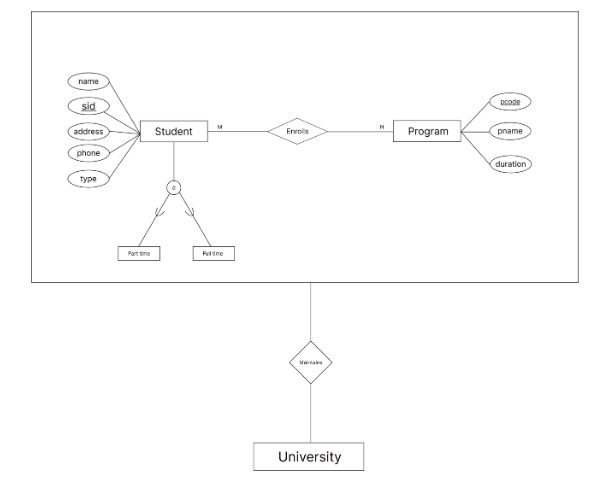
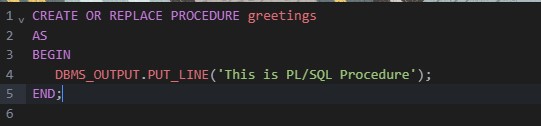
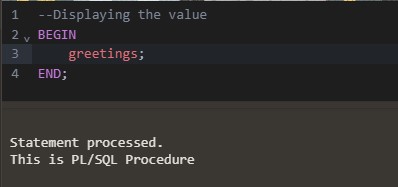
# Develop an EER model for following

A university maintains records of its students and programs in which they have enrolled. It stores student id, name, address, and phone number of student and programs code, program name and dura;on of a program. A student is either a full-;me or a part-;me student (only one of the types). A student can register for many program and programs can have many students.

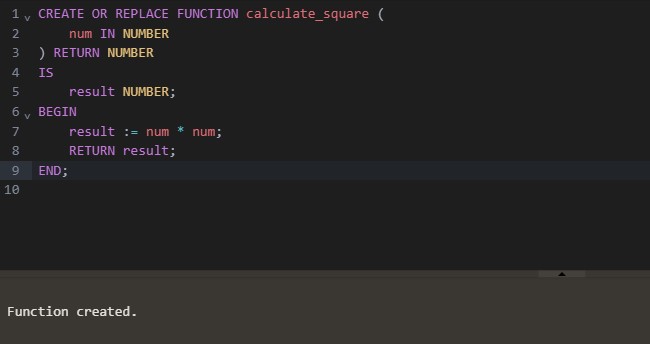
****

# Apply PL/SQL for processing database. Give examples of Procedure, Functions and Triggers

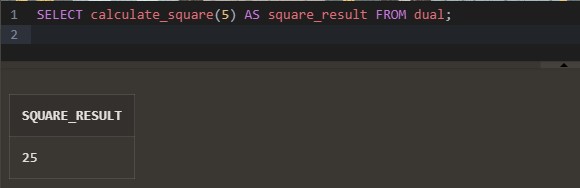
PL/SQL (Procedural Language/Structured Query Language) is a powerful extension of SQL used for writing procedural code in Oracle databases. You can use PL/SQL to create stored procedures, functions, and triggers for processing data in the database. Here are examples of each:  
  
**Procedure**  
  


**Output:**  
  


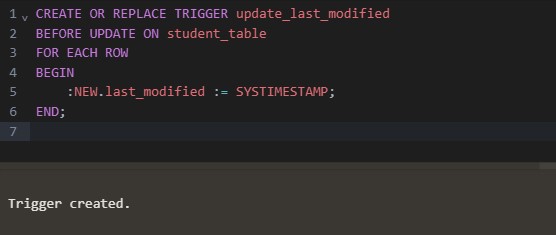
Functions



**Output**



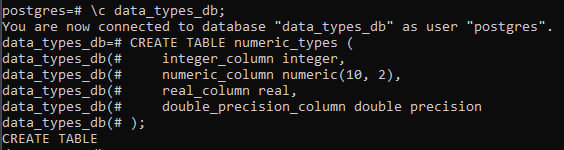
**Triggers**



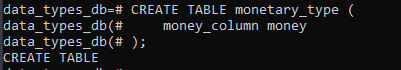
# Illustrate different data types in PostgreSQL

****

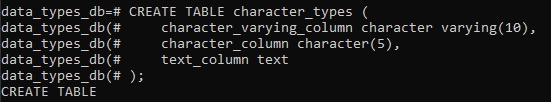
* **Numeric Types**

****

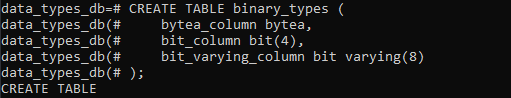
* **Monetary Types**

****

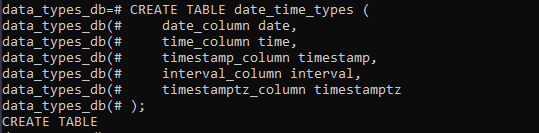
* **Character Types**

****

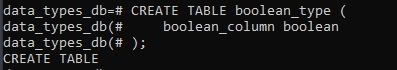
* **Binary Types**

****

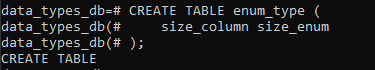
* **Date-Time Types**

****

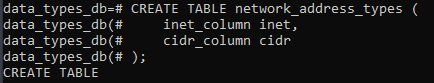
* **Boolean Types**

****

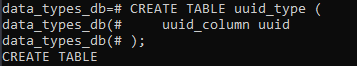
* **Enum Types**

****

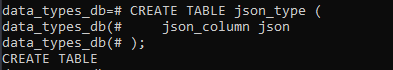
* **Network Address Types**

****

* **UUID Types**

****

* **JSON Types**

****

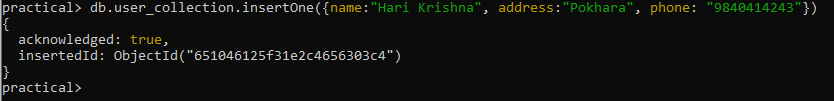
# Apply CRUD operations and retrieve data in NoSQL environment (Use MongoDB or any NoSQL database)

* Using MongoDB for performing CRUD operations

1. **Creating a Database**



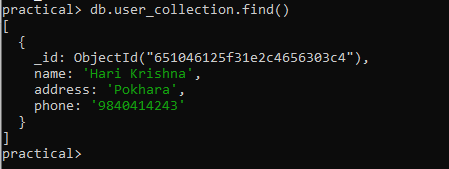
1. **Creating a Collection and adding a Document for one User:**



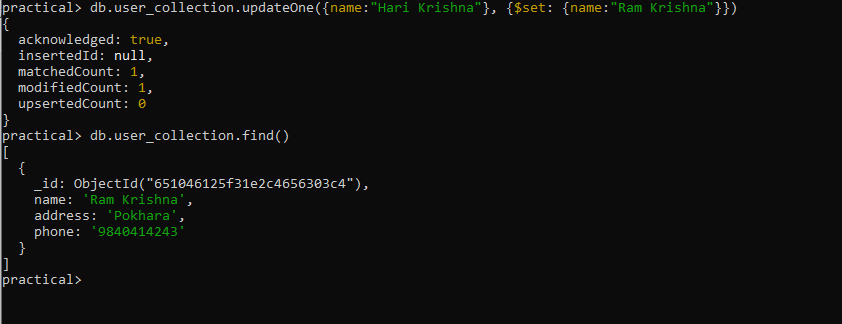
1. **Showing the Current Collections**



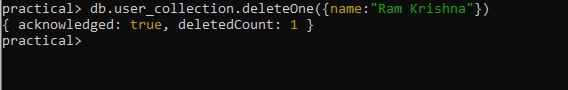
1. **Reading the Current Collection Data**



1. **Updating the Data**



1. **Deleting the Data**



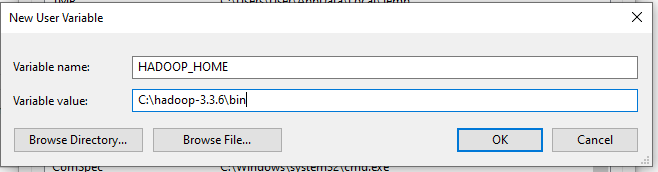
# Understand the basic storage architecture of distributed file systems. Setup Apache Hadoop in your local machine

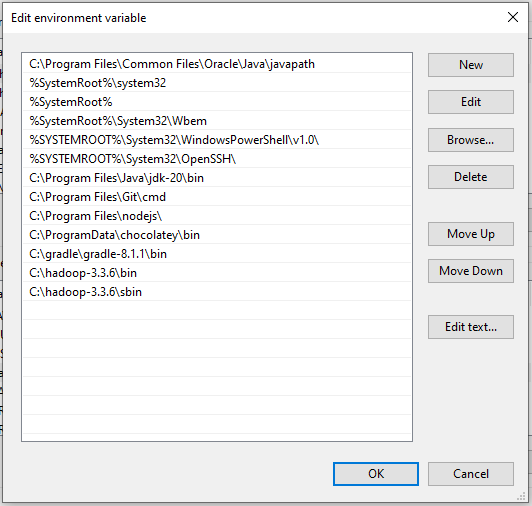
A distributed database file system is designed to manage and store data across multiple nodes or servers in a distributed computing environment. It combines aspects of both distributed databases and file systems to provide scalable and reliable storage for large volumes of data. The basic storage architecture of a distributed database file system typically includes the following components:

1. **Nodes and Server:** These are individual machines or servers that participate in the distributed file system. Each node has its own storage capacity and processing capabilities.
2. **Metadata Server:** In many distributed file systems, there is a central metadata server or a set of metadata servers that store information about the file system's structure and file attributes. This metadata includes details such as file names, directories, access permissions, and file locations.
3. **Data Storage:** The actual data files are distributed across multiple nodes in the system. Data can be broken down into smaller blocks or chunks, which are stored across different nodes for redundancy and load balancing.
4. **Replication and Redundancy:** To ensure data durability and fault tolerance, many distributed file systems replicate data across multiple nodes. This means that copies of the same data are stored on different servers. If one node fails, data can still be retrieved from other replicas.
5. **Access Control and Security:** Distributed file systems have mechanisms for access control and security to protect data from unauthorized access. This includes authentication, authorization, and encryption features.
6. **Load Balancing:** Load balancing techniques may be employed to evenly distribute data and query workloads across nodes to avoid overloading any single server.
7. **Data Recovery and Backup:** Distributed file systems often include mechanisms for data recovery and backup, allowing administrators to restore data in case of data corruption or node failures.
8. **Monitoring and Management Tools:** To monitor the health and performance of the distributed file system, management tools and monitoring systems are essential. These tools help administrators diagnose issues, optimize performance, and ensure the system is operating as expected.

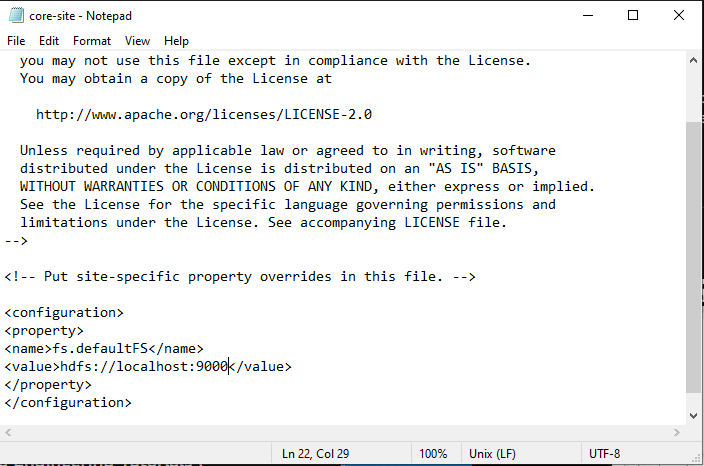
**Apache Hadoop Setup:**

1. First configure the Hadoop env file by setting up the Java path.
2. Add the Hadoop path to Environment Variable.

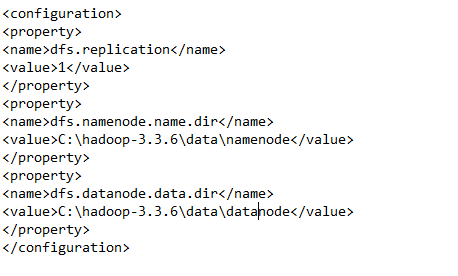




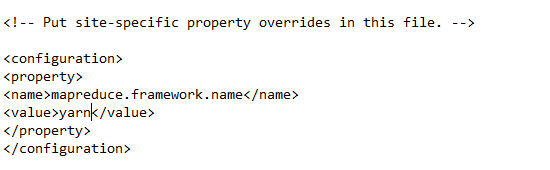
1. Configure core-site document



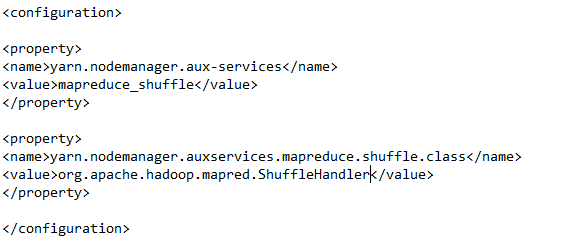
1. Now configure httpfs-site file or hdfs-site, either one will work.
   1. Create one folder called data and inside the create two more folder namenode and datanode simultaneously.
   2. Add the folder location to the configurable file httpfs-site or hdfs-site



1. Now configure the mapred-site file



1. We need to configure yarn-site file



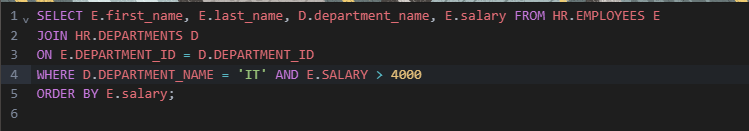
1. Format the hdfs namenode



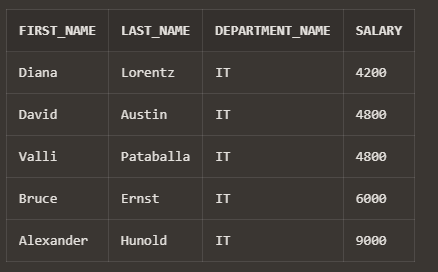
1. Start namenode and datanode by using command **start-dfs.cmd**
2. Start yarn by using command **start-yarn.cmd**

# Distributed Horizontal Fragmentation

Divide Employees table with IT and Sales department in different fragments

****

Output:

****

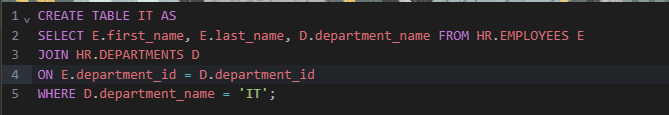
Find all employees who works on department located at country 'United States of America' and have salary>4000

Output:

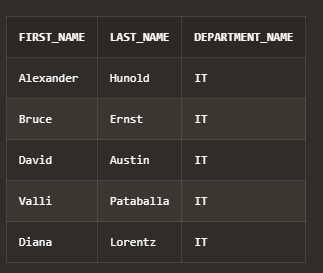
****

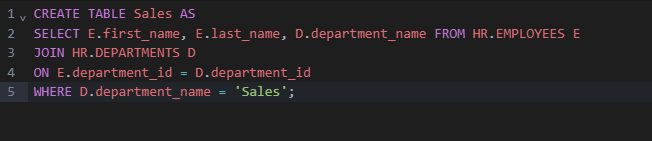
# Distributed Vertical Fragmentation

Divide Employees table with IT and Sales department in different fragments

****

Output:



****

Output:



# Using prolog, perform the following tasks for the Family

# Relations:

1. Define facts to represent parent-child relationships in the family tree.

parent(rajesh, sunita).

parent(rajesh, arjun).

parent(sunita, rina).

parent(sunita, amit).

parent(arjun, neha).

parent(arjun, rohan).

parent(rina, vishal).

parent(amit, preeti).

parent(neha, karan).

parent(rohan,sneha).  
  
female(sunita).

female(rina).

female(neha).

female(preeti).

female(sneha).

male(rajesh).

male(arjun).

male(amit).

male(vishal).

male(karan).

male(rohan).

1. Define facts to represent parent-child relationships in the family tree.

% ancestor relationship

ancestor(X, Y) :- parent(X, Y).

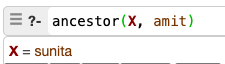
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).

% descendant relationship

descendant(X, Y) :- parent(Y, X).

descendant(X, Y) :- parent(Z, X), descendant(Z, Y).

1. Write queries to find all ancestors and descendants of a given person.

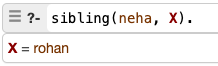




1. Define rules to represent sibling relationships.

sibling(X,Y):-parent(P,X),parent(P,Y),X\=Y.

1. Write queries to find all siblings of a person.



1. Define rules to represent the aunt and uncle relationships.

% Define aunt relationship

aunt(Aunt, NieceNephew) :-

parent(Parent, NieceNephew),

sibling(Aunt, Parent),

female(Aunt).

% Define uncle relationship

uncle(Uncle, NieceNephew) :-

parent(Parent, NieceNephew),

sibling(Uncle, Parent),

male(Uncle).

1. Write queries to find aunts and uncles of a person.





1. Define rules to represent the aunt and uncle relationships.

% Define cousin relationship

cousin(Cousin, Person) :-

parent(Parent1, Cousin),

parent(Parent2, Person),

sibling(Parent1, Parent2),

\+ sibling(Cousin, Person),

Cousin \= Person.

1. Write a query to find all cousins of a given person.

