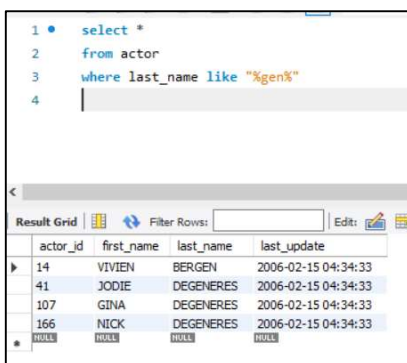


Projet SQL

1^{ère} partie : Base de données Sakila

1. Trouvez tous les acteurs dont le nom de famille contient les lettres "gen".

```
select *  
from actor  
where last_name like "%gen%"
```



The screenshot shows a SQL query editor with the following query:

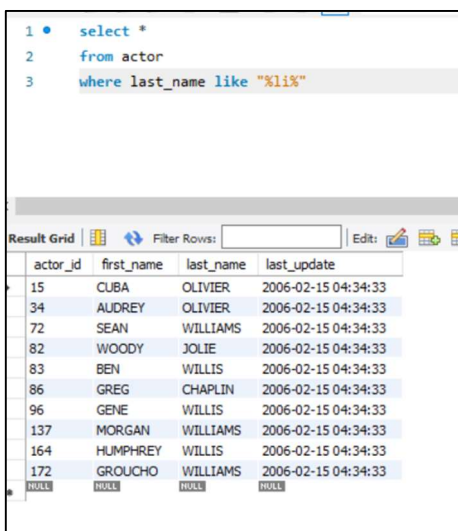
```
1 • select *  
2   from actor  
3   where last_name like "%gen%"  
4
```

Below the query, a "Result Grid" displays the results of the query. The grid has four columns: actor_id, first_name, last_name, and last_update. The results are as follows:

actor_id	first_name	last_name	last_update
14	VIVIEN	BERGEN	2006-02-15 04:34:33
41	JODIE	DEGENERES	2006-02-15 04:34:33
107	GINA	DEGENERES	2006-02-15 04:34:33
166	NICK	DEGENERES	2006-02-15 04:34:33

2. Trouvez tous les acteurs dont le nom de famille contient les lettres "li".

```
select *  
from actor  
where last_name like "%li%"
```



The screenshot shows a SQL query editor with the following query:

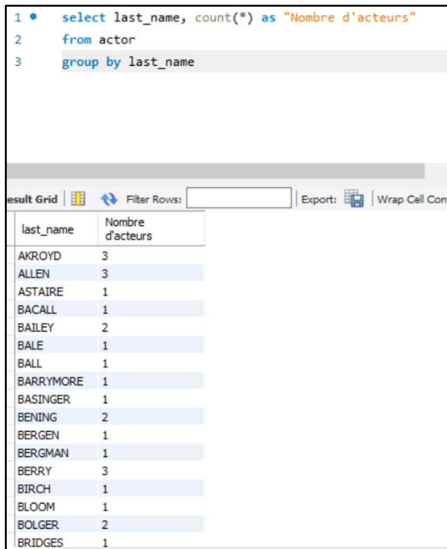
```
1 • select *  
2   from actor  
3   where last_name like "%li%"
```

Below the query, a "Result Grid" displays the results of the query. The grid has four columns: actor_id, first_name, last_name, and last_update. The results are as follows:

actor_id	first_name	last_name	last_update
15	CUBA	OLIVIER	2006-02-15 04:34:33
34	AUDREY	OLIVIER	2006-02-15 04:34:33
72	SEAN	WILLIAMS	2006-02-15 04:34:33
82	WOODY	JOLIE	2006-02-15 04:34:33
83	BEN	WILLIS	2006-02-15 04:34:33
86	GREG	CHAPLIN	2006-02-15 04:34:33
96	GENE	WILLIS	2006-02-15 04:34:33
137	MORGAN	WILLIAMS	2006-02-15 04:34:33
164	HUMPHREY	WILLIS	2006-02-15 04:34:33
172	GROUCHO	WILLIAMS	2006-02-15 04:34:33

3. Liste des noms de famille de tous les acteurs, ainsi que le nombre d'acteurs portant chaque nom de famille.

```
select last_name, count(*) as "Nombre d'acteurs"
from actor
group by last_name
```



The screenshot shows a SQL query editor with the following query:

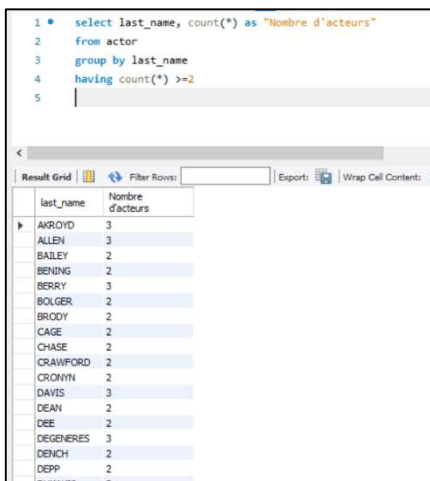
```
1 * select last_name, count(*) as "Nombre d'acteurs"
2 from actor
3 group by last_name
```

Below the query, a "Result Grid" displays the results in a table with two columns: "last_name" and "Nombre d'acteurs". The table contains 15 rows of data.

last_name	Nombre d'acteurs
AKROYD	3
ALLEN	3
ASTAIRE	1
BACALL	1
BAILEY	2
BALE	1
BALL	1
BARRYMORE	1
BASINGER	1
BENING	2
BERGEN	1
BERGMAN	1
BERRY	3
BIRCH	1
BLOOM	1
BOLGER	2
BRIDGES	1

4. Lister les noms de famille des acteurs et le nombre d'acteurs qui portent chaque nom de famille, mais seulement pour les noms qui sont portés par au moins deux acteurs.

```
select last_name, count(*) as "Nombre d'acteurs"
from actor
group by last_name
having count(*) >=2
```



The screenshot shows a SQL query editor with the following query:

```
1 * select last_name, count(*) as "Nombre d'acteurs"
2 from actor
3 group by last_name
4 having count(*) >=2
5
```

Below the query, a "Result Grid" displays the results in a table with two columns: "last_name" and "Nombre d'acteurs". The table contains 20 rows of data, filtered to show only names with a count of 2 or more.

last_name	Nombre d'acteurs
AKROYD	3
ALLEN	3
BAILEY	2
BENING	2
BERRY	3
BOLGER	2
BRODY	2
CAGE	2
CHASE	2
CRAWFORD	2
CROWIN	2
DAVIS	3
DEAN	2
DIE	2
DEGENERES	3
DENCH	2
DEPP	2
DUKAKIS	2

5. Utilisez JOIN pour afficher le montant total perçu par chaque membre du personnel en août 2005.

```
select s.staff_id, CONCAT(s.first_name, ' ', s.last_name) AS staff_name, sum(amount) as Total_montant
from staff as s join payment as p on s.staff_id=p.staff_id
where p.payment_date between "2005-08-01 00:00:00" and "2005-08-31 00:00:00"
group by staff_id
```

```

1 select s.staff_id, CONCAT(s.first_name, ' ', s.last_name) AS staff_name, sum(amount) as Total_montant
2 from staff as s join payment as p on s.staff_id=p.staff_id
3 where p.payment_date between "2005-08-01 00:00:00" and "2005-08-31 00:00:00"
4 group by staff_id

```

staff_id	staff_name	Total_montant
1	Mike Hillyer	11853.65
2	Jon Stephens	12216.49

6. Afficher les titres des films commençant par les lettres K et Q dont la langue est l'anglais.

```

select f.title
from film as f left join language as l on f.language_id=l.language_id
where UPPER(f.title) like "K%" or UPPER(f.title) like "Q%" and UPPER(l.name)="ENGLISH"

```

```

1 select f.title
2 from film as f left join language as l on f.language_id=l.language_id
3 where UPPER(f.title) like "K%" or UPPER(f.title) like "Q%" and UPPER(l.name)="ENGLISH"

```

title
KANE EXORCIST
KARATE MOON
KENTUCKIAN GIANT
KICK SAVANNAH
KILL BROTHERHOOD
KILLER INNOCENT
KING EVOLUTION
KISS GLORY
KISSING DOLLS
KNOCK WARLOCK
KRAMER CHOCOLATE
KWAI HOMEWARD
QUEEN LUKE
QUEST MUSSOLINI
QUILLS BULL

7. Affichez les noms et les adresses électroniques de tous les clients canadiens.

```

select c.first_name, c.email
from customer as c left join address as a on c.address_id=a.address_id
left join city as ci on a.city_id=ci.city_id
left join country as co on ci.country_id=co.country_id
where upper(country)="CANADA"

```

```

1 • select c.first_name, c.email
2   from customer as c left join address as a on c.address_id=a.address_id
3   left join city as ci on a.city_id=ci.city_id
4   left join country as co on ci.country_id=co.country_id
5   where upper(country)="CANADA"

```

first_name	email
DERRICK	DERRICK.BOURQUE@sakilacustomer.org
DARRELL	DARRELL.POWER@sakilacustomer.org
LORETTA	LORETTA.CARPENTER@sakilacustomer.org
CURTIS	CURTIS.IRBY@sakilacustomer.org
TROY	TROY.QUIGLEY@sakilacustomer.org

8. Quelles sont les ventes de chaque magasin pour chaque mois de 2005 ? (CONCAT)

```

select sto.store_id,concat(month(p.payment_date),"-",year(p.payment_date)) as month_year,
sum(p.amount) as "somme des ventes"
from store as sto join staff as sta on sto.store_id=sta.store_id
join payment as p on sta.staff_id=p.staff_id
where year(p.payment_date)=2005
group by month_year, sto.store_id
order by sto.store_id,month_year

```

```

1 • select sto.store_id,concat(month(p.payment_date),"-",year(p.payment_date)) as month_year, sum(p.amount) as "somme des ventes"
2   from store as sto join staff as sta on sto.store_id=sta.store_id
3   join payment as p on sta.staff_id=p.staff_id
4   where year(p.payment_date)=2005
5   group by month_year, sto.store_id
6   order by sto.store_id,month_year

```

store_id	month_year	somme des ventes
1	5-2005	2621.83
1	6-2005	4774.37
1	7-2005	13998.56
1	8-2005	11853.65
2	5-2005	2201.61
2	6-2005	4855.52
2	7-2005	14370.35
2	8-2005	12216.49

9. Trouvez le titre du film, le nom du client, le numéro de téléphone du client et l'adresse du client pour tous les DVD en circulation (qui n'ont pas prévu d'être rendus)

```

select f.title, concat(c.first_name," ",c.last_name) as nom_client,a.phone, a.address
from rental as r join inventory as i on r.inventory_id=i.inventory_id
join film as f on i.film_id=f.film_id
join customer as c on r.customer_id=c.customer_id
join address as a on c.address_id=a.address_id
where r.return_date is null

```

```

1 select f.title, concat(c.first_name," ",c.last_name) as nom_client,a.phone, a.address
2 from rental as r join inventory as i on r.inventory_id=i.inventory_id
3 join film as f on i.film_id=f.film_id
4 join customer as c on r.customer_id=c.customer_id
5 join address as a on c.address_id=a.address_id
6 where r.return_date is null

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	title	nom_client	phone	address
▶	HYDE DOCTOR	GAIL KNIGHT	904253967161	185 Novi Sad Place
	HUNGER ROOF	GREGORY MAULDIN	80303246192	507 Smolensk Loop
	FRISCO FORREST	LOUISE JENKINS	800716535041	929 Tallahassee Loop
	TITANS JERK	WILLIE HOWELL	991802825778	1244 Allappuzha (Alleppey) Place
	CONNECTION MICROCOSMOS	EMILY DIAZ	333339908719	588 Vila Velha Manor
	HAUNTED ANTI-TRUST	LAURIE LAWRENCE	956188728558	9 San Miguel de Tucumán Manor
	BULL SHAWSHANK	LISA ANDERSON	635297277345	1542 Tariac Parkway
	GHOST GROUNDHOG	FREDDIE DUGGAN	644021380889	1103 Quilmes Boulevard
	PEACH INNOCENT	HEATHER MORRIS	697760867968	17 Kabul Boulevard
	SUIT WALLS	ROLAND SOUTH	25865528181	1993 O Loop
	WOMEN DORADO	NATALIE MEYER	873492228462	1201 Qomsheh Manor
	GRAIL FRANKENSTEIN	SCOTT SHELLEY	165450987037	587 Benguela Manor
	ZHEVAGO CORE	LOUIS LEONE	45554316010	1191 Tandl Drive
	SONS INTERVIEW	CATHY SPENCER	819416131190	1287 Xi'angfan Boulevard
	ENOUGH RAGING	GUY BROWNLEE	978430786151	346 Cam Ranh Avenue
	TITANIC BOONDOCK	JUSTIN NGO	764680915323	519 Nyeri Manor
	GUNFIGHT MOON	JERRY JORDON	647899404952	124 al-Manama Way
	EL SEÑOR MONTECRO	ALYCE MULLER	761370480540	1062 Macau Place

2^{ème} partie : Test technique (type entreprise)

1. How can SQL queries be optimized?

Il permet d'optimiser le stockage et le rangement des données afin de les mobiliser plus rapidement à travers une vue

2. How do you remove duplicate rows from a table?

```
DELETE FROM table
WHERE (noms_colonnes) IN (
  SELECT noms_colonnes
  FROM table
  GROUP BY noms_colonnes
  HAVING COUNT(*) > 1
);
```

3. What are the main differences between HAVING and WHERE SQL clauses?

Le Having est un attribut après le groupby alors que le where est avant et permet de filtrer avant de grouper

4. What is the difference between normalization and denormalization?

La normalisation est utilisée pour éviter les redondances et optimiser le stockage des données qui se fait avec des grandes tables mais en plus petites tables. Ainsi on définit des relations entre ces tables et on a un schéma normalisé, permettant un stockage optimal des données

La dénormalisation vise à limiter le nombre de relation pour optimiser au maximum la performance des requêtes. Moins il y aura de relation, plus les requêtes seront performantes.

5. What are the key differences between the DELETE and TRUNCATE SQL commands?

La différence est dans le traitement qu'elles utilisent pour fonctionner. La fonction DELETE va supprimer les lignes une à une en tenant une journalisation des opérations alors que TRUNCATE est une commande par définition qui va effacer toutes les lignes d'un coup ? DELETE permet d'ailleurs de supprimer des lignes en fonction de caractéristiques alors que TRUNCATE non.

6. What are some ways to prevent duplicate entries when making a query?

La première est d'inclure DISTINCT dans le select pour ne pas avoir de doublons dans le résultat.

La deuxième est d'associer une clé primaire à chaque table, qui garantira l'unicité des lignes

La troisième peut être de regrouper les lignes que l'on souhaite et ainsi avoir un résultat unique avec le group by

7. What are the different types of relationships in SQL?

Relation 1-1 : Dans cette relation, chaque occurrence dans une table correspondra exactement à une autre occurrence dans une autre table. Cela signifie qu'il existe une correspondance unique entre les enregistrements des deux tables. Par exemple, une table "Utilisateur" pourrait avoir une relation 1-1 avec une table "Profil", où chaque utilisateur a un seul profil associé et chaque profil est lié à un seul utilisateur.

Relation 1-N : Dans cette relation, une occurrence dans une table peut être associée à plusieurs occurrences dans une autre table, mais chaque occurrence dans la seconde table est associée à une seule occurrence dans la première table. Par exemple, une table "Département" pourrait avoir une relation 1-N avec une table "Employé", où un département peut avoir plusieurs employés, mais chaque employé appartient à un seul département.

Relation N-N : Dans cette relation, plusieurs occurrences dans une table peuvent être associées à plusieurs occurrences dans une autre table. Cela signifie qu'il n'y a pas de contrainte sur le nombre d'occurrences associées entre les deux tables. Par exemple, une table "Étudiant" pourrait avoir une relation N-N avec une table "Cours", où un étudiant peut suivre plusieurs cours et un cours peut avoir plusieurs étudiants inscrits. Pour modéliser cette relation, une table de jointure intermédiaire est généralement utilisée pour mapper les associations entre les deux tables.

8. Give an example of the SQL code that will insert the 'Input data' into the two tables. You must ensure that the student table includes the correct [dbo].[Master].[id] in the [dbo].[student].[Master_id] column.

Insertion des données dans la table "subject"

```
INSERT INTO subject (subject_id, subject_name, max_score, lecturer)
VALUES
```

```
(11, 'Math', 130, 'Charlie Sole'),
(12, 'Computer Science', 50, 'James Pillet'),
(13, 'Biology', 300, 'Carol Denby'),
(14, 'Geography', 220, 'Yollanda Balang'),
(15, 'Physics', 110, 'Chris Brother'),
(16, 'Chemistry', 400, 'Manny Donne');
```

Insertion des données dans la table "student"

```
INSERT INTO student (student_id, student_name, city, subject_id)
VALUES
```

```
(2001, 'Olga Thorn', 'New York', 11),
(2002, 'Sharda Clement', 'San Francisco', 12),
(2003, 'Bruce Shelkins', 'New York', 13),
(2004, 'Fabian Johnson', 'Boston', 15),
(2005, 'Bradley Camer', 'Stanford', 11),
(2006, 'Sofia Mueller', 'Boston', 16),
(2007, 'Rory Pietman', 'New Haven', 12),
(2008, 'Carly Walsh', 'Tulsa', 14),
(2011, 'Richard Curtis', 'Boston', 11),
(2012, 'Cassey Ledgers', 'Stanford', 11),
(2013, 'Harold Ledgers', 'Miami', 13),
(2014, 'Davey Bergman', 'San Francisco', 12),
(2015, 'Darcey Button', 'Chicago', 14);
```

Then give an example of the SQL code that shows courses', subject names, and the number of students taking the course **only** if the course has three or more students on the course.

Table: subject			
subject_id	subject_name	max_score	lecturer
11	Math	130	Charlie Sole
12	Computer Science	50	James Pillet
13	Biology	300	Carol Denby
14	Geography	220	Yollanda Balang
15	Physics	110	Chris Brother
16	Chemistry	400	Manny Donne

Table: student			
student_id	student_name	city	subject_id
2001	Olga Thorn	New York	11
2002	Sharda Clement	San Francisco	12
2003	Bruce Shelkins	New York	13
2004	Fabian Johnson	Boston	15
2005	Bradley Camer	Stanford	11
2006	Sofia Mueller	Boston	16
2007	Rory Pietman	New Haven	12
2008	Carly Walsh	Tulsa	14
2011	Richard Curtis	Boston	11
2012	Cassey Ledgers	Stanford	11
2013	Harold Ledgers	Miami	13
2014	Davey Bergman	San Francisco	12

2015	Darcey Button	Chicago	14
------	---------------	---------	----

9. Write a query to retrieve the `order_id`, `customer_id`, and `total` from the `orders` table where the `total` is greater than 400.

Select `order_id`, `customer_id`, `total`

From `orders`

Where `total > 400`

Then do a query to retrieve the `customer_id` and the total amount spent by each customer from the `orders` table, ordered by the total amount spent in descending order.

select `customer_id`, `sum(total)` as `total_spent`

from `orders`

group by `customer_id`

order by `total_spent` desc

Table: Orders

order_id	customer_id	order_date	total
1	100	01/01/2021	200
2	101	02/02/2021	300
3	102	03/03/2021	400
4	103	04/04/2021	500
5	104	05/05/2021	600

Table: Order items

order_id	product_id	quantity	price
1	10	2	50
1	11	3	25
2	12	4	30
2	13	5	20
3	14	6	15
3	15	7	10
4	16	8	5
4	17	9	4
5	18	10	3
5	19	11	2

10. Write a query that shows the total quantity sold for each product.

Select product_id, sum(quantity) as "total_quantity"
 From "Order items"
 Groupby product_id

Table: Order items

order_id	order_date	customer_id	product_id	quantity
1	01/01/2022	101	1	2
2	01/01/2022	102	1	1
3	01/01/2022	103	2	5
4	02/01/2022	104	3	3
5	02/01/2022	105	1	2
6	02/01/2022	101	3	1
7	03/01/2022	102	2	4
8	03/01/2022	103	1	2
9	03/01/2022	104	2	1
10	04/01/2022	105	3	2

11. Assume we have a large excel spreadsheet with customer orders data. Each row contains information about a single order, including the customer name, order date, order ID, order quantity, and order total. We want to divide this data into three tables: Customers, Orders, and OrderDetails. Customers will store customer information, Orders will store order information (including customer ID), and OrderDetails will store details about individual order items (including order ID).

Customers:

id	name	address	city	country
1	John Smith	123 Main St.	Anytown	USA
2	Jane Doe	456 Oak St.	Somewhere	USA
3	Bob Johnson	789 Pine St.	Anytown	USA
4	Alice Lee	1010 Elm St.	Nowhere	USA
5	David Kim	1234 Maple St.	Anytown	USA

Orders:

id	customer_id	order_date	total
1	1	01/01/2022	100
2	1	02/01/2022	150
3	2	03/01/2022	75
4	3	04/01/2022	200
5	4	05/01/2022	50

OrderDetails:

id	order_id	product	quantity	price
1	1	Widget A	2	25
2	1	Widget B	1	50
3	2	Widget C	1	75
4	2	Widget D	2	37.5
5	3	Widget A	1	25
6	3	Widget B	2	50
7	4	Widget D	1	200
8	5	Widget A	2	25

We want to insert the customer orders data into the three tables Customers, Orders, and OrderDetails. Write an SQL query that inserts the data into the appropriate tables, and ensures that the customer ID and order ID are maintained across all three tables. The Orders table should have a foreign key reference to the Customers table, and the OrderDetails table should have a foreign key reference to the Orders table. Assume that the source data is stored in a single table named 'customer_orders', and that the schema for each destination table is already defined.

Insertion des clients

```
insert into customers (name, address, city, country)
```

```
select distinct
```

```
customer_name,
customer_address,
customer_city,
customer_country
```

```
from customer_orders;
```

Insertion des commandes en liaison avec les clients

```
insert into orders (customer_id, order_date, total)
```

```
select
```

```
  c.id,
```

```
  order_date,
```

```
  total
```

```
from customer_orders co
```

```
inner join customers c on co.customer_name = c.name
```

```
  and co.customer_address = c.address
```

```
  and co.customer_city = c.city
```

```
  and co.customer_country = c.country;
```

Insertion des details des commandes

```
insert into orderdetails (order_id, product, quantity, price)
```

```
select
```

```
  o.id,
```

```
  product,
```

```
  quantity,
```

```
  price
```

```
from customer_orders co
```

```
inner join orders o on co.customer_name = (select name from customers where id = o.customer_id)
```

```
  and co.order_date = o.order_date
```

```
  and co.total = o.total;
```