[SQL Assignment-3 Query.sql](https://github.com/premsingh36/PW-Assignment-3/commit/b93f1e5ad9f8506704aeac1d28cffecb2cac996b" \l "diff-fbbd64196d093d78b0e53f2da5052beaaa779a310f301fecda1c2a9ce91e0a93" \o "SQL Assignment-3 Query.sql)

|  |  |  |
| --- | --- | --- |
|  |  | @@ -0,0 +1,111 @@ |
|  |  | use mavenmovies; |
|  |  | -- Q1. \*\*Rank the customers based on the total amount they've spent on rentals.\*\* |
|  |  | select distinct c.customer\_id as id ,concat(first\_name, " ", last\_name) as name, sum(amount) over(), |
|  |  | sum(amount) over (order by c.customer\_id) as total |
|  |  | from customer c inner join payment p on c.customer\_id = p.customer\_id; |
|  |  |  |
|  |  | -- Q2. Calculate the cumulative revenue generated by each film over time. |
|  |  | Select distinct f.film\_id, title, sum(amount) over (order by film\_id) from |
|  |  | film f inner join inventory i on f.film\_id = i.film\_id |
|  |  | inner join rental r on r.inventory\_id = i.inventory\_id |
|  |  | inner join payment p on p.rental\_id = r.rental\_id; |
|  |  |  |
|  |  | -- Q3. Determine the average rental duration for each film, considering films with similar lengths. |
|  |  | select distinct f.film\_id, title, avg(rental\_duration) over (order by film\_id) from |
|  |  | film f inner join inventory i on f.film\_id = i.film\_id |
|  |  | inner join rental r on r.inventory\_id = i.inventory\_id; |
|  |  |  |
|  |  | -- Q4. Identify the top 3 films in each category based on their rental counts. |
|  |  | WITH RankedFilms AS ( |
|  |  | SELECT |
|  |  | c.name AS category, |
|  |  | f.title AS film\_title, |
|  |  | COUNT(r.rental\_id) AS rental\_count, |
|  |  | ROW\_NUMBER() OVER (PARTITION BY c.category\_id ORDER BY COUNT(r.rental\_id) DESC) AS rank\_within\_category |
|  |  | FROM category c |
|  |  | inner JOIN film\_category fc ON c.category\_id = fc.category\_id |
|  |  | inner JOIN film f ON fc.film\_id = f.film\_id |
|  |  | LEFT JOIN inventory i ON f.film\_id = i.film\_id |
|  |  | LEFT JOIN rental r ON i.inventory\_id = r.inventory\_id |
|  |  | GROUP BY c.category\_id, f.film\_id, f.title |
|  |  | ) |
|  |  | SELECT category, film\_title, rental\_count |
|  |  | FROM RankedFilms |
|  |  | WHERE rank\_within\_category <= 3; |
|  |  |  |
|  |  | -- Q5. calculate the difference in total rental counts between each customer total rentals and the avg rental across all the customer. |
|  |  |  |
|  |  | SELECT |
|  |  | customer\_id, |
|  |  | total\_rentals, |
|  |  | AVG(total\_rentals) OVER () AS avg\_rentals\_across\_all\_customers, |
|  |  | total\_rentals - AVG(total\_rentals) OVER () AS rental\_count\_difference |
|  |  | FROM ( |
|  |  | SELECT |
|  |  | c.customer\_id, COUNT(r.rental\_id) AS total\_rentals |
|  |  | FROM customer c |
|  |  | inner JOIN rental r ON c.customer\_id = r.customer\_id |
|  |  | GROUP BY c.customer\_id |
|  |  | ) AS customer\_rentals; |
|  |  |  |
|  |  | -- 6. find the monthly revenue trend for entire rental store over time. |
|  |  |  |
|  |  | WITH MonthlyRevenue AS ( |
|  |  | SELECT |
|  |  | DATE\_FORMAT(payment.payment\_date, '%Y-%m') AS month, |
|  |  | SUM(payment.amount) AS monthly\_revenue, |
|  |  | ROW\_NUMBER() OVER (ORDER BY DATE\_FORMAT(payment.payment\_date, '%Y-%m')) AS month\_rank |
|  |  | FROM payment |
|  |  | GROUP BY DATE\_FORMAT(payment.payment\_date, '%Y-%m') |
|  |  | ) |
|  |  | SELECT month, monthly\_revenue, |
|  |  | LAG(monthly\_revenue) OVER (ORDER BY month\_rank) AS previous\_month\_revenue |
|  |  | FROM MonthlyRevenue; |
|  |  |  |
|  |  | -- Q7. identify the customers whose total spending on rental falls within the top 20% of all customers. |
|  |  | with top\_customer as ( |
|  |  | select c.customer\_id, concat(c.first\_name, " ", c.last\_name) as name, sum(p.amount) as spending, |
|  |  | percent\_rank() over (order by sum(amount) desc) as spending\_percentage from |
|  |  | customer c inner join payment p on c.customer\_id = p.customer\_id |
|  |  | group by c.customer\_id) |
|  |  | select customer\_id,name, spending |
|  |  | from top\_customer |
|  |  | where spending\_percentage <= 0.2; |
|  |  |  |
|  |  |  |
|  |  | -- 9. find the films that have been rented less than the average rental count for their respective category. |
|  |  | with film\_rental as ( |
|  |  | select f.film\_id, fc.category\_id, count(r.rental\_id) as rental\_count, |
|  |  | avg(count(r.rental\_id)) over (partition by fc.category\_id) as avg\_count, |
|  |  | row\_number() over (partition by fc.category\_id order by count(r.rental\_id) desc) as film\_rank |
|  |  | from film f |
|  |  | inner join film\_category fc on f.film\_id = fc.film\_id |
|  |  | inner join inventory i on i.film\_id = f.film\_id |
|  |  | inner join rental r on r.inventory\_id = i.inventory\_id |
|  |  | group by f.film\_id, fc.category\_id) |
|  |  |  |
|  |  | select fr.film\_id, fr.category\_id, f.title, fr.rental\_count, fr.avg\_count |
|  |  | from film\_rental fr inner join film f on f.film\_id = fr.film\_id |
|  |  | where fr.rental\_count < fr.avg\_count |
|  |  | order by fr.category\_id, fr.film\_rank; |
|  |  |  |
|  |  | -- 10. Identify the top 5 film with the highest revenue and display the revenue generated in each month. |
|  |  |  |
|  |  | with monthly\_revenue as ( |
|  |  | select f.film\_id, f.title, |
|  |  | date\_format(r.rental\_date, '%y,%m') as rental\_month, sum(p.amount) as revenue, |
|  |  | row\_number() over (partition by date\_format(r.rental\_date, '%y,%m') order by sum(p.amount) desc) as film\_rank |
|  |  | from film f |
|  |  | inner join inventory i on f.film\_id = i.film\_id |
|  |  | inner join rental r on r.inventory\_id = i.inventory\_id |
|  |  | inner join payment p on p.rental\_id = r.rental\_id |
|  |  | group by f.film\_id, f.title, rental\_month) |
|  |  |  |
|  |  | select mr.film\_id, mr.title, mr.rental\_month, mr.revenue from monthly\_revenue mr |
|  |  | where mr.film\_rank <= 5 |
|  |  | order by mr.rental\_month, mr.film\_rank; |
|  |  |  |
|  |  | select fr.film\_id, fr.category\_id, f.title, fr.rental\_count, fr.avg\_count |
|  |  | from film\_rental fr inner join film f on f.film\_id = fr.film\_id |
|  |  | where fr.rental\_count < fr.avg\_count |
|  |  | order by fr.category\_id, fr.film\_rank; |