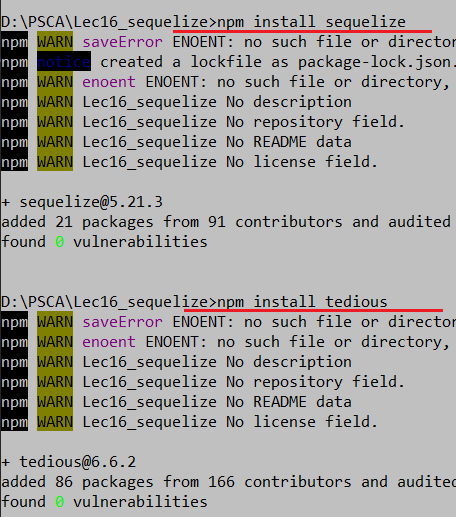
Лекция 15

ПСКП

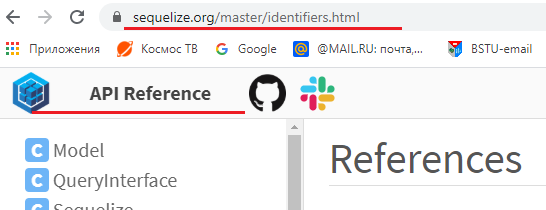
ПОИТ-3

**Node.js: Sequelize**

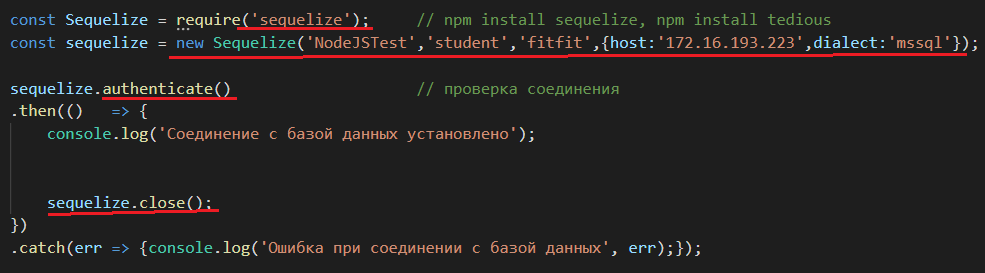
1. **ORM: Object-Relational Mapping**  - технология программирования, которая позволяет работать с SQL-базой данных, как с набором программных объектов. **Mapping**: база данных – объект contextDB, таблица – коллекция объектов, строка в таблице – объект, структура таблицы – класс.
2. **Sequelize:** npm-пакет, реализующий ORM-технологию. Может применяться для: Postgres, MySQL, mariadb, sqlite3, Microsoft SQL Server.

****

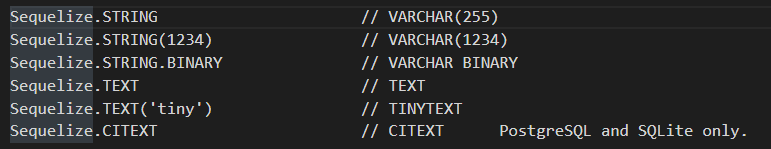
1. **Sequelize:** <https://www.npmjs.com/package/sequelize>
2. **Sequelize:** <https://sequelize.org/master/manual/getting-started>
3. **Sequelize:** <https://metanit.com/web/nodejs/9.1.php>
4. **Sequelize:** документация API

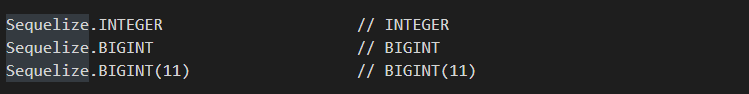
****

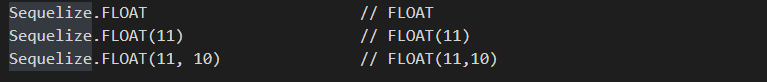
1. **Sequelize:** соединение с БД, проверка соединения, закрытие соединения.

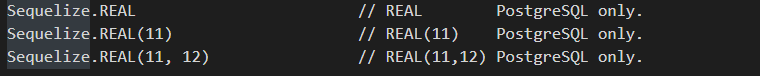
****

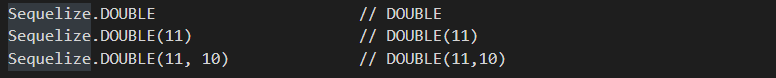
1. **Sequelize:** типы данных

****

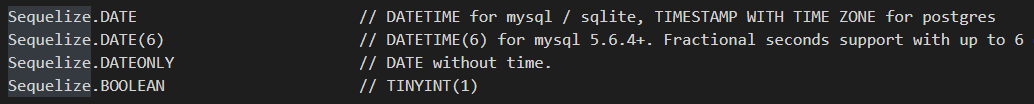
****

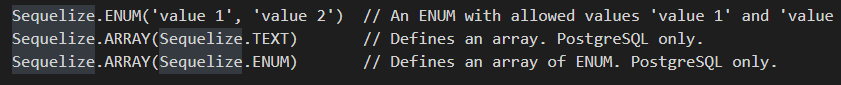
****

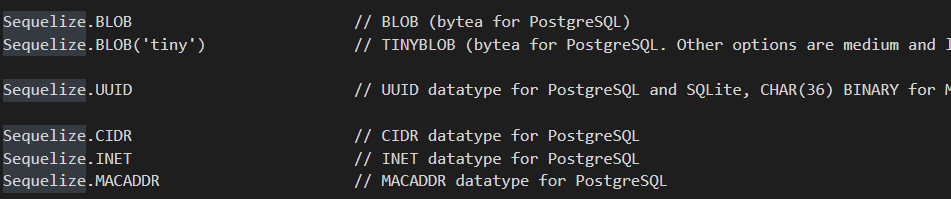
****

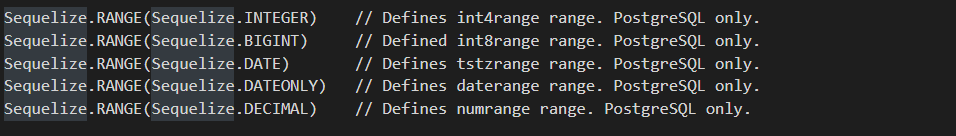
****

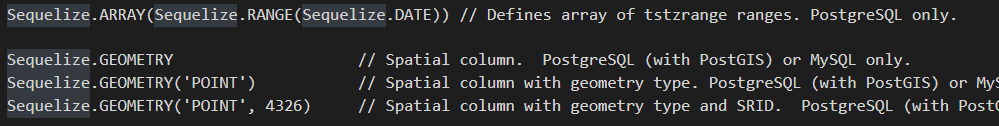
****

****

****

****

****

****

1. **Sequelize:** модель данных

$ npm install --save pg pg-hstore *# Postgres*

$ npm install --save mysql2

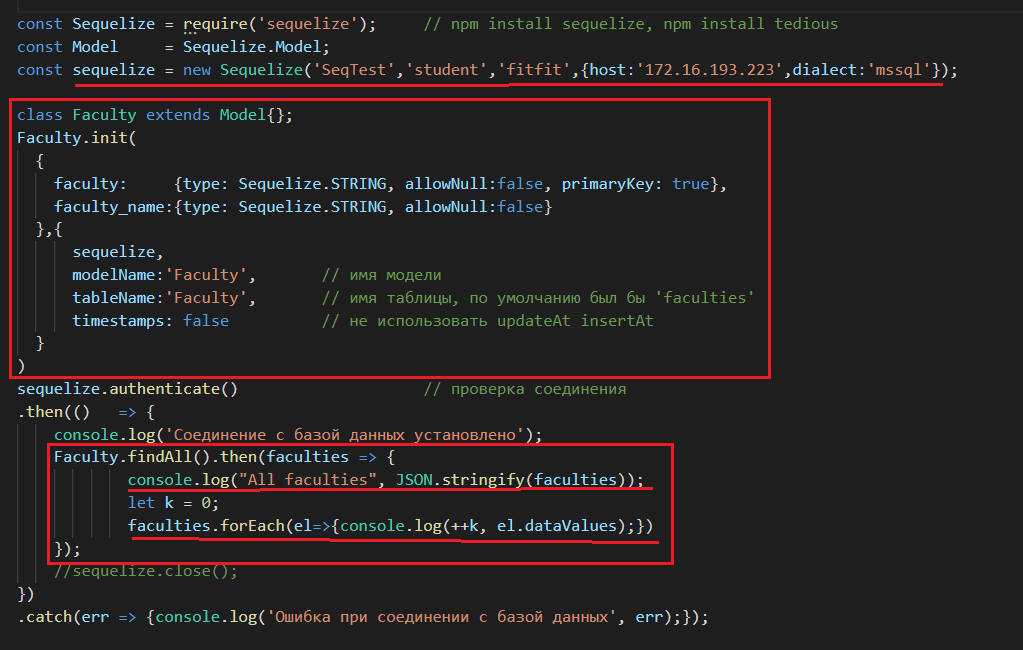
$ npm install --save mariadb

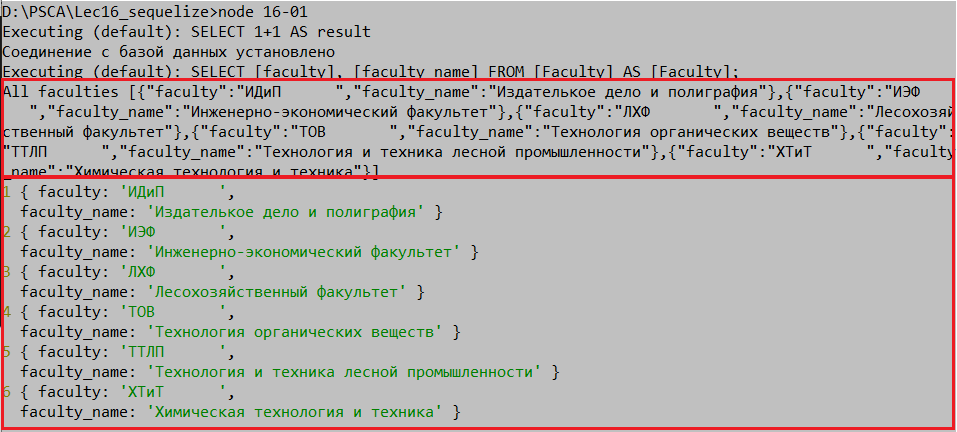
$ npm install --save sqlite3

$ npm install --save tedious *# Microsoft SQL Server*

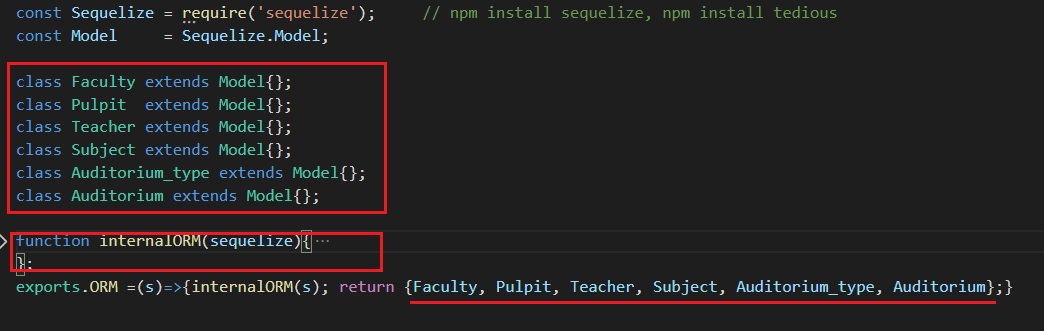


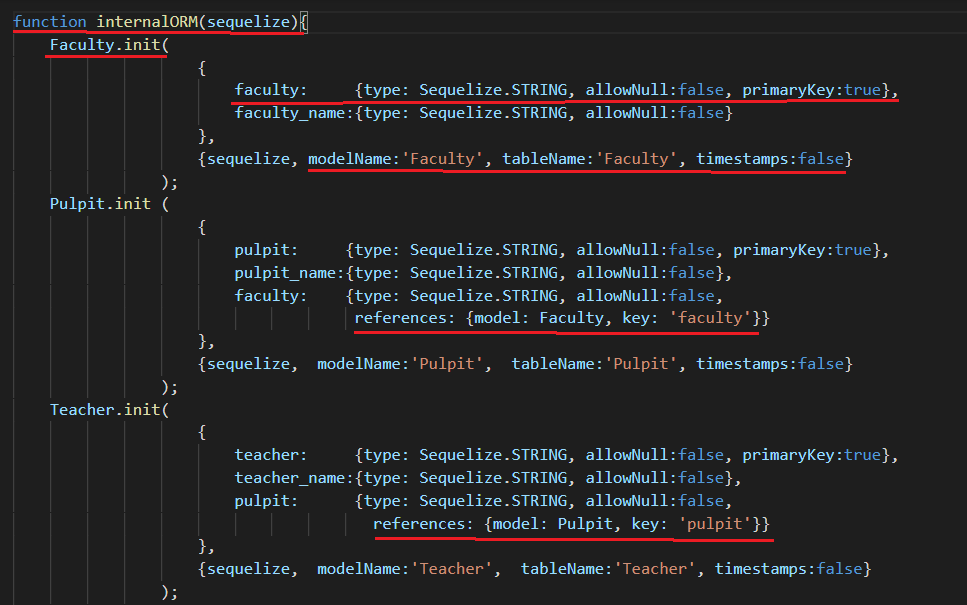
1. **Sequelize:** модель данных

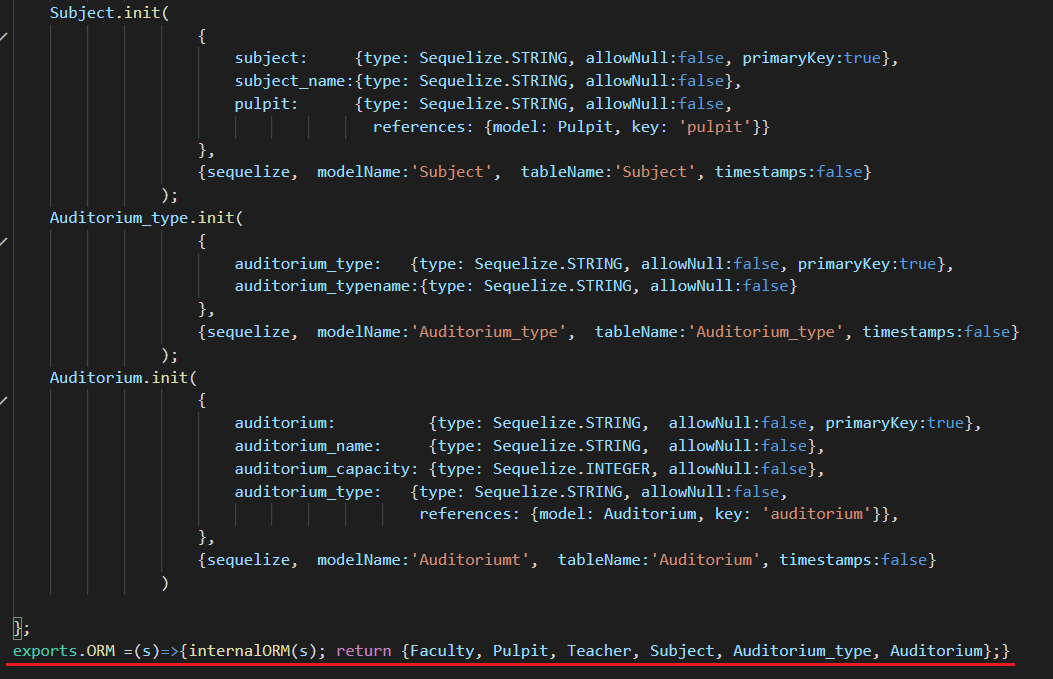
****

****

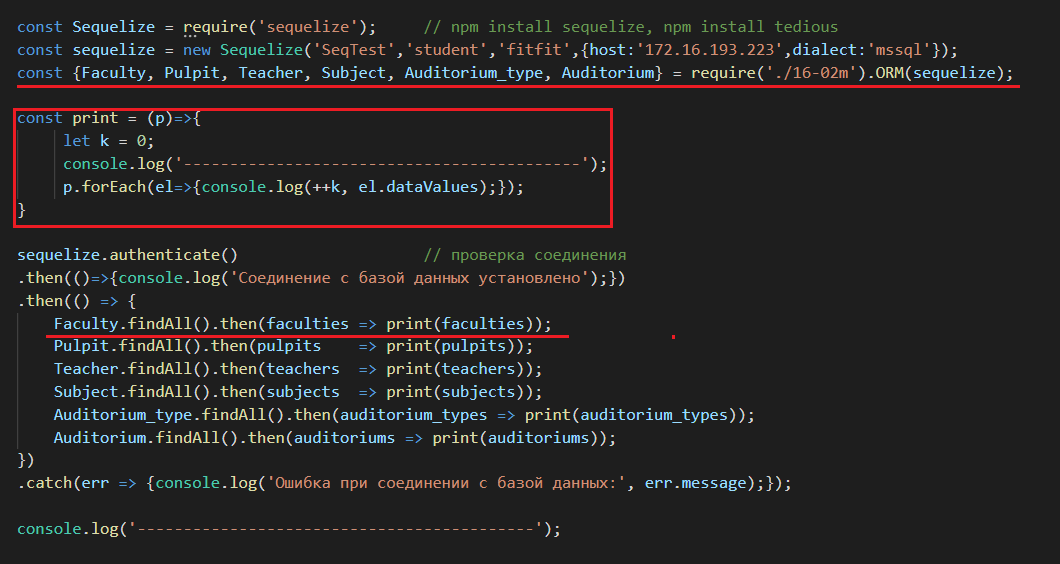
1. **Sequelize:** модель данных, reference

****

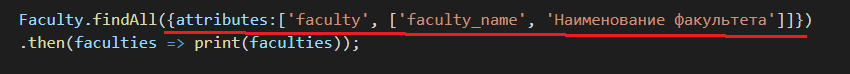
****

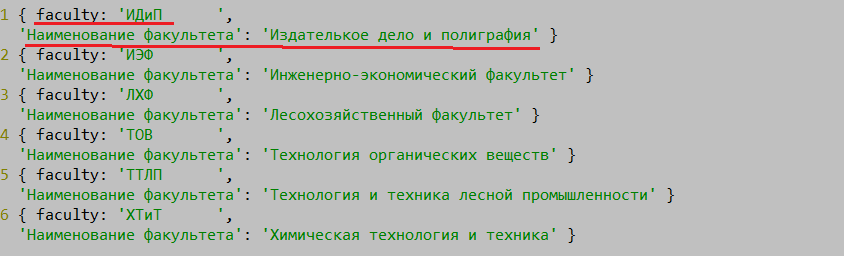
****

1. **Sequelize:** select

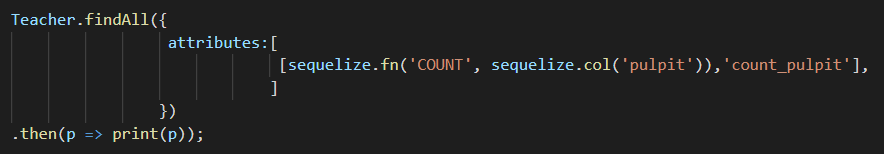
****

1. **Sequelize:** select XX, YY as ’VVV’

****

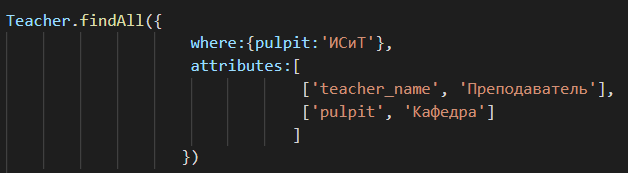
****

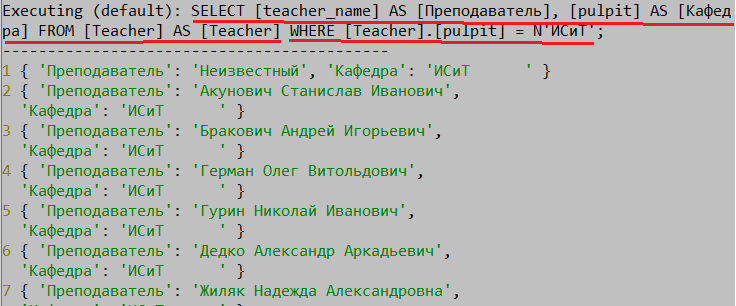
1. **Sequelize:** select COUNT(XX) as ’VVV’

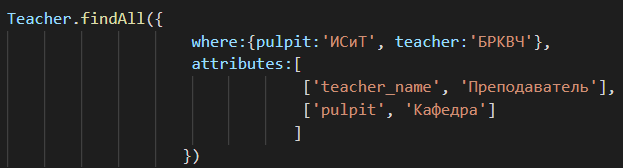
****

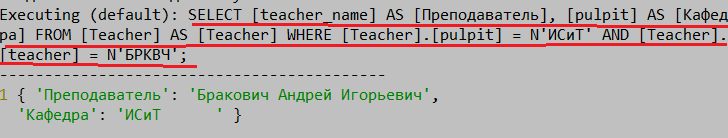
****

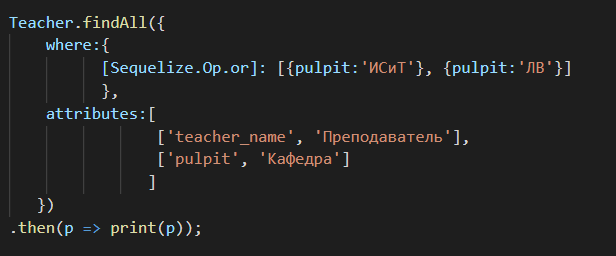
1. **Sequelize:** select …where

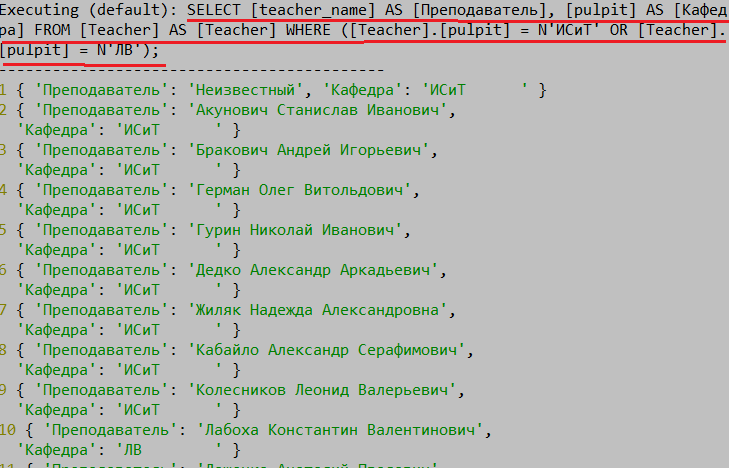
****

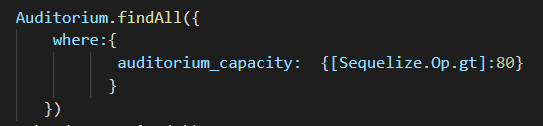
****

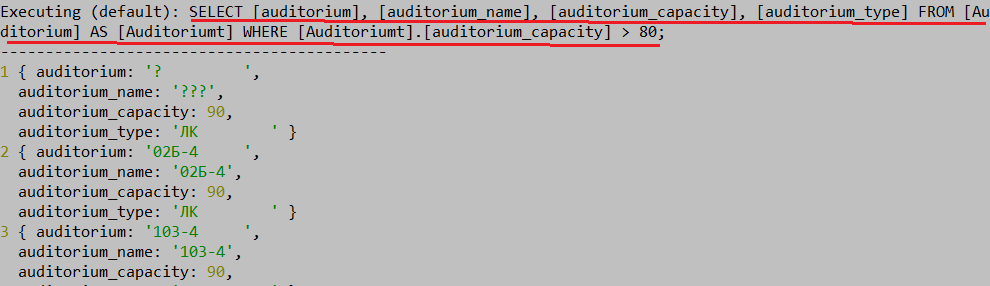
****

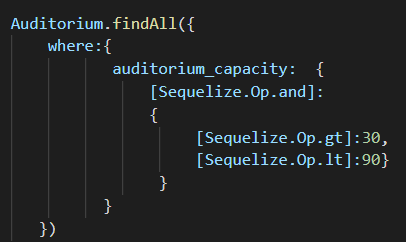
****

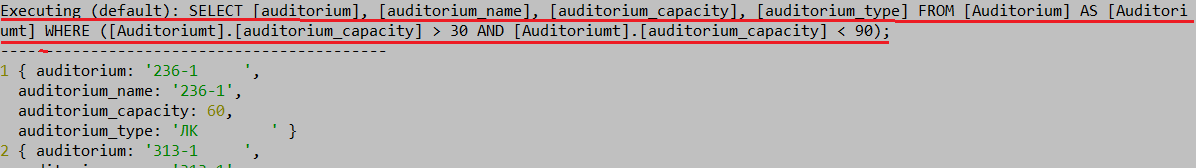
****

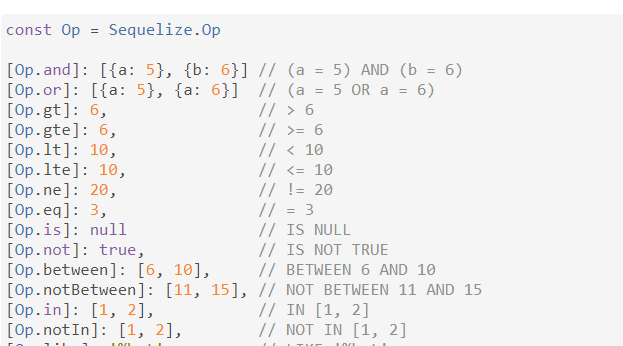
****

****

****

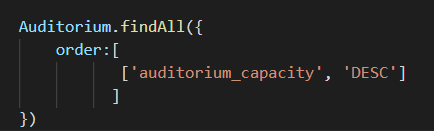
****

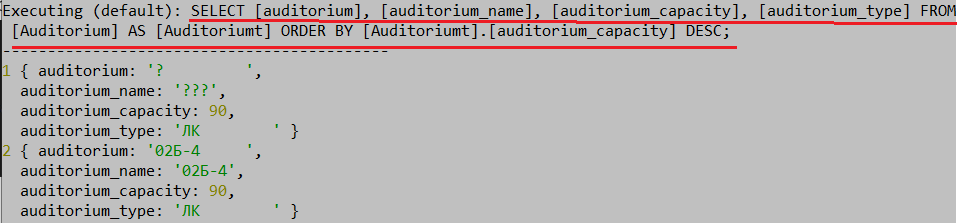
****

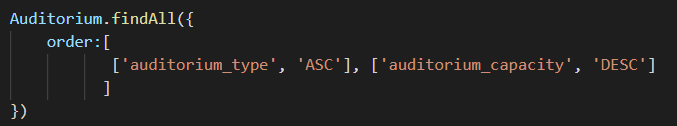
1. **Sequelize:** Sequelize.Op, <https://sequelize.org/master/manual/querying.html>

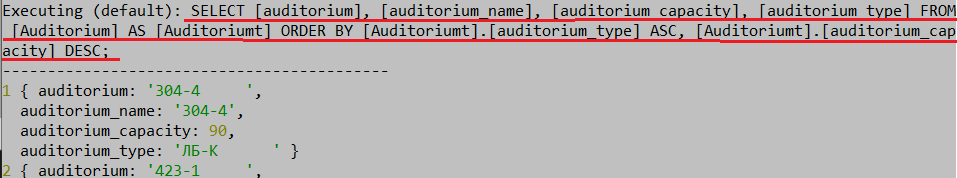
См. <https://sequelize.org/master/manual/querying.html>

1. **Sequelize:** select … order by

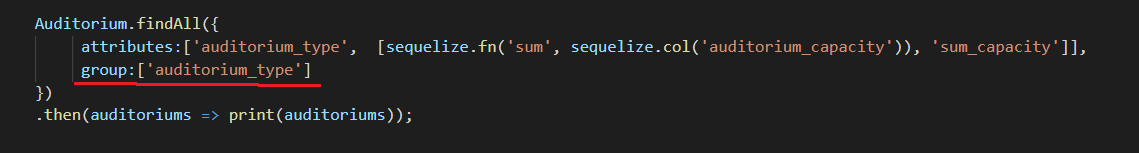
****

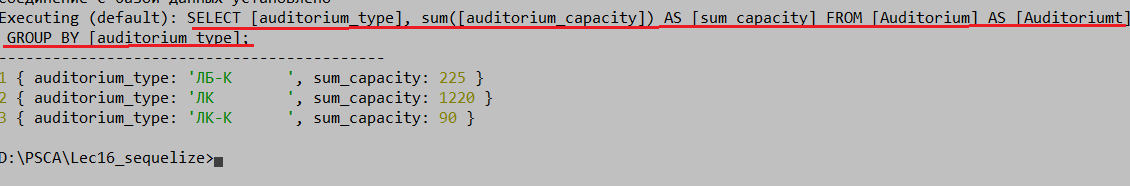
****

****

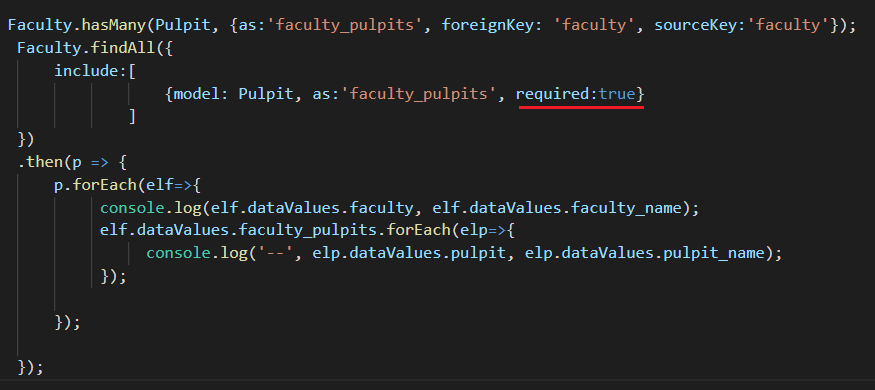
****

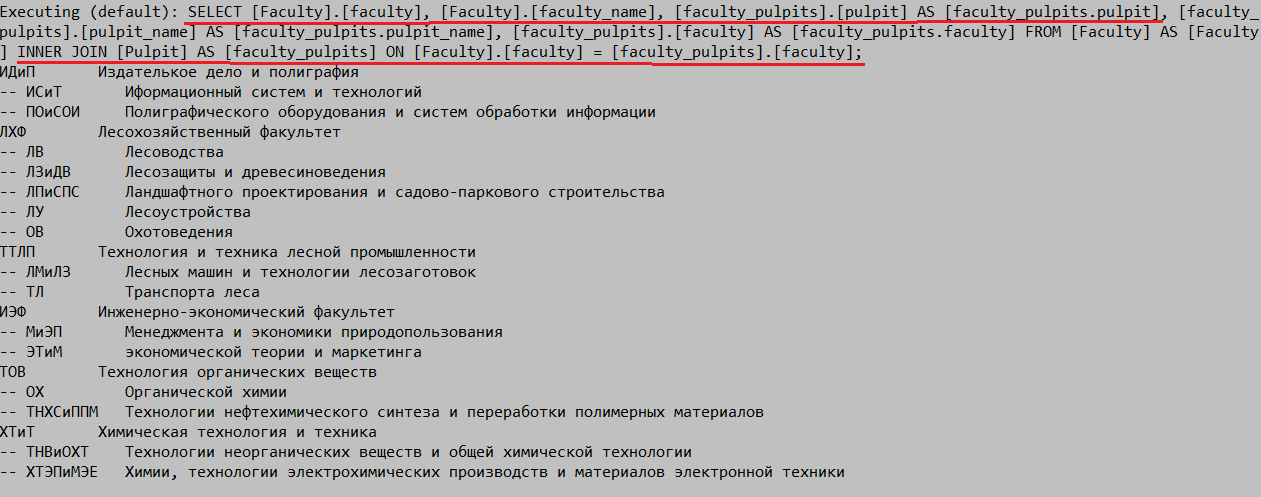
1. **Sequelize:** select … group by



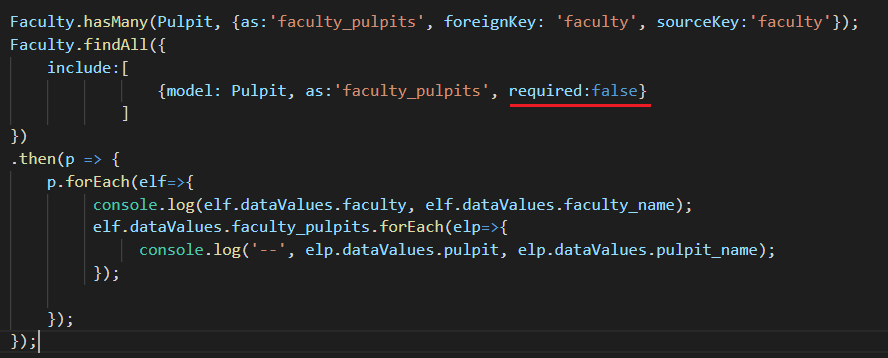


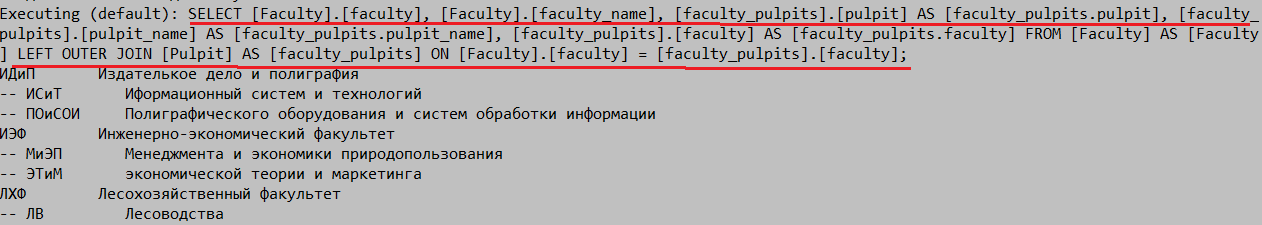
1. **Sequelize:** select … inner join

****

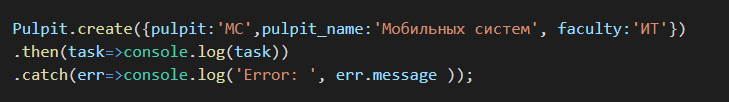
****

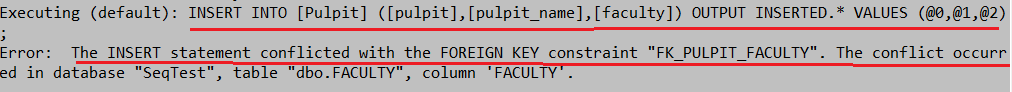
1. **Sequelize:** select … left outer join

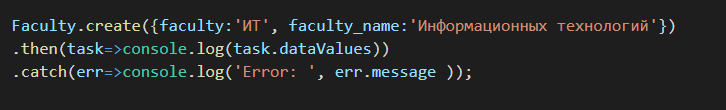




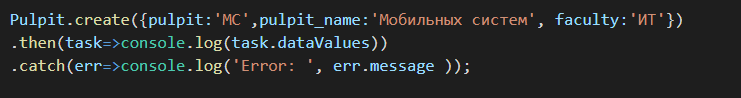
1. **Sequelize:** insert…

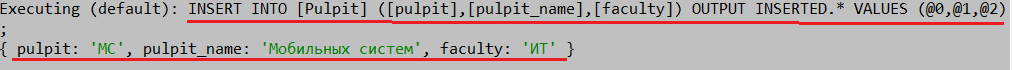
****

****

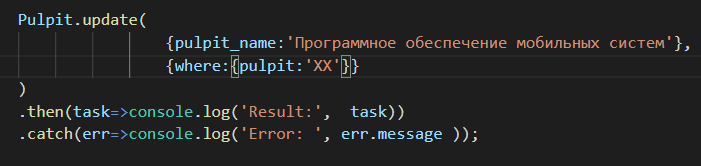
****

****

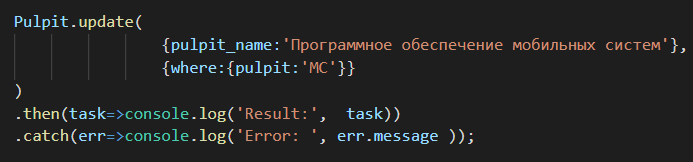
****

****

1. **Sequelize:** update…

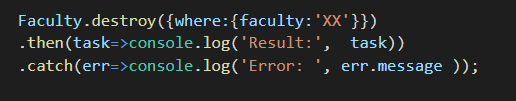
****

****

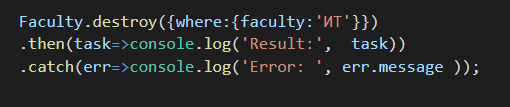
****

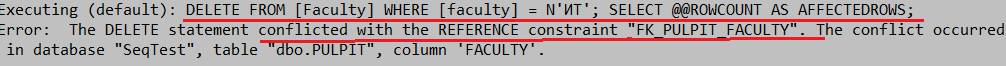
****

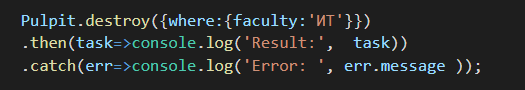
1. **Sequelize:** delete…

****

****

****

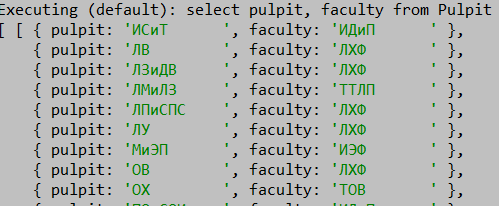
****

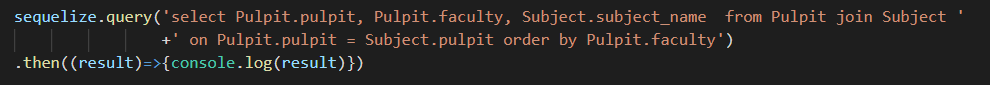
****

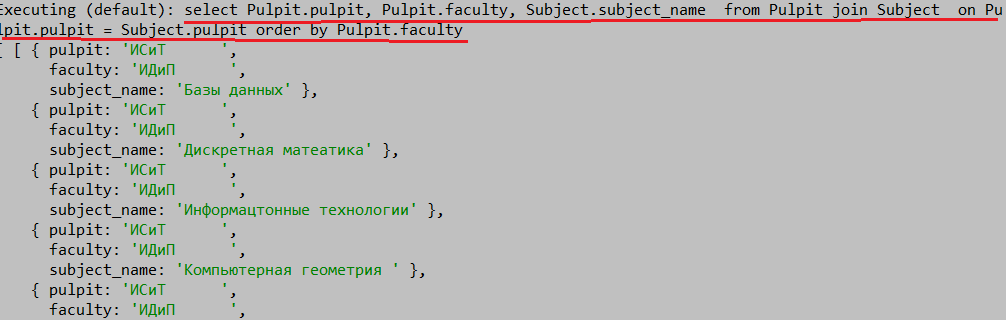
****

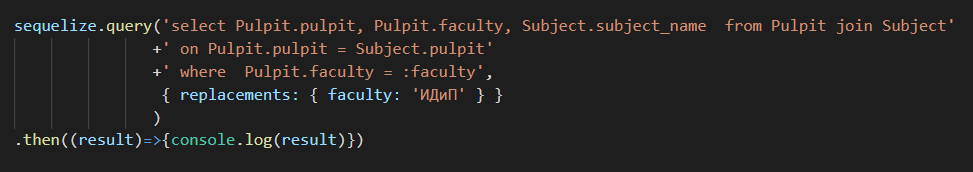
1. **Sequelize:** raw query…

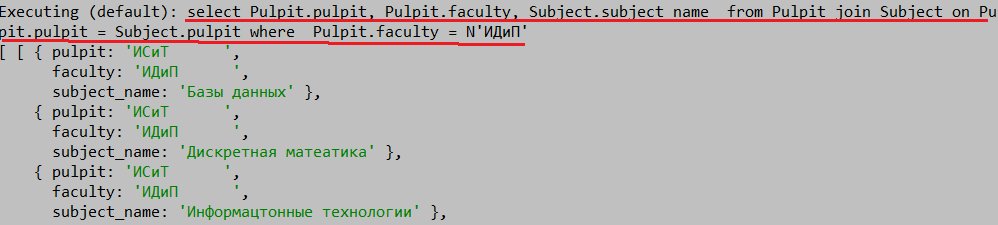
****

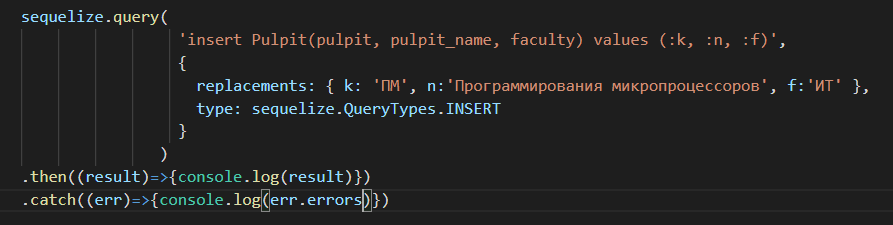
****

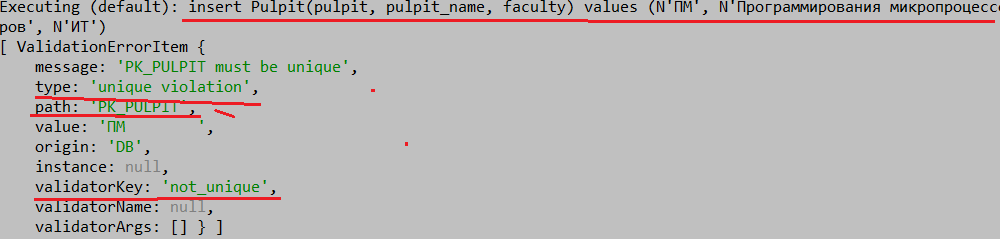
****

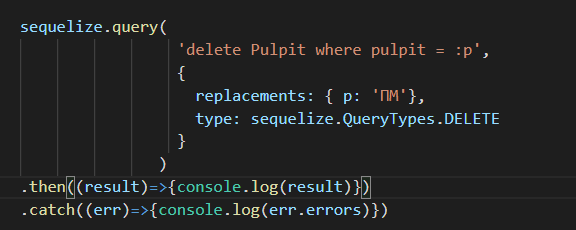
****

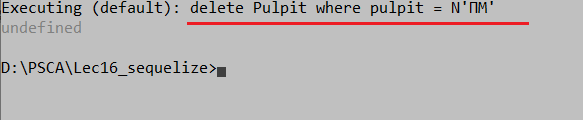
****

****

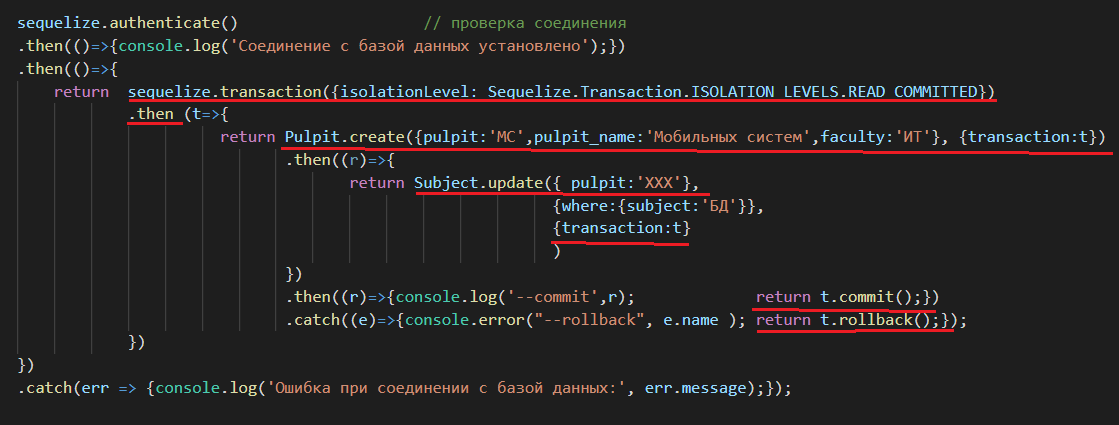
****

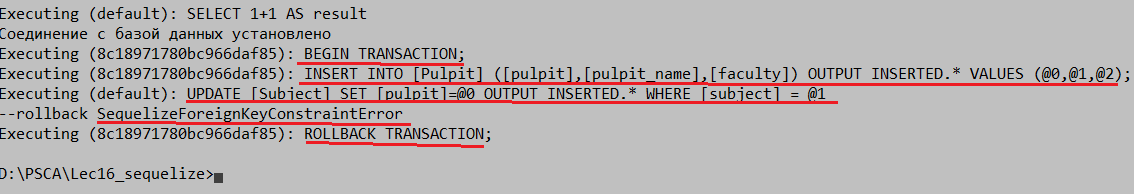
****

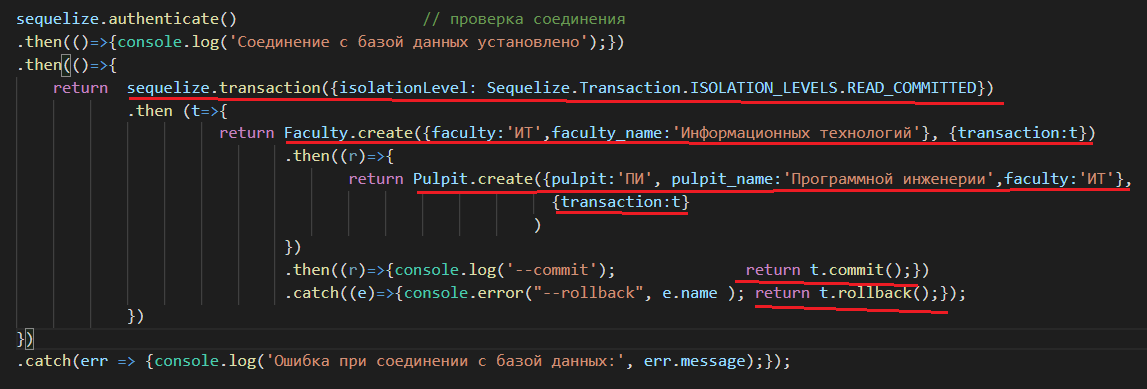
****

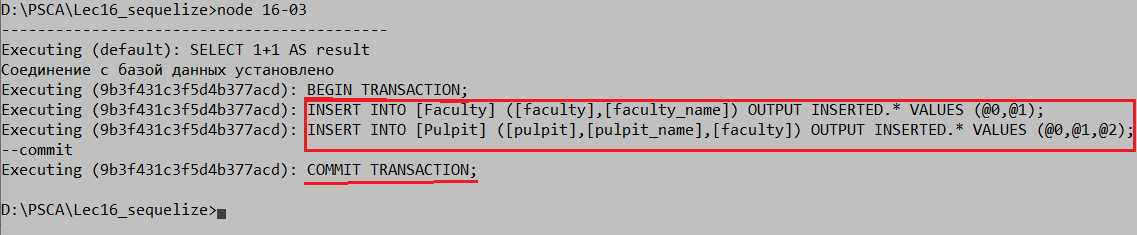
****

1. **Sequelize:** transaction… (unmanaged)

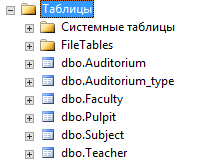
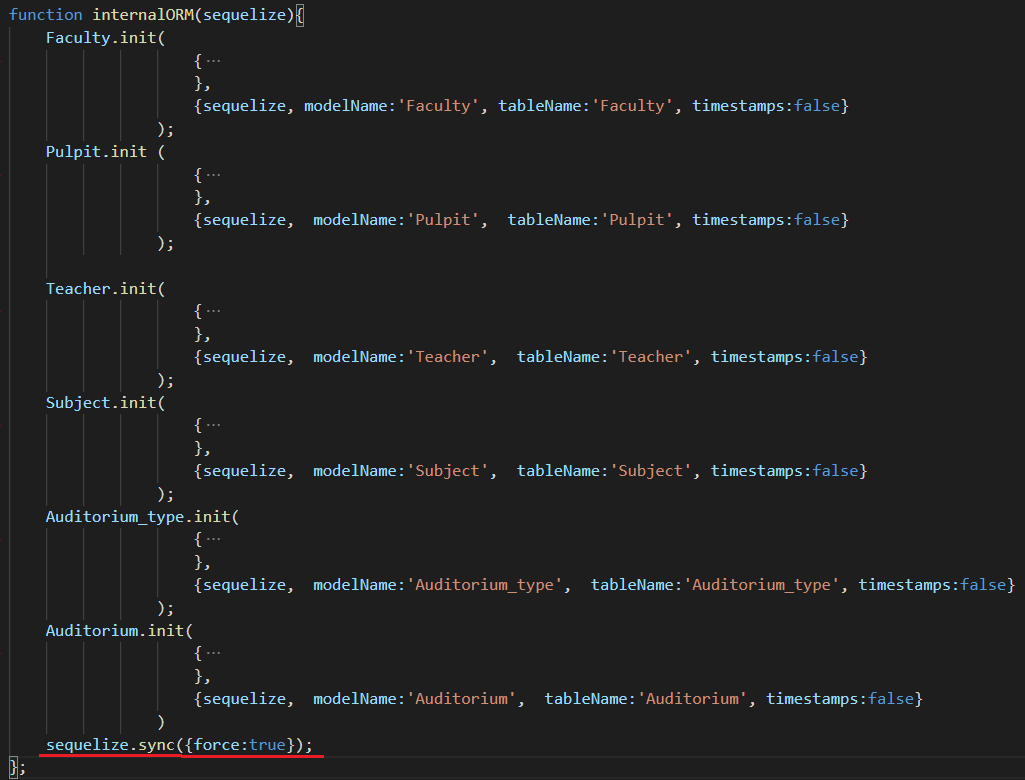
****

****

****

****

1. **Sequelize:** code first

****

1. **Sequelize:** having
2. **Sequelize:**

**Sequelize.STRING // VARCHAR(255)**

**Sequelize.STRING(1234) // VARCHAR(1234)**

**Sequelize.STRING.BINARY // VARCHAR BINARY**

**Sequelize.TEXT // TEXT**

**Sequelize.TEXT('tiny') // TINYTEXT**

**Sequelize.CITEXT // CITEXT PostgreSQL and SQLite only.**

**Sequelize.INTEGER // INTEGER**

**Sequelize.BIGINT // BIGINT**

**Sequelize.BIGINT(11) // BIGINT(11)**

**Sequelize.FLOAT // FLOAT**

**Sequelize.FLOAT(11) // FLOAT(11)**

**Sequelize.FLOAT(11, 10) // FLOAT(11,10)**

**Sequelize.REAL // REAL PostgreSQL only.**

**Sequelize.REAL(11) // REAL(11) PostgreSQL only.**

**Sequelize.REAL(11, 12) // REAL(11,12) PostgreSQL only.**

**Sequelize.DOUBLE // DOUBLE**

**Sequelize.DOUBLE(11) // DOUBLE(11)**

**Sequelize.DOUBLE(11, 10) // DOUBLE(11,10)**

**Sequelize.DECIMAL // DECIMAL**

**Sequelize.DECIMAL(10, 2) // DECIMAL(10,2)**

**Sequelize.DATE // DATETIME for mysql / sqlite, TIMESTAMP WITH TIME ZONE for postgres**

**Sequelize.DATE(6) // DATETIME(6) for mysql 5.6.4+. Fractional seconds support with up to 6 digits of precision**

**Sequelize.DATEONLY // DATE without time.**

**Sequelize.BOOLEAN // TINYINT(1)**

**Sequelize.ENUM('value 1', 'value 2') // An ENUM with allowed values 'value 1' and 'value 2'**

**Sequelize.ARRAY(Sequelize.TEXT) // Defines an array. PostgreSQL only.**

**Sequelize.ARRAY(Sequelize.ENUM) // Defines an array of ENUM. PostgreSQL only.**

**Sequelize.JSON // JSON column. PostgreSQL, SQLite and MySQL only.**

**Sequelize.JSONB // JSONB column. PostgreSQL only.**

**Sequelize.BLOB // BLOB (bytea for PostgreSQL)**

**Sequelize.BLOB('tiny') // TINYBLOB (bytea for PostgreSQL. Other options are medium and long)**

**Sequelize.UUID // UUID datatype for PostgreSQL and SQLite, CHAR(36) BINARY for MySQL (use defaultValue: Sequelize.UUIDV1 or Sequelize.UUIDV4 to make sequelize generate the ids automatically)**

**Sequelize.CIDR // CIDR datatype for PostgreSQL**

**Sequelize.INET // INET datatype for PostgreSQL**

**Sequelize.MACADDR // MACADDR datatype for PostgreSQL**

**Sequelize.RANGE(Sequelize.INTEGER) // Defines int4range range. PostgreSQL only.**

**Sequelize.RANGE(Sequelize.BIGINT) // Defined int8range range. PostgreSQL only.**

**Sequelize.RANGE(Sequelize.DATE) // Defines tstzrange range. PostgreSQL only.**

**Sequelize.RANGE(Sequelize.DATEONLY) // Defines daterange range. PostgreSQL only.**

**Sequelize.RANGE(Sequelize.DECIMAL) // Defines numrange range. PostgreSQL only.**

**Sequelize.ARRAY(Sequelize.RANGE(Sequelize.DATE)) // Defines array of tstzrange ranges. PostgreSQL only.**

**Sequelize.GEOMETRY // Spatial column. PostgreSQL (with PostGIS) or MySQL only.**

**Sequelize.GEOMETRY('POINT') // Spatial column with geometry type. PostgreSQL (with PostGIS) or MySQL only.**

**Sequelize.GEOMETRY('POINT', 4326) // Spatial column with geometry type and SRID. PostgreSQL (with PostGIS) or MySQL only.**