MYSQL Part-II

Lesson Plan -2 **02/04/2020**

Link for Video Session:

meet.google.com/kdh-sjit-ryq

Step 1: Learning Objectives:

To enable students to :

- Implement the sorting of data using order by clause
- Update the data in table(records)
- Delete data from table
- Use aggregate functions
- Implement joins in SQL

Step 2: Introduction:

- **SORTING RESULTS**

The **ORDER BY clause** allow sorting of query result. The sorting can be done either in ascending or descending order, the default is ascending and to arrange in descending order we use desc after the column name.

SYNTAX:

SELECT <column name> , <column name>…. FROM <tablename>
WHERE <condition> **ORDER BY** <column name>;

e.g. Write a query to display the details of employees in EMPLOYEE table in alphabetical order.

Solution: SELECT * FROM EMPLOYEE ORDER BY ENAME ;

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1002	Akash	M	A1	35000
1004	Neela	F	B2	38965
1009	Neema	F	A2	52000
1001	Ravi	M	E4	50000
1006	Ruby	F	A1	45000
1005	Sunny	M	A2	30000

e.g. To display list of employees in descending alphabetical order whose salary is greater than 40000.

Solution:
SELECT ENAME **FROM** EMPLOYEE **WHERE** GROSS > 40000 **ORDER BY** ENAME desc ;

Output will be:

ENAME
Ravi
Ruby
Neema

- **MODIFYING DATA IN TABLES**

The UPDATE command of SQL is used to modify data in a table.
The UPDATE command specifies the record to be changed / updated using the WHERE clause, and the new data using the SET keyword.

Syntax:
UPDATE <tablename> SET <columnname>=value, <columnname>=value WHERE <condition>;

e.g. Write a query to change the salary of employee of those in EMPLOYEE table having employee code 1009 to 55000.

Solution:

UPDATE EMPLOYEE **SET** GROSS = 55000 **WHERE** ECODE = 1009 ;

UPDATING MORE THAN ONE COLUMNS

Multiple columns of a record can also be updated using UPDATE command.

e.g. To update the salary to 58000 and grade to B2 for those employee whose employee code is 1001.

Solution:

UPDATE EMPLOYEE **SET** GROSS = 58000, GRADE='B2' **WHERE** ECODE = 1009 ;

Few More Examples:

e.g.1. Increase the salary of each employee by 1000 in the EMPLOYEE table.
UPDATE EMPLOYEE
SET GROSS = GROSS +100 ;

e.g.2. Double the salary of employees having grade as 'A1' or 'A2' .
UPDATE EMPLOYEE
SET GROSS = GROSS * 2 ;
WHERE GRADE='A1' **OR** GRADE='A2' ;

e.g.3. Change the grade to 'A2' for those employees whose employee code is 1004 and name is Neela.
UPDATE EMPLOYEE **SET** GRADE='A2' **WHERE** ECODE=1004 **AND** GRADE='NEELA';

- **DELETING DATA FROM TABLES**

The DELETE command removes rows from a table.

Syntax:

DELETE FROM <tablename> **WHERE** <condition> ;

For example, Query to remove the details of those employee from EMPLOYEE table whose grade is A1.

DELETE FROM EMPLOYEE WHERE GRADE ='A1';

TO DELETE ALL THE CONTENTS FROM A TABLE, we use

DELETE FROM EMPLOYEE ;

That is, if we do not specify any condition with WHERE clause, then all the rows of the table will be deleted.

- DROPPING TABLES

The DROP TABLE command lets you drop a table from the database i.e. removal of table from storage.

Syntax:

DROP TABLE <tablename> ;

e.g. Write a query to delete a table employee

Solution: **DROP TABLE** employee ;

Once this command is given, the table name is no longer recognized and no more commands can be given on that table.

After this command is executed, all the data in the table along with table structure will be deleted.

Difference between Delete and Drop Commands:

DELETE COMMAND	**DROP TABLE COMMAND**
It is a DML command.	It is a DDL Command.
This command is used to delete only data rows /records from a table.	This command is used to delete all the data of the table along with the table structure. The table is no longer recognized after execution of this command.
Syntax: **DELETE FROM** <tablename> **WHERE** <condition> ;	Syntax: **DROP TABLE** <tablename> ;

- **ALTER TABLE COMMAND**

The ALTER TABLE command is used to change definitions of existing tables.(adding columns, deleting columns etc.).

The ALTER TABLE command is used for:

1. Adding columns to a table
2. Modifying column-definitions of a table.
3. Deleting columns of a table.
4. Adding constraints to table.
5. Enabling/Disabling constraints.

ADDING COLUMNS TO TABLE

To add a column to a table, ALTER TABLE command can be used as per following syntax:

ALTER TABLE <tablename> **ADD** <Column name> <datatype> <constraint> ;

e.g. to add a new column ADDRESS to the EMPLOYEE table, we can write command as :

ALTER TABLE EMPLOYEE **ADD** ADDRESS VARCHAR(50);

A new column by the name ADDRESS will be added to the table, where each row will contain NULL value for the new column.

ECODE	ENAME	GENDER	GRADE	GROSS	ADDRESS
1001	Ravi	M	E4	50000	NULL
1002	Akash	M	A1	35000	NULL
1004	Neela	F	B2	38965	NULL
1005	Sunny	M	A2	30000	NULL
1006	Ruby	F	A1	45000	NULL
1009	Neema	F	A2	52000	NULL

*However **if you specify NOT NULL constraint while adding a new column**, MySQL adds the new column with the default value of that datatype e.g. for INT type it will add 0 , for CHAR types, it will add a space, and so on.*

MODIFYING COLUMNS

Column name and data type of column can be changed as per following syntax :

ALTER TABLE <table name>**CHANGE** <old column name> <new column name> <new datatype>;

If **Only datatype of column need to be changed**, then

ALTER TABLE <table name> **MODIFY** <column name> <new datatype>;

e.g.1. In table EMPLOYEE, change the column GROSS to SALARY.

ALTER TABLE EMPLOYEE **CHANGE** GROSS SALARY INTEGER;

e.g.2. In table EMPLOYEE , change the column ENAME to EM_NAME and data type from VARCHAR(20) to VARCHAR(30).

ALTER TABLE EMPLOYEE **CHANGE** ENAME EM_NAME VARCHAR(30);

e.g.3. In table EMPLOYEE , change the datatype of GRADE column from CHAR(2) to VARCHAR(2).

ALTER TABLE EMPLOYEE **MODIFY** GRADE VARCHAR(2);

DELETING COLUMNS

To delete a column from a table, the ALTER TABLE command takes the following form :

ALTER TABLE <table name> **DROP** <column name>;

e.g. to delete column GRADE from table EMPLOYEE:

ALTER TABLE EMPLOYEE **DROP** GRADE ;

ADDING/REMOVING CONSTRAINTS TO A TABLE

ALTER TABLE statement can be used to add constraints to your existing table by using it in following manner:

a) TO ADD PRIMARY KEY CONSTRAINT

ALTER TABLE <table name> **ADD PRIMARY KEY** (Column name);

e.g. to add PRIMARY KEY constraint on column ECODE of table EMPLOYEE

ALTER TABLE EMPLOYEE **ADD PRIMARY KEY** (ECODE) ;

b) TO ADD FOREIGN KEY CONSTRAINT

ALTER TABLE <table name> **ADD FOREIGN KEY** (Column name) REFERENCES Parent Table (Primary key of Parent Table);

REMOVING CONSTRAINTS

To remove primary key constraint from a table, we use ALTER TABLE command as :
ALTER TABLE <table name> **DROP** PRIMARY KEY ;

To remove foreign key constraint from a table, we use ALTER TABLE command as :
ALTER TABLE <table name> **DROP** FOREIGN KEY ;

INTEGRITY CONSTRAINTS/CONSTRAINTS

- A constraint is a condition or check applicable on a field(column) or set of fields(columns).
- Common types of constraints include :

S.No.	Constraints	Description
1	NOT NULL	Ensures that a column cannot have NULL value
2	DEFAULT	Provides a default value for a column when none is specified
3	UNIQUE	Ensures that all values in a column are different
4	CHECK	Makes sure that all values in a column satisfy certain criteria
5	PRIMARY KEY	Used to uniquely identify a row in the table
6	FOREIGN KEY	Used to ensure referential integrity of the data

NOT NULL CONSTRAINT

By default, a column can hold NULL. It you not want to allow NULL value in a column, then NOT NULL constraint must be applied on that column.

DEFAULT CONSTARINT

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.

UNIQUE CONSTRAINT

- The UNIQUE constraint ensures that all values in a column are distinct. In other words, no two rows can hold the same value for a column with UNIQUE constraint.

CHECK CONSTRAINT

- The CHECK constraint ensures that all values in a column satisfy certain conditions. Once defined, the table will only insert a new row or update an existing row if the new value satisfies the CHECK constraint.

PRIMARY KEY CONSTRAINT

- A primary key is used to identify each row in a table. A primary key can consist of one or more fields(column) on a table. When multiple fields are used as a primary key, they are called a **composite key.**

FOREIGN KEY CONSTRAINT

- Foreign key is a non key column of a table (**child table**) that draws its values from **primary key** of another table(**parent table).**
- The table in which a foreign key is defined is called a **referencing table or child table.** A table to which a foreign key points is called **referenced table or parent table.**

REFERENCING ACTIONS

Referencing action with ON DELETE clause determines what to do in case of a DELETE occurs in the parent table. Referencing action with ON UPDATE clause determines what to do in case of an UPDATE occurs in the parent table.

Q: Create two tables
Customer(customer_id, name) and Customer_sales(transaction_id, amount , **customer_id**)

Underlined columns indicate primary keys and bold column names indicate foreign key. Make sure that no action should take place in case of a DELETE or UPDATE in the parent table.

Sol :

CREATE TABLE Customer (customer_id int Not Null Primary Key , name varchar(30)) ;

CREATE TABLE customer_sales (transaction_id Not Null Primary Key , amount int , customer_id int , FOREIGN KEY(customer_id) REFERENCES Customer (customer_id) ON DELETE NO ACTION ON UPDATE NO ACTION);

- **AGGREGATE / GROUP FUNCTIONS**

Aggregate / Group functions work upon groups of rows , rather than on single row, and return one single output. Different aggregate functions are : COUNT( ) , AVG( ) , MIN( ) , MAX( ) , SUM ( )

Table : EMPL

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20
8912	SUR	NULL	3000	10

1. **AVG( )**
This function computes the average of given data. e.g. SELECT AVG(SAL) FROM EMPL ;

Output

AVG(SAL)
6051.6

2. **COUNT( )**
This function counts the number of rows in a given column, where COLUMN is not null. If you specify the asterisk (*), this function returns all rows, including duplicates and nulls.

e.g. SELECT COUNT(*) FROM EMPL ;
Output

COUNT(*)
5

e.g.2 SELECT COUNT(JOB) FROM EMPL ;

Output

COUNT(JOB)
4

3. **MAX( )**

This function returns the maximum value from a given column or expression.

e.g. SELECT MAX(SAL) FROM EMPL ;

Output

MAX(SAL)
9870

4. **MIN( )**

This function returns the minimum value from a given column or expression.

e.g. SELECT MIN(SAL) FROM EMPL;

Output

MIN(SAL)
2985

5. **SUM( )**

This function returns the sum of values in given column or expression.

e.g. SELECT SUM(SAL) FROM EMPL ;

Output

SUM(SAL)
30258

GROUPING RESULT – GROUP BY

The GROUP BY clause combines all those records(row) that have identical values in a particular field(column) or a group of fields(columns).

GROUPING can be done by a column name, or with aggregate functions in which case the aggregate produces a value for each group.

Table : EMPL

EMPNO	ENAME	JOB	SAL	DEPT NO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20

e.g. Calculate the number of employees in each grade.

SELECT JOB, COUNT(*) FROM EMPL GROUP BY JOB;

Output

JOB	COUNT(*)
CLERK	1
SALESMAN	2
MANAGER	1

e.g.2. Calculate the sum of salary for each department.

SELECT DEPTNO ,SUM(SAL) FROM EMPL GROUP BY DEPTNO ;

Output

DEPTNO	SUM(SAL)
10	2985
20	15513
30	8760

*** One thing that you should keep in mind is that while grouping , you should include only those values in the SELECT list that either have the same value for a group or contain a group(aggregate) function. Like in e.g. 2 given above, DEPTNO column has one(same) value for a group and the other expression SUM(SAL) contains a group function.*

PLACING CONDITION ON GROUPS – HAVING CLAUSE

- The **HAVING clause places conditions on groups** in contrast to WHERE clause that places condition on individual rows.
  While **WHERE conditions cannot include aggregate functions, HAVING conditions can do so.**
  e.g. To display the jobs where the number of
  employees is less than 2, SELECT JOB, COUNT(*)
  FROM EMPL
  GROUP BY JOB
  HAVINGCOUNT(*) < 2 ;
  Output

JOB	COUNT(*)
CLERK	1
MANAGER	1

- **DATABASE TRANSACTIONS**

TRANSACTION

A Transaction is a logical unit of work that must succeed or fail in its entirety. This statement means that a transaction may involve many sub steps, which should either all be carried out successfully or all be ignored if some failure occurs. A Transaction is an atomic operation which may not be divided into smaller operations.

Example of a Transaction

a) Begin transaction
b) Get balance from account X Calculate new balance as X – 1000 Store new balance into database file Get balance from account Y Calculate new balance as Y + 1000 Store new balance into database file
c) End transaction

TRANSACTION PROPERTIES (ACID PROPERTIES)

1. **ATOMICITY**(All or None Concept) – This property ensures that either all operations of the transaction are carried out or none are.
2. **CONSISTENCY** – This property implies that if the database was in a consistent state before the start of transaction execution, then upon termination of transaction, the database will also be in a consistent state.
3. **ISOLATION** – This property implies that each transaction is unaware of other transactions executing concurrently in the system.
4. **DURABILITY** – This property of a transaction ensures that after the successful completion of a transaction, the changes made by it to the database persist, even if there are system failures.

TRANSACTION CONTROL COMMANDS(TCL)

- The TCL of MySQL consists of following commands: BEGIN or START

1. COMMIT – Ends the current transaction by saving database changes and starts a new transaction.
2. ROLLBACK – Ends the current transaction by discarding database changes and starts a new transaction.
3. SAVEPOINT – Define breakpoints for the transaction to allow partial rollbacks.
4. SET AUTOCOMMIT – Enables or disables the default auto commit mode.

JOINS

- A join is a query that combines rows from two or more tables. In a join- query, more than one table are listed in FROM clause.

Table : empl

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20

Table : dept

DEPT NO	DNAME	LOC
10	ACCOUNTING	NEW DELHI
20	RESEARCH	CHENNAI
30	SALES	KOLKATA
40	OPERATIONS	MUMBAI

CARTESIAN PRODUCT /CROSS JOIN

- Consider the following query :
  SELECT * FROM EMPL, DEPT ;

empno	ename	job	sal	deptno	deptno	dname	loc
8369	SMITH	CLERK	2985	10	10	ACCOUNTING	NEW DELHI
8499	ANYA	SALESMAN	9870	20	10	ACCOUNTING	NEW DELHI
8566	AMIR	SALESMAN	8760	30	10	ACCOUNTING	NEW DELHI
8698	BINA	MANAGER	5643	20	10	ACCOUNTING	NEW DELHI
8369	SMITH	CLERK	2985	10	20	RESEARCH	CHENNAI
8499	ANYA	SALESMAN	9870	20	20	RESEARCH	CHENNAI
8566	AMIR	SALESMAN	8760	30	20	RESEARCH	CHENNAI
8698	BINA	MANAGER	5643	20	20	RESEARCH	CHENNAI
8369	SMITH	CLERK	2985	10	30	SALES	KOLKATA
8499	ANYA	SALESMAN	9870	20	30	SALES	KOLKATA
8566	AMIR	SALESMAN	8760	30	30	SALES	KOLKATA
8698	BINA	MANAGER	5643	20	30	SALES	KOLKATA
8369	SMITH	CLERK	2985	10	40	OPERATIONS	MUMBAI
8499	ANYA	SALESMAN	9870	20	40	OPERATIONS	MUMBAI
8566	AMIR	SALESMAN	8760	30	40	OPERATIONS	MUMBAI
8698	BINA	MANAGER	5643	20	40	OPERATIONS	MUMBAI

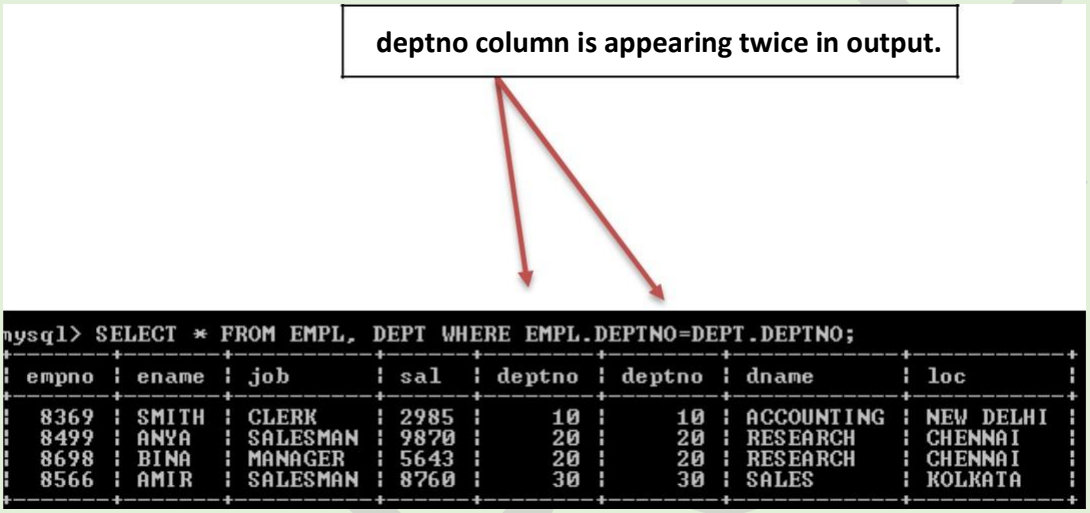
This query will give you the Cartesian product i.e. all possible concatenations are formed of all rows of both the tables EMPL and DEPT. Such an operation is also known as **Cross Join.** It returns n1 x n2 rows where n1 is number of rows in first table and n2 is number of rows in second table.

EQUI-JOIN

- The join in which columns are compared for equality, is called Equi - Join.

In equi-join, all the columns from joining table appear in the output even if they are identical.

e.g. SELECT * FROM empl, dept WHERE empl.deptno = dept.deptno ;



Q: With reference to empl and dept table, display the location of employee SMITH.

SELECT ENAME, LOC FROM EMPL, DEPT WHERE EMPL.DEPTNO = DEPT.DEPTNO AND ENAME='SMITH';

Q: Display details like department number, department name, employee number, employee name, job and salary. And order the rows by employee number.

SELECT EMPL.deptno, dname,empno,ename,job,sal FROM EMPL,DEPT
WHERE EMPL.DEPTNO=DEPT.DEPTNO ORDER BY EMPL.DEPTNO;

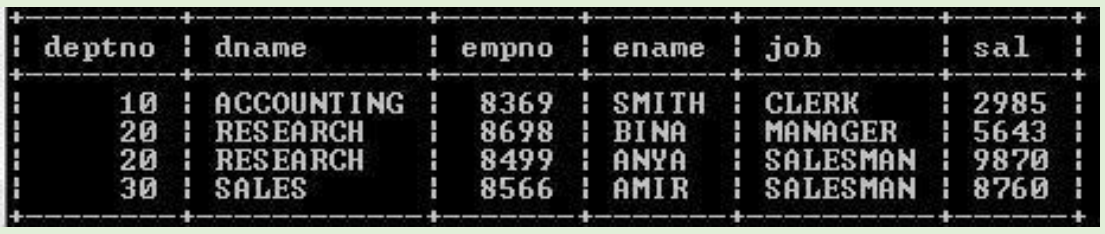


TABLE ALIAS

-A table alias is a temporary label given along with table name in FROM clause.

e.g.
SELECT E.DEPTNO, DNAME,EMPNO,ENAME,JOB,SAL FROM EMPL E, DEPT D
WHERE E.DEPTNO = DEPT.DEPTNO ORDER BY E.DEPTNO;

In above command table alias for EMPL table is E and for DEPT table , alias is D.

Q: Display details like department number, department name, employee number, employee name, job and salary. And order the rows by employee number with department number. These details should be only for employees earning atleast Rs. 6000 and of SALES department.

SELECT E.DEPTNO, DNAME,EMPNO, ENAME, JOB, SAL FROM EMPL E, DEPT D
WHERE E.DEPTNO = D.DEPTNO AND DNAME='SALES'
AND SAL>=6000 ORDER BY E.DEPTNO;

DEPTNO	DNAME	EMPNO	ENAME	JOB	SAL
30	SALES	8566	AMIR	SALESMAN	8760

NATURAL JOIN

By default, the results of an equijoin contain two identical columns. One of the two identical columns can be eliminated by restating the query. This result is called a Natural join.

e.g. SELECT empl.*, dname, loc
FROM empl,dept
WHERE
empl.deptno = dept.deptno ;

empno	ename	job	sal	deptno	dname	loc
8369	SMITH	CLERK	2985	10	ACCOUNTING	NEW DELHI
8499	ANYA	SALESMAN	9870	20	RESEARCH	CHENNAI
8698	BINA	MANAGER	5643	20	RESEARCH	CHENNAI
8566	AMIR	SALESMAN	8760	30	SALES	KOLKATA

empl.* means select all columns from empl table. This thing can be used with any table.

The join in which only one of the identical columns(coming from joined tables) exists, is called Natural Join.

Step 3: Assignment (to be done in the registers)

Q1.

Observe the following STUDENTS and EVENTS tables carefully and write the name of the RDBMS operation which will be used to produce the output as shown in LIST. Also, find the Degree and Cardinality of the LIST. ***(CBSE- Delhi 2016)***

STUDENTS

No	Name
1	Tara mani
2	Jaya Sarkar
3	Tarini Trikha

EVENTS

EVENTCODE	EVENTNAME
1001	Programming
1002	IT Quiz

LIST

NO	NAME	EVENTCODE	EVENTNAME
1	Tara mani	1001	Programming
1	Tara mani	1002	IT Quiz
2	Jaya Sarkar	1001	Programming
2	Jaya Sarkar	1002	IT Quiz
3	Tarini Trikha	1001	Programming
3	Tarini Trikha	1002	IT Quiz

Q2.

Given the following relation: STUDENT

No.	Name	Age	Department	Dateofadm	Fee	Sex
1	Pankaj	24	Computer	10/01/97	120	M
2	Shalini	21	History	24/03/98	200	F
3	Sanjay	22	Hindi	12/12/96	300	M
4	Sudha	25	History	01/07/99	400	F
5	Rakesh	22	Hindi	05/09/97	250	M
6	Shakeel	30	History	27/06/98	300	M
7	Surya	34	Computer	25/02/97	210	M
8	Shikha	23	Hindi	31/07/97	200	F

Write SQL commands for the following queries

a) To show all information about the students of History department.

b) To list the names of female students who are in Hindi department.

c) To list the names of all students with their date of admission in ascending order.

d) To display student's name, fee, age for male students only.

e) To count the number of students with Age>23.

Q3.

Answer the (a) and (b) on the basis of the following tables **STORE** and **ITEM**: ***(CBSE 2014)***

STORE

SNo	SName	AREA
S01	ABC Computronics	GK II
S02	All Infotech Media	CP
S03	Tech Shoppe	Nehru Place
S05	Hitech Tech Store	SP

ITEM

INo	IName	Price	SNo
T01	Mother Board	12000	S01
T02	Hard Disk	5000	S01
T03	Keyboard	500	S02
T04	Mouse	300	S01
T05	Mother Board	13000	S02
T06	Key Board	400	S03
T07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T10	Hard disk	4500	S03

(a) Write the SQL queries (1 to 4):

1) To display IName and Price of all the items in the ascending order of their Price.
2) To display the SNo and SName o all stores located in CP.
3) To display the minimum and maximum price of each IName from the table Item.
4) To display the IName, price of all items and their respective SName where they are available.

(b) Write the output of the following SQL commands (1 to 4):

1) SELECT DISTINCT INAME FROM ITEM WHERE PRICE >= 5000;
2) SELECT AREA, COUNT(*) FROM STORE GROUP BY AREA;
3) SELECT COUNT(DISTINCT AREA) FROM STORE;
4) SELECT INAME, PRICE*0.05 DISCOUNT FROM ITEM WHERE SNO IN ('S02', 'S03');

Q4.

Write SQL queries for (i) to (iv) and find outputs for SQL queries (v) to (viii), which are based on the tables. ***(CBSE- Outside Delhi 2016)***

Table: VEHICLE

VCODE	VEHICLETYPE	PERKM
V01	VOLVO BUS	150
V02	AC DELUXE BUS	125
V03	ORDINARY BUS	80
V05	SUV	30
V04	CAR	18

Note:

- PERKM is Freight Charges per kilometer

Table: TRAVEL

CNO	CNAME	TRAVELDATE	KM	VCODE	NOP
101	K.Niwal	2015-12-13	200	V01	32
103	Fredrick Sym	2016-03-21	120	V03	45
105	Hitesh Jain	2016-04-23	450	V02	42
102	Ravi anish	2016-01-13	80	V02	40
107	John Malina	2015-02-10	65	V04	2
104	Sahanubhuti	2016-01-28	90	V05	4
106	Ramesh jaya	2016-04-06	100	V01	25

Note :

- KM is Kilometer travelled
- NOP is number of travellers travelled in vehicle
- TDATE is Travel Date

(i) To display CNO, CNAME, TRAVELDATE from the table TRAVEL in descending order of CNO.

(ii) To display the CNAME of all the customers from the table TRAVEL who are travelling by vehicle with code V01 or V02.

(iii) To display the CNO and CNAME of those customers from the table TRAVEL who travelled between '2015-12-31' and '2015-05-01'.

(iv) To display all the details from table TRAVEL for the customers, who have travelled distance more than 120 KM in ascending order of NOP.

(v) SELECT COUNT (*), VCODE FROM TRAVEL GROUP BY V CODE HAVING COUNT(*)>1;

(vi) SELECT DISTINCT VCODE FROM TRAVEL;

(vii) SELECT A.VCODE,CNAME,VEHICLETYPE FROM TRAVEL A,VEHICLE B
WHERE A.VCODE=B.VCODE AND KM<90;

(viii) SELECT CNAME,KM*PERKM FROM TRAVEL A, VEHICLE B WHERE
A.VCODE=B.VCODE AND A.VCODE='V05';