Peer Learning Documentation

Question 1: Write a query that gives an overview of how many films have replacement costs in the following cost ranges.

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

My Solution:-

```
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;
```

My Approach:-

1. Here in this particular question we have been asked to give the overview of the films where the replacement_cost is in the given range categorizing it into low,medium and high.

2. So I have used a window function where I have used a CASE statement for filtering the data according to the given range of replacement cost and I categorize those filtered data for each case statement as low,medium and high.

3. Logic in the case statement is if the particular replacement_cost is within the specified range for the particular category then returning 1 else 0 which we are summing up using SUM() function which results in a total number of films in that particular range .

Atul’s Solution :-

```
SELECT
SUM(
CASE
WHEN replacement_cost between 9.99 and 19.99  THEN 1 ELSE 0 END
)  AS 'no_of_low' ,
SUM(
CASE
WHEN replacement_cost between 20.00 and 24.99  THEN 1 ELSE 0 END
)  AS 'no_of_medium' ,
SUM(
CASE
WHEN replacement_cost between 25.00 and 29.99  THEN 1 ELSE 0 END
)  AS 'no_of_high'
FROM film
```

Differences :

1. His approach is similar to mine. But he is using different name for aliasing.

Akshay’s Solution :-

```
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;
```

Differences :

1. His approach is similar to mine

Question 2: 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'.

"STAR OPERATION" "Sports" 181

"JACKET FRISCO" "Drama" 181

My Solution:-

```
SELECT    F.title,   C.name,   F.length
FROM   film  F  INNER JOIN  film_category  FC
ON     F.film_id = FC.film_id
INNER JOIN  CATEGORY C
ON C.category_id=FC.category_id
WHERE C.name='Sports'  OR  C.name='Drama'
ORDER BY  F.length DESC;
```

My Approach:-

1. Here we have to create a list of the film titles including their film title, film length and film category name ordered descendingly by the film length for the category “sports” and “drama”.
2. So for that I have joined three tables i.e., film, film_category, category and extracted the film_title, film_length and film_category by applying the filter for category name to be only “Sports” and “Drama” ordering by the length of the film in descending order.

Atul’s Solution :-

```
SELECT   title  AS   'Film Title' ,
c.name  AS   'Film Category',
length_   AS   'Film Length'
FROM   film f   JOIN film_category fc
ON    f.film_id = fc.film_id
JOIN  category c
ON   fc.category_id = c.category_id
WHERE    c.name = 'Drama'  or   c.name='Sports'
ORDER  BY  f.length_ DESC ;
```

Differences :

1. His approach is similar to mine. But he is using different name for aliasing the required column.

Akshay's Solution :-

```
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;
```

Differences :

1. His approach is similar to mine.

Question 3: Write a query to create a list of the addresses that are not associated to any customer.

Solution:-

```
SELECT  address_id ,  address
FROM address
WHERE address_id NOT IN (SELECT address_id FROM customer);
```

My Approach:-

1. We have been asked to create a list of the addresses that are not associated with any customer.
2. So For that I have used two tables i.e., address and customer to extract address_id, address, district, city_id and customer_id.
3. I have used left join above to retrieve all the data from the address table irrespective of whether it is present in the customer table.

   So whichever customer_id is not associated with any address will result in null values which I am using the where clause to filter out the data where customer_id is null which will give me all the addresses which are not related to any customer.

Atul's Solution :-

```
SELECT a.address_id ,
a.address
FROM customer c
RIGHT JOIN address a
ON c.address_id = a.address_id
WHERE c.customer_id IS NULL
```

Differences :

1. He is using JOIN between customer and address table . but I am using subqueries.

Akshay's Solution :-

```
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;
```

Differences :

1. He is using JOIN between customer and address table . but I am using subqueries.

Question 4:

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

My Solution:

```
 SELECT  concat(CO.country,'    ' ,CI.city  ) AS country_city ,SUM(P.amount)
 FROM country CO,city CI ,address A,customer C ,payment P
 WHERE   CO.country_id=CI.country_id   AND
CI.city_id=A.city_id     AND
 A.address_id=C.address_id    AND
C.customer_id=P.customer_id
GROUP BY CO.country,CI.city
ORDER BY SUM(P.amount) DESC;
```

Approach:-

1. Here we need to create a list of the revenue based on country, city according to the decreasing amount of revenue.
2. So for getting the desired result I have joined 5 tables i.e., country, city, address customer and payment to fetch country name, city name which I am concatenating using CONCAT() function.
3. Also calculated the revenue by using GROUP BY function which is calculating the sum of amount partitioning by Country,City and then ordering revenue by descending order .

Atul’s Solution :-

```
 SELECT concat(cty.country,' , ' ,ct.city  ) AS country_city ,
SUM(py.amount)  AS 'List of the revenue'
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 ct
ON a.city_id = ct.city_id
JOIN country cty
ON cty.country_id = ct.country_id
JOIN payment py
ON r.rental_id = py.rental_id
GROUP BY cty.country, ct.city;
```

Differences :

1. He is using the ANSI method to join the table but I am using THETA method.

Akshay’s Solution :-

```
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;
```

Differences :

1. His approach is similar to mine.

Question 5:

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
```

My Solution:-

```
 SELECT C.staff_id, ROUND(AVG(S),2)
 FROM
(SELECT S.staff_id,sum(P.amount)  AS  S
FROM staff s, payment P
WHERE S.staff_id=P.staff_id
GROUP BY S.staff_id,P.customer_id)  C
GROUP BY C.staff_id;
```

Approach:-

1. Here we are asked to create a list with the average of sales per customer with each staff id.
2. Here each customer has multiple transaction with particular staff id, so first we need to find total amount of transaction each customer had with particular staff id, so I am calculating that In the derived table (C) above using GROUP BY function which is calculating the sum of amount partitioning by staff_id,customer_id.
3. Now using the derived table I am extracting staff_id and calculating average of the total_amount per customer calculated in the derived table to find out the average amount of sales each staff had.

Atul’s Solution :-

```
SELECT staff_id, ROUND(AVG(sum_amount), 2) AS sales_amount
FROM (
SELECT DISTINCT staff_id, customer_id,
SUM(amount) OVER(PARTITION BY staff_id,customer_id) AS sum_amount
FROM payment
)  a
GROUP BY staff_id
```

Differences :

1. His approach is similar to mine. But in derived queries I am using GROUP BY but he is using WINDOW FUNCTION.

Akshay’s Solution :-

```
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;
```

Differences :

1. His approach is similar to mine.

Question 6: Write a query that shows average daily revenue of all Sundays.

Solution:-

```
 SELECT AVG(D.AMOUNT)   AS   'Average daily Revenue of all Sundays'
 from
(SELECT C.DATE_ONLY ,SUM(C.amount) AS AMOUNT
FROM
(SELECT date(payment_date) AS DATE_ONLY, amount
FROM payment
WHERE WEEKDAY(payment_date)=6) C
GROUP BY C.DATE_ONLY ) D;
```

Approach:-

1. We have been asked to show the average daily revenue of all Sundays.
2. So for that I have used a single payment table wherein in the inner derived table I am extracting the payment_date and amount by filtering the data where the date is of sunday by using WEEKDAY() function in where clause.
3. In each sunday their can be multiple transaction so in order to find out the average, first we need to calculate the total of all the transaction in every sunday.So in outer derived table I am calculating the sum of amount grouping by the date (obtain from inner derived table by filtering out on the basis of sunday) which will give me the total transaction in all sundays.
4. Now that I have got the total amount ,I can find the average of that total amount which will give me the desired result that is the average daily revenue of all sundays.

Atul’s Solution :-

```
SELECT ROUND(avg(am),2) AS 'average daily revenue of all sundays'
FROM (
SELECT SUM(amount) as am
FROM payment
WHERE WEEKDAY(DATE(payment_date)) = 6
GROUP BY DATE(payment_date)
) a
```

Differences :

1. His approach is similar to mine.

Akshay’s Solution :-

```
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;
```

Differences :

1. His approach is similar to mine. He is using DAYNAME keyword but I am using the WEEKDAY keyword.

Question 7: 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:-

```
SELECT B.district,AVG(B.total_payment_per_customer)
FROM
(SELECT A.district,C.customer_id ,SUM(P.amount) AS      total_payment_per_customer
FROM address A , customer C , payment P
where A.address_id=C.address_id AND C.customer_id=P.customer_id
GROUP BY A.district,C.customer_id
) B
GROUP BY B.district;
```

Approach:-

1. Here we need to find out the average amount of all customer spent in total based on each district
2. For that I have used three tables i.e., customer, address and payment and extracted customer_id, district and the total amount for each customer as each customer have multiple transaction in each district
3. Now making above as derived table I extracted district and Average of total_sum calculated for each customer in the derived table partitioning by the district.So this way I am getting the district and the average of customers total spent grouped by each district

Atul’s Solution :-

```
WITH t1 AS (
SELECT DISTINCT a.district  ,
SUM(py.amount) OVER(PARTITION BY a.district )  AS sm
FROM address a
JOIN customer c
ON a.address_id = c.address_id
JOIN payment py
ON py.customer_id = c.customer_id
),
t2 AS (
SELECT a.district ,
COUNT(c.customer_id) AS cnt
FROM  address a
JOIN customer c
ON a.address_id = c.address_id
GROUP BY a.district
)
SELECT t1.district , t1.sm/t2.cnt AS 'average_amount'
FROM t1,t2
WHERE t1.district = t2.district
```

Differences :

1. He has created two CTE (COMMON TABLE EXPRESSION) t1 and t2 using the ‘WITH’ keyword.
2. T1 table is created for total amount of customers spent in total (customer life-time value) in each district and t2 table is used for count of districts .
3. Using both the tables he found the desired result.

Akshay’s Solution :-

```
 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;
```

Differences :

1. His approach is similar to mine.

Question 8:

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:-

```
SELECT N.title , N.CATEG_NAME , N.total_collection
 From
 (SELECT M.title, M.CATEG_NAME, M.total_collection,
 MAX(total_collection)
 OVER(PARTITION BY M.CATEG_NAME) MAXIMUM_OVER_CATEG
 FROM
 (SELECT   F.title , C.name  AS  CATEG_NAME , sum(P.amount)  AS  total_collection
 from film F,  inventory I ,  rental R,  payment P,  film_category FC,  category C
 where  F.film_id=I.film_id
 AND   I.inventory_id=R.inventory_id
 AND   R.rental_id=P.rental_id
 AND   F.film_id=FC.film_id
 AND   FC.category_id=C.category_id
 GROUP  BY  F.title,C.name )  M
 )  N
WHERE N.total_collection=N.MAXIMUM_OVER_CATEG;
```

Approach:-

1. Here we need to list out the film_title and highest overall revenue collected sum of amount per title by a film for each category For this I have used 6 tables i.e., film, film_category ,category, inventory, rental, payment and from that I have extracted film_title, film category and sum of amount partition by film_title, so this will give me the overall revenue for each film title.
2. Now from inner derived table I have the data of film_title, film category and overall revenue generated from each film.So in outer derived table I extracted the film_title, category and Maximum of overall revenue for each category by using the window function for finding MAX of overall_revenue partition by category.
3. Now from the outer derived table I extracted the film_title,category name and the max revenue where my max_revenue= overall_revenue. This way I am getting those film titles who have generated the highest revenue in each category.

Atul’s Solution :-

```
SELECT a.title ,
a.name , a.Total_Revenue
FROM (
SELECT f.title ,
ct.name ,
SUM(py.amount) as 'Total_Revenue' ,
DENSE_RANK() OVER(PARTITION BY ct.name ORDER BY sum(py.amount) DESC ) AS rn
FROM film f
JOIN film_category fc
ON f.film_id = fc.film_id
JOIN category ct
ON ct.category_id = fc.category_id
JOIN inventory inv
ON inv.film_id = f.film_id
JOIN rental r
ON r.inventory_id = inv.inventory_id
JOIN payment py
ON py.rental_id = r.rental_id
GROUP BY  f.title , ct.name
) a
WHERE a.rn<=1
```

Differences :

1. He is using the ANSI method to join the table but I am using THETA method.
2. He has creating another column using WINDOW FUNCTION and ORDER BY total sum of amount , which stores DENSE_RANK for every film category and taking only those row , which are having DENSE RANK =1;

Akshay's Solution :-

```
 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;
```

Differences :

1. His approach is similar to mine.

Question 9:

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:-

```
ALTER TABLE rental
 PARTITION BY RANGE(YEAR(rental_date))
 (
 PARTITION rental_less_than_2005 VALUES LESS THAN (2005),
 PARTITION rental_between_2005_2010 VALUES LESS THAN (2011),
 PARTITION rental_between_2011_2015 VALUES LESS THAN (2016),
 PARTITION rental_between_2016_2020 VALUES LESS THAN (2021),
 PARTITION rental_greater_than_2020 VALUES LESS THAN MAXVALUE
  );
```

Approach:-

1. Here In this particular question first I have to drop the foreign keys as partitioning was not supporting the foreign key constraints in MYSQL as it can change the underlying structure of the table.
2. Next I partitioned the rental table by rental_date with the specified range of years.
3. First Partition “rental_less_than_2005” will contain those values where the year is less than 2005.
4. Second Partition “rental_between_2005_2010” will contain those values where the year is between 2005 and 2010.
5. Third Partition “rental_between_2011_2015” will contain those values where the year is between 2011 and 2015.
6. Fourth Partition “rental_between_2016_2020” will contain those values where the year is between 2016 and 2020.
7. And everything after 2020 will be stored in the partition “rental_greater_than_2020”.

Atul’s Solution :-

```
ALTER TABLE rental
PARTITION BY RANGE(YEAR(rental_date))
(
PARTITION rent_lt_2005 VALUES LESS THAN (2005),
PARTITION rent_between_2005_2010 VALUES LESS THAN (2011),
PARTITION rent_between_2011_2015 VALUES LESS THAN (2016),
PARTITION rent_between_2016_2020 VALUES LESS THAN (2021),
PARTITION rent_gt_2020 VALUES LESS THAN MAXVALUE
);
```

Differences :

1. His approach is similar to mine.But the partition names are different.

Akshay's Solution :-

```
 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)
);
```

Differences :

1. His approach is similar to mine. But the partition names are different.

Question 10:

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:-

```
ALTER TABLE film
PARTITION BY LIST(rating)
SUBPARTITION BY HASH(film_id) SUBPARTITIONS 4
(
PARTITION  P_R  values('R'),
PARTITION  P_gs  values('PG-13', 'PG'),
PARTITION  P_GNC  values('G', 'NC-17')
);
```

Approach:-

1. Here first I have used PARTITION BY LIST based on rating.
2. Then I have used SUBPARTITION BY HASH based on fil .

Atul's Solution :-

```
ALTER TABLE film
PARTITION BY LIST(rating)
SUBPARTITION BY HASH(film_id) SUBPARTITIONS 4
(
PARTITION PR values('R'),
PARTITION Pgs values('PG-13', 'PG'),
PARTITION GNC values('G', 'NC-17')
 );
```

Differences :

1. His approach is similar to mine.

Akshay's Solution :-

```
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'),
);
```

Differences :

1. His approach is similar to mine.

QUESTION 11:

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 expressions.

- ***91**, 92**, 93**, 94**, 9*5*****

Solution:-

```
SELECT count(postal_code) AS 'No_of_postal_code'
FROM address
WHERE postal_code REGEXP '^9[0-9][1-5][0-9]{2}$';
```

Approach:-

1.Here I have used the address table to extract postal_code of the specific pattern by using a regular expression in the where clause to filter out the data.

2. Next, I used the Aggregate function COUNT to count the number of postal codes following the specific pattern.

Atul's Solution :-

```
SELECT count(postal_code)

FROM address
WHERE postal_code REGEXP '^9[0-9][1-5][0-9]{2}';

 -- Using regular expression to retrieve all
 – the postal code with the given pattern
```

Differences :

1. His approach is similar to mine.

Akshay's Solution :-

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

Differences :

1. I have used charet and dollar key for specifying the postal code .
2. It should start with '9' and end with a numeric value. Third character must be between 1 to 5 and 2nd , 4th , 5th character must be
3. But he has only specified that the first character must be '9' and the third character must be between 1 to 5 and 2nd, 4th ,5th characters can be any character.

Question 12:

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:-

```
  DELIMITER $$
   CREATE EVENT refresh_payment_between_5_8
-- Creating an event which refreshes the view every day.
   ON SCHEDULE EVERY 1 DAY
   DO
   BEGIN
   CREATE OR REPLACE VIEW payment_between_5_8 AS
   SELECT *
   FROM payment
   WHERE amount BETWEEN 5 AND 8;
   END$$
   DELIMITER ;

   -- APPLYING SELECT CLAUSE IN VIEW CREATED ABOVE WHICH CONTAINS THE DATA WHERE
  – AMOUNT IS BETWEEN $5 AND $8

  SELECT * FROM payment_between_5_8;
```

Approach:-

1. As materialized view was not supported by MYSQL so I have created the event name “refresh_payment_between_5_8” which is scheduled to refresh in every one day and inside that I have created the normal view which stores all the data of the payment table where amount is between $5 and $ 8.

Atul’s Solution :-

```
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 ;


SELECT * FROM payment_between_5_8;
```

Differences :

1. His approach is similar to mine

Akshay's Solution :-

```
  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 ;
```

Differences :

1. His approach is similar to mine.

Question 13:

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:-

```
SELECT  pay.staff_id, pay.customer_id,
GROUPING(pay.staff_id)  as staff,
GROUPING(pay.customer_id)  as  customer,sum(pay.amount)  as  SumOfSales
FROM  payment  pay
GROUP BY  pay.staff_id , pay.customer_id
WITH ROLLUP;
```

Approach:-

1. Here we have been asked to list down all the total sales of each staff with each customer. Also we need to list down the total sales of each staff i.e. sum of sales from all customers for a particular staff.
2. So for that I have used the payment table to extract staff_id,customer_id and applied grouping on both customer_id and staff_id to handle null values and clearly indicate the ROLLUP sum for the particular column.

Atul’s Solution :-

```
 SELECT p.staff_id,p.customer_id,
GROUPING(p.staff_id) as staff,
GROUPING(p.customer_id) as customer,sum(p.amount) as sum_of_sales
FROM payment p
GROUP BY p.staff_id,p.customer_id
WITH ROLLUP;
```

Differences :

1. His approach is similar to mine.

Akshay’s Solution :-

```
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;
```

Differences :

1. His approach is similar to mine.

QUESTION 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’.

CODE:

```
SELECT customer_id,payment_id,staff_id,
LEAD(amount) OVER(ORDER BY payment_id) next_payment,
LAG(amount) OVER(ORDER BY payment_id) previous_amount,
LEAD(amount) OVER(PARTITION BY customer_id,staff_id ORDER BY payment_id) AS next_payment_sales,
LAG(amount) OVER(PARTITION BY customer_id,staff_id ORDER BY payment_id) AS previous_payment_sales
FROM payment
WHERE customer_id=269 and staff_id=1;
```

Approach:-

1. Here we have been asked to display the customer_id, staff_id, payment_id, amount, amount on immediately previous payment_id, amount on immediately next payment_id.
2. For this we have used the payment table and used the window function LEAD to extract the next payment and LAG to extract the previous payment order by payment_id.
3. Next we need to find the ny_sales - the amount of the next payment made by the same customer to the same staff member and py sales - the amount of the previous payment made by the same customer to the same staff member.For this as well I have used lead and lag function partitioned by customer_id and staff_id.
4. Used where clause to filter out the data where customer_id=269 and staff_id=1.

Atul’s Solution :-

```
SELECT customer_id,payment_id,staff_id,
LEAD(amount) OVER(ORDER BY payment_id) next_payment,
LAG(amount) OVER(ORDER BY payment_id) previous_amount,
LEAD(amount) OVER(PARTITION BY customer_id,staff_id ORDER BY payment_id) AS
next_payment_sales,
LAG(amount) OVER(PARTITION BY customer_id,staff_id ORDER BY payment_id) AS
previous_payment_sales
FROM payment
WHERE customer_id=269 and staff_id=1;
```

Differences :

1. His approach is similar to mine.

Akshay’s Solution :-

```
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;
```

Differences :

1. His approach is similar to mine.