```
SELECT * FROM guardians JOIN addresses ON addresses.id =
guardians.address_id WHERE guardians.name='Jon Young';
```

Normalization is the process of reducing data redundancy by organizing fields and table of a database

Denormalization is the process of adding redundant data to speed up complex queries

To know database name of postgreSQLS

Stuff(): its delete or replace the part of the strings.
View : is a virtual table here it only show female data from table.

```
create view female_employee as select * from employee where e_gender='Female'
```

temporary table : automatically terminate once session over . To create use # its store in TempDB

```
create table #book(b_id int, b_cost int)
```

Copy single column into another table

```
INSERT INTO table(column1, column2,...)
SELECT
   column1, column2,...
FROM
   another_table name
WHERE
   condition
```

OLTP : online transaction processing: ability of maintain concurrency its work on decentralized arecheture: its use for fast return process.
OLAP : online analytical processing: its widely used for data mining. Mostly use multi-dimensional schema.
Truncate: delete all of the data from the table.
Delete : delete one or more existing table. You can specify deferent condition and column. If table reference of foregain key then truncate don't work.

Union All operator gives all rows from both tables including the duplicates

Union operator is used to combine the result set of two or more SELECT statements

And if the same record present in both table then its show only one record in union table.
Intersect: its return only common record in both select statement (table)
Copying table : use insert statement

```
insert into employee_duplicate select * from employee
```

https://www.youtube.com/watch?v=TuxuHHaclWU&list=PL08903FB7ACA1C2FB&index=2 sql server

https://www.khanacademy.org/computing/computer-programming/sql/sql-basics/pt/creating-a-table-and-inserting-data sql tutorial
https://www.youtube.com/watch?v=gDqQf4Ekr2A&list=PLeo1K3hjS3uu_n_a__MI_KktGTLYopZ12&index=3 to learn data structure https://codebasicshub.com/

Driver/connector for connection	
Mysql :	Mysql- connector
SQLserver:	Pyodbc
PostgreSQL :	psycopg2
MongoDB	Pymongo

https://www.sqltutorial.org/sql-group-by/ **SQL Tutorial** linux mongo:https://docs.mongodb.com/manual/reference/mongo-shell/
https://www.quackit.com/mongodb/tutorial/mongodb_create_a_collection.cfm **mongo db** https://dzone.com/refcardz/mongodb?chapter=1
https://scotch.io/tutorials/getting-started-with-python-and-mongodb **python with mongodb** https://blog.codecentric.de/files/2012/12/MongoDB-CheatSheet-v1_0.pdf
https://pynative.com/python-postgresql-tutorial/ **postgre sql tutorial**

GitHub, Inc. [US] | github.com/julietplatoon/schema-modifications for migration

https://realpython.com/caching-in-django-with-redis/ **for cashing in django**

How to connect oracle database in python

https://www.vertabelo.com/blog/how-to-create-a-database-model-from-scratch/ **design**
https://djangobook.com/mdj2-models/ database with expiation
<https://www.fullstackpython.com/blog/install-mysql-ubuntu-1604.html> MySQL on Linux
https://www.khanacademy.org/computing/computer-science/algorithms/intro-to-algorithms/v/what-ar
https://mode.com/sql-tutorial/sql-data-types/ advance SQL and Data Annalists.

```
# importing module
import cx_Oracle


# Create a table in Oracle database
try:

    con = cx_Oracle.connect('scott/tiger@localhost')

    # Now execute the sqlquery
    cursor = con.cursor()

    # Creating a table srollno heading which is number
    cursor.execute("create table student(srollno number, \
                    name varchar2(10), efees number(10, 2)")

    print("Table Created successful")

except cx_Oracle.DatabaseError as e:
    print("There is a problem with Oracle", e)

# by writing finally if any error occurs
# then also we can close the all database operation
finally:
    if cursor:
        cursor.close()
    if con:
        con.close()
```

From <https://www.geeksforgeeks.org/oracle-database-connection-in-python/>

Connecting to PostgreSQL

```
import psycopg2
try:
    connection = psycopg2.connect(user = "sysadmin",
                                  password = "pynative@#29",
                                  host = "127.0.0.1",
                                  port = "5432",
                                  database = "postgres_db")

    cursor = connection.cursor()
    # Print PostgreSQL Connection properties
    print ( connection.get_dsn_parameters(),"\n")

    # Print PostgreSQL version
    cursor.execute("SELECT version();")
    record = cursor.fetchone()
    print("You are connected to - ", record,"\n")

except (Exception, psycopg2.Error) as error :
    print ("Error while connecting to PostgreSQL", error)
finally:
    #closing database connection.
        if(connection):
            cursor.close()
            connection.close()
            print("PostgreSQL connection is closed")
```

You should get the following output after connecting to PostgreSQL from Python

```
{'user': 'postgres', 'dbname': 'pynative_DB', 'host': '127.0.0.1', 'port': '5432', 'tty': '',
 'options': '', 'sslmode': 'prefer', 'sslcompression': '1', 'krbsrvname': 'postgres',
'target_session_attrs': 'any'}

You are connected to -  ('PostgreSQL 10.3')

PostgreSQL connection is closed
```

Binary log (bin log) : whenever we insert or update or delete into a sql database then its image a log outside
We can use for search or big data transformation.

ravel () = it's not touch the original array but its turn any number array in one dimension array .

primary key : A primary key is a special key which uniquely identifies each record in a table.

Foreign Key: It a link between records in two different tables in a database. It's important while doing normalization especially when tab les need to access other tables.

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right tab

Mango db : its document oriented database program its store data in hardisk in BSON(binary json) format . Its mostly use real-time web application or in big data like run in ajax base application like chat app
Document mean Row,, and collection mean table in mangodb
Key is like **column**
Row mean **document**

Condition	Syntax	RDBMS Equivalent
Equality	{ <key> : <value> }	WHERE col = value
Less Than	{ <key> : { $lt : <value> } }	WHERE col < value
Less Than Equal	{ <key> : { $lte : <value> } }	WHERE col <= value
Greater Than	{ <key> : { $gt : <value> } }	WHERE col > value
Greater Than Equal	{ <key> : { $gte : <value> } }	WHERE col >= value
Not Equal	{ <key> : { $ne : <value> } }	WHERE col != value

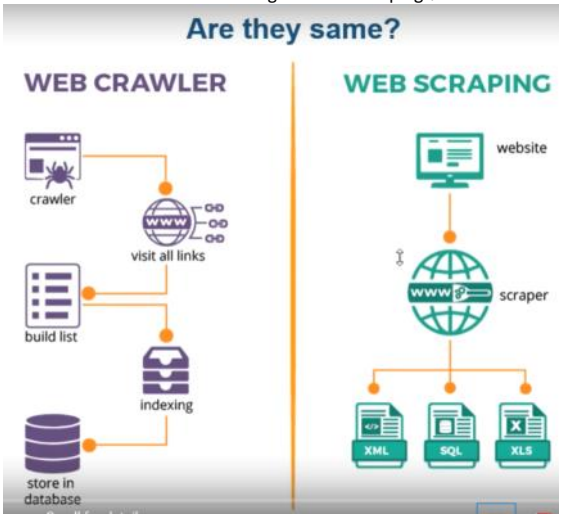
```
Connect to Mongo Server
Create database
Create collection under that database
----CRUD---
Create Document     -- insert()
Read Document       -- find()
Update Document     -- update()
Delete Document     -- delete() / remove()


copy collection
copy document
export collection
drop collection
drop database
Disconnect from Server
```

https://www.youtube.com/watch?v=o3tYiyE_OXE web scraping with selenium

Web scraping with bountiful Soap

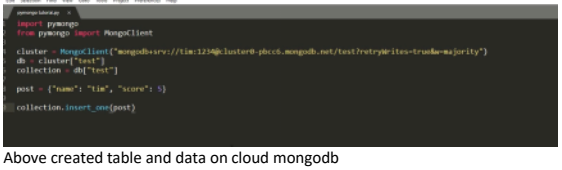
https://www.interviewqs.com/blog/web_scrape

Difference between web crawling and web scraping ?



https://docs.mongodb.com/manual/reference/sql-comparison/ sql to mongodb mapping chart

Mnongo Db using pymango modules



Above created table and data on cloud mongodb
Inserting multiple data

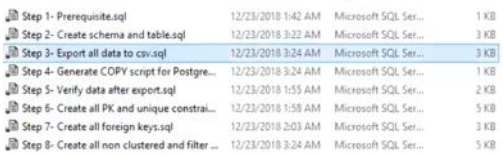
Secure way of sql data

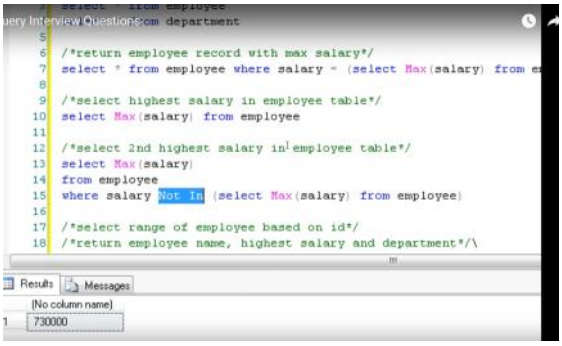
by BEGI N TRANSACTI ON and COMMI T
By using LIMIT
by Replication of data
by Grant (access permition)
by Making backup

Data encryption: Besic
We can use **pycrypto** (3rd party).
We can use **django-pgcrypto** for Django web framework

```
from django.db import models
import pgcrypto

class Employee (models.Model):
    name = models.CharField(max_length=100)
    ssn = pgcrypto.EncryptedTextField()
    pay_rate = pgcrypto.EncryptedDecimalField()
    date_hired = pgcrypto.EncryptedDateField(cipher='Blowfish', key='datekey')
```

For data migration sql server to postgresql need command to run





Sql query on salary : NOT IN mean not equal to

```
/*select range of employee based on id*/
select *
from employee
where employee_id between 2003 and 2008

/*return employee name, highest salary and department*/
select e.first_name, e.last_name, e.salary, d.department_name
from Employee e Inner Join Department d ON (e.department_id = d.department_id)
where salary IN (select Max(salary) from employee)

/*return highest salary, employee name, department name for each department*/
select e.first_name, e.last_name, e.salary, d.department_name
from Employee e Inner Join Department d ON(e.department_id = d.department_id)
where salary IN (select Max(salary) from employee group by department_id)
```

SQL Server	MySQL
Developed by Microsoft	Developed by Oracle
Licensed software	Open-source software
Supports C#, Java C++, PHP, Visual Basic, Perl, Python, Ruby, etc	Supports PHP, Perl, Python, Ruby, etc

```
cluster = MongoClient("mongodb+srv://tim:1234@cluster0-pbcc6.mongodb.net/test?retryWrites=true&w=majority")
db = cluster["test"]
collection = db["test"]

post1 = {"_id":5, "name":"joe"}
post2 = {"_id":6, "name":"bill"}

results = collection.find({"name":"bill"})

for result in results:
    print(result["_id"])
```

Above finding and searching data

```
results = collection.update_one({"_id":5}, {"$set":{"hello":5}})
```

Updating

```
post_count = collection.count_documents({})
print(post_count)
```

Counting a number of document available in db

Licensed software	Open-source software
Supports C#, Java C++, PHP, Visual Basic, Perl, Python, Ruby, etc	Supports PHP, Perl, Python, Ruby, etc
Doesn't allow any kind of database file manipulation while running	Allows database file manipulation while running.
Allows query cancellation mid-way in the process	Doesn't allow query cancellation mid-way in the process.
While backing up the data, It doesn't block the database	While backing up the data, it blocks the database
Takes a large amount of operational storage space.	Takes less amount of operational storage space.
Available in Express and Custom mode.	Available in MySQL Community Edition, and MySQL Enterprise Edition

SQL Command

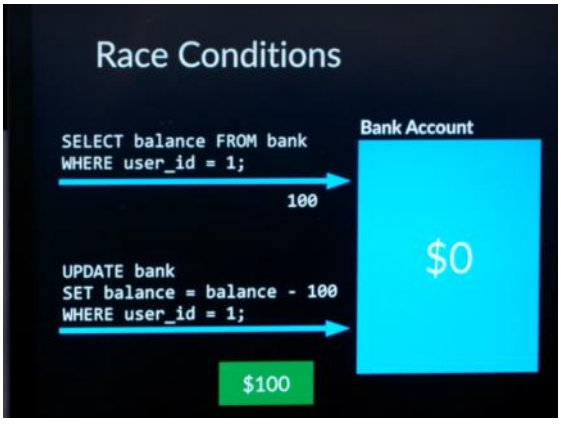
```
lecture3=# SELECT AVG(duration) FROM flights WHERE origin = 'New York';
         avg
---------------------
 425.0000000000000000
(1 row)
```

```
lecture3=# SELECT origin, COUNT(*) FROM flights GROUP BY origin;
  origin   | count
-----------+-------
 Shanghai  |     1
 New York  |     2
 Lima      |     1
 Istanbul  |     1
 Moscow    |     1
(5 rows)
```

```
lecture3=# SELECT origin, COUNT(*) FROM flights GROUP BY origin HAVING COUNT(*) > 1;
  origin   | count
-----------+-------
 New York  |     2
(1 row)
```

```
SELECT * FROM flights
WHERE origin LIKE '%a%';
```

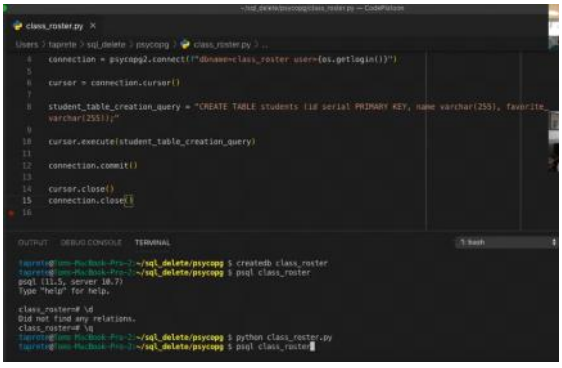
id	origin	destination	duration
1	New York	London	415
2	Shanghai	Paris	760
3	Istanbul	Tokyo	700
4	New York	Paris	435
5	Moscow	Paris	245
6	Lima	New York	455




```
UPDATE flights
    SET duration = 430
    WHERE origin = 'New York'
    AND destination = 'London';
```

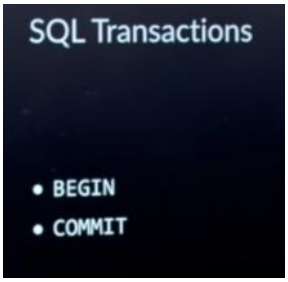
```
lecture3=# SELECT origin, destination, name FROM flights JOIN passengers ON passengers.fli
ght_id = flights.id WHERE name = 'Alice';
  origin   | destination | name
-----------+-------------+-------
 New York  | London      | Alice
(1 row)
```

Connection to postgreSQL in Linux base



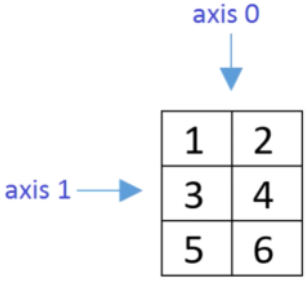
Showing student table

```
psql (11.5, server 10.7)
Type "help" for help.

class_roster=# \d
              List of relations
 Schema |       Name       |   Type   |  Owner
--------+------------------+----------+---------
 public | students         | table    | taprete
 public | students_id_seq  | sequence | taprete
(2 rows)
```



By using sql transaction we can avoid duplicate transaction exple: if money want to withdraw by atm at the same time the we have to use sql transaction

Axist - 1 mean rows
Axis 0 mean column



```
                List of relations
 Schema |       Name        |   Type   |  Owner
--------+-------------------+----------+---------
 public | students          | table    | taprete
 public | students_id_seq   | sequence | taprete
(2 rows)

class_roster=#
class_roster=# \d students
```

Now creating database and putting CSV file into database

```
OUTPUT  DEBUG CONSOLE  TERMINAL

taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ ls
class_roster.py
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ ls
class_roster.py
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ rm class_roster.py
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ ls
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ touch real_estate.py sacramento_realestate.csv
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ code sacramento_realestate.csv
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $
```

real_estate.py

```
Users > taprete > sql_delete > psycopg > real_estate.py > ...
1   # import csv
2   # import psycopg2
3
4
5   with open('sacramento_realestate.csv', mode='r') as csv_file:
6       csv_reader = csv.DictReader(csv_file)
7       for row in csv_reader:
8
```

```
OUTPUT  DEBUG CONSOLE  TERMINAL

taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ code real_estate.py
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ ls
real_estate.py          sacramento_realestate.csv
taprete@Toms-MacBook-Pro-2:~/sql_delete/psycopg $ clear\
```

Asigning row data of CSV file column name to created table column name

```
Users > taprete > sql_delete > psycopg > real_estate.py > ...
1   import csv
2   import os
3   import psycopg2
4
5   from decimal import Decimal
6   from datetime import datetime
7
8   connection = psycopg2.connect(f"dbname=real_estate user={os.getlogin()}")
9
10  cursor = connection.cursor()
```

```
real_estate.py
Users > taprete > sql_delete > psycopg > real_estate.py > clean_data
cursor = connection.cursor()

def table_creation_query():
    return "CREATE TABLE properties (id serial PRIMARY KEY, street_address varchar, city varchar, zip_code
    state varchar, number_of_beds integer, number_of_baths integer, square_feet integer, property_type var
    sale_date timestamp, sale_price integer, latitude decimal, longitude decimal);"

cursor.execute(table_creation_query())

def clean_data(csv_row):
    cleaned = {}
    cleaned['street_address'] = csv_row['street']
    cleaned['city'] = csv_row['city']
    cleaned['zip_code'] = csv_row['zip']
    cleaned['state'] = csv_row['state']
    cleaned['number_of_beds'] = int(csv_row['beds'])
    cleaned['number_of_baths'] = int(csv_row['baths'])
    cleaned['square_feet'] = int(csv_row['sq__ft'])
    cleaned['property_type'] = csv_row['type']
    cleaned['sale_date'] = datetime.strptime(csv_row['sale_date'], '%m/%d/%y')
    cleaned['sale_price'] = csv_row['price']
    cleaned['latitude'] = Decimal(csv_row['latitude'])
    cleaned['longitude'] = Decimal(csv_row['longitude'])
```

```
cleaned['latitude'] = Decimal(csv_row['latitude'])
cleaned['longitude'] = Decimal(csv_row['longitude'])
return cleaned
```

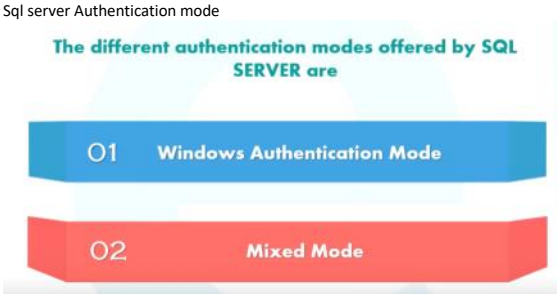
```
cursor = connection.cursor()
# Creating DB table
cursor.execute("DROP TABLE IF EXISTS properties CASCADE;")
cursor.execute(table_creation_query())

with open('sacramento_realestate.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        new_data = clean_data(row)
        cursor.execute("INSERT INTO properties (street_address, city, zip_code, state, number_of_beds,
        number_of_baths, square_feet, property_type, sale_date, sale_price, latitude, longitude) VALUES (%s,
        %s, %s, %s, %s, %s, %s, %s, %s, %s)", (new_data['street_address'], new_data['city'], new_data['zip_c
        new_data['state'], new_data['number_of_beds'], new_data['number_of_baths'], new_data['square_feet'],
        ['property_type'], new_data['sale_date'], new_data['sale_price'], new_data['latitude'], new_data['lo
        )


connection.commit()

cursor.close()
```

Caching setup in django, inside setting.py under middleware

axis 1 →

3	4
5	6

Sql server Authentication mode



Windows authentication mode: authenticate by winnows account sql server take windows user and pass then authenticate also disable sql authentication in this mode.
Mixed mode: we have set username and password by user then we can use to authenticate by enter it
To know the version sql server

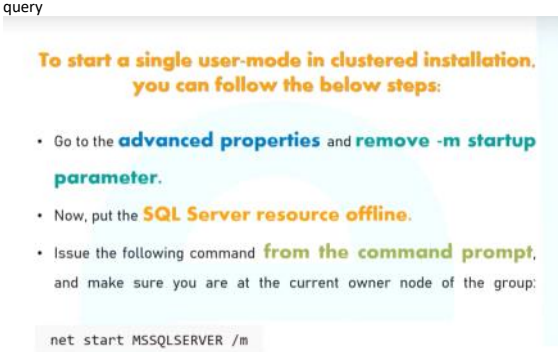
```
SELECT @@VERSION
```

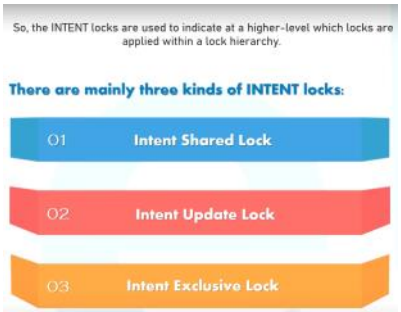
Single user mode

```
sqlcmd -m
```

Sql profiler
TCP/IP sql server run on port number 1433

Subquery : query inside query called sub query , sub query excute first then result passed to the main query





SQL Server Data Quality Services (DQS) enable the user to build a knowledge base and thereafter use it to perform tasks such as correction, deduplication, enrichment, standardization of data.

Magic Tables are tables **automatically created tables** in SQL Serve used to internally store the inserted, updated

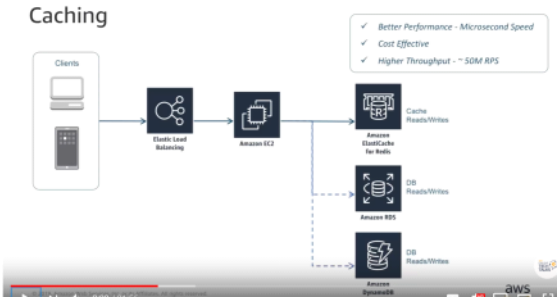
```
cursor.close()
```

Cashing setup in django inside setting.py under middleware

```
#'django.middleware.cache.UpdateCacheMiddleware',
'django.middleware.common.CommonMiddleware',
#'django.middleware.cache.FetchFromCacheMiddleware',
```

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    }
```

Redis : store data as key - value store, all data reside(store data) in main memory server while other DB like postgresql, mongoDB and other store data in Disk



a tool for testing performance.

Installing loadtest as root is simple:

```
sudo npm install -g loadtest
```

```
$ loadtest -n 100 -k  http://localhost:8000/store/

# result
INFO Requests per second: 55
```

The **FLOOR function** is used to round up a non-integer value to the previous least integer value. This function returns a unique value after rounding the digits.

Write a SQL Query to find first weekday of the month?

```
SELECT DATENAME(dw, DATEADD(dd, - DATEPART(dd, GETDATE()) +
1, GETDATE())) AS FirstDay
```

Rename the database name

```
sp_renamedb 'OldDatabaseName', 'NewDatabaseName';
```

UPDATE_STATISTICS

Used to update the information used by indexes such as the distribution of key values for one or more statistics groups in the mentioned indexed view or table.

SCOPE_IDENTITY

Used to create identity value for tables in the current execution scope.

Hotfixes are single, cumulative software packages applied to live

Magic Tables are tables **automatically created tables** in SQL Serve used to internally store the inserted, updated values **for DML operations** such as (SELECT, DELETE, INSERT, UPDATE, etc).

A recursive stored procedure is a problem-solving method through which you can arrive at the solution again and again.

The process of automation of backup to restore databases from one standalone server to another standalone standby server is known as Log Shipping.

SUBSTR	CHARINDEX
Used to return a specific portion of the string in a given string	Used to return a character position in a given specified string
Example: SUBSTRING('Edureka',1,4) **Output:** Edur	**Example:** CHARINDEX('r','Edureka',1) **Output:** 4

Mirroring in SQL Server is designed to maintain a hot standby server, that is consistent with the primary server in terms of a transaction.

Also, the transaction log records are sent from the principal server to the secondary server.

The CHECK constraint in SQL Server is used to limit the values or type of data stored in a column.

The **COALESCE function** is used to return the first non-null expression within arguments.

Django's cache framework:
Memcached : store session data cashe into the memcashed
Setting.py

```
SESSION_ENGINE = "django.contrib.sessions.backends.cache"

CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    }
}
```

Now download and install if you use window

```
pip install python-memcached
pip install python3-memcached


Download Link
http://downloads.northscale.com/memcached-1.4.5-amd64.zip
```

Redis caching
If don't wanna cashe then use decorator **@never_cashe**
https://docs.djangoproject.com/en/3.0/topics/cache/

```
PS C:\Users\Taimoor\Desktop\python> python ^C
PS C:\Users\Taimoor\Desktop\python> pip install redis requests json
```

Redis command in linux

```
sudo apt-get install redis-server

systemctl status redis

mkdir rediscache

cd rediscache

virtualenv env -p python3

source env/bin/activate

pip install django

pip install django-redis

#pip install psycopg2

pip install psycopg2-binary

pip install djangorestframework

django-admin startproject django_cache
```

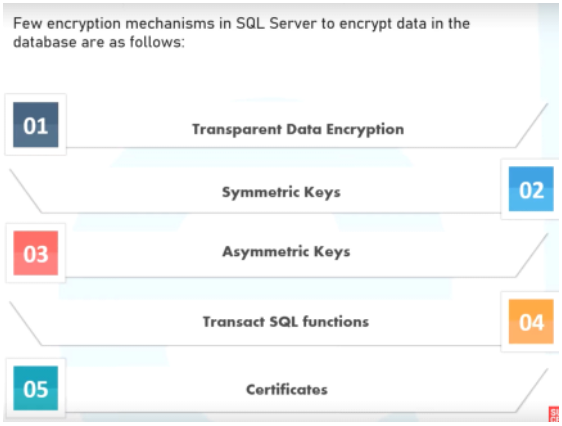
the current execution scope.

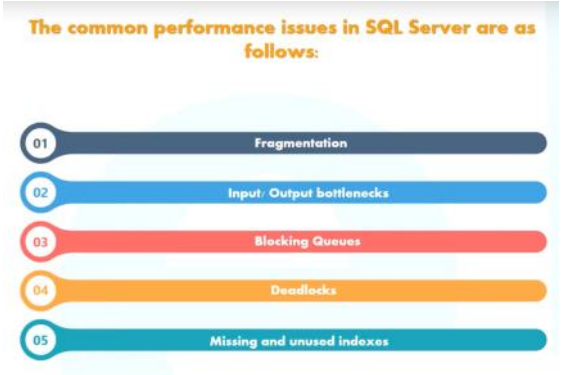
Hotfixes are single, cumulative software packages applied to live systems.

This includes one or more files used to address a problem in a software product.

Patches are a programs installed on the machines to rectify the problem occured in the system and ensured the security of the system.

Protect data





Common issue in sql server

PL/SQL

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2  NAME VARCHAR2(10) := 'SAM';
  3  BEGIN
  4  DBMS_OUTPUT.PUT_LINE(NAME);
  5  END;
  6  /
SAM

PL/SQL procedure successfully completed.
```

SQL server : we can insert data into sql server with the two way
1st by **datafile** like **.mdf**

```
SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\WideWorldImporters.mdf' I S
```

2nd way

Data size and limitation

```
\MSSQL\DATA\WideWorldImporters.mdf' , SIZE = 1048576KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB ),
\MSSQL\DATA\WideWorldImporters_UserData.mdf' , SIZE = 2097152KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB ),
```

```
pip install psycopg2-binary

pip install djangorestframework

django-admin startproject django_cache

cd django_cache

python manage.py startapp testappmodel

'testappmodel', # add app name here
'rest_framework', # add here too

#postgres

DATABASES = {
```

Template fragments

```
{% load cache %}
{% cache 600 homepage_footer %}
<a href="https://in.pycon.org">PyCon</a>
{% endcache %}
{% load cache %}
{% cache 600 homepage_footer request.user.username %}
<a href="https://in.pycon.org">PyCon</a>
{% endcache %}
```

Template fragments

- Use inside blocks
- Multiple cache blocks within a single block
- Use lazy objects

```
from django.core.cache import cache
from django.core.cache.utils import
make_template_fragment_key
cache.delete(make_template_fragment_key("homepage_fo
oter", vary_on=[request.user.username])
```

Sessions

- Very good candidate for using a cache (as opposed to file system or database)

```
SESSION_SERIALIZER='django.contrib.sessions.serializer
s.PickleSerializer'
SESSION_ENGINE =
'django.contrib.sessions.backends.cache'
```

Cashing individual object

Caching objects

```
from django.core.cache import cache
cache.set('Python', 'Django', timeout=600)
cache.get('Python')
cache.delete('Python')
cache.set_many({'Bangalore':2015,'Mumbai':1997,'Delhi':2
004,'Pune':2005,'Hyderabad':2003,'Jaipur':1981})
cache.get_many(['Bangalore','Mumbai','Delhi','Pune','Hyde
rabad','Jaipur'])
delete, delete_many, incr, decr
```

Show new thing first

Invalidating cache

- Post-save signals come handy

```
from django.db.models import signals

def invalidate_flower_cache(sender, instance, **kwargs):
    cache_key = "flower_details" + instance.id
    cache.delete(cache_key)
    cache_key = make_template_fragment_key('flower_details', [flower.id])
    cache.delete(cache_key)

    signals.post_save.connect(invalidate_flower_cache, sender=Flower)
```

Database cashes

Object cache vs. MySQL query cache

- MySQL query cache invalidated as soon as 'any' part of the table is modified
- query_cache_type

Data size and limitation

```
\MSSQL\DATA\WideWorldImporters.mdf' , SIZE = 1048576KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB ),
\MSSQL\DATA\WideWorldImporters_UserData.ndf' , SIZE = 2097152KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB ),
JLSERVER\MSSQL\DATA\WideWorldImporters_InMemory_Data_1' , MAXSIZE = UNLIMITED)
JL\DATA\WideWorldImporters.ldf' , SIZE = 102400KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )
```

- MySQL query cache invalidated as soon as 'any' part of the table is modified
- query_cache_type
- SQL_CACHE, SQL_NO_CACHE hints for select queries

we can create Transactional log also **.ldf**

```
LOG ON
( NAME = N'WWI_Log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\WideWorldImporte
GO
```

to cashe entire website after setup cashe use this inside settigs.py
Middleware cashe

```
MIDDLEWARE = [
    'django.middleware.cache.UpdateCacheMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.cache.FetchFromCacheMiddleware',
]
```

IDENTITY : its **automatically increment the value** while each raw added to the table
Primary key : **uniquely identify each row** in the table
Not null : **you have to put value anyhow,** you can't leave value blank
DEFALUT **GETDATE()** : ITS show the current date
Forgain key : use to related to the primary key of the another table mean forgainkey use to make relationship between tables
INDEXES: we can use index Improve performance of database , index mean **right order** of data
MEMORY OPTIMIZATION: memory performance
Sp_helptext : To view definition of system design store procedure
Sp_depends : help to see dependency of table or store procedure.
Return value : 0 mean success and non-zero mean failure
DISTINCT: statement is used to return only **distinct** (different) values.
Temporary table : we can create using # before table name 2 types of temp table 1st **#employee(local)** 2nd **##employee(globle)** its store in tempDB folder **but we can't create view or funtion on temporary table because its drop table automatically after execution of store procedure .**
local temp table: only available in that connection where created not available for other connection and its drop automatically when connection close and no need unique table name while create table in different connection **but global are available for all connection and need unique name of table if we create table in different connection** .
Wildcard : when we use % one side or both side its called wildcard **% employee id % its** used with the SQL **LIKE** operator.
SYSOBJECTS: for see all the table in database **select * from SYSOBJECTS where xtype = 'U';**
https://docs.microsoft.com/en-us/sql/relational-databases/system-compatibility-views/sys-sysobjects-transact-sql?view=sql-server-ver15

```
-- Gets the list of tables only
Select * from SYSOBJECTS where XTYPE='U'

-- Gets the list of tables only
Select * from SYS.TABLES

-- Gets the list of tables and views
Select * from INFORMATION_SCHEMA.TABLES
```

Re -runnable script : **create using exist()** like if not exists(create table .....)
to find the all blocking query (transactions) : https://www.sqlskills.com/blogs/paul/script-open-transactions-with-text-and-plans/
DBCC OPENTRAN : helps to identify active transactions and old active transactions details
Pivot operator : **present data in cross tab format.**
To see all login user and active connections :
select is_user_process, original_login_name from sys.dm_exec_sessions order by login_time desc;
You can read all error log : using **Execute sp_readerrorlog**
Diffrence in where and having : **where cannot be use with aggregate function** while having can use .**where filter rows before aggregate perform** while having filter after aggregate calculations
Coalesce: return first NON NULL value
Normalization : mean **reducing data duplication and moving repeated (duplicate) data into separate table** (minimize data redundancy)

Return status only return one value output return n number of value

Return Staus Value	Output Parameters
Only Integer Datatype	Any Datatype
Only one value	More than value
Use to convey success or failure	Use to return values like name, count etc..

Most databases contain multiple tables
- This is a direct result of normalization

Use keys to determine relationships between rows in differe tables
- A primary key identifies a unique row in a table
- A foreign key references a row in another table

Create — INSERT ...
Read — SELECT ...
Update — UPDATE ...
Delete — DELETE ...

Advantage of store procedure
1. Execution plan retention and reusability
2. Reduces network traffic
3. Code reusability and better maintainability
4. Better Security
5. Avoids SQL Injection attack

- Data in tables is managed through CRUD operations (Create, Retrieve, Update, Delete)
- To create data, use INSERT
- To retrieve data, use SELECT
- To update data, use UPDATE
- To delete data, use DELETE
- Wrap your SELECT statements in views or procedures
- Wrap your INSERTs, UPDATEs, DELETEs and even SELECTs in procedures

views : Wrapping a crud operation mean if you **don't want show or don't want to give access of real table and cluman to the user** then we can create views. So user don't have to write sql code

Dbo. Is schema name & customer is real table name

Schema : its kind of container witch hold the table itself and hold name of the table

```
CREATE TABLE dbo.Customer
```

Views and store processer always store in the schema

If dbserver hosted in the cloud then its also called on-premise database like aws and azure database

```
ALTER TABLES Sales.product ADD supplier INTEGER NOT NULL CONSTRAINT def_supplier DEFAULT 1 ;
```

IF YOU DON'T INTER ANY VALUE THEN DEFULT VALUE WILL BE ALWAYS 1 FOR SUPLIER CLOUMAN
DELETE : delete use to delete data in a table.
DROP: drop is use to delete a table. Its delete hole table from database

Create tables using the CREATE TABLE statement
- Specify a table name and column specifications

Modify table definition using the ALTER TABLE statement
- Add or remove columns, constraints, keys

Remove tables using the DROP TABLE statement
- Removes the table from the database

Product

ProductID	Name	Price	Supplier
1	Widget	12.99	1
2	Thingybob	3.75	2

```
INSERT INTO Product (Name, Price, Supplier)
VALUES ('Thingybob', 3.75, 2);
```

ANOTHER WAY TO INSERT WITHOUT SPECIFY COLUMAN NAME

```
INSERT INTO Product
VALUES ('Widget', 12.99, 1);
```

Selecting way and filtering way of table data

```
SELECT Name, Price
FROM Product
WHERE Supplier = 2;
```

Name	Price
Thingybob	3.75

```
SELECT Name AS Product,
       Price * 0.9 AS SalePrice
FROM Product;
```

Product	SalePrice
Widget	11.691
Thingybob	3.375
Knicknack	NULL
Wotsit	NULL

Index: it use to find data quickly from table and also use for database engine to avoid scan top to button

Index Types
1. Clustered
2. Nonclustered
3. Unique
4. Filtered
5. XML
6. Full Text
7. Spatial
8. Columnstore
9. Index with included columns
10. Index on computed columns

Cluster use for physical order of data in table and primary key create automatically cluster index in id column. And use **drop to delete index tabl**
Clustered index store data in table itself so it's faster than non-cluster
compose clustered index: If index contain more than one column then its call compose cluster index. Since index store separate need extra memory
Non-clustered index : data store in one palace and index store another place and index have pointer to the store data location.
To check cluster index use : **sp_helpindex with name of table** .
Unique index: its enforce uniqueness of key value in index like id primary key its provide uniqueness in id
INDEX on view: usually can't store data on views but if use index mean it capable to store data in view.

Difference between Unique Constraint and Unique Index

views : Wrapping a crud operation mean if you **don't want show or don't want to give access of real table and cluman to the user** then we can create views. So user don't have to write sql code
View also known as virtual table

```
CREATE VIEW vw_ProductPrice
AS
SELECT Name, Price
FROM Product
WHERE Supplier = 2;

SELECT Name, Price
FROM vw_ProductPrice;
```

```
CREATE PROCEDURE transferFunds
AS

  UPDATE SAVINGS
  SET Balance += 500
  WHERE AccountID = 3;
  UPDATE CHECKING
  SET Balance -= 500
  WHERE AccountID = 3;

                          EXEC transferFunds;
```

Savings

AccountID	Balance
1	1000
2	2000
3	1500
4	2000

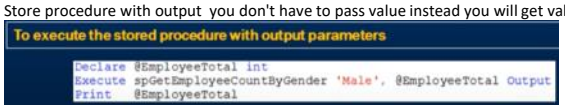
Checking

AccountID	Balance
1	1000
2	2000
3	500
4	2000

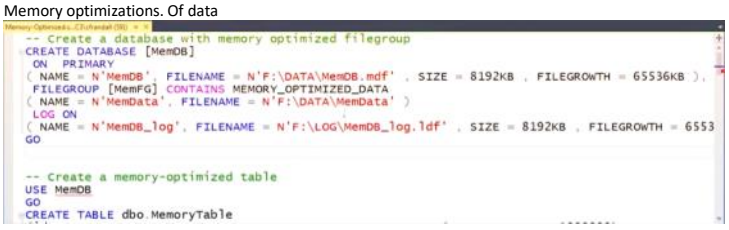
When we wrap two separate operation and run together then we use **store procedure**
To alter and delete store procedure

```
sp_helptext spGetEmployees

ALTER PROCEDURE spGetEmployees
AS
BEGIN
  Select Name, Gender from tblEmployee order by Name
END

Drop proc spGetEmployees
```

Store procedure with output you don't have to pass value instead you will get value



Memory optimizations. Of data



```
-- Create a database with memory optimized filegroup
CREATE DATABASE [MemDB]
 ON  PRIMARY
( NAME = N'MemDB', FILENAME = N'F:\DATA\MemDB.mdf' , SIZE = 8192KB , FILEGROWTH = 65536KB ),
 FILEGROUP [MemFG] CONTAINS MEMORY_OPTIMIZED_DATA
( NAME = N'MemData', FILENAME = N'F:\DATA\MemData' )
 LOG ON
( NAME = N'MemDB_log', FILENAME = N'F:\LOG\MemDB_log.ldf' , SIZE = 8192KB , FILEGROWTH = 6553
GO


-- Create a memory-optimized table
USE MemDB
GO
CREATE TABLE dbo.MemoryTable
```

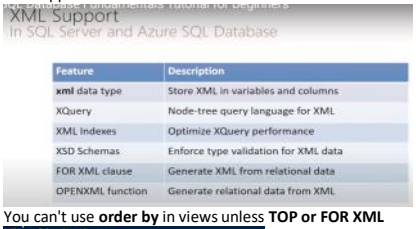
```
-- Create a memory-optimized table
USE MemDB
GO
CREATE TABLE dbo.MemoryTable
(id INTEGER NOT NULL PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 1000000),
 date_value DATETIME NULL)
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);

-- Create a disk-based table
CREATE TABLE dbo.DiskTable
(id INTEGER NOT NULL PRIMARY KEY NONCLUSTERED,
 date_value DATETIME NULL);
```

```
-- Insert 500,000 rows into DiskTable. This code uses a transaction to insert rows into the d
-- When code execution is complete, look at the lower right of the query editor status bar an

BEGIN TRAN
    DECLARE @Diskid int = 1
    WHILE @Diskid <= 500000
    BEGIN
        INSERT INTO dbo.DiskTable VALUES (@Diskid, GETDATE())
        SET @Diskid += 1
    END
COMMIT;

-- Verify DiskTable contents. Confirm that the table now contains 500,000 rows
SELECT COUNT(*) FROM dbo.DiskTable;
```

Create Netive store procedure

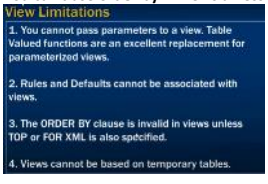
```
-- Create a native stored proc
CREATE PROCEDURE dbo.InsertData
    WITH NATIVE_COMPILATION, SCHEMABINDING, EXECUTE AS OWNER
AS
BEGIN ATOMIC WITH (TRANSACTION ISOLATION LEVEL = SNAPSHOT, LANGUAGE = 'us_english')
    DECLARE @Memid int = 1
    WHILE @Memid <= 500000
    BEGIN
        INSERT INTO dbo.MemoryTable VALUES (@Memid, GETDATE())
        SET @Memid += 1
    END
END;
GO
```

Xml support

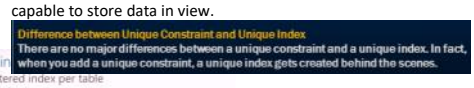


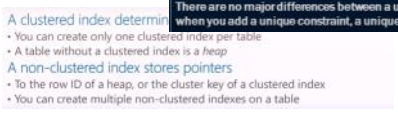
Feature	Description
xml data type	Store XML in variables and columns
XQuery	Node-tree query language for XML
XML Indexes	Optimize XQuery performance
XSD Schemas	Enforce type validation for XML data
FOR XML clause	Generate XML from relational data
OPENXML function	Generate relational data from XML

You can't use **order by** in views unless **TOP or FOR XML**



Unique index: its enforce uniqueness of key value in index like id primary key its provide uniqueness in id
INDEX on view: usually can't store data on views but if use index mean it capable to store data in view.





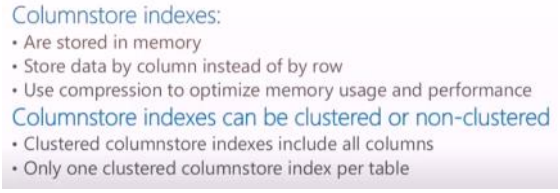
Curser : like pointer to the row and **curser return one row at a time.** We can use join instead of curser ,its use as last option

Only **one cluster index for one table**
Multiple non-cluster for one table
here Creating index on raw table

```
-- Get products and quantities for a specific order
SELECT ProductID, OrderQty
FROM dbo.SalesOrderDetail
WHERE SalesOrderID = 58125;

-- Craete a clustered index
CREATE CLUSTERED INDEX idx_SalesOrderID
ON dbo.SalesOrderDetail(SalesOrderID);

-- Try the query again
SELECT ProductID, OrderQty
FROM dbo.SalesOrderDetail
WHERE SalesOrderID = 58125;
```

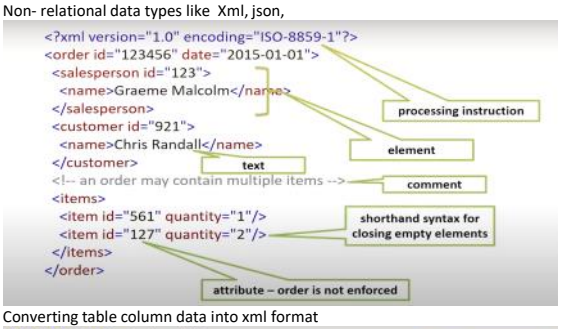
Non-cluster index

```
-- Create a nonclustered index
CREATE NONCLUSTERED INDEX idx_ProductID
ON dbo.SalesOrderDetail(ProductID);

-- Try the query again
SELECT SalesOrderID
FROM dbo.SalesOrderDetail
WHERE ProductID = 758;

-- Include a non-indexed field
SELECT SalesOrderID, OrderQty
FROM dbo.SalesOrderDetail
WHERE ProductID = 758;
```




```
ON p.ProductID = i.ProductID
WHERE p.Name = 'Rear Brakes';

-- Create NonClustered ColumnStore index
CREATE NONCLUSTERED COLUMNSTORE INDEX idx_Product
ON dbo.Product (ProductID, Name, ListPrice);

-- Query tables
SELECT p.Name, p.ListPrice, i.Shelf, i.Bin, i.Quantity
FROM dbo.Product AS p
JOIN dbo.ProductInventory AS i
ON p.ProductID = i.ProductID
WHERE p.Name = 'Rear Brakes';
```

Non- relational data types like Xml, json,



Converting table column data into xml format

```
SELECT TOP 5 LastName
FROM SalesLT.Customer
FOR XML AUTO, ROOT('Customer')

<Customer>
  <SalesLT.Customer LastName="Gee" />
  <SalesLT.Customer LastName="Harris" />
  <SalesLT.Customer LastName="Carreras" />
  <SalesLT.Customer LastName="Gates" />
  <SalesLT.Customer LastName="Harrington" />
</Customer>
```

LastName
Gee
Harris
Carreras
Gates
Harrington

https://www.youtube.com/watch?v=PxTqDm6nbQE for sql from Microsoft
Another tutorial for sql server https://www.youtube.com/watch?v=mhsp1SSiugk
Normalization

Normalizing a database

Normalization the process of organizing data in a database that includes creating tables and establishing relationships between the tables
Process is used to help eliminate redundant data
Five ***normalization forms*** (NFs)
1NF: Eliminate Repeating Groups
2NF: Eliminate Redundant Data
3NF: Eliminate Columns Not Dependent on Key
4NF: Isolate Independent Multiple Relationships
5NF: Isolate Semantically Related Multiple Relationships

What we do in normalization

The ***first normal form*** means the data is in an entity format, which means the following conditions have been met:
Eliminate repeating groups in individual tables
Create separate table for each set of related data
Identify each set of related data with primary key

Do not use multiple fields in a single table to store similar data

Data backup

Common types of backups

Full backup - contains all the data in a specific database, or set of filegroups or files, and also the portion of the transaction log necessary to recover all the data
Differential backup - contains all the data that has changed since the differential base
Incremental backup (transaction log) - contains only the data that has changed since the last full or incremental backup
Other backup types

Mysql connect with python

```
import mysql.connector
mydb =mysql.connector.connect(host="localhost", user="root", passwd="password123")

mycursor =mydb.cursor()

mycursor.execute("Show databases")

for db in mycursor:
  print(db)
```

```
import mysql.connector
mydb =mysql.connector.connect(host="localhost", user="root", passwd="password123", datab

mycursor =mydb.cursor()

sqlform = "Insert into employee(name,sal) values(%s,%s)"

employees = [("harshit", 20000), ("amit", 30000), ("ankita", 40000), ]

mycursor.executemany(sqlform, employees)

mydb.commit()
```

```
import mysql.connector
mydb =mysql.connector.connect(host="localhost", user="root", passwd="password123", datab

mycursor =mydb.cursor()
mycursor.execute("Select * from employee")

myresult =mycursor.fetchall()

for row in myresult:
    print(row)
```

Mysql Installation on Linux

sudo apt-get update From <https://www.fullstackpython.com/blog/install-mysql-ubuntu-1604.html>
sudo apt-get install mysql-server

Securing MySQL

sudo mysql_secure_installation

Creating MySQL Users

mysql -u root -p
CREATE USER 'mynewuser'@'localhost' IDENTIFIED BY 'goodPassword';
GRANT ALL PRIVILEGES ON * . * TO 'mynewuser'@'localhost';

FLUSH PRIVILEGES;

Another way of doing

```
SELECT *
FROM OPENXML (@idoc, '/ROOT/Customer',1)
WITH (CustomerID  varchar(10), ContactName varchar(20));
```

Now we talk about JSON
Converting SQL table column to Json data

```
SELECT TOP 5 LastName
FROM SalesLT.Customer
FOR JSON AUTO
```

LastName
Gee
Harris
Carreras
Gates
Harrington

```
[{"LastName":"Gee"},
{"LastName":"Harris"},
{"LastName":"Carreras"},
{"LastName":"Gates"},
{"LastName":"Harrington"}]
```

Another way of doing fo Json

```
declare @json nvarchar(255)
set @json =
N'[ null, "string", 1, true]';

select *
from OPENJSON( @json )
```

Storing JSON data in the NVARCHAR data type

And also using DocumentDB like talking about json documents
There is no schema in Json document just storing as text inside Nvachar data type in sql database

Connecting sql server2016 with pyodbc

```
import pyodbc

conn = pyodbc.connect(
    "Driver={SQL Server Native Client 11.0};"
    "Server=LAPTOP-OGHOFHSQ;"
    "Database=Test;"
    "Trusted_Connection=yes;"
```

```
def read(conn):
    print("Read")
    cursor = conn.cursor()
    cursor.execute("select * from dummy")
    for row in cursor:
        print(f'row = {row}')
    print()

conn = pyodbc.connect(
    "Driver={SQL Server Native Client 11.0};"
    "Server=LAPTOP-OGHOFHSQ;"
    "Database=Test;"
```

```
def create(conn):
    print("Create")
    cursor = conn.cursor()
    cursor.execute(
        'insert into dummy(a,b) values(?,?);',
        (3232, 'catzzz')
    )
```

```
# loop through all the drivers we have access to
for driver in pyodbc.drivers():

    print(driver)
```

```
L Server
L Server Native Client 11.0
L Server Native Client RDA 11.0
BC Driver 13 for SQL Server
SQL ODBC 8.0 ANSI Driver
SQL ODBC 8.0 Unicode Driver
BC Driver 17 for SQL Server
crosoft Access Driver (*.mdb, *.accdb)
crosoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
crosoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
crosoft Access Text Driver (*.txt, *.csv)
```

```
mysql -u root -p
CREATE USER 'mynewuser'@'localhost' IDENTIFIED BY 'goodPassword';
GRANT ALL PRIVILEGES ON * . * TO 'mynewuser'@'localhost';

FLUSH PRIVILEGES;
```

New User Connection

```
CREATE DATABASE fullstackpython;
use fullstackpython;
CREATE TABLE pages (name VARCHAR(50), url VARCHAR(1024));
```

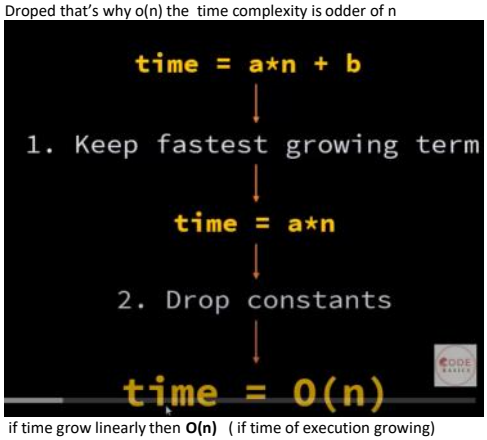
Way of maintain security in applications
es required

Data validation

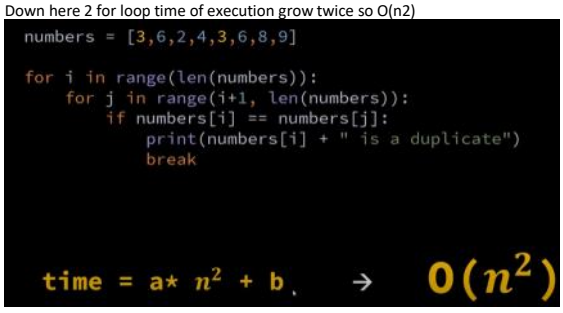
Data structure : is just a container where we can store data in specific memory layout

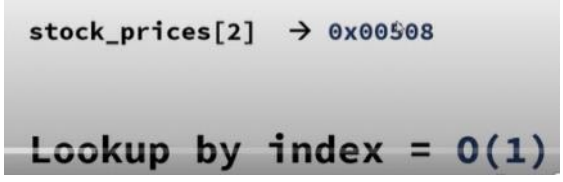
Data Structure	Python	Java	C++
Array	list	Native array ArrayList	Native array std::vector
Hash Table	dictionary	HashMap LinkedHashMap	std::map
Linked List	Not available	LinkedList	std::list

Big O work as per liner time complexity like it keep fastest growing term and drop constants so here A Droped that's why o(n) the time complexity is odder of n



if time grow linearly then **O(n)** (if time of execution growing)
If time is constants then **O(1)** (if time of execution constants)
Down here 2 for loop time of execution grow twice so O(n2)





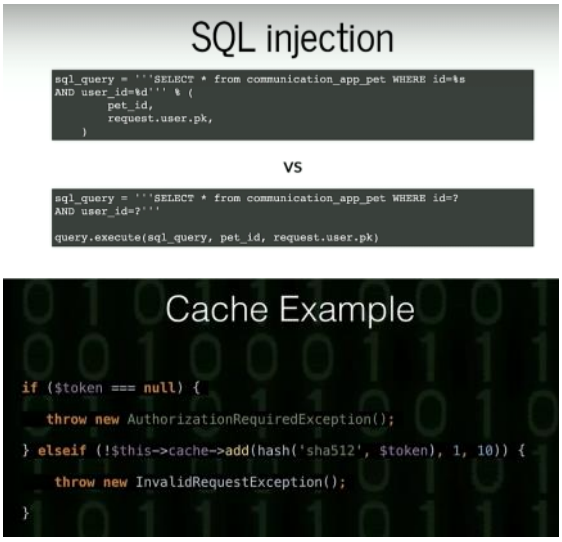
Array time complexity is **O(n)**
Insert and deletion O(n) because once we insert then other memory space(element) has to shift down an number of time same if delete memory element has shift up an number to time.

Now we will see **space complexity**
https://www.youtube.com/watch?v=gDqQf4Ekr2A&list=PLeo1K3hjS3uu_n_a__MI_KktGTLYopZ12&index=3
Array are 2 types static array and dynamic array but here as a **list** in **python use dynamic array**
In java : **static array has fix size** we can insert more than size but in **dynamic array no fix size** we can

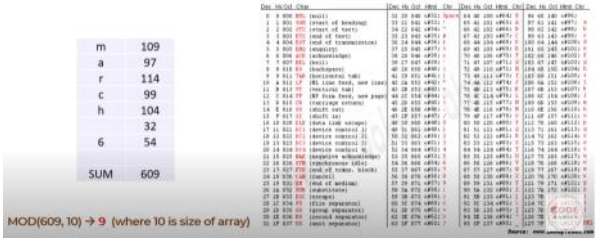
```
crosoft Access Driver (*.mdb, *.accdb)
crosoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
crosoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
crosoft Access Text Driver (*.txt, *.csv)
```

```
# define the server name and the database name
server = 'DESKTOP-MOA8O6M\SQLEXPRESS'
database = 'stock_database'

# define our connection string
cnxn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server}; \
                       'SERVER=' + server + '; \
                       'DATABASE=' + database +';\
                       'Trusted_Connection=yes;')
```

```
C:\Users\Aarya Amar>install mysql-connector
'install' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Aarya Amar>pip install mysql-connector
Collecting mysql-connector
  Downloading https://files.pythonhosted.org/packages/28/04/e40098f3730e75bbe36a338926f
/mysql-connector-2.2.9.tar.gz (11.9MB)
     |████████████████████████████████| 11.9MB 3.3MB/s
Installing collected packages: mysql-connector
  Running setup.py install for mysql-connector ... done
Successfully installed mysql-connector-2.2.9
WARNING: You are using pip version 19.1.1, however version 20.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\Aarya Amar>python -m pip install --upgrade pip
```



Hash map: **dictionary** has **string index** but array use integer index
Hashmap and HAS Table are internal data structure and in python dictionary is implement of hash table
What hash function do : there are many way of implement hash table but implement sing ASCII numb its take each of string and find out their ASCII number and sum all the ASCII number and divide it by size of array(list) and reminder store in memory as index (**basically hash function gives us index)**



ASCII stands for American Standard Code for Information Interchange. t she how hash fun work

```
class HashTable:
    def __init__(self):
        self.MAX = 100
        self.arr = [None for i in range(self.MAX)]

    def get_hash(self, key):
        h = 0
        for char in key:
            h += ord(char)
        return h % self.MAX

    def add(self,key,val):
        h = self.get_hash(key)
        self.arr[h] = val

    def get(self, key):
        h = self.get_hash(key)
        return self.arr[h]
```

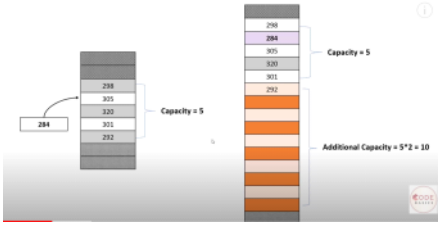
Here 100 is size of array(list)

```
class HashTable:
    def __init__(self):
        self.MAX = 100
        self.arr = [None for i in range(self.MAX)]

    def get_hash(self, key):
        h = 0
        for char in key:
            h += ord(char)
        return h % self.MAX

    def __setiten__(self,key,val):
        h = self.get_hash(key)
        self.arr[h] = val

    def __getiten__(self, key):
        h = self.get_hash(key)
        return self.arr[h]

t = HashTable()
t['march 6'] = 130
t['march 1'] = 20
```

Now we will see **space complexity**
https://www.youtube.com/watch?v=gDqQf4Ekr2A&list=PLeo1K3hjS3uu_n_a__MI_KktGTLYopZ12&index=3
Array are 2 types static array and dynamic array but here as a **list** in **python use dynamic array**
In java : **static array has fix size** we can insert more than size but in **dynamic array no fix size** we can keep adding on it if memory not available then arraylist allocate new memory space for add **example: ArrayList is dynamic array**

why we use link list instead array

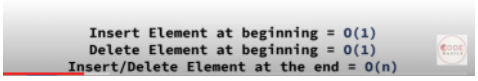
In dynamic array : when we insert value then it will **swap the value to make space** and **allocate new memory space** and **copy all the existing element one by one to new memory space from old memory space** so this is efficient way



	Array	Linked List
Contiguous memory?	Yes	No

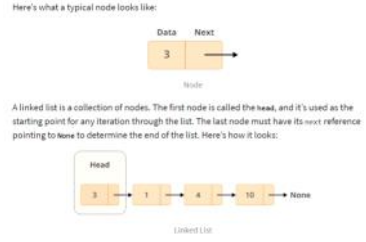
LinkList : link list where individual value store in different area of memory **linklist store value at random memory location and those are link by pointer. First element store the reference of next element**

In linklist : if you insert any value you don't have to swap and copy the value one memory space to another memory space all you do just change the address of the link and link with list it

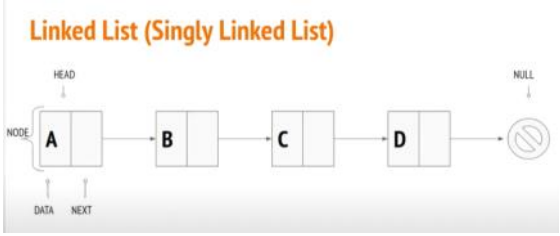


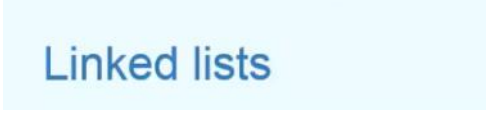
	Array	Linked List
Indexing	O(1)	O(n)
Insert/Delete Element At Start	O(n)	O(1)
Insert/Delete Element At End	O(1) - amortized	O(n)
Insert Element in Middle	O(n)	O(n)

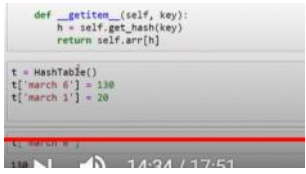
Double linklist : where we also have reference link of previous element and next element. traversing in previous direction become easy using double link list



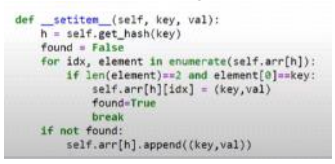
Next is pointer to the next element





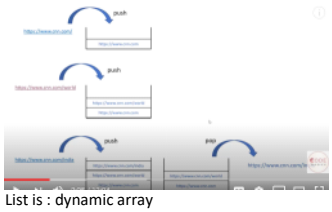


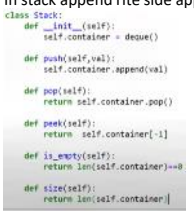
Chaining : collision handling
Multiple key can share the same hash value
Linear probing : mean **searching for empty slot to store key value pair** (if lost is not empty then goes reverse and check for empty slot)
Hash table Collision handling



First check key are available in hash table if not then Append but if exist then modify

Stack: stack is lifo data structure where we keep in pushing element and when pop it will pop out last element that we pushed (stack use LIFO(last in first out) technique)
In stack append rite side append()



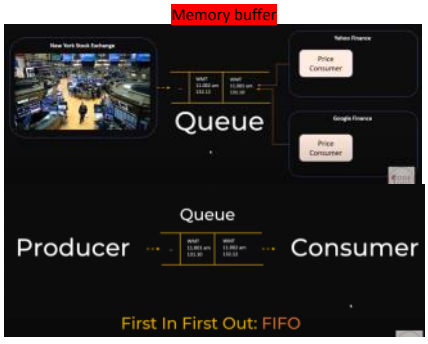


List is : dynamic array
if you have capacity of list is 10 and you want to insert 11th element then internally list go some other memory area and allocate another extra memory capacity (usually allocate current memory space*2 size of memory) then copy all the old element in new memory space then insert 11th element. So the problem here **allocating new memory and also copy all old element to new**) **using list as stack is not good**
List as stack not recommended to use
Collection.deque is recommended to use collection.deque impalemented using doubly link list
So now implement **stack using deque**
Pop() it will return element and remove element as well
Peek() it will return element and not remove element

Queue: it called FIFO data structure

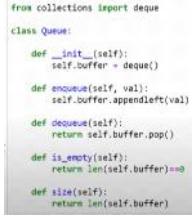
List as queue not recommended to use Collection.deque is recommended to use
suppose u getting stock data from stock change if sql server down then u missed
Posted data which posted from stock change to avoid this type of issue better use queue.

To avoid this data loose problem we can keep **memory buffer which store data in order** between data sender and receiver



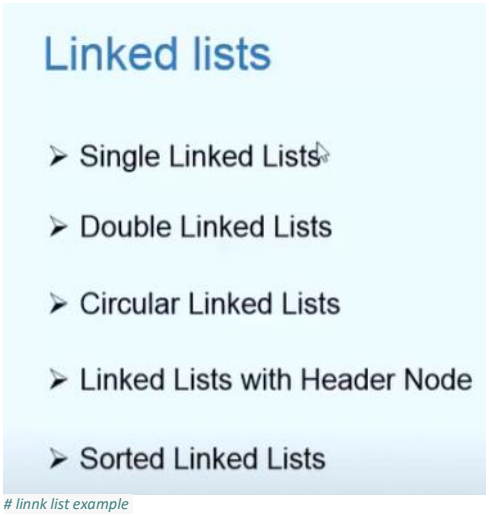
In queue always append left side appendleft()
Producer use enqueue() to place element in queue
Consumer use dequeue() to out element from queue

Producer Whatever pushed first consumer receive first



Doubly linked list : it can also track previous node as well as next node

Double linked list : where we also have reference link of previous element and next element. traversing in previous direction become easy using double link list




```
# linnk list example

class Node:
  def __init__(self, data=None):
    self.data = data
    self.next = None


class LinkedList:
  def __init__(self):
    self.start = None          # its just starting node (head) to link with other node

  def listview(self):          # print all the list
    if self.start == None:     # if list is empty
      print("list is empty nothing to print ")
    else:
      temp = self.start        # reference(first node ka data)  of start kept into temp for traverse but start still pointing first node only temp is traversing between node
                               # becz we dont want to change .self value
      while temp != None:
        print(temp.data, end= ' ')  # printing data through temp
        temp = temp.next         # temp pointing to the next value
  def deleteFirst(self):
    if self.start == None:     # if link list is empty then
      print("linked list is empty no node in Linked list")
    else:
      #temp = self.start          # if you want to keep reference of start node seprate then
                                  # you can keep in temporary vairable but no need in python
      self.start = self.start.next  # next node ka refrence start node mai aana chahiye
                                  # then start node point to next node then we can delete next node
                                   # if there is only one node then start mai Null aajayega



# now we insert new node in last so we have to create new done and pass the value by new node

  def insertLast(self, value):
    newNode = Node(value)     # created new node to add here and pass with value( created Node class object )
    if self.start == None:    # checking if there is no need inserted then insert new node
      self.start = newNode    # adding new node with start node
    else:
      temp = self.start  # temp is just like start becouse start asin to temp
      while temp.next != None:    # jabtak tem.ka next None nahi hai tab tak travase karo
        temp = temp.next  # traverse the next untill none
      temp.next = newNode  #jab none aajaye to new node add karo  becouse none node is last node so adding in last


mylist = LinkedList()
mylist.insertLast(10)
mylist.insertLast(20)
mylist.insertLast(30)
mylist.insertLast(40)
mylist.listview()
print()
mylist.deleteFirst()
mylist.listview()
input()
```

Inserting new node in front (in first)

```
class LinkedList(object):

    def __init__(self, head = None, end = None):
        self.head = head
        self.end = end

    def insert_front(self, data):

        # define a new node
        new_node = Node(data)

        # set the second node to equal the old first one
```

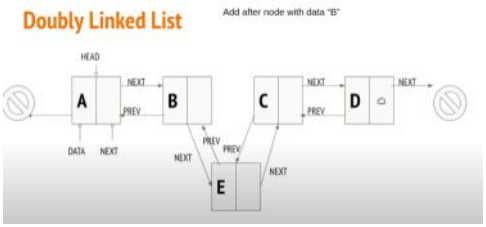

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None_
        self.prev = None

class DoublyLinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        if self.head is None:
            new_node = Node(data)
            new_node.prev = None
            self.head = new_node
        else:
            new node = Node(data)
            cur = self.head
```

```
            self.head = new_node
        else:
            new_node = Node(data)
            cur = self.head
            while cur.next:
                cur = cur.next
            cur.next = new_node
            new node.prev = cur
            new_node.next = None

    def prepend(self, data):
        pass

    def print_list(self):
        cur = self.head
        while cur:
            print(cur.data)
            cur = cur.next


dllist = DoublyLinkedList()
dllist.append(1)
dllist.append(2)
dllist.append(3)
dllist.append(4)
```

Prepends

```
def prepend(self, data):
    if self.head is None:
        new_node = Node(data)
        new_node.prev = None
        self.head = new_node
    else:
        new_node = Node(data)
        self.head.prev = new_node
        new_node.next = self.head
        self.head = new_node
        new_node.prev = None
```

Adding into middle




```
def add_after_node(self, key, data):
    cur = self.head
    while cur:
        if cur.next is None and cur.data == key:
            self.append(data)
            return
        elif cur.data == key:
            new_node = Node(data)
```

```
        # define a new node
        new_node = Node(data)

        # set the second node to equal the old first one
        new_node.set_next(self.head)

        # the new head will the be the new node
        self.head = new_node
```

```
linked_list = LinkedList()

linked_list.insert_front(35)
linked_list.insert_front(37)
```

For linklist tutorial https://www.youtube.com/watch?v=FSsriWQ0qYE

```
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def print_list(self):
        cur_node = self.head
        while cur_node:
            print(cur_node.data)
            cur_node = cur_node.next

    def append(self, data):
        new_node = Node(data)

        if self.head is None:
            self.head = new_node
            return

        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node


llist = LinkedList()
llist.append("A")
llist.append("B")
llist.print_list()
```

if Cur_node head is != None then print current node data
And move to the cur_node to next node to right one by one

If there is no any node then easily able to insert new node

While next pointer is not!=None of the node currently we are on
Continue loop until None and head move to right node

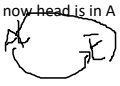
```
last_node = last_node.next
```

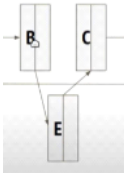
moving head pointer to right untill head_next !=None
once you reached to the null Append(insert) the new node

```
last_node.next = new_node
```

```
    def prepend(self, data):
        new_node = Node(data)

        new_node.next = self.head
        self.head = new_node


llist = LinkedList()
llist.append("A")
llist.append("B")
llist.append("C")
llist.append("D")
llist.prepend("E")

llist.print_list()
```

First created new node next E make pointer to head node A like
New_node.next = self.head now head is in A node so
move head to new node E





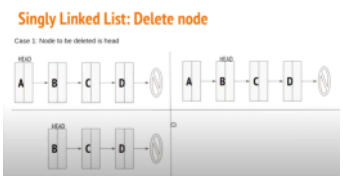
First E make link to C then B make link to E

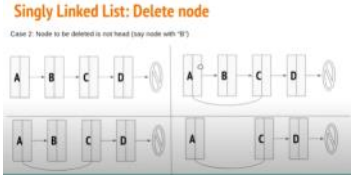
```
    def insert_after_node(self, prev_node, data):

        if not prev_node:
            print("Previous node is not in the list")
            return

        new_node = Node(data)

        new_node.next = prev_node.next
        prev_node.next = new_node

llist = LinkedList()
llist.append("A")
llist.append("B")
llist.append("C")
llist.append("D")

llist.insert_after_node(llist.head.next, "E")

llist.print_list()
```

First check previous node is present in the list or not
If NOT then show previous node not in list ,if YES then create n
And insert between then make link E created note to right side
node Then Previous node B make link to E

For delete the node (if deleting node have head)
First **check current head is available in current node or not** then **check node data is == to key or not**
First we have to **move head to the next node** then **remove the pointer from next node** and **set current node as Null**

```
def delete_node(self, key):

    cur_node = self.head

    if cur_node and cur_node.data == key:
        self.head = cur_node.next
        cur_node = None
        return
```





If No head : first we have to find out the node between privies node and next node and remove pointer from B node and set B as Null then make pointer to A to C directly

```
        if cur.next is None and cur.data == key:
            self.append(data)
            return
        elif cur.data == key:
            new_node = Node(data)
            nxt = cur.next
            cur.next = new_node
            new_node.next = nxt
            new_node.prev = cur
            nxt.prev = new_node
        cur = cur.next
```

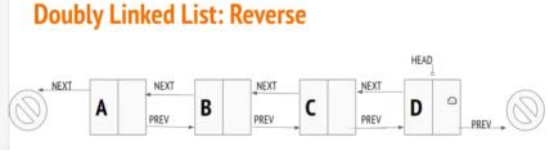
```
def add_before_node(self, key, data):
    cur = self.head
    while cur:
        if cur.prev is None and cur.data == key:
            self.prepend(data)
            return
        elif cur.data == key:
            new_node = Node(data)
            prev = cur.prev
            prev.next = new_node
            cur.prev = new_node
            new_node.next = cur
            new_node.prev = prev
        cur = cur.next
```


```
def delete(self, key):
    cur = self.head
    while cur:
        if cur.data == key and cur == self.head:
            # Case 1:
            if not cur.next:
                cur = None
                self.head = None
                return

            # Case 2:
            else:
                nxt = cur.next
                cur.next = None
                nxt.prev = None
                cur = None
                self.head = nxt
                return
```

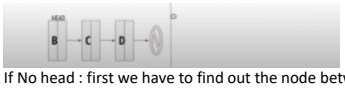
```
    elif cur.data == key:
        # Case 3:
        if cur.next:
            nxt = cur.next
            prev = cur.prev
            prev.next = nxt
            nxt.prev = prev
            cur.next = None
            cur.prev = None
            cur = None
            return


        # Case 4:
        else:
            prev = cur.prev
            prev.next = None
            cur.prev = None
            cur = None
            return
```

```
cur = cur.next
```

Reversing




```
def reverse(self):
    tmp = None
    cur = self.head
```




If No head : first we have to find out the node between privies node and next node and remove pointer from B node and set B as Null then make pointer to A to C directly
2nd : go through loop and find the data is the same which passed by Key if match then go and make deletions process

```
prev = None
while cur_node and cur_node.data != key:
    prev = cur_node
    cur_node = cur_node.next

if cur_node is None:
    return

prev.next = cur_node.next
cur_node = None
```

While current node is not null
And not != key

If current node is not present

Pointing A to C
Set as null so deleted

Delete as given position
here given position is 1
First check if given position is = 0 then move the head to next node set cur_node as None

```
def delete_node_at_pos(self, pos):

    cur_node = self.head

    if pos == 0:
        self.head = cur_node.next
        cur_node = None
        return

    prev = None
    count = 1
    while cur_node and count != pos
        prev = cur_node
        cur_node = cur_node.next
        count += 1

    if cur_node is None:
        return

    prev.next = cur_node.next
    cur_node = None
```

Reverse the linked list node print_helper show how reversing

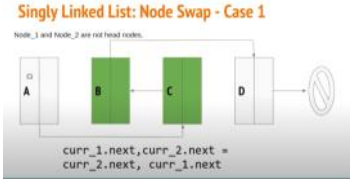
```
def print_helper(self, node, name):
    if node is None:
        print(name + ": None")
    else:
        print(name + ":" + node.data)

# A -> B -> C -> D -> 0
# D -> C -> B -> A -> 0
# A <- B <- C <- D <- 0

def reverse_iterative(self):

    prev = None_
    cur = self.head
    while cur:
        nxt = cur.next
        cur.next = prev

        self.print_helper(prev, "PREV")
        self.print_helper(cur, "CUR")
        self.print_helper(nxt, "NXT")

        prev = cur_
        cur = nxt_
    self.head = prev
```

We can reverse node 2 way

1. Iterative way
2. Recursive way

```
def reverse_recursive(self):

    def _reverse_recursive(cur, prev):
        if not cur:
            return prev

        nxt = cur.next
        cur.next = prev
        prev = cur_
        cur = nxt_
        return _reverse_recursive(cur, prev)

    self.head = reverse_recursive(cur=self.head, prev=None)
```

Swap node




```
def swap_nodes(self, key_1, key_2):

    if key_1 == key_2:
        return_

    prev_1 = None_
    curr_1 = self.head_
    while curr_1 and curr_1.data != key_1:
        prev_1 = curr_1_
        curr_1 = curr_1.next

    prev_2 = None
    curr_2 = self.head
    while curr_2 and curr_2.data != key_2:
        prev_2 = curr_2
        curr_2 = curr_2.next

    if not curr_1 or not curr_2:
        return

    if prev_1:
        prev_1.next = curr_2
    else:

    if prev_2:
        prev_2.next = curr_1
```

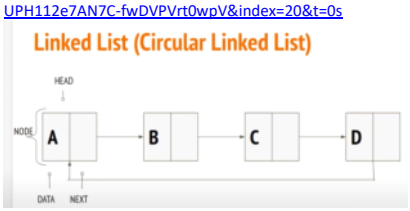
```
    if prev_1:
        prev_1.next = curr_2
    else:
        self.head = curr_2

    if prev_2:
        prev_2.next = curr_1
    else:
        self.head = curr_1

    curr_1.next, curr_2.next = curr_2.next, curr_1.next

llist = LinkedList()
llist.append("A")
llist.append("B")
llist.append("C")
llist.append("D")

llist.swap_nodes("B", "C")

llist.print_list()

#print(llist.len_recursive(llist.head))
```

How many times appear this number in list find out (can be done by 2 way iterative and recursive

```
def count_occurences_iterative(self, data):

    count = 0
    cur = self.head_
    while cur:
        if cur.data == data:
            count += 1
        cur = cur.next
    return count

def count_occurences_recursive(self, node, data):
    if not node:
        return 0
```

```
def reverse(self):
    tmp = None
    cur = self.head
    while cur:
        tmp = cur.prev
        cur.prev = cur.next
        cur.next = tmp
        cur = cur.prev
    if tmp:
        self.head = tmp.prev
```

circular link list
https://www.youtube.com/watch?v=5WoNhm7sOnA&list=PL5tcWHG-UPH112e7AN7C-fwDVPVrt0wpV&index=20&t=0s

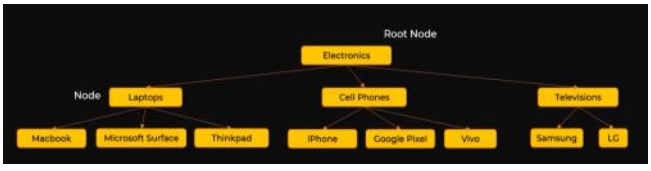



```
class CircularLinkedList:
    def __init__(self):
        self.head = None_

    def prepend(self, data):
        pass

    def append(self, data):
        if not self.head:
            self.head = Node(data)
            self.head.next = self.head
        else:
            new_node = Node(data)
            cur = self.head
            while cur.next != self.head
                cur = cur.next
            cur.next = new_node
            new_node.next = self.head
```

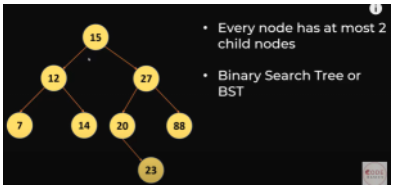
Remove circular node

```
def remove(self, key):
    if self.head.data == key:
        cur = self.head
        while cur.next != self.head:
            cur = cur.next
        cur.next = self.head.next
        self.head = self.head.next
```

Tree : tree is recursive data structure
Jeneral tree (mustly use in ecommerce kind of application)



Binary Tree is just like Set data structure in binary tree small value always should be in left
And big value should be in right side. Always traverse(visit) left then right . All element shold Not be duplicated
Basic properties of binary tree search : element should be unique, small value always should be in left of the current node value And **big value should be in right side of the current node value.**



In order traversal : if you put **root node between left and right** tree
Pre order traversal : if you put **root node Before left and right t**ree
Post order traversal :if you put **root node after left and right t**ree

- Inorder Traversal (Left-Root-Right)
- Preorder Traversal (Root-Left-Right)
- Postorder Traversal (Left-Right-Root)

Left node data should be **less than current data** and **right side should be more than current data**
Delete while 1 child of the node: remove it and move child to top
Delete while 2 child of the node :
1st approach : Find minimum value from right subtree, Copy that element from right and delete

```
            count += 1
        cur = cur.next
    return count

def count occurences_recursive(self, node, data):
    if not node:
        return 0
    if node.data == data:
        return 1 + self.count occurences recursive(node.next, data)
    else:
        return self.count_occurences_recursive(node.next, data)


1 -> 2 -> 1 -> 3 -> 1 -> 4 -> 1
Number of ones: 4
```

Left node data should be **less than current data** and **right side should be more than current data**
Delete while 1 child of the node: remove it and move child to top
Delete while 2 child of the node :
1st approach : Find minimum value from right subtree ,Copy that element from right and delete
2nd approach: find the maximum value from left subtree Copy that element delete

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	O(1)	O(1)	O(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

mongoDB : is documents based database and document is just JSON content we store data
On Json structure unlike sql store data in table structure.
https://docs.mongodb.com/manual/reference/sql-comparison/ mapping chart
https://www.mongodb.com/compare/mongodb-mysql
For manual look https://docs.mongodb.com/manual/aggregation
https://docs.mongodb.com/manual/core/gridfs/ to handle binary file like vedio,audio file
https://dzone.com/refcardz/mongodb?chapter=1 for learn
https://university.mongodb.com/courses/catalog free video for mongodb
Mongo in Linux : https://docs.mongodb.com/manual/reference/mongo-shell/
Transitional sql -to mongo
https://www.mongodb.com/presentations/webinar-transitioning-sql-mongodb

Need to know

https://docs.mongodb.com/manual/faq/ for questions
https://mlab.com/ for free hosting db
To Generate data : http://generatedata.com/

Triggers types

Triggers

In SQL server there are 3 types of triggers
1. DML triggers
2. DDL triggers
3. Logon trigger

DML triggers are fired automatically in response to DML events (INSERT, UPDATE & DELETE)

DML triggers can be again classified into 2 types
1. After triggers (Sometimes called as FOR triggers)
2. Instead of triggers

After triggers, fires after the triggering action. The INSERT, UPDATE, and DELETE statements, causes an after trigger to fire after the respective statements complete execution.

INSTEAD of triggers, fires instead of the triggering action. The INSERT, UPDATE, and DELETE statements, causes an INSTEAD OF trigger to fire INSTEAD OF the respective statement execution.

Transaction : is a group of command that change store data its consider as single unit

Transactions

What is a Transaction?
A transaction is a group of commands that change the data stored in a database. A transaction, is treated as a single unit. A transaction ensures that, either all of the commands succeed, or none of them. If one of the commands in the transaction fails, all of the commands fail, and any data that was modified in the database is rolled back. In this way, transactions maintain the integrity of data in a database.

Transaction processing follows these steps:
1. Begin a transaction.
2. Process database commands.
3. Check for errors.
   If errors occurred,
   rollback the transaction,
   else,
   commit the transaction

Note: NOT able to see the un-committed changes
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
Isolation Levels : Covered in a later session

If you use **begintransaction** its not effect immediately in to database until **commit transaction** so we can undo our code using **rollback transaction**

```
Create Procedure spUpdateInventory_and_Sell
a
Begin
 Begin Try
  Begin Transaction
   Update tblProduct
   set QtyAvailable = (QtyAvailable - 10)
   where ProductId = 1

   Insert into tblProductSales values(3,1,10)
  Commit Transaction
 End Try
 Begin Catch
   Rollback Transaction
 End Catch
End
```

When sailing good first update inventory table updating with - qty then insert into product sale table
ACID Test: **Atomic consistent isolated durable** test

Blocking v/s Deadlocking

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

Blocking : Occurs if a transaction tries to acquire an incompatible lock on a resource that another transaction has already locked. The blocked transaction remains blocked until the blocking transaction releases the lock.

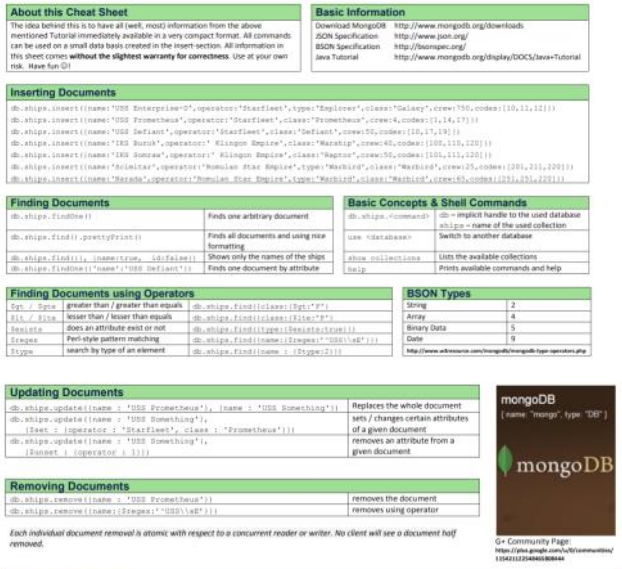
Table A		Table B	
Id	Name	Id	Name
1	Mark	1	Mary

Example:

```
--Transaction 1
Begin Tran
Update TableA set
Name='Mark Transaction 1' where Id = 1
Waitfor Delay '00:00:10'
```

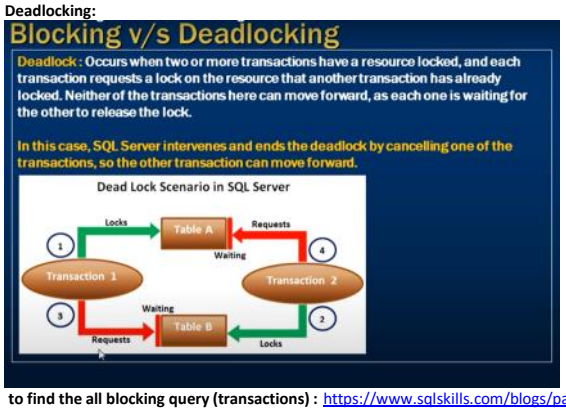
```
--Transaction 2
Begin Tran
Update TableA set
Name='Mark Transaction 2' where Id = 1
Commit Transaction
```

Cheat sheet for mongo



Transaction 2 can't modify the same table until 1st transaction commit becouse 1st transaction acquire block until commit.

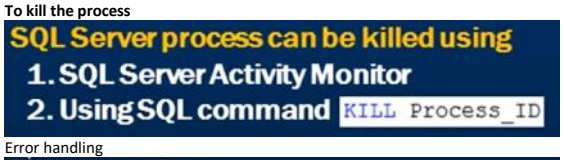
Deadlocking:



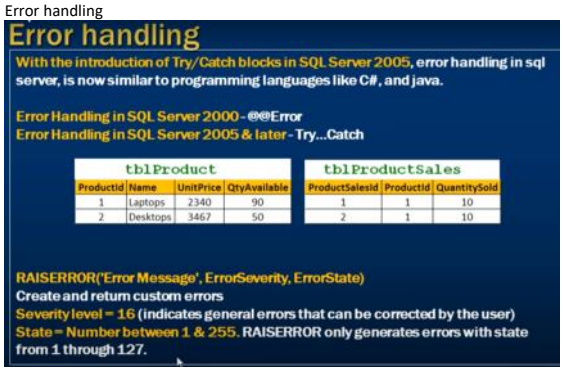
to find the all blocking query (transactions) : https://www.sqlskills.com/blogs/paul/script-open-transactions-with-text-and-plans/

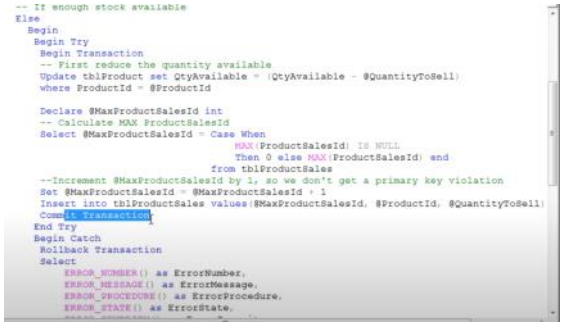
Another way to find use : DBCC OpenTran

To kill the process

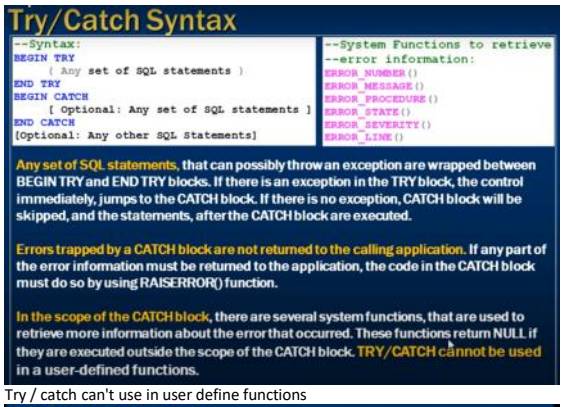


Error handling

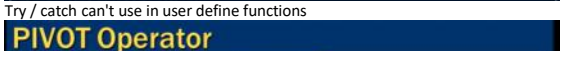


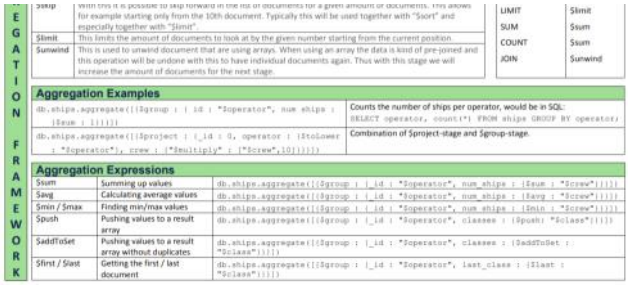


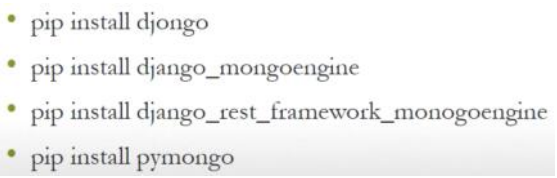
```
-- If enough stock available
Else
  Begin
   Begin Try
    Begin Transaction
    -- First reduce the quantity available
    Update tblProduct set QtyAvailable = (QtyAvailable - @QuantityToSell)
    where ProductId = @ProductId

    Declare @MaxProductSalesId int
    -- Calculate MAX ProductSalesId
    Select @MaxProductSalesId = Case When
                                  MAX(ProductSalesId) IS NULL
                                  Then 0 else MAX(ProductSalesId) end
                                from tblProductSales
    --Increment @MaxProductSalesId by 1, so we don't get a primary key violation
    Set @MaxProductSalesId = @MaxProductSalesId + 1
    Insert into tblProductSales values(@MaxProductSalesId, @ProductId, @QuantityToSell)
    Commit Transaction
   End Try
   Begin Catch
    Rollback Transaction
    Select
        ERROR_NUMBER() as ErrorNumber,
        ERROR_MESSAGE() as ErrorMessage,
        ERROR_PROCEDURE() as ErrorProcedure,
        ERROR_STATE() as ErrorState,
```



Try / catch can't use in user define functions



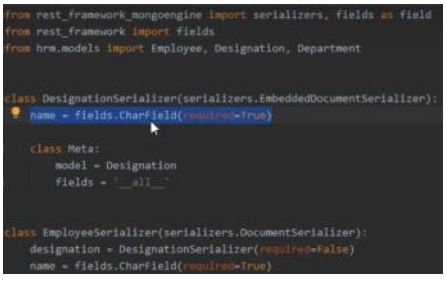




Inside setting.py

```
    'rest_framework',
    'rest_framework_mongoengine',
    'django_mongoengine',
    'django_mongoengine.mongo_auth',
    'django_mongoengine.mongo_admin',
```

Serializers for rest in django



```
from rest_framework_mongoengine import serializers, fields as field
from rest_framework import fields
from hrm.models import Employee, Designation, Department


class DesignationSerializer(serializers.EmbeddedDocumentSerializer):
    name = fields.CharField(required=True)

    class Meta:
        model = Designation
        fields = '__all__'


class EmployeeSerializer(serializers.DocumentSerializer):
    designation = DesignationSerializer(required=False)
    name = fields.CharField(required=True)
```

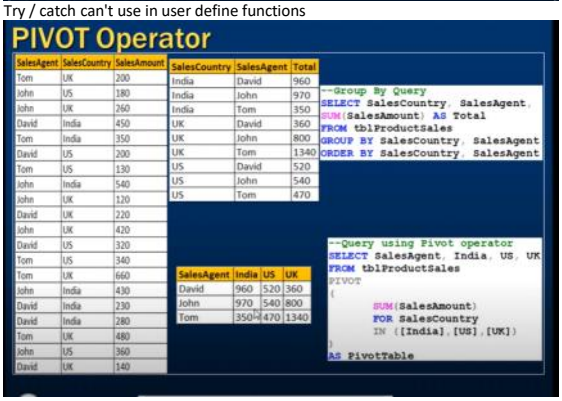
Encrypto in django

Complete code for encryption_util.py is below.

```
from cryptography.fernet import Fernet
import base64
import logging
import traceback
from django.conf import settings
#this is your "password/ENCRYPT_KEY". keep it in settings.py file
#key = Fernet.generate_key()
def encrypt(txt):
    try:
        # convert integer etc to string first
        txt = str(txt)
        # get the key from settings
        cipher_suite = Fernet(settings.ENCRYPT_KEY) # key should be byte
        # #input should be byte, so convert the text to byte
        encrypted_text = cipher_suite.encrypt(txt.encode('ascii'))
        # encode to urlsafe base64 format
        encrypted_text = base64.urlsafe_b64encode(encrypted_text).decode("ascii")
        return encrypted_text
    except Exception as e:
        # log the error if any
        logging.getLogger("error_logger").error(traceback.format_exc())
        return None
def decrypt(txt):
    try:
        # base64 decode
        txt = base64.urlsafe_b64decode(txt)
        cipher_suite = Fernet(settings.ENCRYPT_KEY)
        decoded_text = cipher_suite.decrypt(txt).decode("ascii")
        return decoded_text
    except Exception as e:
        # log the error
        logging.getLogger("error_logger").error(traceback.format_exc())
        return None
```

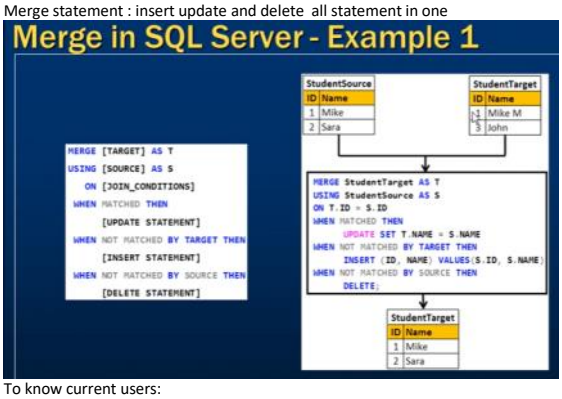
From <https://morioh.com/p/4f5288b77c14>

Try / catch can't use in user define functions



It represent as cross tab format

Merge statement : insert update and delete all statement in one



To know current users:

select is_user_process, original_login_name
from sys.dm_exec_sessions order by login_time desc;

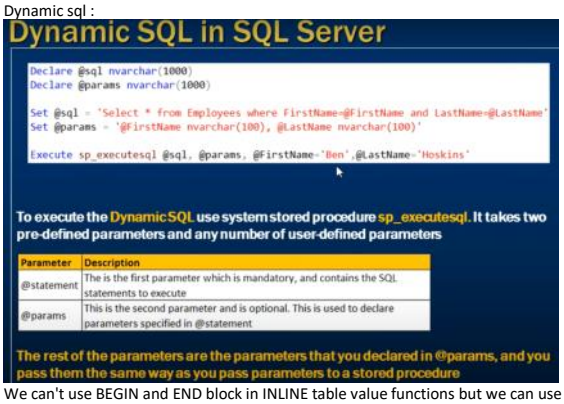
You can set up logon triger in connection like

You can't open 4th time of new query connection

```
Create trigger tr_AuditLogin
ON ALL SERVER
FOR LOGON
AS
BEGIN
    Declare @LoginName nvarchar(100)
    Set @LoginName = Original_Login()

    IF(Select Count(*) from sys.dm_exec_sessions
    where is_user_process = 1 AND
    original_login_name = @LoginName) > 3
    BEGIN
        Print 'Fourth connection attempt by ' + @LoginName + ' blocked'
        ROLLBACK;
    END
END
```

You can read all error log : using Execute sp_readerrorlog

Dynamic sql :



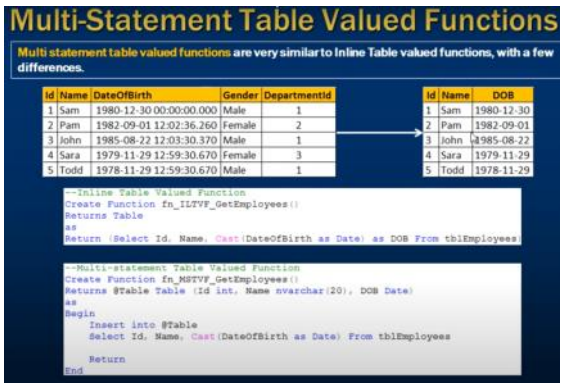
We can't use BEGIN and END block in INLINE table value functions but we can use in multi statements table functions

```
    except Exception as e:
        # log the error
        logging.getLogger("error_logger").error(traceback.format_exc())
        return None
```

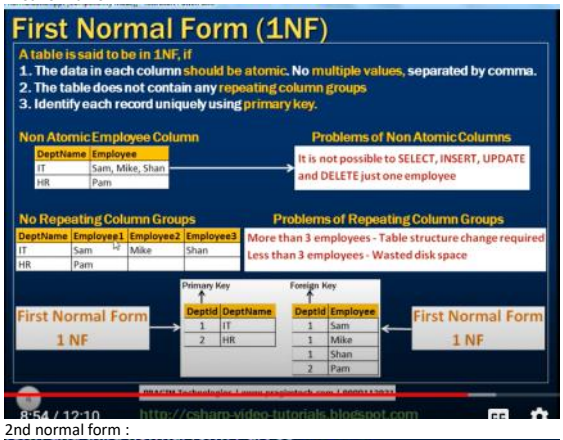
From <https://morioh.com/p/4f5288b77c14>

How can I create an encrypted django field that converts data when it's retrieved from the database?

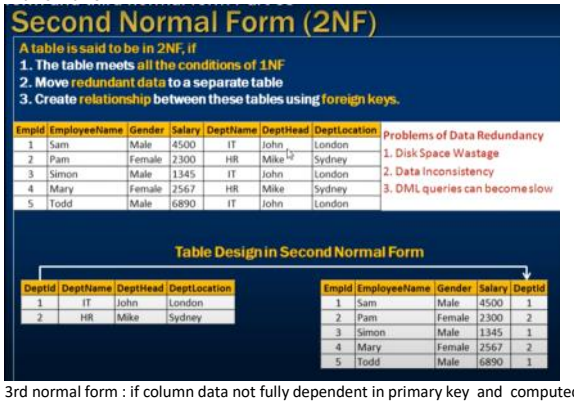
https://stackoverflow.com/questions/13077109/how-can-i-create-an-encrypted-django-field-that-converts-data-when-its-retrieve



First normal form:



2nd normal form :



3rd normal form : if column data not fully dependent in primary key and computed data then keep in separate table those column data.