# SI 664

## meeting 6

9 Oct 2018

Anthony Whyte
arwhyte@umich.edu
twitter: @arwhyte
University of Michigan



UMSI

# preliminaries

# Midterm: date / time

Tuesday
23 October 2018
1:00 - 2:50 pm
NQ 2245

# Midterm: prerequisites

A *functioning* heritagesites app connected to an *updated* MySQL unesco_heritage_sites database

# Midterm: themes

- SQL statements
- Django ORM queries
- Django Models, Views and Templates

open app, open book



UMSI

# Midterm: warning

Do NOT wait until the evening before the exam to contact arwhyte about an app/db issue.

office hours (NQ 1274)
11 Oct, 10:00 am - 12:00 pm
18 Oct, 10:00 am - 12:00 pm

special "office" hours (NQ 1245)
16 Oct, 1:00 pm - 2:50 pm

special "office" hours (NQ 2244)
16 Oct, 4:00 pm - 5:20 pm

# SQL/ORM hack session

session will be scheduled

after the midterm

# meeting 5 assignment

# Model: no trailing _id on FK field names

```
class Location(models.Model):
    location_id = models.AutoField(primary_key=True)
    planet = models.ForeignKey('Planet', models.DO_NOTHING)
    region = models.ForeignKey('Region', models.DO_NOTHING, blank=True, null=True)
    sub_region = models.ForeignKey('SubRegion', models.DO_NOTHING, blank=True, null=True)
    intermediate_region = models.ForeignKey('IntermediateRegion', models.DO_NOTHING, blank=True, null=True)

    class Meta:
        managed = False
        db_table = 'location'
        ordering = ['planet', 'region', 'sub_region', 'intermediate_region']
        verbose_name = 'UNSD M49 Location Hierarchy'
        verbose_name_plural = 'UNSD M49 Location Hierarchies'

    def __str__(self):
        if self.intermediate_region:
            return self.intermediate_region.intermediate_region_name
        elif self.sub_region:
            return self.sub_region.sub_region_name
        elif self.region:
            return self.region.region_name
        elif self.planet:
            return self.planet.unsd_name
        else:
            return 'error'
```

# Exercise: SQL

## One solution

```
SELECT r.region_name, sr.sub_region_name, ca.country_area_name,
        hs.site_name, hs.area_hectares
  FROM heritage_site hs
        LEFT JOIN heritage_site_jurisdiction hsj
              ON hs.heritage_site_id = hsj.heritage_site_id
        LEFT JOIN country_area ca
              ON hsj.country_area_id = ca.country_area_id
        LEFT JOIN location l
              ON ca.location_id = l.location_id
        LEFT JOIN region r
              ON l.region_id = r.region_id
        LEFT JOIN sub_region sr
              ON l.sub_region_id = sr.sub_region_id
WHERE r.region_name LIKE 'Africa'
ORDER BY hs.area_hectares DESC LIMIT 1;
```

# Exercise: SQL

## Another solution

```
SELECT r.region_name AS `region`, sr.sub_region_name AS `subregion`,
        ca.country_area_name AS `country / area`, hs.site_name AS `heritage site`,
        hs.area_hectares AS `area (hectares)`
FROM heritage_site hs
    LEFT JOIN heritage_site_jurisdiction hsj
        ON hs.heritage_site_id = hsj.heritage_site_id
    LEFT JOIN country_area ca
        ON hsj.country_area_id = ca.country_area_id
    LEFT JOIN location l
        ON ca.location_id = l.location_id
    LEFT JOIN region r
        ON l.region_id = r.region_id
    LEFT JOIN sub_region sr
        ON l.sub_region_id = sr.sub_region_id
WHERE TRIM(r.region_name) = 'Africa'
    AND hs.area_hectares = (SELECT MAX(hs1.area_hectares)
                            FROM heritage_site hs1
                                LEFT JOIN heritage_site_jurisdiction hsj1
                                    ON hs1.heritage_site_id = hsj1.heritage_site_id
                                LEFT JOIN country_area ca1
                                    ON hsj1.country_area_id = ca1.country_area_id
                                LEFT JOIN location l1
                                    ON ca1.location_id = l1.location_id
                                LEFT JOIN region r1
                                    ON l1.region_id = r1.region_id
                                LEFT JOIN sub_region sr1
                                    ON l1.sub_region_id = sr1.sub_region_id
                            WHERE TRIM(r1.region_name) = 'Africa');
```

# Exercise: SQL

## Answer

```
*************************** 1. row ***************************
        region: Africa
     subregion: Sub-Saharan Africa
country / area: Niger
 heritage site: Air and Ténéré Natural Reserves
area (hectares): 7736000
1 row in set (0.00 sec)
```

# which query is faster?

# Exercise: ORM

## Slide hint

### Developing/Developed status in Southern Africa (intermediate region)

```
from heritagesites.models import Location
from django.db.models import F
from django.db.models import Count

loc = Location.objects
.values(sub_region_name = F('sub_region__sub_region_name'),
                                    dev_status = F('countryarea__dev_status__dev_status_name'))
.annotate(count=Count('countryarea__dev_status__dev_status_name'))
.filter(sub_region__sub_region_name = 'Sub-Saharan Africa')
.order_by('countryarea__dev_status__dev_status_name')

print(loc)
<QuerySet [{'sub_region_name': 'Sub-Saharan Africa', 'dev_status': 'Developing', 'count': 53}]>
```

# ORM: annotate (GROUP BY), F object

## Return count of developed vs developing countries/areas in Sub-Saharan Africa

```
>>> from heritagesites.models import Location, Region, CountryArea, DevStatus
>>> from django.db.models import Count
>>> from django.db.models import F

>>> loc = Location.objects
.values(sub_region_name = F('sub_region__sub_region_name'), dev_status =
F('countryarea__dev_status__dev_status_name'))
.annotate(count=Count('countryarea__dev_status__dev_status_name'))
.filter(sub_region__sub_region_name = 'Sub-Saharan Africa')
.order_by('countryarea__dev_status__dev_status_name')

>>> for l in loc:
...     print(l)
...
...
{'sub_region_name': 'Sub-Saharan Africa', 'dev_status': 'Developing', 'count': 53}
```

# Exercise: ORM

## Actual Problem

### Developing/Developed status in Asia (region)

```
from heritagesites.models import Location
from django.db.models import F
from django.db.models import Count

loc1 = Location.objects
.values(region_name = F('region__region_name'),
        dev_status = F('countryarea__dev_status__dev_status_name'))
.annotate(count=Count('countryarea__dev_status__dev_status_name'))
.filter(region__region_name = 'Asia')
.order_by('countryarea__dev_status__dev_status_name')

print(loc1)
<QuerySet [{'region_name': 'Asia', 'dev_status': 'Developed', 'count': 3},
           {'region_name': 'Asia', 'dev_status': 'Developing', 'count': 47}]>
```

# heritagesites app

# heritagesites: sites list (with pagination)

SI664 Heritage Sites  Sites  About

## UNESCO Heritage Sites

« 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 »

- Lake Turkana National Parks
- Lakes of Ounianga
- Lamu Old Town
- Land of Frankincense
- Landscape of Grand Pré
- Landscape of the Pico Island Vineyard Culture
- Landscapes of Dauria
- Laponian Area
- Las Médulas
- Late Baroque Towns of the Val di Noto (South-Eastern Sicily)
- Laurisilva of Madeira
- Lavaux, Vineyard Terraces
- Le Havre, the City Rebuilt by Auguste Perret
- Le Morne Cultural Landscape
- Lednice-Valtice Cultural Landscape
- Lena Pillars Nature Park
- León Cathedral
- Levoča, Spišský Hrad and the Associated Cultural Monuments
- Levuka Historical Port Town
- Lines and Geoglyphs of Nasca and Palpa
- Litomyšl Castle
- Liverpool – Maritime Mercantile City
- Longmen Grottoes
- Longobards in Italy. Places of the Power (568-774 A.D.)
- Lord Howe Island Group

# heritagesites: site detail

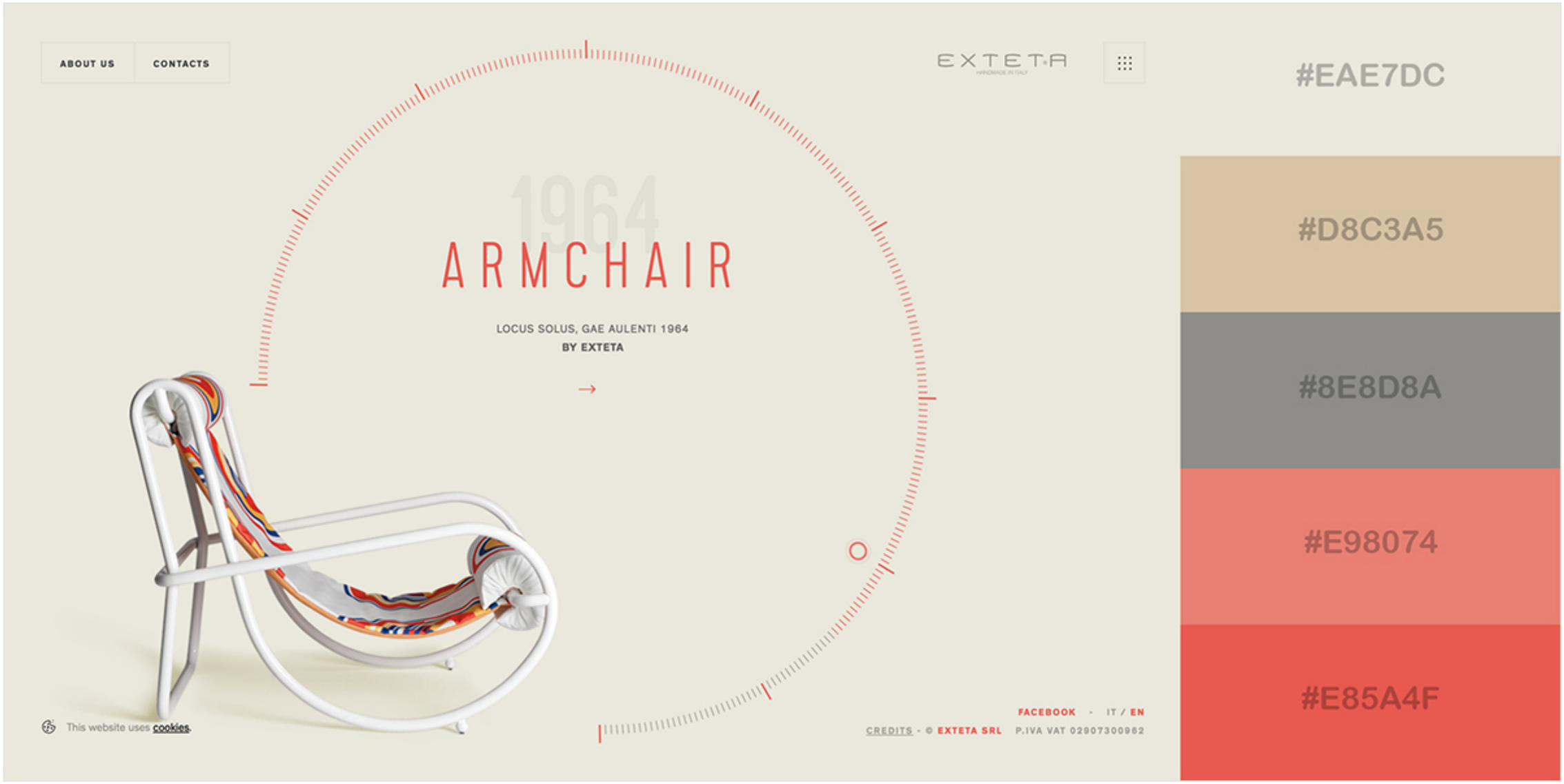
SI664 Heritage Sites  Sites  About

## Lake Turkana National Parks

| | |
|---|---|
| Category | Natural |
| Description | The most saline of Africa's large lakes, Turkana is an outstanding laboratory for the study of plant and animal communities. The three National Parks serve as a stopover for migrant waterfowl and are major breeding grounds for the Nile crocodile, hippopotamus and a variety of venomous snakes. The Koobi Fora deposits, rich in mammalian, molluscan and other fossil remains, have contributed more to the understanding of paleo-environments than any other site on the continent. |
| Justification | The Committee inscribed this property on the basis of natural *criteria (viii)* and *(x)* for the discoveries of mammal fossil remains in the site which led to the scientific reconstruction of the palaeo-environment of the entire Turkana Lake basin of the Quarternary Period. The Lake Turkana ecosystem with its diverse bird life and desert environment offers an exceptional laboratory for studies of plant and animal communities. |
| Date inscribed | 1997 |
| Geo coordinates | 3.05130556, 36.50366667 (*lat.*, *long.*) |
| Area | 161485.0 hectares |

# CSS: color palette

## Beg, Borrow . . .




https://blog.visme.co/website-color-schemes/



UMSI

# Django "sausage making" pattern

View — Template — URL

Model

# views

# Views: TemplateView

## Function and class views

```
def index(request):
    return HttpResponse("Hello, world.")


class AboutPageView(generic.TemplateView):
    template_name = 'heritagesites/about.html'
```

# View: DetailView

```
class SiteDetailView(generic.DetailView):
    model = HeritageSite
    context_object_name = 'site'
    template_name = 'heritagesites/site_detail.html'
```

# Views: ListView

```
class SiteListView(generic.ListView):
    model = HeritageSite
    context_object_name = 'sites'
    template_name = 'heritagesites/site.html'
    paginate_by = 50
```

```
def get_queryset(self):          <-- override
    return HeritageSite.objects
                .all()\
                .select_related('heritage_site_category')\
                .order_by('site_name')
```

# templates

# Template: dot-lookup syntax

Django Team

"The template system uses dot-lookup syntax to access variable attributes. In the example of {{ question.question_text }}, first Django does a dictionary lookup on the object question. Failing that, it tries an attribute lookup – which works, in this case. If attribute lookup had failed, it would've tried a list-index lookup.

Method-calling happens in
the {% for %} loop: question.choice_set.all is interpreted as the Python codequestion.choice_set.all(), which returns an iterable of Choice objects and is suitable for use in the {% for %} tag."

# Templates: location (default)

## Django loves these path hierarchies

```
heritagesites/                  <-- project
    heritagesites/              <-- app
        templates/
            heritagesites/
                about.html
                base.html
                home.html
                site.html
                site_detail.html
mysite/
...
```

# Templates: location (settings.py)

App level (default)

```
TEMPLATES = [
    {
        …
        'APP_DIRS': True,
    }
]
```

Project level

```
TEMPLATES = [
    {
        …
        'DIRS' = ['templates'],
    }
]
```

# Template: base

## load static assets (.css, .png, etc.)

{% load static %}

# Template: base

## content blocks

child templates inject content via inheritance

```
<main>
  <div class="container-fluid">
    {% block content %}

    {% endblock content %}
  </div>
</main>
```

# Template: inheritance

## extends directive

```
{% extends 'heritagesites/base.html' %}

{% block content %}

<h2>About UNESCO Heritage Sites</h2>

<p>This site combines UNSD M49 …</p>

{% endblock content %}
```

# Template: extending

```
{% extends 'heritagesites/base.html' %}

{% block content %}

<h1>UNESCO Heritage Sites</h1>

{% if sites %}
  <ul>
    {% for site in sites %}
    <!-- safe filter on for raw HTML stored in database -->
    <li><a href="{% url 'site_detail' site.pk %}">{{ site.site_name | safe }}</a></li>
    {% endfor %}
  </ul>
{% else %}
  <p>No Heritage Sites are available to view.</p>
{% endif %}

{% endblock content %}}
```

# Raw HTML: safe filter

## Handling raw HTML stored in the database

```
<!-- safe filter on for raw HTML stored in database -->

<h1>{{site.site_name | safe}}</h1>

{{site.description | safe}}

{% if site.justification %}
  {{site.justification | safe}}
{% endif %}

<p>{{site.date_inscribed}}</p>
```

# static files

# Static: location (default)

Simpler path than the templates path

```
heritagesites/                  <-- project
    heritagesites/              <-- app
        static/
            css/
                heritagesites.css
mysite/
...
```

https://docs.djangoproject.com/en/2.1/howto/static-files/

# running tests

# Tests: unmanaged models

For tests involving models with **managed=False**, it's up to you to ensure the correct tables are created as part of the test setup.

See https://docs.djangoproject.com/en/2.1/ref/models/options/#managed

# Tests: routes, template

```
class AboutViewTest(TestCase):

    def test_view_route(self):
        response = self.client.get('/heritagesites/about/')
        self.assertEqual(response.status_code, 200)

    def test_view_route_fail(self):
        response = self.client.get('/about/')
        self.assertEqual(response.status_code, 404)

    def test_view_route_name(self):
        response = self.client.get(reverse('about'))
        self.assertEqual(response.status_code, 200)

    def test_view_template(self):
        response = self.client.get(reverse('about'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'heritagesites/about.html')
```

# Tests: Model

```
class SiteModelTest(TestCase):

    def setUp(self):
        HeritageSiteCategory.objects.create(category_name='Cultural')
        category = HeritageSiteCategory.objects.get(pk=1)
        HeritageSite.objects.create(
            site_name='Cultural Landscape and Archaeological Remains …',
            heritage_site_category_id=category.category_id,
            description='The cultural landscape and archaeological remains ...',
            justification='The Buddha statues and the cave art in Bamiyan Valley are ...',
            date_inscribed='2003',
            longitude='67.82525000',
            latitude='34.84694000',
            area_hectares='158.9265',
            transboundary=0)

def test_site_name(self):
        site = HeritageSite.objects.get(pk=1)
        expected_object_name = f'{site.site_name}'
        self.assertEqual(expected_object_name, 'Cultural Landscape and Archaeological Remains …')
```

# Tests: command line

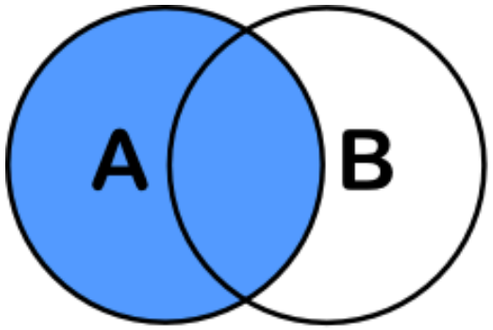
```
(venv) $ python3 manage.py test -n

Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....................
----------------------------------------------------------------------
Ran 18 tests in 0.100s

OK
Destroying test database for alias 'default'...
```

# finis

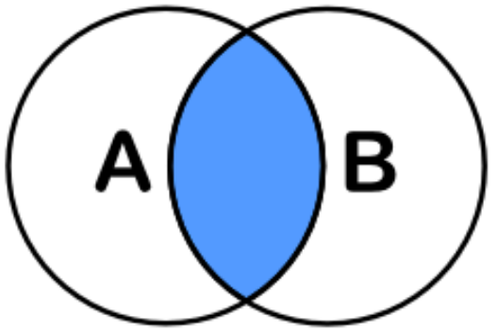
# directors cut

# SQL JOINS

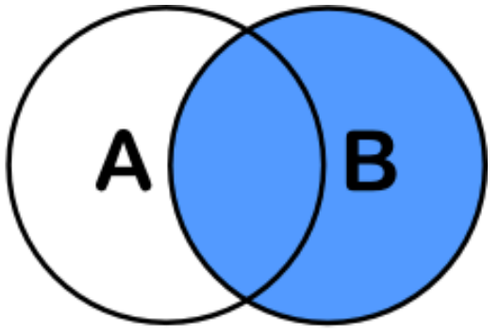
CHEATSHEET
SQL
JOINS



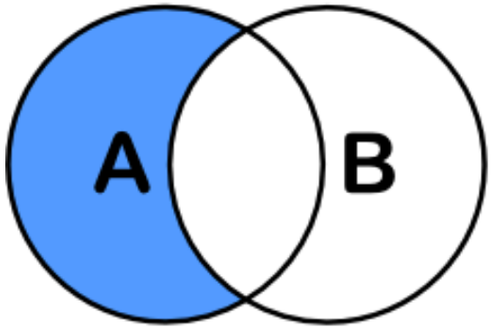

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```



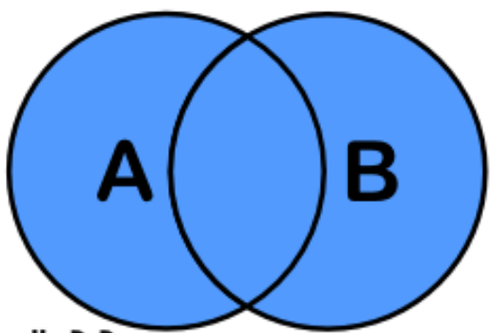

```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```



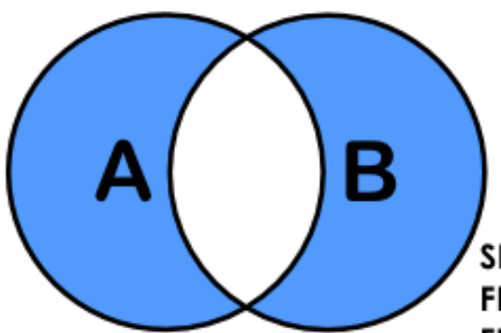

```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```



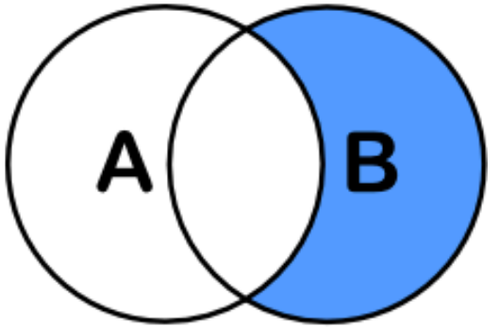

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
```



UMSI

# SQL: COUNT(*)

```
SELECT COUNT(*) AS `site count` FROM heritage_site;

+------------+
| site count |
+------------+
|       1092 |
+------------+
1 row in set (0.00 sec)
```

# SQL: GROUP_BY, COUNT(*)

## Return count of heritage sites by region

```
SELECT r.region_name AS `region`, COUNT(*) AS `heritage sites`
  FROM heritage_site hs
          LEFT JOIN heritage_site_jurisdiction hsj
                ON hs.heritage_site_id = hsj.heritage_site_id
          LEFT JOIN country_area ca
                ON hsj.country_area_id = ca.country_area_id
          LEFT JOIN location l
                ON ca.location_id = l.location_id
          LEFT JOIN region r
                ON l.region_id = r.region_id
GROUP BY r.region_name
ORDER BY `heritage sites` DESC;
```

```
+---------+----------------+
| region  | heritage sites |
+---------+----------------+
| Europe  |            488 |
| Asia    |            314 |
| Americas|            191 |
| Africa  |            144 |
| Oceania |             30 |
| NULL    |              1 |
+---------+----------------+
6 rows in set (0.02 sec)
```

# ORM: .count(), Count

```
>>> from heritagesites.models import HeritageSite
>>> hs = HeritageSite.objects.all().count()
>>> print(hs)
1092
```

```
>>> from heritagesites.models import HeritageSite
>>> from django.db.models import Count
>>> hs = HeritageSite.objects.annotate(site_count=Count('site_name'))
>>> hs.count()
1092
```

# SQL: MAX() with a subquery

## Return the largest heritage site by area (hectares)

```
SELECT r.region_name AS `region`, sr.sub_region_name AS `subregion`,
        ca.country_area_name AS `country / area`,
        hs.site_name AS `heritage site`, hs.area_hectares AS `area (hectares)`
 FROM heritage_site hs
        LEFT JOIN heritage_site_jurisdiction hsj
                ON hs.heritage_site_id = hsj.heritage_site_id
        LEFT JOIN country_area ca
                ON hsj.country_area_id = ca.country_area_id
        LEFT JOIN location l
                ON ca.location_id = l.location_id
        LEFT JOIN region r
                ON l.region_id = r.region_id
        LEFT JOIN sub_region sr
                ON l.sub_region_id = sr.sub_region_id
WHERE hs.area_hectares = (SELECT MAX(hs1.area_hectares)
                          FROM heritage_site hs1)\G
```

# SQL: MAX() with a subquery

## Return the largest heritage site by area (hectares)

```
SELECT r.region_name AS `region`, sr.sub_region_name AS `subregion`,
        ca.country_area_name AS `country / area`,
        hs.site_name AS `heritage site`, hs.area_hectares AS `area (hectares)`
 FROM heritage_site hs
        LEFT JOIN heritage_site_jurisdiction hsj
                ON hs.heritage_site_id = hsj.heritage_site_id
        LEFT JOIN country_area ca
                ON hsj.country_area_id = ca.country_area_id
        LEFT JOIN location l
                ON ca.location_id = l.location_id
        LEFT JOIN region r
                ON l.region_id = r.region_id
        LEFT JOIN sub_region sr
                ON l.sub_region_id = sr.sub_region_id
WHERE hs.area_hectares = (SELECT MAX(hs1.area_hectares)
                          FROM heritage_site hs1)\G
```

# ORM: aggregate

## SQL equivalent: SUM(), AVG(), MIN(), MAX() without a GROUP BY

```
>>> from heritagesites.models import HeritageSite
>>> from django.db.models import Max
>>> hs = HeritageSite.objects.all().aggregate(max_hectares=Max('area_hectares'))
>>> print(hs)
{'max_hectares': 40825000.0}
```

```
SELECT MAX(area_hectares) FROM heritage_site;
+------------------+
| MAX(area_hectares) |
+------------------+
|          40825000 |
+------------------+
1 row in set (0.00 sec)
```

# ORM: annotation

## SQL equivalent: GROUP BY on the column id

```
>>> from heritagesites.models import HeritageSite
>>> from django.db.models import Count
>>> hs = HeritageSite.objects.all()
.values('heritage_site_category_id')
.annotate(count=Count('heritage_site_category_id'))
>>> hs.count()
3
```

```
mysql> SELECT heritage_site_category_id, COUNT(heritage_site_category_id)
    -> FROM heritage_site
    -> GROUP BY heritage_site_category_id;
+--------------------------+--------------------------------+
| heritage_site_category_id | COUNT(heritage_site_category_id) |
+--------------------------+--------------------------------+
|                        1 |                            845 |
|                        2 |                            209 |
|                        3 |                             38 |
+--------------------------+--------------------------------+
3 rows in set (0.00 sec)
```