**SQL** - 'Structured Query Language

A Database can be broken down as follows :- **Table ,- Columns (Fields),- Rows (Records)**

**Primary Key** - 'a special column (or combination of columns) in a relational database that uniquely identifies the records for that table. It must contain a unique value for each row of data. It can not be Null.

**Foreign Key** - 'A primary key in another table in Relational Database System. It can be Null. Null -(Undefined)

**Unique Key** - 'a unique column in a table that only has unique values but it is not the primary key of the table

**Constraints** - 'Rules that each column in a Database has to follow (Primary Key, Foreign Key, Unique Key).

=======================================================

**Select Statement** - Retrieve Existing data from the database

**AS** - Renaming (ALIAS), Blank works as ALIAS as well.

**Order By** - Sorts the output by Asc or Desc order

**Distinct** - only displays Unique values will remove duplicates

**AND** - Checks both sides of conditions, only applies the filter when both conditions are true

**OR** - Checks both sides of conditions, applies filter when one of the two is true

**IN** - allows you apply multiple 'OR' conditions to a set of values that belong to the same column

**LIKE** - allows partial searching within the database LIKE '%S'

**RowNum** -> Restricts database output based on row numbers to be displayed

**Between** Operator: Will filter the data output based 2 arguments the minimum and the maximum value

**Dual table** - A special table with 1 column and one row. It is by default installed

You can check:-Sysdate,-Calculations

=============================================================================================

**'Functions in SQL**: They are a powerful feature of SQL

- Perform Data calculations ,- Modifying individual Data ,- Manipulate the output of multiple rows

- Formatting certain columns such as Date Data Type

SQL Function will take an argument as value and return an output

Types of Functions in SQL:

Single Row Function - Return one result per row

Multiple Row Function - Return one result per set of rows

**Single Row Function** - **Input 1 row and output 1 row**

**Lower** - Lower(stringvalue) --> convert to lower case

**InitCap** - InitCap(stringvalue) --> converts to where the first letter of the string becomes capitalized

**Upper** - Upper (Stringvalue) --> converts to uppercase

**Length** - length(stringvalue) --> returns the amount of characters present in the string

**INSTR** --> instr((stringvalue),'a')🡪 It looks for a character in a string and returns it if there is a position in the string

**Replace** --> Looks for a character in a string, if it finds the character then it will replace the character with what is defined in the replace function. Example: String = "Water" -🡪Replace(String,'t','p') --> Waper

**Number relater:**

**Round ->rounds number**

**TRUNC->does not round but trunCates**

**MOD-> is mod%**

**Multiple Row Functions - Multiple Rows input - Output 1 row**

**Max** - Syntax Max(column\_name) --> the highest value of that column

**Min** - Syntax Min(column\_name) --> the minimum value of that column

**Count** - Count(column\_name) --> the count of rows for that particular query

**avg** - Syntax avg(column\_name) --> Add all the rows in the column divides by the

**Sum** - Syntax sum(column\_name)--> Adds the column and returns 1 value that will contain the sum of all columns

**Group By Statement** - You can divide rows in a table into smaller groups by using the Group by clause.

It organizes the output of an aggregated query by subgroups

you have to group by the column selected. The conflict happens when 1 column has an

aggregated function applied and the other selected column does not

Select job\_id,max(salary) 🡪 from Employees 🡪 group by job\_id;

**HAVING** Clause - Used to filter on Aggregate Functions.

**Where** Clause - Used to filter on Single row functions

JOINS

There are 3 kinds of joins:

-**Inner join** - Returns records that have matching values in both tables

-**Outter join**

1.**Left outer join** - Return all records from the left table, and the matched records from the right table

2.**Right outer join** - Return all records from the right table, and the matched records from the left table

3.**Full outer join** - Return all records when there is a match in either left or right table

-**Self-join** - is nothing but a normal join, however it joins with itself.

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

What defines matching data?

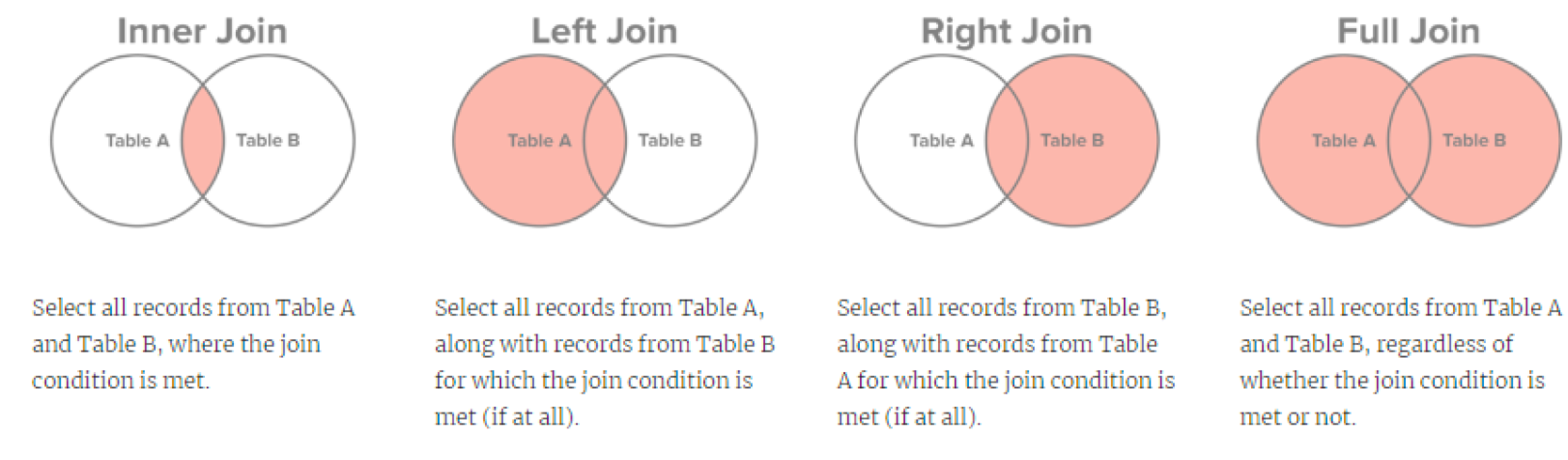
The on condition that specifies the primary key/foreign from the tables that are joining

**Interview Question**: Why do you Use a Table Alias?

We use table aliases while joining tables because of the complexity of the query and the length of certain table names.

**Explaining Joins:**

Let’s say we have two sets of data in our relational database: table A and table B, with some sort of relation specified by primary and foreign keys. The result of joining these tables together can be visually represented by the following diagram:



**Self Join Explanation:**

Let’s say table A has Employee Id, Name, Address, Phone number, Supervisor ID columns. In order to find out the name of Supervisor of an Employee, we need to look first at Employees Supervisor ID, then find it in Employee ID.

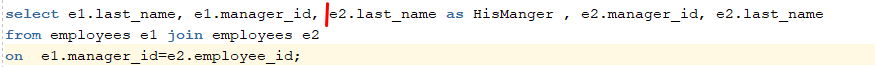
So let’s say we want to write a query that will show Employee ID, Employee Name, Supervisor ID and Supervisor Name:

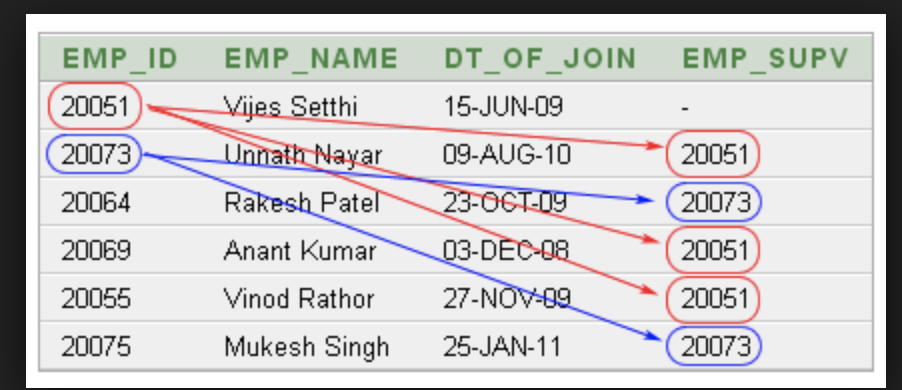
Select table1.Employee ID “employee\_id”,table1.Name “employee\_name”

table2.Employee ID “supervisor\_id”,table2.Name “supervisor\_name”

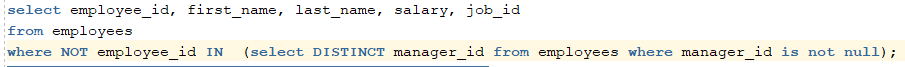
from A table1 join A table2

on table1.Supervisor ID = table2. Employee ID





**Subqueries**: The Result of one query becomes the input for another query



Execution Flow of Subqueries:

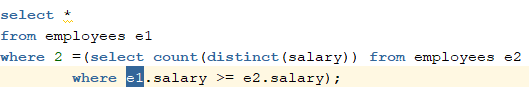
1. The Inner Query will run first

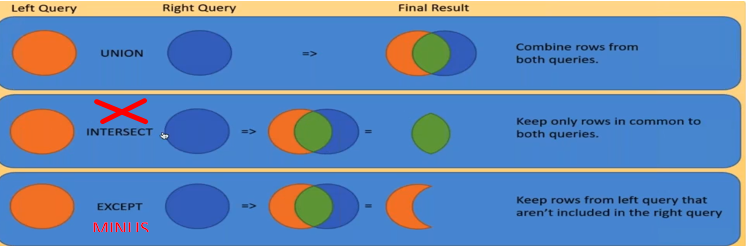
2. The Output of Inner Query will serve as the input for the Outer Query

Select \* From Employees

where salary < (select salary from employees where last\_name = 'Fripp');

**Co-Related Subqueries: (Second lowest salary)**





**SET OPERATORS**: Set operators work with two independent queries to return values based on the set operator

SET OPERATOR TYPES:

- **Union** ,- **Union All** ,- **Minus** ,- **Intersect**

**Rules** for using SET operators:

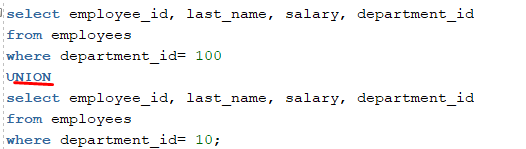
1. You must have the same amount of column selected in both queries

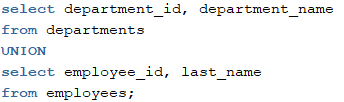
2. The positioning of the column selected must match the Datatype of the matched column in position

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

**Union ALL**: -It will merge,-It will not sort the output ,-it will not remove duplicates

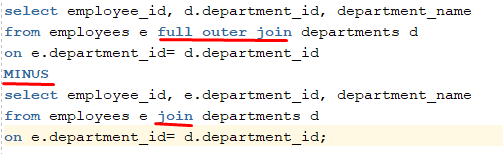
**UNION:** -It will merge ,-It will remove duplicates ,-It will sort by Asccensing order





**MINUS OPERATOR**: Example -🡪 AABCDE - OIABS = CDE

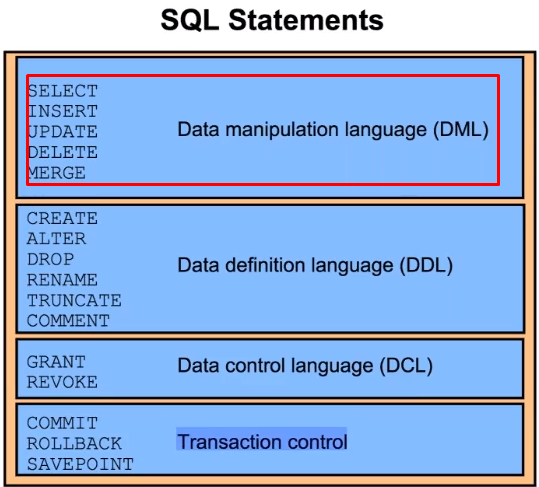
Minus Operator will Run the first independent query and second independent query**, it will return what is contained in the FIRST independent query** output that is not contained in the second independent query output.

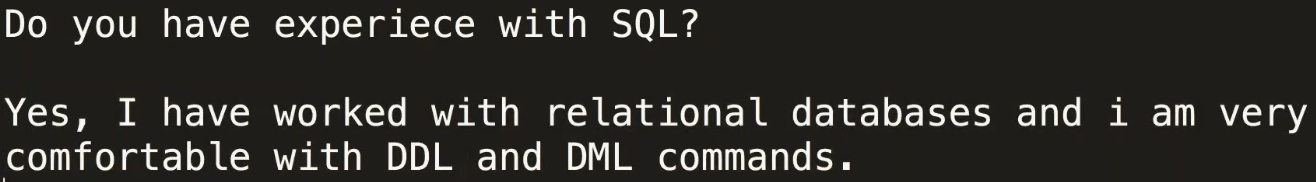


(WILL SHOW ONLY WHAT IS NOT IN SECOND QUERY ) (EMPLYEE WHO Does not have department and ViseVersa)

**INTERSECT:** Example -🡪 abcdfer INTERSECT reabder = abder

The INTERSECT operator will run both independent queries and return records that are **PRESENT** in both query outputs

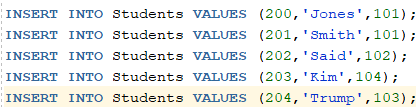




**--> DML --> Data Manipulation language <! Mainly for Testers**

-->**SELECT** -> We can select existing data

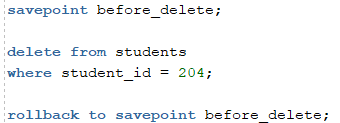
-->**Insert** -> Inserting a new record in the database



-->**Update** -> Where you selectively update the value of an existing record (be careful! If u don’t put **conditions** it will update for **ALL** )

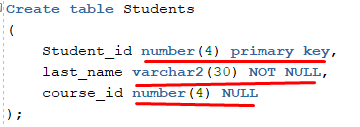


-->**Delete** --> Where you can remove specific records from the Database. Archive table to store deleted records



**--> DDL --> Data Definition Language - < Mostly for Developers**

-->**Create Table**--> Creating a Table



-->**Truncate** --> Will remove the table data only and **will keep empty table**.



-->**Drop** --> Will remove data and the table from the Database



**DML** commands **can** be reversed/RolledBack.**rollback;**

**DDL** commands **cannot** be reversed/RolledBack . **rollback;**

**INSERT INTO**: Allows a user to INSERT a data to a table

INSERT INTO cars VALUES('YV1672MK9D2304784','Honda','Accord',2015,40756);

**UPDATE statement**: Allows a user to update an existing record to a new value

Update Cars set YEAR = 2003 where vin\_number = '1GNGC26RXXJ407648';

**DELETE COMMAND**: Using Delete will allow us to remove 1 record or a set of records matching condition

DELETE From Cars where MAKE = 'Honda';

**Truncating a Table**: Truncating will remove all data from the table but not delete the table

Truncate Table Cars;

**Dropping a Table**: Dropping a Table will remove the table and data from the database

Drop Table Cars;

**ADODB Connection: Connect UFT to Database**

It's done using VBScript based Classes: ADODB.Connection, ADODB.Recordset,

Example 🡪

Set DBConnection = CreateObject("ADODB.Connection")

Set DBRecordSet = CreateObject("ADODB.Recordset")

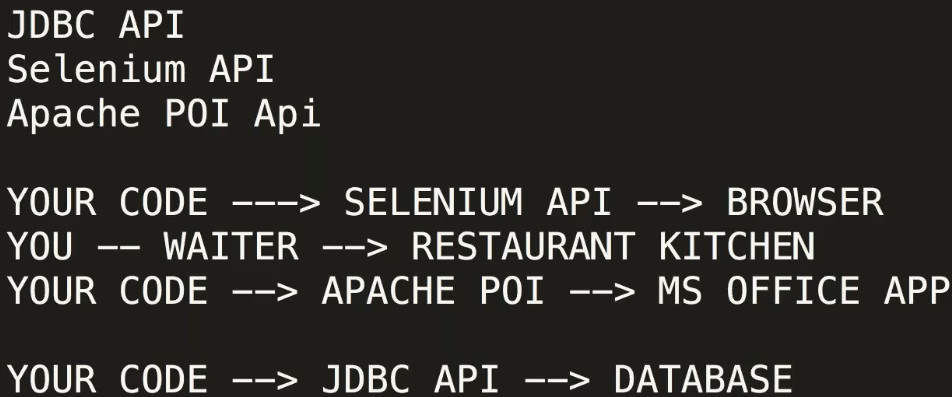
DBConnection.Open "DSN=Oracle;UID=HR;PWD=HR;"

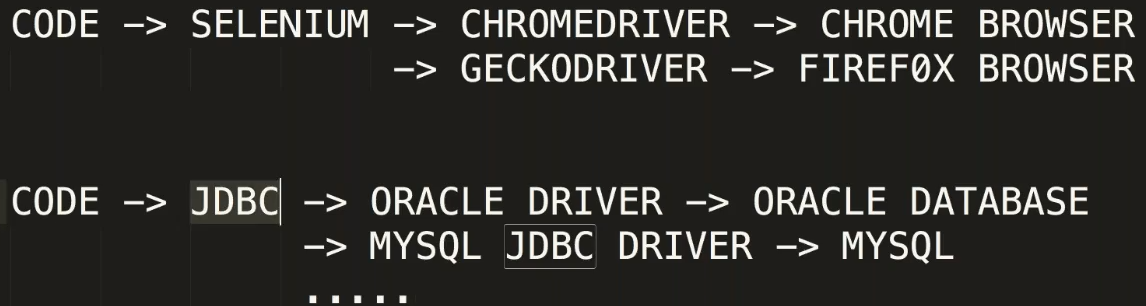
DBRecordSet.Open "Select \* from employees where salary > 1000",DBConnection

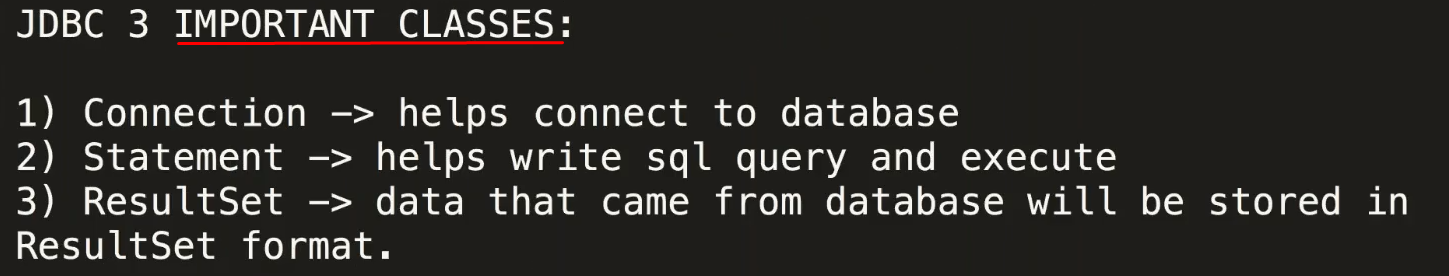
msgbox DBRecordSet.Fields("employee\_id")

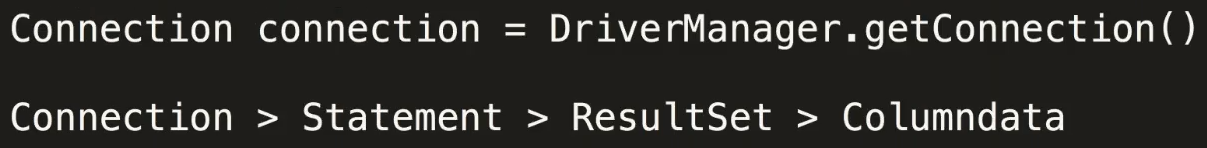
msgbox DBRecordSet.Fields("Salary")

**JDBC can connect to any Data base library.**

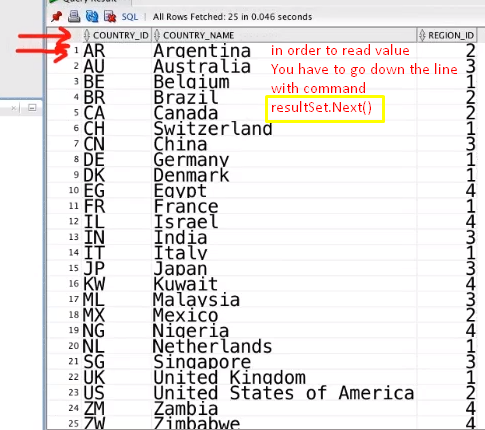


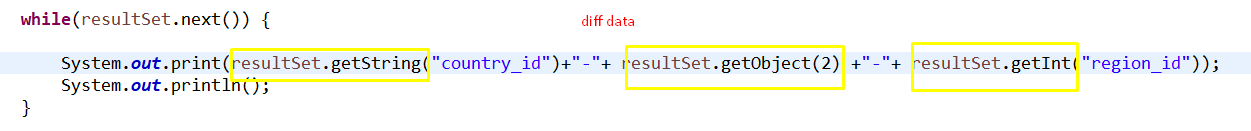


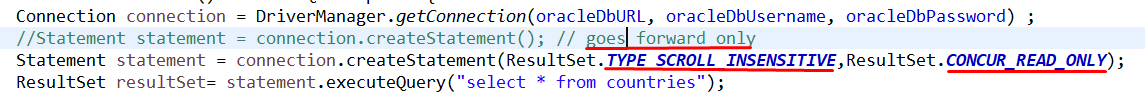


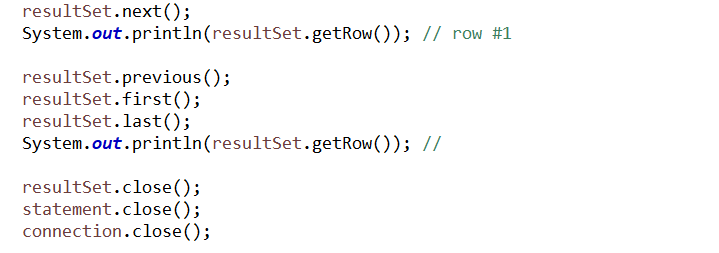


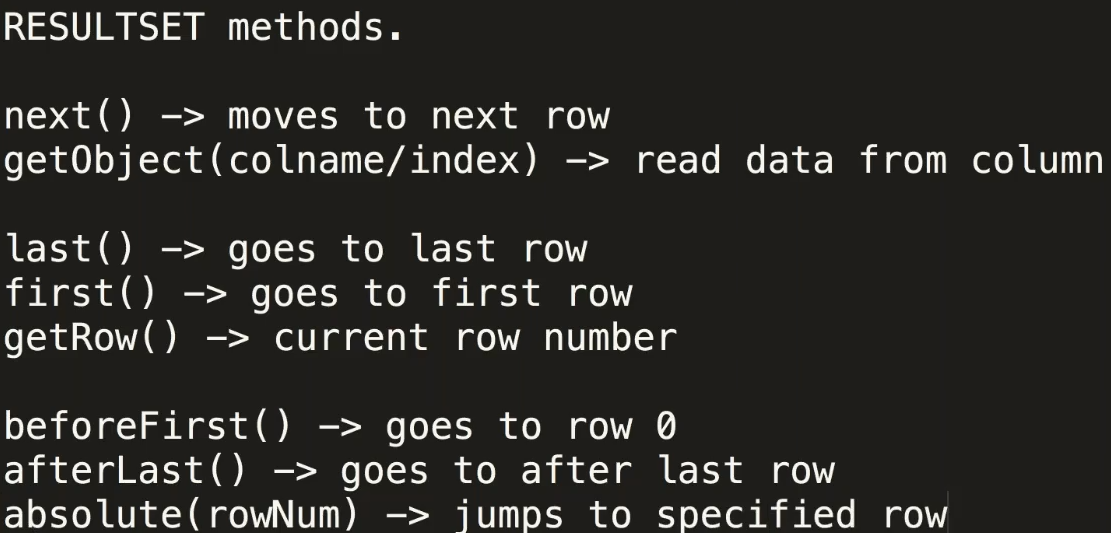








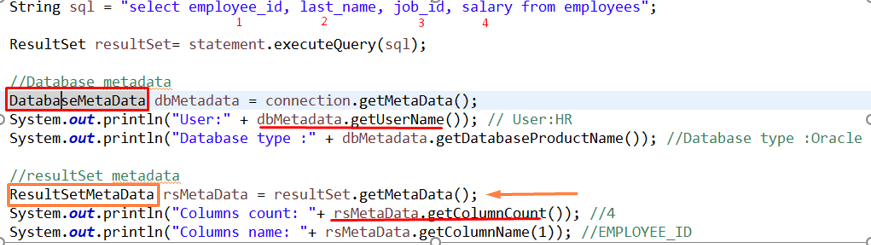


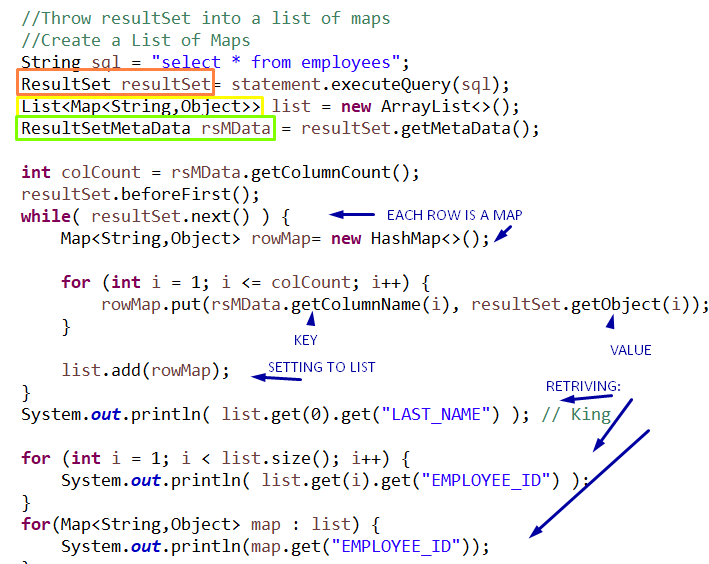


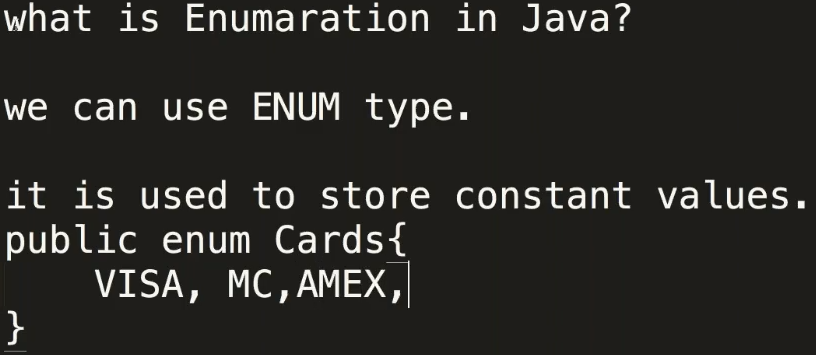
**MY JDBC**

**Meta DATA – DATA ABOUT DATA (ADDITIONAL INFORMATION ABOUT DATA)**

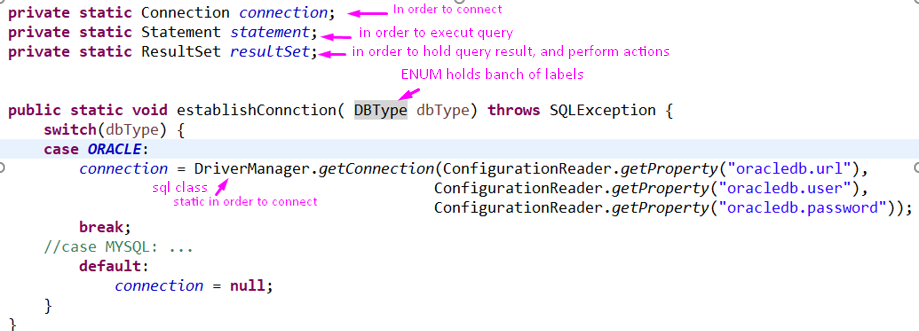
**In order to get more inf we use DatabaseMetaData and** ResultSetMetaData(used more to get data about specific TABLE like COLOUMN INFO) interfaces.

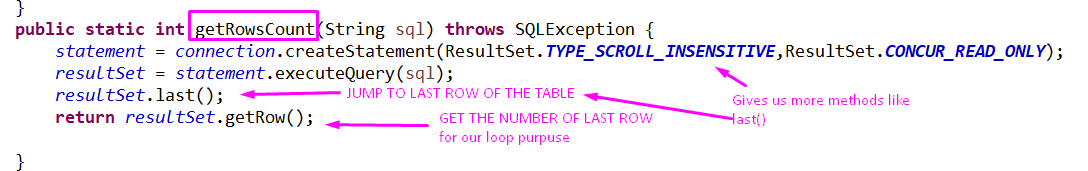


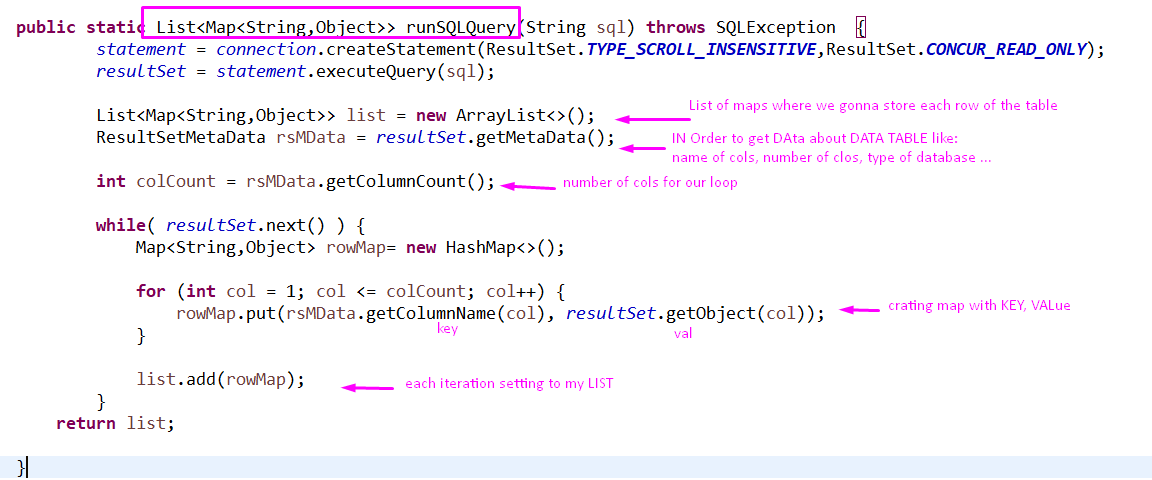


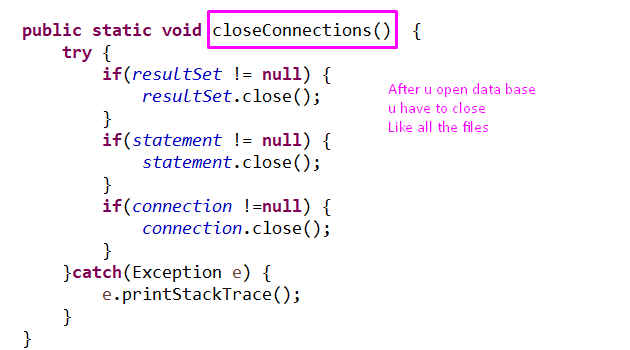


**UTILITIES:**

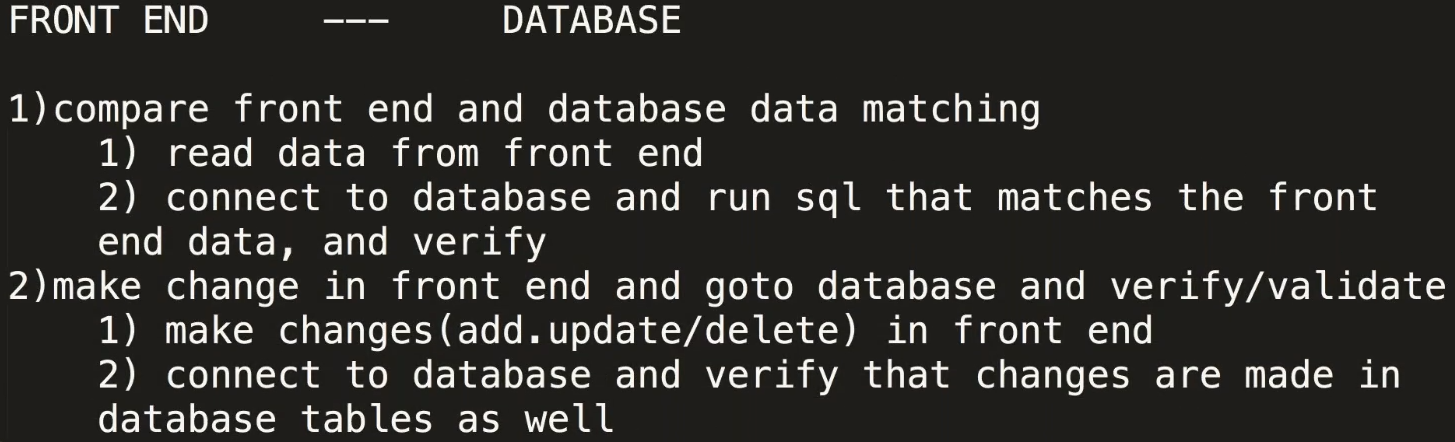












**Pagination with ROWNUM ca use (=) and (>) after all**

**select \***

**from**

**(**

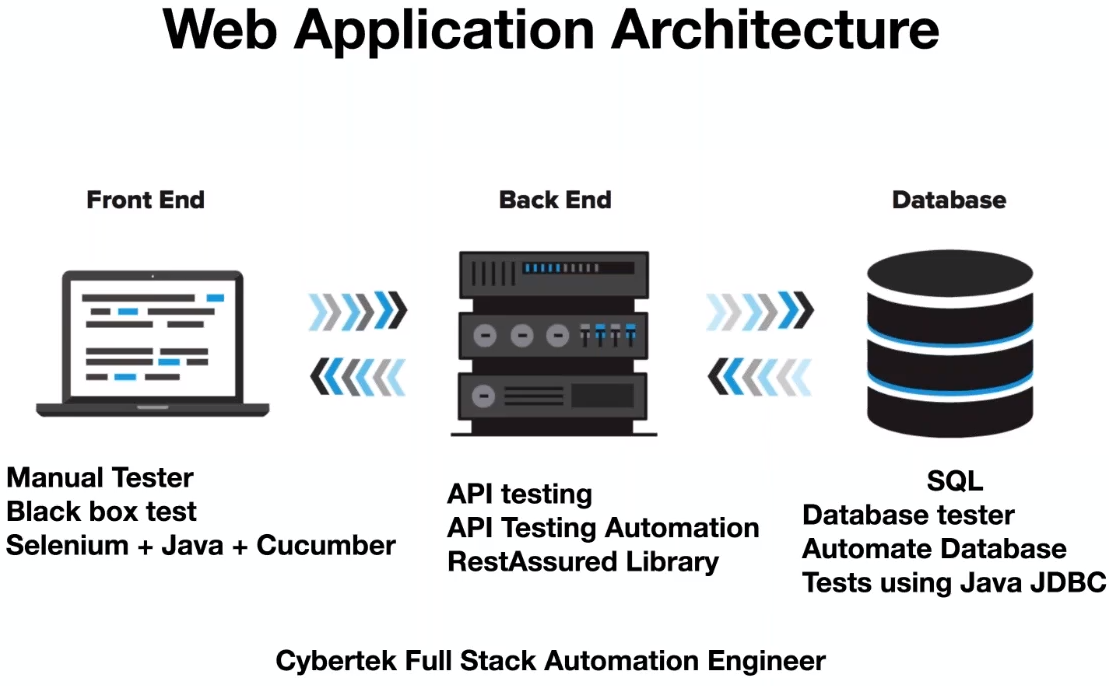
**select a.\*, rownum as rn**

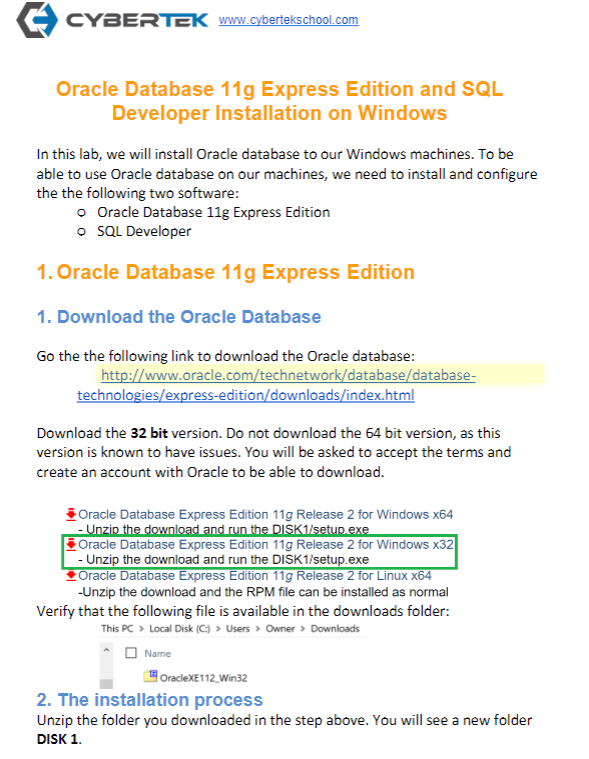
**from employees a**

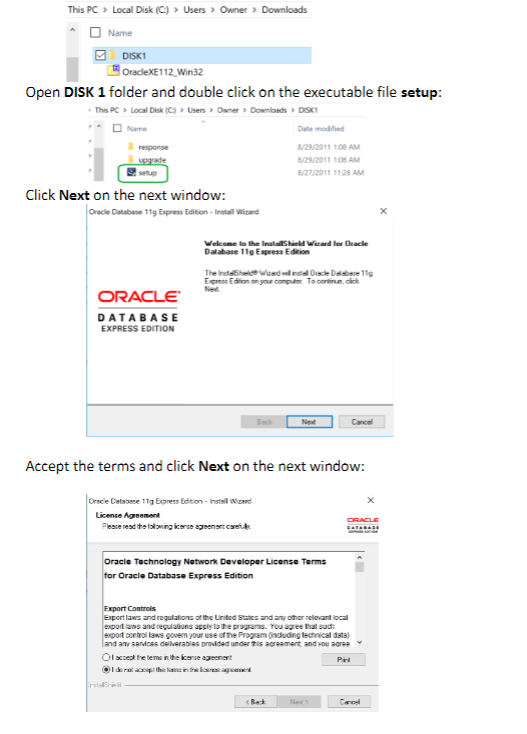
**where rownum <= 100**

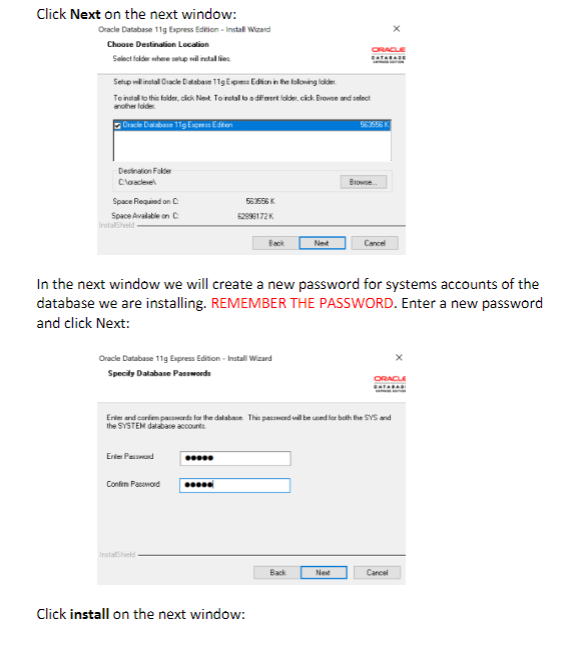
**)**

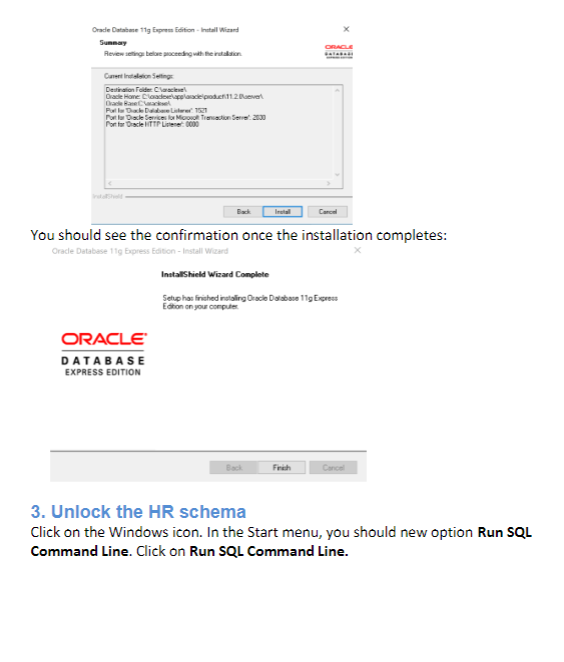
**where rn = 50 ;**

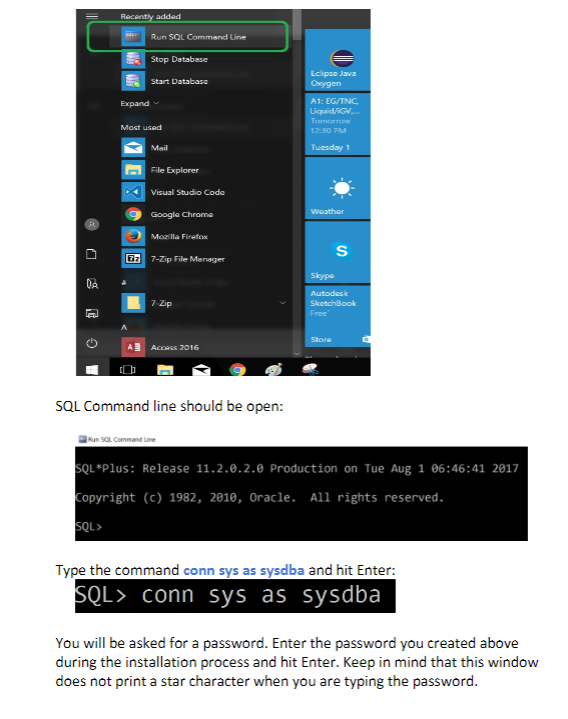


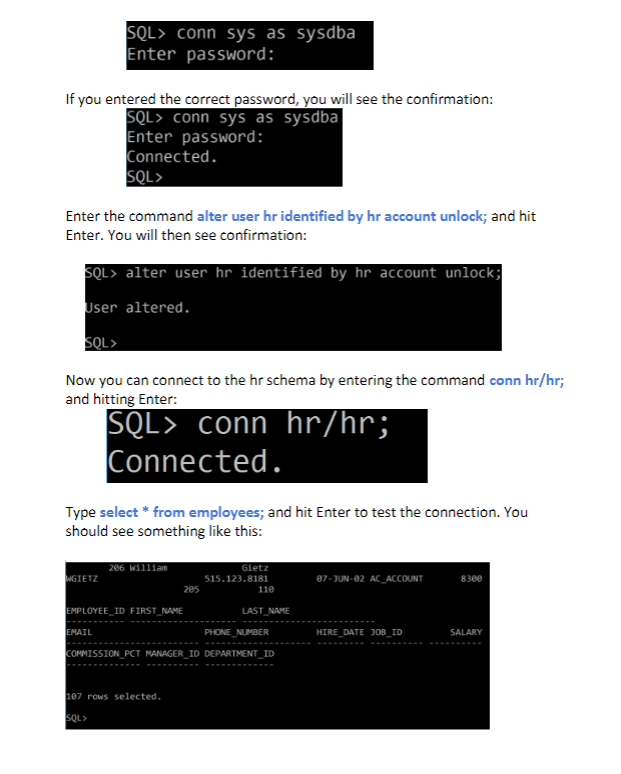




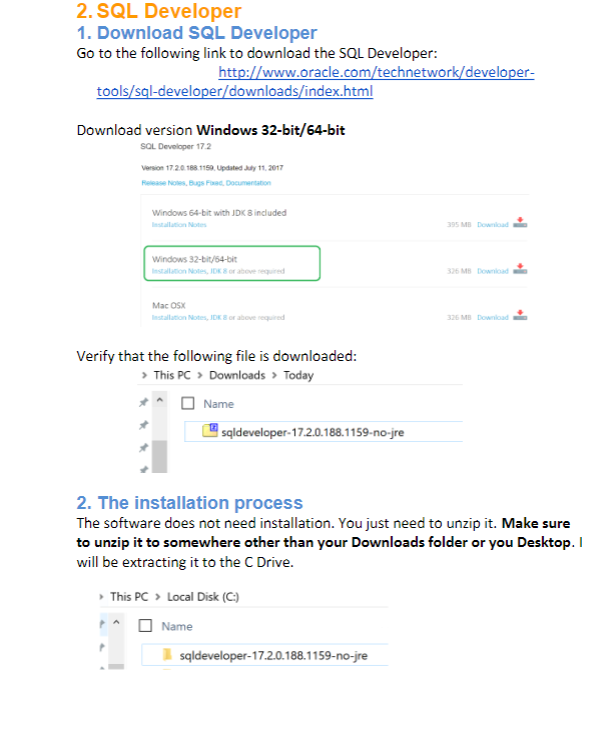


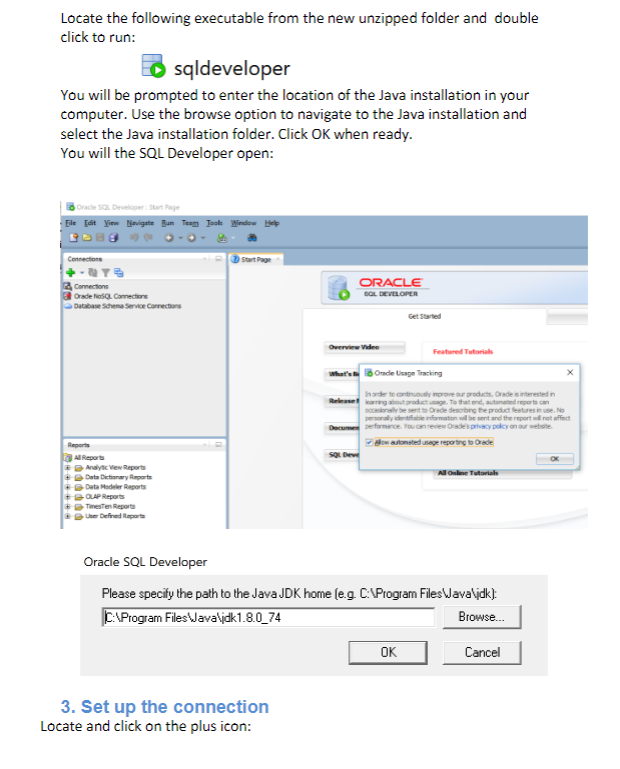


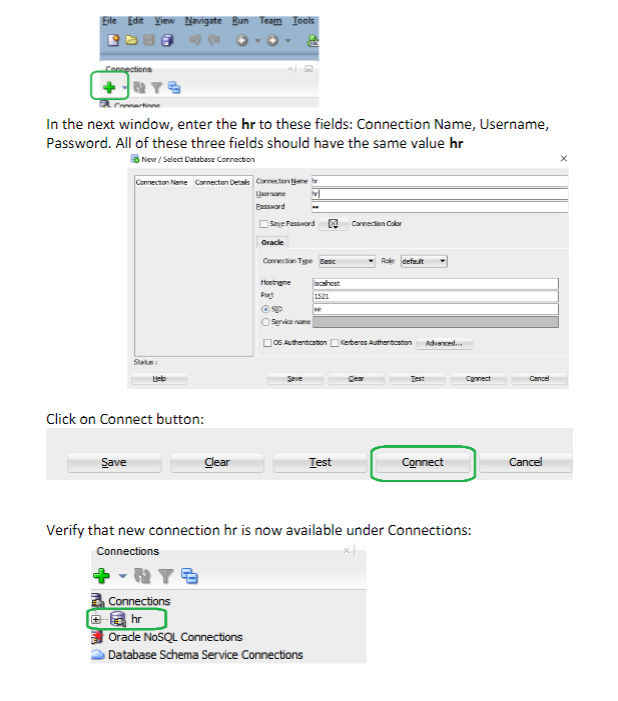




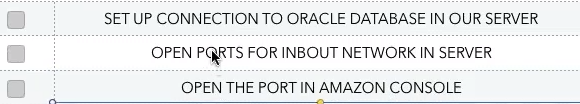
ON MAIN MACHINE!. NOW VM.



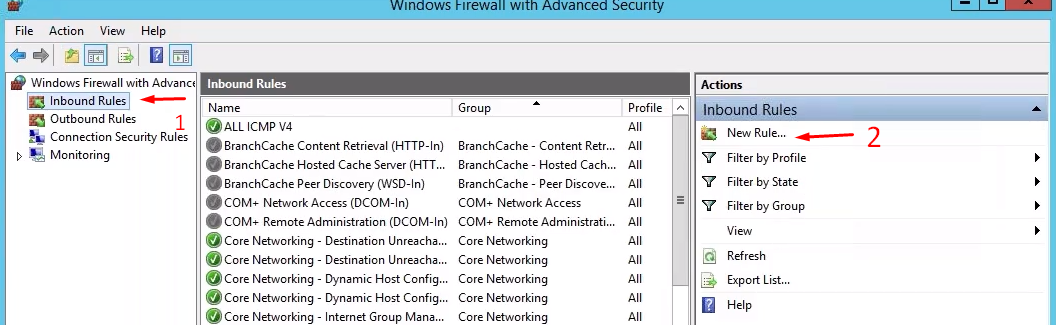




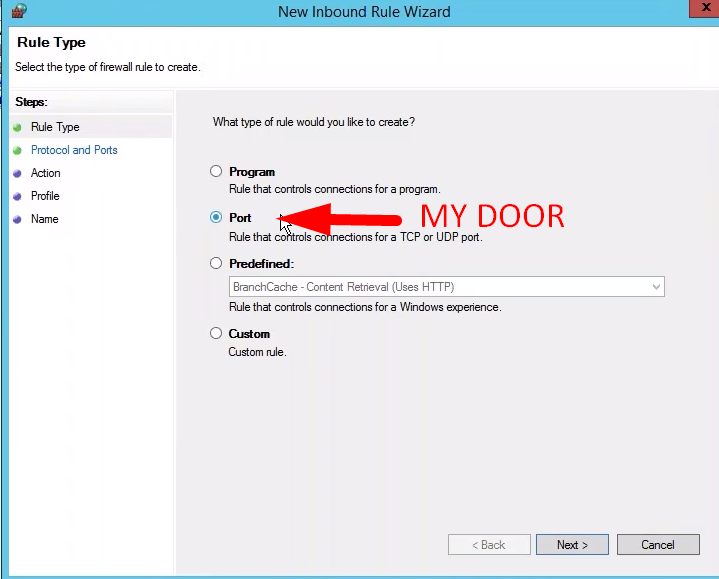
THEN .



1. I HAVE TO OPEN THE DOORS TO MY SERVER.



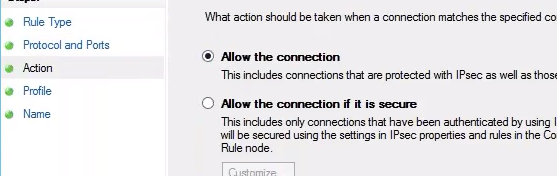
2.When u install some data base or application it will be accessible only from one door/port.



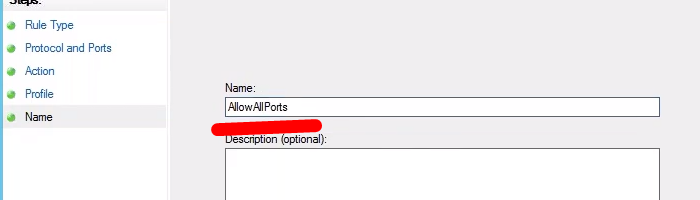
3.U can specify or use all local.



4.Allow conection



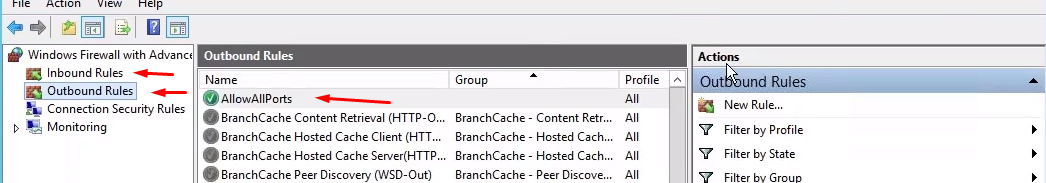
5.Allow conection



6. Then do the sAme for OUTBOUND rule.

So now it’s all open in and out ports .

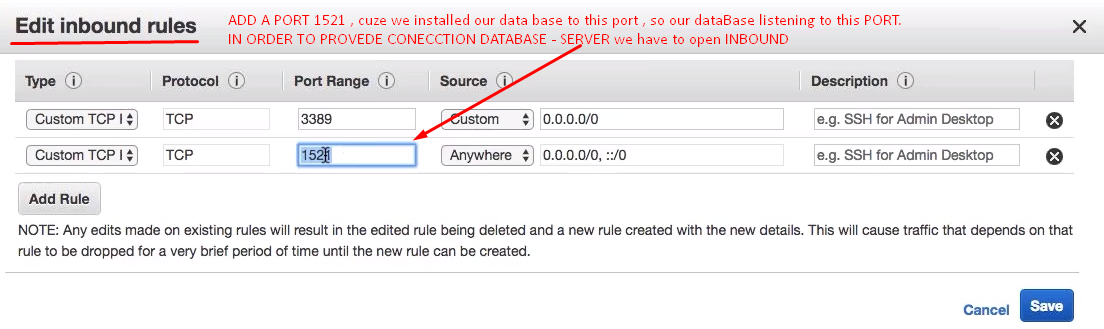
AND WE CAN REACH THIS MACHINE.



7.

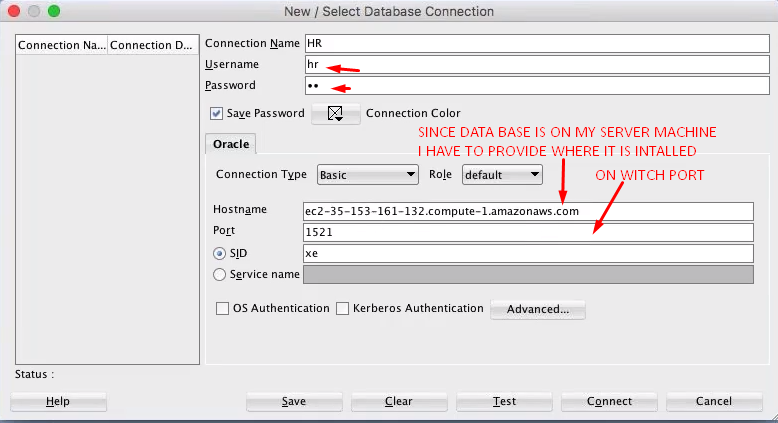






8. NOW WE HAVE TO



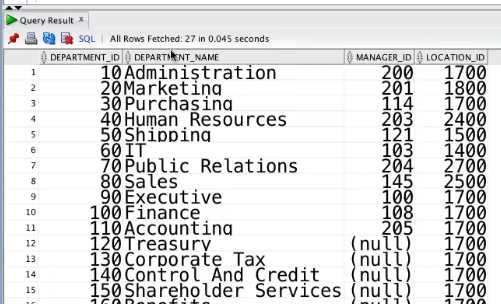


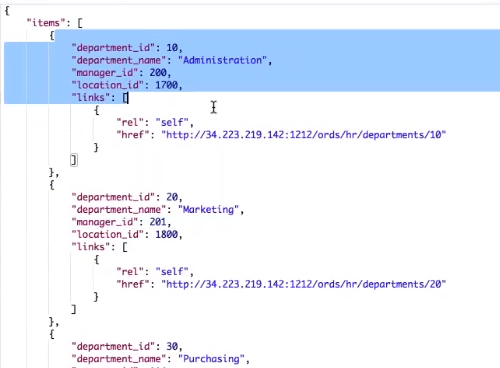
So when u write sql , THE DATA WILL BE COMING FROM DATABASE ON MY INSTALLED SERVER.

### ORDS: ([Oracle REST Data Services](https://www.oracle.com/database/technologies/appdev/rest.html))

(**TECHNOLOGY IS USE TO PUT TOGETHER API ON THE TOP OF ORACLE DATABASE**)  
TAKES API REQUST -> PROCCES QUERRY ->  
MAKING JASON -> AND SEND HIM BACK!

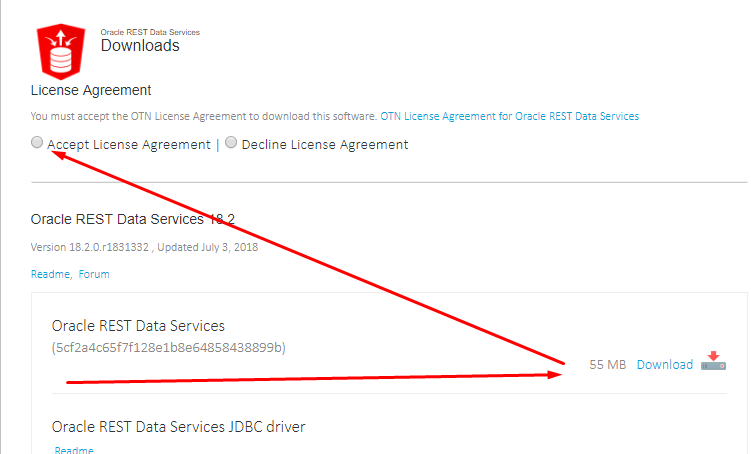


1. API COMES to data base taking data by query
2. Converting To JSON AND SENDIG BACK THE REQUST.

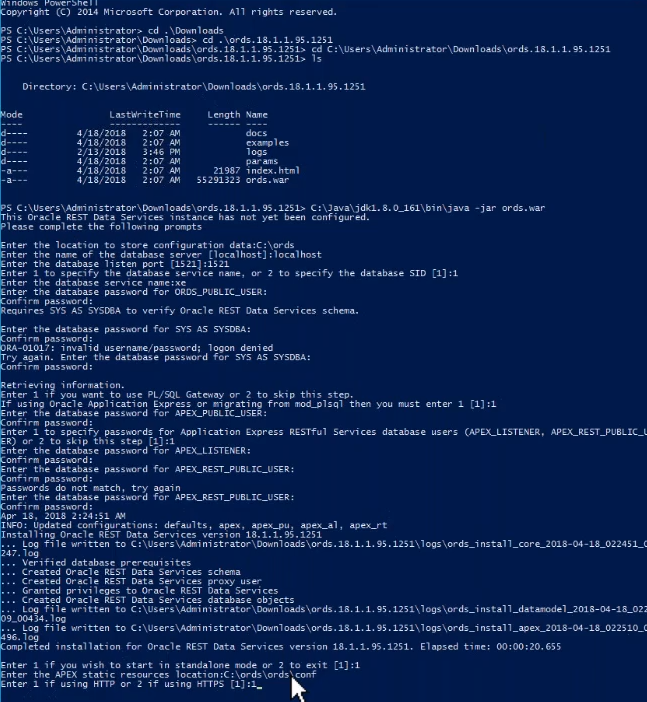


ORDS INSTALATION:

1.DOWNLOAD



2.INSTALL



3.

