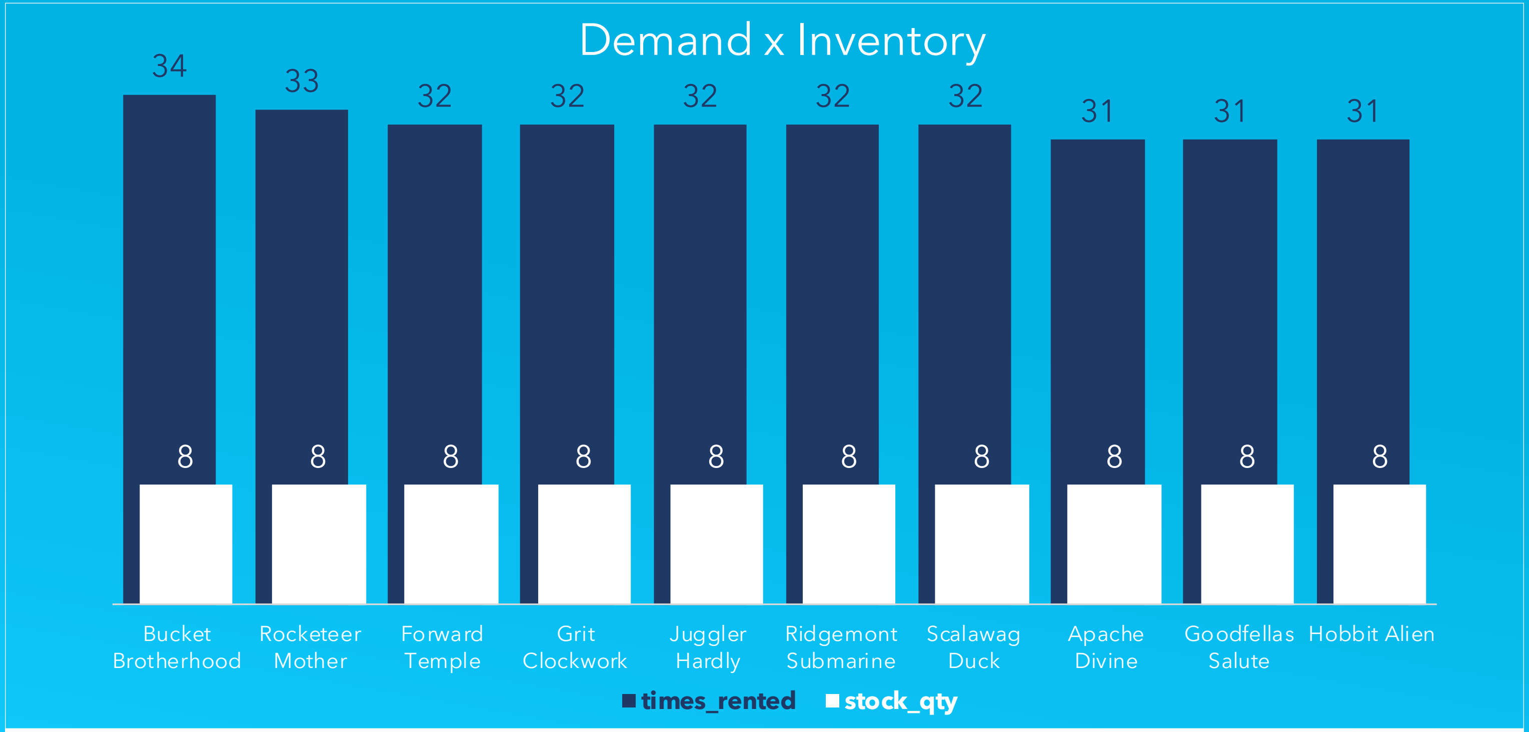
Programming for Data Science with Python Nanodegree Program

Introduction to SQL

Project: Sakila DVD Rental database

1 Is there enough supply to meet Sakila's clients' demand?



Sakila is stocked aplenty considering that, from the sample above (comprising the most sought after titles), we can see that there is more than enough supply to meet demand (8 unities should suffice for an avg of rentals of 30+ over the course of almost a whole year). Besides, the table retrived by the query shows that the most rented out titles (top decile) also are in the top deciles of unities in inventory.

Query Used:

```
WITH
top_rented AS
(SELECT f.title movie, COUNT(*) times_rented,
 NTILE(10) OVER(ORDER BY COUNT(*)) quant_rent
FROM film f
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY 1),

invent AS
(SELECT i.film_id, f.title movie, COUNT(*) stock_qty,
NTILE(10) OVER(ORDER BY COUNT(*)) quant_stock
FROM inventory i
JOIN film f ON i.film_id = f.film_id
GROUP BY 1,2)

SELECT t.movie, t.quant_rent, i.quant_stock , i.stock_qty,
t.times_rented
FROM top_rented t
JOIN invent i ON t.movie = i.movie
ORDER BY 5 DESC, 1
LIMIT 100;
```

Enzo Yamamura

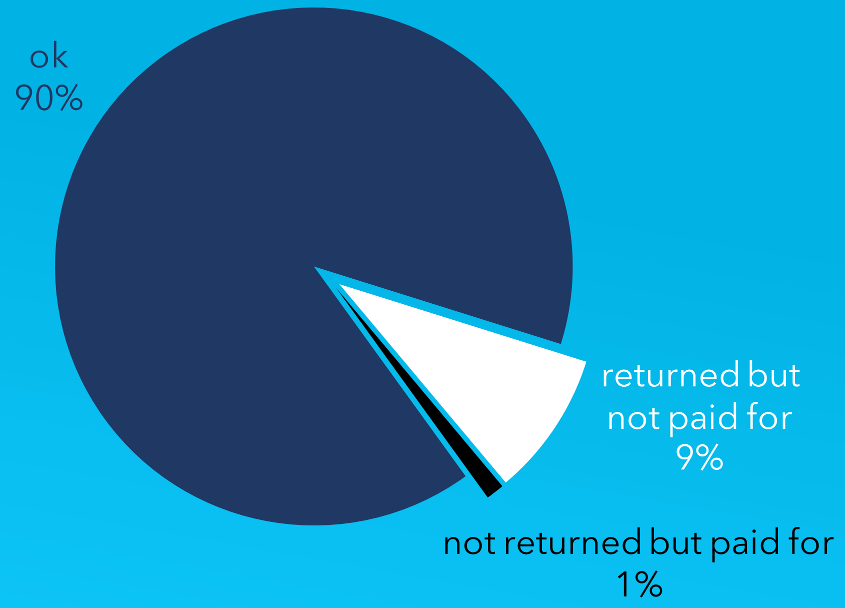
Programming for Data Science with Python Nanodegree Program

Introduction to SQL

Project: Sakila DVD Rental database

2 Are customers returning / paying for the rented DVDs?

Rental Status (both stores)



The query clearly shows that there is no significant difference between return / payment between the two stores. Either by analysing them individually or at once we get the same picture: the majority of customers both return and pay for the rentals, although a significant amount (9%) returned the DVDs without paying. 1% have not returned the DVDs but paid the rental. Fortunatelly, none of the clients forgot both to pay and to return the DVDs. Maybe payment should have been enforced better.

Query Used:

```
WITH rent_pay AS
(SELECT
i.store_id,
r.rental_id,
r.rental_date,
r.return_date,
p.payment_date,
p.amount
FROM inventory i
JOIN rental r ON i.inventory_id = r.inventory_id
LEFT JOIN payment p ON r.rental_id = p.rental_id
/* Left join due to some of the rentals apparently not being paid for*/
),

hist_data AS
(SELECT
rent_pay.store_id store,
rent_pay.rental_id,
DATE_TRUNC('day',rent_pay.rental_date) rental,
DATE_TRUNC('day',rent_pay.return_date) return,
DATE_TRUNC('day',rent_pay.payment_date) payment_date,
CASE WHEN DATE_TRUNC('day',rent_pay.return_date) IS NULL AND
DATE_TRUNC('day',rent_pay.payment_date) IS NULL  THEN 'not returned nor paid for'
 WHEN DATE_TRUNC('day',rent_pay.return_date) IS NULL AND
DATE_TRUNC('day',rent_pay.payment_date) IS NOT NULL  THEN 'not returned but paid for'
 WHEN DATE_TRUNC('day',rent_pay.return_date) IS NOT NULL AND
DATE_TRUNC('day',rent_pay.payment_date) IS NULL  THEN 'returned but not paid for'
ELSE 'ok' END AS status
FROM rent_pay)


SELECT store,status, COUNT(*) FROM hist_data
GROUP BY 1,2
ORDER BY 1,2
```

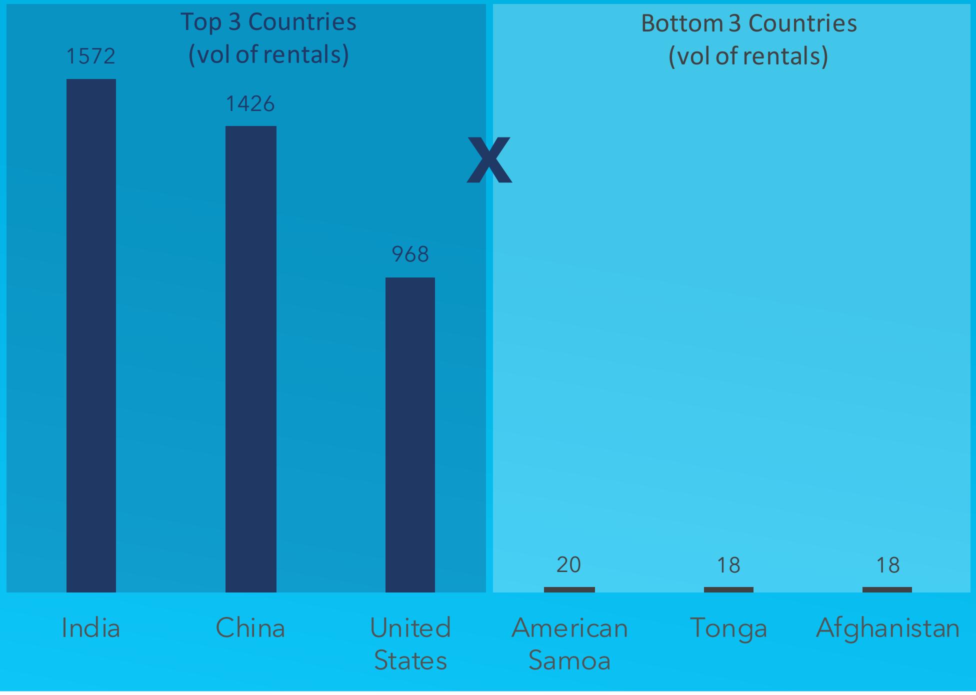
Enzo Yamamura

Programming for Data Science with Python Nanodegree Program

Introduction to SQL

Project: Sakila DVD Rental database

3 Suppose Sakila's is considering regional marketing strategies. Where are the clients from?



Sakila rents DVDs all over the globe. Still, both the query and the chart above depict the stark contrast between its total rentals among countries. In the chart above it is shown both the top 3 and bottom 3 countries in total volume of rentals during the analysed time period, which would have shed light on the need of hypothetical marketing campaigns in the countries it has less market share.

Query Used:

```
(SELECT country.country,
COUNT(*)
FROM rental r
JOIN customer c ON r.customer_id = c.customer_id
JOIN address a ON c.address_id = a.address_id
JOIN city ON a.city_id = city.city_id
JOIN country ON city.country_id = country.country_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 3)
UNION
(SELECT country.country,
COUNT(*)
FROM rental r
JOIN customer c ON r.customer_id = c.customer_id
JOIN address a ON c.address_id = a.address_id
JOIN city ON a.city_id = city.city_id
JOIN country ON city.country_id = country.country_id
GROUP BY 1
ORDER BY 2 ASC
LIMIT 3)
ORDER BY 2 DESC
```

Enzo Yamamura

Programming for Data Science with Python Nanodegree Program

Introduction to SQL

Project: Sakila DVD Rental database

4 Suppose Sakila's owner decided to start making suggestions for customers based on the data collected. Is it possible?

All possible suggestions for client "Adam Goochi"

suggestion	categ
Rocketeer Mother	Foreign

suggestion	categ
Pulp Beverly	Horror

suggestion	categ
Robbers Joon	Children

Yes, as shown above. Basically, the query retrieves genre preference of the selected client via his/her previous rentals. If there are multiple favorite genres (same count of rentals) the query randomly chooses between any of them. It proceeds to select between the most popular titles one that: 1) matches with the genre 2) is the best ranked among those that had not been previously rented by said client. Above are all the possible suggestions for "Adam Gooch" retrieved by running the query multiple times. It works the same whatever client name is chosen, just substitute it in.

Query Used:

```
WITH t1 AS
(
SELECT cust.first_name ||' '||cust.last_name client,r.rental_id id,f.title, c.name
FROM category c
JOIN film_category fc ON c.category_id = fc.category_id
JOIN film f ON fc.film_id = f.film_id
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
JOIN customer cust ON r.customer_id = cust.customer_id),

t2 AS
(
SELECT name, title,COUNT(*) times_rented,
ROW_NUMBER(*) OVER (PARTITION BY name ORDER BY COUNT(*) DESC) rank_categ
FROM t1
GROUP BY 1,2),

client_categ AS
(
SELECT client, name category,COUNT(*) count
FROM t1
GROUP BY 1, 2),

fav_list AS
(SELECT client_categ.* FROM client_categ
JOIN (SELECT client, MAX(count) fav FROM client_categ GROUP BY 1) max
ON (client_categ.client = max.client AND client_categ.count = max.fav)
ORDER BY 1)
/* the code above returns each clients favorte categories. Some clients have more than one favorite genre */

SELECT t2.title suggestion,
t2.name categ
FROM t2
WHERE t2.title NOT IN
(SELECT t1.title
FROM t1
WHERE t1.client = 'Adam Gooch')
AND t2.name =
(SELECT category FROM fav_list WHERE client ='Adam Gooch' ORDER BY RANDOM() LIMIT 1)
ORDER BY t2.rank_categ
LIMIT 1
```

Enzo Yamamura