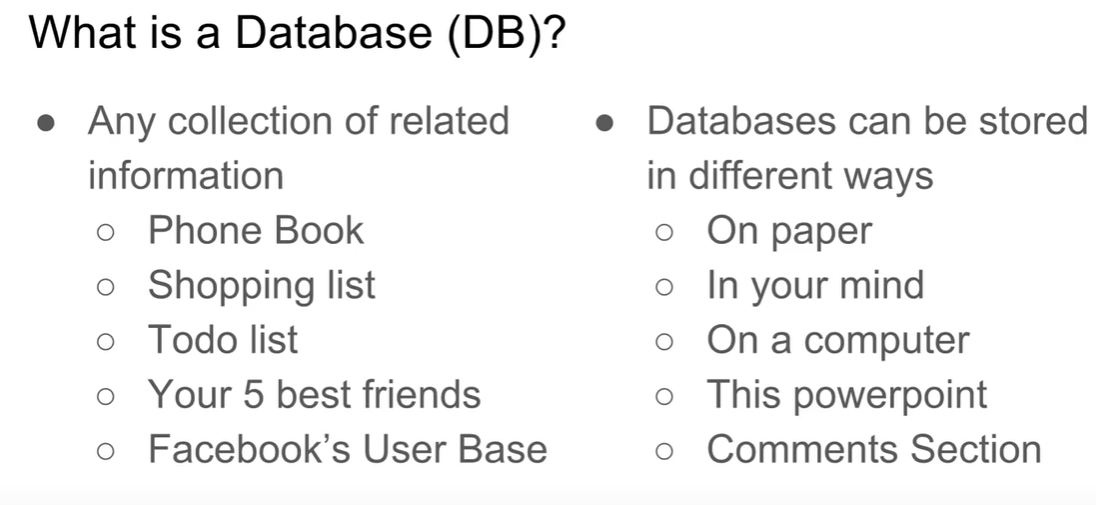
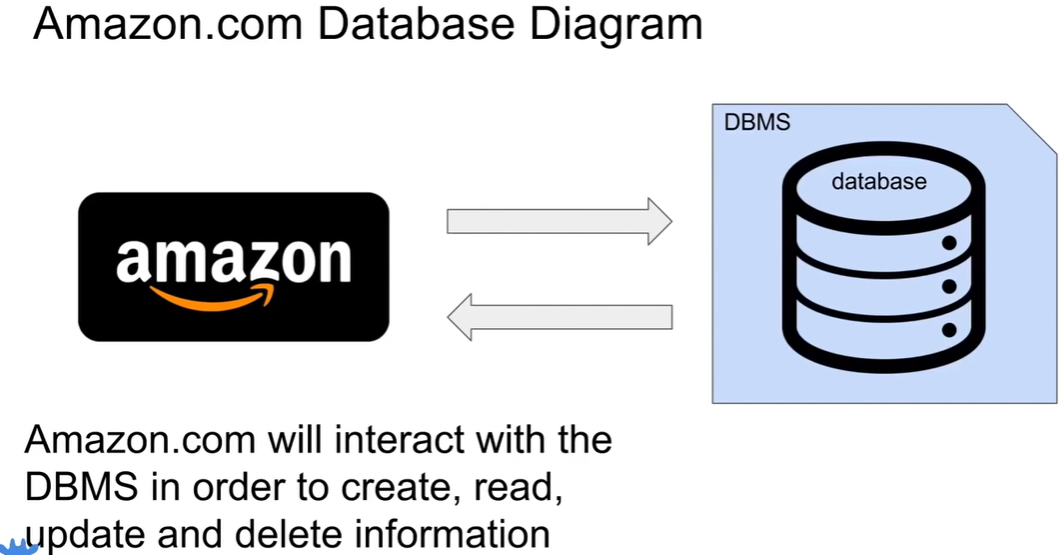
**DATABASE**

**Database:** Collection of related information that can be stored



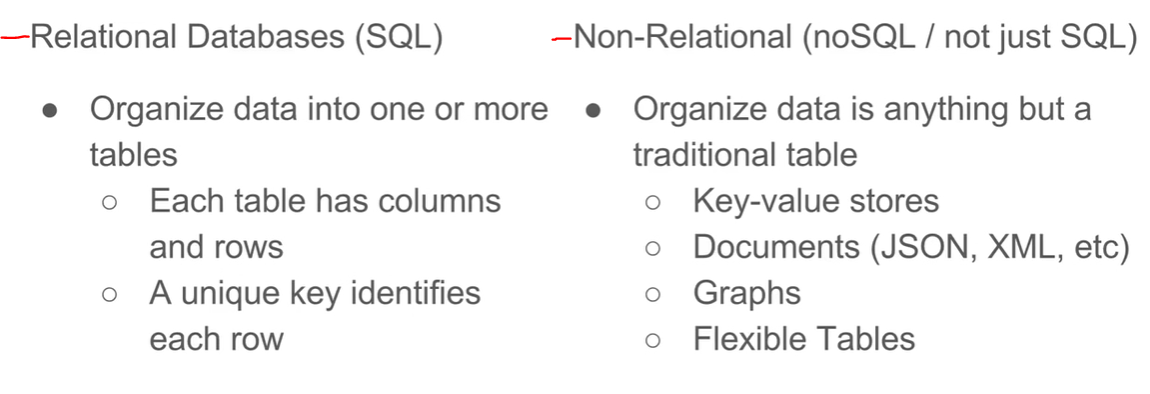




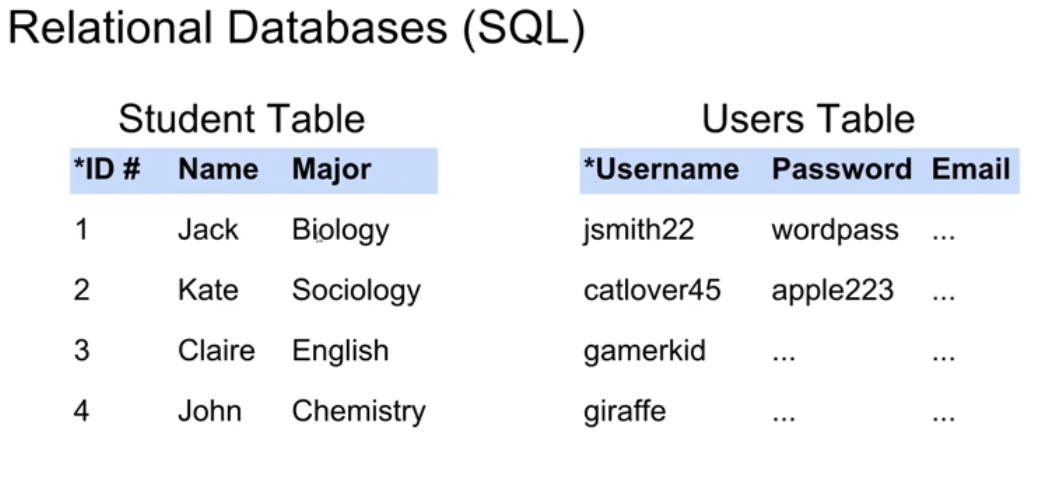
**Four Core Functions Database Management Systems Perform**



**Two Types of Databases**



**Example of Relational Databases and How They Store Data**



**Relational Database Management Systems**

RDBMS helps users create and maintain databases

mySQL, Oracle, PostgreSQL, MariaDB etc..

**SQL – Structured Query Language**

SQL is standardized language to interact with Database Management Systems to create or maintain databases.

**SQL is used for:**

* CRUD operations (create, read, update, delete)
* Administrative tasks (user management, security, backup)
* Defining tables and structures

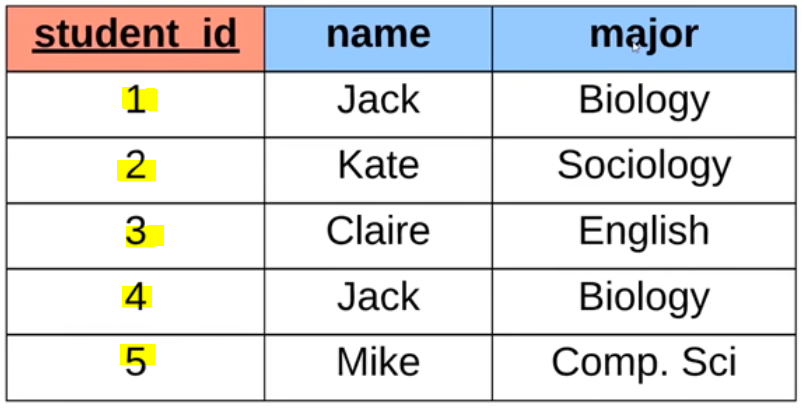
**Database Query**

As the databases structures become more and more complex, it becomes harder to get the specific information we want.

**Query is a request that’s made to the Database Management System for specific information.**

Google search is a query.

**Tables and Keys**

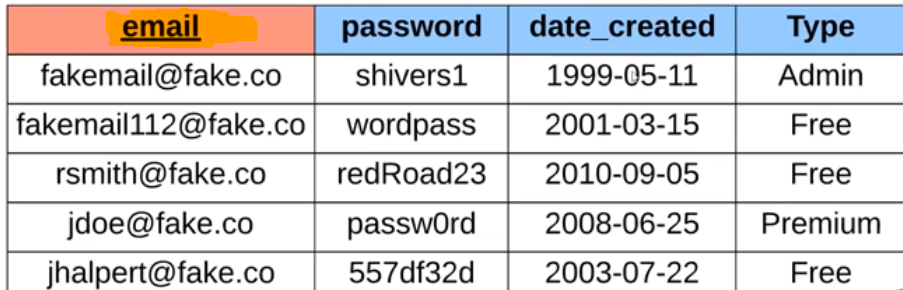
Student Table

**Primary Key:** is a field uniquely identifies the row/record

* Unique
* Never changing
* Never Null

In this table Student\_id is Primary Key

When we say **Student\_ID 1**, we would be able to find the rest of the information about that row/record (name Jack and major Biology) without any confusion with Student\_ID 4 Jack



In this table Email is the

Primary Key

**Primary keys can be Surrogate key or Natural Key**

* **Natural Key is a key that has real-world value**

**SSN number, Phone number, Email address can be examples of natural keys that has real world meaning**

* **Surrogate Key is a key that does not have a real-world value**

**Employee\_ID which can be a made-up number that has nor real-world meaning**

**Sale\_ID which can be a made-up number that has nor real-world meaning**

**It can use Auto Increment that means each time you add an employee or sale**

**ID numbers automatically will be created**

* **Alternate Key is any other key on the table that is not the primary key**

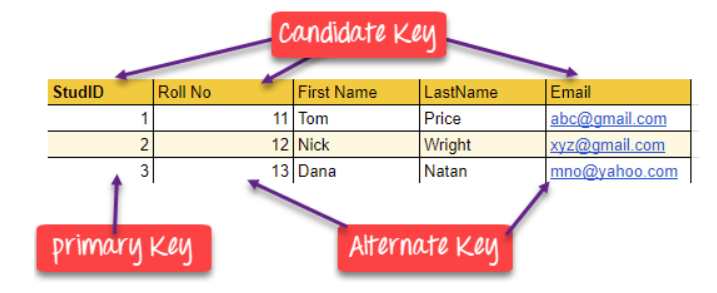


**In this table StudID, Roll No and Email**

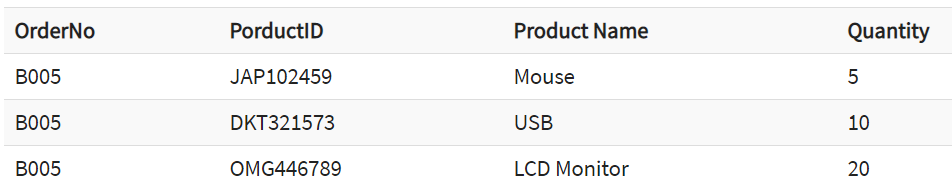
**All these three columns are qualified to be a primary key because they all can uniquely identify the row/record.However only StudID is defined as primary key for this table**

**So, in this example Roll No and Email are Alternate Keys as well as Frist Name and Last Name**

* **Candidate Keys are the keys in the tables that qualify to be the primary key**

**StudID, Roll No and Email are Candidate Keys** 

* **Compound key is used when one column is not enough to uniquely identify a row/record**



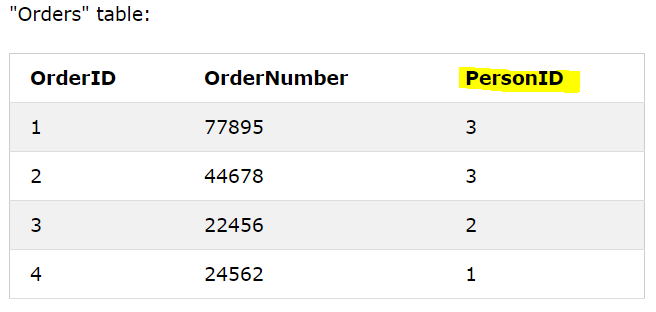
**OrderNo is not qualified to be a primary key by itself**

**How ever OrderNo and ProductID together can identify uniquely a row**

**So OrderNo and ProductID together is a Compound Key**

* **Composite key is same as compound key but part of the key may be a foreign key from another table is the difference.**
* **Foreign key** is a key used to link two tables together.

**PersonID is primary key in Persons Table**

**When you look at the Orders Table**

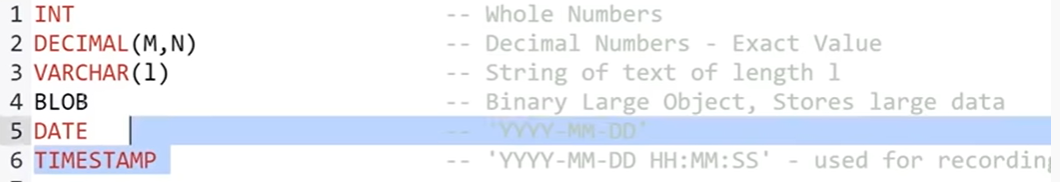
**PersonID is Foreign Key for Orders Table**

**SQL**

Before we are writing queries, we just need to understand the concept of how databases are created in simple examples.

Databases consist tables and inside these tables we store information.

* **Some Basic DATA TYPES**



1. **INTEGER**: INT Exact numbers with no decimals
2. **DECIMAL: DECIMAL (M,N)**: Exact number with decimals Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after
3. **VARIABLECHARACTER**: VARCHAR(n) Variable character with max n characters
4. **BINARY LARGE OBJECT**: BLOB Stores large data (images, files)
5. **DATE**: DATE Stores date YYYY-MM-DD
6. **TIMESTAMP**: TIMESTAMP Stores date and tame YYYY-MM-DD HH:MM:SS

* **Sql command to create database so that we can create a table under:**

**Create database databaseName;**

* **After creating the database, command to use that database:**

**Use databaseName;**

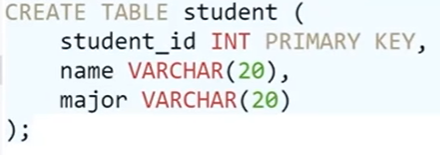
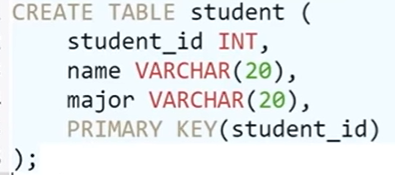
**CREATING DATA TABLES**

Let’s Create this table

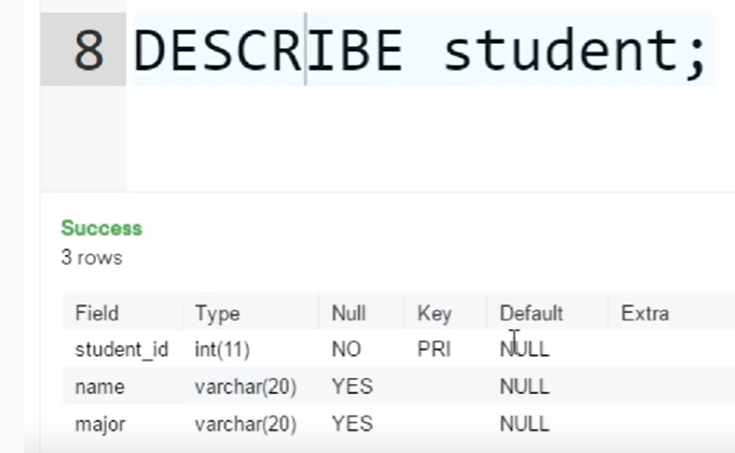


* We want to create a table with the name of student
* We need three columns **student\_id, name, major**
* student\_id column will use exact numbers with no decimals so we use INT also student\_id is primary key so we define that in SQL command
* name column will use text and 20 characters is enough for a name so we use VARCHAR(20)
* majot column will use text and 20 characters is enough for a name so we use VARCHAR(20)

We can also define the primary key at the end. It does the same thing.

**DESCRIBE** is used to describe the table



**DROP TABLE**

**We can use DROP TABLE To delete the table from database**



**ALTER TABLE**

**We can add a column to the table**



**We can delete a column from the table**

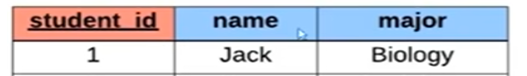


**INSERTING VALUES to TABLES**

General code for inserting values to the table is: (student is the table name)



The order of columns we created our table is important **(student\_id, ‘Jack’, ’Biology’)**



So our code is:



Suppose we want to enter a row without the major since student has not decided yet

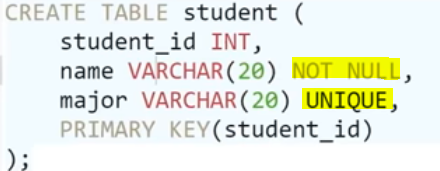
After table name student, we define the columns that we want to enter value.

In this case major for Claire will be define automatically as NULL. If Major was a PRIMARY KEY we would not be able to leave it blank because PRIMARY KEY cannot be NULL.



**NOT NULL and UNIQUE**

We can define the columns while creating the table so it helps us to limit what kind of data can be put in that column.

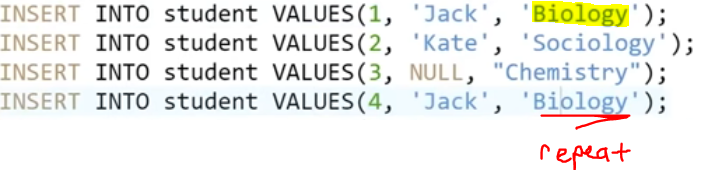


Name column has to be inserted while inserting values to the row because we defined name column NOT NULL



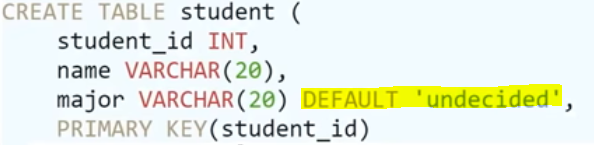
This code will be rejected because we described name NOT NULL and any name entry as NULL will be rejected.

Also, Major column cannot be repeated and cannot be the same in another major column in the table because we described it as UNIQUE



Last entry will be rejected because we described major as UNIQUE.

**DEFAULT**



While creating the table we can set some columns for DEFAULT values.

That means if they are left blank, Automatically the default value you described will be entered to that column.

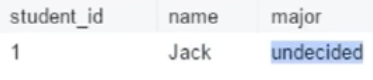
In this case its ‘undecided’ and as we can see we left major column blank in this code.

So major will be entered automatically as undecided

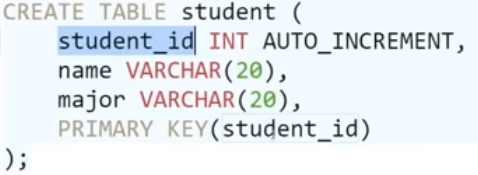
In this case its ‘undecided’ and as we can see we left major column blank in this code.

So major will be entered automatically as undecided



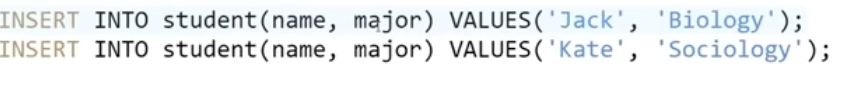


**AUTO INCREMENT**



We described student\_id as AUTO INCREMENT

So anytime we enter value to the row we don’t have to enter student\_id column because it will be automatically incremented.



Now we inserted values to this table without entering student\_id, however it is automatically created for us.



**UPDATE**

This is how our table at the moment.

We use UPDATE command to change a data in the table.

Example: School decided to change the Biology major name to be only Bio major.

So, we only want to update the student whose major are Biology to Bio.

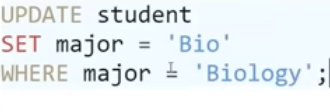
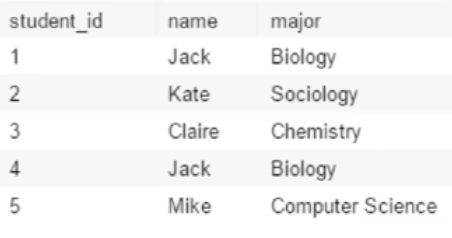
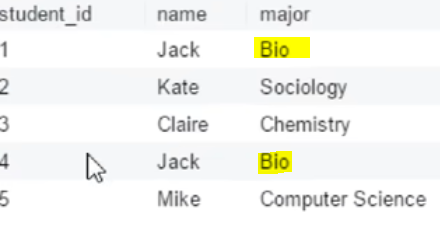
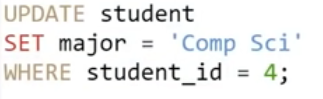


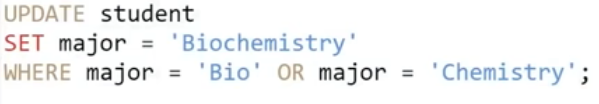
Table becomes



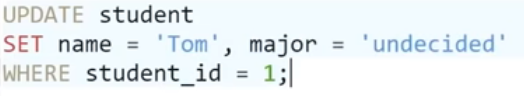
**Another example:** updating student\_id 4 major as Comp Sci

**Another example:** updating bio or chemistry majors to Biochemistry



**Another example**

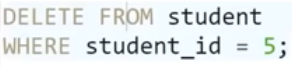


**DELETE**

**To delete all the rows and columns from the table**



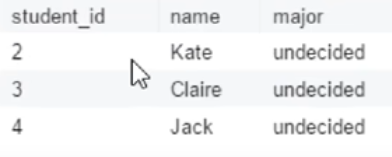
**To delete certain row from the table**



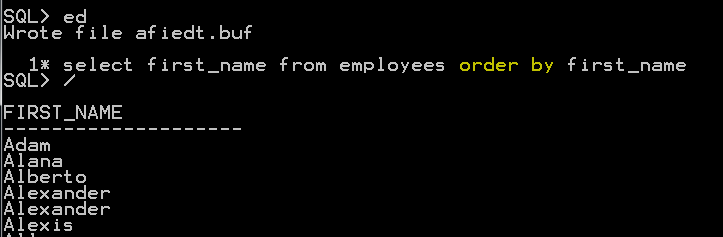


**Another example**





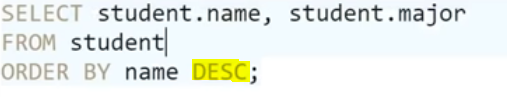
**QUERY**

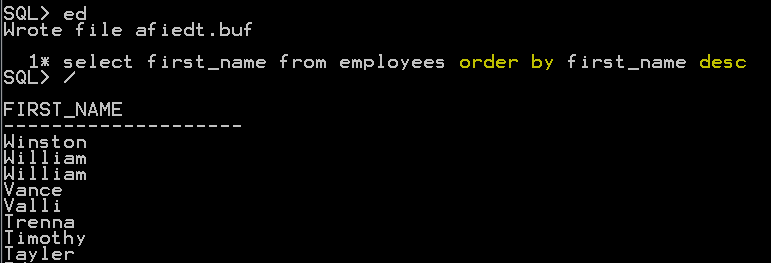


So, when you’re writing complex queries it will be easier to see which table values came from.

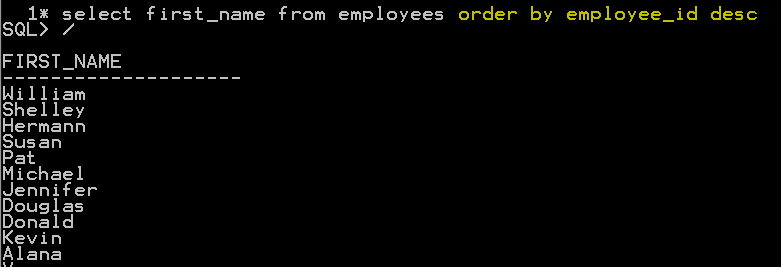
Also ORDER BY name command will put the names in ascending alphabetic order by default or you can add ASC.

If we want to put the names in **descending** order you can add DESC





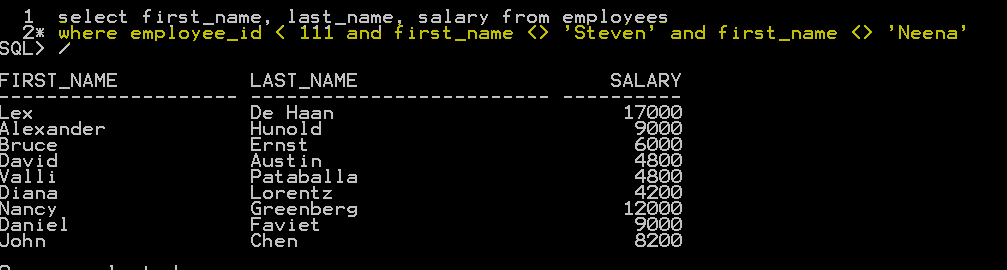
Order by employee\_id descending order

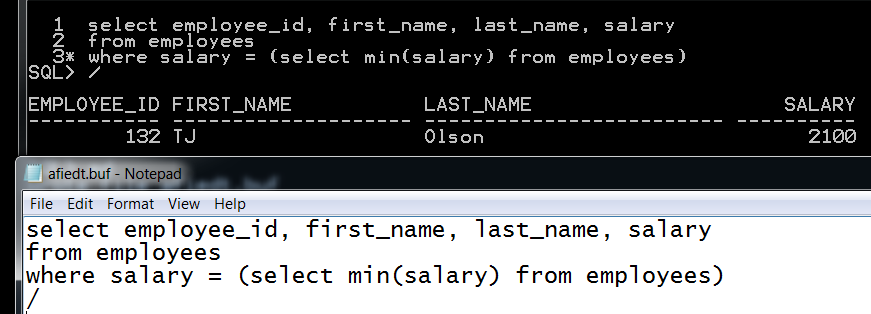


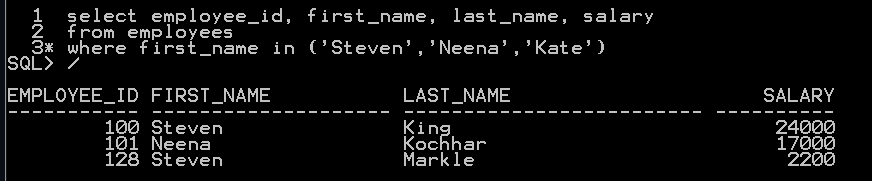
You can also order by multiple columns

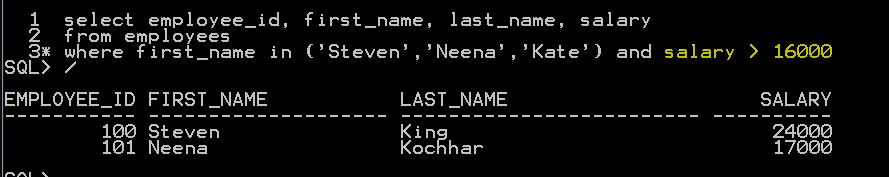
First, orders by name in descending order, then orders by employee\_id in ascending order



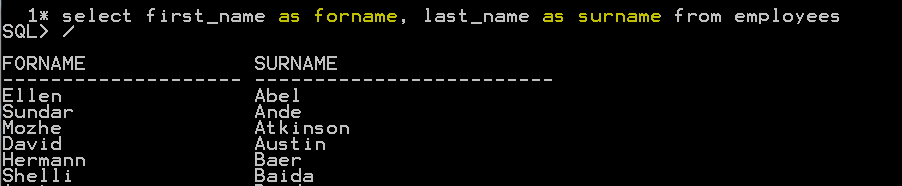








Find all employee name and last name as forename and surname



Find all different type of commission percentage employees getting

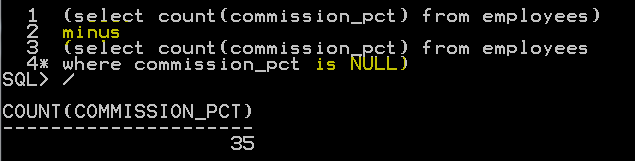


**Find all the different salary amount employees are getting and order them from highest to lowest**



**Find all the employees who are getting commission percentage**

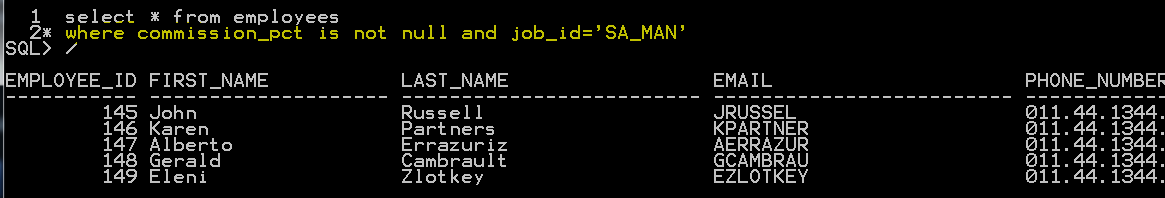
**We can do is by using MINUS keyword**



**Also we can do it shorter without minus**

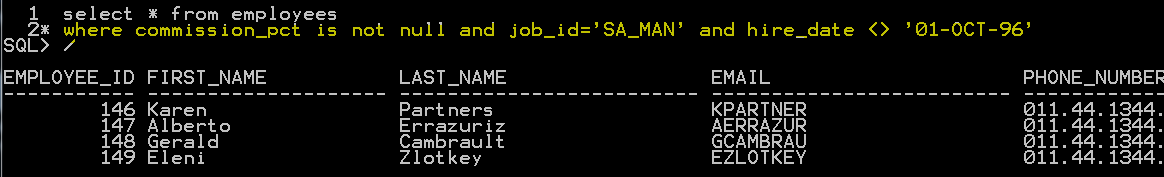


**Find all the salesman that gets commissions**

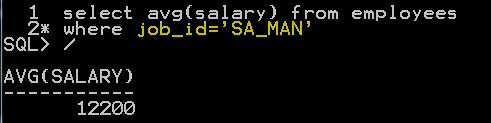


**Find all the salesman that gets commissions and hire date is not 01-OCT-96**

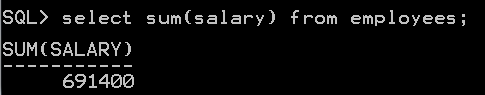
**As you see John Russell is out because his hire-date is 01-OCT-96**



**Find average salesman salary**



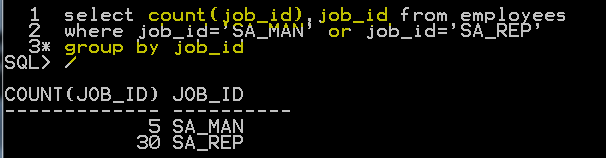
**Find the payroll per month**



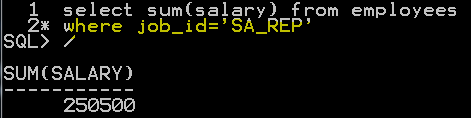
**Find how many different job positions available in the company**



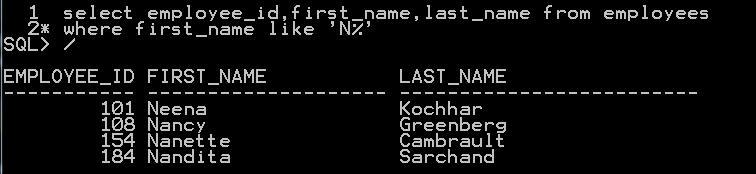
**Find how many SA\_MAN and SA\_REP in the company**



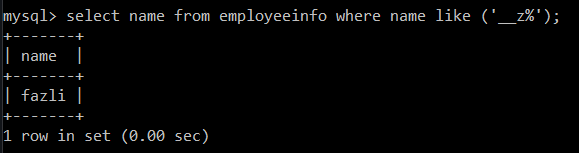
**Find how much salary per month is paid for sales reps**



**Find employees with their name starts with N (percentage sign is on the right side)**



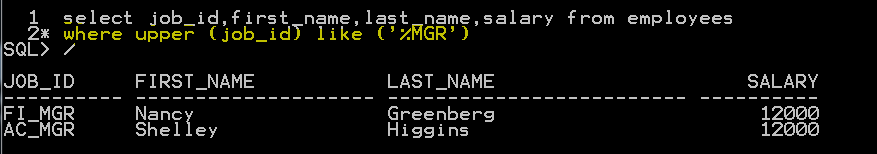
**Underscore stands for one character**



**Find employees with their name ends with N (percentage sign is on the left side)**



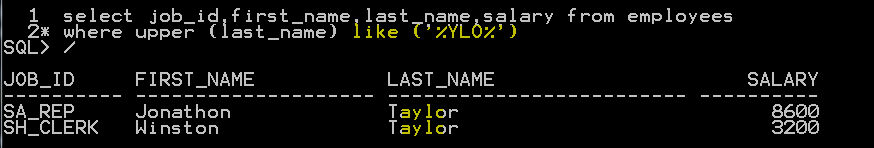
**Find all types of managers and their salaries (percentage sign is on the left side)**



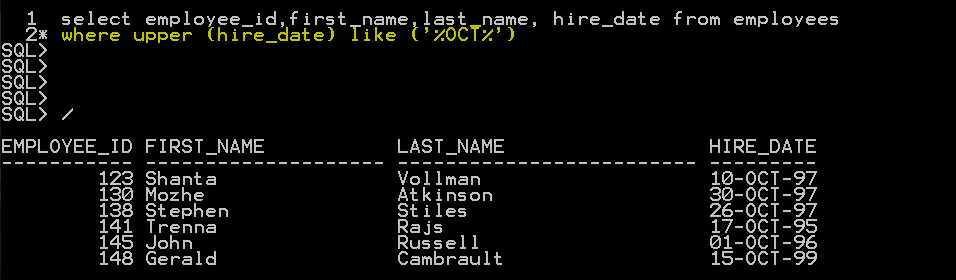
**Find employees whose last name includes YLO anywhere**

So, we learn to search letter at the beginning or end by putting % either to the beginning or to the end

If we want to search in between the words, we will put % at the beginning and at the end (‘%xyz%)



Find any employee that were hired in October

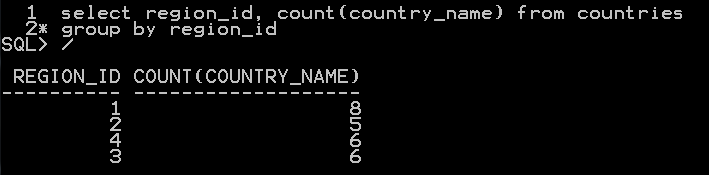


**Order by for column names --- group by for count**

Find all countries and order them by region Order by for column names

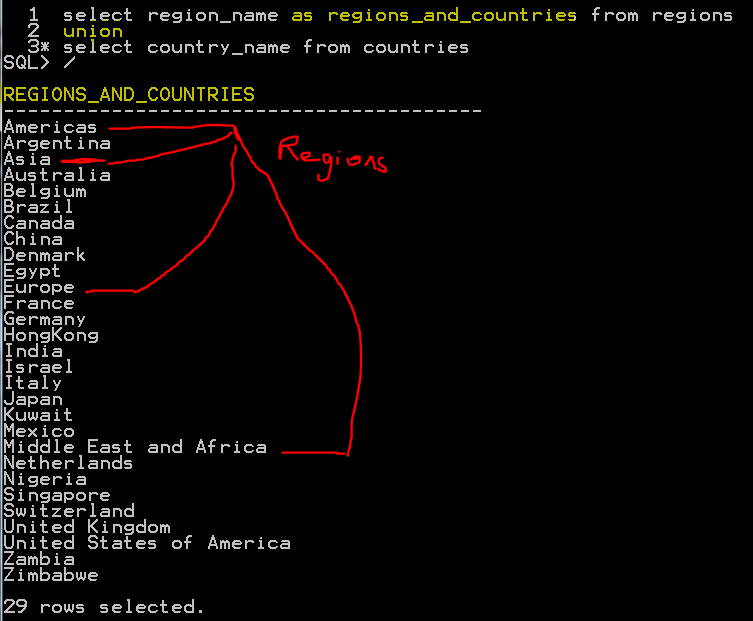


Find number of countries in every region group by for count



**Using the UNION Key Word**

**UNION** keyword is used to gather information from more than one table and show it in the same result

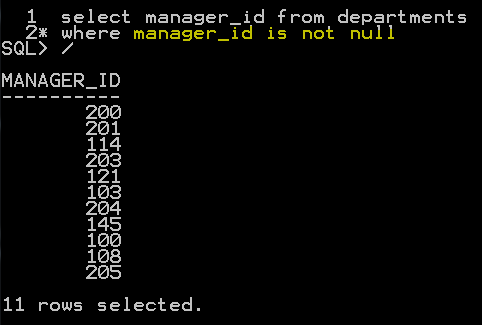


**NESTED QUERIES**

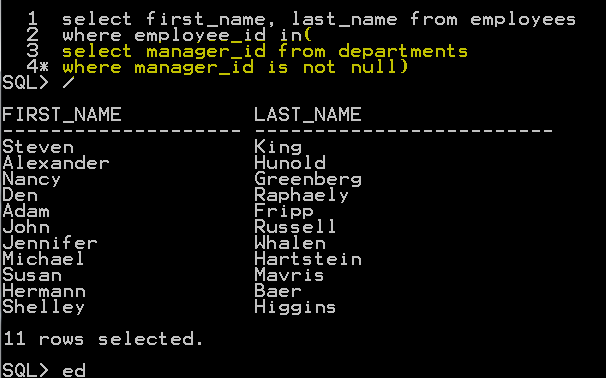
Nested queries basically writing two step queries in one step by nesting first on inside the second.

**EXAMPLE** **Find all the names of the department managers**

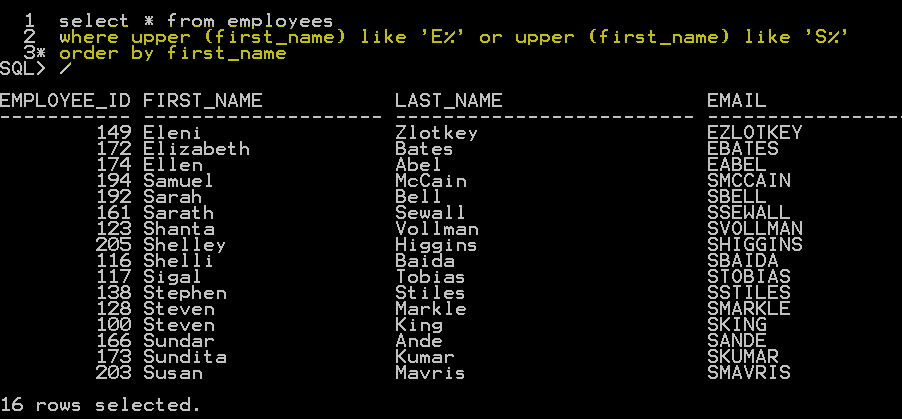
1. First, we will find manager\_id of the managers from departments tables

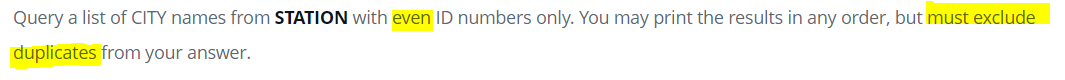


1. Then we are going to match these manager\_id’s with employee\_id’s from employees table



**Basically, we put the first line inside of second line to get the final information of names of managers. where employee\_id in (first line);**

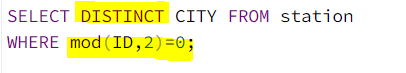


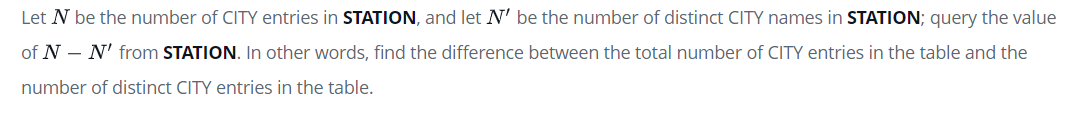


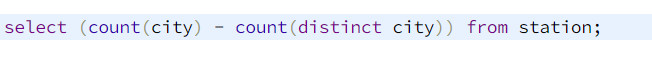
**AVOID Duplicates with DISTINCT keyword**

**To find even numbers mod(column\_name,2)=0**

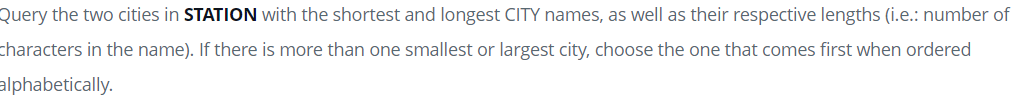
**To find odd numbers mod(column\_name,2)=1**

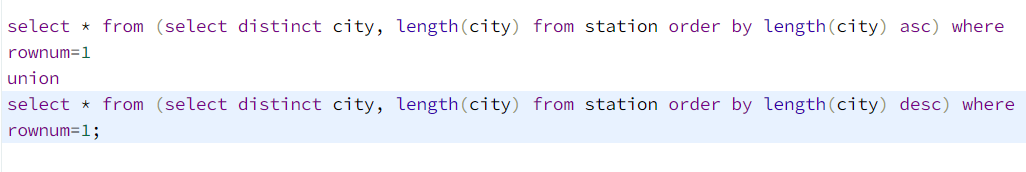




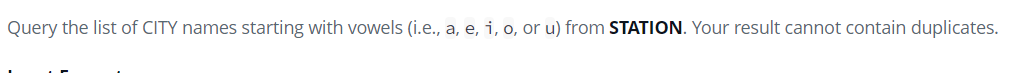


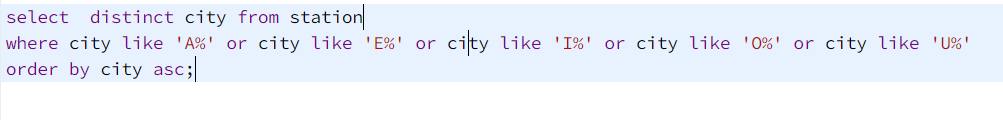
**----**



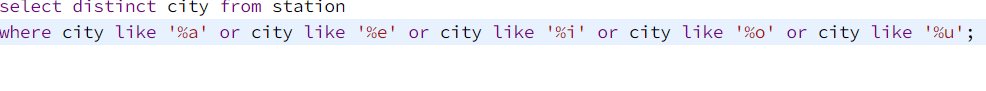


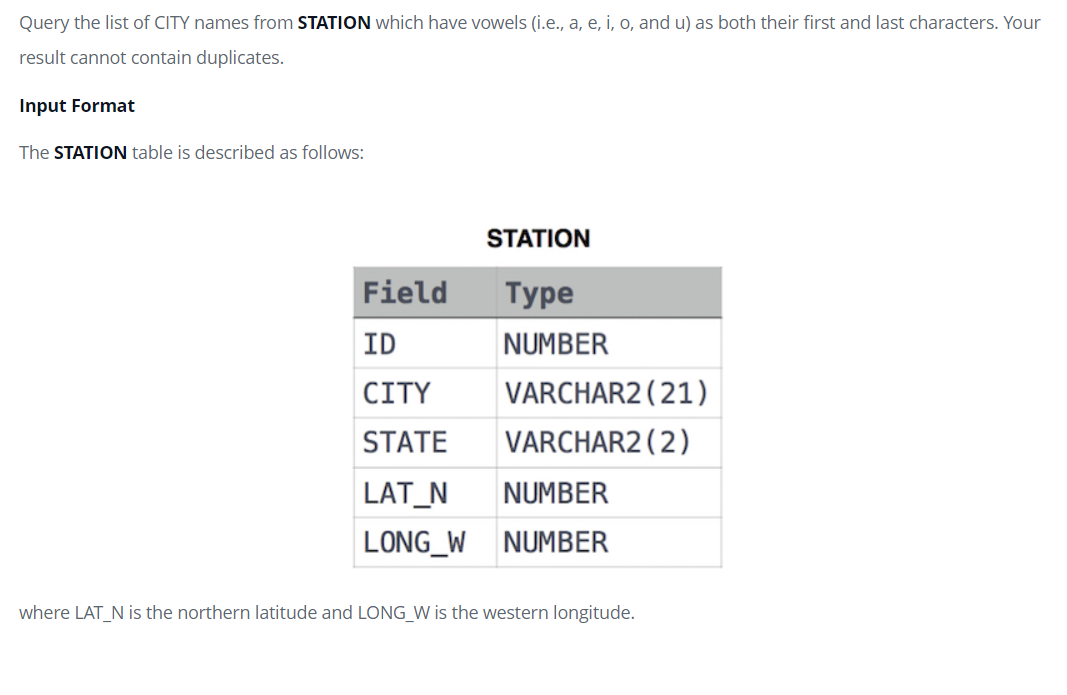
**----**





**ENDING WITH**





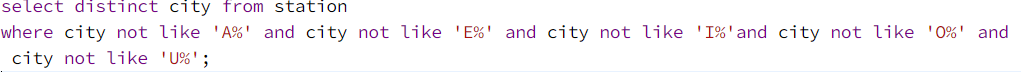
**ASK**

SELECT DISTINCT(CITY) FROM STATION WHERE CITY REGEXP '^[AEIOU]' AND CITY REGEXP '[AEIOU]$' ORDER BY CITY

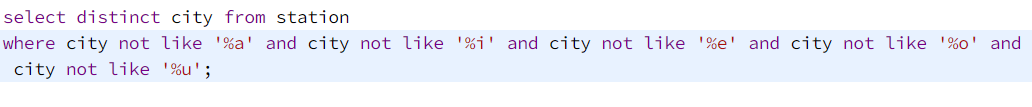
select distinct(city) from station where (city like '%a' or city like '%e' or city like '%i' or city like '%o' or city like '%u') and (city like 'A%' or city like 'E%' or city like 'I%' or city like 'O%' or city like 'U%') order by city

**---**

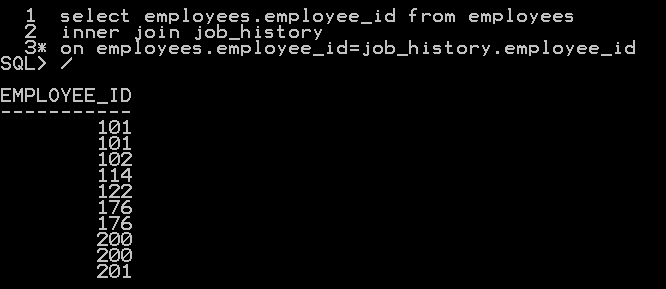


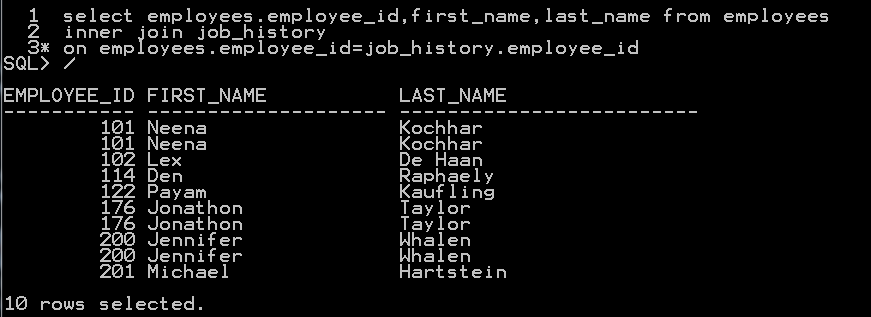






**Inner join**



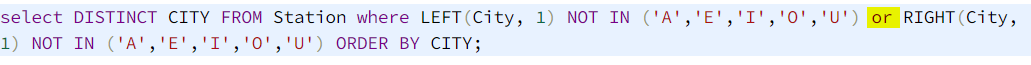


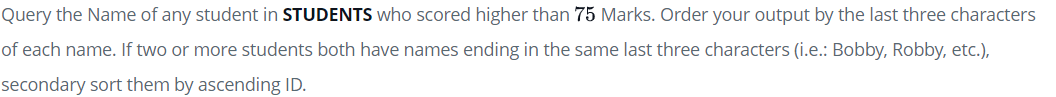
**LEFT and RIGHT**

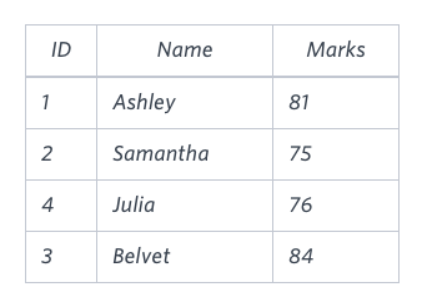
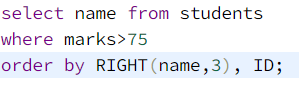
**City names that doesn’t start and end with a,e,i,o,u**



**City names that either doesn’t start or end with a,e,i,o,u**

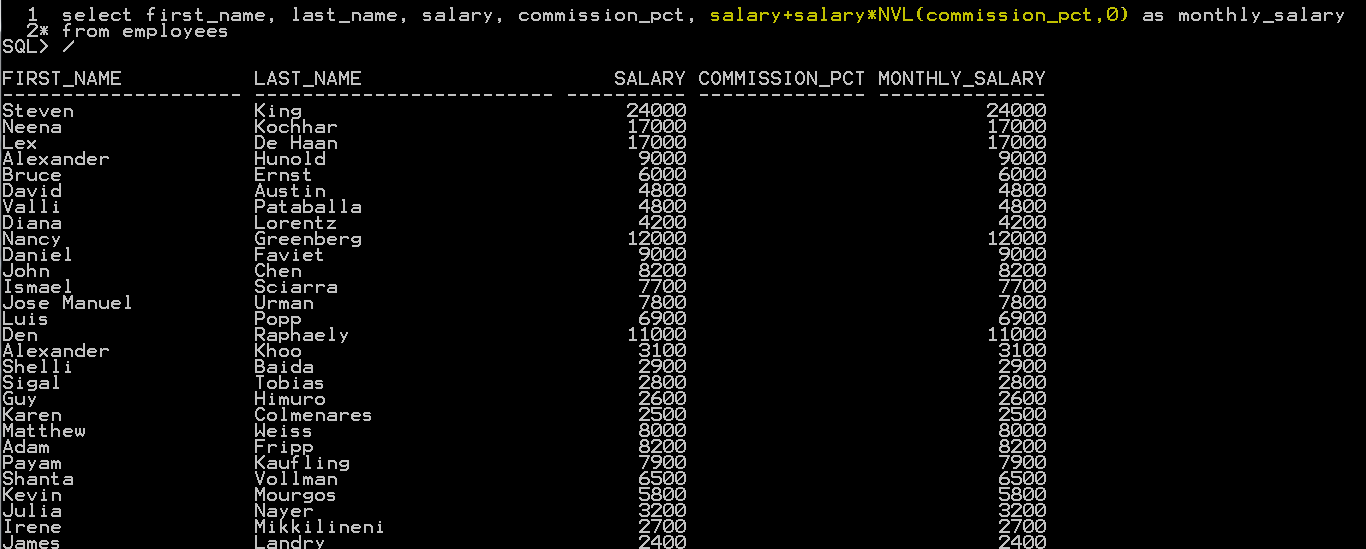




**DATA MANIPULATION**

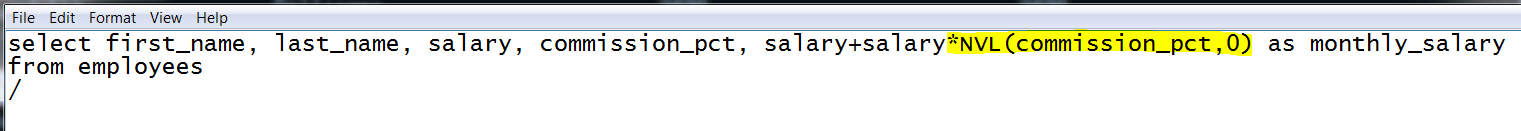
NULL VALUE (NVL)



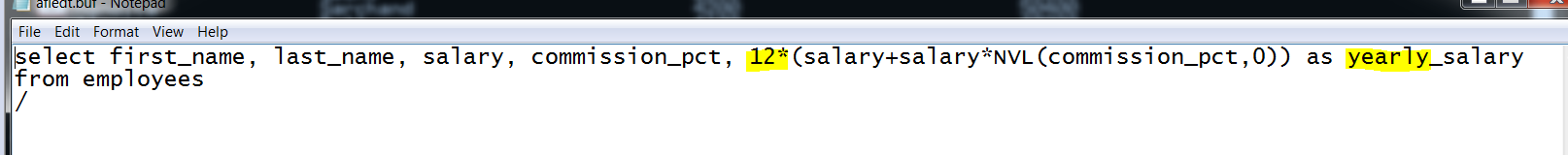
**NVL(ColumnNameX,0)**

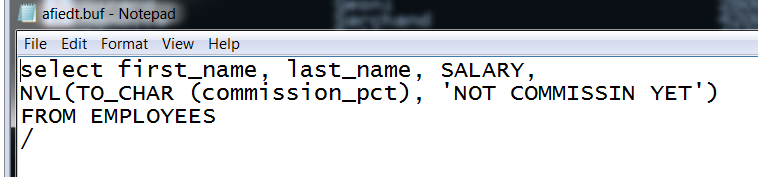
**This says where there is Null use 0**

**Calculate monthly salary with commission**

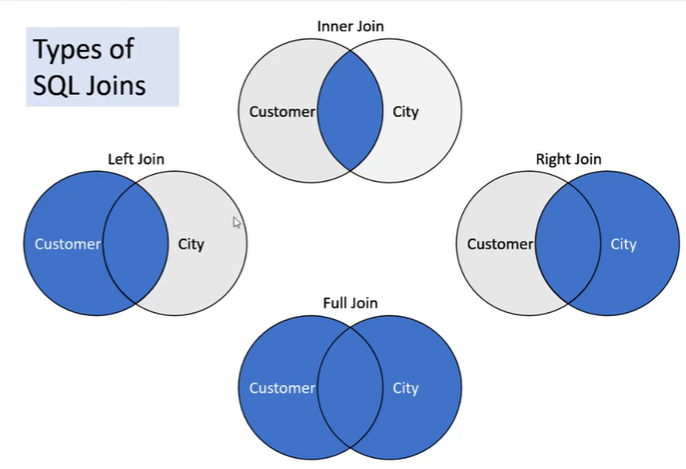


**Calculate yearly salary with commission**





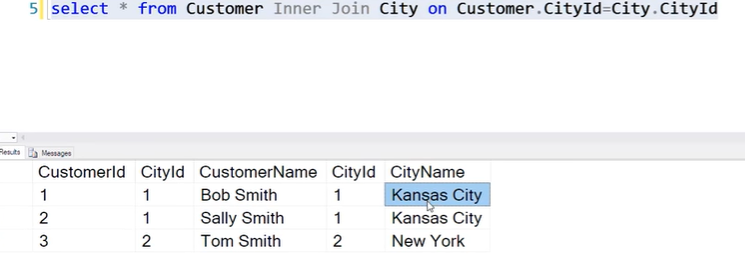
**JOINS**



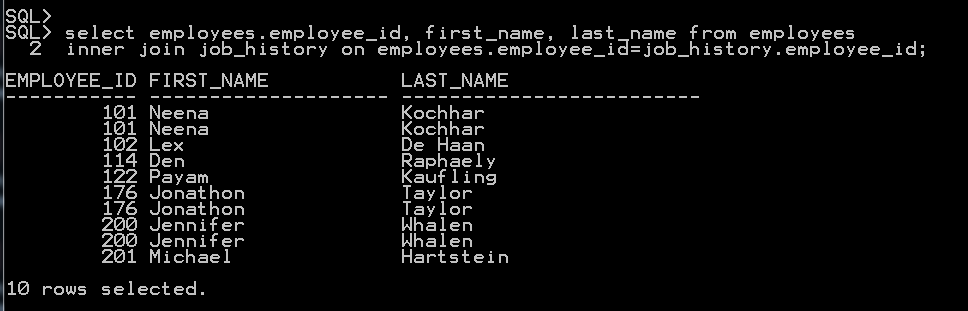
**Customer Table City Table**

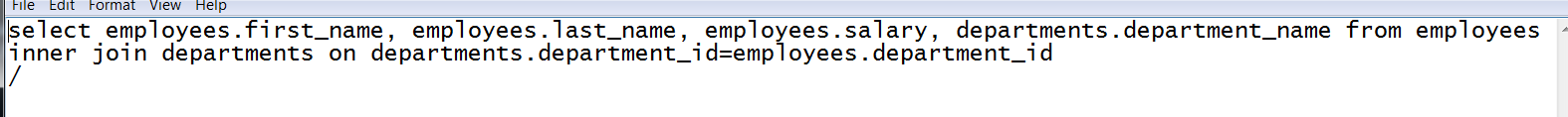


**INNER JOIN**

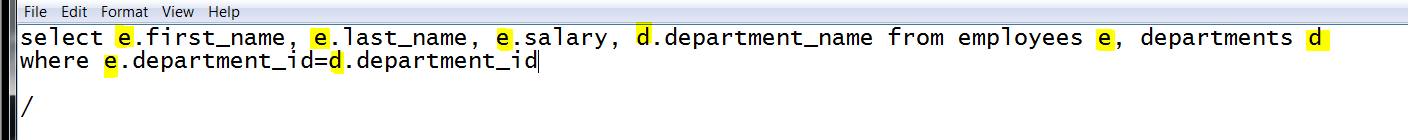


**Inner join selects only the records that matches on the both tables**

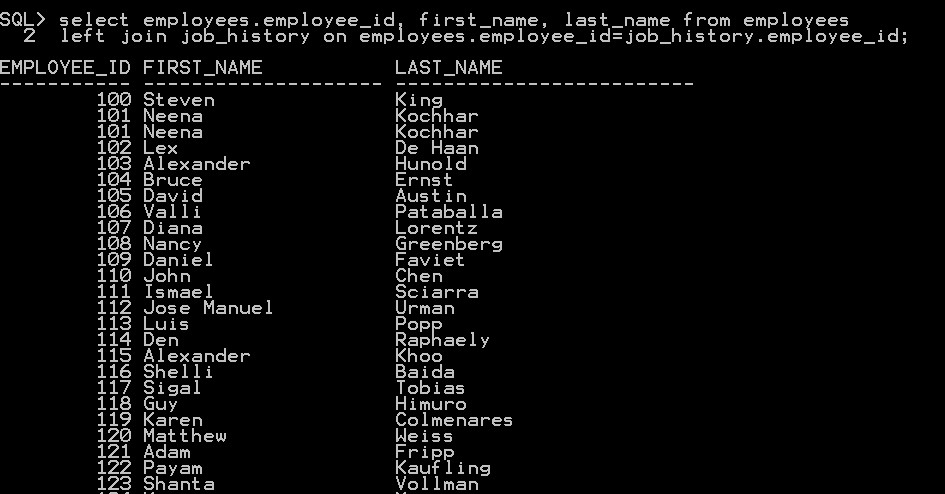




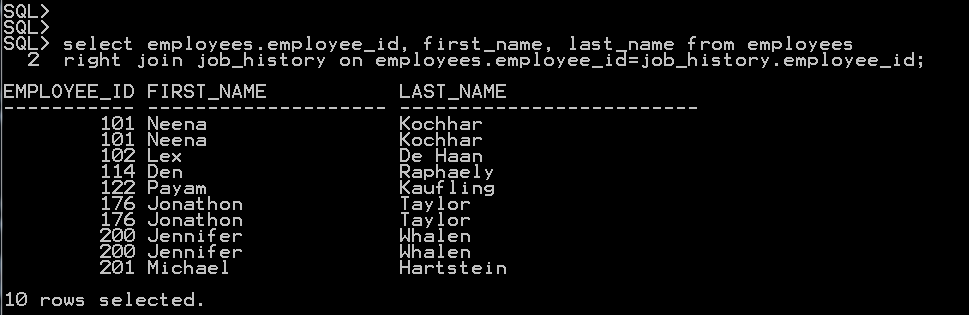
**You can define short cut for table names by defining in from statement**



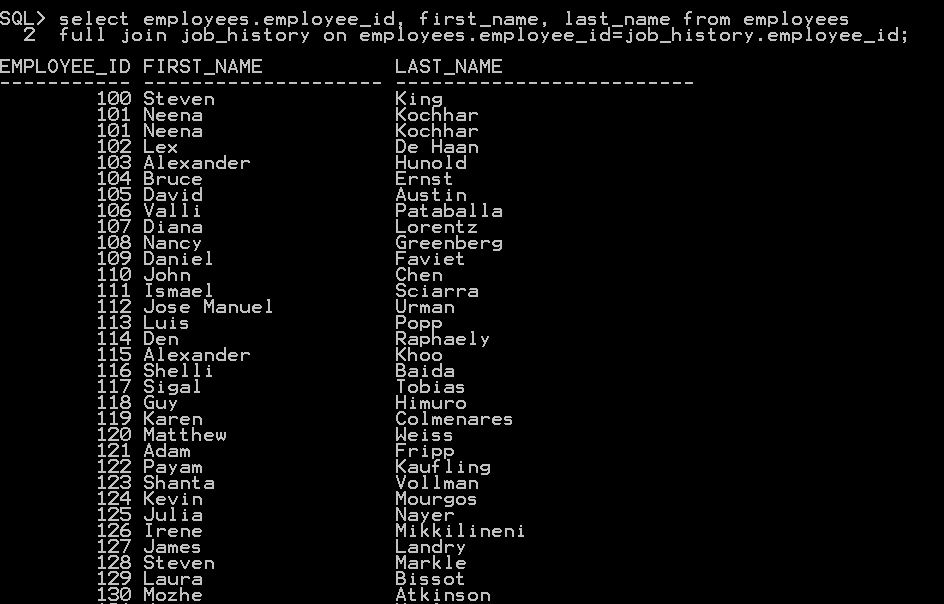
**Left join selects every record on the left table and plus the records that matches on the both tables**

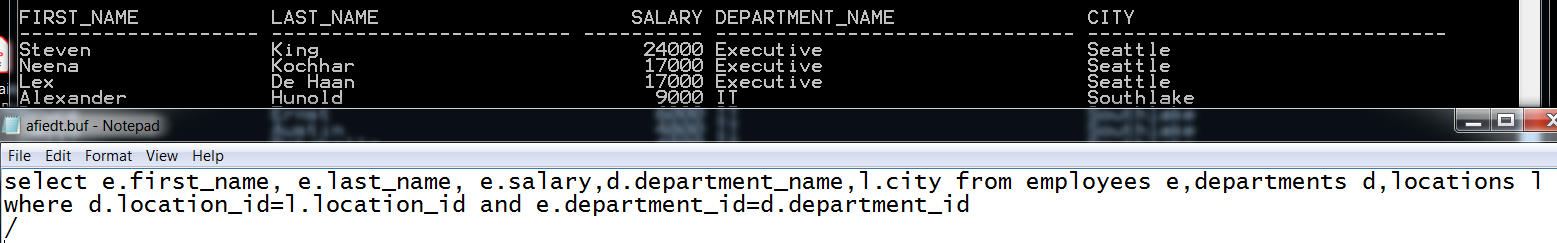


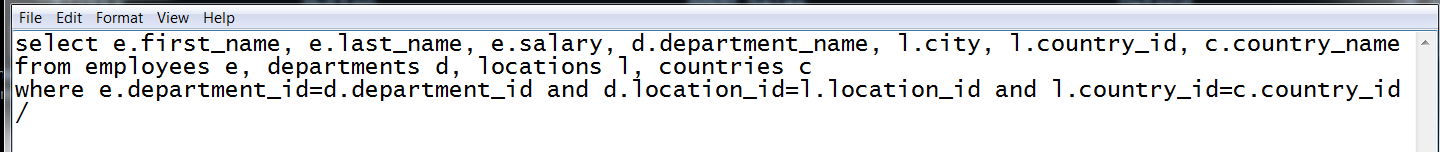
**Right join selects every record on the right table and plus the records that matches on the both tables**



**Full join selects every record on both tables**





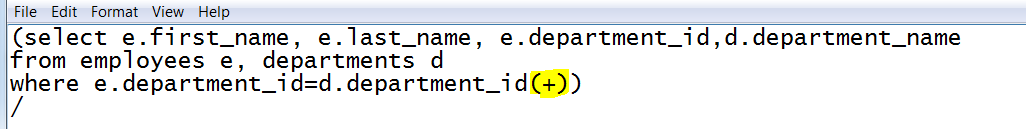


**Join all 7 tables together**

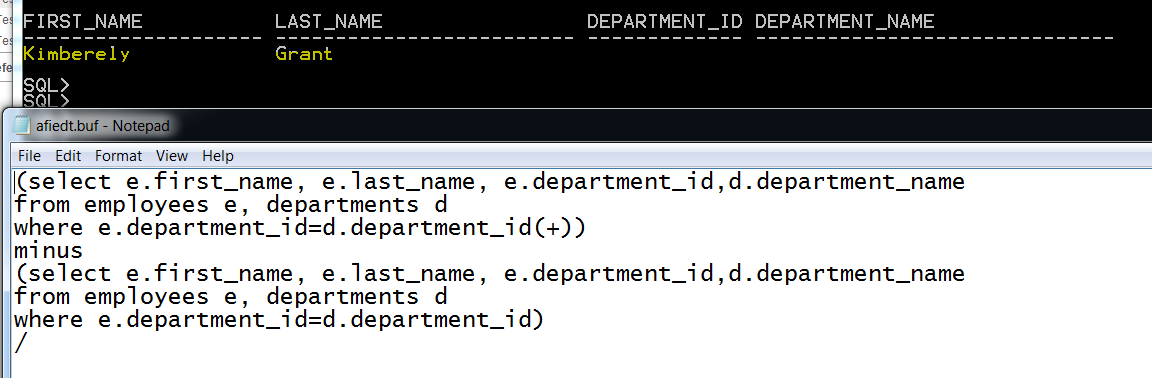


**THIS + plus sign basically says that even tho we want the department\_id matching employees**

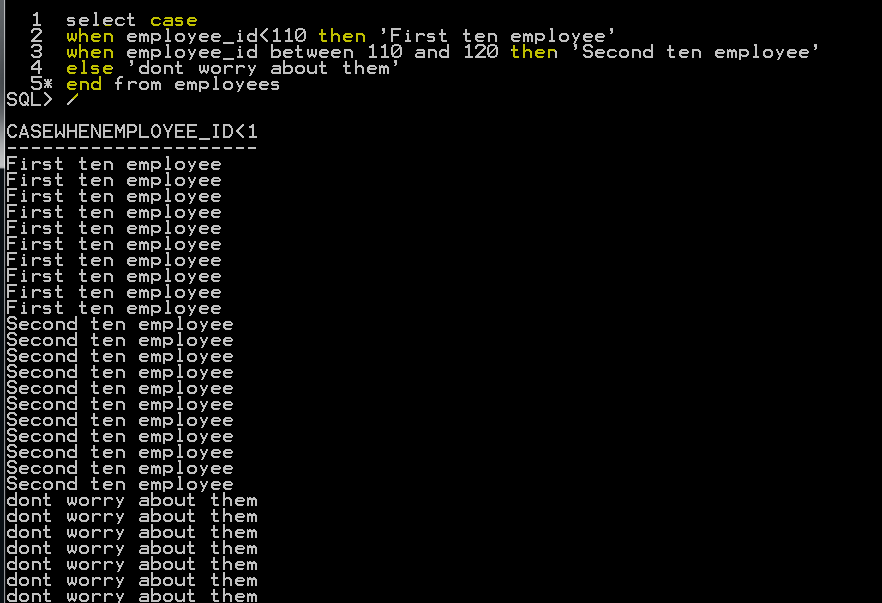
**Still include the ones not matching**



**Then if we wanna find the only employe who doesn’t have department id we use minus keyword and take out the ones have department id**

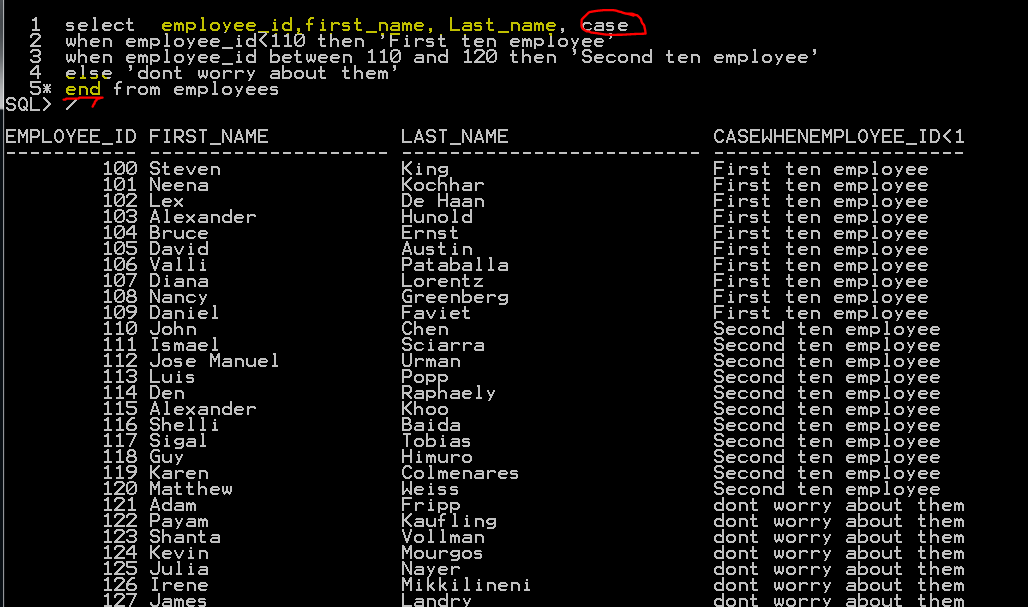


**Case keyword**



**Or**

**Don’t forget the end keyword before from statement**



**DATABASE TESTING**

**Test case**

**1) login to application with url for phpmyadmin**

**2) search 'dbtestuser \_9999MD' to make sure its not in the front end,then create a user account'dbtestuser\_9999MD' then make sure that it is created**

**3) login to the application's database,click on the database forumlab1,**

**click on table name smf\_members,**

**click on SQL,**

**write the query**

**SELECT \* FROM smf\_members**

**WHERE upper(member\_name)**

**='dbtestuser\_9999MD'**

**4) edit email from ddasd@gmail.com**

**to palyanco@gmail.com using front end forumlab.piit.us**

**5) login to database and write a query to verify email has been changed**

**SELECT email\_address FROM smf\_members**

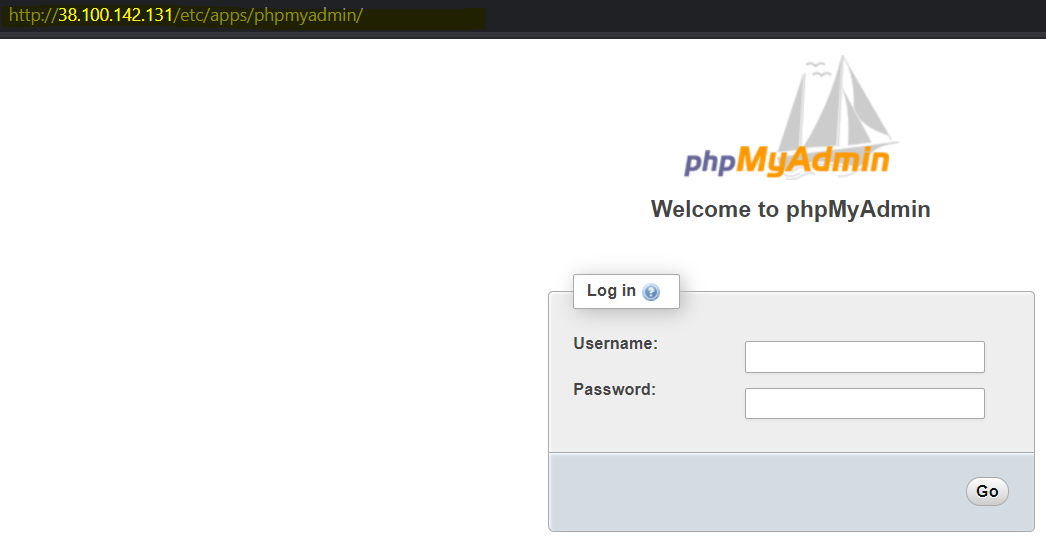
**WHERE upper(member\_name)**

**='dbtestuser\_9999MD'**

**1) login to application with url for phpmyadmin**

**Username: syed**

**Password: abc\*1234**

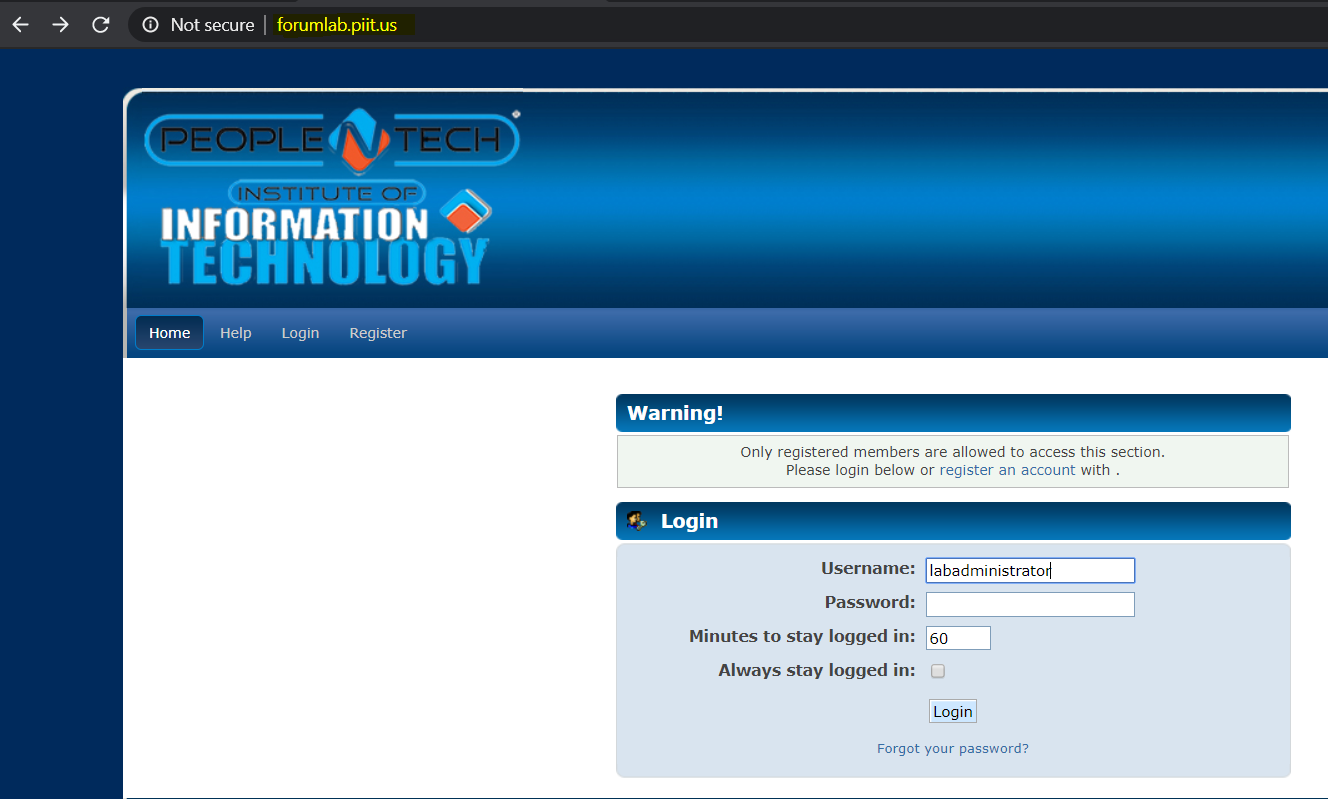


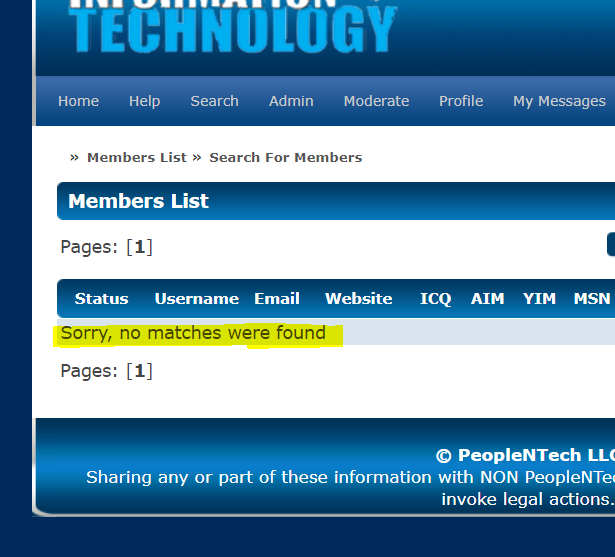
**2) search 'dbtestuser \_9999MD' to make sure its not in the front end,then create a user account'dbtestuser\_9999MD' then make sure that it is created**

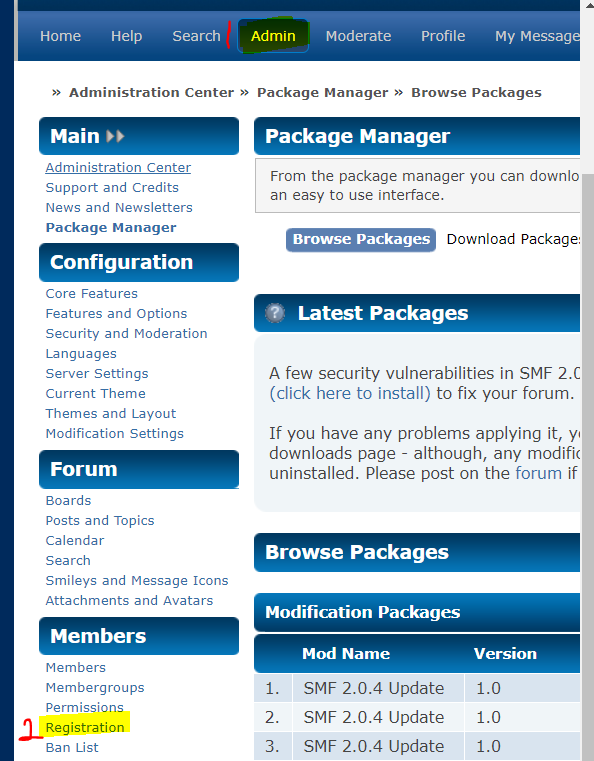
**Testing portal for peopleNtech : https://forumlab.piit.us**

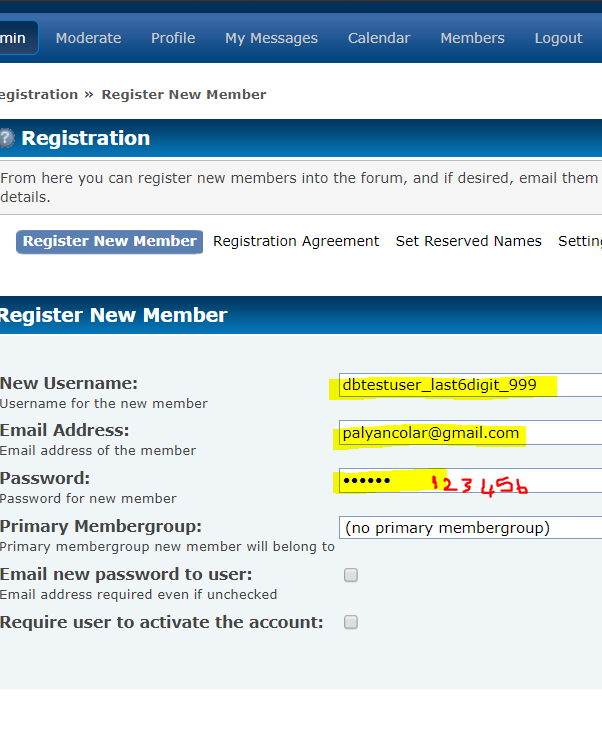
**Username : labadministrator**

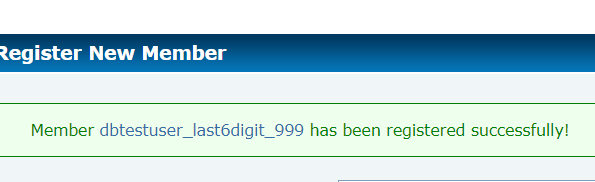
**Password: abc\*1212**











**3) login to the application's database,click on the database forumlab1,**

**click on table name smf\_members,**

**click on SQL,**

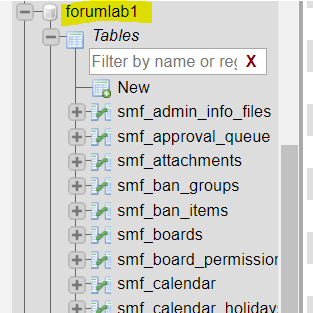
**write the query**

**SELECT \* FROM smf\_members**

**WHERE upper(member\_name)**

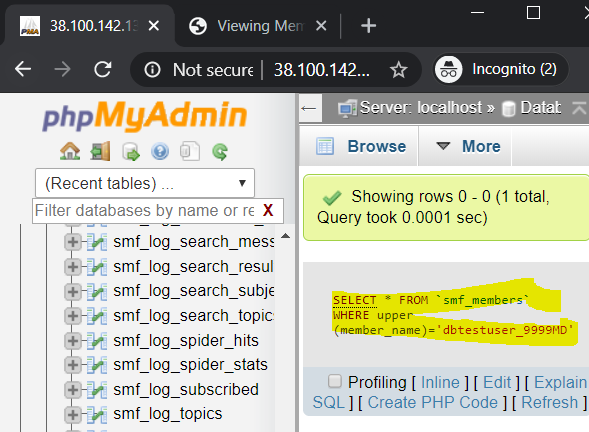
**='dbtestuser\_9999MD'**

**Go to forumlab1 database**



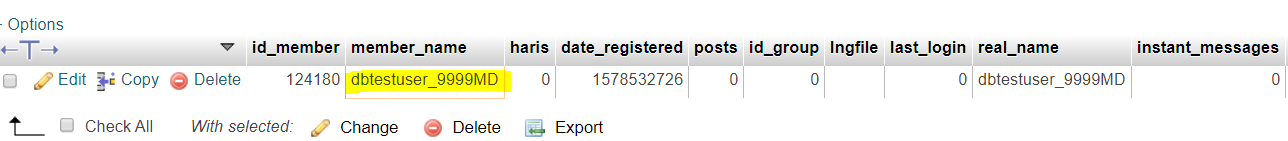
**Select your table to work on**





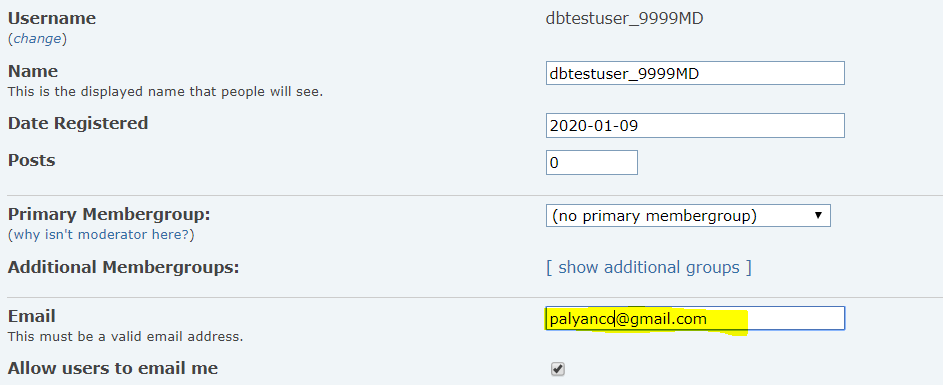


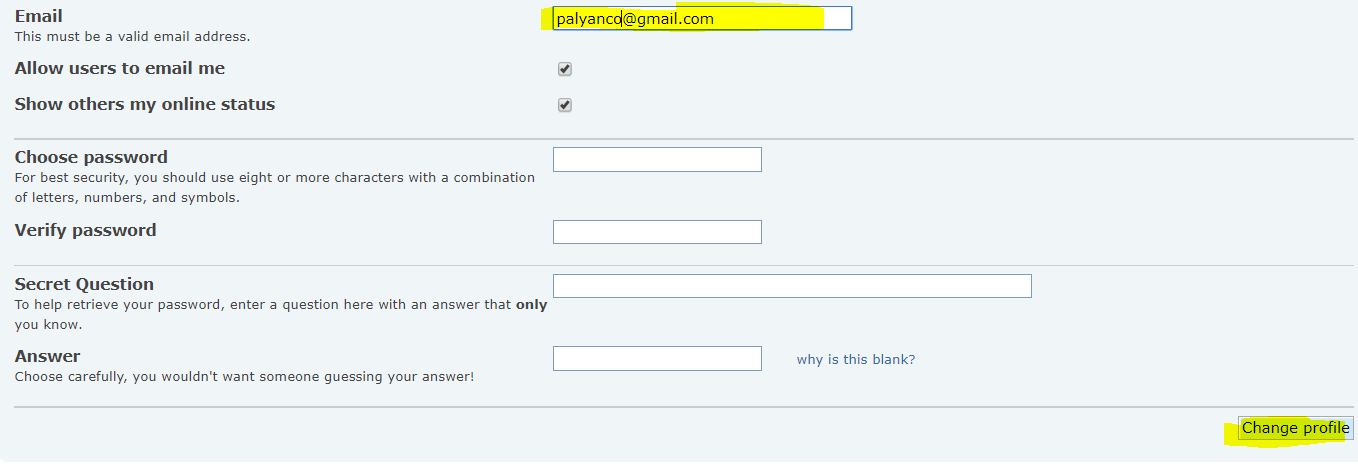
**Find the information about member\_name = dbtester\_999MD**



**4) edit email from ddasd@gmail.com**

**to palyanco@gmail.com using front end forumlab.piit.us**





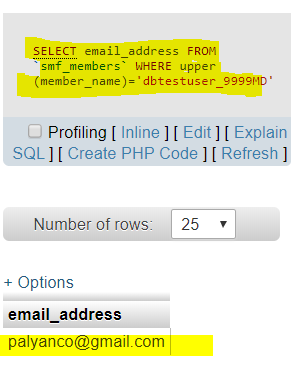


**5) login to database and write a query to verify email has been changed**

**SELECT email\_address FROM smf\_members**

**WHERE upper(member\_name)**

**='dbtestuser\_9999MD'**



**To do more practice from home**

**Call 571-317-8193 to ask questions for database, sql or ALM**

[**http://forumlabadmin.piit.us/apps/phpmyadmin/**](http://forumlabadmin.piit.us/apps/phpmyadmin/)

**username: syed**

**password: abc\*1234**

[**http://forumlab.piit.us/**](http://forumlab.piit.us/)

**username: labadministrator**

**password : abc\*1212**