Mardel Maduro

Student # 3349958

COMP 418

Assignment 1

### **Question 1**

1. **Briefly explain the three main alternatives for storing information in a data entry of an index.**

- The three main alternatives to for storing information in a data entry of an index are as follows;

1) An entry k\* where k\* is the actual data record and its search key value k

2) An entry (k, rid ) where rid is the record id of data record with search key value k>

3) An entry (k , rid-list) where rid-list is a list of record ids of data records with search key k>

1. **Define clustered index, and discuss the relation between the three alternatives and clustered/unclustered indexes.**

- Clustered Index : A clustered index exists when a file is organized so that the ordering of data records is the same as or close to the ordering of data entries in some index.

Alternative 1 by definition would be considered a clustered index, whereas alternative 2 and 3 would be considered unclustered, but if and only if they are sorted on the search key field.

### **Question 2**

**Consider the following file organizations: *sorted files, heap files with an unclustered tree index on the search key*, and *heap files with an unclustered hash index*. Briefly discuss the suitability of each of these file organizations to perform the following operations: file scans, range selections, inserts, and deletes.**

**Sorted Files -**

File scans – Costs B(D+RC) since all pages must be examined and records are retrieved in the order they were sorted in.

Range selections – Assuming the that the range of selection is matches the composite key , the first selection that satisfies the selection is located as for search for equality. Subsequently data pages are sequentially retrieved until a record is found that doesn’t satisfy the range selection (similar to an equality search with multiple qualifying records).

Inserts – Must first find correct position of the entry to be enter (as to keep the sorted order that is already in effect), secondly add the entry and lastly fetch and rewrite all the entries that are to follow (assuming that there is no free slots in the file and shits must occur). Cost B(D+RC) + (cost of search) .

Deletes – Must first find the position of the entry, secondly the record is removed from the page and lastly read and rewrite the records that followed the deleted record so that they can be moved up to compact the space.

**Heap files with an unclustered tree index on the search key -**

File scans – To do a full scan of the file, we need to scan the leaf level of the index and for each data entry, fetch the corresponding data record from the underlying file, obtaining data records in the sort order.

Range selections – Assuming that the range selection matches the composite key, the first record that satisfies the selection is found as it is with search for equality. Subsequent data entires are retrieved using the next and previous links at the leaf level of the index, and this is done until a data entry that doesn’t satisfy the selection is found.

Inserts – We must first insert the record into the heap file at a cost of 2D+C, we must also insert the corresponding data entry in the index.

Deletes – We need to locate the data record in the file and the data entry in the index. We then need to write out the modified pages in the index and the data file.

**Heap files with an unclustered hash index -**

File scans – All entries can be retrieved in-expensively at a cost of O.125B(D+8RC) I/Os . However for each entry we add the cost of one I/O for fetching the corresponding data record.

Range selections – The hash structure offers no help and the entire files needs to be scanned at a cost of B(D+RC)

Inserts – Must first insert record into the file at a cost of 2D+C. Additionally the appropriate page in the index must be found, modified for insertion and rewritten. Additional cost is H+2D+C.

Deletes – Data record in file and data entry in the index both need to be located. This is done at a cost of H+2D+4RC. We also need to write out the modified pages in the index and the data file, this is done at an additional cost of 2D.

### **Question 3**

1. **Briefly describe the two internal organizations for heap files (using lists versus directory of pages).**

- The two internal organizations for heap files are using a list of pages and a directory of pages.

List of Pages - In this implementation a doubly linked list is used to maintain the heap file. The DBMS remembers the location of the first page by maintaining a table of (heap\_file\_name, page\_Laddr) pairs in a known location. This first page is referred to as the header page and all others are data pages. The header page has 2 pointers, one that points to the list of pages with free space and another pointer to a list of full pages. If a new page is required , it is obtained by making a request to the disk space manager and a new page is added to the list of pages with space. If a page is to be deleted, it is removed from whichever list it is currently in and the disk space manager is told to deallocate it. The disadvantage of using this scheme however is that all pages in a heap will be on the “pages with free space” list since records are of varying length and pages will end up having a few free bytes. As such when it comes to inserting records we would have to look at multiple pages on the free list to ensure we can find one with enough free space.

Directory of Pages – In this implementation a directory is used to maintain the heap file. The DBMS remembers where the first directory page of each heap file is located and then the directory itself is just a collection (whether a linked list or some other data structure). Each directory entry identifies a page or sequence of pages and the amount or entries and pages are grown or shrunk depending on the needed space. Free space can be managed using a “bit per entry” to see whether a page has free space or not , or a “count per entry” to see how much space the page has left. If the file contains variable length records, we can look at the free space count and see if the records fits on the page pointed to be the entry. We can then efficiently search for a data entry since multiple entries can fit on a single page.

1. **Explain which organization you would choose if records are variable in length.**

-If records where variable in length I would chose the directory of pages organization. This is due to its increased speed in search afforded by the fact that the directory organization make better use of the space and data structures used for saving entries than the version using the list does.

### **Question 4**

**Compare ISAM and B+ Tree indexes. Explain briefly their differences in handling Search, Insert and Delete, and discuss when you would use ISAM and when you would use B+ Tree index.**

ISAM : completely static (except for overflow pages), pages allocated sequentially on file creation

Search – Starts at the root node, determines which subtree to search by comparing the value in the search field of the record to the ones in the key values in the nodes. (Identical algorithm to that of B+ Tree)

Insert – Uses the search to find the appropriate page for the record to be inserted and either inserts it into the page or create an overflow page and insert it there if needed.

Delete – Uses the search to find the record’s page and deletes it from the page, if page was an overflow page deallocate it.

Preferred Use Case – if storing a fairly static dataset

B+ Tree Indexes : dynamic , uses a balanced tree

Search – Start at root and goes to appropriate leaf. (no checking of each subtree, so faster than ISAM even with identical algorithm)

Insert – Search to find node for placing data entry into. If not full put data in node. If full split node, which adds new nodes which are children of the node that was split, and put data in one of the slots in one of the new child nodes.

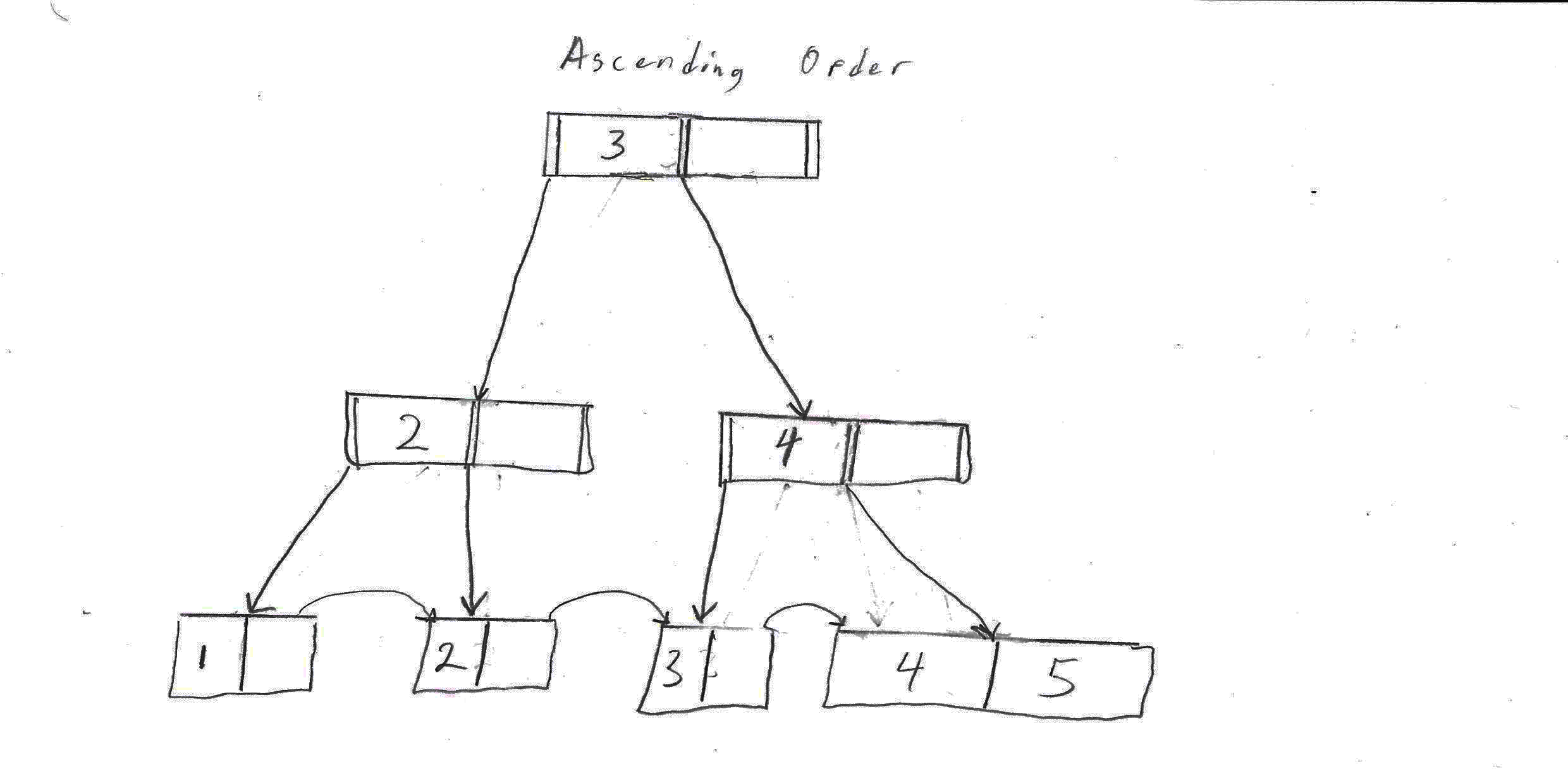
Delete – Search to find data entry and delete it (typically no resizing or restructuring needed, but if node goes below occupancy threshold redistribution takes place).

Preferred Use Case – when storing data that is constantly being changed or updated.

### **Question 5**

Does the final structure of a B+ tree depend on the order in which the terms are added to it? Explain your answer using an illustration example. - YES

Example B+ Trees of Values 1 to 5.



### 

### **Question 6**

**Explain how extendible hashing uses a directory of buckets, and discuss the global depth of the index and local depth of a bucket.**

- Extendible hashing uses a directory of buckets by having pointers to these buckets and doubling the size of the number of buckets by doubling just the directory splitting only the bucket is is overflowing. Meaning at the start there will be an array of buckets (directory) and each bucket has a particular size (can only fit so many data entries). As a bucket becomes filled, it is split in 2 with half of the entries staying in the old bucket and the other have being put in the new bucket. When this split is done we double the size of the directory and update the indexing scheme for picking a bucket (if it was 4 buckets and we were using 2 bits to pick, once a split occurs we would be using 3 bits and now 5 buckets with the split buckets checking the most significant bit to see which of the buckets from the split we’re going to use, and the not split buckets ignoring the most significant bit until that bucket is full and needs to be split, in which case we do the same as with the original split but since there is space for more buckets in the directory we just add the bucket and split the entries instead of doubling the directory size.

## Part 2. Design considerations for application scenarios (34 marks)

### **Question 1**

Consider the following relations:

**Professor (profid: integer, name: varchar, salary: integer, age: integer, depid: integer)  
Department (did: integer, budget: integer, location: varchar, mgr eid: integer)**

**Salaries range from $30,000 to $100,000, ages vary from 20 to 80, each department has about 20 employees on average, there are 10 locations, and budgets vary from $100,000 to $1 million. You can assume uniform distributions of values.**

*For each of the following queries, what index would you choose to speed up the query? If your database system does not consider index-only plans (i.e., data records are always retrieved even if enough information is available in the index entry), how would your answer change? Explain briefly.*

1. Query1: Print name, age, and salary for all professors.

**-** Providing the system supports index-only plans I would use an unclustered hash index on the (name, age, salary) fields in the professor table. Providing that index-only plans are not supported I would still use the unclustered hash index since it would still have the same performance since whether you use an index or not the entire file has to be read, which is does rather efficiently.

1. Query2: Find the dids of departments that are located in Edmonton and have a budget of more than $150,000.

**-** Providing the system supports index-only plans I would use a clustered tree-based index (preferably B+ since I would like the added benefits of the system being dynamic) to index the (location,budget) fields in the Department table. By it being clustered it would already have entries sorted by location first then budget second, meaning the system already knows where to go to find the first record with location = Edmonton and the first record with location = Edmonton & budget >= $150,000.

Provided the systems doesn’t support index-only plans I would still use the clustered B+ tree since it would still be in order and the data would be stored in the index, meaning the cost should be the same as if it was with index-only plans supported.

### **Question 2**

**The CVT Company is a leader in the manufacture of work clothes. You are hired as database administrator for the company and your IT supervisor asked you to solve a retrieval speed problem they used to have with a large file for item records. Your supervisor mentioned that they have sorted the file but the problem didn’t improve, so they need to create a B+ tree index to solve the problem. Your supervisor outlined the way to do it: “The best way to accomplish this task is to scan the file, record by record, inserting each one using the B+ tree insertion procedure.” Being a fresh graduate, you noticed that since the file is already sorted there is a better way to do it.**

1. What performance and storage utilization problems are there with your supervisor’s approach?

-The supervisor’s approach is very expensive since it requires us to start at the root and go down to the corresponding leaf for each insertion. This means we end up spending a lot of time doing it this way. It also prevents us from having sequentially stored leaves, which means slower data retrieval when making queries.

1. Explain how the bulk-loading algorithm provides a better alternative than the proposed scheme.

- The bulk-loading algorithm provides a better alternative by allowing us to have to perform less I/Os during the build. It also allows us to have sequentially stored (linked) leaves, which provide us with faster data retrieval. Since the file is already sorted the first part of the process is already done for us.

### **Question 3**

Your team in charge of database administration was discussing different alternatives for indexing your organization’s databases. Some tables in one database have very few insertions but they are used intensively by different services to check for information about items using the item\_ID number. While many of your colleagues proposed using a tree index, you argued for a Hash index for these tables because it provides an average-case search cost of only slightly more than one disk I/O. The team leader agrees to adopt your solution but has asked you to write a short explanation for two questions:

1. How does Linear Hashing provide an average-case search cost of only slightly more than one disk I/O, given that overflow buckets are part of its data structure?

**-**Linear Hashing provides an average-case search cost of only slightly more than one I/O despite having overflow buckets as part of its structure by minimizing what is in the overflow bucket by constantly redistributing data entries so that overflow buckets don’t get to the point of being more than one page long. This means that data is more evenly distributed and it is faster to get to data entries.

1. If a Linear Hashing index using Alternative (1) for data entries contains 10,000 records, with 10 records per page and an average storage utilization of 80 percent, what is the worst-case cost for an equality search? Under what conditions would this cost be the actual search cost?

-

Average # of records per page = 0.8 \* 10 = 8

# of pages = 1,250

So worst case we have to access each page, so it would take 1,250 I/Os.

This worst-case would occur if all 10,000 values hash to the same key and each of those 1250 pages are created from the constant splitting of that bucket for that key.

Part 3 – Implementation

Tables were made using PostgreSQL 9.6 and the code describing their creation are appended at the end of this report as well as a text file that accompanies this document . The three tables were made as follows:

**Student** – 89 Records with ages ranging from 21 to 48.

**Course** – 31 Records with number of credits ranging from 3 to 6, and course fees ranging from $350 to $1200.

**Registration** - 317 Records with grades ranging from 15.6 to 100;

To create the report on the performance of the queries both before and after adding the indexes, the queries were run times using the ‘\timing’ command to enable timing reports. After the index less run I exited the database and stopped PosrgreSQL from running, waited 10 seconds and restarted it. I then added the indexes and then made the queries again.

**Queries**

1) List the student numbers and names of students who received a grade greater or equal to 70% in the course “COMP418,” sorted by age ascending.

Query - SELECT registration.sid, student.name FROM registration INNER JOIN student ON registration.sid=student.sid where registration.courseno = 'comp418' AND registration.grade >=70 order by age asc;

Time without indexes= 224.664 ms

Time with both indexes = 39.477 ms

2) List the course numbers and titles of courses that have more than 10 students getting a grade lower than 50. [(Use group by courseNo and count(SID)].

Query - select course.courseno, course.title from course inner join registration on

course.courseno=registration.courseno and registration.grade < 50 group by course.courseno,

course.title having count(\*)>10;

Time without indexes = 94.500 ms

Time with both indexes = 68.517 ms

3) List the course numbers and titles of courses whose course fees are between 400 and 600 dollars.

Query - select courseno, title from course where coursefees between 400 and 600;

Time without indexes = 76.157 ms

Time with both indexes = 31.295 ms

4) List all courses in the database.

Query - select \* from course;

Time without indexes = 0.441 ms

Time with both indexes = 0.537 ms

5) Update all the course fees by adding 6 dollars to each course.

Query - update course set coursefees = coursefees+6;

Time without index = 149.073 ms

Time with both indexes = 43.227 ms

Code used to add indexes-

CREATE INDEX sid\_index ON student (sid);

CREATE INDEX courseNum\_index ON registration (courseno);

Reasoning for index choice

1) sid for student since we use it in multiple queries.

2)courseno for registration again because we use it in multiple queries.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Without Indexes | With Indexes | Performance Increase |
| Query 1 | 224.664 ms | 39.477 ms | 5.69x |
| Query 2 | 94.500 ms | 68.517 ms | 1.38x |
| Query 3 | 76.157 ms | 31.295 ms | 2.43x |
| Query 4 | 0.441 ms | 0.537 ms | 0.82x |
| Query 5 | 149.073 ms | 43.227 ms | 3.45x |

Conclusion : By implementing the indexes we were able to get an improvement of about 2.66 times faster on average. Before indexing the query that took the longest time to execute was query 1. It is also the query with the largest amount of records returned as well as the query with the largest performance gain. By looking at this, it seems that the larger the amount of data that is to be looked at ( as long was the query is complex and not just a select \* from X) the bigger the boost in performance we can get by implementing indexes.

Code For Tables

CREATE TABLE Student(

SID serial NOT NULL,

Name VARCHAR (50) NOT NULL,

Address VARCHAR (50) NOT NULL,

Telephone VARCHAR (11) NOT NULL,

Age integer NOT NULL

);

INSERT INTO student (sid, name, address, telephone, age) VALUES

(DEFAULT, 'Linus Torvalds','54 Unix Drive','12345678901',48),

(DEFAULT, 'Bob Roberts','27 Lorna Road','16235469875',21),

(DEFAULT, 'Marcus Hader','97 Goose Street','12544587895',25),

(DEFAULT, 'Sam Fisher','32 Splinter Ave','11244588896',32),

(DEFAULT, 'Tyra Ellain','66 Ha Been Ta','12256589221',25),

(DEFAULT, 'Luke Briggs','90 Gauge Drive','12833456733',28),

(DEFAULT, 'Samantha Keen','37 River Road','19049937745',24),

(DEFAULT, 'Oliver Twigs','44 Bolder Street','17789423467',31),

(DEFAULT, 'Sal Masala','Juniper Street','19900844475',22),

(DEFAULT, 'Bryan Kage','55 Gauge Drive','12833458859',23),

(DEFAULT, 'Kareem Hamm','532 River Road','19047854489',33),

(DEFAULT, 'Omar Scuse','89 Jets Drive','19948589322',45),

(DEFAULT, 'Sam Masala','Juniper Street','19900844473',46),

(DEFAULT, 'Leah Benjamin','106 Keppler Ave','16748932287',23),

(DEFAULT, 'Richard Puller','748 Pebble Drive','19090988743',23),

(DEFAULT, 'Silvia Jerez','56 Braxton Street','18244755677',28),

(DEFAULT, 'James Phillips','78 Bolder Street','17789422234',37),

(DEFAULT, 'Mary Howard','845 Paradise Lane','19998097845',41),

(DEFAULT, 'Jesseie Purcell','67 Sussex Court','18949908956',29),

(DEFAULT, 'Joshua Thompson','598 River Road','19047859900',27),

(DEFAULT, 'Stacey Gonzales','180 Jets Drive','19948587777',32),

(DEFAULT, 'Helen Kim','111 Juniper Street','19900849087',27),

(DEFAULT, 'Lamar Benjamin','106 Keppler Ave','16748932287',25),

(DEFAULT, 'Evan Richards','74 Pebble Drive','19090988885',21),

(DEFAULT, 'Silvia Haris','546 Braxton Street','18244475885',24),

(DEFAULT, 'Juno Parker','324 Green Lane','18998044566',32),

(DEFAULT, 'Mary Jenkens','185 Paradise Drive','16455637784',26),

(DEFAULT, 'Jamie Faulkner','673 Polka Park','13347843375',35),

(DEFAULT, 'Joshua Tempe','358 River Road','19047588856',23),

(DEFAULT, 'Stacey Banks','330 River Bend','12237846734',28),

(DEFAULT, 'Kim Pope','784 Lee Lane','16478394495',29),

(DEFAULT, 'Richard Faux','379 Grasslane','16879930923',36),

(DEFAULT, 'Simmar Thaller','546 Braxton Street','18244757784',26),

(DEFAULT, 'Phillip James','64 Bolder Street','17789428859',35),

(DEFAULT, 'Mary Lou','445 Paradise Lane','19998049957',35),

(DEFAULT, 'Jamie Purcell','104 Sussex Court','18949894564',26),

(DEFAULT, 'Thomas Hann','190 River Bend','190489453',23),

(DEFAULT, 'Stacey Bobb','48 Grasslane','16879938945',24),

(DEFAULT, 'Cooper Krane','244 Juniper Street','19900867855',24),

(DEFAULT, 'Richard Dyke','773 Pebble Drive','19090980000',28),

(DEFAULT, 'Silvia Jimanez','56 Braxton Street','18244759999',29),

(DEFAULT, 'James Howard','189 Bolder Street','17789422990',35),

(DEFAULT, 'Molly Pipper','325 Paradise Lane','19998067792',42),

(DEFAULT, 'Jay Mercer', '624 Sussex Court', '18949904456',43),

(DEFAULT, 'Josh Lowes', '59 Hargrave', '17764789932',26),

(DEFAULT, 'Benjamin Uke', '99 King Street', '18944499903', 24),

(DEFAULT, 'Jenn Heller', '364 Green Park', '18895587445', 27),

(DEFAULT, 'Dick Richards', '834 Hanley', '17785943345', 29),

(DEFAULT, 'Erika Pipps', '536 Braxstate', '18234467854', 24),

(DEFAULT, 'James Erison', '235 Bolder Street', '17789429986', 33),

(DEFAULT, 'Lucas Howard', '845 Paradise Lane', '19998097845', 21),

(DEFAULT, 'Melannie Purcell', '67 Sussex Court', '18949908956', 23),

(DEFAULT, 'Evan Little', '53 River Park', '18857488998', 24),

(DEFAULT, 'Lani Patrice', '456 Jets Drive', '19948588857', 28),

(DEFAULT, 'Jess Picar', '843 Interloop', '11123346657', 24),

(DEFAULT, 'Andre Benjamin','106 Keppler Ave', '16748932287', 28),

(DEFAULT, 'Natasha Paulk', '1019 Camden Place', '12763546745', 27),

(DEFAULT, 'Cassandra Bisbee', '1935 George Street', '12839807855', 32),

(DEFAULT, 'Lisa Kuhn', 'Owagner Blvd', '12627798042', 39),

(DEFAULT, 'Jennifer Frazier', '185 Mason Hill Road', '12278967730', 36),

(DEFAULT, 'Hal Honeycutt', '673 Rockwell Lane', '12227844576', 45),

(DEFAULT, 'Jason Ramirez', '367 Kings Circle', '14037558856', 28),

(DEFAULT, 'Deanna Gwinn', '532 River Bend', '12237847749', 32),

(DEFAULT, 'Kristina Williams', '2267 Walnut Street', '14536477748', 41),

(DEFAULT, 'Brittany Carlin', '546 Grasslane', '16879937755', 46),

(DEFAULT, 'Alicia Allen', '46 Apple Lane', '17747949075', 36),

(DEFAULT, 'Donald Mathis', '77 Eagle Street', '18959058845', 48),

(DEFAULT, 'Shawn Lee', '274 Paradise Lane', '19998043371', 24),

(DEFAULT, 'Susan Cornell', '274 Sussex Court', '18949890084', 30),

(DEFAULT, 'Angela Fred', '560 River Bend', '190480095', 34),

(DEFAULT, 'Fred Flitson', '108 Grasslane', '16879938890', 26),

(DEFAULT, 'David Jones', '564 Dog Hill Lane', '18094758943', 21),

(DEFAULT, 'Wendy Powers', '574 Viking Drive', '12344875537', 39),

(DEFAULT, 'Robert Nicholas', '234 Jefferson Street', '17364466632', 32),

(DEFAULT, 'Nick Roberts', '377 Bolder Street', '17789423845', 27),

(DEFAULT, 'Barbara Tillotson', '300 Kimberly Way', '12733456735', 26),

(DEFAULT, 'Ramon Register', '224 Sussex Court', '18949902536', 26),

(DEFAULT, 'James McGriff', '207 Carriage Lane', '12267834657', 22),

(DEFAULT, 'Alan Celis', '99 Marion Park', '12734499903', 29),

(DEFAULT, 'Edward Daniels', '492 Green Park', '18895589907', 31),

(DEFAULT, 'Kyle Lahr', '274 Hanley', '17785941234', 22),

(DEFAULT, 'Erika Jones', '274 Braxstate', '18234460098', 44),

(DEFAULT, 'Ida Gray', '80 Joy Ave', '12269837974', 41),

(DEFAULT, 'Lamia King', '42 Paradise Lane', '19998097786', 34),

(DEFAULT, 'David Moody', '267 Sussex Court', '18949909907', 28),

(DEFAULT, 'Elric Pascla', '503 River Ave', '16748371124', 34),

(DEFAULT, 'Myra Hunt', '6 Jets Drive', '19948587460', 28),

(DEFAULT, 'Edward Campbell', '647 Interloop', '11123346673', 22),

(DEFAULT, 'Andre Burgos', '347 Keppler Ave', '16748938374', 28);

CREATE TABLE Course(

CourseNo VARCHAR (7) NOT NULL,

Title VARCHAR (50) NOT NULL,

Department VARCHAR (50) NOT NULL,

NumberOfCredits integer NOT NULL,

CourseFees integer NOT NULL

);

INSERT INTO course (courseno, title, department, numberofcredits, coursefees) VALUES

('comp418', 'Distributed DB Systems and DB Tuning', 'Computer Science', 3, 700),

('comp314', 'Operating Systems', 'Computer Science', 3, 600),

('comp101', 'Intro to CS 1', 'Computer Science', 3, 400),

('comp102', 'Intro to CS 2', 'Computer Science', 3, 400),

('stat100', 'Intro to Statistics', 'Statistics', 3, 400),

('stat200', 'Statistics 2', 'Statistics', 3, 500),

('stat463', 'Stochastic Processes', 'Statistics', 3, 700),

('stat349', 'Time Series Analysis', 'Statistics', 3, 600),

('math150', 'Calculus 1', 'Mathematics', 3, 400),

('math130', 'Vector Geometry', 'Mathematics', 3, 400),

('math217', 'Number Theory', 'Mathematics', 3, 500),

('math333', 'Computational Algebra', 'Mathematics', 3, 600),

('engl093', 'English Composition', 'English', 3, 350),

('engl120', 'Representative Literary Works', 'English', 6, 800),

('engl217', 'American Literature to 1900', 'English', 3, 500),

('engl367', 'Studies in the Novel', 'English', 6, 1200),

('geog128', 'Intro to Human Geography', 'Geography', 3, 400),

('geog252', 'Geography for Natural Resources', 'Geography', 3, 500),

('geog346', 'Urban Geography', 'Geography', 6, 1200),

('geog433', 'Concepts in Atmospheric Modeling', 'Geography', 3, 700),

('entr201','Managing the Smaller Business','Entrepreneurship',3,500),

('entr310','Smaller Business Management','Entrepreneurship',3,600),

('entr410','New Venture Analysis','Entrepreneurship',3,700),

('eng143', 'Design in Engineering', 'Engineering', 3, 400),

('eng203', 'Eng Comm : Strategies for the Profession', 'Engineering', 3, 500),

('eng300', 'Engineering Economics', 'Engineering', 3, 600),

('eng402', 'Pro Engineering Practice in MB', 'Engineering', 4, 900),

('civl374', 'Hydraulics', 'Engineering', 4, 700),

('civl404', 'Structual Dynamics', 'Engineering', 4, 900),

('fren115', 'Intro to French', 'Humanities', 3, 400),

('fren282', 'African Cinema', 'Humanities', 3, 500);

CREATE TABLE Registration(

SID serial NOT NULL,

CourseNo VARCHAR (7) NOT NULL,

startDate DATE NOT NULL,

CompleteDate DATE NOT NULL,

Grade real NOT NULL

);

INSERT INTO registration (sid, courseno, startdate, completedate, grade) VALUES

(1,'comp314','2017-01-05','2017-06-05', 100),

(1,'comp418','2017-01-05','2017-06-05', 95.7),

(1,'geog128','2017-01-05','2017-06-05', 87.4),

(1,'math217','2017-01-05','2017-06-05', 77.9),

(2,'comp418','2017-01-05','2017-06-05', 75),

(2,'engl120','2017-01-05','2017-06-05', 95),

(2,'fren282','2017-05-09','2017-11-09', 68.5),

(2,'eng300','2017-05-09','2017-11-09', 45.6),

(2,'stat200','2017-05-09','2017-11-09', 65.4),

(3,'comp314','2017-01-05','2017-06-05', 72.5),

(3,'comp418','2017-01-05','2017-06-05', 95.7),

(3,'geog128','2017-01-05','2017-06-05', 87.4),

(4,'math150','2017-01-05','2017-06-05', 73.8),

(4,'comp102','2017-01-05','2017-06-05', 100),

(4,'engl217','2017-01-05','2017-06-05', 86),

(5,'fren282','2017-05-09','2017-11-09', 68.5),

(5,'eng300','2017-05-09','2017-11-09', 45.6),

(5,'stat200','2017-05-15','2017-11-15', 65.4),

(6,'comp418','2017-05-15','2017-11-15', 76.2),

(6,'comp418','2017-05-15','2017-11-15', 90.4),

(6,'geog128','2017-05-15','2017-11-15', 78.5),

(7,'math217','2017-01-05','2017-06-05', 77.9),

(7,'comp418','2017-01-05','2017-06-05', 75),

(7,'engl367','2018-01-05','2018-06-05', 95),

(7,'fren282','2018-05-09','2018-11-09', 68.5),

(7,'eng300','2017-03-01','2017-12-01', 45.6),

(7,'stat100','2017-03-01','2017-12-01', 65.4),

(8,'comp314','2017-02-05','2017-07-05', 72.5),

(9,'comp418','2017-01-05','2017-06-05', 32.3),

(9,'geog128','2017-01-05','2017-06-05', 67.4),

(9,'math130','2017-01-05','2017-06-05', 73.8),

(10,'comp102','2017-01-05','2017-06-05', 100),

(10,'engl402','2017-01-05','2017-06-05', 86),

(10,'fren282','2017-05-09','2017-11-09', 68.5),

(10,'entr201','2017-05-09','2017-11-09', 85.6),

(10,'stat349','2017-05-09','2017-11-09', 65.4),

(11,'geog252','2017-01-05','2017-06-05', 45),

(11,'comp418','2017-01-05','2017-06-05', 65),

(11,'geog128','2017-06-05','2017-12-05', 87.4),

(11,'math217','2017-06-05','2017-12-05', 77.9),

(12,'comp418','2017-01-05','2017-06-05', 75),

(12,'engl120','2017-01-05','2017-06-05', 95),

(12,'fren282','2017-04-19','2017-10-19', 68.5),

(12,'eng300','2017-04-19','2017-10-19', 75.6),

(12,'stat200','2017-05-09','2017-11-09', 65.4),

(13,'comp314','2017-01-05','2017-06-05', 72.5),

(13,'comp418','2017-01-05','2017-06-05', 95.7),

(13,'geog128','2017-01-05','2017-06-05', 87.4),

(14,'math333','2017-10-25','2018-05-25', 89.7),

(14,'civl364','2017-10-25','2018-05-25', 96.4),

(14,'eng402','2017-10-25','2018-05-25', 92.6),

(15,'fren282','2017-05-09','2017-11-09', 68.5),

(15,'eng300','2017-05-09','2017-11-09', 45.6),

(15,'stat200','2017-05-15','2017-11-15', 65.4),

(16,'comp418','2017-05-15','2017-11-15', 76.2),

(16,'comp418','2017-05-15','2017-11-15', 90.4),

(16,'geog128','2017-05-15','2017-11-15', 78.5),

(17,'math217','2018-01-05','2018-06-05', 42),

(17,'comp418','2017-01-05','2017-06-05', 35.7),

(17,'engl367','2017-01-05','2017-06-05', 49.2),

(17,'fren282','2017-05-09','2017-11-09', 55),

(17,'eng300','2017-05-09','2017-11-09', 45.6),

(17,'stat100','2017-05-09','2017-11-09', 25.4),

(18,'comp314','2017-01-05','2017-06-05', 92.5),

(19,'comp102','2017-01-05','2017-06-05', 95.7),

(19,'geog346','2017-01-05','2017-06-05', 87.4),

(19,'math217','2017-01-05','2017-06-05', 73.8),

(20,'entr410','2017-09-13','2018-03-13', 95),

(20,'engl402','2017-09-13','2018-03-13', 88.2),

(20,'fren282','2017-09-13','2018-03-13', 85.7),

(20,'entr201','2017-09-13','2018-03-13', 91.6),

(20,'stat349','2017-09-13','2018-03-13', 79.7),

(21,'comp314','2017-01-05','2017-06-05', 100),

(21,'comp418','2017-01-05','2017-06-05', 45.7),

(21,'geog128','2017-01-05','2017-06-05', 87.4),

(21,'math217','2017-01-05','2017-06-05', 77.9),

(22,'comp418','2017-01-05','2017-06-05', 75),

(22,'engl120','2017-01-05','2017-06-05', 95),

(22,'fren282','2017-05-09','2017-11-09', 68.5),

(22,'eng300','2017-05-09','2017-11-09', 45.6),

(22,'stat200','2017-05-09','2017-11-09', 65.4),

(23,'comp314','2017-01-05','2017-06-05', 42.5),

(23,'comp418','2017-01-05','2017-06-05', 75.7),

(23,'geog128','2017-01-05','2017-06-05', 87.4),

(24,'math150','2017-01-05','2017-06-05', 73.8),

(24,'comp102','2017-01-05','2017-06-05', 40),

(24,'engl217','2017-01-05','2017-06-05', 86),

(25,'fren282','2017-05-09','2017-11-09', 68.5),

(25,'eng300','2017-05-09','2017-11-09', 45.6),

(25,'stat200','2017-05-15','2017-11-15', 65.4),

(26,'comp418','2017-05-15','2017-11-15', 76.2),

(26,'comp418','2017-05-15','2017-11-15', 60.4),

(26,'geog128','2017-05-15','2017-11-15', 78.5),

(27,'math217','2017-01-05','2017-06-05', 77.9),

(27,'comp418','2017-01-05','2017-06-05', 75),

(27,'engl367','2018-01-05','2018-06-05', 45),

(27,'fren282','2018-05-09','2018-11-09', 68.5),

(27,'eng300','2017-03-01','2017-12-01', 65.6),

(27,'stat100','2017-03-01','2017-12-01', 65.4),

(28,'comp314','2017-02-05','2017-07-05', 72.5),

(29,'comp418','2017-01-05','2017-06-05', 72.3),

(29,'geog128','2017-01-05','2017-06-05', 67.4),

(29,'math130','2017-01-05','2017-06-05', 23.8),

(30,'comp102','2017-01-05','2017-06-05', 80),

(30,'engl402','2017-01-05','2017-06-05', 86),

(30,'fren282','2017-05-09','2017-11-09', 68.5),

(30,'entr201','2017-05-09','2017-11-09', 85.6),

(30,'stat349','2017-05-09','2017-11-09', 65.4),

(31,'geog252','2017-01-05','2017-06-05', 45),

(31,'comp418','2017-01-05','2017-06-05', 65),

(31,'geog128','2017-06-05','2017-12-05', 87.4),

(31,'math217','2017-06-05','2017-12-05', 47.9),

(32,'comp418','2017-01-05','2017-06-05', 75),

(32,'engl120','2017-01-05','2017-06-05', 55),

(32,'fren282','2017-04-19','2017-10-19', 68.5),

(32,'eng300','2017-04-19','2017-10-19', 75.6),

(32,'stat200','2017-05-09','2017-11-09', 65.4),

(33,'comp314','2017-01-05','2017-06-05', 72.5),

(33,'comp418','2017-01-05','2017-06-05', 95.7),

(33,'geog128','2017-01-05','2017-06-05', 87.4),

(34,'math333','2017-10-25','2018-05-25', 89.7),

(34,'civl364','2017-10-25','2018-05-25', 96.4),

(34,'eng402','2017-10-25','2018-05-25', 92.6),

(35,'fren282','2017-05-09','2017-11-09', 68.5),

(35,'eng300','2017-05-09','2017-11-09', 45.6),

(35,'stat200','2017-05-15','2017-11-15', 65.4),

(36,'comp418','2017-05-15','2017-11-15', 76.2),

(36,'comp418','2017-05-15','2017-11-15', 90.4),

(36,'geog128','2017-05-15','2017-11-15', 78.5),

(37,'math217','2018-01-05','2018-06-05', 42),

(37,'comp418','2017-01-05','2017-06-05', 35.7),

(37,'engl367','2017-01-05','2017-06-05', 49.2),

(37,'fren282','2017-05-09','2017-11-09', 55),

(37,'eng300' ,'2017-05-09','2017-11-09', 45.6),

(37,'stat100','2017-05-09','2017-11-09', 25.4),

(38,'comp314','2017-01-05','2017-06-05', 92.5),

(39,'comp102','2017-01-05','2017-06-05', 95.7),

(39,'geog346','2017-01-05','2017-06-05', 87.4),

(39,'math217','2017-01-05','2017-06-05', 73.8),

(40,'entr410','2017-09-13','2018-03-13', 34),

(40,'engl402','2017-09-13','2018-03-13', 52.2),

(40,'fren282','2017-09-13','2018-03-13', 37.7),

(40,'entr201','2017-09-13','2018-03-13', 15.6),

(40,'stat349','2017-09-13','2018-03-13', 20.7),

(41,'fren115','2017-10-17','2018-04-17', 86.7 ),

(41,'civl374','2017-10-17','2018-04-17', 78.7 ),

(41,'entr410','2017-10-17','2018-04-17', 82.9 ),

(41,'geog433','2017-10-17','2018-04-17', 77.4 ),

(42,'math217','2017-10-23','2018-04-23', 46.2 ),

(42,'eng402' ,'2017-10-23','2018-04-23', 32.7 ),

(42,'engl367','2017-10-23','2018-04-23', 36.4 ),

(42,'geog252','2017-10-23','2018-04-23', 29.2 ),

(43,'stat200','2017-10-29','2018-04-29', 93.1 ),

(43,'math333','2017-10-29','2018-04-29', 88.4 ),

(44,'math150','2017-10-29','2018-04-29', 72.4 ),

(44,'engl120','2017-10-29','2018-04-29', 42.6 ),

(44,'civl374','2017-10-29','2018-04-29', 32.1 ),

(44,'comp102','2017-10-29','2018-04-29', 87.8 ),

(45,'math150','2017-11-01','2018-05-01', 70.4 ),

(45,'stat100','2017-11-01','2018-05-01', 64.4 ),

(45,'geog128','2017-11-01','2018-05-01', 49.1 ),

(46,'comp418','2017-11-01','2018-05-01', 42.7 ),

(46,'comp314','2017-11-01','2018-05-01', 36.9 ),

(46,'stat200','2017-11-01','2018-05-01', 60.8 ),

(46,'engl093','2017-11-01','2018-05-01', 42.7 ),

(47,'entr410','2017-11-02','2018-05-02', 77.4 ),

(47,'geog433','2017-11-02','2018-05-02', 87.4 ),

(47,'fren282','2017-11-02','2018-05-02', 92.5 ),

(47,'math130','2017-11-02','2018-05-02', 85.8 ),

(48,'math217','2017-02-07','2018-08-07', 78.8 ),

(48,'engl120','2017-02-07','2018-08-07', 81.3 ),

(48,'entr310','2017-02-07','2018-08-07', 86.2 ),

(49,'math217','2017-02-07','2018-08-07', 78.8 ),

(49,'eng300' ,'2017-02-07','2018-08-07', 76.3 ),

(49,'entr410','2017-02-07','2018-08-07', 87.6 ),

(50,'comp418','2017-02-09','2018-08-09', 48.8 ),

(50,'stat463','2017-02-09','2018-08-09', 34.3 ),

(50,'geog433','2017-02-09','2018-08-09', 47.2 ),

(50,'entr410','2017-02-09','2018-08-09', 29.8 ),

(50,'eng401' ,'2017-02-09','2018-08-09', 43.3 ),

(50,'civl404','2017-02-09','2018-08-09', 27.2 ),

(51,'civl374','2017-10-17','2018-04-17', 78.7 ),

(51,'entr410','2017-10-17','2018-04-17', 82.9 ),

(51,'geog433','2017-10-17','2018-04-17', 77.4 ),

(52,'math217','2017-10-23','2018-04-23', 46.2 ),

(52,'eng402' ,'2017-10-23','2018-04-23', 32.7 ),

(52,'engl367','2017-10-23','2018-04-23', 36.4 ),

(52,'geog252','2017-10-23','2018-04-23', 29.2 ),

(53,'stat200','2017-10-29','2018-04-29', 93.1 ),

(53,'math333','2017-10-29','2018-04-29', 88.4 ),

(54,'math150','2017-10-29','2018-04-29', 72.4 ),

(54,'engl120','2017-10-29','2018-04-29', 42.6 ),

(54,'civl374','2017-10-29','2018-04-29', 32.1 ),

(54,'comp102','2017-10-29','2018-04-29', 87.8 ),

(55,'math150','2017-11-01','2018-05-01', 70.4 ),

(55,'stat100','2017-11-01','2018-05-01', 64.4 ),

(55,'geog128','2017-11-01','2018-05-01', 49.1 ),

(56,'comp418','2017-11-01','2018-05-01', 42.7 ),

(56,'comp314','2017-11-01','2018-05-01', 36.9 ),

(56,'stat200','2017-11-01','2018-05-01', 60.8 ),

(56,'engl093','2017-11-01','2018-05-01', 42.7 ),

(57,'entr410','2017-11-02','2018-05-02', 77.4 ),

(57,'geog433','2017-11-02','2018-05-02', 87.4 ),

(57,'fren282','2017-11-02','2018-05-02', 92.5 ),

(57,'math130','2017-11-02','2018-05-02', 85.8 ),

(58,'math217','2017-02-07','2018-08-07', 78.8 ),

(58,'engl120','2017-02-07','2018-08-07', 81.3 ),

(58,'entr310','2017-02-07','2018-08-07', 86.2 ),

(59,'math217','2017-02-07','2018-08-07', 78.8 ),

(59,'eng300' ,'2017-02-07','2018-08-07', 76.3 ),

(59,'entr410','2017-02-07','2018-08-07', 87.6 ),

(60,'comp418','2017-02-09','2018-08-09', 88.8 ),

(60,'stat463','2017-02-09','2018-08-09', 94.3 ),

(60,'geog433','2017-02-09','2018-08-09', 77.2 ),

(60,'entr410','2017-02-09','2018-08-09', 79.8 ),

(61,'comp314','2017-01-05','2017-06-05', 100),

(61,'comp418','2017-01-05','2017-06-05', 45.7),

(61,'geog128','2017-01-05','2017-06-05', 87.4),

(61,'math217','2017-01-05','2017-06-05', 77.9),

(62,'comp418','2017-01-05','2017-06-05', 75),

(62,'engl120','2017-01-05','2017-06-05', 95),

(62,'fren282','2017-05-09','2017-11-09', 68.5),

(62,'eng300','2017-05-09','2017-11-09', 45.6),

(62,'stat200','2017-05-09','2017-11-09', 65.4),

(63,'comp314','2017-01-05','2017-06-05', 42.5),

(63,'comp418','2017-01-05','2017-06-05', 75.7),

(63,'geog128','2017-01-05','2017-06-05', 87.4),

(64,'math150','2017-01-05','2017-06-05', 73.8),

(64,'comp102','2017-01-05','2017-06-05', 40),

(64,'engl217','2017-01-05','2017-06-05', 86),

(65,'fren282','2017-05-09','2017-11-09', 68.5),

(65,'eng300','2017-05-09','2017-11-09', 45.6),

(65,'stat200','2017-05-15','2017-11-15', 65.4),

(66,'comp418','2017-05-15','2017-11-15', 76.2),

(66,'comp418','2017-05-15','2017-11-15', 60.4),

(66,'geog128','2017-05-15','2017-11-15', 78.5),

(67,'math217','2017-01-05','2017-06-05', 77.9),

(67,'comp418','2017-01-05','2017-06-05', 75),

(67,'engl367','2018-01-05','2018-06-05', 45),

(67,'fren282','2018-05-09','2018-11-09', 68.5),

(67,'eng300','2017-03-01','2017-12-01', 65.6),

(67,'stat100','2017-03-01','2017-12-01', 65.4),

(68,'comp314','2017-02-05','2017-07-05', 72.5),

(69,'comp418','2017-01-05','2017-06-05', 72.3),

(69,'geog128','2017-01-05','2017-06-05', 67.4),

(69,'math130','2017-01-05','2017-06-05', 23.8),

(70,'comp102','2017-01-05','2017-06-05', 80),

(70,'engl402','2017-01-05','2017-06-05', 86),

(70,'fren282','2017-05-09','2017-11-09', 68.5),

(70,'entr201','2017-05-09','2017-11-09', 85.6),

(70,'stat349','2017-05-09','2017-11-09', 65.4),

(71,'geog252','2017-01-05','2017-06-05', 45),

(71,'comp418','2017-01-05','2017-06-05', 65),

(71,'geog128','2017-06-05','2017-12-05', 87.4),

(71,'math217','2017-06-05','2017-12-05', 47.9),

(72,'comp418','2017-01-05','2017-06-05', 75),

(2,'engl120','2017-01-05','2017-06-05', 55),

(72,'fren282','2017-04-19','2017-10-19', 68.5),

(72,'eng300','2017-04-19','2017-10-19', 75.6),

(72,'stat200','2017-05-09','2017-11-09', 65.4),

(73,'comp314','2017-01-05','2017-06-05', 72.5),

(73,'comp418','2017-01-05','2017-06-05', 95.7),

(73,'geog128','2017-01-05','2017-06-05', 87.4),

(74,'math333','2017-10-25','2018-05-25', 89.7),

(74,'civl364','2017-10-25','2018-05-25', 96.4),

(74,'eng402','2017-10-25','2018-05-25', 92.6),

(75,'fren282','2017-05-09','2017-11-09', 68.5),

(75,'eng300','2017-05-09','2017-11-09', 45.6),

(75,'stat200','2017-05-15','2017-11-15', 65.4),

(76,'comp418','2017-05-15','2017-11-15', 76.2),

(76,'comp418','2017-05-15','2017-11-15', 90.4),

(76,'geog128','2017-05-15','2017-11-15', 78.5),

(77,'math217','2018-01-05','2018-06-05', 42),

(77,'comp418','2017-01-05','2017-06-05', 35.7),

(77,'engl367','2017-01-05','2017-06-05', 49.2),

(77,'fren282','2017-05-09','2017-11-09', 55),

(77,'eng300' ,'2017-05-09','2017-11-09', 45.6),

(77,'stat100','2017-05-09','2017-11-09', 25.4),

(78,'comp314','2017-01-05','2017-06-05', 92.5),

(79,'comp102','2017-01-05','2017-06-05', 95.7),

(79,'geog346','2017-01-05','2017-06-05', 87.4),

(79,'math217','2017-01-05','2017-06-05', 73.8),

(80,'entr410','2017-09-13','2018-03-13', 34),

(80,'engl402','2017-09-13','2018-03-13', 52.2),

(80,'fren282','2017-09-13','2018-03-13', 37.7),

(80,'entr201','2017-09-13','2018-03-13', 15.6),

(81,'comp314','2017-01-05','2017-06-05', 100),

(81,'comp418','2017-01-05','2017-06-05', 95.7),

(81,'geog128','2017-01-05','2017-06-05', 87.4),

(81,'math217','2017-01-05','2017-06-05', 77.9),

(82,'comp418','2017-01-05','2017-06-05', 75),

(82,'engl120','2017-01-05','2017-06-05', 95),

(82,'fren282','2017-05-09','2017-11-09', 68.5),

(82,'eng300','2017-05-09','2017-11-09', 45.6),

(82,'stat200','2017-05-09','2017-11-09', 65.4),

(83,'comp314','2017-01-05','2017-06-05', 72.5),

(83,'comp418','2017-01-05','2017-06-05', 95.7),

(83,'geog128','2017-01-05','2017-06-05', 87.4),

(84,'math150','2017-01-05','2017-06-05', 73.8),

(84,'comp102','2017-01-05','2017-06-05', 100),

(84,'engl217','2017-01-05','2017-06-05', 86),

(85,'fren282','2017-05-09','2017-11-09', 68.5),

(85,'eng300','2017-05-09','2017-11-09', 45.6),

(85,'stat200','2017-05-15','2017-11-15', 65.4),

(86,'comp418','2017-05-15','2017-11-15', 76.2),

(86,'comp418','2017-05-15','2017-11-15', 90.4),

(86,'geog128','2017-05-15','2017-11-15', 78.5),

(87,'math217','2017-01-05','2017-06-05', 77.9),

(87,'comp418','2017-01-05','2017-06-05', 75),

(87,'engl367','2018-01-05','2018-06-05', 95),

(87,'fren282','2018-05-09','2018-11-09', 68.5),

(87,'eng300','2017-03-01','2017-12-01', 45.6),

(87,'stat100','2017-03-01','2017-12-01', 65.4),

(88,'comp314','2017-02-05','2017-07-05', 72.5),

(89,'comp418','2017-01-05','2017-06-05', 32.3),

(89,'geog128','2017-01-05','2017-06-05', 67.4),

(89,'math130','2017-01-05','2017-06-05', 73.8);