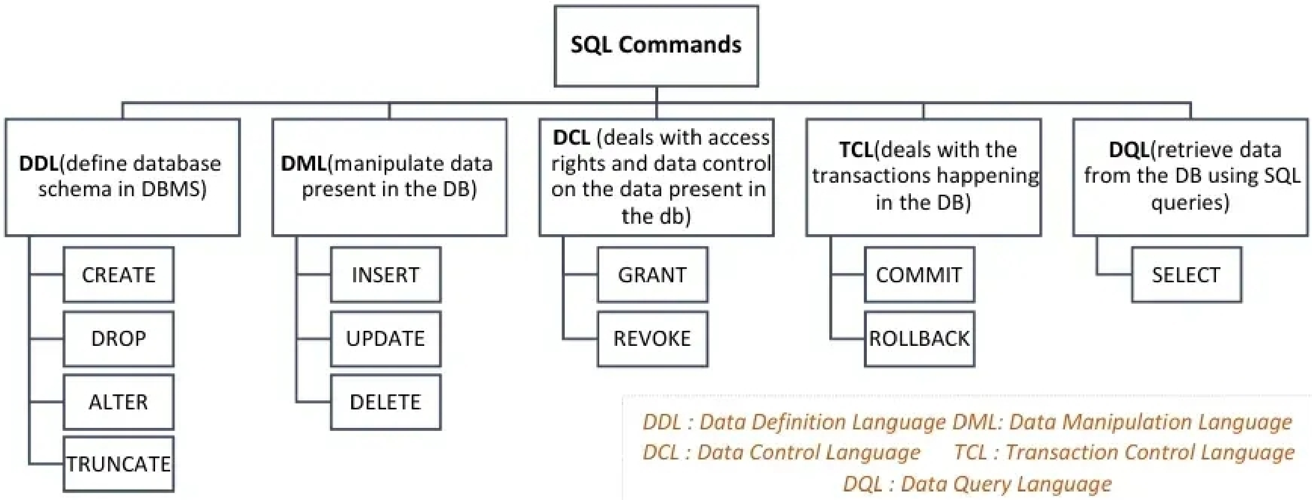
Structured Query language (SQL)



SQL Commands

DDL(define database schema in DBMS)
- CREATE
- DROP
- ALTER
- TRUNCATE

DML(manipulate data present in the DB)
- INSERT
- UPDATE
- DELETE

DCL (deals with access rights and data control on the data present in the db)
- GRANT
- REVOKE

TCL(deals with the transactions happening in the DB)
- COMMIT
- ROLLBACK

DQL(retrieve data from the DB using SQL queries)
- SELECT

DDL : Data Definition Language DML: Data Manipulation Language
DCL : Data Control Language TCL : Transaction Control Language
DQL : Data Query Language

1. Create database	create database sample2
2. Use the database	use sample2
3. Create table	create table customer (customerid int identity(1,1) primary key, customernumber int not null unique check (customernumber>0), lastname varchar(30) not null, firstname varchar(30) not null, areacode int default 71000, address varchar(50), country varchar(50) default 'Malaysia')
4. Insert values into table	insert into customer values (100,'Fang Ying','Sham','418999','sdadasfdfd',default), (200,'Mei Mei','Tan',default,'adssdsadsd','Thailand'), (300,'Albert','John',default,'dfdsfsdf',default)
5. Display record from table	-- display all records select * from customer -- display particular columns select customerid, customernumber, lastname, firstname from customer
6. Add new column to table	alter table customer add phonenumber varchar(20)

7. Add values to newly added column/ Update table	update customer set phonenumber='1234545346' where
	customerid=1
	update customer set phonenumber='45554654' where
	customerid=2

8. Delete a column	alter table customer drop column phonenumber
9. Delete record from table --if not put ‘where’, will delete all record	delete from customer where country='Thailand'
10. Delete table	drop table customer
11. Change data type	alter table customer alter column phonenumber varchar(10)

1. Create database	create database SaleOrder use SaleOrder
2. Use the database	create table dbo.customer (
3. Create tables	CustomerID int NOT null primary key, CustomerFirstName varchar(50) NOT null, CustomerLastName varchar(50) NOT null, CustomerAddress varchar(50) NOT null, CustomerSuburb varchar(50) null, CustomerCity varchar(50) NOT null, CustomerPostCode char(4) null, CustomerPhoneNumber char(12) null,); create table dbo.inventory (InventoryID tinyint NOT null primary key, InventoryName varchar(50) NOT null, InventoryDescription varchar(255) null,); create table dbo.employee (EmployeeID tinyint NOT null primary key, EmployeeFirstName varchar(50) NOT null, EmployeeLastName varchar(50) NOT null, EmployeeExtension char(4) null,); create table dbo.sale (SaleID tinyint not null primary key, CustomerID int not null references customer(CustomerID), InventoryID tinyint not null references Inventory(InventoryID), EmployeeID tinyint not null references Employee(EmployeeID), SaleDate date not null, SaleQuantity int not null, SaleUnitPrice smallmoney not null);
4. Check what table inside	select * from information_schema.tables
5. View specific row	--top: show only the first two select top 2 * from customer --top 40 percent: also means show the first two select top 40 percent * from customer
6. View specific column	--sort result (by default is ascending) select customerfirstname, customerlastname from customer order by customerlastname desc select customerfirstname, customerlastname from customer order by 4, 2, 3 desc -- Order By Based on column no. without typing column name --distinct: only show unique value select distinct customerlastname from customer order by customerlastname

7. Save table to another table	--into file_name: save result in another table (BASE TABLE) select distinct customerlastname into temp from customer order by customerlastname select * from temp --see the table (data type will remain)
8. Like (search something)	-- (underscore sign) _ is only specific for **one character** only -- (percent sign) % represents zero, one, or **multiple characters** select * from customer where customerlastname like '_r%'
9. In (search something)	-- search multiple items select * from customer where customerlastname in ('Brown', 'Michael', 'Jim')
10. > (search something)	select * from customer where customerlastname > 'Brown' or customerlastname>'Cross'
11. <> (Not Equal)	select * from customer where customerlastname <> 'Brown'
12. IS NULL	-- check null values select * from customer where customerlastname IS NULL
13. IS NOT NULL	select * from customer where customerlastname IS NOT NULL
14. between	select * from sale where saleunitprice between 5 and 10 --not include 5 & 10
15. count	-- returns the number of rows in a table -- AS means aliasing, temporary giving name to a column/ table select count(*) as [Number of Records] from customer where customerfirstname like 'B%'
16. sum	select sale.employeeid ,EmployeeFirstName, EmployeeLastName , count(*) as [Number of order] , sum(salequantity) as [Total Quantity] from sale,employee where sale.employeeid = employee.employeeid group by sale.employeeid ,EmployeeFirstName, EmployeeLastName
17. count month	select month(saledate) as [Month], count (*) as [Number of sale], sum(salequantity*saleunitprice) as [Total Amount] from sale group by month(saledate)

18. max	SELECT MAX(Salary)
	FROM EmployeeSalary

19. min	SELECT MIN(Salary)
	FROM EmployeeSalary

20. average	SELECT AVG(Salary)
	FROM EmployeeSalary

21. having

```
SELECT JobTitle, COUNT(JobTitle)
FROM EmployeeDemographics ED
JOIN EmployeeSalary ES
        ON ED.EmployeeID = ES.EmployeeID
GROUP   BY   JobTitle   HAVING
COUNT(JobTitle) > 1

SELECT JobTitle, AVG(Salary)

FROM EmployeeDemographics ED
JOIN EmployeeSalary ES

        ON ED.EmployeeID = ES.EmployeeID
GROUP BY JobTitle
HAVING AVG(Salary) > 45000
ORDER BY AVG(Salary)
```

22. Change data type
    temporary for use

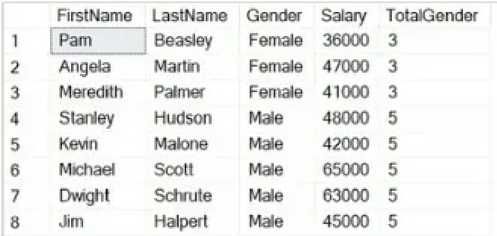
```
-- CAST(expression AS datatype(length))
SELECT CAST('2017-08-25 00:00:00.000' AS date)

-- CONVERT(data_type(length), expression, style)

SELECT CONVERT(date,'2017-08-25 00:00:00.000')
```

23. CASE Statement

```
SELECT FirstName, LastName, Age,
CASE

   WHEN Age > 30 THEN 'Old'
   WHEN Age BETWEEN 27 AND 30 THEN 'Young'
   ELSE 'Baby'
END
FROM EmployeeDemographics ED
WHERE Age IS NOT NULL
ORDER BY Age


--

SELECT FirstName, LastName, JobTitle, Salary,
CASE
   WHEN JobTitle = 'Salesman' THEN Salary + (Salary *.10)
   WHEN JobTitle = 'Accountant' THEN Salary + (Salary *.05)
   WHEN JobTitle = 'HR' THEN Salary + (Salary *.000001)
   ELSE Salary + (Salary *.03)
END AS SalaryAfterRaise
FROM EmployeeDemographics ED
JOIN EmployeeSalary ES
ON ED.EmployeeID = ES.EmployeeID
```

24. Partition By
--returns a single value for each row

```
SELECT FirstName, LastName, Gender, Salary,

COUNT(Gender) OVER (PARTITION BY Gender) AS TotalGender
FROM EmployeeDemographics ED
JOIN EmployeeSalary ES
ON ED.EmployeeID = ES.EmployeeID
```



	FirstName	LastName	Gender	Salary	TotalGender
1	Pam	Beasley	Female	36000	3
2	Angela	Martin	Female	47000	3
3	Meredith	Palmer	Female	41000	3
4	Stanley	Hudson	Male	48000	5
5	Kevin	Malone	Male	42000	5
6	Michael	Scott	Male	65000	5
7	Dwight	Schrute	Male	63000	5
8	Jim	Halpert	Male	45000	5

25. String Functions

```
-- Remove space
Select EmployeeID, TRIM(EmployeeID) AS IDTRIM
FROM EmployeeErrors

Select EmployeeID, RTRIM(EmployeeID) as IDRTRIM
FROM EmployeeErrors
Select EmployeeID, LTRIM(EmployeeID) as IDLTRIM
FROM EmployeeErrors
-- Replace
Select LastName, REPLACE(LastName, '- Fired', '') as
LastNameFixed
FROM EmployeeErrors
-- Substring
Select Substring(err.FirstName,1,3),
Substring(dem.FirstName,1,3), Substring(err.LastName,1,3),
Substring(dem.LastName,1,3)
FROM EmployeeErrors err
JOIN EmployeeDemographics dem


      on Substring(err.FirstName,1,3) =
Substring(dem.FirstName,1,3)
      and Substring(err.LastName,1,3) =
Substring(dem.LastName,1,3)
-- UPPER and LOWER CASE
Select firstname, LOWER(firstname)
from EmployeeErrors

Select Firstname, UPPER(FirstName)
from EmployeeErrors"
```

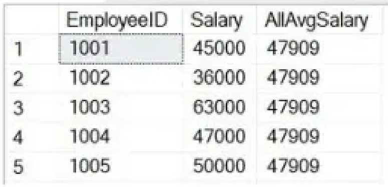
26. Stored Procedure

```
CREATE PROCEDURE Temp_Employee
@JobTitle nvarchar(100)
AS
DROP TABLE IF EXISTS #temp_employee
Create table #temp_employee (
JobTitle varchar(100),
EmployeesPerJob int ,
AvgAge int,
AvgSalary int
)
Insert into #temp_employee
SELECT JobTitle, Count(JobTitle), Avg(Age), AVG(salary)
FROM EmployeeDemographics emp
JOIN EmployeeSalary sal

        ON emp.EmployeeID = sal.EmployeeID
where JobTitle = @JobTitle --- make sure to change this in
this script from original above
group by JobTitle
Select *
From #temp_employee
GO;
```

```
--- only need to run this on next time
EXEC Temp_Employee @JobTitle = 'Salesman'
```

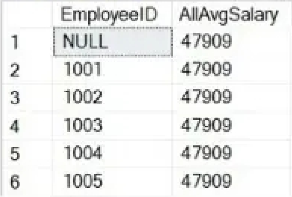
27. Subquery

```
-- Subquery in Select

SELECT EmployeeID, Salary, (SELECT AVG(Salary) FROM
EmployeeSalary) AS AllAvgSalary
FROM EmployeeSalary

-- with Partition By

SELECT EmployeeID, Salary, AVG(Salary) OVER () AS
AllAvgSalary
FROM EmployeeSalary
```



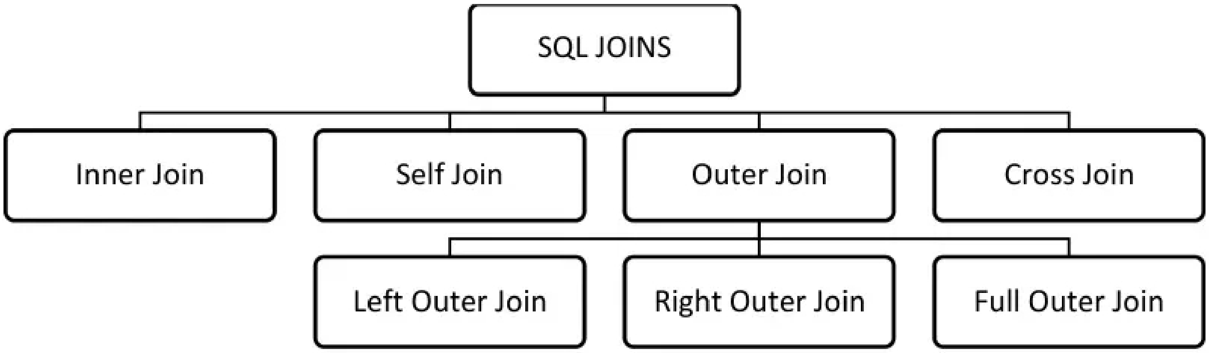
	EmployeeID	Salary	AllAvgSalary
1	1001	45000	47909
2	1002	36000	47909
3	1003	63000	47909
4	1004	47000	47909
5	1005	50000	47909

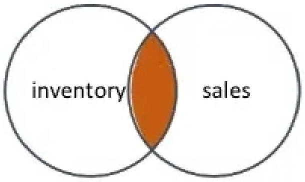
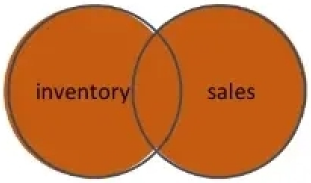
```
-- Subquery in From

SELECT a.EmployeeID, AllAvgSalary
FROM (SELECT EmployeeID, Salary, AVG(Salary) OVER () AS
AllAvgSalary
         FROM EmployeeSalary) a
ORDER BY a.EmployeeID
```



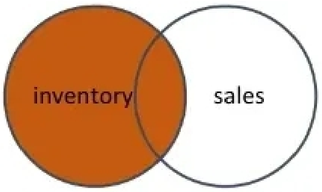
	EmployeeID	AllAvgSalary
1	NULL	47909
2	1001	47909
3	1002	47909
4	1003	47909
5	1004	47909
6	1005	47909

```
-- Subquery in Where

SELECT EmployeeID, JobTitle, Salary
FROM EmployeeSalary
WHERE EmployeeID in (SELECT EmployeeID FROM
EmployeeDemographics
                     WHERE Age > 30)

SELECT EmployeeID, JobTitle, Salary

FROM EmployeeSalary
WHERE Salary in (SELECT Max(Salary) FROM EmployeeSalary)
```

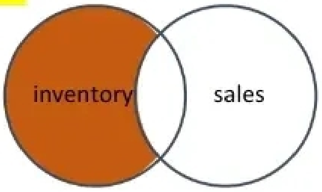


1. getting data from multiple tables (explicit join - without using join command)	select * from inventory,sale where sale.inventoryid=inventory.inventoryid
	select inventoryname,saledate,saleunitprice,salequantity,salequantity*saleunitprice as [Total amount] from sale,inventory where sale.inventoryid=inventory.inventoryid group by sale.inventoryid,inventoryname,saledate,salequantity,saleunitprice order by inventoryname
2. getting data from multiple tables (implicit join - using join command)	--inner join select * from inventory inner join sale on sale.inventoryid=inventory.inventoryid select inventoryname,saledate,saleunitprice,salequantity,saleunitprice*salequantity as [Total Amount] from inventory inner join sale on sale.inventoryid=inventory.inventoryid order by inventoryname 
	--full outer join (shows everything) select sale.inventoryid,inventoryname from inventory full outer join sale on sale.inventoryid=inventory.inventoryid where sale.inventoryid is NULL 

```
--left join (might have NULL value, since some inventory might not have sales)
select inventory.inventoryid,inventoryname
from inventory left join sale on
sale.inventoryid=inventory.inventoryid
```

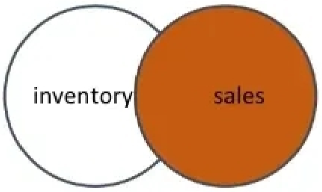


```
--left join
select inventory.inventoryid,inventoryname
from inventory left join sale on
sale.inventoryid=inventory.inventoryid
where sale.inventoryid is NULL
```



```
-- without join: use subquery
select inventoryid,inventoryname from inventory
where inventoryid not in (select inventoryid from sale)
```

```
--right join
select sale.inventoryid,inventoryname
from inventory right join sale on
sale.inventoryid=inventory.inventoryid
```



3. Self Join

--commonly used in processing hierarchy

--inner join

Staff Table

employeeID	employeefirstname	employeelastname	managerID
1001	Tan	Mei Ling	NULL
1002	Kelvin	Koh	1001
1003	Amin	Wong	1002

```
select E.employeeID, E.employeefirstname+' '+E.employeelastname as [Full
Name], E.managerID, , M.employeefirstname+' '+M.employeelastname as
[Manager Name]
from staff E
inner join staff M
on E.managerID = M.employeeID
```

Output:

employeeID	Full Name	managerID	managerName
1002	Kelvin Koh	1001	Tan Mei Ling
1003	Amin Wong	1002	Kelvin Koh

--left outer join (list all the employees)

```
select E.employeeID, E.employeefirstname+' '+E.employeelastname as [F
Name], E.managerID, , M.employeefirstname+' '+M.employeelastname as
[Manager Name]
from staff E
left outer join staff M
on E.managerID = M.employeeID
```

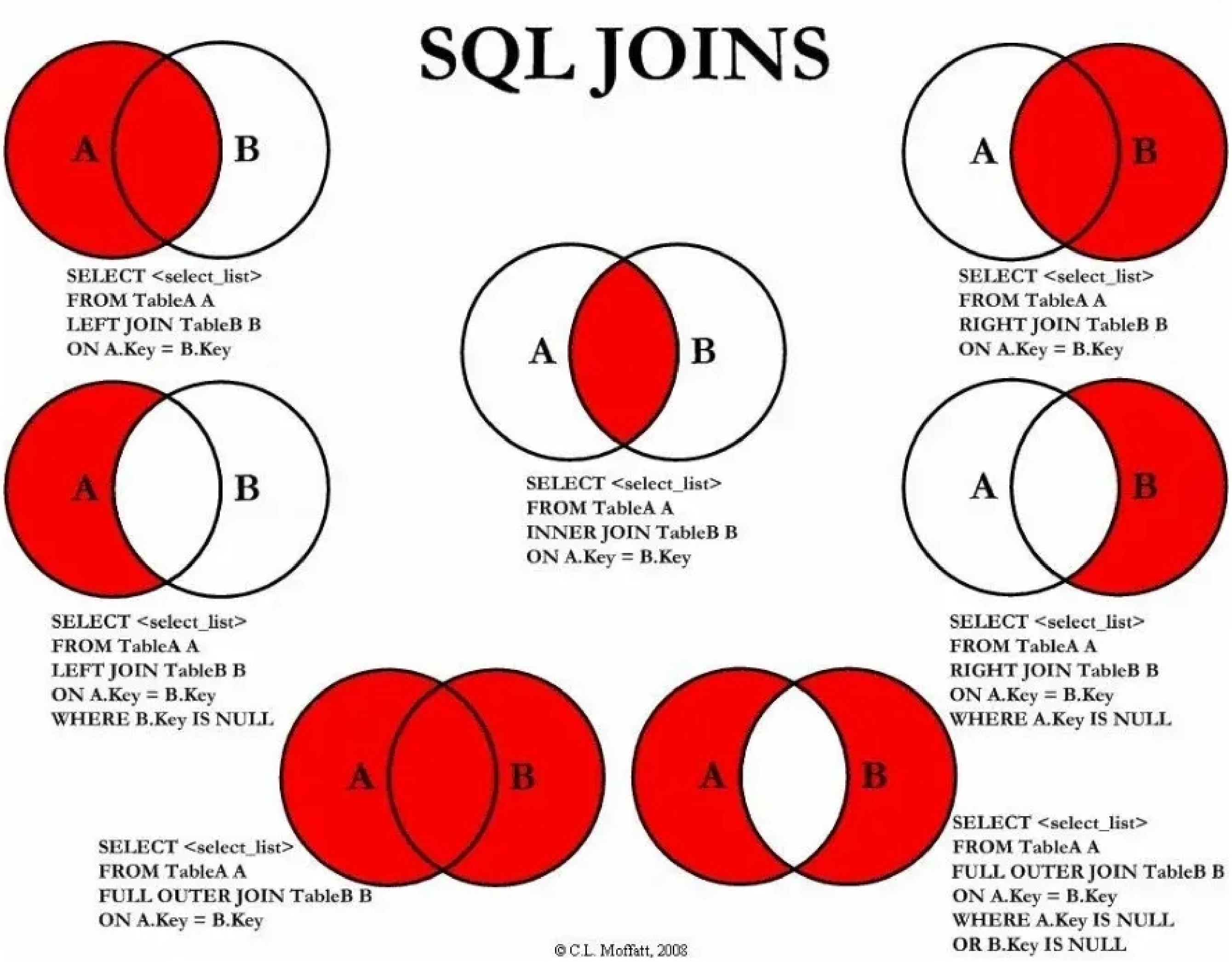
Output:

employeeID	Full Name	managerID	managerName
1001	Tan Mei Ling		
1002	Kelvin Koh	1001	Tan Mei Ling
1003	Amin Wong	1002	Kelvin Koh

4. Cross Join

--generate all combination of records (all possibility) (Cartesian Product)

```
select * from inventory1
cross join inventory2
```



SQL JOINS

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

SQL UNIONS

1. Union

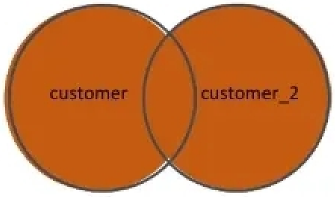
--allow you to combine two tables together (but the no. of columns & each column's data types for 2 tables must be match)
--don't need common key, only need common attributes
--merge, not showing duplicate record

```
select cust_lname,cust_fname from customer
union
select cust_lname,cust_fname from customer_2
```

2. Union all

--merge, but show you everything, even the duplicate record

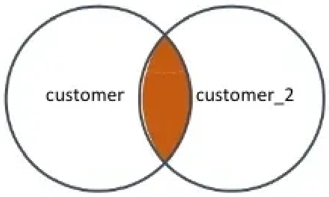
```
select cust_lname,cust_fname from customer
union all
select cust_lname,cust_fname from customer_2
```



3. Intersect

--keep only the rows in common to both query
--not showing duplicate record

```
select cust_lname,cust_fname from customer
intersect
select cust_lname,cust_fname from customer_2
```

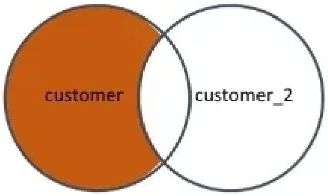


```
select c.cust_lname,c.cust_fname from customer c,customer_2 c2
where c.cust_lname=c2.cust_lname and c.cust_fname=c2.cust_fname
```

4. Except

--generate only the records that are unique to
the CUSTOMER table

```
select cust_lname,cust_fname from customer
except
select cust_lname,cust_fname from customer_2
```



```
--use subquery
select cust_lname,cust_fname from customer
where(cust_lname) not in
(select cust_lname from customer_2) and
(cust_fname) not in
(select cust_fname from customer_2)
```