EXPERIMENT-1

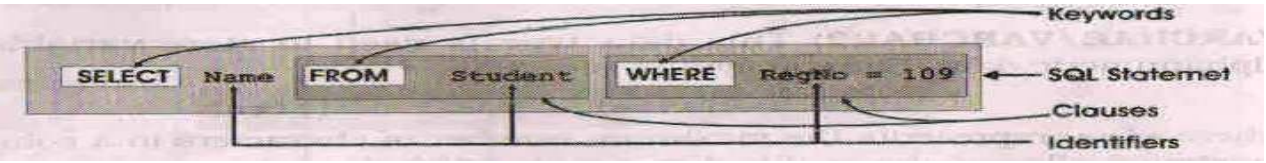
AIM: Creation, altering and dropping of tables and inserting rows into a table (use constraints while creating tables) examples using SELECT command.

DESCRIPTION:

SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. All relational database management systems like MySQL, MS Access, and Oracle, Sybase, Informix, and SQL Server use SQL as standard database language.

➢ Structure of SQL command:

- Any SQL command is a combination of keywords, identifiers and clauses.
- Every SQL command begins with a keyword (CREATE, SELECT, DELETE and so on) which asa specific meaning to the language.



- SELECT, FROM and WHERE are keywords.
- The clauses are "FROM student" and "WHERE RegNo=109".
- Here SELECT and FROM are mandatory, but WHERE is optional.
- Name, Student, RegNo, are identifier that refers to objects in the database.
- Name and RegNo are column names, while Student is a table name.
- The equal sign is an operator and 109 is a numeric constant.

➢ **SQL standard supports 3 types of built-in data types. They are**

1) Numeric Types:

- **INT:** A normal sized integer that can be signed or unsigned and width of up to 11 digits.
- **TINYINT:** A very small integer that can be signed or unsigned and width of up to 4 digits.
- **SMALLINT:** A small integer that can be signed or unsigned and width of up to 5 digits.
- **MEDIUMINT:** A medium-sized integer that can be signed or unsigned and width of up to 9 digits.
- **BIGINT:** A large integer that can be signed or unsigned and width of up to 20 digits.
- **FLOAT(M,D):** A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D):** A double precision floating-point number that cannot be unsigned. Decimal precision can go to 53 places for a DOUBLE.
- **NUMBER(n, d):** Where **n** specifies the number of digits and **d** specifies the number of digits to right of thedecimal point.

2) String Type:

- **CHAR(M)**: A fixed-length string between 1 and 255 characters in length. Defining a length is not required, but the default is 1.
- **VARCHAR(M)/VARCHAR2(M):** A variable-length string between 1 and 255 characters in length. You must when creating a VARCHAR field, length is mandatory.
- **LONG:** It is used to store variable length strings of up to 2GB size.

- **BLOB or TEXT:** A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data.

3) Data and Time Types:

- **DATE** - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31.
- **DATETIME** - A date and time combination in YYYY-MM-DD HH:MM:SS format.
- **TIMESTAMP** - This is same as previous DATETIME format, only without the hyphens between numbers (YYYYMMDDHHMMSS).
- **TIME** - Stores the time in HH:MM:SS format.
- **YEAR(M)** - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

➢ Operator in SQL

- An operator is a reserved word or a character used primarily in an SQL statement's WHEREclause to perform operation(s), such as comparisons and arithmetic operations.
- Operators are used to specify conditions in an SQL statement and to serve as conjunctions formultiple conditions in a statement.
  - Arithmetic operators (+, -, *, / %)
  - Comparison operators (>, <, >=, <=, =, !=, <>, !<, !>)
  - Logical operators (AND, OR, NOT, IN, BETWEEN, EXISTS, ALL, ANY, LIKE, UNIQUE)

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

➢ **Data Definition Language:**

Data Definition Language (DDL) statements are used to define the database structure or schema. Some Examples: CREATE, ALTER, DROP

i. CREATE: This statement is used to create a table in a database. Tables are organized into rows and columns; and each table must have a name.

Syntax:

CREATE TABLE *table_name*
(
*column_name1 data_type*1 [attribute constraint],

*column_name2 data_type*2 [attribute constraint],
*column_name3 data_type*3 [attribute constraint],
. . . . . .
*column_nameN data_type*N [attribute constraint],
[Table constraints]
);

Integrity Constraints: Constraints are used to specify rules for the data in a table. Constraints can be specified when the table is created (inside the CREATE TABLE statement) or after the table is created (inside the ALTER TABLE statement).
Integrity Constraints can be at two levels:

- **Column Level** - Constraint on a single column it allows only certain values for this column.
  Syntax: column_name datatype constraint_Name

- **Table level** - Constraint on a table it can limit the values in certain columns based on values in other columns in the row.
  Syntax: CONSTRAINT constraint_name constraint_name(column_name1,column_name2,……)

Types of Integrity constraints:

i. **NOT NULL:** It enforces a column to NOT accept NULL values.
   Syntax: column_name datatype NOT NULL

ii. **DEFAULT:** It is used to insert a default value into a column. The default value will be added to all new records, if no other value is specified.
   Syntax: column_name datatype DEFAULT default_value

iii. **CHECK:** It is used to limit the value range that can be placed in a column.
   Syntax: column_datatype CHECK(condition)
   OR
   CONSTRAINT constraint_name CHECK(column_name1 condition1,column_name2 condition2,…)

iv. **UNIQUE:** It is used to ensure that the information in the column for each record is unique.
   Syntax: column_name column_type UNIQUE
   OR
   CONSTRAINT constraint_name UNIQUE(column1,coulmn2,...)

v. **PRIMARY KEY:** It uniquely identifies each record in a database table. It is the combination of UNIQUE and NOT NULL.
   Syntax: column_name column_type PRIMARY KEY
   OR
   CONSTRAINT constraint_name PRIMARY KEY(columns)

vi. **FOREIGN KEY:** A foreign key is a field in a table that matches another field of another table. A foreign key places constraints on data in the related tables, which enables MySQL to maintain referential integrity. A FOREIGN KEY in one table points to a PRIMARY KEY in another table.
   Syntax:
   column_name column_type FOREIGN KEY(columns)
   REFERENCES parent_table(columns)
   ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}
   ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}
   OR

CONSTRAINT constraint_name
FOREIGN KEY(columns)
REFERENCES parent_table(columns)
ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}
ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}

Examples:

Bus Reservation database:

- CREATE TABLE STUDENT(RegNo NUMBER (6), Name VARCHAR2 (15), Combination CHAR (4),DOB DATE, Fees NUMBER (9, 2), PRIMARY KEY (RegNo));

- CREATE TABLE Bus(BusNo VARCHAR(5) PRIMARY KEY, Source VARCHAR(20) NOT NULL, Destination VARCHAR(20) NOT NULL, Dep_hr numeric (2) check (Dep_hr >= 0 and Dep_hr < 25), Dep_min numeric (2) check (Dep_min >= 0 and Dep_min < 61));

- CREATE TABLE Ticket (TicketNo INT PRIMARY KEY, Jdate DATE NOT NULL, Age NUMBER(2), Gender VARCHAR(1), Source VARCHAR(20) NOT NULL, Destination VARCHAR(20) NOT NULL, Dept_hr numeric (2) check (Dept_hr >= 0 and Dept_hr < 25), Dept_min numeric (2) check (Dept_min >= 0 and Dept_min < 61), Cost INT);

- CREATE TABLE Passenger(PNRNo NUMBER(4) PRIMARY KEY, PPNo NUMBER(2), Name VARCHAR(15), Age INT, Gender VARCHAR(1), Address VARCHAR(30) NOT NULL, TicketNo NUMBER(3) NOT NULL);

- CREATE TABLE Reservation(PNRNo NUMBER(4), Jdate DATE NOT NULL, NoofSeats NUMBER(2), Address VARCHAR(30) NOT NULL, ContactNo NUMBER(10) UNIQUE, Status VARCHAR(3), CONSTRAINT Resvconst FOREIGN KEY(PNRNo) REFERENCES Passenger(PNRNo));

- CREATE TABLE Cancellation(PNRNo NUMBER(4), Jdate DATE NOT NULL, NoofSeats NUMBER(2), Address VARCHAR(30) NOT NULL, ContactNo NUMBER(10) UNIQUE, Status VARCHAR(3), CONSTRAINT Cancconst FOREIGN KEY(PNRNo) REFERENCES Passenger(PNRNo));

University Database:

- create table classroom(building varchar (15), room_number varchar (7), capacity numeric (4,0), primary key (building, room_number));

- create table department(dept_name varchar (20),building varchar (15),budget numeric (12,2) check (budget > 0),primary key (dept_name));

- create table course(course_id varchar (8),title varchar (50),dept_name varchar (20),credits numeric (2,0) check (credits > 0),primary key (course_id),foreign key (dept_name) references department on delete set null);

- create table instructor(ID varchar (5),name varchar (20) not null,dept_name varchar (20),salary numeric (8,2) check (salary > 29000),primary key (ID),foreign key (dept_name) references department on delete set null);

- create table section(course_id varchar (8),sec_id varchar (8),semester varchar (6) check (semester in('Fall', 'Winter', 'Spring', 'Summer')),year numeric (4,0) check (year > 1701 and year < 2100),building varchar (15),room_number varchar (7),time_slot_id varchar (4),primary key (course_id, sec id, semester, year),foreign key (course_id) references course on delete cascade, foreign key (building, room number) references classroom on delete set null);

- create table teaches(ID varchar (5),course_id varchar (8),sec_id varchar (8),semester varchar (6),year numeric (4,0),primary key (ID, course_id, sec_id, semester, year),foreign key (course_id, sec id, semester, year) references section on delete cascade,foreign key (ID) references instructor on delete cascade);

- create table student(ID varchar (5),name varchar (20) not null,dept_name varchar (20),tot_cred numeric (3,0) check (tot_cred >= 0),primary key (ID),foreign key (dept_name) references department(dept_name) on delete set null);

- create table takes(ID varchar (5),course_id varchar (8),sec_id varchar (8),semester varchar (6),year numeric (4,0),grade varchar(2),primary key (ID, course_id, sec_id, semester, year),foreign key (course_id, sec_id, semester, year) references section on delete cascade,foreign key (ID) references student(ID) on delete cascade);

- create table advisor(s_ID varchar (5),i_ID varchar (5),primary key (s_ID),foreign key (i_ID) references instructor (ID) on delete set null,foreign key (s_ID) references student (ID) on delete cascade);

- create table prereq(course_id varchar(8),prereq_id varchar(8),primary key (course_id, prereq_id), foreign key (cours_id) references course(cours_id) on delete cascade,foreign key(prereq_id) references course(prereq_id));

- create table timeslot(time_slot_id varchar (4),day varchar(1),start_hr numeric (2) check (start_hr >= 0 and end_hr < 24),start_min numeric (2) check (start_min >= 0 and start_min < 60),end_hr numeric (2) check (end_hr >= 0 and end_hr < 24),end_min numeric (2) check (end_min >= 0 and end_min < 60),primary key(time_slot_id, day, start_hr, start_min));

Boat reservation database:

- CREATE TABLE Sailors (sid INTEGER, sname VARCHAR(10), rating INTEGER, age REAL, PRIMARY KEY (sid), CHECK (rating >= 1 AND rating <= 10));

- CREATE TABLE Boats(bid INTEGER, bname VARCHAR(10), color VARCHAR(10), PRIMARY KEY (bid));

- CREATE TABLE Reserves (sid INTEGER, bid INTEGER, day DATE, FOREIGN KEY (sid) REFERENCES Sailors on delete cascade, FOREIGN KEY (bid) REFERENCES Boats on delete cascade);

ii. ALTER: Alter command is used for alteration of table structures.

Syntax:

1. ALTER TABLE Table_name **ADD** (column_name1 DataType, column_name2 DataType……);
2. ALTER TABLE Table_name **MODIFY** (column_name1 DataType, column_name2 DataType……);
3. ALTER TABLE Table_name **DROP COLUMN** (column_name1 DataType, column_name2 DataType……);
4. ALTER TABLE Table_name **RENAME TO** new_table_name;
5. *ALTER TABLE Table_name **ADD CONSTRAINT** constraint_name constraint_type(col1, col2,...);*
6. *ALTER TABLE Table_name **DROP CONSTRAINT** constraint_name;*

Examples:

To add a column in a table:

ALTER TABLE Ticket ADD NoofSeats INT(2);

To delete a column in a table:
ALTER TABLE Ticket DROP COLUMN NoofSeats;

To rename the table name:
ALTER TABLE Cancellation RENAME TO Cancelation;

To change the data type of a column in a table:
ALTER TABLE Reservation MODIFY ContactNo BIGINT;
ALTER TABLE Cancelation MODIFY ContactNo BIGINT;

To change/add the constraint of a column in a table:
ALTER TABLE Ticket MODIFY Cost int(4) DEFAULT 1100;

To add a constraint in a table:
ALTER TABLE Passenger ADD CONSTRAINT tickconst FOREIGN KEY(TicketNo) REFERENCES Ticket(TicketNo);
To drop a constraint in a table:
ALTER TABLE Passenger DROP CONSTRAINT tickconst;

iii. DROP TABLE:
The SQL **DROP TABLE** statement is used to remove a table definition and all data, indexes, triggers, constraints, and permission specifications for that table.
Syntax: DROP TABLE Table_name;
Example: DROP TABLE Student;

- **DML (Data Manipulation Language):** DML provides the data manipulation techniques like selection, insertion, deletion, updating, modification, replacement, retrieval, sorting and display of data or records.

i. INSERT:
- The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax:
There are two basic syntaxes of INSERT INTO statement as follows:

METHOD 1: The SQL INSERT INTO syntax would be as follows:
Syntax: INSERT INTO TABLE_NAME [(column1, column2, column3,...columnN)] **VALUES**(value1, value2, value3,...valueN);

- Here, column1, column2,...columnN are the names of the columns in the table into which you want to insert data.
- You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

Example:
SQL> INSERT INTO STUDENT (REGNO, NAME, FEES) VALUES (1411, 'SHREYA',24000);
1 row created.

SQL> INSERT INTO STUDENT (REGNO, COMBINATION,FEES) VALUES(1412, 'PCMB',21000);
1 row created.

METHOD 2: The SQL INSERT INTO syntax would be as follows:
Syntax: **INSERT INTO** TABLE_NAME **VALUES** (value1,value2,value3,...valueN);

Example:

```
SQL> INSERT INTO STUDENT VALUES(1401,'RAMESH','PCMC','07-AUG-99',14000);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1402,'JOHN','PCMB','15-SEP-99',13500);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1403,'GANESH','PCME','19-AUG-99',16000);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1404,'MAHESH','PCMC','14-JAN-98',17650);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1405,'SURESH','PCMB','03-MAR-98',11500);
1 row created.
SQL> INSERT INTO STUDENT VALUES(1410,'ARUN','PCMC','01-APR-04',13000);
```

All the above statements would produce the following records in STUDENT table:

```
SQL> SELECT * FROM STUDENT;

     REGNO NAME            COMB DOB              FEES
---------- --------------- ---- --------- ----------
      1401 RAMESH          PCMC 07-AUG-99      14000
      1402 JOHN            PCMB 15-SEP-99      13500
      1403 GANESH          PCME 19-AUG-99      16000
      1404 MAHESH          PCMC 14-JAN-98      17650
      1405 SURESH          PCMB 03-MAR-98      11500
      1410 ARUN            PCMC 01-APR-04      13000

6 rows selected.
```

ii. UPDATE:

- SQL provides the ability to change data through UPDATE command.
- The UPDATE command used to modify or update an already existing row or rows of a table.
- The basic syntax of UPDATE command is given below.

UPDATE Table_name
SET column_name=value
[,column_name=value,…]
[**WHERE** condition];

Example:

```
SQL> UPDATE STUDENT SET COMBINATION='CEBA' WHERE REGNO=1411;
1 row updated.
SQL> UPDATE STUDENT SET NAME='AKASH' WHERE REGNO=1412;
 1 row updated.
```

iii. DELETE:

- In SQL, an already existing row or rows are removed from tables through the use of DELETE command.
- The basic syntax of DELETE command is given below.

DELETE Table_name
[**WHERE** condition];

Example:

```
SQL> DELETE STUDENT WHERE REGNO=1412;
1 row deleted.
```

iv. SELECT:

- SQL **SELECT** statement is used to fetch the data from a database table which returns data in the form of result table. These result tables are called result-sets.
- **Syntax:** The basic syntax of SELECT statement is as follows:

```
SELECT column1, column2,…,columnN
FROM Table_name
[WHERE condition(s)]
[GROUPBY column-list]
[HAVING condition(s)]
[ORDER BY column-name(s)];
```

- Here, column1, column2...are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax:

```
SELECT * FROM table_name;
```

- **Example:** Consider the STUDENT table having the following records:

```
SQL> SELECT * FROM STUDENT;

     REGNO NAME            COMB DOB              FEES
---------- --------------- ---- --------- ----------
      1401 RAMESH          PCMC 07-AUG-99      14000
      1402 JOHN            PCMB 15-SEP-99      13500
      1403 GANESH          PCME 19-AUG-99      16000
      1404 MAHESH          PCMC 14-JAN-98      17650
      1405 SURESH          PCMB 03-MAR-98      11500
      1410 ARUN            PCMC 01-APR-04      13000
      1411 SHREYA          CEBA                24000

7 rows selected.
```

- Following is an example, which would fetch REGNO, NAME and COMBINATION fields of the customers available in STUDENT table:

```
SQL> SELECT REGNO, NAME, COMBINATION FROM STUDENT;

     REGNO NAME            COMB
---------- --------------- ----
      1401 RAMESH          PCMC
      1402 JOHN            PCMB
      1403 GANESH          PCME
      1404 MAHESH          PCMC
      1405 SURESH          PCMB
      1410 ARUN            PCMC
      1411 SHREYA          CEBA

7 rows selected.
```

DISTINCT:

- The SQL **DISTINCT** keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.
- There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.
- **Syntax:** The basic syntax of DISTINCT keyword to eliminate duplicate records is as follows:

```
SELECT   DISTINCT  column1[, column2,…,column]
 FROM    Table_name
[WHERE   (condition)]
```

- **Example:** Consider the STUDENT table having the following records:

```
SQL> select * from student;

     REGNO NAME            COMB DOB              FEES
---------- --------------- ---- --------- ----------
      1401 RAMESH          PCMC 07-AUG-99      14000
      1402 JOHN            PCMB 15-SEP-99      13500
      1403 GANESH          PCME 19-AUG-99      16000
      1404 MAHESH          PCMC 14-JAN-98      17650
      1405 SURESH          PCMB 03-MAR-98      11500
      1410 ARUN            PCMC 01-APR-04      13000
      1411 SHREYA          CEBA                24000

7 rows selected.
```

- First, let us see how the following SELECT query returns duplicate combination records:

```
SQL> SELECT COMBINATION FROM STUDENT ORDER BY COMBINATION;

COMB
----
CEBA
PCMB
PCMB
PCMC
PCMC
PCMC
PCME

7 rows selected.
```

- Now, let us use DISTINCT keyword with the above SELECT query and see the result:

SQL> SELECT DISTINCT COMBINATION FROM STUDENT ORDER BY COMBINATION;

- This would produce the following result where we do not have any duplicate entry:

```
SQL> SELECT DISTINCT COMBINATION FROM STUDENT ORDER BY COMBINATION;

COMB
----
CEBA
PCMB
PCMC
PCME
```

WHERE clause – (Extracting specific rows)

- The SQL **WHERE** clause is used to specify a condition while fetching the data from single table or joining with multiple tables.
- If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.
- The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc., which we would examine in subsequent chapters.

Syntax: The basic syntax of SELECT statement with WHERE clause is as follows:

SELECT column1[,column2,columnN]
FROM Table_name
WHERE (condition)

- You can specify a condition using comparison or logical operators like >, <, =, LIKE, NOT, etc.
- Following is an example which would fetch REGNO, NAME and FEES fields from the STUDENT table where FEES is greater than 15000:

```
SQL> SELECT REGNO, NAME, FEES FROM STUDENT WHERE FEES>15000;

     REGNO NAME                 FEES
---------- --------------- ----------
      1403 GANESH               16000
      1404 MAHESH               17650
      1411 SHREYA               24000
```

- Following is an example, which would fetch REGNO, NAME and COMBINATION fields from the STUDENT table for a COMBINATION is "PCMC".
- Here, it is important to note that all the strings should be given inside single quotes (') where as numeric values should be given without any quote as in above example:

```
SQL> SELECT REGNO, NAME, COMBINATION FROM STUDENT WHERE COMBINATION='PCMC';

    REGNO NAME            COMB
--------- --------------- ----
     1401 RAMESH          PCMC
     1404 MAHESH          PCMC
     1410 ARUN            PCMC
```

v. DESCRIBE or DESC: This statement is used to obtain information about table structure.
Syntax: DESC tablename;

Bus reservation database Relation Instance:

Bus Table:

BusNo	Source	Destination	Dep_hr	Dep_min
AP101	Hyderabad	tirupathi	20	12
AP102	Hyderabad	bangalore	12	10
AP234	Hyderabad	kolkata	2	15
AP456	tirupathi	bangalore	13	45
AP123	Hyderabad	chennai	15	20

Passenger Table:

PNRNo	PPNo	Name	Age	Gender	Address	TicketNo
1234	1	ramesh	45	M	Hyderabad	103
1235	2	geetha	36	F	Tirupathi	102
1236	3	ramesh	30	M	Hyderabad	105
1237	4	ravi	50	M	Hyderabad	101
1238	5	seetha	32	F	Hyderabad	104

Ticket Table:

TicketNo	Jdate	Age	Gender	Source	Destination	Dep_hr	Dep_min	Cost
101	2016-01-12	50	M	Hyderabad	bangalore	12	10	300
102	2016-01-14	36	F	tirupathi	bangalore	13	45	500
103	2016-01-13	45	M	Hyderabad	tirupathi	20	12	250
104	2016-01-05	32	F	Hyderabad	kolkata	2	15	700
105	2016-01-18	30	M	Hyderabad	chennai	15	20	550

Reservation Table:

PNRNo	Jdate	NoofSeats	Address	ContactNo	Status
1235	2016-01-14	3	Tirupathi	9849562235	Yes
1238	2016-01-05	4	Hyderabad	8425365526	Yes
1237	2016-01-12	2	Hyderabad	7207153362	No
1234	2016-01-13	1	Hyderabad	8885426635	Yes
1236	2016-01-18	3	Hyderabad	9582153358	No

Cancellation Table:

PNRNo	Jdate	NoofSeats	Address	ContactNo	Status
1238	2016-01-05	4	Hyderabad	8425365526	Yes

Boats reservation database Relation Instance:

SAILORS			
SID	SNAME	RATING	AGE
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Dunkon	9	40
85	Ardhar	3	25.5
95	Bob	3	63.5

BOATS		
BID	BNAME	COLOR
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

RESERVES		
SID	BID	DAY
22	101	10-10-1998
22	102	10-10-1998
22	103	10-08-1998
22	104	10-07-1998
31	102	11-10-1998
31	103	11-06-1998
31	104	11-12-1998
64	101	09-05-1998
64	102	09-08-1998
74	103	09-08-1998