Create Database

```
CREATE DATABASE <Database_name>;
```

```
CREATE DATABASE  npgcbca3;
```

Listing Databases/Tables

```
SHOW DATABASES
SHOW TABLES
```

Selecting database

```
USE <database_name>
```

```
Use npgcbca3;
```

Deleting database

```
DROP <database_name>;
```

```
Drop npgcbca3;
```

Creating table

```
CREATE TABLE <Table_name>
      (<Column_name_1> <data_type> <constraints>,
       <Column_name_2> <data_type> <constraints>,
       ..);
```

```
CREATE TABLE Employee
      (emp_id INT PRIMARY KEY,
       dept_id INT,
       emp_name VARCHAR(30),
       emp_address VARCHAR(30),
       emp_sal INT);
```

Constraints :

- ***NOT NULL*** - Ensures that a column cannot have a NULL value
- ***UNIQUE*** - Ensures that all values in a column are different

- ***PRIMARY KEY*** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

- ***FOREIGN KEY*** - Uniquely identifies a row/record in another table. The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables. E.g.

```
CREATE TABLE Orders (
     OrderID int NOT NULL PRIMARY KEY,
     PersonID int FOREIGN KEY REFERENCES Persons(PersonID));
```

- ***CHECK*** - Ensures that all values in a column satisfies a specific condition

```
CREATE TABLE Persons (
     ID int NOT NULL,
     LastName varchar(255) NOT NULL,
     FirstName varchar(255),
     Age int CHECK (Age>=18));
```

- ***DEFAULT*** - Sets a default value for a column when no value is specified

```
CREATE TABLE Persons (
     ID int NOT NULL,
     LastName varchar(255) NOT NULL,
     FirstName varchar(255),
     Age int,
     City varchar(255) DEFAULT 'Sandnes');
```

- ***INDEX*** - Used to create and retrieve data from the database very quickly

```
CREATE INDEX idx_lastname ON Persons (LastName);

-- TO REMOVE INDEX USE
DROP INDEX table_name.index_name;
```

- ***AUTO_INCREMENT*** – Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

```
CREATE TABLE Persons (
     ID int NOT NULL AUTO_INCREMENT = 100,
     LastName varchar(255) NOT NULL,
     FirstName varchar(255),
     Age int,
     PRIMARY KEY (ID));

-- INITIAL VALUE IS 100. SQL SETS IT TO 1 IF NOT SPECIFIED
```

Create table from another table

Delete Table

```
DROP <table_name>
```

DROP Employee

Truncate Table (Deletes only data but not table)

```
TRUNCATE <table_name>
```

TRUNCATE Employee

Alter Table

1. Inserting new column

```
ALTER TABLE <table_name> Add <column_name> <data_type>
```

ALTER TABLE Employee Add DOB datetime

2. Deleting a column

```
ALTER TABLE <table_name> DROP COLUMN <column_name>
```

ALTER TABLE Employee DROP COLUMN DOB

3. Changing datatype of column

```
ALTER TABLE <table_name> ALTER COLUMN <column_name> <data_type>
```

ALTER TABLE Employee ALTER COLUMN emp_salary BIGINT

4. Changing column name

```
EXEC sp_RENAME '<table_name.old_column_name>', '<new_name>', 'COLUMN'
```

5. Change table name

```
EXEC sp_RENAME '<table_name.old_table_name>', '<new_name>', 'COLUMN'
```

Create View

In SQL, a view is a virtual table based on the result-set of an SQL statement.
A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

```
CREATE VIEW [<view_name>] AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Examples :

The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

```
CREATE VIEW [Current Product List] AS
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued = No;
```

Then, we can query the view as follows:

```
SELECT * FROM [Current Product List];
```

Another view in the Northwind sample database selects every product in the "Products" table with a unit price higher than the average unit price:

```
CREATE VIEW [Products Above Average Price] AS
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice > (SELECT AVG(UnitPrice) FROM Products);
```

We can query the view above as follows:

```
SELECT * FROM [Products Above Average Price];
```

Another view in the Northwind database calculates the total sale for each category in 1997. Note that this view selects its data from another view called "Product Sales for 1997":

```
CREATE VIEW [Category Sales For 1997] AS
SELECT DISTINCT Category_Name, Sum(Product_Sales) AS Category_Sales
FROM [Product Sales for 1997]
GROUP BY CategoryName;
```

We can query the view above as follows:

```
SELECT * FROM [Category Sales For 1997];
```

We can also add a condition to the query. Let's see the total sale only for the category "Beverages":

```
SELECT * FROM [Category Sales For 1997]
WHERE CategoryName = 'Beverages';
```

Inserting data into table

```
INSERT INTO <table_name> VALUES(<column1_value>, <column2_value>, ..);

INSERT INTO <table_name> (<Column_name1>, <column_name_2>, …)
Values (<column1_value>, <column2_value>, …);

-- 8/9/2018
INSERT INTO Department VALUES(100, 'SALES');
INSERT INTO Department VALUES(200, 'MARKETING');
INSERT INTO Department VALUES(300, 'ADMINISTRATION');
INSERT INTO Department VALUES(400, 'PRODUCTION');

INSERT INTO Employee VALUES(1000, 100, 'Satish Kumar', 'Lucknow', 10000);
INSERT INTO Employee VALUES(1001, 100, 'Abhay Bhist', 'Delhi', 12000);
INSERT INTO Employee VALUES(1002, 100, 'Kapil Rajput', 'Mumbai', 15000);
INSERT INTO Employee VALUES(1003, 100, 'Ravi Mishra', 'Kanpur', 11400);
INSERT INTO Employee VALUES(1004, 100, 'Neeraj Yadav', 'Banaras', 16500);

INSERT INTO Employee VALUES(1005, 200, 'Chirag Singh', 'Lucknow', 17000);
INSERT INTO Employee VALUES(1006, 200, 'Raj Singh', 'Delhi', 12000);
INSERT INTO Employee VALUES(1007, 200, 'Ramesh Yadav', 'Allahabad', 21000);
INSERT INTO Employee VALUES(1008, 200, 'Rakesh Maurya', 'Mumbai', 22000);
INSERT INTO Employee VALUES(1009, 200, 'Amar Shukla', 'Lucknow', 20000);

INSERT INTO Employee VALUES(1010, 300, 'Avinash Thakur', 'Mumbai', 21000);
INSERT INTO Employee VALUES(1011, 300, 'Vishal Yadav', 'Lucknow', 25000);
INSERT INTO Employee VALUES(1012, 300, 'Surendra Rao', 'Delhi', 26000);
INSERT INTO Employee VALUES(1013, 300, 'Satyendra Pawar', 'Banaras', 26500);
INSERT INTO Employee VALUES(1014, 300, 'Mohit Mishra', 'Kanpur', 26800);

INSERT INTO Employee VALUES(1015, 400, 'Vivek Singh', 'Mumbai', 19000);
INSERT INTO Employee VALUES(1016, 400, 'Vipin Rajput', 'Kanpur', 18000);
INSERT INTO Employee VALUES(1017, 400, 'Vikas Dixit', 'Delhi', 21000);
INSERT INTO Employee VALUES(1018, 400, 'Ajit Kumar', 'Lucknow', 25000);
INSERT INTO Employee VALUES(1019, 400, 'Aryan Srivastava', 'Lucknow', 29000);
```

Printing String/Expression

```
SELECT “String”;
SELECT <Expression>
```

Exercise

Retrieve data from tables

Write a SQL statement to display all the information of all salesmen.

```
SELECT * FROM Salesman;
```

Write a SQL statement to display a string "This is SQL Exercise, Practice and Solution".

```
SELECT "This is SQL Exercise, Practice and Solution";
```

Write a query to display three numbers in three columns.

```
SELECT 1,2,3;
```

Write a query to display the sum of two numbers 10 and 15 from RDMS server.

```
SELECT 10+15;
```

Write a query to display the result of an arithmetic expression.

```
SELECT (10+1)*5/3+2;
```

Write a SQL statement to display specific columns like name and commission for all the salesmen.

```
SELECT Name, Commission FROM Salesman;
```

Write a query to display the columns in a specific order like order date, salesman id, order number and purchase amount from for all the orders.

```
SELECT * From Orders ORDER BY Ord_Date DESC;
SELECT * From Orders ORDER BY Salesman_ID;
```

Write a query which will retrieve the value of salesman id of all salesmen, getting orders from the customers in orders table without any repeats.

```
SELECT DISTINCT (Salesman_id)  FROM Orders;
```

Write a SQL statement to display names and city of salesman, who belongs to the city of Paris.

```
SELECT Name, City FROM Salesman WHERE city = 'Paris';
```

Write a SQL statement to display all the information for those customers with a grade of 200.

```
SELECT * FROM CUSTOMER WHERE Grade = 200;
```

Write a SQL query to display the order number followed by order date and the purchase amount for each order which will be delivered by the salesman who is holding the ID 5001.

```
SELECT Ord_No, Ord_Date, Purch_Amt FROM Orders WHERE Salesman_ID = 5001;
```

Write a SQL query to display the Nobel prizes for 1970.

```
SELECT * FROM Nobel_Win Where Year = 1970;
```

Write a SQL query to know the winner of the 1971 prize for Literature.

```
SELECT * FROM Nobel_Win Where Year = 1971 AND Subject = 'Literature';
```

Write a SQL query to display the year and subject that won 'Dennis Gabor' his prize.

```
SELECT Year, Subject FROM Nobel_Win Where Winner = 'Dennis Gabor';
```

Write a SQL query to give the name of the 'Physics' winners since the year 1950.

```
SELECT Name FROM Nobel_Win WHERE Year >= 1950 AND Subject = ‘Physics’;
```

Write a SQL query to Show all the details (year, subject, winner, country) of the Chemistry prize winners between the year 1965 to 1975 inclusive.

```
SELECT * FROM Nobel_Win
WHERE Year BETWEEN 1965 AND 1975 AND Subject = 'Chemistry';
```

Write a SQL query to show all details of the Prime Ministerial winners after 1972 of Menachem Begin and Yitzhak Rabin.

```
SELECT * FROM Nobel_Win
WHERE Year > 1972 AND Winner IN('Menachem Begin', 'Yitzhak Rabin');
```

Write a SQL query to show all the details of the winners with first name Louis.

```
SELECT * FROM Nobel_Win WHERE Winner LIKE 'Louis%';
```

Write a SQL query to show all the winners in Physics for 1970 together with the winner of Economics for 1971.

```
SELECT * FROM Nobel_Win
WHERE (Year = 1970 AND Subject = 'Physics')
      OR (Year = 1971 AND Subject = 'Economics');
```

OR

```
SELECT * FROM Nobel_Win
WHERE (Year = 1970 AND Subject = 'Physics')
UNION
SELECT * FROM Nobel_Win
WHERE (Year = 1971 AND Subject = 'Economics')
```

Write a SQL query to show all the winners of nobel prize in the year 1970 except the subject Physiology and Economics.

```
SELECT * FROM Nobel_Win WHERE Subject NOT IN('Physiology', 'Economics');
```

Write a SQL query to show the winners of a 'Physiology' prize in an early year before 1971 together with winners of a 'Peace' prize in a later year on and after the 1974.

```
SELECT * FROM Nobel_Win
WHERE(Year < 1971 AND Subject = 'Physiology')
UNION
SELECT * FROM Nobel_Win
WHERE (Year > 1974 AND Subject = 'Peace')
```

Write a SQL query to find all details of the prize won by Johannes Georg Bednorz.

```
SELECT * FROM Nobel_Win Where Winner = 'Johannes Georg Bednorz';
```

Write a SQL query to find all the details of the nobel winners for the subject not started with the letter 'P' and arranged the list as the most recent comes first, then by name in order.

```
SELECT * FROM Nobel_Win
WHERE Subject Not Like 'P%' ORDER BY Year DESC, Winner;
```

Write a SQL query to find all the details of 1970 winners by the ordered to subject and winner name; but the list contain the subject Economics and Chemistry at last.

```
SELECT * FROM Nobel_Win
WHERE YEAR = 1970 ORDER BY
CASE WHEN Subject IN ('Chemistry', 'Economics') THEN 1 ELSE 0
END, Subject, Winner;
```

Write a SQL query to find all the products with a price between Rs.200 and Rs.600

```
SELECT * FROM Item_Mast WHERE Pro_Price BETWEEN 200 AND 600;
```

Write a SQL query to calculate the average price of all products of the manufacturer which code is 16.

```
SELECT AVG(Pro_Price) FROM Item_Mast Where Pro_Com = 16;
```

Write a SQL query to find the item name and price in Rs.

```
SELECT Pro_Name AS "Item Name", Pro_Price AS "Price In Rs." From Item_Mast;
```

Write a SQL query to display the name and price of all the items with a price is equal or more than Rs.250, and the list contain the larger price first and then by name in ascending order.

```
SELECT Pro_Name, Pro_Price FROM Item_Mast
WHERE Pro_Price >= 250 ORDER BY Pro_Price DESC, Pro_Name;
```

Write a SQL query to display the average price of the items for each company, showing only the company code.

```
SELECT Pro_Com, AVG(Pro_Price) FROM Item_Mast GROUP BY Pro_Com;
```

Write a SQL query to find the name and price of the cheapest item(s).

```
SELECT Pro_Price, Pro_Name FROM Item_Mast
Where Pro_Price = (SELECT MIN(Pro_Price) FROM Item_Mast);
```

Write a query in SQL to find the last name of all employees, without duplicates.

```
SELECT DISTINCT Emp_Lname FROM EMP_DETAILS
```

Write a query in SQL to find the data of employees whose last name is 'Snares'

```
SELECT * FROM EMP_DETAILS WHERE Emp_Lname = 'Snares'
```

Write a query in SQL to display all the data of employees that work in the department 57.

```
SELECT * FROM EMP_DETAILS WHERE Emp_Dept = 57
```

Write a query in SQL to display name, salary and department ID of employees having maximum salary in their respective department.

```
SELECT Emp_Name, Emp_Sal, Dept_ID FROM Employee
WHERE Emp_Sal IN (Select MAX(Emp_Sal) FROM Employee GROUP BY Dept_ID);
```