```java
import javax.persistence.AttributeConverter;
    import javax.persistence.Converter;
    import java.sql.Date;
    import java.time.LocalDate;

@Converter(autoApply = true)
public class LocalDateConverter implements AttributeConverter<LocalDate, Date> {

  @Override
  public Date convertToDatabaseColumn(LocalDate entityDate) {
    return entityDate == null ? null : Date.valueOf(entityDate);
  }

  @Override
  public LocalDate convertToEntityAttribute(Date databaseDate) {
    return databaseDate == null ? null : databaseDate.toLocalDate();
  }
}

@Stateless
public class AlbumFacade {

  @PersistenceContext
  private EntityManager em;

  public void create(Album entity) {
    em.persist(entity);
  }

  public void deleteById(Long id) {
    Album entity = em.find(Album.class, id);
    if (entity != null) {
      em.remove(entity);
    }
  }

  public Album update(Album entity) {
    return em.merge(entity);
  }

  public List<Album> findAll() {
    TypedQuery<Album> query = em.createNamedQuery("Album.findAll", Album.class);
    return query.getResultList();
  }

  public Album findById(Long id) {
    return em.find(Album.class, id);
  }

  public List<Album> listAll(Integer startPosition, Integer maxResult) {
    TypedQuery<Album> findAllQuery = em.createQuery(
        "SELECT DISTINCT a FROM Album a ORDER BY a.id", Album.class);
    if (startPosition != null) {
            findAllQuery.setFirstResult(startPosition);
        }
        if (maxResult != null) {
            findAllQuery.setMaxResults(maxResult);
        }
        return findAllQuery.getResultList();
    }

    public void readJsonContent(String json) {

        JsonStructure structure;

        try (StringReader stringReader = new StringReader(json)) {
            structure = Json.createReader(stringReader).read();
        } catch (Exception e) {
            System.err.println("Json FileFormat not correct: " + e.getMessage());
            throw new JsonException("Json FileFormat not correct");
            //return false;
        }

        switch (structure.getValueType()) {
            case ARRAY:
                JsonArray jsonArray = (JsonArray) structure;

                for (JsonValue jsonValue : jsonArray) {
                    saveJsonObject((JsonObject) jsonValue);
                }
                break;
            case OBJECT:
                saveJsonObject((JsonObject) structure);
                break;
            default:
                throw new JsonException("Unexpected Error while parsing Json");
                //return false;
        }
        //return true;
    }

    private void saveJsonObject(JsonObject jsonAlbum) {
        Album album = new Album();

        LocalDate released;
        album.setTitle(jsonAlbum.getString("Title"));

        try {
            released = LocalDate.parse(jsonAlbum.getString("Released"),
DateTimeFormatter.ofPattern("dd MMMM yyyy"));
        } catch (Exception e) {
            released = LocalDate.parse(jsonAlbum.getString("Released"),
DateTimeFormatter.ofPattern("d MMMM yyyy"));
        }
        album.setReleased(released);
```

```java
        album.setLabel(jsonAlbum.getString("Label"));

        try {
            Integer ukChartPos = Integer.valueOf(jsonAlbum.getString("UK Chart Position"));
            System.out.println("uk " + ukChartPos);
            album.setUkChartPosition(ukChartPos);
        } catch (Exception e) {
            System.err.println("'" + jsonAlbum.getString("UK Chart Position") + "' --> " + e.getMessage());
        }

        try {
            Integer usChartPos = Integer.valueOf(jsonAlbum.getString("US Chart Position"));
            System.out.println("us " + usChartPos);
            album.setUsChartPosition(usChartPos);
        } catch (Exception e) {
            System.err.println("'" + jsonAlbum.getString("US Chart Position") + "' --> " + e.getMessage());
        }

        album.setBpiCertification(jsonAlbum.getString("BPI Certification"));
        album.setRiaaCertification(jsonAlbum.getString("RIAA Certification"));

        create(album);
        System.out.println(album);
    }

    public int countRiaaCertifications(String certification) {

        int countCert = 0;

        List<String> certifications = em
                .createQuery("select a.riaaCertification from Album a where a.riaaCertification like
:CERT",String.class)
                .setParameter("CERT", "%" + certification)
                .getResultList();

        for (String cert : certifications) {
            String[] elements = cert.split("x");
            if (elements.length == 1) {
                countCert++;
            } else {
                countCert += Integer.parseInt(elements[0]);
            }
        }

        return countCert;
    }
}


@Named
@SessionScoped
public class IndexController implements Serializable {
    @Inject
    private AlbumFacade albumFacade;

    private String uploadedText;

    List<Album> albums;

    public IndexController() {
    }

    @PostConstruct
    private void init() {
        loadAlbums();
    }

    public void fileUploadHandler(FileUploadEvent event) {
        uploadedText = new String(event.getFile().getContents());
        try {
            albumFacade.readJsonContent(uploadedText);
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO, "Successful",
event.getFile().getFileName() + " is uploaded.");
            FacesContext.getCurrentInstance().addMessage(null, message);
        } catch (Exception e) {
            System.err.println("*** parsing to json faild");
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Import
failed", event.getFile().getFileName() + " is not uploaded.");
            FacesContext.getCurrentInstance().addMessage(null, message);
        }
        loadAlbums();

        // Update der dataTable
        RequestContext.getCurrentInstance().update("albumTable");
        RequestContext.getCurrentInstance().update("certificationPanel");
    }

    public int getSilverCertifications() {
        return albumFacade.countRiaaCertifications("Silver");
    }

    public int getGoldCertifications() {
        return albumFacade.countRiaaCertifications("Gold");
    }

    public int getPlatinumCertifications() {
        return albumFacade.countRiaaCertifications("Platinum");
    }

    public List<Album> getAlbums() {
        return albums;
    }

    public void setAlbums(List<Album> albums) {
        this.albums = albums;
    }

    public String getUploadedText() {
        return uploadedText;
    }

    public void setUploadedText(String uploadedText) {
        this.uploadedText = uploadedText;
    }

    private void loadAlbums() {
        albums = albumFacade.findAll();
    }

}
```