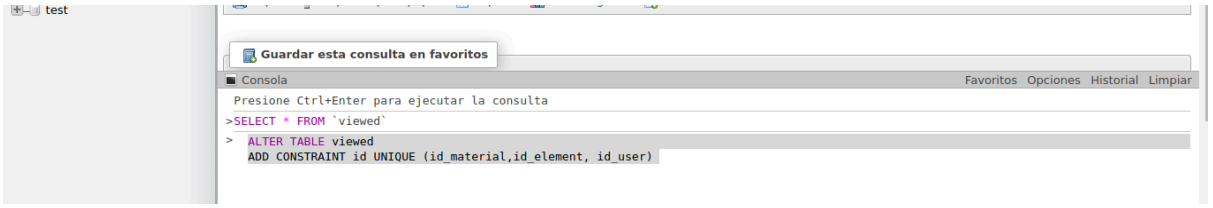
Reporte por fecha con JDBC

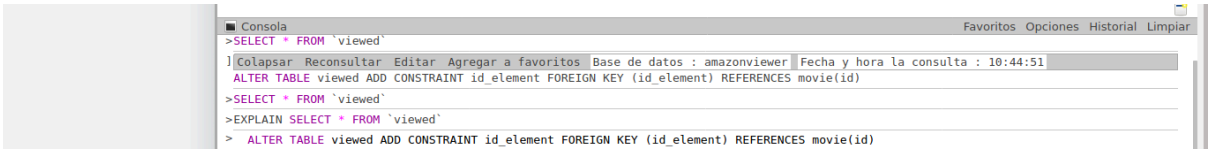
Lo primero que debemos hacer es manejar el posible error de ingresar 2 o más elementos en la tabla viewed que sean exactamente iguales.

Para esto lo que debemos hacer es hacer unicos los id de la tabla viewed de la siguiente forma:
(Apoyate con este tutorial)



ALTER TABLE viewed
ADD CONSTRAINT id UNIQUE (id_material,id_element, id_user)

Ahora, lo siguiente sería hacer una relación entre el id de la tabla movie y el id_element de la tabla viewed para ello ejecutamos el siguiente comando en consola:



ALTER TABLE viewed ADD CONSTRAINT id_element FOREIGN KEY (id_element) REFERENCES movie(id)

Por último solo debemos de agregar la columna de fecha en la tabla viewed como tipo de dato DATE, para ello ejecutamos la siguiente consulta en consola:

ALTER TABLE `viewed` ADD `fecha_visto` DATE NOT NULL DEFAULT CURRENT_TIMESTAMP AFTER id_user;

Y con esto nuestra base de datos ya está lista.

En la clase MovieDAO.java debemos de modificar la inserción de de datos que se hizo previamente en el curso, para ello simplemente debemos agregar un dato Date a la insercion:
(NOTA: en mi caso convertí la consulta a un prepared statement para hacerlo más práctico)

```
default Movie setMovieViewed(Movie movie){
    try(Connection connection = connectToDB()){
        String query = "INSERT INTO " + TVIEWED +
                " (" +TVIEWED_ID_MATERIAL+", "+TVIEWED_ID_ELEMENT_+", "+TVIEWED_ID_USER+", "+ TVIEWED_FECHA_VISTO+")" +
                " VALUES (?, ?, ?, ?)";
        System.out.println(query);
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setInt( i: 1, getIDMaterial(TMOVIE));
        statement.setInt( i: 2, movie.getId());
        statement.setInt( i: 3, ID_ANN);
        statement.setDate( i: 4, new java.sql.Date(new java.util.Date().getTime()));

        statement.executeUpdate();

    }catch (SQLException e){
        e.printStackTrace();
    }
    return movie;
}
```

Lo siguiente sería hacer un método para seleccionar las películas con la fecha que el usuario desee:

```
default ArrayList<Movie> getMoviesForReport(Date date){
    String query = "SELECT  TM.* FROM "+ TVIEWED +" AS TV "+ " LEFT JOIN "+ TMOVIE +" AS TM "+
            " ON TV."+ TVIEWED_ID_ELEMENT_+" = TM."+TMOVIE_ID+
            " WHERE TV."+TVIEWED_FECHA_VISTO+" = ?" +
            " AND TV."+TVIEWED_ID_USER+" = ?" +
            " AND TV."+TVIEWED_ID_MATERIAL+" = ?";

    ResultSet set = null;
    ArrayList<Movie> movies = new ArrayList<>();
    try (Connection connection = connectToDB()){
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.setDate( i: 1, date);
        preparedStatement.setInt( i: 2, ID_ANN);
        preparedStatement.setInt( i: 3, getIDMaterial(TMOVIE));

        set = preparedStatement.executeQuery();

        while (set.next()){
            Movie movie = new Movie(
                    set.getString(TMOVIE_TITLE),
                    set.getString(TMOVIE_GENRE),
                    set.getString(TMOVIE_CREATOR),
                    Integer.parseInt(set.getString(TMOVIE_DURATION)),
                    Short.parseShort(set.getString(TMOVIE_YEAR))
            );
            movies.add(movie);
        }
```

```
                movies.add(movie);
            }
        }catch (SQLException e){
            e.printStackTrace();
            return null;
        }
        if (movies.isEmpty())
            return null;

        return movies;
    }
```

Luego de esto debemos hacer un método de la clase Movie que nos sirva para conectarnos a la interfaz:

```
public static ArrayList<Movie> getMoviesWithDate(java.sql.Date date){
    Movie movie = new Movie();
    return movie.getMoviesForReport(date);
}
```

Lo siguiente es modificar el menú principal, ¿Por qué? porque si bien podemos hacer reportes del día de hoy, también podemos hacer un reporte de una fecha en específico, por lo que en total son 3 maneras de hacer reportes:

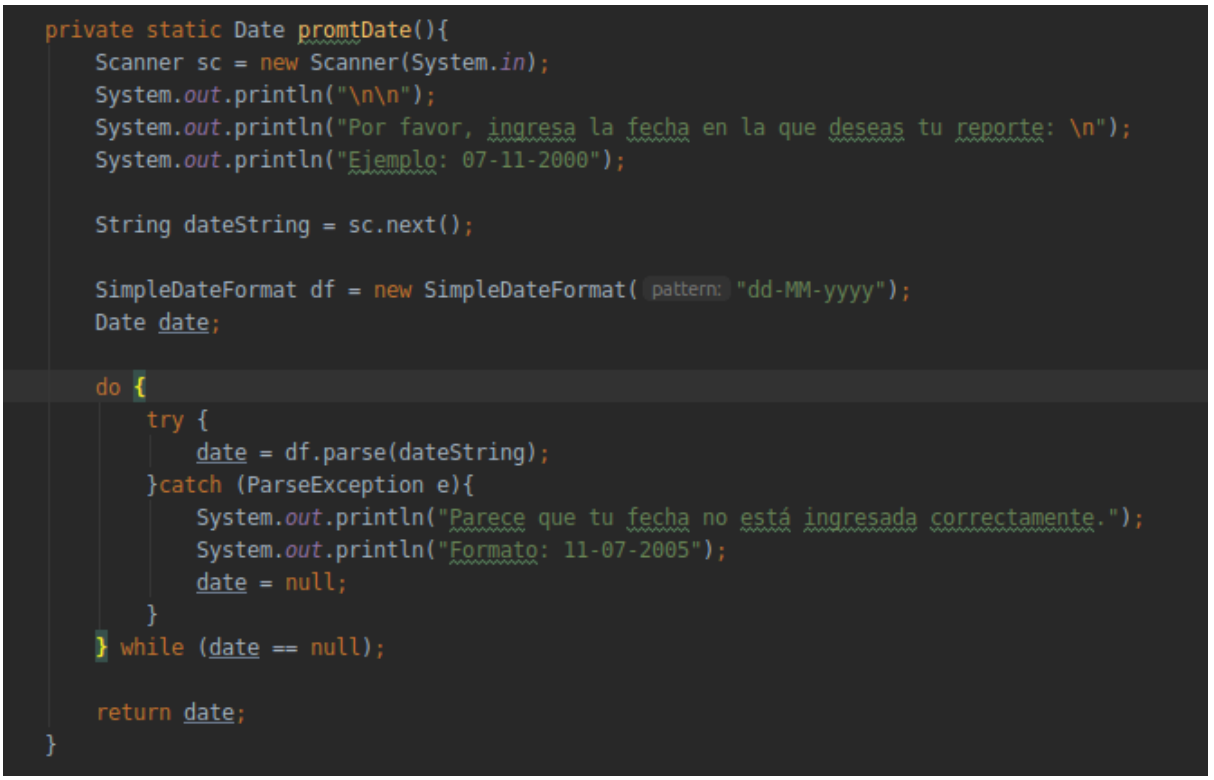
1. General (Todo lo que hemos visto)
2. Del día de hoy
3. De una fecha en específico

Por lo tanto nuestro menú queda así:

```
                break;
            case 3:
                showBooks();
                break;
            case 4:
                showMagazines();
                break;
            case 5:
                makeReportAll();
                exit = 1;
                break;
            case 6:
                makeReportToday(new Date());
                exit = 1;
                break;
            case 7:
                makeReport();
                break;
```

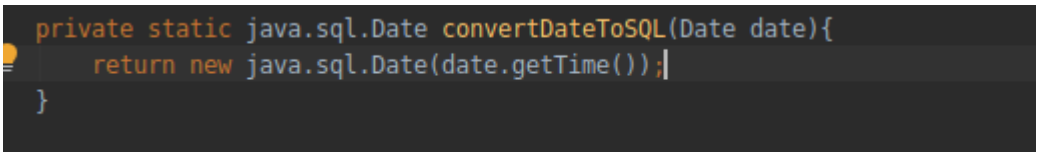
Antes de pasar de lleno al metodo makeReport(); debemos hacer 2 cosas:

1. Preguntar y parsear la fecha que el usuario desea:



```
private static Date promtDate(){
    Scanner sc = new Scanner(System.in);
    System.out.println("\n\n");
    System.out.println("Por favor, ingresa la fecha en la que deseas tu reporte: \n");
    System.out.println("Ejemplo: 07-11-2000");

    String dateString = sc.next();

    SimpleDateFormat df = new SimpleDateFormat( pattern: "dd-MM-yyyy");
    Date date;

    do {
        try {
            date = df.parse(dateString);
        }catch (ParseException e){
            System.out.println("Parece que tu fecha no está ingresada correctamente.");
            System.out.println("Formato: 11-07-2005");
            date = null;
        }
    } while (date == null);

    return date;
}
```

2. Convertir el objeto *java.util.date* a *java.sql.date*



```
private static java.sql.Date convertDateToSQL(Date date){
    return new java.sql.Date(date.getTime());
}
```

Por último realizamos el método de makeReport(); tal como lo hemos hecho en clases del curso, pero claro, utilizando los métodos que acabamos de crear:

```java
public static void makeReport() {
    ArrayList<Movie> movies = new ArrayList<>();
    SimpleDateFormat dfNameDays = new SimpleDateFormat( pattern: "E, D MMM Y");

    Date date = promtDate();
    movies = Movie.getMoviesWithDate(convertDateToSQL(date));

    if (movies == null) {
        System.out.println("Parece que no hay peliculas vistas ese día.");
        return;
    }

    Report report = new Report();
    report.setNameFile("Reporte de: " + dfNameDays.format(date));
    report.setExtention("txt");
    report.setTitle("======================\n\n :::VISTOS::: ====================\n\n");


    String dateString = dfNameDays.format(date);
    String contentReport = "Reporte de la fecha: " + dateString + "\n\n\n";

    for (Movie myMovie: movies){
        contentReport += myMovie.toString() + "\n\n";
    }

    report.setContent(contentReport);
```

```java
        contentReport += myMovie.toString() + "\n\n";
    }

    report.setContent(contentReport);
    report.makeReport();

    System.out.println("\n\n");
    System.out.println("========================================");
    System.out.println("========================================");
    System.out.println("            Reporte Generado        ");
    System.out.println("========================================");
    System.out.println("========================================");
    System.out.println("\n\n");
}
```

Ahora, solo falta un detalle que quizá puede pasar desapercibido, y es que si vemos una película por segunda vez estaremos haciendo una consulta que duplicaría datos en la base de datos (lo cual no se puede por lo que hicimos al inicio del tutorial), por lo que hacemos una condicional simple en el metodo view(); en movie:

```java
//La insercion solo se hace si el usuario no ha visto la pelicula
if (!this.getIsViewed()){
    Movie movie = new Movie();
    movie.setMovieViewed(this);
}
```

De esta manera ya no generamos una inserción innecesaria si el usuario ya vió la película.