Homework 4

****

Problem 1

```
SELECT f.long_desc, d.nutrdesc, n.nutr_val, d.units, n.num_studies
FROM food_des f NATURAL JOIN nut_data n NATURAL JOIN nutr_def d;
```

Indexes	CREATE INDEX h_index1 on food_des USING BTREE(ndb_no);
	CREATE INDEX h_index2 on nut_data USING BTREE(ndb_no);
	CREATE INDEX h_index3 on nutr_def USING BTREE(nutr_no);

Reasoning	Adding indexes to the columns being checked for equality in the join.
Explain	"Hash Join (cost=327.85..7131.84 rows=253825 width=80)" " Hash Cond: (n.nutr_no = d.nutr_no)" " -> Hash Join (cost=322.79..6439.73 rows=253825 width=70)" " Hash Cond: (n.ndb_no = f.ndb_no)" " -> Seq Scan on nut_data n (cost=0.00..5450.25 rows=253825 width=22)" " -> Hash (cost=233.46..233.46 rows=7146 width=60)" " -> Seq Scan on food_des f (cost=0.00..233.46 rows=7146 width=60)" " -> Hash (cost=3.36..3.36 rows=136 width=18)" " -> Seq Scan on nutr_def d (cost=0.00..3.36 rows=136 width=18)"
Speedup	No.

****

Problem 2

```
SELECT f.long_desc, d.nutrdesc, n.nutr_val, d.units, n.num_studies
FROM food_des f NATURAL JOIN nut_data n NATURAL JOIN nutr_def d
WHERE f.long_desc LIKE 'Butter%' AND d.nutrdesc = 'Cholesterol';
```

Indexes	CREATE INDEX h_index4 on food_des(long_desc);
	CREATE INDEX h_index5 on nutr_def USING HASH(nutrdesc);

Reasoning	Added BTree index on long_desc column of food_des for comparison LIKE ‘Butter%’ and hash index for equality nutrdesc = ‘Cholestrol’
Explain	"Nested Loop (cost=0.42..263.48 rows=1 width=80)" " -> Seq Scan on nutr_def d (cost=0.00..3.70 rows=1 width=18)" " Filter: (nutrdesc = 'Cholesterol'::text)" " -> Nested Loop (cost=0.42..259.77 rows=1 width=70)" " -> Seq Scan on food_des f (cost=0.00..251.32 rows=1 width=60)" " Filter: (long_desc ~~ 'Butter%'::text)" " -> Index Scan using nut_data_pkey on nut_data n (cost=0.42..8.44 rows=1 width=22)" " Index Cond: ((ndb_no = f.ndb_no) AND (nutr_no = d.nutr_no))"
Speedup	No.

Problem 3

```
SELECT f.long_desc, d.nutrdesc, n.nutr_val, d.units, n.num_studies
FROM food_des f NATURAL JOIN nut_data n NATURAL JOIN nutr_def d
WHERE n.num_studies > 3;
```

Indexes	CREATE INDEX num_study on nut_data(num_studies)
Reasoning	Added BTree index on num_studies for comparison num_studies > 3
Explain	"Hash Join (cost=346.22..2681.51 rows=1285 width=80)" " Hash Cond: (n.nutr_no = d.nutr_no)" " -> Hash Join (cost=341.16..2672.97 rows=1285 width=70)" " Hash Cond: (n.ndb_no = f.ndb_no)" " -> Bitmap Heap Scan on nut_data n (cost=18.38..2346.82 rows=1285 width=22)" " Recheck Cond: (num_studies > 3)" " -> Bitmap Index Scan on num_study (cost=0.00..18.06 rows=1285 width=0)" " Index Cond: (num_studies > 3)"
Speedup	Yes

Problem 4:

```
SELECT f.long_desc, d.nutrdesc, n.nutr_val, d.units, n.num_studies
FROM food_des f NATURAL JOIN nut_data n NATURAL JOIN nutr_def d
WHERE d.units='mg' AND n.nutr_val > 0
ORDER BY n.nutr_val DESC;
```

Indexes	CREATE INDEX d_units on nutr_def(units);
	CREATE INDEX n_nutr_val ON nut_data(nutr_val);

Reasoning	Units and nutr_val are involved in a WHERE clause so added indexes for those columns
Explain	"Sort (cost=9488.42..9572.53 rows=33644 width=80)" " Sort Key: n.nutr_val DESC" " -> Hash Join (cost=326.82..6958.72 rows=33644 width=80)" " Hash Cond: (n.ndb_no = f.ndb_no)" " -> Hash Join (cost=4.04..6547.56 rows=33644 width=32)"

	" Hash Cond: (n.nutr_no = d.nutr_no)" " -> Seq Scan on nut_data n (cost=0.00..6084.81 rows=169466 width=22)" " Filter: (nutr_val > '0'::double precision)"
Speedup	No

Problem 5 :

```
CREATE VIEW percentages AS
SELECT f.long_desc, d.nutrdesc, n.nutr_val, d.units,
n.num_studies, w.gm_wgt, w.num_data_pts,
CASE WHEN d.units = 'g' THEN (n.nutr_val / w.gm_wgt) * 100
WHEN d.units = 'mg' THEN ((n.nutr_val / 1000) / w.gm_wgt) * 100
WHEN d.units LIKE 'mcg%' THEN ((n.nutr_val / 1000000)/ w.gm_wgt) * 100
ELSE NULL
END AS percent
FROM food_des f NATURAL JOIN nut_data n NATURAL JOIN nutr_def d
NATURAL JOIN weight w;

SELECT *
FROM percentages
WHERE nutrdesc LIKE 'Iron%'
AND percent < 1
AND num_data_pts > 2
ORDER BY percent;
```

Indexes	CREATE INDEX pn on nutr_def(nutrdesc);
	CREATE INDEX nd on nutr_def(units);
	CREATE INDEX ds on weight(num_data_pts);

Reasoning	Column num_data pts from weight is in a where clause in the select statement , percent is in a where clause which involves a case expression - column units from nutr_def has several cases , so an index has been added for that column. Column nutrdesc from nutrdef is in a where clause.
Explain	"Sort (cost=2365.28..2365.29 rows=1 width=100)" " Sort Key: (CASE WHEN (d.units = 'g'::text) THEN ((n.nutr_val / w.gm_wgt) * '100'::double precision) WHEN (d.units = 'mg'::text) THEN (((n.nutr_val / '1000'::double precision) / w.gm_wgt) * '100'::double precision) WHEN (d.units ~~ 'mcg%'::text) THEN (((n.nutr_val / '1000000'::double precision) / w.gm_wgt) * '100'::double precision) ELSE NULL::double precision END)" " -> Nested Loop (cost=331.26..2365.27 rows=1 width=100)" " Join Filter: ((n.nutr_no = d.nutr_no) AND (CASE WHEN (d.units = 'g'::text) THEN ((n.nutr_val / w.gm_wgt) * '100'::double precision) WHEN (d.units = 'mg'::text) THEN (((n.nutr_val / '1000'::double precision) / w.gm_wgt) * '100'::double precision) WHEN (d.units ~~ 'mcg%'::text)

	THEN (((n.nutr_val / '1000000'::double precision) / w.gm_wgt) * '100'::double precision) ELSE NULL::double precision END < '1'::double precision))" " -> Seq Scan on nutr_def d (cost=0.00..3.70 rows=1 width=18)" " Filter: (nutrdesc ~~ 'Iron%'::text)" " -> Nested Loop (cost=331.26..2357.89 rows=86 width=82)" " Join Filter: ((w.num_data_pts)::double precision = n.num_data_pts)" " -> Hash Join (cost=330.84..463.19 rows=486 width=78)" " Hash Cond: (w.ndb_no = f.ndb_no)" " -> Bitmap Heap Scan on weight w (cost=8.05..139.13 rows=486 width=18)" " Recheck Cond: (num_data_pts > 2)" " -> Bitmap Index Scan on ds (cost=0.00..7.93 rows=486 width=0)" " Index Cond: (num_data_pts > 2)" " -> Hash (cost=233.46..233.46 rows=7146 width=60)" " -> Seq Scan on food_des f (cost=0.00..233.46 rows=7146 width=60)" " -> Index Scan using nut_data_pkey on nut_data n (cost=0.42..3.30 rows=40 width=30)" " Index Cond: (ndb_no = f.ndb_no)"
Speedup	I can see that the newly added index on num_data_pts is mentioned in the query planner , so a speedup was expected , but the speed was about the same as prior to adding index.

****

Problem 6 :

Indexes	CREATE INDEX pn on nutr_def(nutrdesc);
	CREATE INDEX nd on nutr_def(units);
	CREATE INDEX ds on weight(num_data_pts);

Reasoning	Same as Problem 5
Explain	"Sort (cost=244.52..244.53 rows=4 width=128) (actual time=0.916..0.919 rows=42 loops=1)" " Sort Key: percent" " Sort Method: quicksort Memory: 30kB" " -> Seq Scan on percentages (cost=0.00..244.48 rows=4 width=128) (actual time=0.054..0.897 rows=42 loops=1)" " Filter: ((nutrdesc ~~ 'Iron%'::text) AND (percent < '1'::double precision) AND (num_data_pts > 2))" " Rows Removed by Filter: 6042" "Planning Time: 0.119 ms" "Execution Time: 0.979 ms"

Speedup	Yes. although the indexes were not used

Problem 7 :
The query computes the natural join of three tables :
Food_des , nut_data , nutr_def

Problem 8 :
Ndb_no is not the primary key in weight , (ndb_no,seq) is the primary key .
One value for ndb_no in weight can have multiple tuples associated with that value .
So if food_des referenced weight , that would be a database violation since the
Column being referred to has to be a primary key in that table .
Hence only weight can reference food_des (ndb_no) as ndb_no is a primary key
In food_des.

****