SQL To Knex Assignment

For this assignment you will be taking your knowledge of SQL and Knex and writing the necessary Knex code to output SQL below. Fork and clone this repo and submit a pull request with your answers. You can use http://michaelavila.com/knex-querylab/ to check your answers

urn the following SQL queries into Knex queries (you can write them next to each SQL query or below):

1. `SELECT * FROM students;`

```
knex('students').select();
```

1. `SELECT * FROM students WHERE id=1;`

```
knex('students').select().where({id:1})
```

1. `SELECT * FROM students WHERE id=5; LIMIT 1`

```
knex('students').select().where({id:5}).limit(1);
```

1. `SELECT COUNT(*) students;`

```
knex('students').count();
```

1. `SELECT MIN('year') FROM students;`

```
knex('students').min('year');
```

1. `SELECT * FROM students WHERE name IS NOT NULL;`

```
knex('students').whereNot('name', null);
```

1. `SELECT * FROM todos WHERE id IN ('1', '2', '3') OR user_id IN ('4', '5', '6');`

```
knex('todos').whereIn('id', ['1', '2', '3']).orWhereIn('user_id', ['4', '5', '6']
```

1. `SELECT * FROM students LIMIT 10 OFFSET 30;`

```
knex('students').limit(10).offset(30);
```

1. `INSERT INTO students (name,fav_color) VALUES ('tyler','purple');`

```
knex('students').insert({
                  name: 'tyler',
                  fav_color: 'purple'
                });
```

1. `INSERT INTO students (name,fav_color) VALUES ('liz','blue') RETURNING *;`

```
knex('students').insert({
                  name: 'liz',
                  fav_color: 'blue'
                }).returning('*');
```

1. `UPDATE students SET name ='cho' WHERE id=5;`

```
knex('students').where({id:5}).update({
                                  name: 'cho'
                                }).offset(30);
```

1. `DELETE * FROM students;`

```
knex('students').del();
```

1. `UPDATE "students" SET "score" = "score" + 10 WHERE id=1;`

```
knex('students').where({id:1}).increment("score",10);
```

1. `SELECT * FROM "students" LEFT OUTER JOIN "todos" ON "students"."id" =`

`"todos"."student_id".`

```
knex('students').leftOuterJoin('todos', 'students.id', 'todos.student_id');
```

1. `SELECT * FROM "students" RIGHT OUTER JOIN "todos" ON "students"."id" = "todos"."student_id";`

```
knex('students').rightOuterJoin('todos', 'students.id', 'todos.student_id');
```

nswer the following questions:

1. See the documentation for `pluck` - when would a method like this be useful?

> This will pluck the specified column from each row in your results, yielding a promise which resolves to the array of values selected.

```
knex('users').pluck('id').then(function(ids) {
  console.log(ids);
});
```

1. How do you specify that a column must be unique using Knex?

```
knex.schema.createTable('accounts', function(table) {
  table.increments().primary();
  table.string('email').unique();
});
```

1. How do you specify that a column can not be NULL using Knex?

```
knex.schema.createTable('address', function(table) {
  t.increments().primary();
  table.string('city',50).notNullable();
  table.string('state',2).notNullable();
  table.integer('zip',5).unsigned().notNullable();
});
```

1. Can you also write raw SQL within knex queries? If so, how do you do that?

> Raw expressions are created by using knex.raw(sql, [bindings]) and passing this as a value for any value in the query chain.

```
knex('users')
  .select(knex.raw('count(*) as user_count, status'))
  .where(knex.raw(1))
  .groupBy('status')
```

Outputs:

```
SELECT COUNT(*) AS user_count, status FROM "users" WHERE 1 GROUP BY "status"
```