When to use

* Process
* Thread
* Asyncio

ASGI (Asynchronous Server Gateway Interface)

* To maintain real-time connection b/w client and server.
* It is based on web sockets, which can be used for real time communication without reloading the page.
* It handles both HTTP & Web Socket requests.
* To use ASGI we need to use Daphini while deployment.
* It is called as stateful protocol.
* Eg. WhatsApp, excel online, etc..

WSGI (Web Server Gateway Interface)

* It only handles HTTP request.
* To use ASGI we need to use Gunicorn while deployment.
* It is also called as connectionless protocol.

Django ORM:

>>> python manage.py shell

>>> from home.models import \*

* Create
  + To create any new entry in the DB table you have two ways to do it

1. Author.object.create(author\_name='Alok Sharma')
2. >>> author = Author(author\_name = 'Rani Sharma')

>>> author.save()

* Read
  + To read entries from database
    - >>> authors = Author.objects.all()
    - >>> for author in authors:
      * >>> print(author.author\_name) #It will print names of all the author in the DB table.
    - >>> authors = Author.objects.all().order\_by(‘author\_name’)
    - >>> author = Author.objects.get(id = 2) #It will return a single object from the DB, if multiple objects with the same id exists then it will return an error.
* Update
  + To update objects in the DB.
    - >>> author = Author.objects.get(author\_name = ‘Alka Sharma’) # First we have to fetch the item for which we have to update the value
    - >>> author.author\_name = “Alka Sharma (Elder Sister)” # updated the value of the name.
    - >>> author.save() # To save the update in the DB.
* Delete
  + To delete records from the DB
    - >>> author = Author.objects.get(author\_name = ‘Rani Sharma’)
    - >>> author.delete() # It deletes only a single item from DB
    - >>> Author.objects.all().delete() #It will delete all the records from the DB table.

How to save data in the foreign key table:

* >>> author = Author.objects.get(id = 2)
* >>> products.objects.create(product\_name = ‘ABC’, author = author) #It will create/insert a record in the db
* >>> products.objects.save() #This will either update the record or insert/create a new record, depending on where you call it.
* >>> products.objects.get\_or\_create() #This will either return the record or insert a new record. If the passed parameter value exists in the table, then it will return that record, if that value is not present then it will insert a new record.

ORM Operation & functions

* Aggregate (When we have to do any operation on entire column)
  + >>> Products.objects.count() #It will return count of all the records in the DB table.
  + >>> from django.db.models import Avg, Sum, Min, Max, Count, Q
  + >>> products.objects.aggregate(price = Avg(‘price’)) #It will return average price of all the books in DB table.
  + >>> products.objects.aggregate(price = Sum(‘price’)) #It will return sum of price of all the books in DB table.
  + Similarly, you can see min and max of price in the DB table.
* Annotate (When we have to do any operation group wise in the column, it is similar to GROUP BY function in SQL)
  + >>> Author.objects.annotate(total\_books = Count(‘book’)) # It will return total books written by each author, you can do a for loop to print out Author wise count.
  + >>> Author.objects.annotate(Avg\_books = Avg(‘book\_\_price’)) # It will find out avg price of books under each author.
  + >>> Author.objects.annotate(Avg\_books = Avg(‘book\_\_price’), total\_books = Count(‘book’)) # It will calculate avga nd total count author wise.
  + >>> Author.objects.annotate(book\_count = Count(‘book’, filter=Q(book\_\_published\_date\_\_year\_gte = 2023)).filter(book\_\_count\_\_gte = 1) # It will return count of books for each author from 2023 year or later. It will return only those authors who has at least one book matching the criteria.
* Subquery
  + >>> from django.db.models import Subquery, OuterRef
  + >>> book = Book.objects.filter(author = OuterRef(‘id’)).order\_by(‘-published\_date’).values(‘book\_name’)[:1]
  + >>> authors = Author.objects.anootate(books = Subquery(book))
  + >>> for author in authors:
    - >>> print(f”Author Name {author.author\_name} and recent book is {author.books}”)
  + # Above subquery will return recent book published bye ach of of the author (‘-published\_date’) means that it will go in descending order. OuterRef will only work inside a Subquery.
  + >>> book = Book.objects.filter(author = OuterRef(‘id’), published\_date\_\_year = 2023).values(‘author’).annotate(total\_price = Sum(‘price’)).values(‘total\_price’) #It will return sum of price of all the books published by each author in the year 2023. Following this we must apply a subquery as above.

Full Text Search (FTS):

* It can be used as an alternative to elastic search.
* Elastic search needs minimum of 8GB of system memory to work, but you can use FTS, with 2GB RAM is required.
* FTS uses combination of SearchVector, SearchQuery, SearchRank, TrigramSimilarity, to create a very similar search as you would get using elastic search. But the accuracy of FTS is behind Elastic Search.
* Note: Go through the TrigramSimilarity Video once again.

Admin Panel

* You can create admin credential using
  + >>>python manage.py createsuperuser # It will ask for username and password