Exercice 1

1.1

Oui, il est de type Btree.

1.2

Oui un index est bien utilisé. C'est un Index Scan

1.3

Il n'y a pas d'index, c'est un Sequence Scan car il y a une fonction

```
SELECT fac_num, fac_date, fac_montant
FROM music.facture
WHERE fac_num >= 500
AND fac_num < 600;
```

1.4

```
SELECT fac_num, fac_date, fac_montant
FROM music.facture
WHERE fac_num >= 500
AND fac_num < 550
AND fac_date > '2017-06-01'
AND fac_date < '2017-08-31';
```

Exercice 2

2.1

```
CREATE INDEX index_floor
ON music.facture USING btree
(floor(fac_num/100))
TABLESPACE pg_default;
```

C'est du 1 pour 100 (pas très séléctif)

2.2

- floor : 600 ms

- index : 500 ms
- range : 550 ms

2.3

```
CREATE INDEX index_date
ON music.facture USING btree
(fac_date)
TABLESPACE pg_default;
```

2.4

```
CREATE INDEX idx_fac_date_hash
ON facture
using hash(date_trunc('month',fac_date));
```

2.5

```
SELECT fac_num, fac_date, fac_montant
FROM music.facture
WHERE fac_date BETWEEN '2017-06-01' AND '2017-08-31' ;
```

Exercice 3

3.1

```
SELECT * FROM music.ligne_facture
WHERE lig_facture = 250 AND lig_produit = 44 ;
```

-> index scan

```
SELECT * FROM music.ligne_facture
WHERE lig_facture BETWEEN 1000 AND 1005 ;
```

-> Bitmap Index Scan | Bitmap Heap Scan

```
SELECT * FROM music.ligne_facture
WHERE lig_produit = 44 ;
```

-> seq scan

3.2

- insert : ne prends pas les bénéfices des index
- update : ne peut pas affecter les index
- delete : utilise les index
- select :

Exercice 4

4.1

Index only scan + Aggregate

L'index only scan est utilisé pour faire le filtre sur la clause where et l'aggrégation est utilisé pour faire le count(*).

```
-- taille de la table
SELECT pg_size_pretty(pg_table_size('music.album'));
```

-> 8192 bytes

```
-- taille de l'index
SELECT pg_size_pretty(pg_indexes_size('music.album'));
```

-> 16 kb

4.2

```
SELECT *
FROM music.album
WHERE al_id = 20;
```

Il a fait un Seq Scan car il n'y a pas beaucoup de lignes

4.3

```
SELECT lig_facture,
SUM(lig_quantite) AS nb_produits,
COUNT(*) AS nb_produits_distincts
FROM music.ligne_facture
GROUP BY lig_facture;
```

Entre 500 et 550 -> Bitmap Index Scan + Bitmap Heap Scan + Aggregate

Entre 200 et 1000 -> Seq Scan + Aggregate

Exercice 5

5.1

Oui c'est toujours le cas avec 2.

5.2

Bitmap Index Scan + Bitmap Heap Scan + Sort + Unique La particulité c'est le `DISTINCT`` On trie avant comme ça c'est plus simple de supprimer les doublons

Exercice 6

6.1

Passage de Index Scan à Index Only Scan