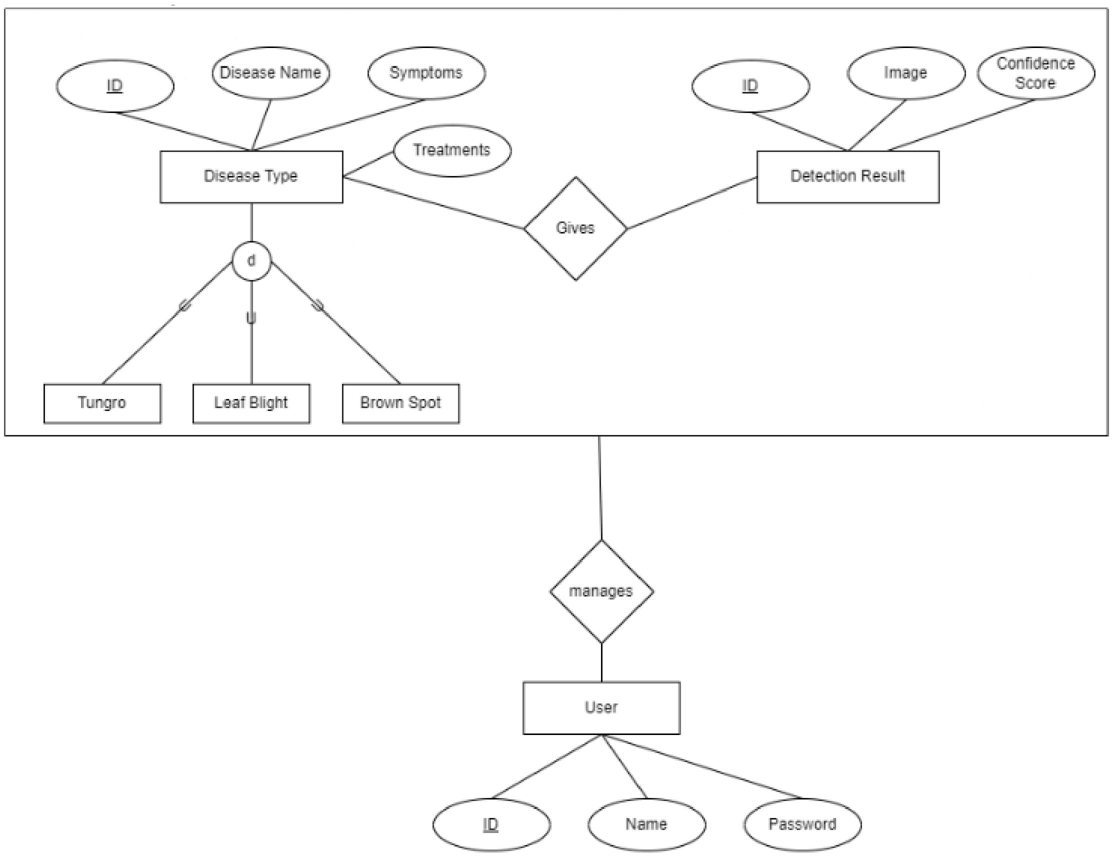
1. Develop an EER model for following.

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 duration of a program. A student is either a full time or a part time student (only one of the types). A student can register for many programs and programs can have many students.



2. Apply PL/SQL for processing databases. Give examples of Procedure, Functions and Triggers.

PL/SQL(procedural constructs to the SQL statements)

= The PL/SQL language is capable of performing various different operations at any given time.

SQL(Structured Query Language)

= The SQL language is capable of executing only a single operation at any given time.

PL/SQL PROCEDURES & Functions

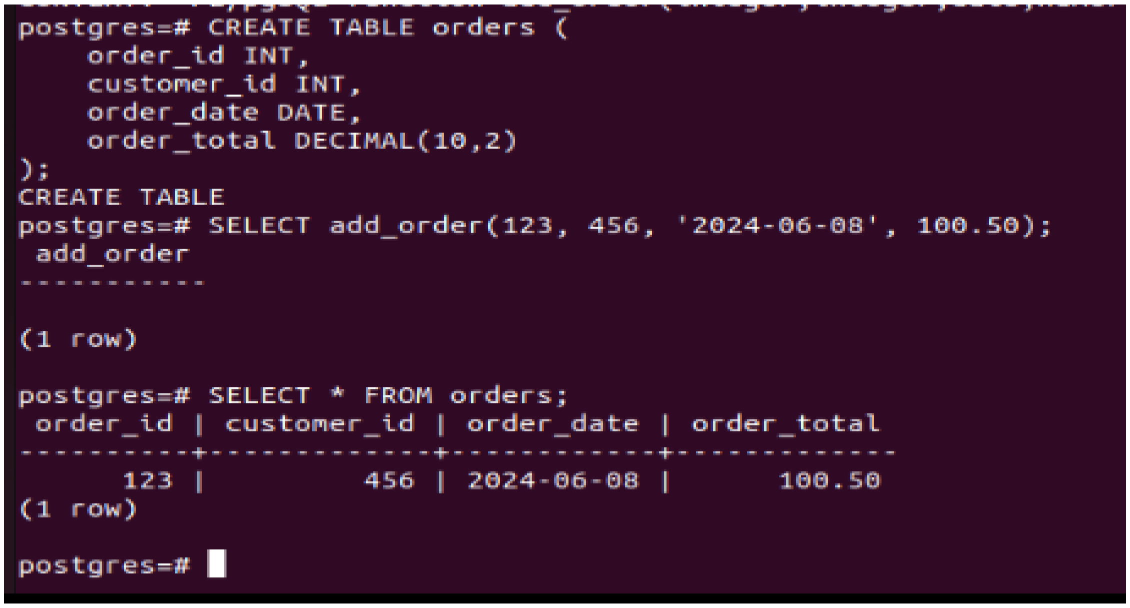
= Procedures and functions! Procedures act like mini-programs, automating tasks and keeping your code clean. Functions are value-driven workhorses, performing calculations and returning results.

PL/SQL TRIGGERS

=PL/SQL triggers are special procedures that automatically execute (or "trigger") in response to certain events, such as data changes.

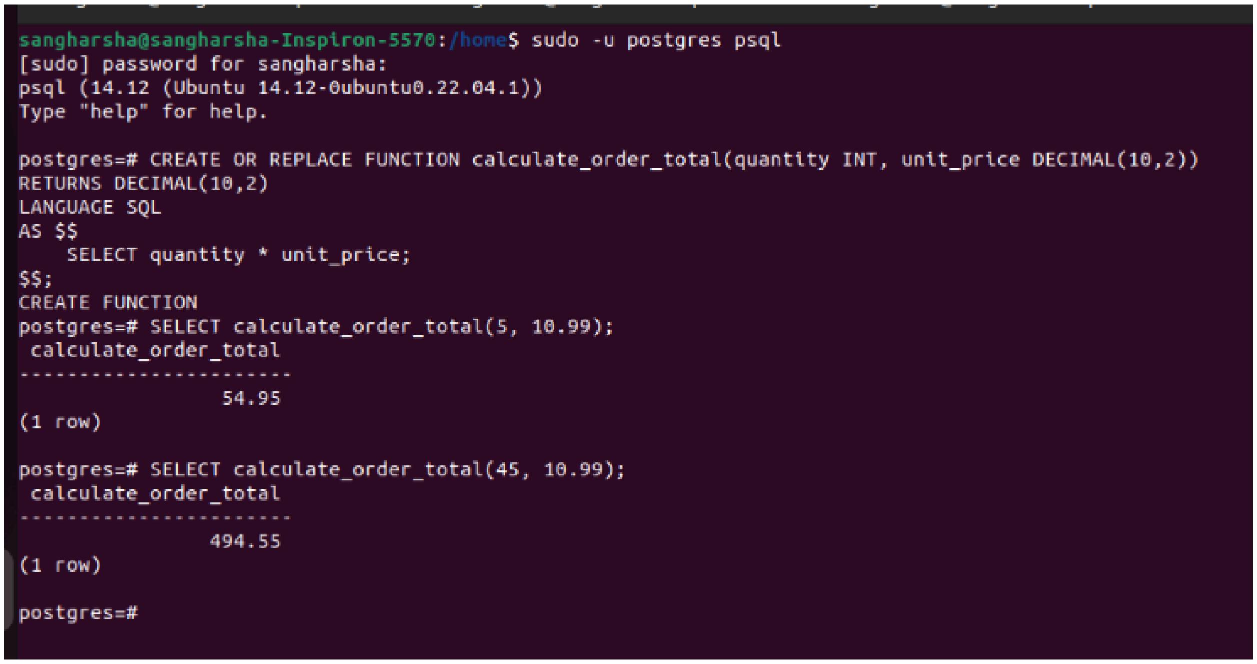
Examples:

Procedures:



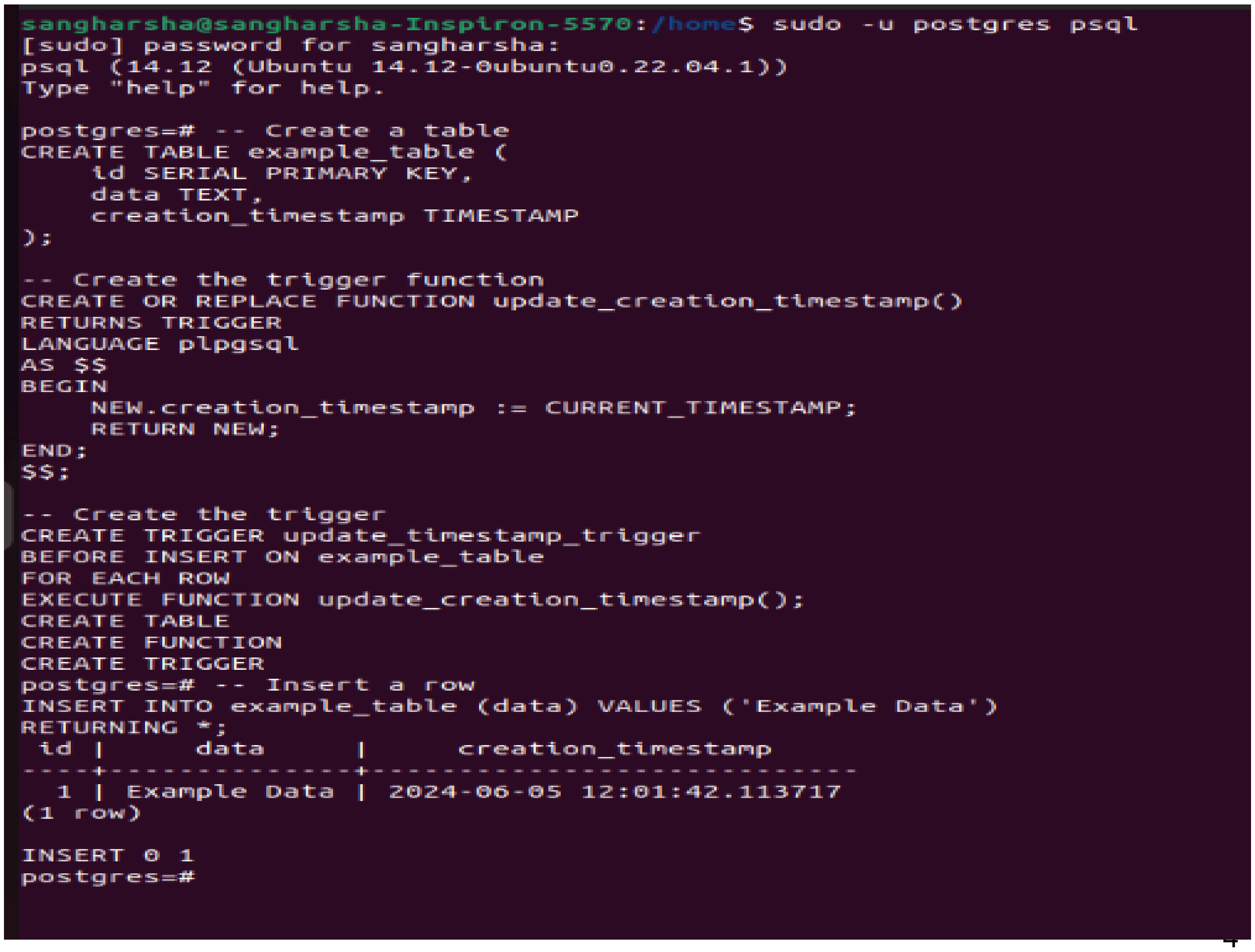
```
postgres=# CREATE TABLE orders (
    order_id INT,
    customer_id INT,
    order_date DATE,
    order_total DECIMAL(10,2)
);
CREATE TABLE
postgres=# SELECT add_order(123, 456, '2024-06-08', 100.50);
 add_order
-----------

(1 row)

postgres=# SELECT * FROM orders;
 order_id | customer_id | order_date | order_total
----------+-------------+------------+-------------
      123 |         456 | 2024-06-08 |      100.50
(1 row)

postgres=# 
```

Functions:



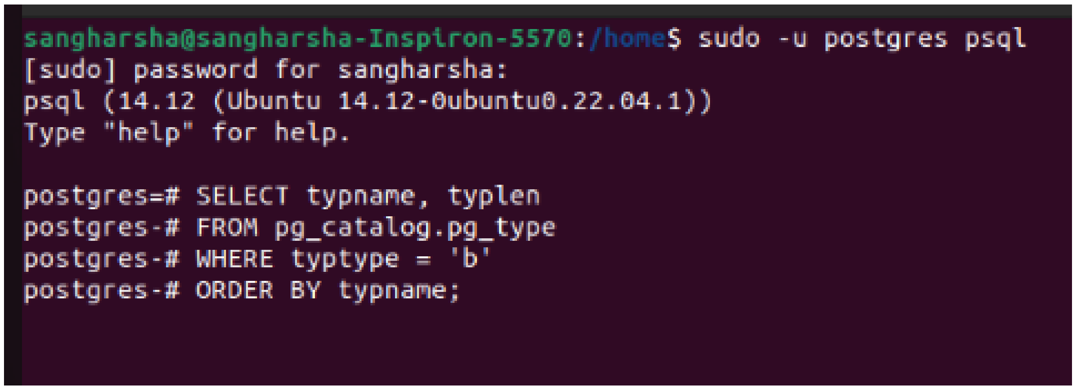
```
sangharsha@sangharsha-Inspiron-5570:/home$ sudo -u postgres psql
[sudo] password for sangharsha:
psql (14.12 (Ubuntu 14.12-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# CREATE OR REPLACE FUNCTION calculate_order_total(quantity INT, unit_price DECIMAL(10,2))
RETURNS DECIMAL(10,2)
LANGUAGE SQL
AS $$
    SELECT quantity * unit_price;
$$;
CREATE FUNCTION
postgres=# SELECT calculate_order_total(5, 10.99);
 calculate_order_total
-----------------------
                 54.95
(1 row)

postgres=# SELECT calculate_order_total(45, 10.99);
 calculate_order_total
-----------------------
                494.55
(1 row)

postgres=#
```

Trigger:

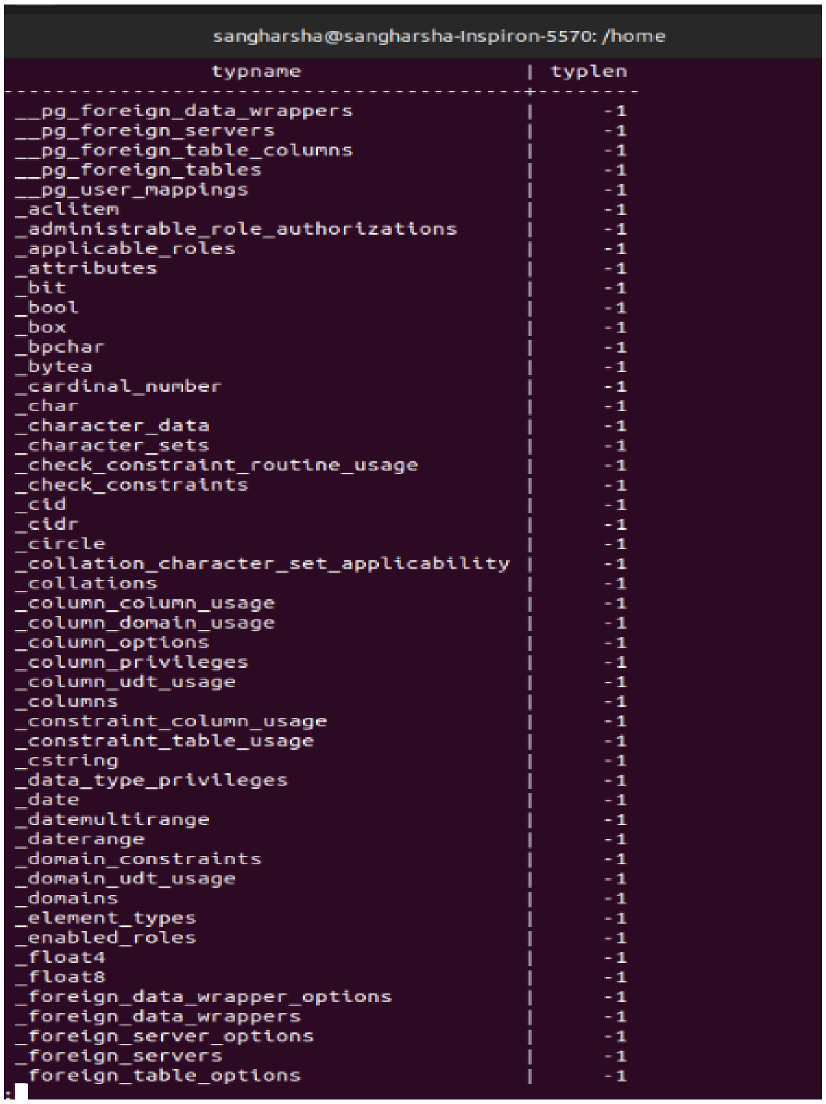


```
sangharsha@sangharsha-Inspiron-5570:/home$ sudo -u postgres psql
[sudo] password for sangharsha:
psql (14.12 (Ubuntu 14.12-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# -- Create a table
CREATE TABLE example_table (
    id SERIAL PRIMARY KEY,
    data TEXT,
    creation_timestamp TIMESTAMP
);

-- Create the trigger function
CREATE OR REPLACE FUNCTION update_creation_timestamp()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    NEW.creation_timestamp := CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$;

-- Create the trigger
CREATE TRIGGER update_timestamp_trigger
BEFORE INSERT ON example_table
FOR EACH ROW
EXECUTE FUNCTION update_creation_timestamp();
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
postgres=# -- Insert a row
INSERT INTO example_table (data) VALUES ('Example Data')
RETURNING *;
 id |     data     |     creation_timestamp
----+--------------+----------------------------
  1 | Example Data | 2024-06-05 12:01:42.113717
(1 row)

INSERT 0 1
postgres=#
```

3. Illustrate different data types in PostgreSQL.



```
sangharsha@sangharsha-Inspiron-5570:/home$ sudo -u postgres psql
[sudo] password for sangharsha:
psql (14.12 (Ubuntu 14.12-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# SELECT typname, typlen
postgres-# FROM pg_catalog.pg_type
postgres-# WHERE typtype = 'b'
postgres-# ORDER BY typname;
```



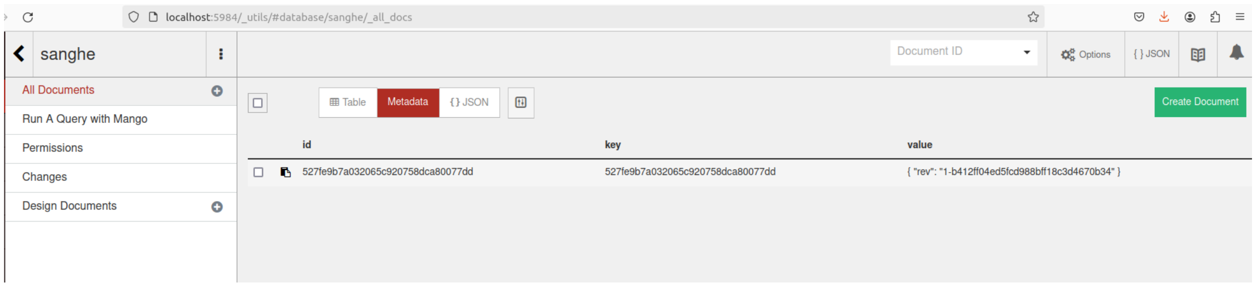
sangharsha@sangharsha-Inspiron-5570: /home

```
                typname                 | typlen
----------------------------------------+--------
 __pg_foreign_data_wrappers             |     -1
 __pg_foreign_servers                   |     -1
 __pg_foreign_table_columns             |     -1
 __pg_foreign_tables                    |     -1
 __pg_user_mappings                     |     -1
 _aclitem                               |     -1
 _administrable_role_authorizations     |     -1
 _applicable_roles                      |     -1
 _attributes                            |     -1
 _bit                                   |     -1
 _bool                                  |     -1
 _box                                   |     -1
 _bpchar                                |     -1
 _bytea                                 |     -1
 _cardinal_number                       |     -1
 _char                                  |     -1
 _character_data                        |     -1
 _character_sets                        |     -1
 _check_constraint_routine_usage        |     -1
 _check_constraints                     |     -1
 _cid                                   |     -1
 _cidr                                  |     -1
 _circle                                |     -1
 _collation_character_set_applicability |     -1
 _collations                            |     -1
 _column_column_usage                   |     -1
 _column_domain_usage                   |     -1
 _column_options                        |     -1
 _column_privileges                     |     -1
 _column_udt_usage                      |     -1
 _columns                               |     -1
 _constraint_column_usage               |     -1
 _constraint_table_usage                |     -1
 _cstring                               |     -1
 _data_type_privileges                  |     -1
 _date                                  |     -1
 _datemultirange                        |     -1
 _daterange                             |     -1
 _domain_constraints                    |     -1
 _domain_udt_usage                      |     -1
 _domains                               |     -1
 _element_types                         |     -1
 _enabled_roles                         |     -1
 _float4                                |     -1
 _float8                                |     -1
 _foreign_data_wrapper_options          |     -1
 _foreign_data_wrappers                 |     -1
 _foreign_server_options                |     -1
 _foreign_servers                       |     -1
 _foreign_table_options                 |     -1
:
```

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

=> Here I use COUCHDB.

Couchdb=Apache CouchDB (link resides outside ibm.com) is an open source NoSQL document database that collects and stores data in JSON-based document formats.



localhost:5984/_utils/#database/sanghe/_all_docs

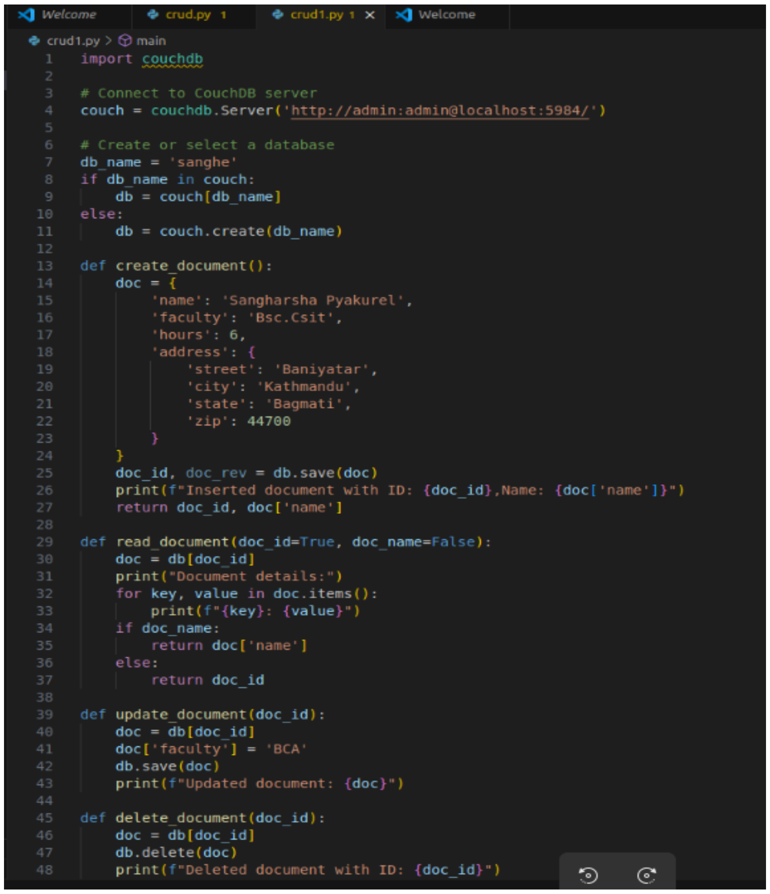
sanghe

- All Documents
- Run A Query with Mango
- Permissions
- Changes
- Design Documents

id	key	value
527fe9b7a032065c920758dca80077dd	527fe9b7a032065c920758dca80077dd	{ "rev": "1-b412ff04ed5fcd988bff18c3d4670b34" }

Fig: DashBoard Of CouchDB.

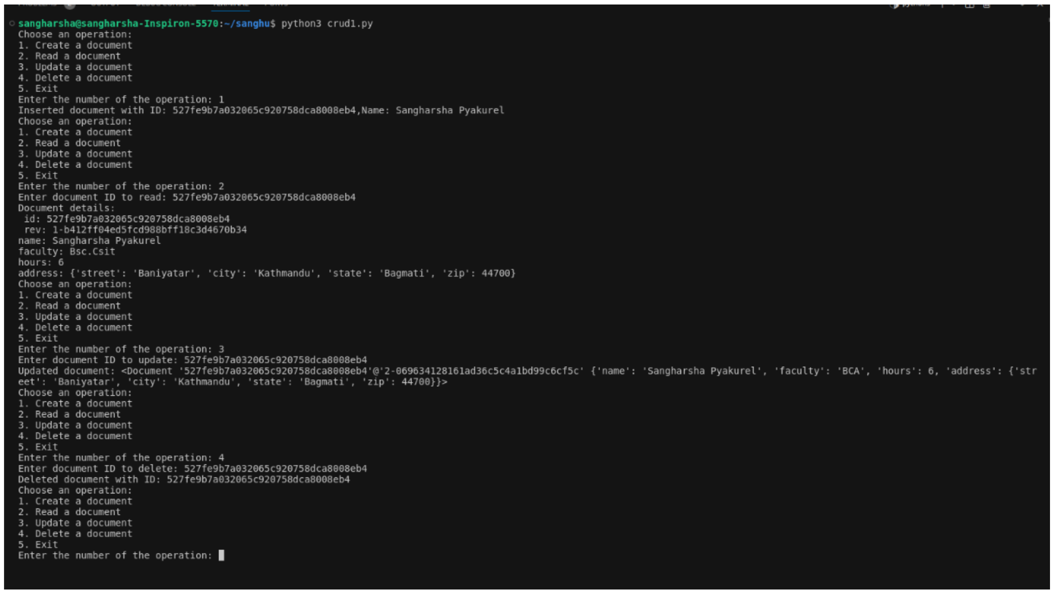
In this lab4. I inject the localhost by using Python Script and I am doing CRUD using python script on COUCHDB.



```python
import couchdb

# Connect to CouchDB server
couch = couchdb.Server('http://admin:admin@localhost:5984/')

# Create or select a database
db_name = 'sanghe'
if db_name in couch:
    db = couch[db_name]
else:
    db = couch.create(db_name)

def create_document():
    doc = {
        'name': 'Sangharsha Pyakurel',
        'faculty': 'Bsc.Csit',
        'hours': 6,
        'address': {
            'street': 'Baniyatar',
            'city': 'Kathmandu',
            'state': 'Bagmati',
            'zip': 44700
        }
    }
    doc_id, doc_rev = db.save(doc)
    print(f"Inserted document with ID: {doc_id},Name: {doc['name']}")
    return doc_id, doc['name']

def read_document(doc_id=True, doc_name=False):
    doc = db[doc_id]
    print("Document details:")
    for key, value in doc.items():
        print(f"{key}: {value}")
    if doc_name:
        return doc['name']
    else:
        return doc_id

def update_document(doc_id):
    doc = db[doc_id]
    doc['faculty'] = 'BCA'
    db.save(doc)
    print(f"Updated document: {doc}")

def delete_document(doc_id):
    doc = db[doc_id]
    db.delete(doc)
    print(f"Deleted document with ID: {doc_id}")
```

.

OUTPUT:



```
○ sangharsha@sangharsha-Inspiron-5570:~/sanghu$ python3 crud1.py
Choose an operation:
1. Create a document
2. Read a document
3. Update a document
4. Delete a document
5. Exit
Enter the number of the operation: 1
Inserted document with ID: 527fe9b7a032065c920758dca8008eb4,Name: Sangharsha Pyakurel
Choose an operation:
1. Create a document
2. Read a document
3. Update a document
4. Delete a document
5. Exit
Enter the number of the operation: 2
Enter document ID to read: 527fe9b7a032065c920758dca8008eb4
Document details:
 id: 527fe9b7a032065c920758dca8008eb4
 rev: 1-b412ff04ed5fcd988bff18c3d4670b34
name: Sangharsha Pyakurel
faculty: Bsc.Csit
hours: 6
address: {'street': 'Baniyatar', 'city': 'Kathmandu', 'state': 'Bagmati', 'zip': 44700}
Choose an operation:
1. Create a document
2. Read a document
3. Update a document
4. Delete a document
5. Exit
Enter the number of the operation: 3
Enter document ID to update: 527fe9b7a032065c920758dca8008eb4
Updated document: <Document '527fe9b7a032065c920758dca8008eb4'@'2-069634128161ad36c5c4a1bd99c6cf5c' {'name': 'Sangharsha Pyakurel', 'faculty': 'BCA', 'hours': 6, 'address': {'str
eet': 'Baniyatar', 'city': 'Kathmandu', 'state': 'Bagmati', 'zip': 44700}}>
Choose an operation:
1. Create a document
2. Read a document
3. Update a document
4. Delete a document
5. Exit
Enter the number of the operation: 4
Enter document ID to delete: 527fe9b7a032065c920758dca8008eb4
Deleted document with ID: 527fe9b7a032065c920758dca8008eb4
Choose an operation:
1. Create a document
2. Read a document
3. Update a document
4. Delete a document
5. Exit
Enter the number of the operation: █
```

5. Illustrate Planning and Optimizing Performance (Use PostgreSQL).

=> Managing the performance of a PostgreSQL database with millions of transactions every week requires careful planning, monitoring, and optimization. Here are some strategies that can help improve the performance of your database:

- **Indexing:** Proper indexing can significantly speed up the performance of queries that are frequently used in your app. Indexes allow the database to quickly find the relevant data without scanning the entire table. You can create indexes on columns that are frequently used in WHERE clauses or JOIN statements.

```
CREATE INDEX idx_user_id ON transactions (user_id);
```

- **Query optimization:** Review the queries used in your app to identify any inefficient or slow-performing queries. Use EXPLAIN and ANALYZE to understand the query plan and performance.

```
SELECT user_id, SUM(amount)
FROM transactions
WHERE transaction_date >= '2022-01-01'
GROUP BY user_id;
```

- **Partitioning:** If your app generates millions of transactions every week, consider partitioning the data into smaller and more manageable chunks.

```
CREATE TABLE transactions (
    transaction_id SERIAL PRIMARY KEY,
    transaction_date DATE,
    user_id INTEGER,
    amount DECIMAL(10, 2),
    ...
)
PARTITION BY RANGE (transaction_date);
```

- **Scaling:** As the volume of data grows, scaling your database can help maintain performance.

- **Monitoring:** Regularly monitor your database to identify any performance bottlenecks or issues.

- **Archiving:** Consider archiving older data to reduce the size of the database and improve performance.

- **Reporting:** For reports that require data from the last year, consider pre-aggregating the data into a separate table or using a reporting tool like Tableau or Looker.

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

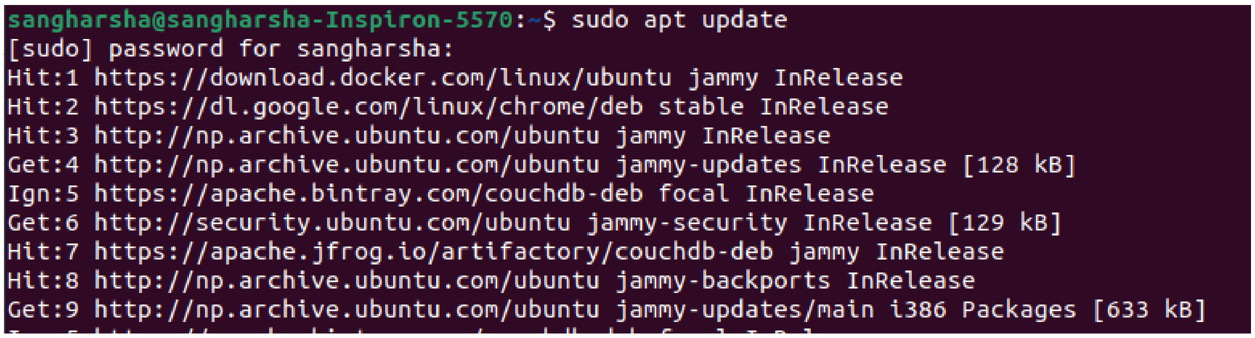
Distributed file systems are designed to store and manage large amounts of data across multiple nodes in a network. They distribute data across different servers to enhance performance, reliability, and scalability. Key components typically include:

- NameNode: Manages the file system metadata and controls access to files by clients.
- DataNodes: Store and retrieve blocks of data.
- Client: Requests service from NameNode and DataNodes.

Hadoop's Distributed File System (HDFS) is a prime example, offering high throughput access to application data and is suitable for applications that require high aggregate bandwidth across many nodes.

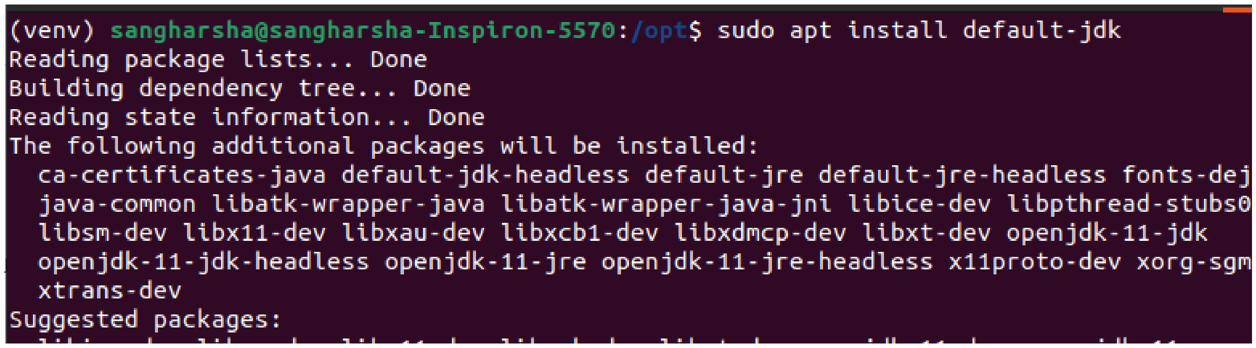
Installing Hadoop on Ubuntu 22.04

Step 1: Update System Repositories



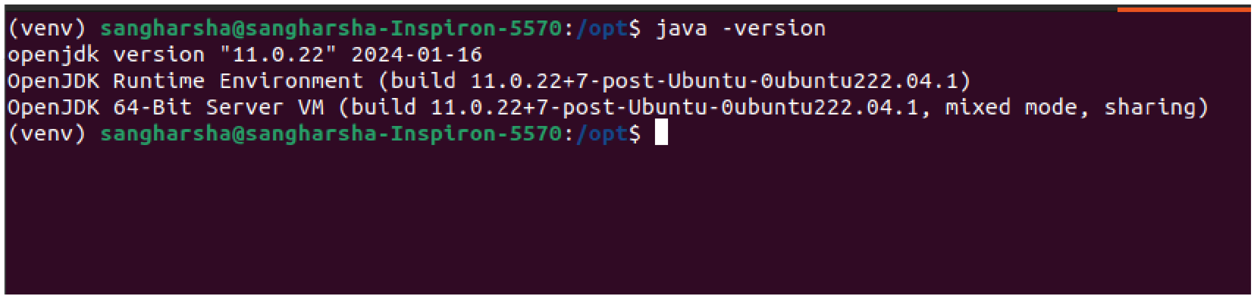
```
sangharsha@sangharsha-Inspiron-5570:~$ sudo apt update
[sudo] password for sangharsha:
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://np.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://np.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Ign:5 https://apache.bintray.com/couchdb-deb focal InRelease
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:7 https://apache.jfrog.io/artifactory/couchdb-deb jammy InRelease
Hit:8 http://np.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:9 http://np.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [633 kB]
```

Step 2:Installing Java Development Kit (JDK)



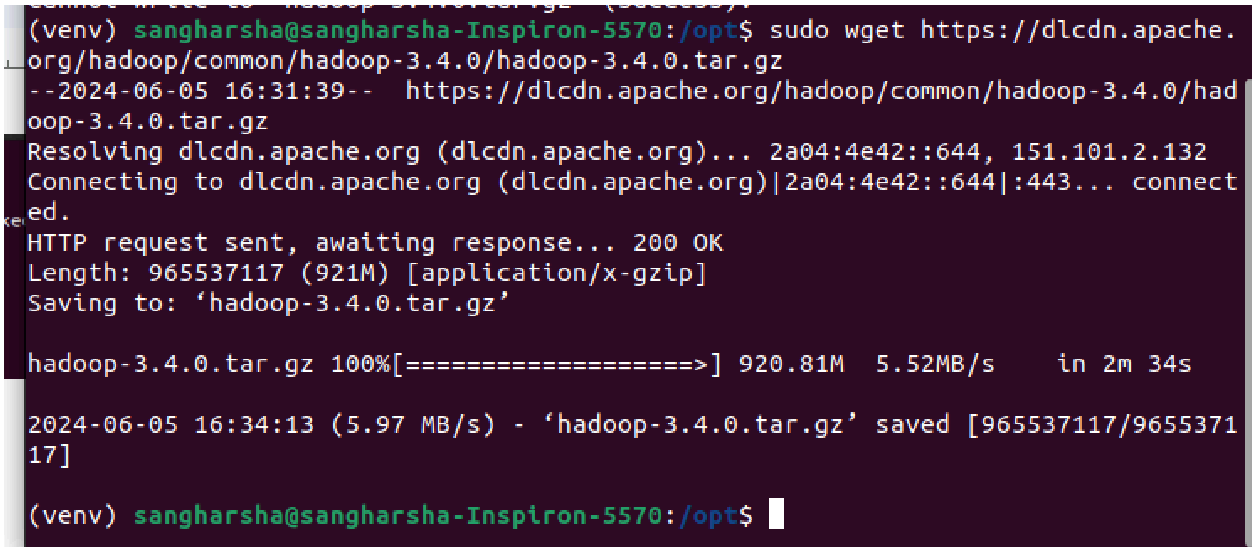
```
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jdk-headless default-jre default-jre-headless fonts-dej
  java-common libatk-wrapper-java libatk-wrapper-java-jni libice-dev libpthread-stubs0
  libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11proto-dev xorg-sgm
  xtrans-dev
Suggested packages:
```

Step 3: Check Java Version



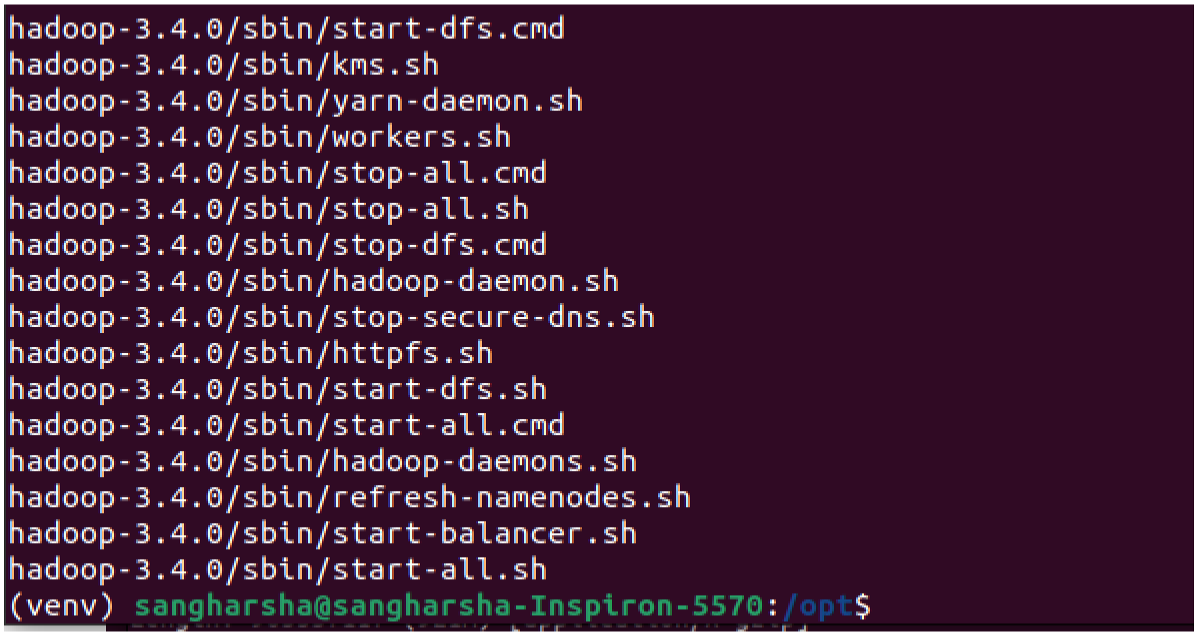
```
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ java -version
openjdk version "11.0.22" 2024-01-16
OpenJDK Runtime Environment (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1)
OpenJDK 64-Bit Server VM (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1, mixed mode, sharing)
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$
```

Step 3: Downloading Hadoop



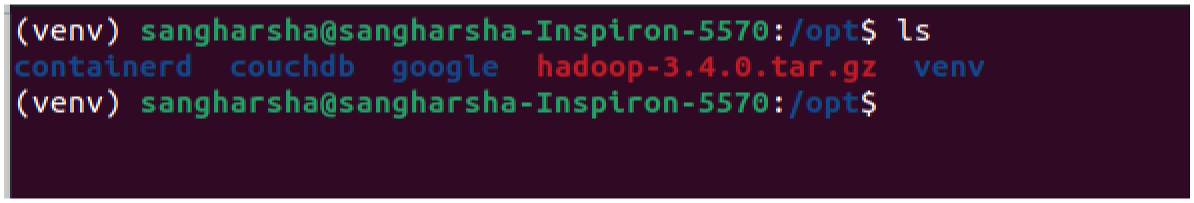
```
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ sudo wget https://dlcdn.apache.
org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz
--2024-06-05 16:31:39--  https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/had
oop-3.4.0.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 2a04:4e42::644, 151.101.2.132
Connecting to dlcdn.apache.org (dlcdn.apache.org)|2a04:4e42::644|:443... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 965537117 (921M) [application/x-gzip]
Saving to: ‘hadoop-3.4.0.tar.gz’

hadoop-3.4.0.tar.gz 100%[===================>] 920.81M  5.52MB/s    in 2m 34s

2024-06-05 16:34:13 (5.97 MB/s) - ‘hadoop-3.4.0.tar.gz’ saved [965537117/9655371
17]

(venv) sangharsha@sangharsha-Inspiron-5570:/opt$
```

Step 4: Extracting the Content



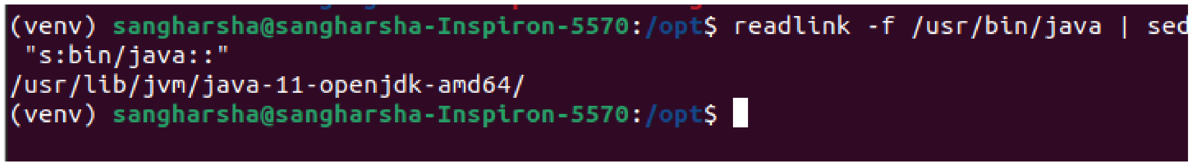
```
hadoop-3.4.0/sbin/start-dfs.cmd
hadoop-3.4.0/sbin/kms.sh
hadoop-3.4.0/sbin/yarn-daemon.sh
hadoop-3.4.0/sbin/workers.sh
hadoop-3.4.0/sbin/stop-all.cmd
hadoop-3.4.0/sbin/stop-all.sh
hadoop-3.4.0/sbin/stop-dfs.cmd
hadoop-3.4.0/sbin/hadoop-daemon.sh
hadoop-3.4.0/sbin/stop-secure-dns.sh
hadoop-3.4.0/sbin/httpfs.sh
hadoop-3.4.0/sbin/start-dfs.sh
hadoop-3.4.0/sbin/start-all.cmd
hadoop-3.4.0/sbin/hadoop-daemons.sh
hadoop-3.4.0/sbin/refresh-namenodes.sh
hadoop-3.4.0/sbin/start-balancer.sh
hadoop-3.4.0/sbin/start-all.sh
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$
```

Step 5: Moving the Extracted Hadoop Directory to the Installation Location



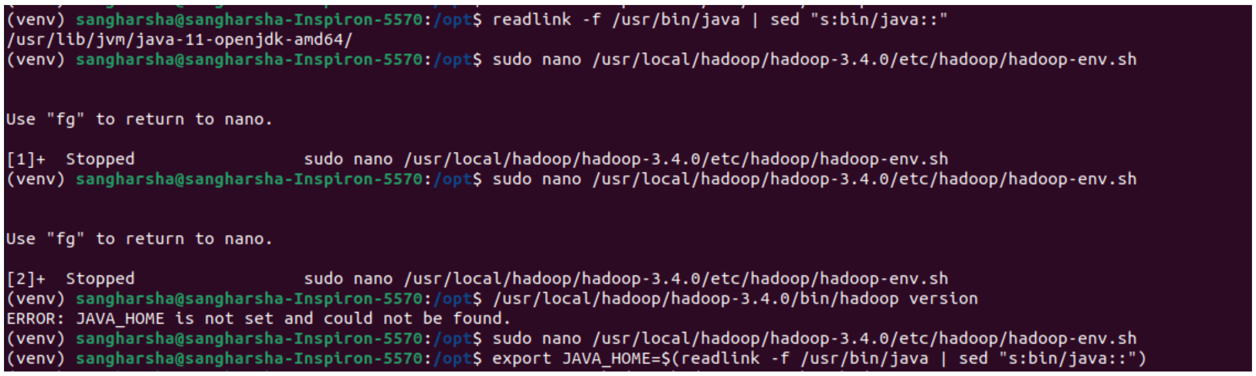
```
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ ls
containerd  couchdb  google  hadoop-3.4.0.tar.gz  venv
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$
```

Step 6: Finding the Installed Java Path



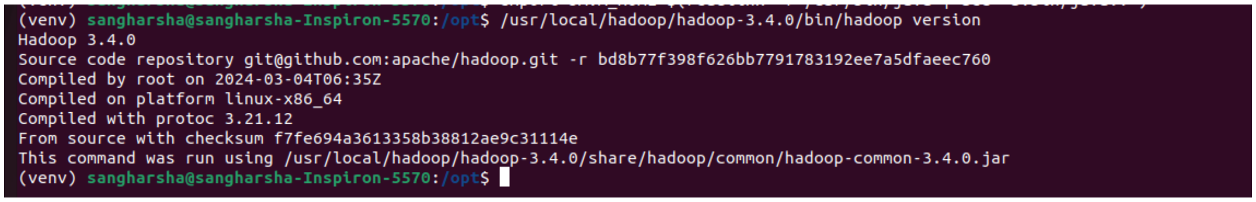
```
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ readlink -f /usr/bin/java | sed
 "s:bin/java::"
/usr/lib/jvm/java-11-openjdk-amd64/
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ █
```

Step 7: Editing Hadoop Environment Configuration File



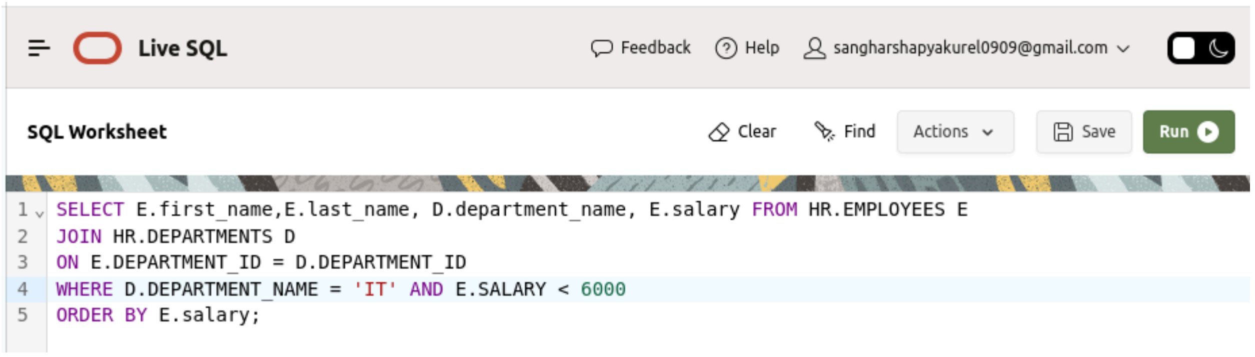
```
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ readlink -f /usr/bin/java | sed "s:bin/java::"
/usr/lib/jvm/java-11-openjdk-amd64/
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ sudo nano /usr/local/hadoop/hadoop-3.4.0/etc/hadoop/hadoop-env.sh


Use "fg" to return to nano.

[1]+  Stopped                 sudo nano /usr/local/hadoop/hadoop-3.4.0/etc/hadoop/hadoop-env.sh
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ sudo nano /usr/local/hadoop/hadoop-3.4.0/etc/hadoop/hadoop-env.sh


Use "fg" to return to nano.

[2]+  Stopped                 sudo nano /usr/local/hadoop/hadoop-3.4.0/etc/hadoop/hadoop-env.sh
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ /usr/local/hadoop/hadoop-3.4.0/bin/hadoop version
ERROR: JAVA_HOME is not set and could not be found.
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ sudo nano /usr/local/hadoop/hadoop-3.4.0/etc/hadoop/hadoop-env.sh
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
```

Step 8:Verifying Hadoop Installation



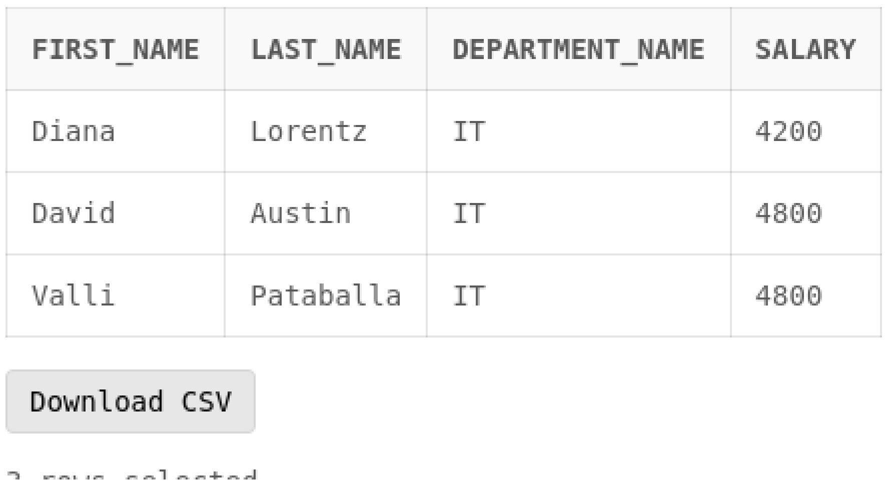
```
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ /usr/local/hadoop/hadoop-3.4.0/bin/hadoop version
Hadoop 3.4.0
Source code repository git@github.com:apache/hadoop.git -r bd8b77f398f626bb7791783192ee7a5dfaeec760
Compiled by root on 2024-03-04T06:35Z
Compiled on platform linux-x86_64
Compiled with protoc 3.21.12
From source with checksum f7fe694a3613358b38812ae9c31114e
This command was run using /usr/local/hadoop/hadoop-3.4.0/share/hadoop/common/hadoop-common-3.4.0.jar
(venv) sangharsha@sangharsha-Inspiron-5570:/opt$ █
```

7. Distributed Horizontal Fragmentation.



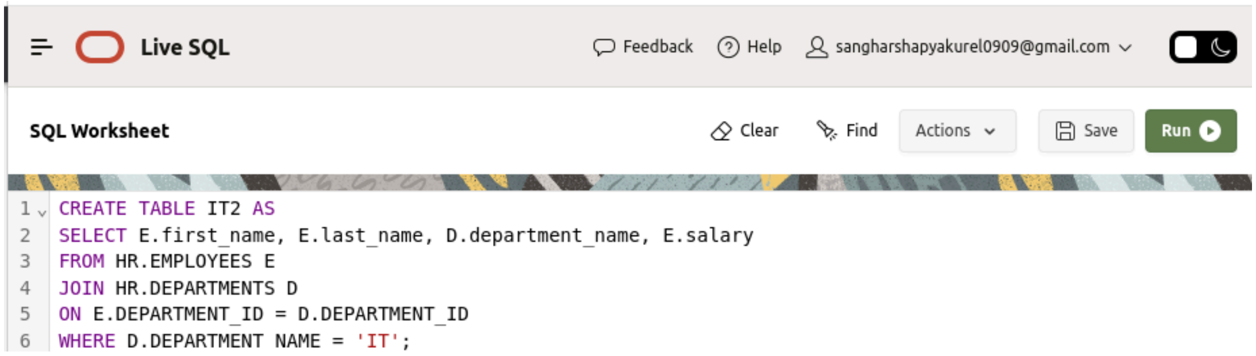
```sql
SELECT E.first_name,E.last_name, D.department_name, E.salary FROM HR.EMPLOYEES E
JOIN HR.DEPARTMENTS D
ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE D.DEPARTMENT_NAME = 'IT' AND E.SALARY < 6000
ORDER BY E.salary;
```

Output:


Download CSV

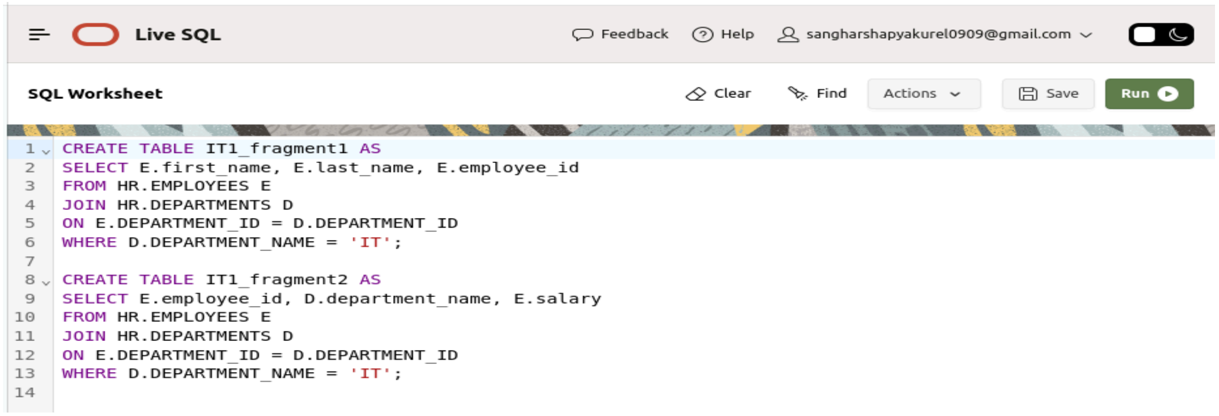
FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	SALARY
Diana	Lorentz	IT	4200
David	Austin	IT	4800
Valli	Pataballa	IT	4800

8. Distributed Vertical Fragmentation.

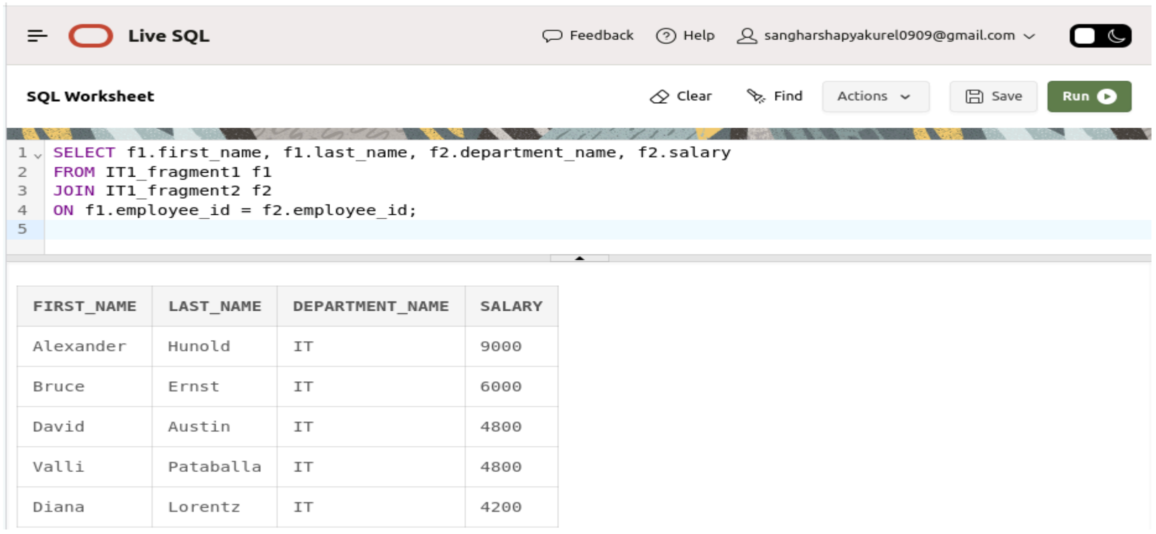


```
CREATE TABLE IT2 AS
SELECT E.first_name, E.last_name, D.department_name, E.salary
FROM HR.EMPLOYEES E
JOIN HR.DEPARTMENTS D
ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE D.DEPARTMENT NAME = 'IT';
```

Create the fragments:



```
CREATE TABLE IT1_fragment1 AS
SELECT E.first_name, E.last_name, E.employee_id
FROM HR.EMPLOYEES E
JOIN HR.DEPARTMENTS D
ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE D.DEPARTMENT_NAME = 'IT';

CREATE TABLE IT1_fragment2 AS
SELECT E.employee_id, D.department_name, E.salary
FROM HR.EMPLOYEES E
JOIN HR.DEPARTMENTS D
ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE D.DEPARTMENT_NAME = 'IT';
```

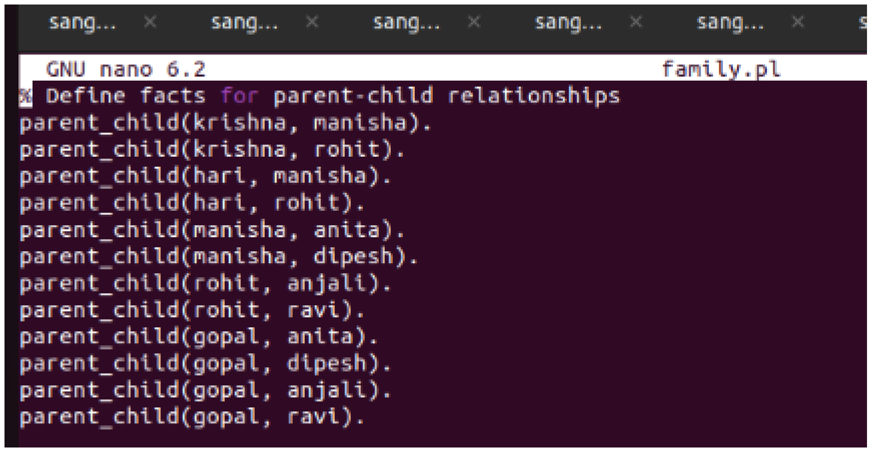


```
SELECT f1.first_name, f1.last_name, f2.department_name, f2.salary
FROM IT1_fragment1 f1
JOIN IT1_fragment2 f2
ON f1.employee_id = f2.employee_id;
```

FIRST_NAME	LAST_NAME	DEPARTMENT_NAME	SALARY
Alexander	Hunold	IT	9000
Bruce	Ernst	IT	6000
David	Austin	IT	4800
Valli	Pataballa	IT	4800
Diana	Lorentz	IT	4200

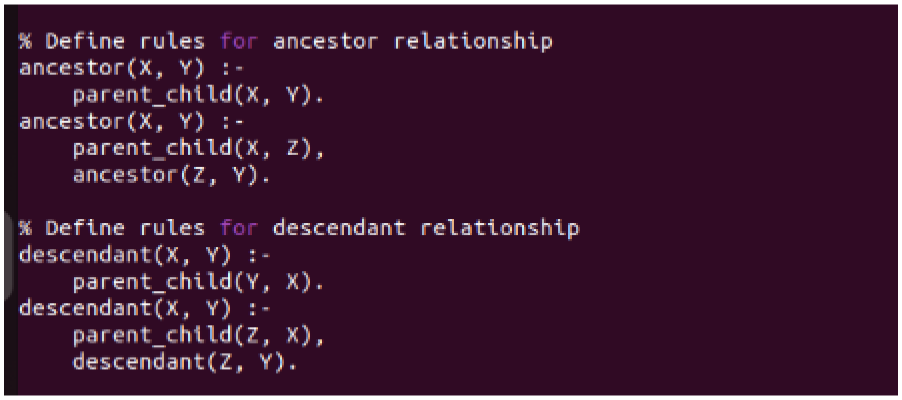
9. Using prolog, perform the following tasks for the Family Relations:

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



```
sang...  ×  sang...  ×  sang...  ×  sang...  ×  sang...  ×
GNU nano 6.2                                  family.pl
% Define facts for parent-child relationships
parent_child(krishna, manisha).
parent_child(krishna, rohit).
parent_child(hari, manisha).
parent_child(hari, rohit).
parent_child(manisha, anita).
parent_child(manisha, dipesh).
parent_child(rohit, anjali).
parent_child(rohit, ravi).
parent_child(gopal, anita).
parent_child(gopal, dipesh).
parent_child(gopal, anjali).
parent_child(gopal, ravi).
```

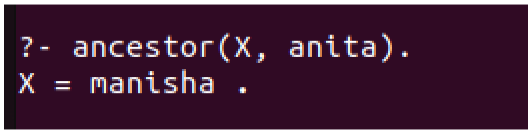
ii. Define rules to represent the ancestor and descendant relationships.



```
% Define rules for ancestor relationship
ancestor(X, Y) :-
    parent_child(X, Y).
ancestor(X, Y) :-
    parent_child(X, Z),
    ancestor(Z, Y).

% Define rules for descendant relationship
descendant(X, Y) :-
    parent_child(Y, X).
descendant(X, Y) :-
    parent_child(Z, X),
    descendant(Z, Y).
```

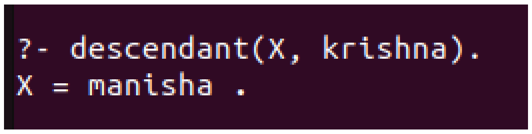
iii. Write queries to find all ancestors and descendants of a given person.

-> Ancestor



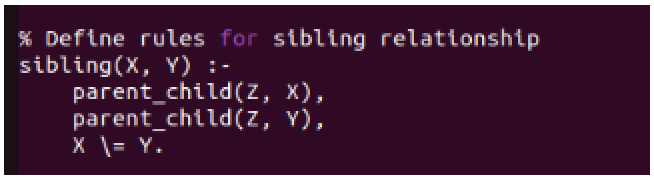
```
?- ancestor(X, anita).
X = manisha .
```

-> Descendant



```
?- descendant(X, krishna).
X = manisha .
```

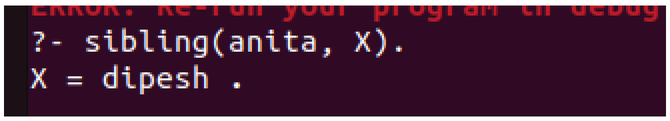
iv. Define rules to represent sibling relationships



```
% Define rules for sibling relationship
sibling(X, Y) :-
    parent_child(Z, X),
    parent_child(Z, Y),
    X \= Y.
```

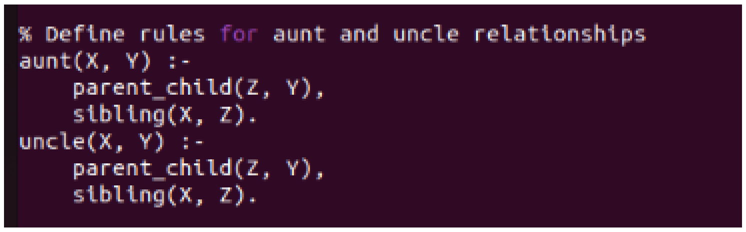
v. Write queries to find all siblings.

-> Query to find all siblings of “anita”:



```
ERROR: Re-run your program in debug
?- sibling(anita, X).
X = dipesh .
```

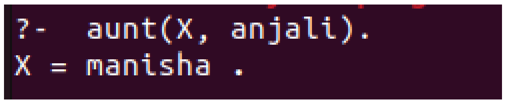
vi. Define rules to represent the aunt and uncle relationships.



```
% Define rules for aunt and uncle relationships
aunt(X, Y) :-
    parent_child(Z, Y),
    sibling(X, Z).
uncle(X, Y) :-
    parent_child(Z, Y),
    sibling(X, Z).
```

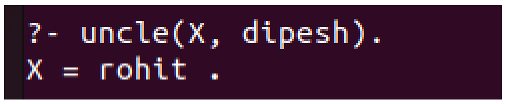
vii. Write queries to find all aunts and uncles of a given person.

-> Query to find all aunts of "anjali":



```
?-  aunt(X, anjali).
X = manisha .
```

-> Query to find all uncles of “dipesh”:

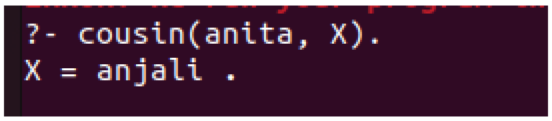


```
?- uncle(X, dipesh).
X = rohit .
```

viii. Define a rule to represent the cousin relationship.

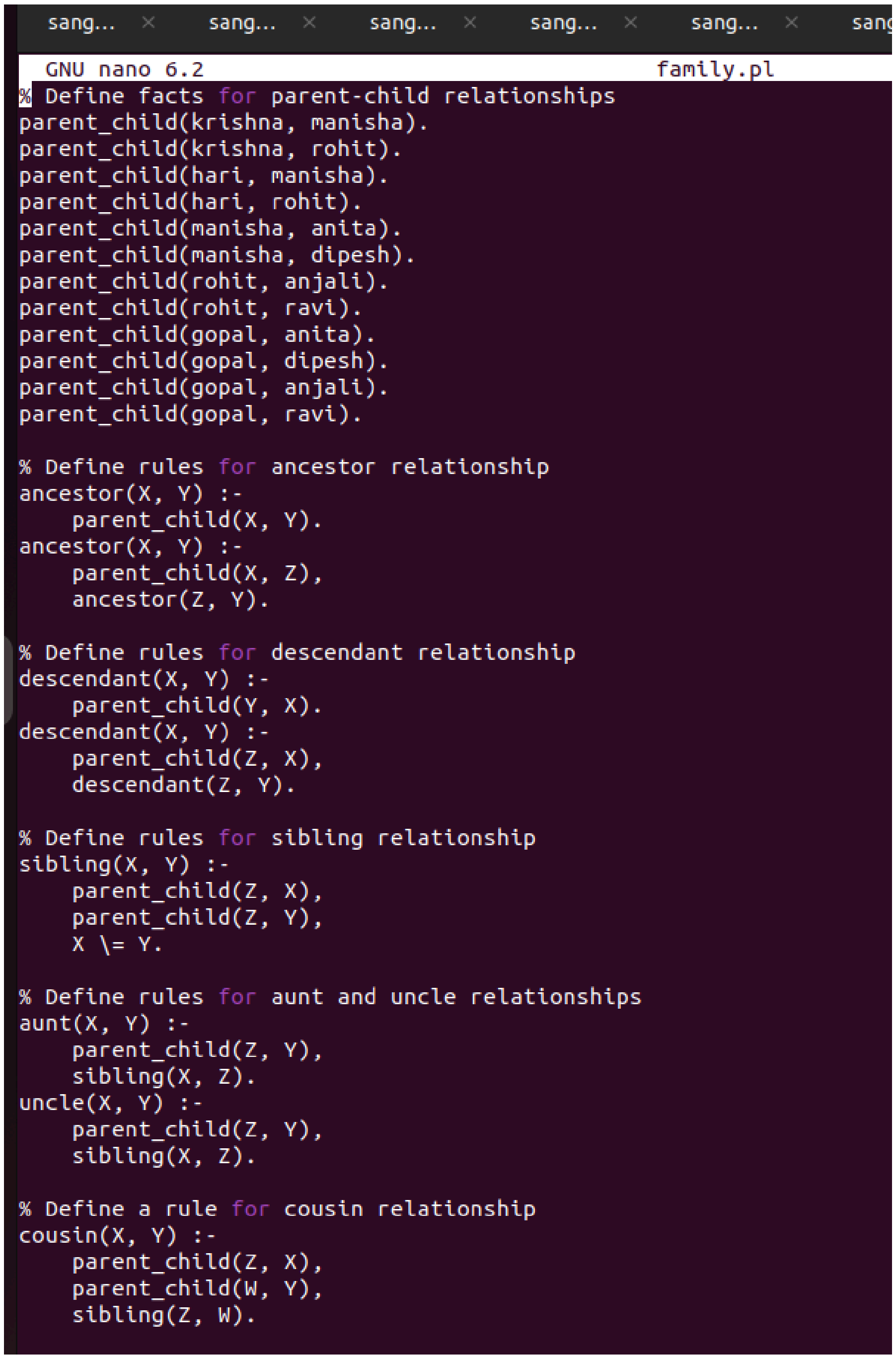
ix. Write a query to find all cousins of a given person.

-> Query to find all cousins of "anita"



```
?- cousin(anita, X).
X = anjali .
```

Complete Code in ‘family.pl’



```
GNU nano 6.2                                          family.pl
% Define facts for parent-child relationships
parent_child(krishna, manisha).
parent_child(krishna, rohit).
parent_child(hari, manisha).
parent_child(hari, rohit).
parent_child(manisha, anita).
parent_child(manisha, dipesh).
parent_child(rohit, anjali).
parent_child(rohit, ravi).
parent_child(gopal, anita).
parent_child(gopal, dipesh).
parent_child(gopal, anjali).
parent_child(gopal, ravi).

% Define rules for ancestor relationship
ancestor(X, Y) :-
    parent_child(X, Y).
ancestor(X, Y) :-
    parent_child(X, Z),
    ancestor(Z, Y).

% Define rules for descendant relationship
descendant(X, Y) :-
    parent_child(Y, X).
descendant(X, Y) :-
    parent_child(Z, X),
    descendant(Z, Y).

% Define rules for sibling relationship
sibling(X, Y) :-
    parent_child(Z, X),
    parent_child(Z, Y),
    X \= Y.

% Define rules for aunt and uncle relationships
aunt(X, Y) :-
    parent_child(Z, Y),
    sibling(X, Z).
uncle(X, Y) :-
    parent_child(Z, Y),
    sibling(X, Z).

% Define a rule for cousin relationship
cousin(X, Y) :-
    parent_child(Z, X),
    parent_child(W, Y),
    sibling(Z, W).
```