Hash Index on Student Information Table <student_id>

```
db.query("CREATE INDEX stu_id USING hash ON student_information(student_id)"
```

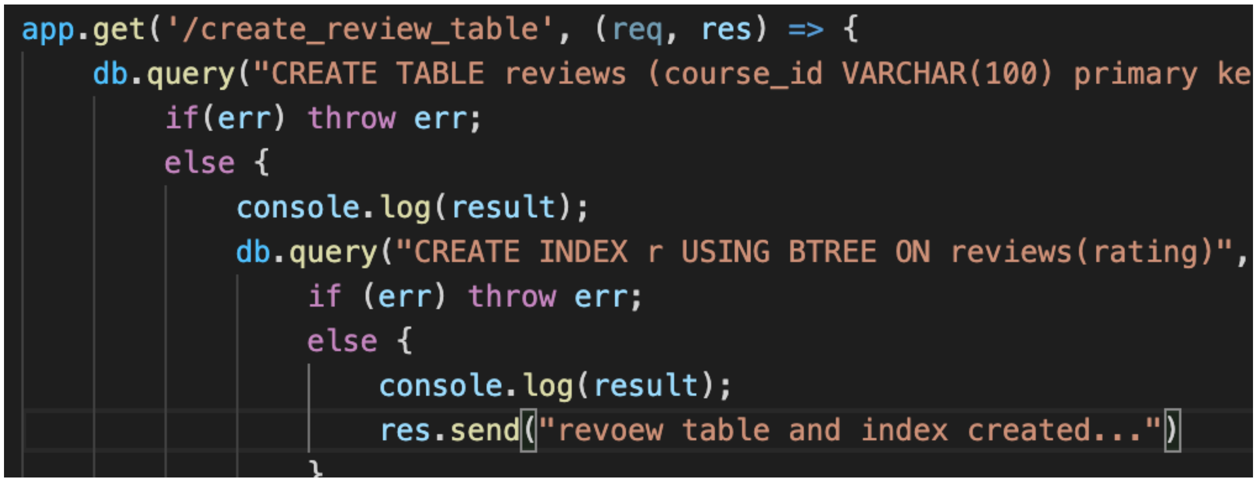
Used in the login page:

```
app.post("/login_student", (req, res) => {
   db.query(
       "Select * FROM student_information WHERE student_id = ?",
```

We want to get a specific users information in order to verify their login credentials and store the user information and the hash index is very useful in this case. The program then stores the users information for the duration of the session.

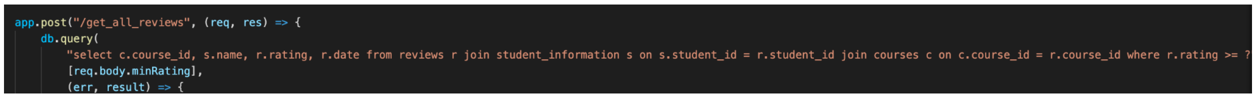
Clustered B+ Tree Index on the review table <rating>

A report that a user can generate is a report where a user can select a minimum grade and the screen will return all of the reviews that are greater than or equal to a certain rating.



```
app.get('/create_review_table', (req, res) => {
    db.query("CREATE TABLE reviews (course_id VARCHAR(100) primary ke
        if(err) throw err;
        else {
            console.log(result);
            db.query("CREATE INDEX r USING BTREE ON reviews(rating)",
                if (err) throw err;
                else {
                    console.log(result);
                    res.send("revoew table and index created...")
                }
```

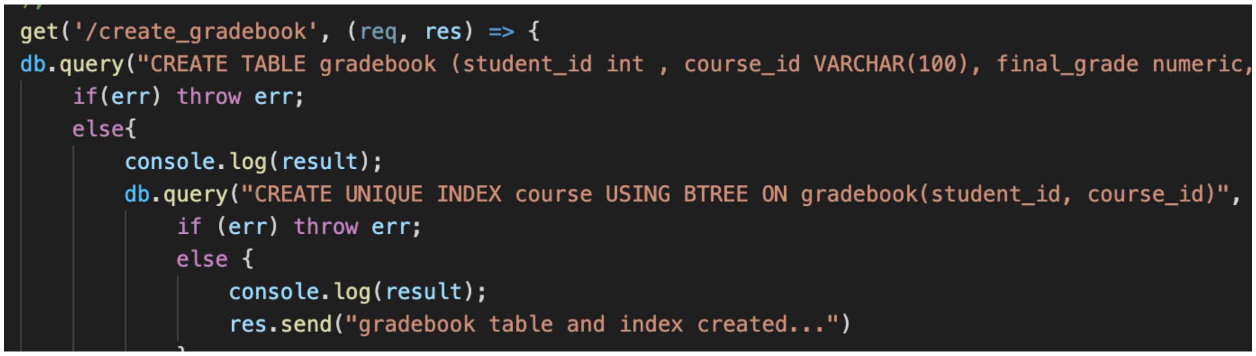
This is a specific query that this index supports:



```
app.post("/get_all_reviews", (req, res) => {
    db.query(
        "select c.course_id, s.name, r.rating, r.date from reviews r join student_information s on s.student_id = r.student_id join courses c on c.course_id = r.course_id where r.rating >= ?
        [req.body.minRating],
        (err, result) => {
```

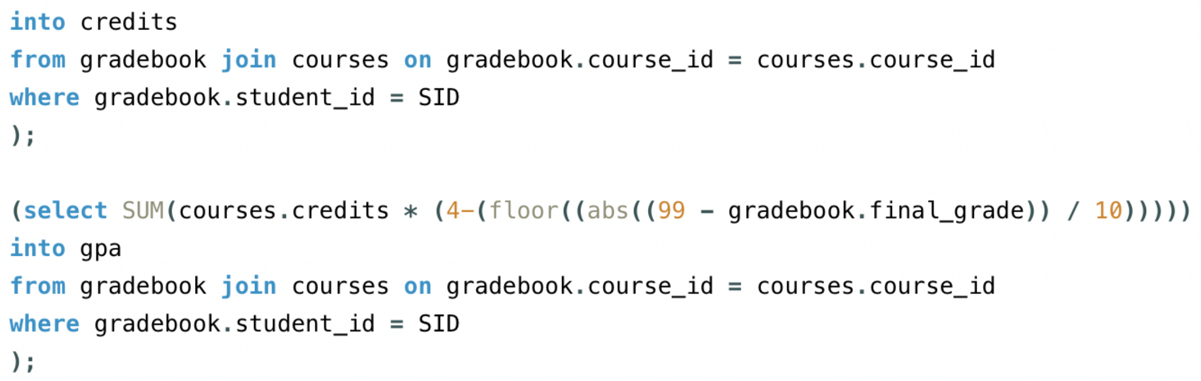
Clustered B+ Tree index on the gradebook table <student_id, course_id>

In all of the queries associated with the gradebook table we are usually trying to find the course information for a specific student and a specific course, therefore we can have a clustered index on the student_id
We create the index below:



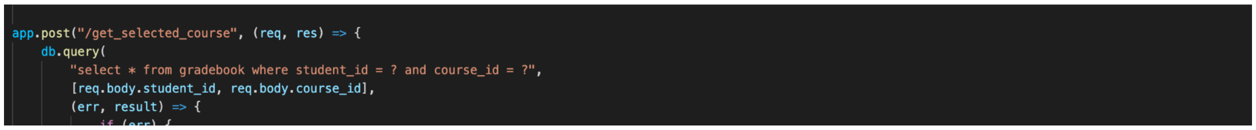
```
get('/create_gradebook', (req, res) => {
db.query("CREATE TABLE gradebook (student_id int , course_id VARCHAR(100), final_grade numeric,
    if(err) throw err;
    else{
        console.log(result);
        db.query("CREATE UNIQUE INDEX course USING BTREE ON gradebook(student_id, course_id)",
            if (err) throw err;
            else {
                console.log(result);
                res.send("gradebook table and index created...")
```

We specifically use this index in one of our stored procedures where we try to find the gpa of a specific student by querying the gradebook table.



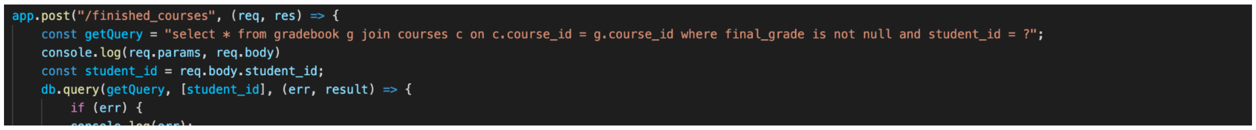
```
into credits
from gradebook join courses on gradebook.course_id = courses.course_id
where gradebook.student_id = SID
);

(select SUM(courses.credits * (4-(floor((abs((99 - gradebook.final_grade)) / 10)))))
into gpa
from gradebook join courses on gradebook.course_id = courses.course_id
where gradebook.student_id = SID
);
```

This index also supports a query we want to get the information for a specific student and course



```
app.post("/get_selected_course", (req, res) => {
    db.query(
        "select * from gradebook where student_id = ? and course_id = ?",
        [req.body.student_id, req.body.course_id],
        (err, result) => {
```

This index is also useful if we want to find all of the courses that a specific student has completed

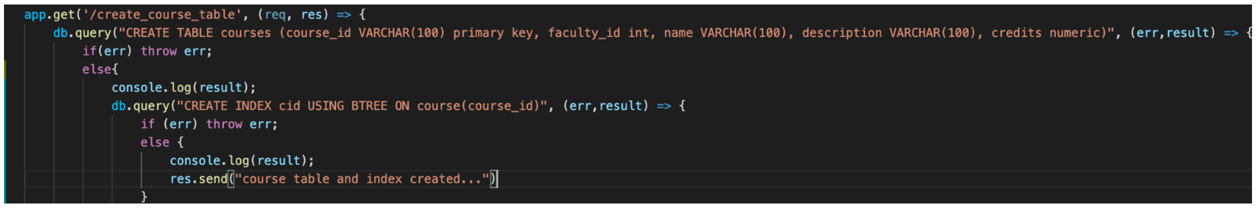


```
app.post("/finished_courses", (req, res) => {
    const getQuery = "select * from gradebook g join courses c on c.course_id = g.course_id where final_grade is not null and student_id = ?";
    console.log(req.params, req.body)
    const student_id = req.body.student_id;
    db.query(getQuery, [student_id], (err, result) => {
        if (err) {
```

Unclustered B+ Index on the Course Table <course_id>

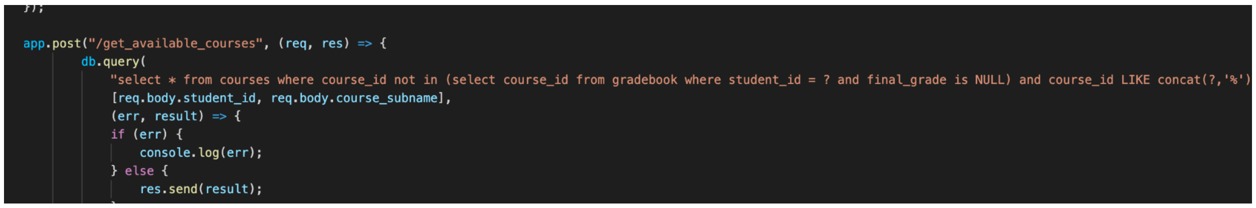
We are usually trying to search for courses based on their course id so an unclustered index on that column would allow us to only find the courses that we are looking for. This cluster will lessen the amount to calls to main memory to get the list of the courses that match the search.

We create the index here:



```
app.get('/create_course_table', (req, res) => {
    db.query("CREATE TABLE courses (course_id VARCHAR(100) primary key, faculty_id int, name VARCHAR(100), description VARCHAR(100), credits numeric)", (err,result) => {
        if(err) throw err;
        else{
            console.log(result);
            db.query("CREATE INDEX cid USING BTREE ON course(course_id)", (err,result) => {
                if (err) throw err;
                else {
                    console.log(result);
                    res.send("course table and index created...")
                }
```

And this query supports the report where you can filter the list of available courses using a search.



```
});

app.post("/get_available_courses", (req, res) => {
        db.query(
            "select * from courses where course_id not in (select course_id from gradebook where student_id = ? and final_grade is NULL) and course_id LIKE concat(?,'%')
            [req.body.student_id, req.body.course_subname],
            (err, result) => {
            if (err) {
                console.log(err);
            } else {
                res.send(result);
```