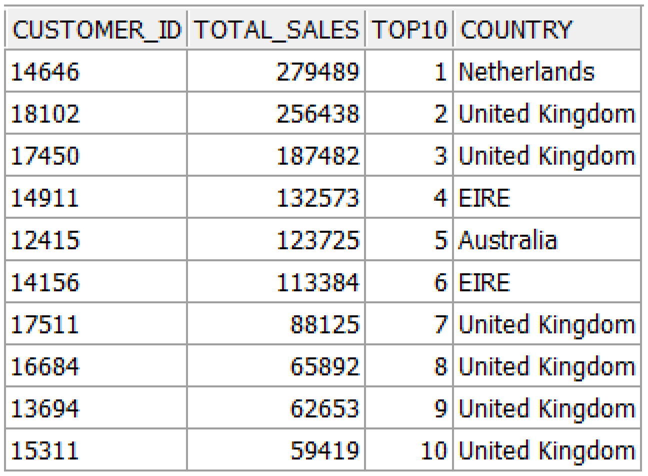
Q1 – 1-

```
WITH customer_sales_cte AS (
        SELECT DISTINCT CUSTOMER_ID, COUNTRY ,ROUND(SUM(QUANTITY * PRICE) OVER
    (PARTITION BY CUSTOMER_ID)) AS TOTAL_SALES
        FROM tableRetail
    ),
    customer_ranks_cte AS (
        SELECT CUSTOMER_ID,COUNTRY ,TOTAL_SALES, RANK() OVER (ORDER BY TOTAL_SALES DESC)
    AS TOP10
        FROM customer_sales_cte
    )
    SELECT CUSTOMER_ID, TOTAL_SALES , TOP10 ,COUNTRY
    FROM customer_ranks_cte
    WHERE TOP10 <= 10;
```

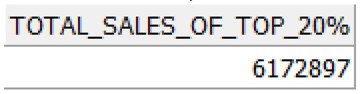


CUSTOMER_ID	TOTAL_SALES	TOP10	COUNTRY
14646	279489	1	Netherlands
18102	256438	2	United Kingdom
17450	187482	3	United Kingdom
14911	132573	4	EIRE
12415	123725	5	Australia
14156	113384	6	EIRE
17511	88125	7	United Kingdom
16684	65892	8	United Kingdom
13694	62653	9	United Kingdom
15311	59419	10	United Kingdom

The business meaning behind this query is to identify the top 10 customers by total sales in a given country. This can be useful for marketing campaigns, customer retention efforts, and understanding which customers contribute the most revenue to the business

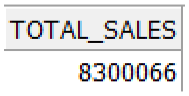
2-

```
WITH customer_sales_cte AS (
   SELECT DISTINCT CUSTOMER_ID, ROUND(SUM(QUANTITY * PRICE) OVER (PARTITION BY
CUSTOMER_ID)) AS TOTAL_SALES
   FROM tableRetail
),
customer_ranks_cte AS (
   SELECT TOTAL_SALES , CUSTOMER_ID ,  ROUND((PERCENT_RANK() OVER (ORDER BY TOTAL_SALES
DESC) * 100)) AS A
   FROM customer_sales_cte
)
SELECT SUM(TOTAL_SALES) AS "TOTAL_SALES_OF_TOP_20%"
FROM customer_ranks_cte
WHERE A <= 20;
```



TOTAL_SALES_OF_TOP_20%
6172897

```
SELECT ROUND(SUM(QUANTITY * PRICE)) AS TOTAL_SALES FROM tableRetail
```



TOTAL_SALES
8300066

(6172897/8300066) * 100 = 74,37%

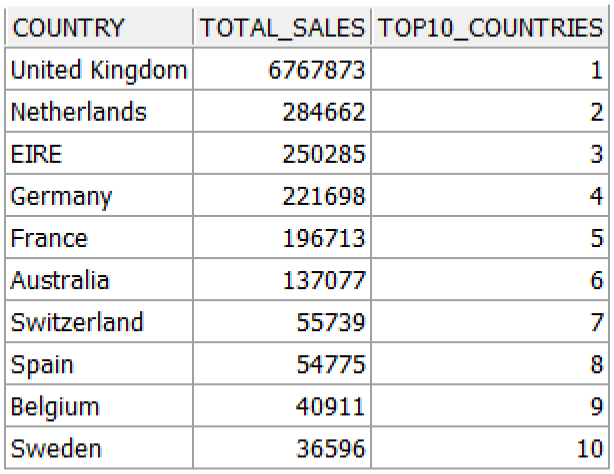
THEN TOP 20 % OF CUSTOMERS MADE 74,37 % OF SALES

For the first SQL query The business meaning behind this query is to identify the total sales generated by the top 20% of customers in the "tableRetail" dataset. This can be useful for understanding the revenue contribution of the most important customers to the business.

The second SQL query simply calculates the total sales for all transactions in the "tableRetail" dataset. The result is rounded and displayed as "TOTAL_SALES". The business meaning behind this query is to obtain the total sales amount of all transactions in the "tableRetail" dataset. This information can be used to calculate revenue and other important financial metrics. Like 20/80 rule

3 -

```
WITH COUNTRY_sales_cte AS (
   SELECT DISTINCT  COUNTRY ,ROUND(SUM(QUANTITY * PRICE) OVER (PARTITION BY COUNTRY)) AS
TOTAL_SALES
   FROM tableRetail
),
COUNTRY_ranks_cte AS (
   SELECT COUNTRY ,TOTAL_SALES, RANK() OVER (ORDER BY TOTAL_SALES DESC) AS TOP10_COUNTRIES
   FROM COUNTRY_sales_cte
)
SELECT  COUNTRY , TOTAL_SALES , TOP10_COUNTRIES
FROM COUNTRY_ranks_cte
WHERE TOP10_COUNTRIES <= 10;
```

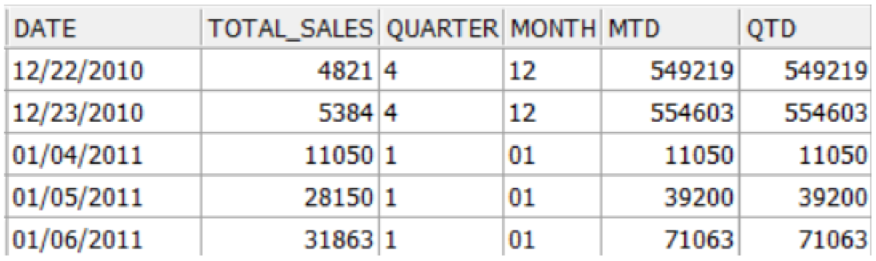


COUNTRY	TOTAL_SALES	TOP10_COUNTRIES
United Kingdom	6767873	1
Netherlands	284662	2
EIRE	250285	3
Germany	221698	4
France	196713	5
Australia	137077	6
Switzerland	55739	7
Spain	54775	8
Belgium	40911	9
Sweden	36596	10

The business meaning behind this query is to identify the top 10 countries by total sales in the "tableRetail" dataset. This can be useful for understanding which countries contribute the most revenue to the business and for planning international marketing campaigns or expansion strategies.

4-

```
WITH sales_data AS (
  SELECT
    TO_CHAR(TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'), 'MM/DD/YYYY') AS DATEE,
    ROUND(SUM(QUANTITY * PRICE)) AS TOTAL
  FROM   tableRetail
  GROUP BY
    TO_CHAR(TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'), 'MM/DD/YYYY') )
SELECT
  DATEE AS "DATE",
  TOTAL AS "TOTAL_SALES",
  TO_CHAR(TO_DATE(DATEE, 'MM/DD/YYYY'), 'Q') AS "QUARTER",
  TO_CHAR(TO_DATE(DATEE, 'MM/DD/YYYY'), 'MM') AS "MONTH",
  SUM(TOTAL) OVER (PARTITION BY TO_CHAR(TO_DATE(DATEE, 'MM/DD/YYYY'), 'MM') ORDER BY
TO_DATE(DATEE, 'MM/DD/YYYY')) AS "MTD",
  SUM(TOTAL) OVER (PARTITION BY TO_CHAR(TO_DATE(DATEE, 'MM/DD/YYYY'), 'Q') ORDER BY
TO_DATE(DATEE, 'MM/DD/YYYY')) AS "QTD"
FROM   sales_data
ORDER BY
  TO_DATE(DATEE, 'MM/DD/YYYY') ASC;
```

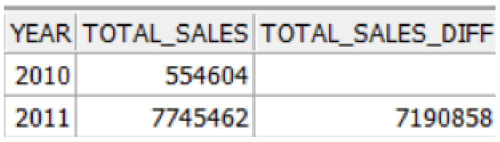


DATE	TOTAL_SALES	QUARTER	MONTH	MTD	QTD
12/22/2010	4821	4	12	549219	549219
12/23/2010	5384	4	12	554603	554603
01/04/2011	11050	1	01	11050	11050
01/05/2011	28150	1	01	39200	39200
01/06/2011	31863	1	01	71063	71063

The business meaning behind this query is to retrieve the sales data from the "tableRetail" dataset and provide an analysis of the data by date, month, and quarter. This query can be useful for tracking sales performance over time, identifying trends in sales, and for making business decisions based on the sales data. For example, the business may use this data to adjust their inventory levels, staffing, or marketing strategies.

5-

```
SELECT
   EXTRACT(YEAR FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')) AS Year,
   ROUND(SUM(QUANTITY * PRICE)) AS Total_Sales,
   ROUND(SUM(QUANTITY * PRICE) - LAG(SUM(QUANTITY * PRICE)) OVER (ORDER BY EXTRACT(YEAR
FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')))) AS Total_Sales_Diff
FROM
   tableRetail
GROUP BY
   EXTRACT(YEAR FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'))
ORDER BY
   EXTRACT(YEAR FROM TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI'));
```



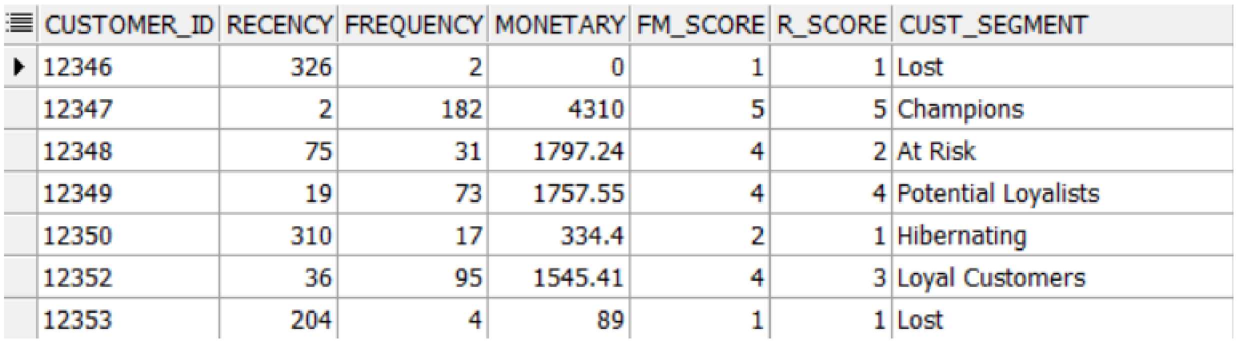
YEAR	TOTAL_SALES	TOTAL_SALES_DIFF
2010	554604	
2011	7745462	7190858

The business meaning behind this query is to analyze the sales data from the "tableRetail" dataset on an annual basis and calculate the total sales and the year-over-year difference in sales. This query can be useful for monitoring the performance of the business over time, identifying growth or decline in sales, and for making business decisions based on the sales data. For example, if the year-over-year difference is negative, the business may need to adjust its sales and marketing strategies to drive growth, or if the year-over-year difference is positive, the business may want to continue with its current strategies to maintain growth.

Q2 –

```
select CUSTOMER_ID,
  recency,
  frequency,
  monetary,
  fm_score ,
  r_score
  , Case
when r_score >= 5 and fm_score >= 5
or   r_score >= 5 and fm_score =4
or   r_score = 4  and fm_score >= 5 then 'Champions'

when r_score >= 5 and fm_score = 2
or   r_score = 4 and fm_score = 2
or   r_score = 3 and fm_score = 3
or   r_score = 4  and fm_score >= 3 then 'Potential Loyalists'

when r_score >= 5 and fm_score = 3
or   r_score = 4 and fm_score = 4
or   r_score = 3 and fm_score >= 5
or   r_score = 3  and fm_score >= 4 then 'Loyal Customers'

when r_score >= 5 and fm_score = 1 then 'Recent Customers'

when r_score = 4 and fm_score = 1
or   r_score = 3  and fm_score = 1 then 'Promising'

when r_score = 3 and fm_score = 2
or   r_score = 2 and fm_score = 3
or   r_score = 2  and fm_score = 2 then 'Customers Needing Attention'

when r_score = 2 and fm_score >= 5
or   r_score = 2 and fm_score = 4
or   r_score = 1  and fm_score = 3 then 'At Risk'

when r_score = 1 and fm_score >= 5
or   r_score = 1  and fm_score = 4 then 'Cant Lose Them'

when r_score = 1 and fm_score = 2
or   r_score = 2  and fm_score = 1 then 'Hibernating'

when r_score = 1  and fm_score <= 1 then 'Lost'
End  cust_segment
from
```

```
(
SELECT   CUSTOMER_ID ,  recency   , frequency,
   monetary,
   NTILE(5) OVER (ORDER BY recency desc) AS r_score ,
   NTILE(5) OVER (ORDER BY (frequency + monetary)/2 )  AS fm_score
from (

SELECT DISTINCT
   CUSTOMER_ID,
   FIRST_VALUE(DAYS_BETWEEN_INVOICES IGNORE NULLS) OVER (PARTITION BY CUSTOMER_ID ORDER
BY DAYS_BETWEEN_INVOICES ASC) AS recency,
   frequency,
   monetary

FROM
   (
      SELECT DISTINCT
         CUSTOMER_ID,
         CEIL(FIRST_VALUE(TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI')) OVER (ORDER BY
TO_DATE(INVOICEDATE, 'MM/DD/YYYY HH24:MI') DESC) - TO_DATE(INVOICEDATE, 'MM/DD/YYYY
HH24:MI')) AS DAYS_BETWEEN_INVOICES,
         SUM(price *quantity) OVER (PARTITION BY CUSTOMER_ID) AS monetary,
         COUNT(INVOICE) OVER (PARTITION BY CUSTOMER_ID) AS frequency
      FROM
         tableRetail
      ORDER BY
         CUSTOMER_ID )
)
ORDER BY
   CUSTOMER_ID )
```

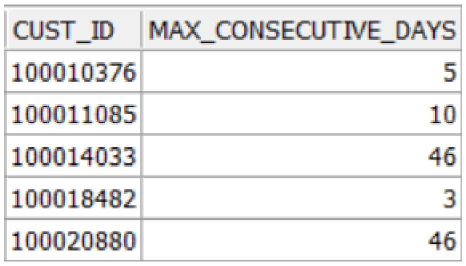


CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	FM_SCORE	R_SCORE	CUST_SEGMENT
12346	326	2	0	1	1	Lost
12347	2	182	4310	5	5	Champions
12348	75	31	1797.24	4	2	At Risk
12349	19	73	1757.55	4	4	Potential Loyalists
12350	310	17	334.4	2	1	Hibernating
12352	36	95	1545.41	4	3	Loyal Customers
12353	204	4	89	1	1	Lost

Q3 –

1 –

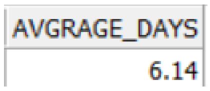
```
SELECT CUST_ID, MAX(cons_days) as max_consecutive_days
FROM (
  SELECT CUST_ID, COUNT(*) AS cons_days
  FROM (
    SELECT CUST_ID, TO_DATE(CALENDAR_DT, 'YYYY-MM-DD') AS order_date,
           ROW_NUMBER() OVER (PARTITION BY CUST_ID ORDER BY TO_DATE(CALENDAR_DT, 'YYYY-MM-
DD')) rn
    FROM SALES
  )
  GROUP BY CUST_ID , TO_CHAR(order_date - rn + 1 , 'YYYY-MM-DD')
)
GROUP BY CUST_ID
order by CUST_ID
 ;
```



CUST_ID	MAX_CONSECUTIVE_DAYS
100010376	5
100011085	10
100014033	46
100018482	3
100020880	46

2-

```
WITH daily_spending AS (
  SELECT
    CUST_ID,
    CALENDAR_DT,
    SUM(AMT_LE) OVER (PARTITION BY CUST_ID ORDER BY CALENDAR_DT) AS total_spending
  FROM
    SALES
),
threshold_unreached AS (
  SELECT
    CUST_ID,
    CALENDAR_DT,
    total_spending
  FROM
    daily_spending
  WHERE
    total_spending < 250
),
threshold_reached AS (
  SELECT
    CUST_ID,
    CALENDAR_DT,
    total_spending
  FROM
    daily_spending
  WHERE
    total_spending >= 250
),
 avg_days as (SELECT
  CUST_ID,
  COUNT( CALENDAR_DT)AS days_to_reach_threshold
FROM
  threshold_unreached
  where CUST_ID in (select CUST_ID from threshold_reached )
GROUP BY
  CUST_ID
 order by CUST_ID )
 SELECT round (avg(days_to_reach_threshold),2) as avgrage_days from avg_days ;
```



AVGRAGE_DAYS
6.14