Advance SQL Assignment

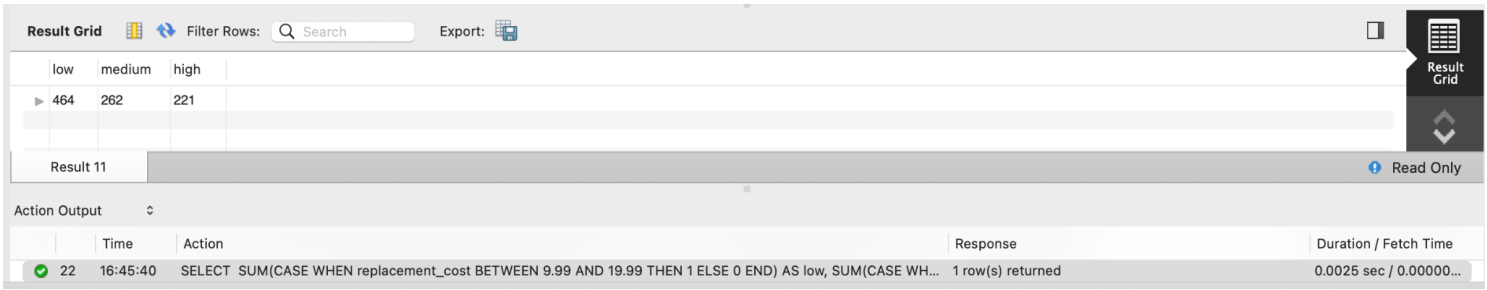
Q1. Write a query that gives an overview of how many films have replacements costs in the following cost ranges

low: 9.99 - 19.99
medium: 20.00 - 24.99
high: 25.00 - 29.99

Solution:
Query:

```
SELECT
SUM(CASE WHEN replacement_cost BETWEEN 9.99 AND 19.99 THEN 1
ELSE 0 END) AS low,
SUM(CASE WHEN replacement_cost BETWEEN 20.00 AND 24.99 THEN 1
ELSE 0 END) AS medium,
SUM(CASE WHEN replacement_cost BETWEEN 25.00 AND 29.99 THEN 1
ELSE 0 END) AS high
FROM film;
```

Output:



Q2. Write a query to create a list of the film titles including their film title, film length and film category name ordered descendingly by the film length. Filter the results to only the movies in the category 'Drama' or 'Sports'.

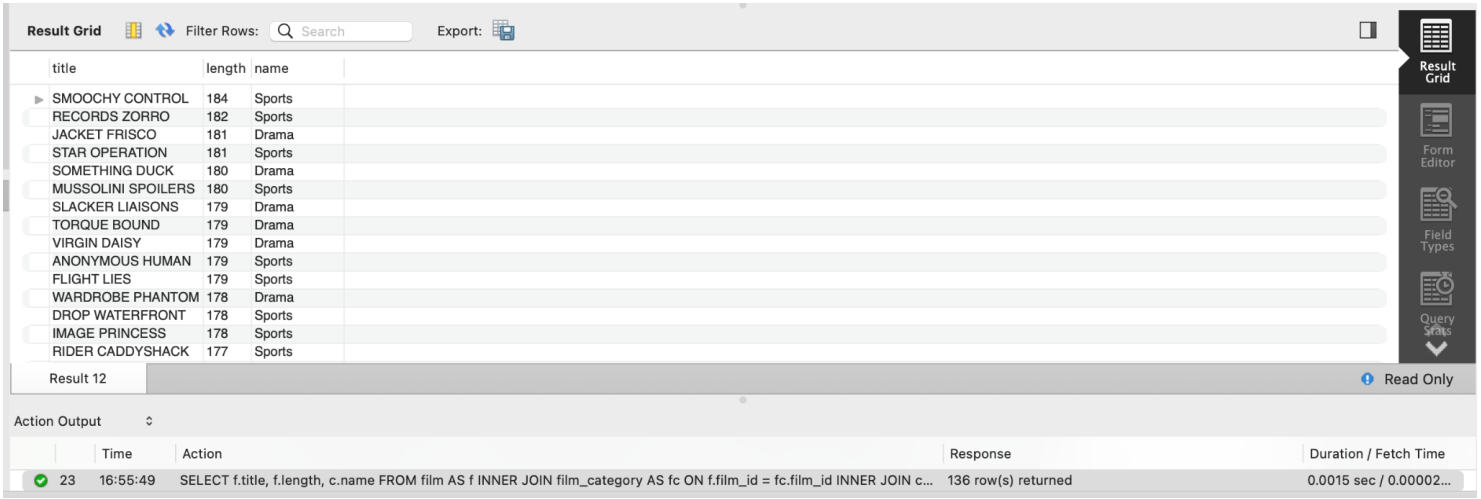
Eg. "STAR OPERATION" "Sports" 181
"JACKET FRISCO" "Drama" 181

Solution:

Query:

```
SELECT f.title, f.length, c.name
FROM film AS f INNER JOIN film_category AS fc
ON f.film_id = fc.film_id
INNER JOIN category AS c
ON fc.category_id = c.category_id
WHERE c.name IN ('Sports', 'Drama')
ORDER BY f.length DESC;
```

Output:



title	length	name
SMOOCHY CONTROL	184	Sports
RECORDS ZORRO	182	Sports
JACKET FRISCO	181	Drama
STAR OPERATION	181	Sports
SOMETHING DUCK	180	Drama
MUSSOLINI SPOILERS	180	Sports
SLACKER LIAISONS	179	Drama
TORQUE BOUND	179	Drama
VIRGIN DAISY	179	Drama
ANONYMOUS HUMAN	179	Sports
FLIGHT LIES	179	Sports
WARDROBE PHANTOM	178	Drama
DROP WATERFRONT	178	Sports
IMAGE PRINCESS	178	Sports
RIDER CADDYSHACK	177	Sports

Q3. Write a query to create a list of the addresses that are not associated to any customer.

Solution:

Query:

```
SELECT a.address_id, a.address, a.district, a.city_id
FROM address AS a LEFT JOIN customer AS c
```

```
ON a.address_id = c.address_id
WHERE c.customer_id IS NULL;
```

Output:

Result Grid | Filter Rows: Search | Export:

address_id	address	district	city_id
1	47 MySakila Drive	Alberta	300
2	28 MySQL Boulevard	QLD	576
3	23 Workhaven Lane	Alberta	300
4	1411 Lillydale Drive	QLD	576

Result 13 | Read Only

Action Output

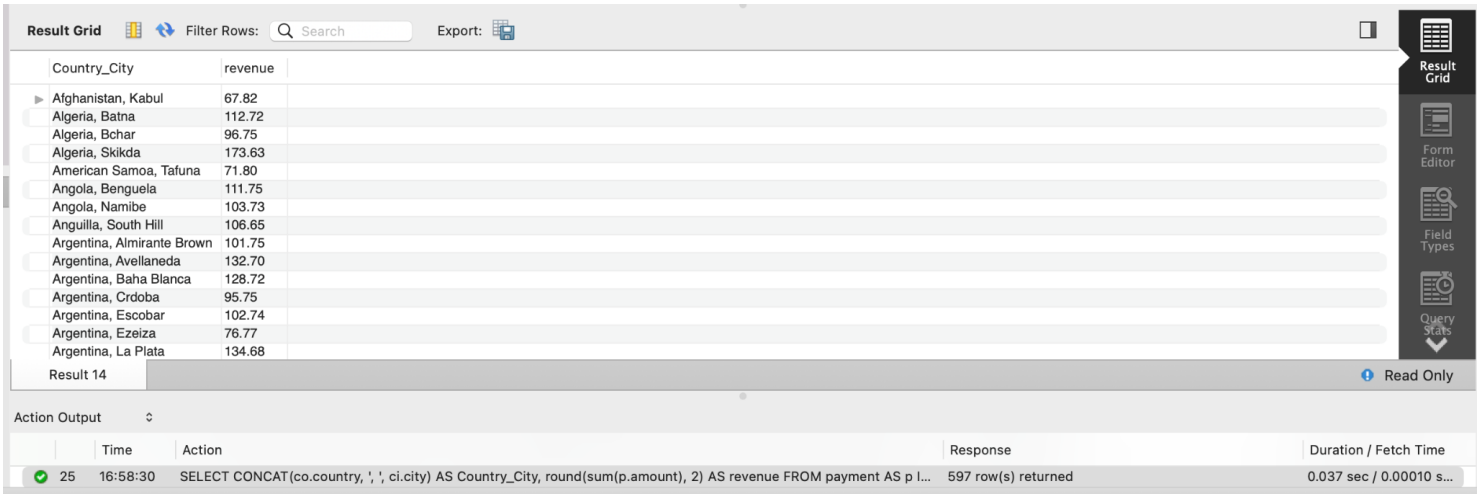
	Time	Action	Response	Duration / Fetch Time
24	16:57:13	SELECT a.address_id, a.address, a.district, a.city_id FROM address AS a LEFT JOIN customer AS c ON a.address_id =...	4 row(s) returned	0.0028 sec / 0.00000...

Q4. Write a query to create a list of the revenue (sum of amount) grouped by a column in the format "country, city" ordered in decreasing amount of revenue.
eg. "Poland, Bydgoszcz" 52.88

Solution:
Query:

```
SELECT CONCAT(co.country, ', ', ci.city) AS Country_City,
round(sum(p.amount), 2) AS revenue
FROM payment AS p INNER JOIN customer AS cu
ON p.customer_id = cu.customer_id
INNER JOIN address AS a
ON cu.address_id = a.address_id
INNER JOIN city AS ci
ON ci.city_id = a.city_id
INNER JOIN country AS co
ON co.country_id = ci.country_id
GROUP BY co.country, ci.city;
```

Output:



Result Grid | Filter Rows: Search | Export:

Country_City	revenue
Afghanistan, Kabul	67.82
Algeria, Batna	112.72
Algeria, Bchar	96.75
Algeria, Skikda	173.63
American Samoa, Tafuna	71.80
Angola, Benguela	111.75
Angola, Namibe	103.73
Anguilla, South Hill	106.65
Argentina, Almirante Brown	101.75
Argentina, Avellaneda	132.70
Argentina, Baha Blanca	128.72
Argentina, Crdoba	95.75
Argentina, Escobar	102.74
Argentina, Ezeiza	76.77
Argentina, La Plata	134.68

Result 14 | Read Only

Result Grid | Form Editor | Field Types | Query Stats

Action Output

	Time	Action	Response	Duration / Fetch Time
25	16:58:30	SELECT CONCAT(co.country, ', ', ci.city) AS Country_City, round(sum(p.amount), 2) AS revenue FROM payment AS p l...	597 row(s) returned	0.037 sec / 0.00010 s...

Q5. Write a query to create a list with the average of the sales amount each staff_id has per customer.

result: **2** **56.64**

1 **55.91**

Solution:

Query:

```
SELECT t1.staff_id, round(AVG(t1.total_sum), 2)
FROM
(SELECT p.staff_id, p.customer_id, SUM(p.amount) AS total_sum
FROM payment AS p
GROUP BY p.staff_id, p.customer_id) AS t1
GROUP BY t1.staff_id;
```

Output:

Q6. Write a query that shows average daily revenue of all Sundays.

Solution:

Query:

```
SELECT ROUND(AVG(t1.sum_by_each_sunday), 2)
FROM
(SELECT DATE(payment_date), SUM(amount) AS sum_by_each_sunday
FROM payment
WHERE DAYNAME(payment_date)='Sunday'
GROUP BY DATE(payment_date)) AS t1;
```

Output:

Q7. Write a query to create a list that shows how much the average customer spent in total (customer life-time value) grouped by the different districts.

Solution:

Query:

```
SELECT t1.district, AVG(t1.avg_amt_by_district)
FROM
(SELECT a.district, c.customer_id, SUM(p.amount) AS avg_amt_by_district
FROM payment AS p
INNER JOIN customer AS c
ON p.customer_id = c.customer_id
INNER JOIN address AS a
ON a.address_id = c.address_id
GROUP BY a.district, c.customer_id) AS t1
GROUP BY t1.district;
```

Output:

Result Grid | Filter Rows: Search | Export:

district	AVG(t1.avg_amt_by_distri...
Nagasaki	118.68000000
California	117.38666667
Attika	135.74000000
Mandalay	81.78000000
Nantou	104.71500000
Texas	101.54200000
Central Serbia	151.67000000
Hamilton	92.76000000
Masqat	89.77000000
Esfahan	148.00666667
Kanagawa	95.77000000
Haryana	128.18500000
Osmaniye	131.73000000
Madhya Pradesh	117.05333333
England	102.04571429

Result 19 | Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
30	17:06:28	SELECT t1.district, AVG(t1.avg_amt_by_district) FROM (SELECT a.district, c.customer_id, SUM(p.amount) AS avg_amt...	376 row(s) returned	0.039 sec / 0.000050...

Q8. Write a query to list down the highest overall revenue collected (sum of amount per title) by a film in each category. Result should display the film title, category name and total revenue.

eg.

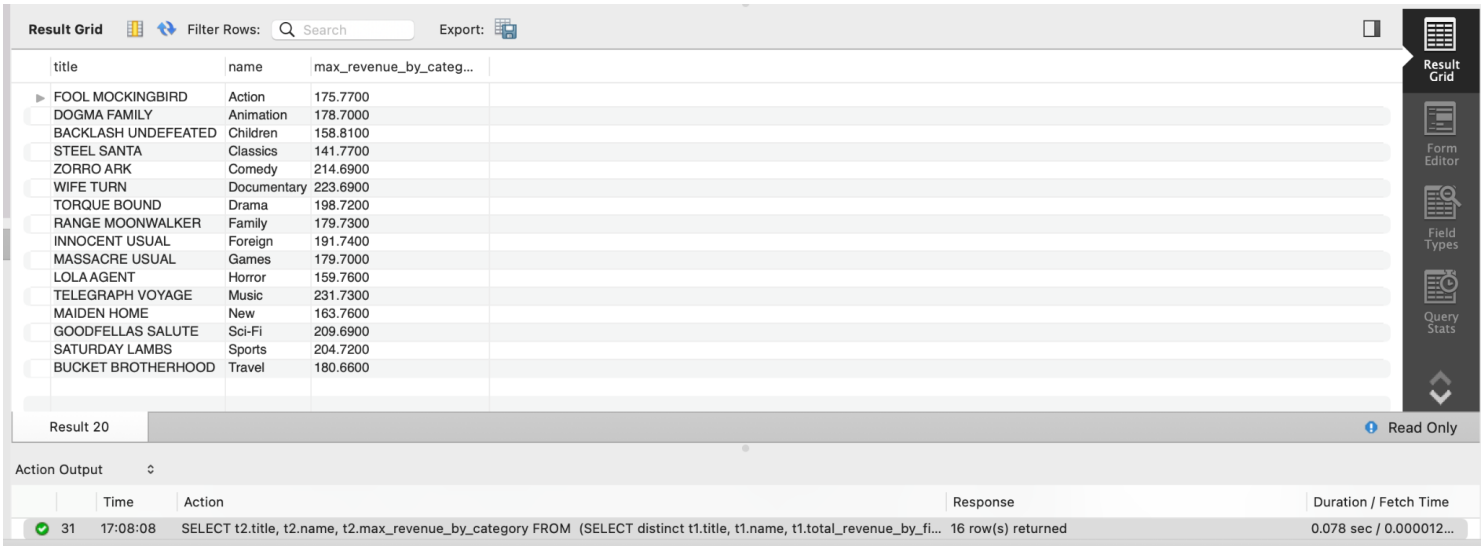
"FOOL MOCKINGBIRD"	"Action"	175.77
"DOGMA FAMILY"	"Animation"	178.7
"BACKLASH UNDEFEATED"	"Children"	158.81

Solution:

Query:

```
SELECT t2.title, t2.name, t2.max_revenue_by_category
FROM
	(SELECT distinct t1.title, t1.name, t1.total_revenue_by_film,
MAX(t1.total_revenue_by_film) OVER(PARTITION BY t1.name) AS
max_revenue_by_category
FROM
		(SELECT f.title, c.name,
		SUM(p.amount) AS total_revenue_by_film
		FROM film AS f
		INNER JOIN film_category AS fc
		ON f.film_id = fc.film_id
		INNER JOIN category AS c
		ON fc.category_id = c.category_id
		INNER JOIN inventory AS i
		ON i.film_id = f.film_id
		INNER JOIN rental AS r
		ON r.inventory_id = i.inventory_id
		INNER JOIN payment AS p
		ON p.rental_id = r.rental_id
		GROUP BY f.title, c.name) AS t1
	) AS t2
WHERE max_revenue_by_category=t2.total_revenue_by_film;
```

Output:



Result Grid Filter Rows: Search Export:

title	name	max_revenue_by_categ...
FOOL MOCKINGBIRD	Action	175.7700
DOGMA FAMILY	Animation	178.7000
BACKLASH UNDEFEATED	Children	158.8100
STEEL SANTA	Classics	141.7700
ZORRO ARK	Comedy	214.6900
WIFE TURN	Documentary	223.6900
TORQUE BOUND	Drama	198.7200
RANGE MOONWALKER	Family	179.7300
INNOCENT USUAL	Foreign	191.7400
MASSACRE USUAL	Games	179.7000
LOLA AGENT	Horror	159.7600
TELEGRAPH VOYAGE	Music	231.7300
MAIDEN HOME	New	163.7600
GOODFELLAS SALUTE	Sci-Fi	209.6900
SATURDAY LAMBS	Sports	204.7200
BUCKET BROTHERHOOD	Travel	180.6600

Result 20 Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
31	17:08:08	SELECT t2.title, t2.name, t2.max_revenue_by_category FROM (SELECT distinct t1.title, t1.name, t1.total_revenue_by_fi...	16 row(s) returned	0.078 sec / 0.000012...

Q9. Modify the table "rental" to be partitioned using PARTITION command based on 'rental_date' in below intervals:

<2005

between 2005–2010

between 2011–2015

between 2016–2020

>2020 - Partitions are created yearly

Solution:

Query:

```
ALTER TABLE rental
PARTITION BY RANGE (YEAR(rental_date))
(
PARTITION p1_less_than_2005 VALUES LESS THAN (2005),
PARTITION p2_between_2005_2010 VALUES LESS THAN (2011),
PARTITION p3_between_2011_2015 VALUES LESS THAN (2016),
PARTITION p4_between_2016_2020 VALUES LESS THAN (2021),
PARTITION p5_greater_than_2020 VALUES LESS THAN (MAXVALUE)
);
```

Q10. Modify the table "film" to be partitioned using PARTITION command based on ‘rating’ from below list. Further apply hash sub-partitioning based on ‘film_id’ into 4 sub-partitions.

partition_1 - "R"
partition_2 - "PG-13", "PG"
partition_3 - "G", "NC-17"

Solution:
Query:

```
ALTER TABLE film
PARTITION BY LIST (rating)
SUBPARTITION BY HASH (film_id) SUBPARTITIONS 4 (
PARTITION partition_1 VALUES ('R'),
PARTITION partition_2 VALUES ('PG-13', 'PG'),
PARTITION partition_3 VALUES ('G', 'NC-17'),
);
```

Q11. Write a query to count the total number of addresses from the “address” table where the ‘postal_code’ is of the below formats. Use regular expression.

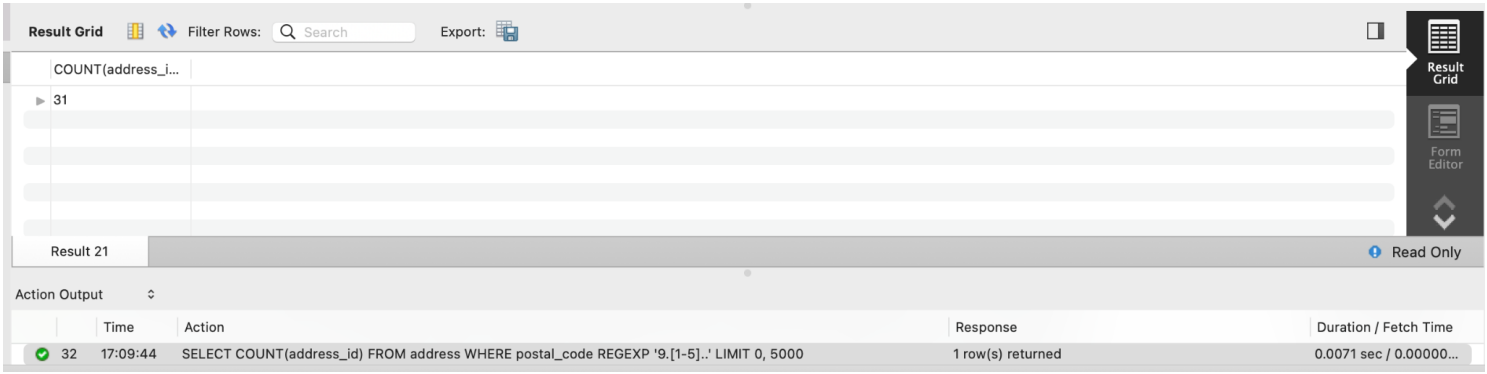
9*1, 9*2**, 9*3**, 9*4**, 9*5****

eg. postal codes - 91522, 80100, 92712, 60423, 91111, 9211
result - 2

Solution:
Query:

```
SELECT COUNT(address_id)
FROM address
WHERE postal_code REGEXP '9.[1-5]..';
```

Output:



Q12. Write a query to create a materialized view from the "payment" table where 'amount' is between(inclusive) $5 to $8. The view should manually refresh on demand. Also write a query to manually refresh the created materialized view.

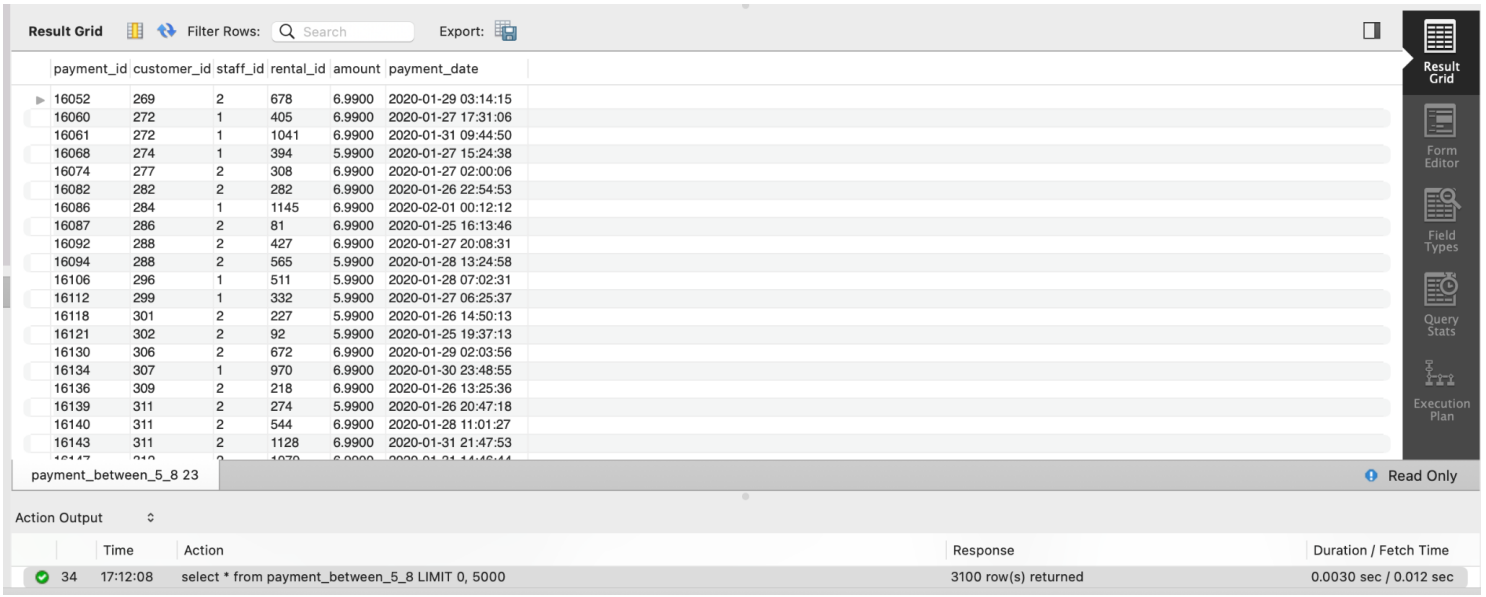
Solution:

Query:

```
CREATE VIEW payment_between_5_8 AS
SELECT *
FROM payment
WHERE amount BETWEEN 5 AND 8;

delimiter $$;
CREATE EVENT refresh_payment_between_5_8
ON SCHEDULE EVERY 1 DAY
DO
BEGIN
  CREATE OR REPLACE VIEW payment_between_5_8 AS
  SELECT *
  FROM payment
  WHERE amount BETWEEN 5 AND 8;
END$$;
delimiter ;
```

Output:



Result Grid | Filter Rows: Search | Export:

payment_id	customer_id	staff_id	rental_id	amount	payment_date
16052	269	2	678	6.9900	2020-01-29 03:14:15
16060	272	1	405	6.9900	2020-01-27 17:31:06
16061	272	1	1041	6.9900	2020-01-31 09:44:50
16068	274	1	394	5.9900	2020-01-27 15:24:38
16074	277	2	308	6.9900	2020-01-27 02:00:06
16082	282	2	282	6.9900	2020-01-26 22:54:53
16086	284	1	1145	6.9900	2020-02-01 00:12:12
16087	286	2	81	6.9900	2020-01-25 16:13:46
16092	288	2	427	6.9900	2020-01-27 20:08:31
16094	288	2	565	5.9900	2020-01-28 13:24:58
16106	296	1	511	5.9900	2020-01-28 07:02:31
16112	299	1	332	5.9900	2020-01-27 06:25:37
16118	301	2	227	5.9900	2020-01-26 14:50:13
16121	302	2	92	5.9900	2020-01-25 19:37:13
16130	306	2	672	6.9900	2020-01-29 02:03:56
16134	307	1	970	6.9900	2020-01-30 23:48:55
16136	309	2	218	6.9900	2020-01-26 13:25:36
16139	311	2	274	5.9900	2020-01-26 20:47:18
16140	311	2	544	6.9900	2020-01-28 11:01:27
16143	311	2	1128	6.9900	2020-01-31 21:47:53

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

payment_between_5_8 23 | Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
34	17:12:08	select * from payment_between_5_8 LIMIT 0, 5000	3100 row(s) returned	0.0030 sec / 0.012 sec

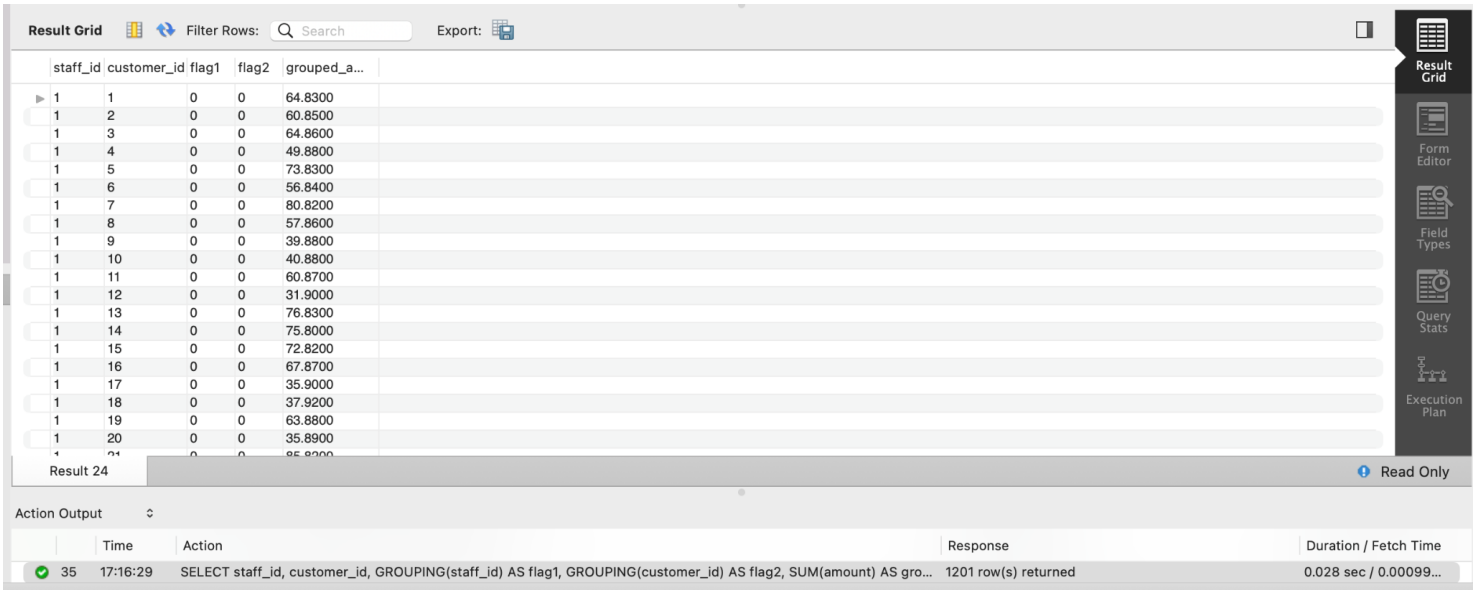
Q13. Write a query to list down the total sales of each staff with each customer from the 'payment' table. In the same result, list down the total sales of each staff i.e. sum of sales from all customers for a particular staff. Use the ROLLUP command. Also use GROUPING command to indicate null values.

Solution:

Query:

```
SELECT staff_id, customer_id, GROUPING(staff_id) AS flag1,
GROUPING(customer_id) AS flag2, SUM(amount) AS grouped_amt
FROM payment
GROUP BY staff_id, customer_id
WITH ROLLUP;
```

Output:



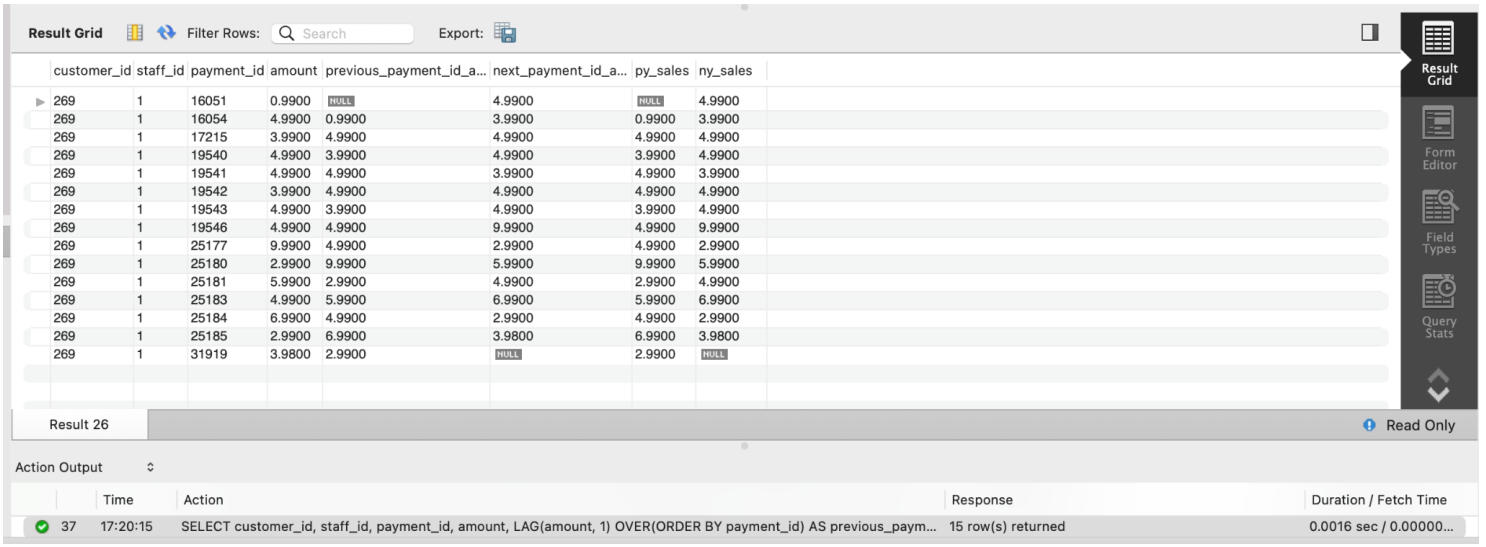
Q.14 Write a single query to display the customer_id, staff_id, payment_id, amount, amount on immediately previous payment_id, amount on immediately next payment_id ny_sales for the payments from customer_id '269' to staff_id '1'.

Solution:

Query:

```
SELECT customer_id, staff_id, payment_id, amount,
LAG(amount, 1) OVER(ORDER BY payment_id) AS previous_payment_id_amt,
LEAD(amount, 1) OVER(ORDER BY payment_id) AS next_payment_id_amt,
LAG(amount, 1) OVER(PARTITION BY customer_id, staff_id ORDER BY
payment_id) AS py_sales,
LEAD(amount, 1) OVER(PARTITION BY customer_id, staff_id ORDER BY
payment_id) AS ny_sales
FROM payment
WHERE customer_id=269 and staff_id=1;
```

Output:



Action Output

	Time	Action	Response	Duration / Fetch Time
37	17:20:15	SELECT customer_id, staff_id, payment_id, amount, LAG(amount, 1) OVER(ORDER BY payment_id) AS previous_paym...	15 row(s) returned	0.0016 sec / 0.00000...