PEER DOCUMENT ADVANCEDSQL ASSIGNMENT

QUESTION 1:

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

MY 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
```

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;
```

MAHESH'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;
```

AKSHAY'S APPROACH :

Similar to Mine.

MAHESH'S APPROACH:

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 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 ;
```

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;

MAHESH'S 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;

AKSHAY'S APPROACH :

Askhay used an IN operator in this condition .

MAHESH'S APPROACH :

Similar Approach .

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

MY 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

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;
```

MAHESH'S SOLUTION:

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

AKSHAY'S APPROACH :

Similar Approach But He used Left Join I have used Right Join.

MAHESH'S APPROACH :

Mahesh used a subquery.

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(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
```

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;
```

MAHESH'S 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;
```

AKSHAY'S APPROACH -

Similar to Mine.

MAHESH'S APPROACH -

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 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
```

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;
```

MAHESH'S 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;
```

AKSHAY'S APPROACH :

He used group by function and I have used window functions.

MAHESH'S APPROACH:

He used group by function and I have used window functions.

QUESTION 6:

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

MY 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
```

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;
```

MAHESH'S SOLUTION:

```
        SELECT AVG(D.AMOUNT) 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;
```

Akshay's Approach :
He used the Dayname function and I have used weekday.

Mahesh's Approach:
Similar Approach.

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.

MY 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
```

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;
```

MAHESH'S 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;
```

AKSHAY'S APPROACH:

Major Difference is that I have used CTEs and his code has inline sub queries.

MAHESH'S APPROACH:

Major Difference is that I have used CTEs and his code has inline sub queries.

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

MY 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
```

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;
```

MAHESH'S SOLUTION:

```
  select * from film;
            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;
```

AKSHAY'S APPROACH :

I have used Dense Rank but he calculated the maximum for each first then compared to total revenue of each category.

MAHESH'S APPROACH:

I have used Dense Rank but he calculated the maximum for each first then compared to total revenue of each category.

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

MY 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
);
```

AKSHAY'S APPROACH: Similar to mine.

MAHESH'S APPROACH: Similar to mine.

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.

MY 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')
);
```

AKSHAY'S APPROACH: Similar to mine.

MAHESH'S APPROACH: 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****

MY SOLUTION:

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

AKSHAY'S SOLUTION:

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

Mahesh' 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.
```

AKSHAY'S APPROACH: Similar to mine.

MAHESH'S APPROACH: Similar to mine.

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.

MY 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;
```

AKSHAY'S APPROACH: Similar to mine.

MAHESH'S APPROACH: 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.

MY 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;
```

AKSHAY'S APPROACH: Similar to mine.

MAHESH'S APPROACH: 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'.

MY 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;
```

AKSHAY'S APPROACH: Similar to mine.

MAHESH'S APPROACH: Similar to mine.