J.MAX\_SALARY;

CROSS JOIN

-------------------

If you skip the JOIN condition, then you will get how many rows??????

If Table1 has 5 rows, Table2 has 10 rows, then you will get 5 \* 10 = 50 rows

This is called as Cartesian product.

A Cross join produces cartesian product.

SELECT \* FROM HR.DEPARTMENTS CROSS JOIN HR.JOBS;

SELF JOIN

--------------

SELECT E.EMPLOYEE\_ID, E.FIRST\_NAME, E.LAST\_NAME, E.MANAGER\_ID, M.FIRST\_NAME AS "MANAGER NAME"

FROM HR.EMPLOYEES E

LEFT JOIN HR.EMPLOYEES M

ON E.MANAGER\_ID=M.EMPLOYEE\_ID

ORDER BY E.EMPLOYEE\_ID;