Final Research Project:

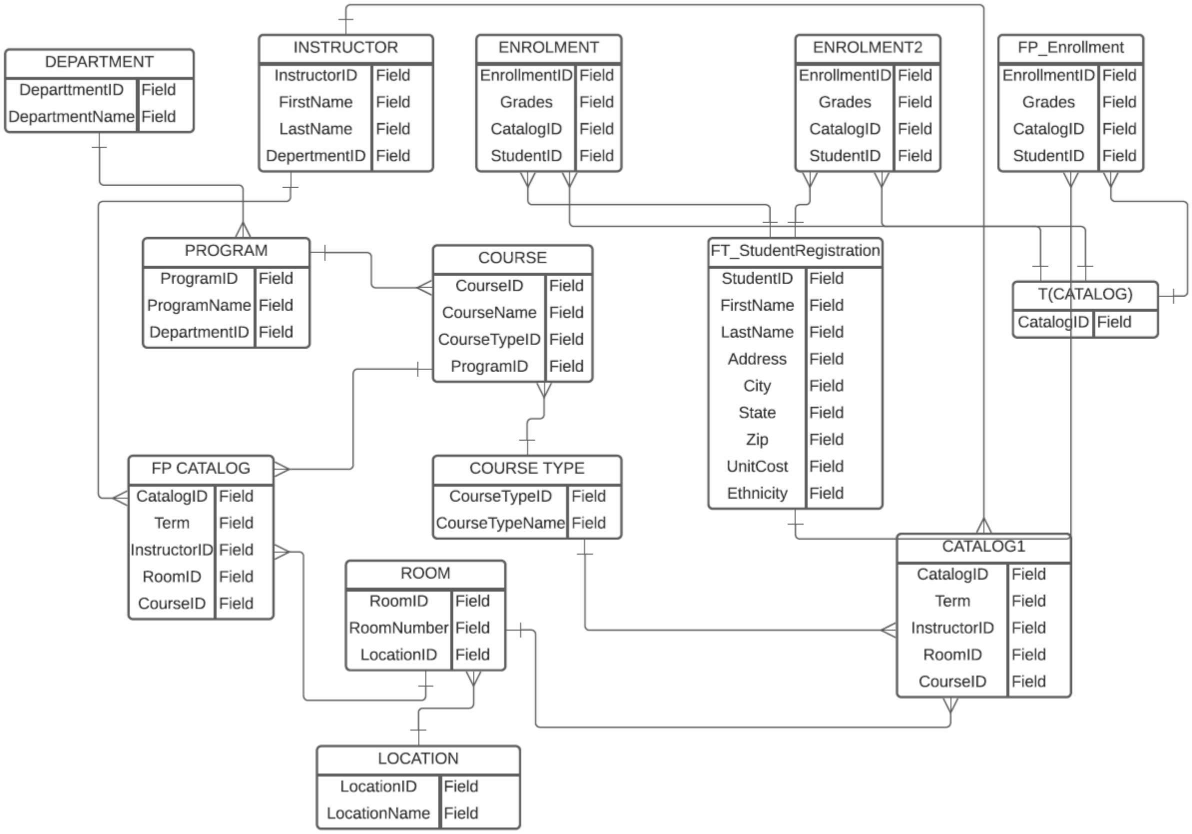
Golden Gate University

Spring 2021 ITM 304.
SF1 Managing Data Structures

Otajon Yokubov

I. RDBMS

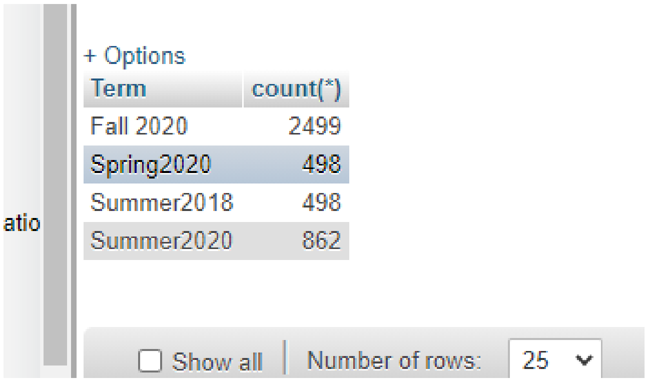
I have used FP_School_Normalized database for the project and this project intends to answer few analytical question by running corresponding queries on the database. Before going to analysis, let us first have a look at ERD of the dataset to understand the relationship between entity and attributes and how they are related.



Analytical Queries

1. How many students are admitted in each term?

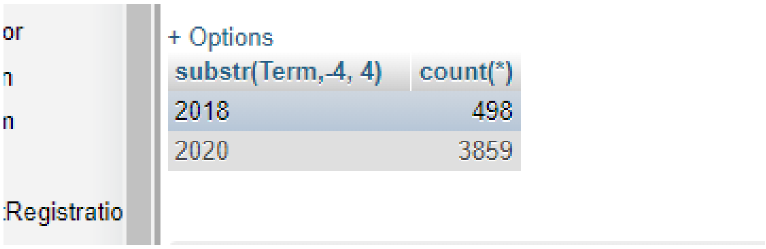
```
SELECT Term, count (*)
FROM  FP_Catalog,
           FP_Enrollment
WHERE FP_Catalog.CatalogID = FP_Enrollment.CatalogID
 GROUP BY 1
```



Term	count(*)
Fall 2020	2499
Spring2020	498
Summer2018	498
Summer2020	862

2. How many students are admitted yearly?

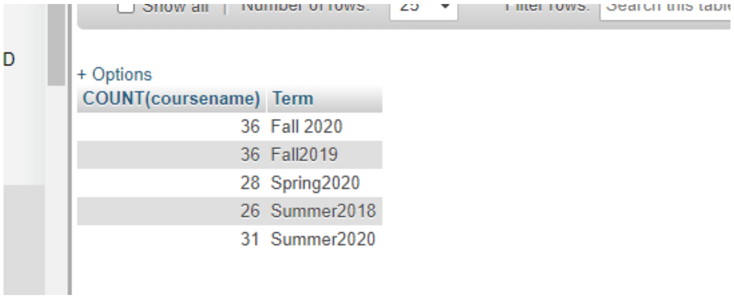
```
SELECTsubstr(Term,-4, 4), count(*)
FROM  FP_Catalog,
           FP_Enrollment
WHERE FP_Catalog.CatalogID = FP_Enrollment.CatalogID
GROUP BY      1
```



substr(Term,-4, 4)	count(*)
2018	498
2020	3859

3. What are the amount of courses per term?

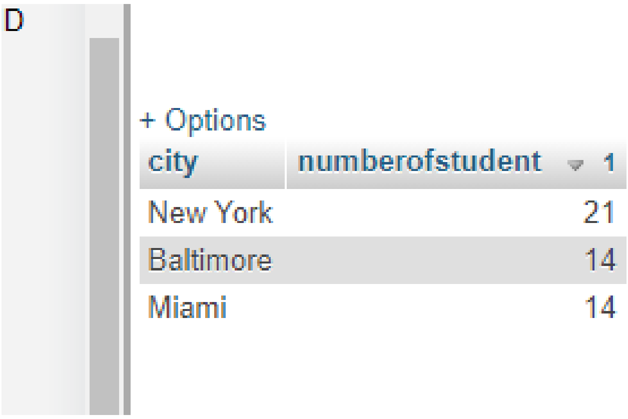
```
SELECT COUNT(coursename),
          Term
FROM  FP_Course,
          FP_Catalog
WHERE FP_Course.CourseID = FP_Catalog.CourseID

GROUP BY Term
```



COUNT(coursename)	Term
36	Fall 2020
36	Fall2019
28	Spring2020
26	Summer2018
31	Summer2020

4. What are the top 3 cities with highest number of Student

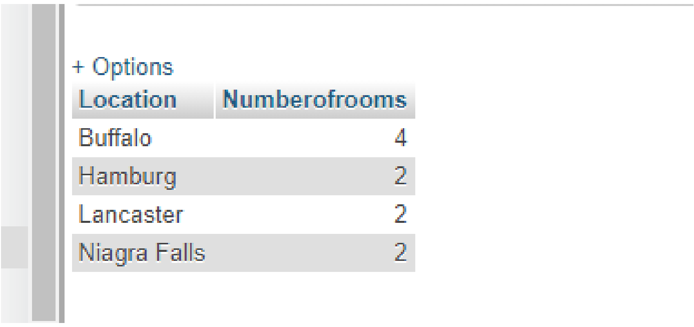
```
SELECT city, COUNT(*) as numberofstudent
FROM FP_StudentRegistration ,FP_Enrollment
WHERE FP_StudentRegistration.StudentID=FP_Enrollment.StudentID
GROUP by city
ORDER by numberofstudent DESC
LIMIT 3
```



city	numberofstudent
New York	21
Baltimore	14
Miami	14

5. How many rooms per location

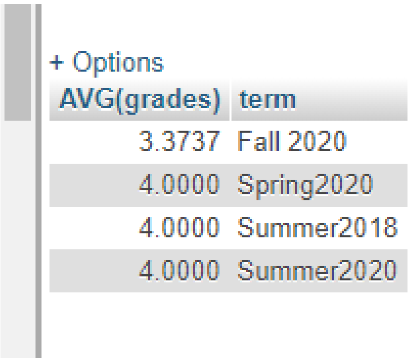
```
SELECT LocationName as Location, count(*)as Numberofrooms
from FP_Location,
FP_Room
where FP_Location.LocationID=FP_Room.LocationID
GROUP BY Location
```



Location	Numberofrooms
Buffalo	4
Hamburg	2
Lancaster	2
Niagra Falls	2

6. What is the average grade per term?

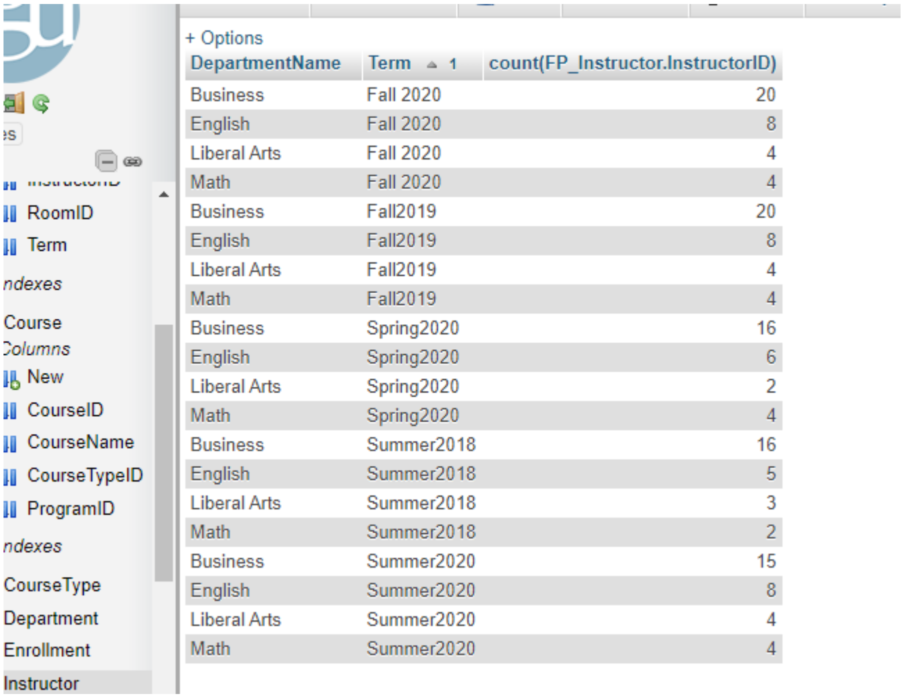
```
SELECT AVG(grades), term

from FP_Catalog,

FP_Enrollment

where FP_Catalog.CatalogID=FP_Enrollment.CatalogID

GROUP BY term
```



AVG(grades)	term
3.3737	Fall 2020
4.0000	Spring2020
4.0000	Summer2018
4.0000	Summer2020

7. How many instructors each department has?

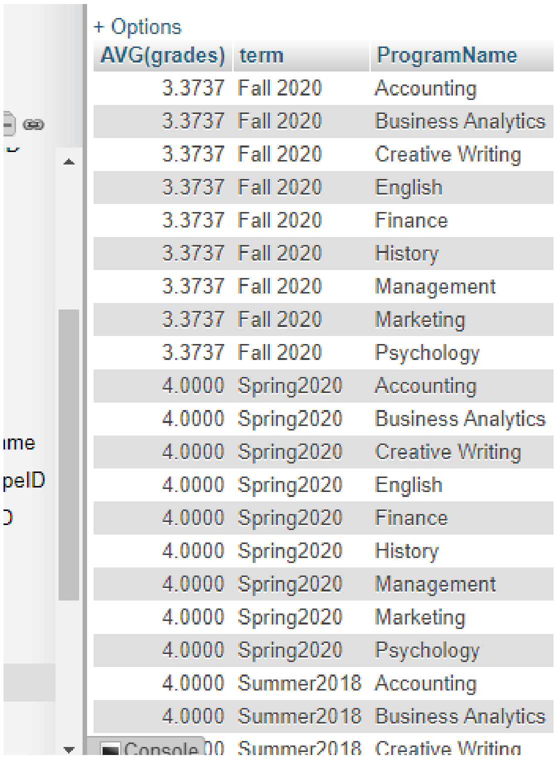
```
Select FP_Department.DepartmentName,Term, count (FP_Instructor.InstructorID)
from FP_Instructor, FP_Department,FP_Catalog
where FP_Instructor.InstructorID=FP_Catalog.InstructorID
and FP_Instructor.DepartmentID=FP_Department.DepartmentID
GROUP by FP_Department.DepartmentName,Term
ORDER by Term, FP_Department.DepartmentName
```



DepartmentName	Term ▲ 1	count(FP_Instructor.InstructorID)
Business	Fall 2020	20
English	Fall 2020	8
Liberal Arts	Fall 2020	4
Math	Fall 2020	4
Business	Fall2019	20
English	Fall2019	8
Liberal Arts	Fall2019	4
Math	Fall2019	4
Business	Spring2020	16
English	Spring2020	6
Liberal Arts	Spring2020	2
Math	Spring2020	4
Business	Summer2018	16
English	Summer2018	5
Liberal Arts	Summer2018	3
Math	Summer2018	2
Business	Summer2020	15
English	Summer2020	8
Liberal Arts	Summer2020	4
Math	Summer2020	4

8. What is the average grade by program?

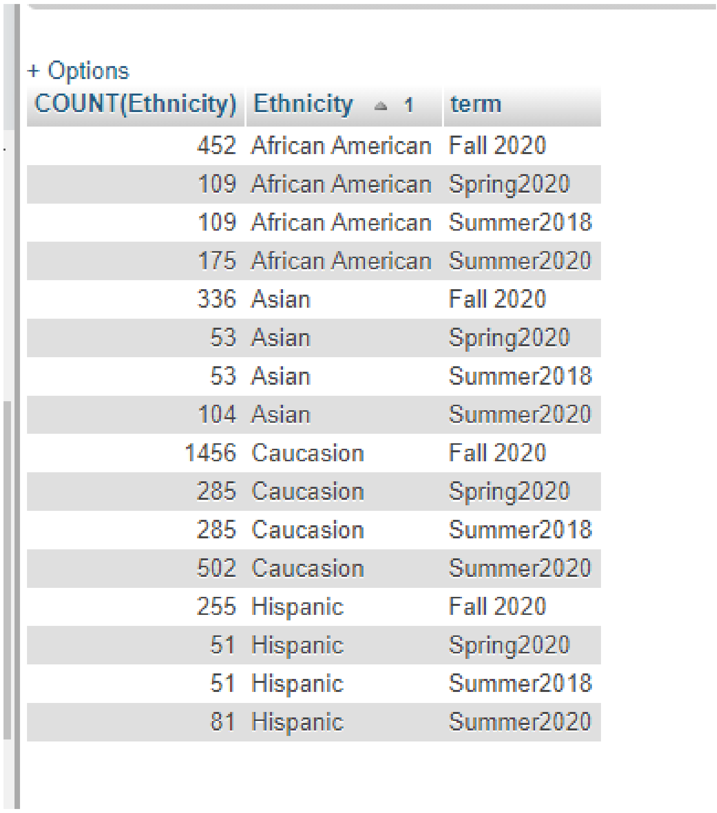
```
SELECT AVG(grades), term, ProgramName
FROM FP_Enrollment,FP_Catalog,FP_Course,FP_Program
where FP_Enrollment.CatalogID=FP_Catalog.CatalogID and
FP_Course.CourseID=FP_Program.ProgramID
GROUP By TERM, programName
```



AVG(grades)	term	ProgramName
3.3737	Fall 2020	Accounting
3.3737	Fall 2020	Business Analytics
3.3737	Fall 2020	Creative Writing
3.3737	Fall 2020	English
3.3737	Fall 2020	Finance
3.3737	Fall 2020	History
3.3737	Fall 2020	Management
3.3737	Fall 2020	Marketing
3.3737	Fall 2020	Psychology
4.0000	Spring2020	Accounting
4.0000	Spring2020	Business Analytics
4.0000	Spring2020	Creative Writing
4.0000	Spring2020	English
4.0000	Spring2020	Finance
4.0000	Spring2020	History
4.0000	Spring2020	Management
4.0000	Spring2020	Marketing
4.0000	Spring2020	Psychology
4.0000	Summer2018	Accounting
4.0000	Summer2018	Business Analytics
00	Summer2018	Creative Writing

9. What is the enthicity by term?

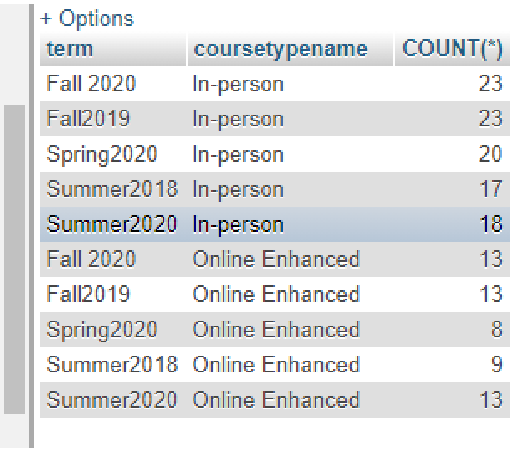
```
SELECT COUNT(Ethnicity),Ethnicity,term
FROM FP_StudentRegistration,FP_Enrollment,FP_Catalog
WHERE FP_Enrollment.StudentID=FP_StudentRegistration.StudentID and
        FP_Catalog.CatalogID=FP_Enrollment.CatalogID
        Group by Ethnicity, Term
        ORDER BY Ethnicity, Term
```



COUNT(Ethnicity)	Ethnicity ▲ 1	term
452	African American	Fall 2020
109	African American	Spring2020
109	African American	Summer2018
175	African American	Summer2020
336	Asian	Fall 2020
53	Asian	Spring2020
53	Asian	Summer2018
104	Asian	Summer2020
1456	Caucasion	Fall 2020
285	Caucasion	Spring2020
285	Caucasion	Summer2018
502	Caucasion	Summer2020
255	Hispanic	Fall 2020
51	Hispanic	Spring2020
51	Hispanic	Summer2018
81	Hispanic	Summer2020

10. Online vs in Person Classe?

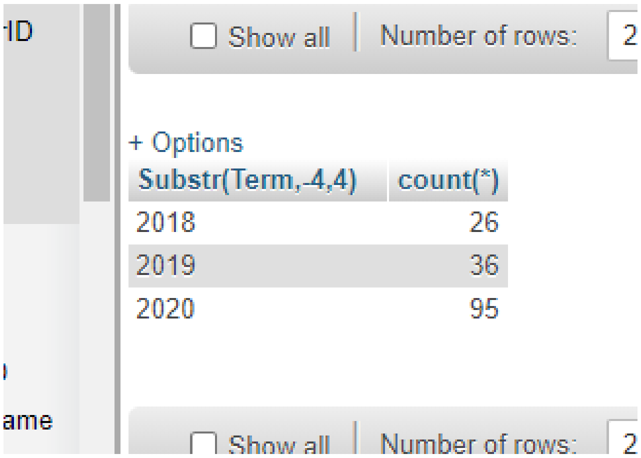
```
SELECT term, coursetypename,COUNT(*)
FROM FP_CourseType,FP_Course, FP_Catalog
where FP_CourseType.CourseTypeID=FP_Course.CourseTypeID
and FP_Course.CourseID=FP_Catalog.CourseID
GROUP by 2,1
```



term	coursetypename	COUNT(*)
Fall 2020	In-person	23
Fall2019	In-person	23
Spring2020	In-person	20
Summer2018	In-person	17
Summer2020	In-person	18
Fall 2020	Online Enhanced	13
Fall2019	Online Enhanced	13
Spring2020	Online Enhanced	8
Summer2018	Online Enhanced	9
Summer2020	Online Enhanced	13

11. Number of Instructers Per Year?

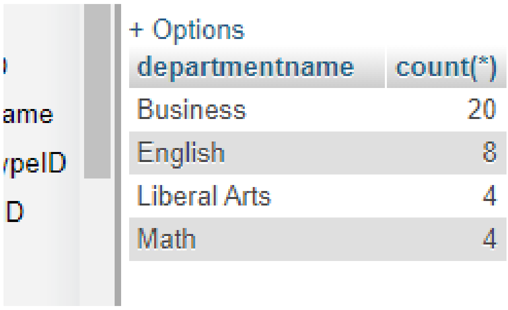
```
SELECT Substr(Term,-4,4), count(*)
from FP_Catalog,FP_Instructor
where FP_Catalog.InstructorID=FP_Instructor.InstructorID
group by 1
```



Substr(Term,-4,4)	count(*)
2018	26
2019	36
2020	95

12. Number of Instructers per departnment

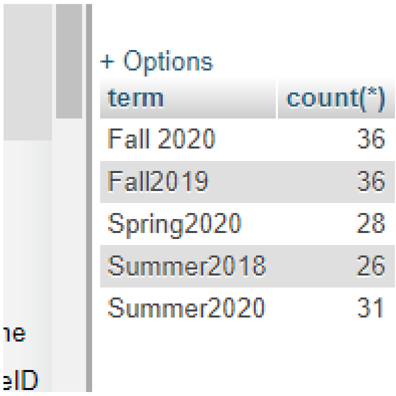
```
SELECT departmentname, count(*)
from FP_Department,FP_Instructor
where FP_Department.DepartmentID=FP_Instructor.DepartmentID
group by 1
```



departmentname	count(*)
Business	20
English	8
Liberal Arts	4
Math	4

13. Number of Instructers per term?

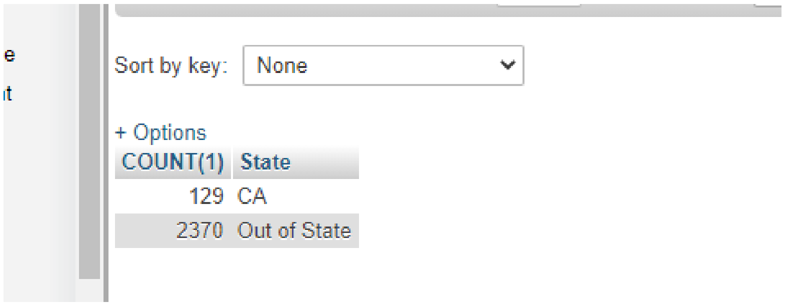
```
SELECT term, count(*)
from FP_Catalog,FP_Instructor
where FP_Catalog.InstructorID=FP_Instructor.InstructorID
group by 1
```



term	count(*)
Fall 2020	36
Fall2019	36
Spring2020	28
Summer2018	26
Summer2020	31

14. What is the number of Local and Not Local Student?

```
SELECT COUNT(1), State
FROM FP_StudentRegistration where state like 'CA' GROUP by 2
UNION SELECT COUNT(1), 'Out of State' FROM
FP_StudentRegistration WHERE state not like 'CA'GROUP by 2
```

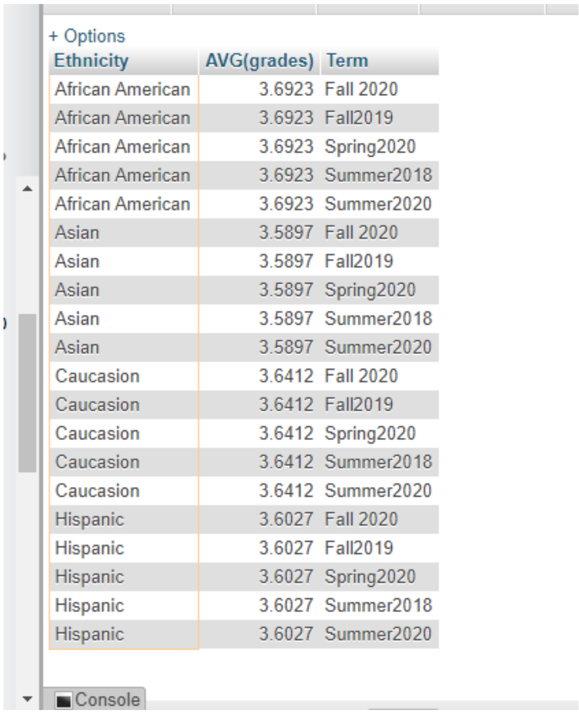


Sort by key: None

COUNT(1)	State
129	CA
2370	Out of State

15.Average grade by Ethnicity and term

```
SELECT Ethnicity,AVG(grades),FP_Catalog.Term
FROM FP_StudentRegistration,
        FP_Enrollment,
        FP_Catalog
WHERE FP_StudentRegistration.StudentID=FP_Enrollment.StudentID and
FP_StudentRegistration.Ethnicity=FP_StudentRegistration.Ethnicity and
FP_Enrollment.StudentID=FP_StudentRegistration.StudentID
GROUP By Ethnicity,term
```



Ethnicity	AVG(grades)	Term
African American	3.6923	Fall 2020
African American	3.6923	Fall2019
African American	3.6923	Spring2020
African American	3.6923	Summer2018
African American	3.6923	Summer2020
Asian	3.5897	Fall 2020
Asian	3.5897	Fall2019
Asian	3.5897	Spring2020
Asian	3.5897	Summer2018
Asian	3.5897	Summer2020
Caucasion	3.6412	Fall 2020
Caucasion	3.6412	Fall2019
Caucasion	3.6412	Spring2020
Caucasion	3.6412	Summer2018
Caucasion	3.6412	Summer2020
Hispanic	3.6027	Fall 2020
Hispanic	3.6027	Fall2019
Hispanic	3.6027	Spring2020
Hispanic	3.6027	Summer2018
Hispanic	3.6027	Summer2020

II. NoSQL:

NoSQL Database is a non-relational Data Management System that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps.Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive.

The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out."

- NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale
- In the year 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- NoSQL databases never follow the relational model it is either schema-free or has relaxed schemas
- Four types of NoSQL Database are 1).Key-value Pair Based 2).Column-oriented Graph 3). Graphs based 4).Document-oriented
- NOSQL can handle structured, semi-structured, and unstructured data with equal effect
- CAP theorem consists of three words Consistency, Availability, and Partition Tolerance
- BASE stands for **B**asically **A**vailable, **S**oft state, **E**ventual consistency
- The term "eventual consistency" means to have copies of data on multiple machines to get high availability and scalability
- NOSQL offer limited query capabilities

As NoSQL databases do not adhere to a strict schema, they can handle large volumes of structured, semi-structured, and unstructured data. This allows developers to be more agile and push code changes much more quickly than with relational databases. In this report, we compare two popular open-source NoSQL databases

Cassandra is widely favored for its enterprise features, like scalability and high availability that allow it to handle large amounts of data and provide near real-time analysis. Written in Java, Cassandra offers synchronous and asynchronous replication for each update. Its durability and fault-tolerant capabilities make it ideal for always-on applications.

Unlike MongoDB, Cassandra uses a masterless “ring” architecture which provides several benefits over legacy architectures like master-slave architecture. This, in turn, means that all nodes in a cluster are treated equally, and a majority of nodes can be used to achieve quorum.

Although Cassandra stores data in columns and rows like a traditional RDBMS, it provides agility in the sense that it allows rows to have different columns, and even allows a change in the format of the columns. Apart from this, its query language, Cassandra Query Language (CQL), closely resembles the traditional SQL syntax, and thus, can be easier for SQL users to understand. This gives it some leverage in any comparison of Cassandra vs. HBase.

Cassandra offers advanced repair processes for read, write, and entropy (data consistency), which makes its cluster highly available and reliable. Owing to its lack of a single point of failure, it can provide a highly available architecture if a quorum of nodes is maintained and the replication factor is tuned accordingly. This also allows for better fault tolerance compared to document stores like MongoDB, which might take up to 40 seconds to recover.

Some of Cassandra’s most common use cases include messaging systems (for its superior read and write performance), real-time sensor data, and e-commerce websites.

MongoDB offers both a community and an enterprise version of the software. The enterprise version offers additional enterprise features like LDAP, Kerberos, auditing, and on-disk encryption.

MongoDB is a schema-less database and stores data as JSON-like documents (binary JSON). This provides flexibility and agility in the type of records that can be stored, and even the fields can vary from one document to the other. Also, the embedded documents provide support for faster queries through indexes and vastly reduce the I/O overload generally associated with database systems. Along with this is support for schema on write and dynamic schema for easy evolving data structures.

MongoDB also provides several enterprise features, like high availability and horizontal scalability. High availability is achieved through replica sets which boast features like data redundancy and automatic failover. This ensures that your application keeps serving, even if a node in the cluster goes down.

MongoDB also provides support for several storage engines, ensuring that you can fine-tune your database based on the workload it is serving. Some of the most common use cases of MongoDB include a real-time view of your data, mobile applications, IoT applications, and content management systems. Finally, it includes a nested object structure, indexable array attributes, and incremental operations

Reference : guru99.com/nosql-tutorial.html