السؤال الأول: شو الفرق بين hasOne و belongsTo بـLaravel ؟

شوف الـ hasOneيعني شغلة مرتبطة بشغلة وحدة بس
متل مثلاً إذا عندك "مستخدم" وعندك "بروفايل" خاص فيه
المستخدم hasOneبروفايل يعني كل مستخدم إلو بروفايل واحد بس
فبتكتبها بملف موديل المستخدم هيك

```
public function profile()
{
return $this->hasOne(Profile::class);
}
```

أما الـ belongsToيعني شغلة بتنتمي لشغلة تانية
مثلاً نفس المثال تبع "البروفايل" و"المستخدم"
البروفايل belongsToمستخدم يعني كل بروفايل بينتمي لمستخدم واحد
فبتكتبها بملف موديل البروفايل هيك

```
public function user()
{
return $this->belongsTo(User::class);
}
```

فالخلاصة hasOneمن جهة الأب و belongsToمن جهة الابن اللي إلو أب

السؤال الثاني: ليش ممكن تستخدم الـ Queues بـLaravel ؟

الـ Queues بتستخدمها لما يكون عندك شي بدو وقت طويل لينفذ وما بدك المستخدم يستنى ليخلص
يعني بتخلي المستخدم يكمل شغله والعملية الصعبة بتصير بالخلفية بدون ما يشوفها أو يحس فيها

مثلا تخيل عندك موقع كبير والناس بتحمل صور كتير أو ملفات فيديو
ولما يرفعوا الصورة بدك تعمل عليها معالجة مثلاً تصغيرها أو تحويلها لأكتر من حجم
هي العملية ممكن تاخد وقت
فبدل ما المستخدم يرفع الصورة ويقعد يستنى خمس دقايق ليعالجها الموقع
بتحط هي مهمة المعالجة بالـQueue
المستخدم بيرفع الصورة وبتروح للموقع رسالة "تم رفع الصورة بنجاح" وهو بيقدر يكمل تصفح عادي
والـ Queue بياخد الصورة بيعالجها بالخلفية بس تخلص بيبعت إشعار للمستخدم مثلاً

هيك بتخلي الموقع سريع وما بتخلي المستخدم يمل أو يزهق من الانتظار

السؤال الثالث: إذا تطبيق الواجهة الأمامية بطيء عدلي 3 خطوات لتحسين الأداء

إذا واجهة التطبيق بطيئة ممكن تعمل كذا شغلة لتسرعها

1. **ضغط الصور والملفات** :**(Optimize Assets)** يعني كل الصور اللي بالموقع بدك تصغر حجمها قدر الإمكان بدون ما تخرب جودتها وفي برامج بتعمل هالشي. كمان ملفات الـ CSS والـ JavaScript فيك تضغطها أو تشيل منها المسافات الزايدة والأشياء اللي مالها لزوم لتصير أصغر وأسرع بالتحميل.

2. **تخفيف طلبات الـ API (Reduce API Calls)** :يعني إذا الواجهة عم تبعت طلبات كتير للسيرفر لتجيب بيانات ممكن تحاول تقلل هي الطلبات. مثلاً بدل ما تبعت 5 طلبات منفصلة لتجيب معلومات عن شي معين فيك تعمل طلب واحد يجيبلك كل المعلومات اللي بدك ياها. أو ممكن تعمل "تخزين مؤقت (Caching) "لبعض البيانات اللي ما بتتغير كتير مشان ما تطلبها من السيرفر كل مرة.

3. **تأجيل تحميل بعض المكونات** :**(Lazy Loading)** يعني لما بتفتح الصفحة ما بتخلي كل شي يحمل فورا. بتخلي بس الأشياء اللي ظاهرة قدام المستخدم تحمل بالبداية. أما الأشياء اللي موجودة بأسفل الصفحة أو اللي ما بيشوفها المستخدم إلا إذا نزل لتحت بتخليها تحمل بس يوصل لعندها. هيك الصفحة بتحمل أسرع وبتبين أخف على المستخدم.

السؤال الرابع: كيف ممكن تتعامل مع ترقيم الصفحات بكفاءة لما يكون عندك ملايين الصفوف؟

لما يكون عندك ملايين الصفوف بجدول قاعدة البيانات والصفحات كتير بدك تنتبه مشان ما يصير بطء

1. **استخدام الـ OFFSET والـ LIMIT بحذر** :**(Efficient Pagination with LIMIT/OFFSET)** عادةً الترقيم بيعتمد على LIMIT(كم صف بدك) و OFFSET(من وين تبدأ العد). بس لما تصير الأوفسيت رقم كبير جداً (مثلاً بدك الصفحة رقم 100000) بتصير قاعدة البيانات بطيئة كتير لأنها بدها تعد كل الصفوف اللي قبل هاد الرقم.

الحل الأفضل هو إنك تعتمد على **آخر معرف (ID) تمت مشاهدته** بالصفحة السابقة.
يعني بدل ما تقول "أعطيني 20 صف بدءاً من الصف رقم مليون"
بتقول "أعطيني 20 صف اللي الـ ID تبعهم أكبر من آخر ID شفته بالصفحة اللي قبلها"

مثلاً هيك بتصير استعلام الـ:SQL

```
SELECT * FROM your_table

WHERE id > [last_id_from_previous_page]

ORDER BY id ASC

LIMIT 20;
```

هي الطريقة أسرع بكتير مع الجداول الكبيرة لأنها بتستخدم الـ index تبع الـ ID بشكل فعال.

2. **التخزين المؤقت للعد الإجمالي** :**(Cache Total Count)** إذا بدك تعرض للمستخدم عدد الصفحات الكلي أو عدد العناصر الكلي، ممكن هالشي يكون بطيء جداً لما يكون عندك ملايين الصفوف (الاستعلام .((*)COUNTفممكن تخزن عدد الصفوف الكلي بجدول تاني صغير أو بـ cache وتحدثه كل فترة أو لما يصير تعديل كبير.

3. **فهرسة الأعمدة المستخدمة للفرز** :**(Index Columns Used for Ordering)** أي عمود بتستخدمه للترتيب (ORDER BY) لازم يكون عليه فهرس (Index) بقاعدة البيانات. الفهارس بتخلي البحث والترتيب أسرع بشكل كبير.

هيك بتصير الأمور أسرع وأكثر كفاءة حتى لو كان عندك بيانات كتير كتير.

أما بالنسبة لسؤال الفهارس:

شو هي الفهارسIndexes ؟

تخيل عندك كتاب كتير كبير وبدك تلاقي معلومة معينة فيه
إذا ما في فهرس بدك تقلب كل الصفحات صفحة صفحة لتعثر عليها وهاد الشي بياخد وقت كتير

الفهرس بقاعدة البيانات متل فهرس الكتاب بالزبط
بدل ما قاعدة البيانات تفتش بكل سطر بالجدول لتعثر على المعلومة اللي بدك ياها
الفهرس بيساعدها تروح مباشرة عالسطر الصح بسرعة
وهالشي بيخلي البحث واستخراج البيانات أسرع بكتير خصوصاً لما تكون البيانات كتيرة

ليش بدنا الفهارس؟

إذا ما في فهارس قاعدة البيانات بدها تعمل "مسح كامل للجدول" كل مرة بتدور على شي
يعني بدها تمر على كل سطر بالجدول وهاد الشي بصير بطيء جداً كل ما كبر الجدول وزادت البيانات فيه

كيف بدنا نحط فهارس لمشروعنا الحالي المستخدمين والمنتجات والطلبات؟

1. **المفاتيح الأساسيةPrimary Keys**

   متل عمود الـ idبكل الجداول (المستخدمين والمنتجات والطلبات)

   الطريقة Laravel :وقواعد البيانات متل MySQL بيعملوا فهرس لعمود الـ idلحالهن بشكل تلقائي
   وهاد الشي ضروري كتير مشان نلاقي السجلات بسرعة ومشان العلاقات بين الجداول تكون سريعة

2. **المفاتيح الخارجيةForeign Keys**

   متل عمود الـ user_idبجدول الطلبات اللي بيربط كل طلب بالمستخدم اللي عامله

   الطريقة :لازم دايماً نعمل فهرس للمفاتيح الخارجية
   لما تكتب بـLaravel
   $table->foreignId('user_id')->constrained()->onDelete('cascade');
   هاد السطر بيعمل فهرس للمفتاح الخارجي لحاله كمان
   وهاد الشي كتير مهم لتسريع عملية الربط بين الجداول(JOIN)

3. **الأعمدة اللي بنستخدمها بـ WHERE شروط البحث**

   متل عمود الـ created_atبجدول الطلبات إذا بدنا نطلع الطلبات تبع شهر معين

   أو عمود الـ statusبجدول الطلبات إذا بدنا نشوف الطلبات اللي لسا ما خلصت

   أو عمود الـ nameبجدول المنتجات إذا بدنا نفتش عن منتج بالاسم

   أو عمود الـ emailبجدول المستخدمين وهاد موجود عليه فهرس فريد لحاله

   الطريقة :أي عمود بتستخدمه كتير بـ WHEREمشان تصفي البيانات لازم تحطله فهرس
   مثلاً هيك بتكتب بملف الهجرة تبع الطلبات:

```
$table->index('created_at');

$table->index('status');
```

وبملف الهجرة تبع المنتجات:

```
$table->index('name');
```

إذا كان عندك عمود فيه نصوص كتيرة وطويلة text) أو (longText الفهرس عليه كله ما بيكون فعال كتير ممكن تحتاج تفهرس جزء منه أو تستخدم شي اسمو فهارس نصية كاملة(Full-Text Indexes)

و الأعمدة اللي فيها قيم قليلة ومو متكررة كتير متل عمود is_active(يا 0 يا 1) الفهرس عليها ما بيفيد كتير

الأعمدة اللي بنستخدمها بـ ORDER BY للترتيب

إذا بدك ترتب الطلبات حسب مبلغها الكلي total_amount

الطريقة :إذا بتعمل ترتيب للنتائج كتير بناءً على عمود معين حطله فهرس مشان عملية الترتيب تصير أسرع
بملف الهجرة تبع الطلبات:

```
$table->index('total_amount');
```

الأعمدة اللي بنستخدمها بـ GROUP BY للتجميع

متل عمود الـ created_atلما تجمع الإيرادات حسب الشهر

الطريقة :الفهرس على الأعمدة اللي بتستخدمها بـ GROUP BYبيخلي عملية التجميع أسرع

أنواع الفهارس:

فهرس عادي :**Normal Index** بيسرع البحث والترتيب

فهرس فريد :**Unique Index** بيضمن إنو كل القيم بالعمود هاد فريدة وما بتتكرر متل الإيميل بجدول المستخدمين

فهرس مركب :**Composite/Compound Index** هاد فهرس على عمودين أو أكتر مع بعض
مفيد كتير إذا كنت بتستخدم هالأعمدة مع بعض بشروط البحث WHEREأو الترتيب ORDER BY
مثلاً فهرس على user_idو created_atبجدول الطلبات بيكون مفيد إذا بدك تلاقي طلبات لمستخدم معين بتاريخ معين
مثال:

```
$table->index(['user_id', 'created_at']);
```

ملاحظة كتير مهمة :ترتيب الأعمدة بالفهرس المركب مهم
يعني فهرس على (A, B) بيفيد لما تبحث عن A لحاله أو A و B مع بعض
بس ما بيفيد لما تبحث عن B لحاله

أمثلة على تحسين استعلاماتنا بالفهارس:

لأفضل 3 مستخدمين (حسب الطلبات مثلاً):

تأكد إنو في فهرس على total_amountبجدول الطلبات

يوجد فهرس على user_idبجدول الطلبات لأن هوي مفتاح رئيسي و فهرس افتراضي

الأفضل تعمل فهرس مركب على user_idو total_amountمع بعض:

```
$table->index(['user_id', 'total_amount']);
```

للإيرادات الشهرية: تأكد إنو في فهرس على created_atبجدول الطلبات:

```
$table->index('created_at');
```

مساوئ الفهرسة:

الفهارس مو مجانية :الفهارس بتخلي قاعدة البيانات أكبر شوي
وكمان بتخلي عمليات الإضافة INSERTوالتعديل UPDATEوالحذف DELETEأبطأ
لأنو قاعدة البيانات بدها تحدث الفهارس بعد كل عملية كتابة

لا تبالغ بالفهرسة :حط فهارس بس على الأعمدة اللي بتستخدمها كتير بالبحث والترتيب والتجميع

راقب الأداء :استخدم أدوات مراقبة قاعدة البيانات (متل EXPLAINبـMySQL)
مشان تشوف أداء الاستعلامات تبعك وتحدد إذا الفهرس عم يشتغل صح أو بدك تغير فيه أو تضيف فهارس تانية

Laravel والفهارس :فيك تضيف الفهارس بملفات الهجرة تبعك بـLaravel
باستخدام طرق متل '$table->indexاسم_العمود 'أو '$table->uniqueاسم_العمود'