```sql
-- Comment one line at a time
 /* This is how you comment
 multiple lines at a time
 Got it!! */

 -- You can use the SELECT statement to perform a simple calculation as follows:

SELECT
 1 + 4 as sum1;
 -- SOLVE - ? How do you give name to column here?




 -- You can use multiple expressions in the SELECT statement as follows:

 SELECT
 10 / 5 as div1 , 2 * 4 as mul1;

 select
 trackid, name, albumid, unitprice
 from tracks;

 -- Select a few columns (trackid, name, composer and unitprice) from tracks table in DB

SELECT
 trackid,
 name,
 composer,
 unitprice
FROM
 tracks;
 /* SOLVE- ? How to select All the columns from tracks table?
 - Way/s - Which one is easier and which is recommended? */
 select *
 from tracks;



-- Get the cities from where customer belongs

SELECT
 city
FROM
 customers;
 -- SOLVE - ? What is the issue with above query?





 -- Use of DISTINCT
SELECT DISTINCT
 city
FROM
 customers;

 -- SOLVE - ? Get the companies of the customers

SELECT
 company
FROM
 customers;
  -- What do you observe in result set?
 -- Use DISTINCT
SELECT DISTINCT
 company
FROM
 customers;
 -- Again, what do you observe?



 -- USE OF ORDER BY
 -- Get name, milliseconds, and album id columns form tracks
 select name, milliseconds, albumid
 from tracks
 order by 2 DESC;

SELECT
 name,
 milliseconds,
        albumid
FROM
 tracks;


 -- Now get the same data sorted based on AlbumId column in ascending order

SELECT
 name,
 milliseconds,
 albumid
FROM
 tracks
ORDER BY
 albumid ;
```

```
-- SOLVE:- Now sort the sorted result (by AlbumId) above...
-- by the Milliseconds column in descending order

SELECT
 name,
 milliseconds,
 albumid
FROM
 tracks
ORDER BY
 albumid ASC,
        milliseconds DESC;

-- Using column positions in ORDER BY
SELECT
 name,
 milliseconds,
 albumid
FROM
 tracks
ORDER BY
 3,2;

-- Use of LIMIT clause
-- get the first 10 rows with track id and track name from the tracks table

SELECT
 trackId,
 name
FROM
 tracks
LIMIT 10;
```

```
-- get 10 rows starting from the 10th row in the tracks table

SELECT
 trackId,
 name
FROM
 tracks
LIMIT 10 OFFSET 10;
```

```
-- get the top 10 largest tracks in bytes
SELECT
 trackid,
 name,
 bytes
FROM
 tracks
ORDER BY
 bytes DESC
LIMIT 10;
```

```
-- USE of WHERE
-- SOLVE - ? Get all customers and their first name where company is missing

select customerid, firstname, company
from customers
where company is NULL;

-- Get all tracks in the album id 1
```

```
SELECT
 name,
 milliseconds,
  bytes,
 albumid
FROM
 tracks
WHERE
 albumid = 1;




 -- Get tracks on the album 1 that have the length greater than 250,000 milliseconds

SELECT
 name,
 milliseconds,
 bytes,
 albumid
FROM
 tracks
WHERE
 albumid = 1
AND milliseconds > 250000;

-- Find which tracks are composed by all people with 'Smith' in thier names

SELECT
 name,
 albumid,
 composer
FROM
```

```sql
 tracks
WHERE
 composer LIKE '%Smith%'
ORDER BY
 albumid;




 -- Find tracks that have media type id 2 or 3

 SELECT
 name,
 albumid,
 mediatypeid
FROM
 tracks
WHERE
 mediatypeid = 2 or mediatypeid = 3;



 -- You can achieve the same result using the IN operator

SELECT
 name, albumid, mediatypeid
FROM
 tracks
WHERE
 mediatypeid IN (2, 3);
```

```
-- list of tracks whose genre id is not in a list of (1,2,3)

SELECT
 trackid,
 name,
 genreid
FROM
 tracks
WHERE
 genreid NOT IN (1,2,3);
```

```
 -- Use of BETWEEN operator
 -- finds invoices whose total is between 14.91 and 18.86:

SELECT
    InvoiceId,
    BillingAddress,
    Total
FROM
    invoices
WHERE
    Total BETWEEN 14.91 and 18.86
ORDER BY
    Total;
```

```
-- find the invoices whose total are not between 1 and 20
SELECT
    InvoiceId,
    BillingAddress,
    Total
FROM
    invoices
WHERE
    Total NOT BETWEEN 1 and 20
ORDER BY
    Total;



-- finds invoices whose invoice dates are from January 1 2010 and January 31 2010:
SELECT
    InvoiceId,
    BillingAddress,
    InvoiceDate,
    Total
FROM
    invoices
WHERE
    InvoiceDate BETWEEN '2010-01-01' AND '2010-01-31'
ORDER BY
    InvoiceDate;



-- Use of wildcards % and _
-- ind the tracks with 'Wild' in their name
SELECT
 trackid,
 name
FROM
 tracks
WHERE
```

```
 name LIKE 'Wild%';



SELECT
 trackid,
 name
FROM
 tracks
WHERE
 name LIKE '%Wild';



SELECT
 trackid,
 name
FROM
 tracks
WHERE
 name LIKE '%Wild%';


 -- GROUP BY
 -- Find the number of tracks per album

SELECT
 albumid,
 COUNT(trackid) as track_count
FROM
 tracks
GROUP BY
 albumid;
 -- order above result by count of tracks

SELECT
 albumid,
 COUNT(trackid) as track_count
FROM
 tracks
GROUP BY
 albumid
ORDER BY 2 DESC;



-- Get total length and bytes for each album

SELECT
 albumid,
 sum(milliseconds) as length,
 sum(bytes) as size
FROM
 tracks
GROUP BY
 albumid;

  -- Get count of tracks by media type and genre
SELECT
 mediatypeid,
 genreid,
 count(trackid)
FROM
 tracks
GROUP BY
 mediatypeid,
 genreid;
```

```sql
 -- USE OF HAVING CLAUSE
 -- Get all albums and their total length and bytes with album length more than a minute


SELECT
 albumid,
 sum(milliseconds) as length,
 sum(bytes) as size
FROM
 tracks
GROUP BY
 albumid
HAVING sum(milliseconds) > 60000;

 -- SQL CASE when
 /* make a report of the customer groups with the logic that
 if a customer locates in the USA, this customer belongs to the domestic group,
 otherwise the customer belongs to the foreign group.*/

 SELECT customerid,
       firstname,
       lastname,
   country,
       CASE  WHEN country = 'USA'
               THEN 'Dosmetic'
           ELSE 'Foreign'
       END CustomerGroup
FROM
   customers
ORDER BY
   LastName,
   FirstName;
```

```
--Classify them based on eamil id

SELECT customerid,
       firstname,
       lastname,
       CASE
          WHEN email like '%yahoo%' THEN 'YAHOO'
          WHEN email like '%gmail%' THEN 'GMAIL'
          ELSE 'Others'
       END CustomerGroup
FROM
    customers
ORDER BY
    LastName,
    FirstName;

-- classify the tracks based on its length such as less a minute,
-- the track is short; between 1 and 5 minutes, the track is medium;
-- greater than 5 minutes, the track is long

SELECT
 trackid,
 name,
 CASE
 WHEN milliseconds < 60000 THEN
 'short'
 WHEN milliseconds > 6000 AND milliseconds < 300000 THEN 'medium'
 ELSE
 'long'
 END category
FROM
 tracks;
```