```
/*Select*/
use Padhai;

/*select */column1,column2...*/
select * from
FilmLocations;

select Title,[Release Year]
from
FilmLocations;

/*select-dictinct*/
select distinct Title,[Release Year]
from
FilmLocations
where Title like '_r%';

/*Select-count*/
select count(distinct Title)
from
FilmLocations;

/*Select-top('limit' at the end for mysql)*/
select distinct top 50
percent Title,[Release Year]
from FilmLocations
where Title like 'T%';

select top 10
*
from FilmLocations;

Select Distinct TOP 10 ,
       FirstName + ' ' + LastName AS
FullName
FROM   Person.Person
ORDER BY LastName

/*order by*/
select distinct
Title,[Release Year]
from FilmLocations
order by [Release Year] asc; /*if select distinct is
used then the order by column must be present in the first line,here it is [Release
Year]*/


/*Inbuilt Functions*/
/*min()/max()*//*avg(),sum()*//*Round,
datalength,upper,lower */
select min(Title) as minimumtitle from FilmLocations;

select
max([Release Year]) as maximumReleaseYear from FilmLocations;

select avg([Release Year]) as
AvgRealeaseYear from FilmLocations;

select sum([Release Year]) as SumRealeaseYear from
FilmLocations;


SELECT ROUND([Release Year],-1) AS RoundValue
from FilmLocations
where
[Release Year]>1950

select DATALENGTH(Title)  from FilmLocations;
select Lower(Title)
from FilmLocations;
select UPPER(Title) from FilmLocations;
```

```
/*Aliases-as*/
select
Title,[Actor 1]+','+[Actor 2]+','+[Actor 3] as Actors
from FilmLocations
where Title like
'G%';

/*Subqueries+(ANY,SOME,ALL,EXISTS)*/
--https://www.youtube.com/watch?v=nD7JAdEQYAE&ab_channel=WeLearnSQL
select distinct
Title,[Release Year]
from FilmLocations
where [Release Year]>any(select ([Release Year])
from FilmLocations where [Release Year]> 2000 ) --used any here with subquery, RY will be
retrived > any of the (values in subquery)
order by [Release Year];

select distinct
Title,[Release Year]
from FilmLocations
where [Release Year]>all(select ([Release Year])
from FilmLocations where [Release Year] between 1999 and 2009 ) --used all here with subquery,
RY will be retrived > all of the (values in subquery)
order by [Release Year];

 /*Stored
procedures*/
 --example1
CREATE PROCEDURE countoftitles
as
select count(distinct
Title)
from FilmLocations
go;
exec countoftitles;
--example2,parameter passing
create
procedure ReleaseYearSpecifics @releaseyear int,@Titlename varchar(500)
as
select *
from
FilmLocations
where ([Release Year]>@releaseyear) and (Title like @Titlename)
go
exec
ReleaseYearSpecifics @releaseyear=1947,@Titlename='Gr%'

/*Union*combine columns--must have
same number,type,name of columns in both select*/
select * from table1
union
select * from
table2

/*Groupby-useful if select has count,min,max,sum,avg*/
select count(distinct Title)
as counttitle,[Release Year]
from FilmLocations
where [Release Year]>1960
Group by
[Release Year]

/*Having*-used mostly after group-by or after a function where 'where' clause
is not possible to use because of its(where clause) restriction*/
select count(distinct Title)
as counttitle,[Release Year]
from FilmLocations
where [Release Year]>1960
Group by
[Release Year]
Having count(distinct Title)>3
order by count(distinct Title)
asc;
```

```
/*'select into newtable'*--here the columns with its data is copied from oldtable and
transffered to newtable -u cannot update an existing table*/
select *
into FilmLocations2

from FilmLocations
where [Release Year] between 2010 and 2015

delete
Filmlocations2;

/*insert into newtable*--here u are updating an existing table*/
insert
into FilmLocations2
select *
from FilmLocations
where [Release Year]>2017

select *
from FilmLocations2

/*View*/--It creates view of a temporary table.--this temporary table
actually doesn't exist and we can play with it.
create view RY2000
as
select distinct Title

from FilmLocations
where [Release Year]>2000

select *
from
RY2000


/*INDEX**--Updating a table with indexes takes more time than updating a table
without (because the indexes also need an update).
So, only create indexes on columns that
will be frequently searched against.*/

create index index_column  --we can use 'create
unique index' to avoid duplicate values
on
FilmLocations(Title,[Release Year]);

select *
from FilmLocations
where Title like 'G%';

DROP INDEX
FilmLocations.index_column;


/*Subqueries*/
--in where clause
select distinct(Title) from
FilmLocations
where [Release Year]>(select avg([Release Year]) from FilmLocations);
--in
select columns
select Title,[Release Year],(select avg([Release Year]) from FilmLocations) as
avgry from FilmLocations;
--in from clause
select * from (select distinct(Title),[Release
Year] from FilmLocations) as emp ;


/*Mathematical functions*/
--select round(decimal
values),floor(decimal values),ceiling(decimal values)
--round--> 133.53--->
134
--floor--> 34.34---->  34
--ceiling->35.3----->  36
```

```
/*String functions*/
--1>concat(joins two column),2>trim(removes spaces),3>extract(shows only selected
letters)
--1>select concat(column1,column2) ...
--2>select
ltrim(column),rtrim(column),trim(column) ...
--3>select left(column,2) , right(column,4)
...

/*Date and Time Data Types
Data type        Description
DATE        A date. Format: YYYY-MM-DD. The
supported range is from '1000-01-01' to '9999-12-31'
DATETIME(fsp)        A date and time
combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to
'9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic
initialization and updating to the current date and time
TIMESTAMP(fsp)        A timestamp. TIMESTAMP
values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC).
Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to
'2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time
can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column
definition
TIME(fsp)        A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to
'838:59:59'
YEAR        A year in four-digit format. Values allowed in four-digit format: 1901 to
2155, and 0000.
MySQL 8.0 does not support year in two-digit format.*/

--if a column has a
date datatype (lets that column be D)
--following date inbuilt funtions are
useful
--Day(D)
--week(D),Month(D),Year(D)
--Dayofweek(D),dayofmonth(D),dayofyear(D)
--usef
ul command ----->                    select X + 3 days from tablename


--if a column has
a Time datatype (lets that column be T)
--following date inbuilt funtions are
useful
--Second(T),minute(T),hour(T)

--special registers-> current_date ,
current_time
--select current_date-X from table
--curdate(),now(),curtime()

use
padhai;
create table tablefordate
(rescuedate date,
rescuetime time
);

insert into
tablefordate
(rescuedate,rescuetime)
values ('1997-05-17','11:25:58');

select
(GETDATE()-rescuedate) from tablefordate

select now from FilmLocations;

/*Practice
important codes*/
select * from Customers;
select sum(PostalCode) from Customers where
CustomerID=1 or CustomerID=2 or CustomerID=3;
select sum(PostalCode) from Customers where
CustomerID in (1,2,3);
```

```
select * from Customers where CustomerID between 1 and 2;
select
distinct CustomerName
from Customers
order by CustomerName;

select * from Customers where
CustomerID not in (1,2);
```