Business Problems

Q1. Find different payment method and number of transactions, number of qty sold

```
CREATE VIEW diff_payment_qnty_sold AS
SELECT payment_method,
COUNT(*) AS no_payments,
SUM(quantity) AS no_qty_sold
FROM walmart
GROUP BY payment_method;

--OUTPUT:
SELECT * FROM diff_payment_qnty_sold;
```

Q2. Identify the highest-rated category in each branch, displaying the branch, category
avg ratings.

```
CREATE VIEW avg_rating_by_branch_category AS
SELECT * FROM (
SELECT branch, category,
ROUND(AVG(rating)::numeric,2) AS avg_rating,
RANK() OVER (PARTITION BY branch ORDER BY AVG(rating) DESC)
AS rnk
FROM walmart
GROUP BY 1,2
)
WHERE rnk=1;

--OUTPUT:
SELECT * FROM avg_rating_by_branch_category;
```

Q3. Identify the busiest day for each branch based on the number of transactions

```
CREATE VIEW busiest_day_by_branch AS
SELECT *
FROM (
SELECT branch,
TO_CHAR(TO_DATE(date, 'DD/MM/YY'), 'Day') as day_name,
COUNT(*) as no_transactions,
RANK() OVER(PARTITION BY branch ORDER BY COUNT(*) DESC) as rank
FROM walmart
GROUP BY 1, 2
          )
```

```
WHERE rank = 1;

--OUTPUT:
SELECT * FROM busiest_day_by_branch;
```

Q4. Calculate the total quantity of items sold per payment method. List payment_method and total_quantity.

```
CREATE VIEW payment_method_total_quan AS
SELECT payment_method,
          SUM(quantity) as no_qty_sold
FROM walmart
GROUP BY payment_method;

--OUTPUT:
SELECT * FROM payment_method_total_quan;
```

Q5. Determine the average, minimum, and maximum rating of category for each city.
list the city, average_rating, min_rating, and max_rating.

```
CREATE VIEW min_max_avg_ratings AS
SELECT city,category,
MIN(rating) as min_rating,
MAX(rating) as max_rating,
ROUND(AVG(rating)::numeric,2) as avg_rating
FROM walmart
GROUP BY 1, 2;


--OUTPUT:
SELECT * FROM min_max_avg_ratings;
```

Q6. Calculate the total profit for each category by considering total_profit as (unit_price * quantity * profit_margin).
List category and total_profit, ordered from highest to lowest profit.

```
CREATE VIEW to_rev_pft_by_category AS
SELECT category,
ROUND(SUM(total)::numeric,2) AS total_revenue,
ROUND(SUM(total * profit_margin)::numeric,2) AS profit
FROM walmart
GROUP BY 1
ORDER BY 3 DESC;
```

```
--OUTPUT:
SELECT * FROM to_rev_pft_by_category;
```

Q7. Determine the most common payment method for each Branch.
Display Branch and the preferred_payment_method.

```
CREATE VIEW preferred_payment_method AS
WITH cte AS(
SELECT branch, payment_method,
COUNT(*) as total_trans,
RANK() OVER(PARTITION BY branch ORDER BY COUNT(*) DESC) as rank
FROM walmart
GROUP BY 1, 2
)
SELECT *
FROM cte
WHERE rank = 1;

--OUTPUT:
SELECT * FROM preferred_payment_method;
```

Q8. Categorize sales into 3 group MORNING, AFTERNOON, EVENING

```
CREATE VIEW sales_in_each_shift_by_category AS
SELECT branch,
CASE
WHEN EXTRACT(HOUR FROM(time::time)) < 12 THEN 'Morning'
WHEN EXTRACT(HOUR FROM(time::time)) BETWEEN 12 AND 17 THEN 'Afternoon'
ELSE 'Evening'
END day_time,
COUNT(*)
FROM walmart
GROUP BY 1, 2
ORDER BY 1, 3 DESC;

--OUTPUT:
SELECT * FROM sales_in_each_shift_by_category;
```

Q9. Identify 5 branch with highest decrese ratio in revevenue compare to last year(current year 2023 and last year 2022)   ## rdr == last_rev-cr_rev/ls_rev*100

```
SELECT *,
EXTRACT(YEAR FROM TO_DATE(date, 'DD/MM/YY')) AS formated_date
FROM walmart;
```

```
-- 2022 sales
CREATE VIEW rev_dec_ratio AS
WITH revenue_2022 AS(
SELECT branch,
SUM(total) as revenue
FROM walmart
WHERE EXTRACT(YEAR FROM TO_DATE(date, 'DD/MM/YY')) = 2022
GROUP BY 1
),
revenue_2023 AS (
SELECT branch,
SUM(total) as revenue
FROM walmart
WHERE EXTRACT(YEAR FROM TO_DATE(date, 'DD/MM/YY')) = 2023
GROUP BY 1
)

SELECT ls.branch,
ls.revenue AS last_year_revenue,
cs.revenue AS cr_year_revenue,
ROUND(
(ls.revenue - cs.revenue)::numeric/ls.revenue::numeric * 100,2) AS rev_dec_ratio
FROM revenue_2022 AS ls
JOIN revenue_2023 AS cs
ON ls.branch = cs.branch
WHERE ls.revenue > cs.revenue
ORDER BY 4 DESC
LIMIT 5;


--OUTPUT:
SELECT * FROM rev_dec_ratio;
```