**Meets Specifications**

**Congratulations on submitting this project! You should be proud of how far you’ve come . I really enjoyed reviewing your work.**

**You have successfully demonstrated a strong knowledge of both basic and advanced SQL queries, as well as how to answer questions like a data analyst would and used visualizations to show insights. Great job!**

**Queries**

**All SQL queries run without errors and produce the intended results.**

All your SQL queries run without errors .

**Each SQL query needs to include one or more explicit JOINs. The JOIN or JOINs should be necessary to the query.**

**If a question does not require a JOIN please change the question to be one that does.**

All SQL queries have JOINs and you used them correctly.

**Each SQL query needs to include one or more aggregations. This could be a COUNT, AVG, SUM, or other aggregation.**

All your queries included at least one aggregation.

**At least 2 of the 4 SQL queries need to include either a subquery OR a CTE.**

Nice use of subqueries!

**At least 1 of the 4 queries should use a Window Function.**

Really great use of window functions!

**The SQL queries are well formatted and use aliases.**

All your queries were properly formatted and use Aliases

**Presentation**

**Each slide should have a question and an appropriate visualization descriptions to address the question. The slides should be free of significant factual, spelling and grammatical mistakes.**

You answered the questions properly, had uncluttered charts and no significant spelling or grammatical mistakes. Way to go!

**All visualizations should make logical sense and provide accurate analysis based on their query results.**

All your visualizations made logical sense and provided accurate analysis based on their query results.

1. **All visualizations include a title and axis labels, have a legend where applicable, and are easily understood.**
2. **Every visualization should have:**
   * **chart title**
   * **x axis title**
   * **x axis label**
   * **y axis title**
   * **y axis labels**

All your charts included a title and axis labels, had a legend where applicable, and were easily understood by me.

**Code Review Comments**

**Q1.**

**/\*Question 1:**

**We want to understand more about the movies that families are watching. The following categories are considered family movies: Animation, Children, Classics, Comedy, Family and Music.**

**Create a query that lists each movie, the film category it is classified in, and the number of times it has been rented out.**

**Query 1**

**\*/**

**SELECT**

**film.title film\_title,**

**category.name category\_name,**

**SUM(rental.rental\_id)**

**FROM film**

**JOIN film\_category**

**ON film.film\_id = film\_category.film\_id**

**JOIN category**

**ON category.category\_id = film\_category.category\_id**

**JOIN rental**

**ON rental.rental\_id = film.film\_id**

Q1 SUGGESTION

The join between the rental & film tables is incorrect. When you join on non-matching fields, it results in wrong results.

Here's how you should join the two tables:

SELECT

f.title film\_title,

c.name category\_name,

COUNT(r.rental\_id)

FROM film AS f

JOIN film\_category AS fc

ON f.film\_id = fc.film\_id

JOIN category AS c

ON fc.category\_id = c.category\_id

JOIN inventory AS i

ON f.film\_id = i.film\_id

JOIN rental AS r

ON i.inventory\_id = r.inventory\_id

WHERE c.name IN

('Animation','Classics','Children','Comedy','Family','Music')

GROUP BY 1,

2

ORDER BY 2, 1;

**Q2**

**GROUP BY 1,**

**2**

**ORDER BY 3 DESC;**

**/\***

**Question 2:**

**Now we need to know how the length of rental duration of these family-friendly movies compares to the duration that all movies are rented for.**

**Can you provide a table with the movie titles and divide them into 4 levels (first\_quarter, second\_quarter, third\_quarter, and final\_quarter) based on the quartiles (25%, 50%, 75%) of the rental duration for movies across all categories?**

**Make sure to also indicate the category that these family-friendly movies fall into.**

**Query 2**

**\*/**

**SELECT**

**f.title film\_title,**

**c.name category\_name,**

**f.rental\_duration AS rental\_duration,**

**NTILE(4) OVER (ORDER BY f.rental\_duration) AS quartiles**

**FROM film f**

**JOIN film\_category fc**

**ON f.film\_id = fc.film\_id**

**JOIN category c**

**ON c.category\_id = fc.category\_id**

**WHERE c.name IN ('Animation', 'Children', 'Classics', 'Comedy', 'Family', 'Music')**

**GROUP BY 1,**

Q2 SUGGESTION

The GROUP BY here is not needed because you aren't aggregating on any fields in your select statement.  
GROUP By's are used when there are aggregations like SUM, MIN, MAX e.t.c in your query.

SELECT

f.title film\_title,

c.name category\_name,

f.rental\_duration AS rental\_duration,

NTILE(4) OVER (ORDER BY f.rental\_duration) AS quartiles

FROM film f

JOIN film\_category fc

ON f.film\_id = fc.film\_id

JOIN category c

ON c.category\_id = fc.category\_id

WHERE c.name IN ('Animation', 'Children', 'Classics', 'Comedy', 'Family', 'Music')

ORDER BY 4;

**Q3.**

**2,**

**3**

**ORDER BY 3 DESC;**

**/\*Question 3:**

**provide a table with the family-friendly film category, each of the quartiles, and the corresponding count of movies within each combination of film category for each corresponding rental duration category.**

**The resulting table should have three columns:**

**Category**

**Rental length category**

**Count**

**Query 3:**

**\*/**

**SELECT**

**category\_name,**

**quartiles,**

**COUNT(category\_name)**

**FROM (SELECT**

**c.name category\_name,**

**NTILE(4) OVER (ORDER BY f.rental\_duration) AS quartiles**

**FROM film f**

**JOIN film\_category fc**

Q3 AWESOME

Way to go! Your query ran without any errors and returned the correct results.

**Q4.**

**ON f.film\_id = fc.film\_id**

**JOIN category c**

**ON c.category\_id = fc.category\_id**

**WHERE c.name IN ('Animation', 'Children', 'Classics', 'Comedy', 'Family', 'Music')) t1**

**GROUP BY 1,**

**2**

**ORDER BY 1, 2;**

**/\*Question 4:**

**Write a query that returns the store ID for the store, the year and month and the number of rental orders each store has fulfilled for that month.**

**Your table should include a column for each of the following: year, month, store ID and count of rental orders fulfilled during that month.**

**Query 4\*/**

**SELECT**

**DATE\_PART('YEAR', rental\_date) YEARs,**

**DATE\_PART('MONTH', rental\_date) MONTHs,**

**store.store\_id,**

**COUNT(\*)**

**FROM rental**

**JOIN payment**

Q4 SUGGESTION

You are currently JOINing to the payment table, this JOIN is not needed to fetch any fields. Hence it's unnecessary and making your results slightly off. This is usually the case when GROUP BY's are involved.  
It is best practice to avoid JOINing on tables you won't be selecting anything from.

Here's your query with my suggestion included:

SELECT

DATE\_PART('YEAR', rental\_date) YEARs,

DATE\_PART('MONTH', rental\_date) MONTHs,

store.store\_id,

COUNT(\*)

FROM rental as r

JOIN staff

ON staff.staff\_id = r.staff\_id

JOIN store

ON store.store\_id = staff.store\_id

GROUP BY 1,

2,

3

order by 4 desc;

**ON payment.rental\_id = rental.rental\_id**

**JOIN staff**

**ON staff.staff\_id = payment.staff\_id**

**JOIN store**

**ON store.store\_id = staff.store\_id**

**GROUP BY 1,**

**2,**

**3**

**order by 4 desc;**