Table of Contents

Question 1 2

Level Easy 2

Story 1: Grand Total By Year 2

Story 2: Most Selling Product 3

Story 3: Top 5 paying Customers 4

Level Medium 5

Story 4: Group Data By Year and Quarter 5

Story 5: Rush Hour 6

Level Hard 7

Story 6: Running Totals in 2010 7

Story 7: Time Series Analysis 9

Question 2 11

#sol 1 The Lazy way 11

Screenshot of the output 12

#sol 2: Greater and Smaller than 13

Screenshot of the output 14

Another Sol to the CTE: 15

Question 1

Level Easy

Story 1: Grand Total By Year

Knew about rollup() function

```
--grand total by year
SELECT extract(year from (to_date(invoicedate, 'MM/DD/YYYY HH24:MI'))) year,
            SUM(price*quantity) AS sales_total
FROM tableRetail
GROUP BY ROLLUP(extract(year from (to_date(invoicedate, 'MM/DD/YYYY HH24:MI'))));
```

YEAR	SALES_TOTAL
2010	13422.96
2011	242295.42
	255718.38

Story 2: Most Selling Product

```
--without analytical
SELECT stockcode, sum(quantity) frequent_buy
from tableRetail
group by stockcode
order by frequent_buy desc;
```

STOCKCODE	FREQUENT_BUY
84077	7824
84879	6117
22197	5918
21787	5075
21977	4691
21703	2996
17096	2019
15036	1920
23203	1803
21790	1579
22988	1565
23215	1492
20974	1478
22992	1359
21731	1342
22693	1320
40016	1284
22991	1227
23084	1194
22970	1160

```
--with analytical
select stockcode, sum(quantity) over (partition by
stockcode) as frequent_buy
from tableRetail
order by frequent_buy desc;
```

STOCKCODE	FREQUENT_BUY
84077	7824
84077	7824
84077	7824
84077	7824
84077	7824
84077	7824
84077	7824
84077	7824
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117
84879	6117

It is always a good practice to know what your most selling products are, to always make sure there is always sufficient stock for example; and what your least selling ones are as well to adjust the quantities of your market depending on the demand.

Story 3: Top 5 paying Customers

Could be used for loyalty points or something, or could help us know the least paying customers as well to target them to encourage them to pay more often.

```
select *
from
(
SELECT customer_id,
            SUM(quantity * price) as paid,
            row_number() OVER (ORDER BY (SUM(quantity * price)) DESC) AS rnk
            FROM tableRetail
           GROUP BY customer_id
) rnk_paid
WHERE rnk_paid.rnk <=5;
```

Could be done also by rank(), dense_rank() as well

CUSTOMER_ID	PAID	RNK
12931	42055.96	1
12748	33719.73	2
12901	17654.54	3
12921	16587.09	4
12939	11581.8	5

Level Medium

Story 4: Group Data By Year and Quarter

```
SELECT EXTRACT(year FROM (to_date(invoicedate, 'MM/DD/YYYY HH24:MI'))) as year,
          TO_CHAR((TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')), 'Q') as quarter,
          ROUND(SUM(price * quantity)) as sales,
          RANK() OVER(ORDER BY SUM(quantity * price)) as  rnk
FROM tableRetail
GROUP BY EXTRACT(year FROM (to_date(invoicedate, 'MM/DD/YYYY HH24:MI'))),
          TO_CHAR((TO_DATE(invoicedate, 'MM/DD/YYYY HH24:MI')), 'Q')
ORDER BY rnk DESC;
```

Cancel

YEAR	QUARTER	SALES	RNK
2011	3	81893	5
2011	4	76493	4
2011	2	43994	3
2011	1	39916	2
2010	4	13423	1

Story 5: Rush Hour

To know the rush hour of the day, it will help us know when we are overstaffed or understaffed, the best day/hour to make campaigns for new products, what are the reasons behind our busy store at this specific hour, what can we do to make it better for our customers while waiting in the queue and many more other insights

```
select  to_char(to_date(invoicedate, 'MM/DD/YYYY HH24:MI'), 'DAY HH24') rush_hr,
        max(count(distinct(invoice))) over (partition by  to_char(to_date(invoicedate, 'MM/DD/YYYY
HH24:MI'), 'DAY HH24')) as countt,
        rank() over(order by count(distinct invoice) desc) as rankk
from TableRetail
group by  to_char(to_date(invoicedate, 'MM/DD/YYYY HH24:MI'), 'DAY HH24');
```

RUSH_HR	COUNTT	RANKK
THURSDAY 12	30	1
WEDNESDAY 12	25	2
WEDNESDAY 13	23	3
FRIDAY 14	22	4
FRIDAY 12	22	4
THURSDAY 14	21	6
FRIDAY 13	20	7
THURSDAY 13	19	8
WEDNESDAY 11	19	8
WEDNESDAY 14	19	8
SUNDAY 12	19	8
MONDAY 13	18	12
TUESDAY 10	17	13
FRIDAY 11	16	14
TUESDAY 13	16	14
THURSDAY 15	16	14
WEDNESDAY 09	15	17
FRIDAY 10	15	17
TUESDAY 15	14	19
MONDAY 12	14	19

25: 1 Row 1 of 62 total rows HR@XE Modif

CAPS NUM INS

This means that the busiest hour is on Thursday at 12 pm, with count 30 invoices,

Second busiest time is on wed 12 pm, then wed 1 pm, then Friday 2 pm,

As we can see that most busy hours are around noon, specifically from 12pm → 2pm.

Level Hard

Story 6: Running Totals in 2010

```
SELECT  DISTINCT trunc(to_date(invoicedate, 'MM/DD/YYYY HH24:MI')) as dt,
            stockcode,
            SUM (price * quantity) OVER (PARTITION BY stockcode ORDER BY (to_date(invoicedate,
'MM/DD/YYYY HH24:MI')) ASC) AS running_tot
FROM tableRetail
WHERE extract(year from (to_date(invoicedate, 'MM/DD/YYYY HH24:MI')))  = '2010'
ORDER BY stockcode;
```

Output Screenshot

DT	STOCKCODE	RUNNING_TOT
09/12/2010	10002	8.5
14/12/2010	10002	11.05
10/12/2010	10120	0.21
05/12/2010	10133	2.55
05/12/2010	10133	11.05
05/12/2010	15036	9
05/12/2010	15039	10.2
10/12/2010	15056BL	5.95
05/12/2010	16010	1.44
05/12/2010	16011	1.05
16/12/2010	16011	6.09
16/12/2010	16012	5.04
10/12/2010	16054	0.84
10/12/2010	16161U	10.5
06/12/2010	16169K	10.5
10/12/2010	16169P	10.5
10/12/2010	16236	0.21
10/12/2010	16237	0.21
10/12/2010	16237	2.31
14/12/2010	17013D	4.95

1: 25 Row 1 of 500 fetched so far (more rows exist)

Its description

Below is a code for illustration only, the details for stockcode 10002,
we can see than on 12/14/2010, its price was 0.85 x 3 quantity = 2.55
and on 12/9/2010 → 0.85 X 10 = 8.5 + last day which is 2.55 = 11.05 which is the first result of our query for stockcode 10002.

It is a running total on each day for 2010 for each stockcode, so we can track each day how much we gained by this item.

```
select * from tableretail
where stockcode ='10002' and extract(year from (to_date(invoicedate, 'MM/DD/YYYY HH24:MI'))) = '2010';
```

Data Grid

Data Grid | Auto Trace | DBMS Output (disabled) | Query Viewer | CodeXpert | Explain Plan | Script Output

Cancel

INVOICE	STOCKCODE	QUANTITY	INVOICEDATE	PRICE	CUSTOMER_ID	COUNTRY
538890	10002	3	12/14/2010 16:39	0.85	12867	United Kingdom
538086	10002	10	12/9/2010 14:44	0.85	12872	United Kingdom

Story 7: Time Series Analysis

```
--needed it to compare dates for the first time
ALTER SESSION SET NLS_DATE_fORMAT = 'MM-DD-YYYY';
```

```
SELECT trunc(to_date(invoicedate, 'MM/DD/YYYY HH24:MI')) dt,
       SUM(price*quantity) AS daily_sum,
       (SUM(price*quantity)-LAG(SUM(price*quantity)) OVER (ORDER BY trunc(to_date(invoicedate,
'MM/DD/YYYY HH24:MI')) ASC)) AS daily_diff
FROM tableRetail
WHERE  trunc(to_date(invoicedate, 'MM/DD/YYYY HH24:MI')) > '09/30/2011' AND
trunc(to_date(invoicedate, 'MM/DD/YYYY HH24:MI')) <= '10/31/2011'
GROUP BY trunc(to_date(invoicedate, 'MM/DD/YYYY HH24:MI'));
```

Output Screenshot

Cancel

DT	DAILY_SUM	DAILY_DIFF
03/10/2011	2512.76	
04/10/2011	675.38	-1837.38
05/10/2011	740.34	64.96
06/10/2011	3360.58	2620.24
07/10/2011	929.51	-2431.07
09/10/2011	224.53	-704.98
10/10/2011	12.12	-212.41
11/10/2011	1200.82	1188.7
14/10/2011	603.44	-597.38
16/10/2011	1164.27	560.83
17/10/2011	105.22	-1059.05
18/10/2011	790.96	685.74
19/10/2011	509.42	-281.54
23/10/2011	124.59	-384.83
24/10/2011	1328.07	1203.48
25/10/2011	394.44	-933.63
26/10/2011	1942.2	1547.76
27/10/2011	2015.9	73.7
28/10/2011	105	-1910.9
30/10/2011	318.06	213.06

Its description

if we want to analyze a specific month in detail to discover whether there was a drop in transactions and where? On which day exactly?

The above query shows *daily changes of transaction values made by customers during October 2011*

To calculate the daily difference, you need to deduct the previous day's transaction sum from the current day's sum and this is returned by the lag()

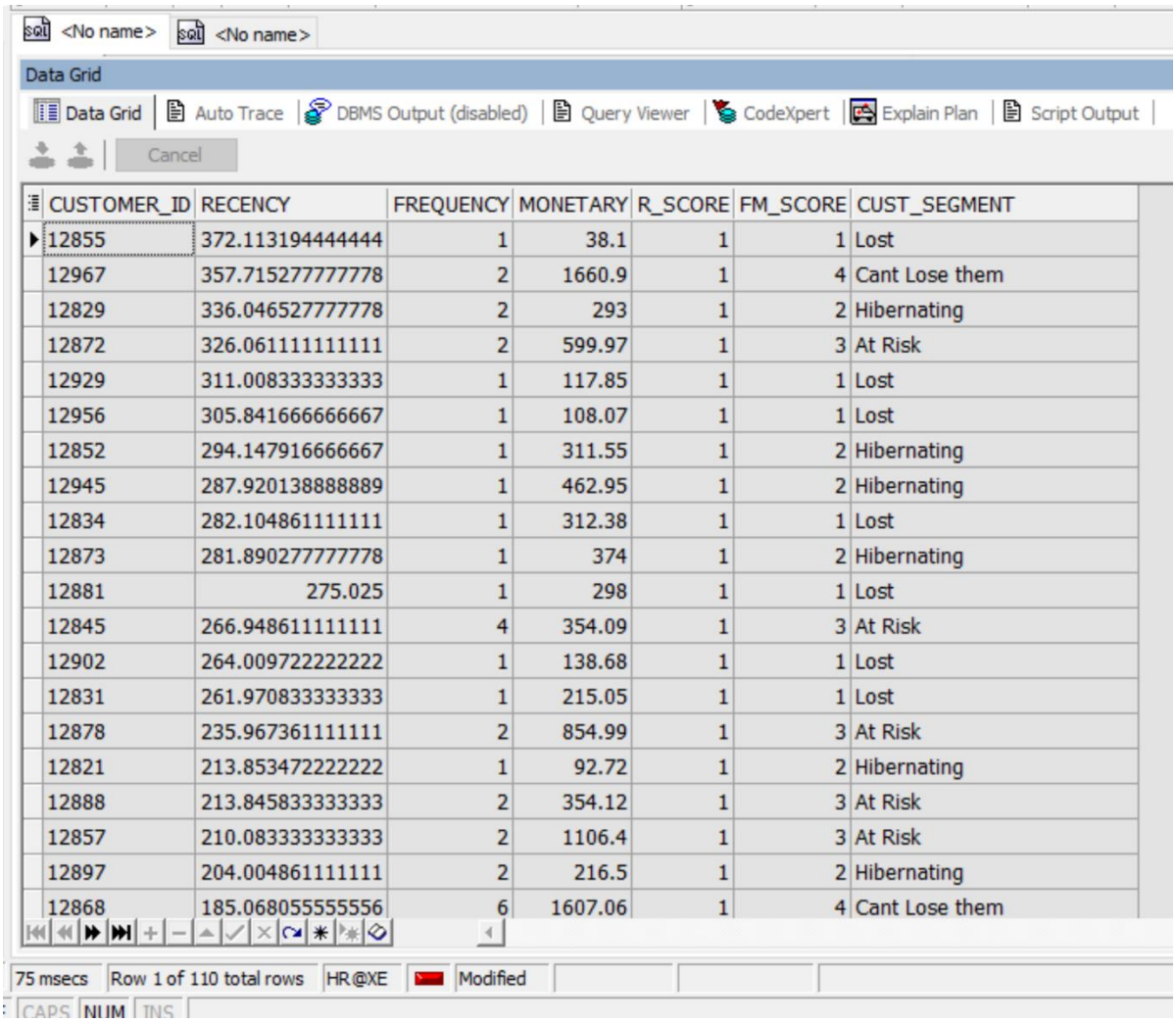
Question 2

#sol 1 The Lazy way

```
WITH rfm AS (SELECT customer_id,
                    (SELECT MAX(TO_DATE(invoicedate, 'MM/DD/YYYY hh24:mi')) from tableRetail) -
(MAX(TO_DATE(invoicedate, 'MM/DD/YYYY hh24:mi'))) as recency,
                    COUNT(DISTINCT(invoice)) as frequency,
                    SUM((price * quantity)) as monetary
                    FROM tableRetail
                    GROUP BY customer_id),
      rfm_calc AS(SELECT rfm.*,
                    NTILE(5) OVER (ORDER BY recency desc) as r_score,
                    ROUND(((NTILE(5) OVER (ORDER BY frequency)) + (NTILE(5) OVER(ORDER BY
monetary)))/2) as fm_score
                    FROM rfm)

SELECT rfm.*,
      (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) OR (r_score = 2 AND fm_score = 1)
          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)
          THEN 'Hibernating'
          WHEN (r_score = 1 AND fm_score = 1)
          THEN 'Lost'
      END) as cust_segment
FROM rfm_calc rfm;
```

Screenshot of the output



CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	R_SCORE	FM_SCORE	CUST_SEGMENT
12855	372.113194444444	1	38.1	1	1	Lost
12967	357.715277777778	2	1660.9	1	4	Cant Lose them
12829	336.046527777778	2	293	1	2	Hibernating
12872	326.061111111111	2	599.97	1	3	At Risk
12929	311.008333333333	1	117.85	1	1	Lost
12956	305.841666666667	1	108.07	1	1	Lost
12852	294.147916666667	1	311.55	1	2	Hibernating
12945	287.920138888889	1	462.95	1	2	Hibernating
12834	282.104861111111	1	312.38	1	1	Lost
12873	281.890277777778	1	374	1	2	Hibernating
12881	275.025	1	298	1	1	Lost
12845	266.948611111111	4	354.09	1	3	At Risk
12902	264.009722222222	1	138.68	1	1	Lost
12831	261.970833333333	1	215.05	1	1	Lost
12878	235.967361111111	2	854.99	1	3	At Risk
12821	213.853472222222	1	92.72	1	2	Hibernating
12888	213.845833333333	2	354.12	1	3	At Risk
12857	210.083333333333	2	1106.4	1	3	At Risk
12897	204.004861111111	2	216.5	1	2	Hibernating
12868	185.068055555556	6	1607.06	1	4	Cant Lose them

#sol 2: Greater and Smaller than

```
WITH rfm AS (SELECT customer_id,
                    (SELECT MAX(TO_DATE(invoicedate, 'MM/DD/YYYY hh24:mi')) from tableRetail) -
(MAX(TO_DATE(invoicedate, 'MM/DD/YYYY hh24:mi'))) as recency,
                    COUNT(DISTINCT(invoice)) as frequency,
                    SUM((price * quantity)) as monetary
                    FROM tableRetail
                    GROUP BY customer_id),
      rfm_calc AS(SELECT rfm.*,
                    NTILE(5) OVER (ORDER BY recency desc) as r_score,
                    ROUND(((NTILE(5) OVER (ORDER BY frequency)) + (NTILE(5) OVER(ORDER BY
monetary)))/2) as fm_score
                    FROM rfm)

SELECT rfm.*,
     (CASE WHEN r_score = 1 AND fm_score >= 4
          THEN 'Cant Lose them'
          WHEN r_score = 1 AND fm_score = 2
          THEN 'Hibernating'
          WHEN r_score = 1 AND fm_score = 1
          THEN 'Lost'
          WHEN r_score = 5 AND fm_score = 1
          THEN 'Recent Customers'
          WHEN r_score = 5 AND fm_score = 3
          THEN 'Loyal Customers'
          WHEN r_score = 3 AND fm_score = 2
          THEN 'Customers Needing Attention'
          WHEN r_score >= 4 AND fm_score >= 4
          THEN 'Champions'
          WHEN r_score >= 3 AND fm_score BETWEEN 2 AND 3
          THEN 'Potential Loyalists'
          WHEN r_score >= 3 AND fm_score >= 3
          THEN 'Loyal Customers'
          WHEN r_score >= 3 AND fm_score = 1
          THEN 'Promising'
          WHEN r_score >= 2 AND fm_score BETWEEN 2 AND 3
          THEN 'Customers Needing Attention'
          WHEN r_score >= 1 AND fm_score >=3
          THEN 'At Risk'
      END) as cust_segment
FROM rfm_calc rfm;
```

Screenshot of the output

Data Grid | Auto Trace | DBMS Output (disabled) | Query Viewer | CodeXpert | Explain Plan | Script Output

Cancel

CUSTOMER_ID	RECENCY	FREQUENCY	MONETARY	R_SCORE	FM_SCORE	CUST_SEGMENT
12855	372.113194444444	1	38.1	1	1	Lost
12967	357.715277777778	2	1660.9	1	4	Cant Lose them
12829	336.046527777778	2	293	1	2	Hibernating
12872	326.061111111111	2	599.97	1	3	At Risk
12929	311.008333333333	1	117.85	1	1	Lost
12956	305.841666666667	1	108.07	1	1	Lost
12852	294.147916666667	1	311.55	1	2	Hibernating
12945	287.920138888889	1	462.95	1	2	Hibernating
12834	282.104861111111	1	312.38	1	1	Lost
12873	281.890277777778	1	374	1	2	Hibernating
12881	275.025	1	298	1	1	Lost
12845	266.948611111111	4	354.09	1	3	At Risk
12902	264.009722222222	1	138.68	1	1	Lost
12831	261.970833333333	1	215.05	1	1	Lost
12878	235.967361111111	2	854.99	1	3	At Risk
12821	213.853472222222	1	92.72	1	2	Hibernating
12888	213.845833333333	2	354.12	1	3	At Risk
12857	210.083333333333	2	1106.4	1	3	At Risk
12897	204.004861111111	2	216.5	1	2	Hibernating
12868	185.068055555556	6	1607.06	1	4	Cant Lose them

78 msecs | Row 1 of 110 total rows | HR@XE | Modified

Another Sol to the CTE:

--This is only for my learning journey to expand on the options :)

But as the CTE is better so I solved as shown above

```
SELECT *
FROM
(SELECT customer_id, recency, frequency, monetary,
        NTILE(5) OVER (ORDER BY recency desc) as r_score,
        ROUND((NTILE(5) OVER (ORDER BY frequency)) + (NTILE(5) OVER(ORDER BY monetary))/2)
as fm_score
        FROM
            (SELECT customer_id,
                (SELECT MAX(TO_DATE(invoicedate, 'MM/DD/YYYY hh24:mi')) from tableRetail) -
(MAX(TO_DATE(invoicedate, 'MM/DD/YYYY hh24:mi'))) as recency,
                COUNT(DISTINCT(invoice)) as frequency,
                SUM((price * quantity)) as monetary
                FROM tableRetail
                GROUP BY customer_id));
```