Plan

1. SQL SELECT
2. SQL DISTINCT
3. SQL WHERE
4. Syntaxe d'utilisation des opérateurs AND et OR
5. SQL IN
6. SQL BETWEEN
7. SQL IS NULL / IS NOT NULL
8. SQL UNION
9. SQL INTERSECT
10. SQL EXCEPT / MINUS

Commandes SQL illustrées avec des exemples

1. SQL SELECT

La commande SELECT retourne des enregistrements dans un tableau de résultat.
L'utilisation basique de cette commande s'effectue de la manière suivante:

```
SELECT nom_du_champ FROM nom_du_tableau
```

Exemple

Table "client" :

identifiant	prenom	nom	ville
1	ali	fouratio	tunis
2	Sabrina	fourati	sfax
3	Jalel	dhoib	monastir
4	Daoued	Benzarti	gabes
5	Mariem	Lili	sousse

```
SELECT ville FROM client
```

On obtient le résultat suivant :

ville
tunis
sfax
monastir
gabes
sousse

```
SELECT prenom, nom FROM client
```

Ce qui retourne ce résultat:

prenom	nom
ali	fouratio
Sabrina	fourati
Jalel	dhoib
Daoued	Benzarti
Mariem	Lili

Obtenir toutes les colonnes d'un tableau

```
SELECT * FROM client
```

Cette requête SQL retourne exactement les mêmes colonnes qu'il y a dans la base de

identifiant	prenom	nom	ville
1	ali	fouratio	tunis
2	Sabrina	fourati	sfax
3	Jalel	dhoib	monastir
4	Daoued	Benzarti	gabes
5	Mariem	Lili	sousse

2. SQL DISTINCT

La commande SELECT en SQL peut potentiellement afficher des lignes en double. Pour éviter les doublons simplement ajouter DISTINCT après le mot SELECT.

```
SELECT DISTINCT ma_colonne
FROM nom_du_tableau
```

```
SELECT DISTINCT nom FROM client
```

nom
fouratio
dhoib
Benzarti
Lili

3. SQL WHERE

La commande WHERE dans une requête SQL permet d'extraire les lignes d'une base de données qui respectent une condition.

```
SELECT nom_colonnes FROM nom_table WHERE condition
```

Exemple

id	nom	nbr_commande	ville
1	faouzi	3	tunis
2	Maher	0	monastir
3	Jalila	1	Mednine
4	yasmine	7	tunis

Pour obtenir seulement la liste des clients qui habitent à Tunis.

```
SELECT * FROM client WHERE ville = 'tunis'
```

id	nom	nbr_commande	ville
1	faouzi	3	tunis
4	yasmine	7	tunis

Attention: il faut faire attention aux majuscules et minuscules

Opérateurs de comparaisons qui font partie de votre programme

Opérateur	Description
=	Égale
<>	Pas égale
!=	Pas égale
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égale à
<=	Inférieur ou égale à
IN	Liste de plusieurs valeurs possibles
BETWEEN	Valeur comprise dans un intervalle donnée (utile pour les nombres ou dates)
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle

4. Syntaxe d'utilisation des opérateurs AND et OR

Les opérateurs sont à ajoutés dans la condition WHERE.

```
SELECT nom_colonnes
FROM nom_table
WHERE condition1 AND condition2
```

```
SELECT nom_colonnes FROM nom_table
 WHERE condition1 OR condition2
```

```
SELECT nom_colonnes FROM nom_table
 WHERE condition1 AND (condition2 OR condition3)
```

Exemple de données

id	nom	categorie	stock	prix
1	ordinateur	informatique	5	950
2	clavier	informatique	32	35
3	souris	informatique	16	30
4	crayon	fourniture	147	2

```
SELECT * FROM produit
WHERE categorie = 'informatique' AND stock < 20
```

id	nom	categorie	stock	prix
1	ordinateur	informatique	5	950
3	souris	informatique	16	30

```
SELECT * FROM produit
WHERE nom = 'ordinateur' OR nom = 'clavier'
```

id	nom	categorie	stock	prix
1	ordinateur	informatique	5	950
2	clavier	informatique	32	35

```
SELECT * FROM produit
WHERE ( categorie = 'informatique' AND stock < 20 )
OR ( categorie = 'fourniture' AND stock < 200 )
```

id	nom	categorie	stock	prix
1	ordinateur	informatique	5	950
2	souris	informatique	16	30
4	crayon	fourniture	147	2

5. SQL IN

L'opérateur logique IN dans SQL s'utilise avec la commande WHERE pour vérifier si une colonne est égale à une des valeurs comprise dans set de valeurs déterminés.

```
SELECT nom_colonne
FROM table
WHERE nom_colonne IN ( valeur1, valeur2, valeur3, ... )
```

Exemple

id	nom	nbr_commande	ville
1	faouzi	3	tunis
2	Maher	0	monastir
3	Jalila	1	Mednine
4	yasmine	7	tunis

```
SELECT prenom
FROM client
WHERE ville IN ( 'tunis', 'Mednine' )
```

id	nom	nbr_commande	ville
1	faouzi	3	tunis
3	Jalila	1	Mednine
4	yasmine	7	tunis

6. SQL BETWEEN

L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates.

```
SELECT *
FROM table
WHERE nom_colonne BETWEEN 'valeur1' AND 'valeur2'
```

Exemple : filtrer entre 2 dates

id	nom	date_inscription
1	Mourad	2012-03-02
2	Sinen	2012-03-05
3	Chiraz	2012-04-14
4	Mariem	2012-04-15

id	nom	date_inscription
5	faouzi	2012-04-26

obtenir les membres qui se sont inscrit entre le 1 avril 2012 et le 20 avril 2012 il est possible d'effectuer la requête suivante:

```
SELECT *
FROM utilisateur
WHERE date_inscription BETWEEN '2012-04-01' AND '2012-04-20'
```

id	nom	date_inscription
3	Chiraz	2012-04-14
4	Mariem	2012-04-15

7. SQL IS NULL / IS NOT NULL

Dans le langage SQL, l'opérateur IS permet de filtrer les résultats qui contiennent la valeur NULL.

```
SELECT *
FROM `table`
WHERE nom_colonne IS NULL
```

A l'inverse, pour filtrer les résultats et obtenir uniquement les enregistrements qui ne sont pas null.

```
SELECT *
FROM `table`
WHERE nom_colonne IS NOT NULL
```

Exemple

Table "utilisateur" :

id	nom	date_inscription	fk_adresse_livraison_id	fk_adresse_facturation_id
23	jamel	2013-02-12	12	12
24	Sarah	2013-02-17	NULL	NULL
25	Anis	2013-02-21	13	14
26	Ferid	2013-03-02	NULL	NULL

Exemple 1 : utilisateurs sans adresse de livraison

```
SELECT *
FROM `utilisateur`
WHERE `fk_adresse_livraison_id` IS NULL
```

Résultat :

id	nom	date_inscription	fk_adresse_livraison_id	fk_adresse_facturation_id
24	Sarah	2013-02-17	NULL	NULL
26	Ferid	2013-03-02	NULL	NULL

Exemple 2 : utilisateurs avec une adresse de livraison

```
SELECT *
FROM `utilisateur`
WHERE `fk_adresse_livraison_id` IS NOT NULL
```

Résultat :

id	nom	date_inscription	fk_adresse_livraison_id	fk_adresse_facturation_id
23	Jamel	2013-02-12	12	12
25	Anis	2013-02-21	13	14

8. SQL UNION

La commande UNION de SQL permet de mettre bout-à-bout les résultats de plusieurs requêtes utilisant elles-même la commande SELECT. C'est donc une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l'utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre (pas de doublons).

```
SELECT * FROM table1
UNION
SELECT * FROM table2
```

Exemple

La table du magasin n°1 s'appelle "magasin1_client" et contient les données suivantes :

prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39

La table du magasin n°2 s'appelle "magasin2_client" et contient les données suivantes :

prenom	nom	ville	date_naissance	total_achat
Marion	Leroy	Lyon	1982-10-27	285
Paul	Moreau	Lyon	1976-04-19	133
Marie	Bernard	Paris	1993-07-03	75
Marcel	Martin	Paris	1976-11-24	39

```
SELECT * FROM magasin1_client
UNION
SELECT * FROM magasin2_client
```

Résultat :

prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39
Marie	Leroy	Lyon	1982-10-27	285
Paul	Moreay	Lyon	1976-04-19	133

9. SQL INTERSECT

La commande SQL INTERSECT permet d'obtenir l'intersection des résultats de 2 requêtes. Cette commande permet donc de récupérer les enregistrements communs à 2 requêtes. Cela peut s'avérer utile lorsqu'il faut trouver s'il y a des données similaires sur 2 tables distinctes.
A savoir : pour l'utiliser convenablement il faut que les 2 requêtes retourne le même nombre de colonnes, avec les mêmes types et dans le même ordre.

```
SELECT * FROM `table1`
INTERSECT
SELECT * FROM `table2`
```

Exemple
La table du magasin n°1 est "magasin1_client" :

prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39

La table du magasin n°2 est “magasin2_client” :

prenom	nom	ville	date_naissance	total_achat
Marion	Leroy	Lyon	1982-10-27	285
Paul	Moreau	Lyon	1976-04-19	133
Marie	Bernard	Paris	1993-07-03	75
Marcel	Martin	Paris	1976-11-24	39

```
SELECT * FROM `magasin1_client`
INTERSECT
SELECT * FROM `magasin2_client
```

Résultat :

nom	prenom	ville	date_naissance	total_achat
Marie	Bernard	Paris	1993-07-03	75
Marcel	Martin	Paris	1976-11-24	39

10. SQL EXCEPT / MINUS

Dans le langage SQL la commande EXCEPT s’utilise entre 2 instructions pour récupérer les enregistrements de la première instruction sans inclure les résultats de la seconde requête. Si un même enregistrement devait être présent dans les résultats des 2 syntaxes, ils ne seront pas présent dans le résultat final.

```
SELECT * FROM table1
EXCEPT
SELECT * FROM table2
```

Exemple

Table “clients_inscrits” :

id	prenom	nom	date_inscription
1	Lionel	Martineau	2012-11-14
2	Paul	Cornu	2012-12-15
3	Sarah	Schmitt	2012-12-17
4	Sabine	Lenoir	2012-12-18

Table “clients_refus_email” :

id	prenom	nom	date_refus
1	Paul	Cornu	2013-01-27
2	Manuel	Guillot	2013-01-27
3	Sabine	Lenoir	2013-01-29

id	prenom	nom	date_refus
4	Natalie	Petitjean	2013-02-03

Pour pouvoir sélectionner uniquement le prénom et le nom des utilisateurs qui accepte de recevoir des emails informatifs. La requête SQL à utiliser est la suivante :

```
SELECT prenom, nom FROM clients_inscrits
EXCEPT
SELECT prenom, nom FROM clients_refus_email
```

Résultats :

prenom	nom
Lionel	Martineau
Sarah	Schmitt