Sheet-4

1. (a) [2, 6, 8, 12, 20. 25, 26, 29]

omit all coefficients ≤ 3, over 0 to 7

Soln:

Resolution	Averages	Diff
8	2, 6, 8, 12, 20, 25, 26, 29	
4	4, 10, 22.5, 27.5	4, 4, 5, 3
2	7, 25	6, 5
1	16	18

wavelet transform,

$\therefore \hat{S} = [16, 18, 6, 5, 4, 4, 5, 3]$

Omitting less than or equal to 3 values, we get,

$\hat{S} = [16, 18, 6, 5, 4, 4, 5, 0]$

Coefficient calculation:-

$a = 16 - \frac{1}{2} \times 18 = 7$

$b = 16 + \frac{1}{2} \times 18 = 25$

for 7: $a = 7 - \frac{1}{2} \times 6 = 4$

$b = 7 + \frac{1}{2} \times 6 = 10$

for 25: $a = 25 - \frac{1}{2} \times 5 = 22.5$

$b = 25 + \frac{1}{2} \times 5 = 27.5$

for 4: $a = 4 - 1/2 \times 4 = 2$

$b = 4 + 1/2 \times 4 = 6$

for 10: $a = 10 - \frac{1}{2} \times 4 = 8$

$b = 10 + \frac{1}{2} \times 4 = 12$

for 22.5: $a = 22.5 - \frac{1}{2} \times 5 = 20$

$b = 22.5 + \frac{1}{2} \times 5 = 25$

for 27.5: $a = 27.5 - 0 = 27.5$

$b = 27.5 + 0 = 27.5$

① ② ③ ④ ⑤ ⑥ ⑦ ⑧

So, $\hat{S} = [2, 6, 8, 12, 20, 25, 27.5, 27.5]$

Comparing it with original resolution we get errors -1.5 and 1.5 at position 7 and 8

So, mean squared error

$$MSE = \frac{(1.5)^2 + (1.5)^2}{8} = \frac{4.5}{8} = 0.5625$$

(b) Given, $K = [1, 2, 3, 4, 5]$

$$\hat{n} = (K-1) / U_{(K)}$$

$$Ae = \hat{n}^{UB} - n \text{ (no. of real data in table)}$$

K	$\hat{n}^{UB}$ (Estimator)	Absolute error (Ae)
1	Toyota $(1-1/0.05) = 0$	$10 - 0 = 10$
2	Mercedes $(2-1/0.15) = 6.66$	$10 - 6.66 = 3.33$
3.	BMW $(3-1/.281) = 7.11$	$10 - 7.11 = 2.89$
4.	Nissan $(4-1/.395) = 7.59$	$10 - 7.59 = 2.41$
5.	Ford $(5-1/.489) = 8.18$	$10 - 8.18 = 1.82$

From the table we can see that, with the increase of value of K, absolute error decreases, which generally illustrates, availability of large dataset will result in less error.

2. (a)

null_frac :- Fraction of column entries that are null

attname :- Name of the column described by this row

n_distinct :- If greater than 0, the estimated no. of distinct values in column. If less than 0, negative of the no. of distinct values divided by no.

of rows.

most_common_vals: A list of most common values in the column. (Null if no values seem to be more common then others.

most_common_freqs: A list of frequencies of the most common values, i.e. no. of occurences of each divided by total no. of rows.

(b)

1. Statistics won't help because all values will be retrieved.

2. null_frac can be used to return no. of null items, which in this case is 0.

3. The row with value 406631 isn't present in the most_common_vals

column. We can still make estimation which results in 600215/41964 ≈ 14.33 roughly 15 rows.

4. 32274.0 is contained in most common value. So resulting size can easily be estimated, which, in this case is 600215 * 0.0001 ≈ 600

5. pg-stats can't be used in this case because value of extended-price is −0.11964 * 600215 ≈ 717985. The resulting size wouldn't help us to determine the range less than or equal to 9500.

(c)

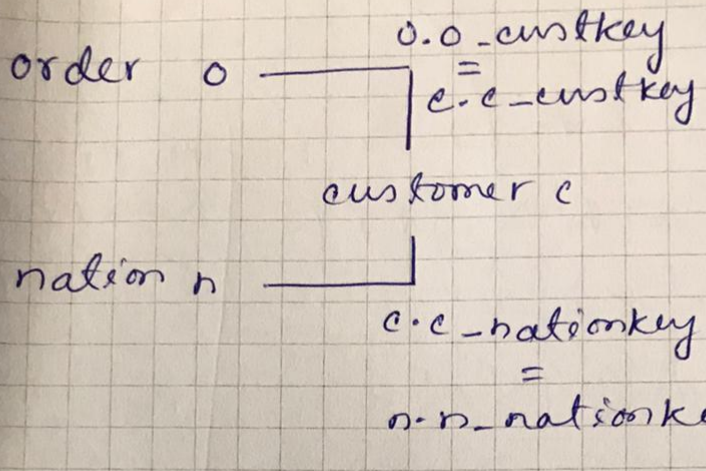
1. For query A :- 6001215 rows
2. For query B :- 6001215 rows
3. For query C : 6001215/419664 ≈ 15 rows
4. For query C : 6001215 * 0.0001 ≈ 600 rows
5. Range query which can't exactly determine the resulting size using most_common_vals / most_common_freqs

(d) Additional histogram_bound column is useful to determine result size in range query which is Query E.

The resulting range falls between buckets 12 and 13. So we need at least 12 cells which results (12 * 6001215)/n no. of tuples.

3

(a)



(b) From database, cardinality as follows

$|O| = 1500000$

$|C| = 150000$

$|N| = 25$

Join selectivity

$|O \bowtie C| = 1500000$

$|C \bowtie N| = 150000$

$|C \times N| = 3750000$

$|O \times C| = 2.5 \times 10^{11}$

$$\therefore t_{o,c} = \frac{|o \bowtie c|}{|o \times c|} = \frac{1500000}{2.55 \times 10^{11}}$$

$$= 6.67 \times 10^{-6}$$

$$t_{c,n} = \frac{|c \bowtie n|}{|c \times n|}$$

$$= \frac{150000}{3750000} = 0.04$$

(c)

$o \bowtie c$	1500750	
$c \bowtie n$	150000	
$(o \bowtie c) \bowtie n$	3001500	1500750
$(c \bowtie n) \bowtie o$	1650750	1500750

4. Did not attempted.

5. Output:

```
Join order exercise:
==========================================
Reading file "data.txt" ...
Added relation "R1" with size 200
Added relation "R2" with size 300
Added relation "R3" with size 20
Added relation "R4" with size 140
Added selectivity of 0.01 between "R1" and "R3"
Added selectivity of 0.02 between "R2" and "R3"
Added selectivity of 0.2 between "R3" and "R4"
==========================================
Greedy 1:       (((R3 ? R4) ? R1) ? R2) Cost: 9060
Greedy 2:       (((R1 ? R3) ? R4) ? R2) Cost: 8540
Greedy 3:       ((((R2 ? R3) ? R3) ? R4) ? R1)  Cost: 1122
Best:           ((((R2 ? R3) ? R3) ? R4) ? R1)  Cost: 1122
Worst:          (((R3 ? R4) ? R1) ? R2) Cost: 9060
```