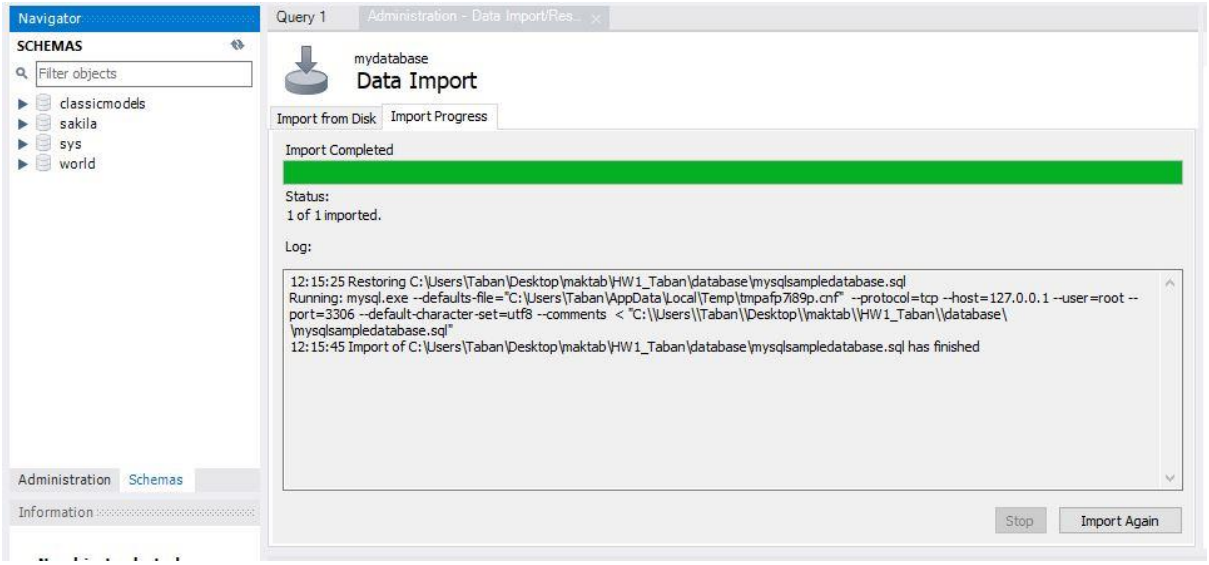
بـخش اول

در این قســـمت، پس از نصـــب MySQL و MySQL Workbench فایل دیتابیس موجود را Import می‌کنیم (مطابق شکل۱) و با ا ستفاده از show Query لی ست دیتابیس‌های موجود را می توان مشاهده کرد.



شکل ۱

برای سوال اول از Query نشان داده شده استفاده کردیم، نتایج مربوطه در فایل Question۱ ۰csv ذخیره شده است.

```
SELECT email
FROM classicmodels.employees
WHERE jobTitle = 'Salees Rep';
```

سپس از Query نشان داده شده برای نمایش لیست نام‌خانوادگی و کدپستی کسانی که در Spain یا USA هستند، استفاده کردیم و نتایج مربوطه در فایل Question۲۰csv ذخیره شده است.

```
SELECT contactLastName, postalCode
FROM classicmodels.customers
WHERE country = 'USA' OR country = 'Spain' ;
```

برای قسمت سوم، Query نمایش داده شده در کادر زیر را استفاده کردیم و نتایج در فایل Question۳۰csv ذخیره و ضمیمه شده است.

```
SELECT phone, customerName
FROM classicmodels.customers
WHERE creditLimit < ۱۰۰۰۰۰;
```

برای قسمت چهارم، با استفاده از INNER JOIN Query نام و قیمت خرید محصولاتی که بیش از ۳۰ عدد سفارش دارند را لیست کرده و نتایج مورد نظر را در فایل Question٤٠csv ذخیره و ضمیمه شده است.

```
SELECT productName, byPrice
FROM classicmodels.products
INNER JOIN orderdetails ON quantityOrdered > 30;
```

برای قسمت پنجم نیز با استفاده از Query زیر، تمام ستون‌های جدول payment که تاریخ آن به قبل از ۲۰۰۵ مربوط می‌شد و مبلغ پرداختی بالای ۱۰ هزار دلار بود را در فایل Question٥٠csv ذخیره کردیم.

```
SELECT * FROM classicmodels.product
WHERE (paymentDate BETWEEN '۰۰۰۰-۰۰-۰۰' AND '۲۰۰۵-۰۰-۰۰') AND amount > ۱۰۰۰۰/۰۰;
```

۵ و ٦- دیتابیس‌های تکنولوژی NoSQL، اطلاعات را به جای ذخیره‌سازی در جداول مرتبط به یکدیگر، آنها را در چندین JSON document یا XML ذخیره می‌کنند . باتوجه به انعطاف‌پذیری این تکنولوژی (به دلیل دارا بودن ساختار پویا برای ذخیره‌ی دیتا) امکان ذخیره‌سازی اطلاعات مانند SQL نیز وجود دارد. این دسته از دیتابیس‌ها به دلیل انعطاف‌پذیری، مقیاس‌پذیری بالا و قابلیت پاسخدهی سریع به درخواست‌های مدیریت دیتا در صنایع مدرن به کار می‌روند. دیتابیس های NoSQL می‌توانند تعداد زیادی از کاربران را پشتیبانی کنند، همچنین می‌توانند با آپدیت‌های متعدد، تغییر نیازها و ویژگی‌های جدید خودشان را سازگار کنند و دیتاهای سامان نیافته یا نیمه سامان یافته را نیز هندل می‌کنند.

از Relational DBMS بسیار قبل تر از ظهور اینترنت، کلان دیتا، پردازش ابری، موبایل و ... استفاده می شد. و در ابتدا این دسته از دیتابیس‌ها برای اجرا بر روی تنها یک سرور ساخته می‌شدند. تنها راه برای افزایش ظرفیت این دیتابیس‌ها ارتقای آنها برای مقیاس پذیری بالا(به منظور پردازش و ذخیره سازی بهینه‌تر , سرعت بالا) بود. از طرفی دیگر، در دیتابیس‌های SQL، ذخیره‌سازی حجم بالایی از دیتای سازمان نیافته منجر به کاهش سرعت و کارایی دیتابیس می‌گردد. بنابراین نیاز بود تا تکنولوژی جدیدی به نام دیتابیس‌های NoSQL با هدف ذخیره سازی و کار با دیتای بدون ساختار و حجیم تعریف گردد. به دلیل این که NoSQL باید بتواند انواع مختلف دیتای سازمان نیافته را ذخیره کند در ساختار داخلی آن "Dynamic Schema" به کار رفته است. در مقایسه با SQL، NoSQL قادر به پاسخگویی به کوئری‌های پیچیده نمی‌باشد و نیز امکان بروز خطای پیش بینی نشده به هنگام ثبت و تغییر دیتا وجود دارد.

انواع دیتابیس‌های NoSQL و کاربرد آنها: (همچنین ممکن است این دیتابیسها به صورت تلفیقی به کار روند.)

دیتابیس‌های Key-value NoSQL : این دسته از دیتابیسها، ساده‌ترین و در عین حال پرکاربردترین نوع دیتابیسهای NoSQL هستند. در این دسته، دیتابیس از یک key برای دریافت و ذخیره‌سازی value استفاده می‌کند.

دیتابیس‌های Document NoSQL : این دسته از دیتابیس‌ها برای ذخیره سازی و کار با document ها با فرمت‌های ... , XML, JSON به کار می‌روند. از این نوع دیتابیسها برای ذخیره‌ی دیتای بدون ساختار مشخص با پراکندگی بالا استفاده می‌شود.

دیتابیس‌های Wide-column NoSQL : این دسته از دیتابیس‌ها در ظاهر مانند SQL از جدول استفاده می‌کنند اما عملکردشان مانند SQL نمی‌باشد. بر خلاف SQL ، در اینجا نوع تعریف و فرمت یک ستون میتواند در هر سطر متفاوت باشد. این دیتابیس‌ها دارای انعطاف پذیری بالا در ثبت و کار با کلان دیتاها هستند.

دیتابیس‌های Graph NoSQL: این دسته، برای ذخیره کردن حجم زیادی از relational data طراحی شده‌اند. در شکل هندسی این گونه از دیتابیسها، هر داده به صورت یک رأس و هر لینک به صورت یک یال ترسیم میگردد و از این دیتابیس می توان برای ذخیره‌ی انواع معماری‌های داده‌های شبکه‌ای نیز استفاده کرد.

۷- از دیتابیس‌های in-memory، برای مدیریت سیستم مبتنی بر حافظه (دیتا به جای ذخیره‌ی روی دیسک بر memory یا RAM ذخیره می‌گردد) استفاده می‌شود. اکثر دیتابیسها اطلاعات را روی دسیک ذخیره می‌کنند و به دلیل اینکه مکانیسم خواندن و نوشتن بر روی دیسک‌ها فیزیکی است، دسترسی به اطلاعات نیاز به صرف زمان زیادی دارد. به منظور دسترسی سریع به داده‌ها از in-memory Databases یا به اختصار IMDB استفاده می‌کنیم. (برای دسترسی به آدرسهای مختلف در RAM باید از CPU با قدرت پردازش بالا استفاده کنیم.)

بنابراین هرجا که نیاز به دسترسی سریع، با تعدا فراخوانی بالا به داده‌ها وجود داشته باشد، (به عنوان مثال در بانکداری، بازیهای آنلاین تعاملی، پردازش داده‌ی سنسورها و ...) از IMDB استفاده می‌کنیم. یکی از معایب دیتابیس‌های IMDB، پاک شدن یا از دست رفتن دیتا در صورت خاموشی و یا خرابی سرور (به علت ذخیره ی دیتا بر روی RAM) می‌باشد. برای رفع این مشکل می توان از مکانیزم‌های Transaction logging، Snapshot files و ... استفاده کرد.

از جمله از دیتابیسهای IMDB می توان به Redis، H۲، TaranTool، ArangoDB اشاره کرد.

بـخش دوم

نتیجه‌ی تمامی متدهای خوا سته شده برای کلاس Sequence با کلاس String تطابق دا شته و این قضـیه به طور کامل چک شـده اسـت (نتایج مربوطه در شـکلهای ۲، ۳، ۴ و ۵ برای هرکدام از متدها به صـورت جداگانه آمده اسـت.)، با این تفاوت که پس از اعمال آنها مقدار المان های آرایه value در کلاس Sequence نیز تغییر می‌کند.

```
concat method results:
Value1 is: [H, e, l, l, o], Value2 is: [ , T, a, b, a, n]
Result by using Sequence Class: Hello Taban | Result by using String Class: Hello Taban
Result is matched :)
```

شکل ۲ concat method

```
indexOf method results:
Value is: [H, e, l, l, o], char is: l
Result by using Sequence Class: 2 | Result by using String Class: 2
Result is matched :)
+
Value is: [H, e, l, l, o], char is: L
Result by using Sequence Class: -1 | Result by using String Class: -1
Result is matched :)
+
Value is: [H, e, l, l, o], char is: e
Result by using Sequence Class: 1 | Result by using String Class: 1
Result is matched :)
+
Value is: [H, e, l, l, o], char is: space
Result by using Sequence Class: -1 | Result by using String Class: -1
Result is matched :)
```

شکل ۳ indexOf method

```
replace method results:
Seq is: [N, o, i, s, e], oldSeq is: [i, s, e], newSeq is: []
Result by using Sequence Class: No | Result by using String Class: No
Result is matched :)
+
Seq is: [N, o], oldSeq is: [], newSeq is: [+, +]
Result by using Sequence Class: ++N++o++ | Result by using String Class: ++N++o++
Result is matched :)
+
Seq is: [H, e, r, m, a, n, o], oldSeq is: [r, m, a, n], newSeq is: [l, l]
Result by using Sequence Class: Hello | Result by using String Class: Hello
Result is matched :)
+
Seq is: [P, i, v, o, t], oldSeq is: [i, v, y], newSeq is: [e, a, n, u]
Result by using Sequence Class: Pivot | Result by using String Class: Pivot
Result is matched :)
```

شکل۴ replace method

```
equals method results:
Value1 is: [f, l, a, g, s], Value2 is: [f, l, a, g, s]
Result by using Sequence Class: true | Result by using String Class: true
Result is matched :)
+
Value1 is: [f, l, a, g, s], Value2 is: [f, r, o, g, s]
Result by using Sequence Class: false | Result by using String Class: false
Result is matched :)
```

شکل ۵ equals method

۹- در این سوال چهار کلاس با نام‌های TravelAgency به عنوان آژانس هواپیمایی، FlightTicket به عنوان بلیط پرواز، SalesPerson به عنوان فروشنده و Customer به عنوان مشتری طراحی شده‌اند.

۱۰- برای این سوال ۸ کلاس تعریف شده است. کلاس‌های Stadium برای استادیوم، Team برای هریک از تیم‌ها، Ticket برای بلیط بازی، Spectator برای تماشاگرها، Referee برای داور، Player برای هریک از بازیکن‌ها، Match برای مسابقه و کلاس Coach برای مربی تعریف شده اند.