SQL

1. Table Name: PERSONS

P_ID	LASTNAME	FIRSTNAME	ADDRESS	CITY
1	HANSEN	OLA	TIMOTEIVN 10	SANDNES
2	SVENDSON	TOVE	BORGVN23	SANDNES
3	PETTERSEN	KARI	STORGT20	STAVANGER

a. Write a SQL statement to create a table Persons and insert the values.
b. To select the content of the columns named LASTNAME AND FIRSTNAME from the PERSONS table.
c. To select the all columns from the PERSONS table.
d. To select only the persons living in the city "SANDNES" from the table PERSONS.
e. To select only PERSONS with first name "TOVE" And last name "SVENDSON"
f. To select only Persons with first name "TOVE" OR "OLA".

2. Table Name: EMPLOYEE

E_ID	SALARY	BONUS
101	2000	200
102	3000	150
103	1000	100
104	2050	125
105	3500	200

a. To find total salary + Bonus from Employee table.
b. Using select statement Subtraction (-) operator used in SALARY & BONUS table.

3. Answer the a) & b) question on the basic of the following tables SHOPPE and ACCESSORIES.

TABLE: SHOPPE

ID	SNAME	AREA
S01	ABC COMPUTERONICS	CP
S02	ALL INFOTECH MEDIA	GK II
S03	TECH SHOPPE	CP
S04	GEEKS TECNO SOFT	Nehru Place
S05	HITECH TECH STORE	Nehru Place

TABLE: ACCESSORIES

NO	NAME	PRICE	ID
A01	Mother Board	12000	S01
A02	Hard Disk	5000	S01
A03	Keyboard	500	S02
A04	Mouse	300	S01
A05	Mother Board	13000	S02
A06	Keyboard	400	S03

A07	LCD	6000	S04
A08	LCD	5500	S05
A09	Mouse	350	S05
A10	Hard Disk	4500	S03

a) Write the SQL queries.

i) To display Name & Price of all the Accessories in ascending order of their price.

ii) To display ID AND SNAME of all Shoppe located in Nehru Place.

iii)To display Minimum and Maximum Price of each Name of Accessories.

iv) To display Name, Price of all Accessories and their respective SNAME, Where they are available.

b) Write the output of the following SQL commands.

i) SELECT DISTINCT NAME FROM ACCESSORIES WHERE PRICE>=5000;

ii) SELECT AREA, COUNT (*) FROM SHOPPE GROUP BY AREA;

iii) SELECT COUNT (DISTINCT AREA) FROM SHOPPE;

iv) SELECT NAME, PRICE*0.05 DISCOUNT FROM ACCESSORIES WHERE ID IN('S02', 'S03');

4. TABLE: ITEMS

CODE	INAME	QTY	PRICE	COMPANY	TCODE
1001	DIGITAL PAD 121	120	11000	XENITA	T01
1006	LED SCREEN 40	70	38000	SANTORA	T02
1004	CAR GPS SYSTEM	50	2150	GEOKNOW	T01
1003	DIGITAL CAMERA 12X	160	8000	DIGICLICK	T02
1005	PEN DRIVE 32 GB	600	1200	STOREHOME	T03

TABLE: TRADERS

TCODE	TNAME	CITY
T01	ELECTRONIC SALES	MUMBAI
T03	BUSY STORE CORP	DELHI
T02	DISP HOUSE INC	CHENNAI

a) To display the details of all the items in ascending order of item names (i.e. INAME)

b) To display item name and price of all those items, whose price is in the range of 10000 and 22000 (both values inclusive)

c) To display the number of items, which are traded by each trader. The expected output of this query should be:

T01	2
T02	2
T03	1

d) To display the price, item name and quantity of those items, which have quantity more than 150.

e) To display the names of those traders who are either from DELHI or from MUMBAI.

f) To display the name of companies and the name of the items in descending order of company names.

g) The expected output of these query should be:

i)

MAX(price)	MIN(price
38000	1200

ii)

AMOUNT
107500

iii)

TCODE
T01
T02
T03

iv)

INAME	TNAME
CAR GPS SYSTEM	ELECTRONIC SALES
LED SCREEN 40	DISP HOUSE INC

5. TABLE NAME: APPLICANTS

NO	NAME	FEE	GENDER	C_ID	JOINYEAR
1012	AMANDEEP	30000	M	A01	2012
1102	AVISHA	25000	F	A02	2009
1103	EKANT	30000	M	A02	2011
1049	ARUN	30000	M	A03	2009
1025	AMBER	40000	M	A02	2011
1106	ELA	40000	F	A05	2010
1017	NIKITA	35000	F	A03	2012
1108	ARLUNA	30000	F	A03	2012
2109	SHAKTI	35000	M	A04	2011
1101	KIRAT	25000	M	A01	2012

TABLE NAME: COURSES

C_ID	COURSE
A01	FASHION DESIGN
A02	NETWORKING
A03	HOTEL MANAGEMENT
A04	EVENT MANAGEMENT
A05	OFFICE MANAGEMENT

a) To display NAME, FEE, GENDER, JOINYEAR about the APPLICANTS, who have joined before 2010.

b) To display the names of applicants, who are paying FEE more than 30000.

c) To display the names of all applicants in ascending order of their JOINYEAR.

d) To display the year and the total number of applicants joined in each year from the table APPLICANTS.

e) To display the C_ID and the number of applicants registered in the course from the APPLICANTS table.

f) To display the applicant's name with their respective courses name from the tables applicants and courses.

g) Give the SQL statements of the outputs:

i)

NAME	JOINYEAR
AVISHA	2009

ii)

MIN(JOINYEAR)
2009

iii)

AVG(FEE)
31666.666

iv)

SUM(FEE)	C_ID
55000	A01

NOTES:

The SQL DELETE Statement:

The DELETE statement is used to delete existing records in a table.

Syntax: DELETE FROM table_name WHERE condition;

The SQL WHERE Clause

The WHERE clause is used to filter records. The WHERE clause is used to extract only those records that fulfill a specified condition.

Syntax:

SELECT column1, column2, ...
FROM table_name
WHERE condition;

The SQL SELECT DISTINCT Statement: The SELECT DISTINCT statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

Syntax:

SELECT DISTINCT column1, column2, ...
FROM table_name;

The SQL SELECT Statement: The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

Syntax: SELECT * FROM table_name;

SQL

6. TABLE: CUSTOMER

CID	CNAME	GENDER	SID	AREA
1001	R SHARMA	FEMALE	101	NORTH
1002	M R TIWARY	MALE	102	SOUTH
1003	M K KHAN	MALE	103	EAST
1004	A K SINGH	MALE	102	EAST
1005	S SEN	FEMALE	101	WEST
1006	R DUBEY	MALE	104	NORTH
1007	M AGARWAL	FEMALE	104	NORTH
1008	S DAS	FEMALE	103	SOUTH
1009	R K PATIL	MALE	102	NORTH
1010	N KRISHNA MURTY	MALE	102	SOUTH

TABLE: ONLINESHOP

SID	SHOP
101	MY BUY
102	ECO BUY
103	JUST SHOPPING
104	SHOPPING EASY

a) To display CNAME, AREA of all female customers from CUSTOMER table.

b) To display the details of all the CUSTOMERS in ascending order of CNAME within SID.

c) To display the total number of customers for each AREA from CUSTOMER table.

d) To display CNAME and CORRESPONDING SHOP from CUSTOMER table and ONLINESHOP table.

e) Write the code to see the output:

COUNT(*)	GENDER
4	FEMALE
6	MALE

f) Write the code to see the output:

COUNT(*)
4

g) SELECT CNAME FROM CUSTOMER WHERE CNAME LIKE ‘L%’; [Write the output]

h) SELECT DISTINCT AREA FROM CUSTOMER; [Write the output]

NOTES

UPDATE: The UPDATE statement is used to modify the existing records in a table.
Syntax: UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;

AND, OR and NOT Operators: The WHERE clause can be combined with AND, OR, and NOT operators.

SQL

The AND & OR operators are used to filter records based on more than one condition:
The AND operator displays a record if all the conditions separated by AND are TRUE.
The OR operator displays a record if any of the conditions separated by OR is TRUE.
The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax:

SELECT column1, column2, ... FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;

OR Syntax:

SELECT column1, column2, ... FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;

NOT Syntax:

SELECT column1, column2, ... FROM table_name
WHERE NOT condition;

7. TABLE: CARDEN

CCODE	CARNAME	MAKE	COLOR	CAPACITY	CHARGES
501	A-STAR	SUZUKI	RED	3	14
503	INDIGO	TATA	SILVER	3	12
502	INNOVA	TOYOTA	WHITE	7	15
509	SX4	SUZUKI	SILVER	4	14
510	C-CLASS	MERECEDES	RED	4	35

TABLE: CUSTOMER

CODE	CNAME	CCODE
1001	HAMANT SAHU	501
1002	RAJ LAL	509
1003	FEROZA SHAH	503
1004	KETAN DHAL	502

a)
i) To display the name of all the SILVER colored cars.
ii) To display name of Car, Make and sitting Capacity of cars in descending order of their sitting CAPACITY.
iii) To display the highest charges at which a vehicle can be hired from CARDEN.
iv) To display the CUSTOMER name and the corresponding name of the CARS hired by them.
b) Give the output of following SQL queries:
i) SELECT COUNT(DISTINCT MAKE) FROM CARDEN;
ii) SELECT MAX(CHARGES), MIN(CHARGES) FROM CARDEN;
iii) SELECT COUNT(*) MAKE FROM CARDEN;
iv) SELECT CARNAME FROM CARDEN WHERE CAPACITY=4;

SQL

SQL RENAME TABLE: SQL RENAME TABLE syntax is used to change the name of a table. Sometimes, we choose non-meaningful name for the table. So it is required to be changed.
Syntax: ALTER TABLE table_name RENAME TO new_table_name;

SQL COPY TABLE: If you want to copy a SQL table into another table in the same SQL server database, it is possible by using the select statement.
Syntax: SELECT * INTO <destination_table> FROM <source_table> ;

SQL ALTER TABLE DROP Column Syntax:
ALTER TABLE table_name DROP COLUMN column_name;

SQL ALTER TABLE Add Column Syntax:
ALTER TABLE table_name ADD column_name datatype;

SQL SELECT FIRST: The SQL first() function is used to return the first value of the selected column. Syntax: SELECT FIRST(column_name) FROM table_name;[The SELECT FIRST statement is only supported by MS Access]

SQL SELECT LAST: The last() function is used to return the last value of the specified column. Syntax: SELECT LAST (column_name) FROM table_name;[;[The SELECT LAST statement is only supported by MS Access]

SQL SELECT SUM Syntax: SELECT SUM (expression) FROM tables WHERE conditions;

8. Create a table to store information about weather observation stations. [No duplicate ID fields allowed]

```
CREATE TABLE STATION
(ID INTEGER PRIMARY KEY,
CITY CHAR(20),
STATE CHAR(2),
LAT_N REAL,
LONG_W REAL);
```

Populate the table STATION with a few rows:

```
INSERT INTO STATION VALUES (13, 'Phoenix', 'AZ', 33, 112);
INSERT INTO STATION VALUES (44, 'Denver', 'CO', 40, 105);
INSERT INTO STATION VALUES (66, 'Caribou', 'ME', 47, 68);
```

1.i) Display the all stations.
ii) Query to select Northern stations (Northern latitude > 39.7).

iii) Query to select only ID, CITY, and STATE columns.

2. Create another table to store normalized temperature and precipitation data:

```
-- ID field must match some STATION table ID(so name and location will be known).
-- Allowable ranges will be enforced for other values.
-- No duplicate ID and MONTH combinations.
-- Temperature is in degrees Fahrenheit.
-- Rainfall is in inches.
CREATE TABLE STATS
(ID INTEGER REFERENCES STATION(ID),
MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
TEMP_F REAL CHECK (TEMP_F BETWEEN -80 AND 150),
RAIN_I REAL CHECK (RAIN_I BETWEEN 0 AND 100),
PRIMARY KEY (ID, MONTH));

INSERT INTO STATS VALUES (13, 1, 57.4, 0.31);
INSERT INTO STATS VALUES (13, 7, 91.7, 5.15);
INSERT INTO STATS VALUES (44, 1, 27.3, 0.18);
INSERT INTO STATS VALUES (44, 7, 74.8, 2.11);
INSERT INTO STATS VALUES (66, 1, 6.7, 2.10);
INSERT INTO STATS VALUES (66, 7, 65.8, 4.52);
```

i) Display the all values of STATS table.

ii) Query to look at table STATS, picking up location information by joining with table STATION on the ID column.

iii) Query to look at the table STATS, ordered by month and greatest rainfall, with columns rearranged.

iv) Query to look at temperatures for July from table STATS, lowest temperatures first, picking up city name and latitude by joining with table STATION on the ID column.

v) Query to show MAX and MIN temperatures as well as average rainfall for each station.

vi) Query (with sub query) to show stations with year-round average temperature above 50 degrees.

vii) Create a view (derived table or persistent query) to convert Fahrenheit to Celsius and inches to centimeters.

viii) Query to look at table STATS in a metric light (through the new view).

ix) Another metric query restricted to January below-freezing (0 Celsius) data, sorted on rainfall.

x) Update all rows of table STATS to compensate for faulty rain gauges known to read 0.01 inches low.

SQL

Create a database using the CREATE DATABASE command:

CREATE DATABASE database_name;

Insert data multiple rows, use a comma to separate each row, like this:

INSERT INTO table_name
VALUES
(value_1, value_2, value_3),
(value_1, value_2, value_3),
(value_1, value_2, value_3),
(value_1, value_2, value_3);

SQL PRIMARY KEY: Primary keys must contain UNIQUE values, and cannot contain NULL values.

SQL PRIMARY KEY on CREATE TABLE EXMAPLE:

MySQL:	SQL Server / Oracle / MS Access
CREATE TABLE Persons (ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int, PRIMARY KEY (ID));	CREATE TABLE Persons (ID int NOT NULL PRIMARY KEY, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int);

9. Create table Worker.

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
001	NIHARIKA	ARORA	20000	2013-02-25 09:00:00	HR
002	AYUSHI	GURONDIA	5000	2015-02-10 09:00:00	ADMIN
003	PRIYANSHA	CHOUKSEY	25000	2014-05-16 09:00:00	HR
004	APARNA	DESHPANDE	8000	2016-12-20 09:00:00	ADMIN
005	SHAFALI	JAIN	21000	2015-08-29 09:00:00	ADMIN
006	SUCHITA	JOSHI	20000	2017-02-12 09:00:00	ACCOUNT
007	SHUBHI	MISHRA	15000	2018-03-23 09:00:00	ADMIN
008	DEVYANI	PATIDAR	18000	2014-05-02 09:00:00	ACCOUNT

SQL

1. i) Query to create the Table

CREATE TABLE Worker (WORKER_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT, FIRST_NAME CHAR(25), LAST_NAME CHAR(25),SALARY INT(15),JOINING_DATE DATETIME,DEPARTMENT CHAR(25));

ii) Query to insert values into the Table Worker.

iii) Create table **Bonus**

WORKER_REF_ID	BONUS_DATE	BONUS_AMOUNT
1	2015-04-20 00:00:00	5000
2	2015-08-11 00:00:00	3000
3	2015-04-20 00:00:00	4000
1	2015-04-20 00:00:00	4500
2	2015-08-11 00:00:00	3500

CREATE TABLE Bonus(WORKER_REF_ID INT, BONUS_DATE DATETIME, BONUS_AMOUNT INT(10), FOREIGN KEY (WORKER_REF_ID) REFERENCES WORKER(WORKER_ID) ON DELETE CASCADE);

iv)Query to insert values into table Bonus.

v) Create table **Title**

WORKER_REF_ID	WORKER_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00
4	Asst. Manager	2016-06-11 00:00:00
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

CREATE TABLE Title (WORKER_REF_ID INT, WORKER_TITLE CHAR(25), AFFECTED_FROM DATETIME, FOREIGN KEY (WORKER_REF_ID) REFERENCES Worker(WORKER_ID) ON DELETE CASCADE);

vi) Query to insert values into table Title.

2. i) Write an SQL query for fetching “FIRST_NAME” from the WORKER table using <WORKER_NAME> as alias.

ii) What is an SQL Query for fetching the “FIRST_NAME” from WORKER table in upper case?

iii) What is an SQL query for fetching the unique values of the column DEPARTMENT from the WORKER table?

iv) Write an SQL query for printing the first three characters of the column FIRST_NAME.

v) What is an SQL query for finding the position of the alphabet (‘A’) in the FIRST_NAME column of Ayushi.

vi) What is an SQL Query for printing the FIRST_NAME from Worker Table after the removal of white spaces from right side.

vii) Write an SQL Query for printing the DEPARTMENT from Worker Table after the removal of white spaces from the left side.
viii) What is an SQL query for fetching the unique values from the DEPARTMENT column and thus printing is the length?
ix) Write an SQL query for printing the FIRST_NAME after replacing 'A' with 'a'.
x) What is an SQL query for printing the FIRST_NAME and LAST_NAME into a column named COMPLETE_NAME? (A space char should be used)
xi) What is an SQL query for printing all details of the worker table which ordered by FIRST_NAME ascending?
xii) Write an SQL query for printing the all details of the worker table which ordered by FIRST_NAME ascending and the DEPARTMENT in descending
xiii) What is an SQL query to print the details of the workers 'NIHARIKA' and 'PRIYANSHA'.
xiv) What is an SQL query printing all details of workers excluding the first names of 'NIHARIKA' and 'PRIYANSHA'
xv) Write an SQL query for printing the details of DEPARTMENT name as "Admin".
xvi) What is an SQL query for printing the details of workers whose FIRST_NAME Contains 'A'?
xvii) What is an SQL Query for printing the FIRST_NAME of workers whose name ends with 'A'?
xviii) What is an SQL Query for printing the details of the workers whose FIRST_NAME ends with 'H' and contains six alphabets?
xix) Write an SQL Query for printing the details of workers whose SALARY lies between 10000 and 20000.
xx) Write an SQL Query for printing the details of workers who joined inFeb'2014
xxi) Write an SQL Query for fetching the count of workers in DEPARTMENT with 'Admin'.
xxii) Write an SQL Query for fetching the details of workers with Salaries >= 5000 and <= 10000.
xxiii) What is an SQL Query for fetching the no. of workers in each department in descending order?
xxiv) What is an SQL Query for printing the details of workers who are also managers?
xxv) Write an SQL Query for fetching the details of duplicate records in some fields.
xxvi) What is an SQL Query for only showing odd rows?
xxvii) What is an SQL Query for only showing even rows?
xxviii) Write an SQL Query for cloning a new table from another table.
xxix) Write an SQL query to show the top n (say 10) records of a table.
xxx) Write an SQL query to determine the 5th highest salary from a table.

3.

i) Write an SQL query to fetch the list of employees with the same salary.
ii) Write an SQL query to show the second highest salary from a table.
iii) Write an SQL query to show one row twice in results from a table.
iv) Write an SQL query to fetch the first 50% records from a table.

v) Write an SQL query to fetch the departments that have less than five people in it.
vi) Write an SQL query to show the last record from a table.
vii) Write an SQL query to fetch the first row of a table.
viii) Write an SQL query to print the name of employees having the highest salary in each department.

Drop Column Syntax:

MYSQL	Oracle and SQL Server
ALTER TABLE "table_name" DROP "column_name";	ALTER TABLE "table_name" DROP COLUMN "column_name";

Rename Column Syntax:

MySQL	Oracle
ALTER TABLE "table_name" Change "column 1" "column 2" ["Data Type"];	ALTER TABLE "table_name" RENAME COLUMN "column 1" TO "column 2";

MYSQL NUMERIC FUNCTIONS

ABS() - SELECT ABS(-243.5);	CEIL() - SELECT CEIL(35.75);
ACOS() - SELECT ACOS(0.5);	CEILING() - SELECT CEILING(25.75);
ATAN() - SELECT ATAN(.25);	COS() - SELECT COS(1);
ATAN2()- SELECT ATAN2(0.4, 2);	COT() - SELECT COT(6);
DEGREES() - SELECT DEGREES(2.5);	GREATEST()- SELECT GREATEST(30, 120, 20, 81, 205);
EXP() - SELECT EXP(3);	FLOOR() - SELECT FLOOR(45.95);
DIV() - SELECT 200 DIV 2;	LEAST()-SELECT LEAST(8, 102, 74, 81, 275);
LN() (logarithm) -SELECT LN(10);	LOG()-SELECT LOG(4, 10);
LOG10() - SELECT LOG10(2);	MOD()-SELECT MOD(28, 3);
PI() - SELECT PI();	POW()-SELECT POW(5, 3);
POWER()-SELECT POWER(7, 3);	RADIANS() - SELECT RADIANS(180);
RAND()-SELECT RAND();	ROUND()-SELECT ROUND(135.375, 2);
SIGN() - SELECT SIGN(-78);	SQRT()-SELECT SQRT(25);
TAN() - SELECT TAN(90);	TRUNCATE()- SELECT TRUNCATE(7235.37589, 2);
ASIN() - SELECT ASIN(0.5);	CONV()- SELECT CONV('a',16,2);

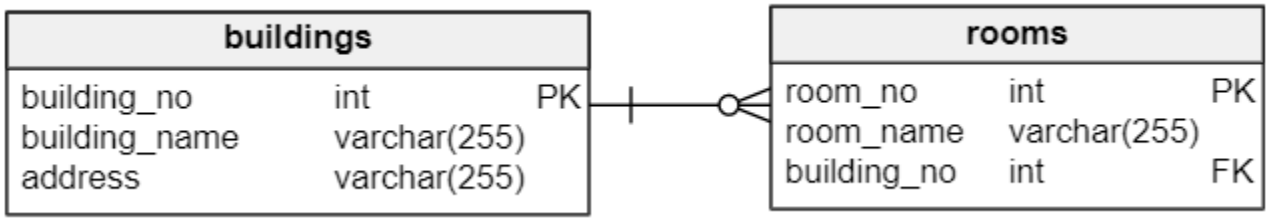
SQL

10. Table: Student

ID	STD_ID	NAME	MARKS
1	3	ABHI	99
2	5	GEETHASRI	89
3	6	RAHIM	49
4	9	RAM	69
5	1	RAHUL	87
6	1	RAHUL	96
7	1	RAHUL	96
8	9	RAM	96
9	9	RAM	96

1. Query To Find Second Highest Marks Of A Student?
2. Query To Find Duplicate Rows In Table?
3. What Is The Query To Fetch First Record From Student Table?
4. What Is The Query To Fetch Last Record From The Student Table?
5. What Is Query to Display First 4 Records from Student Table?
6. What Is Query to Display Last 3 Records from Student Table?
7. What Is Query To Display Nth Record From Student Table?
8. How to Get 3 Highest Marks from Student Table?
9. How to Display Odd Rows in Student Table?
10. How to Display Even Rows in Student Table?
11. How Can I Create Table With Same Structure Of Student Table?
12. Select All Records from Student Table Whose Name Is ‘abhi’ and ‘geethasri’.

MySQL 5.7 ON DELETE CASCADE: Deleting Data from Multiple Related Tables:



1. Create the buildings table:

CREATE TABLE buildings (building_no INT PRIMARY KEY AUTO_INCREMENT, building_name VARCHAR(255) NOT NULL, address VARCHAR(255) NOT NULL);

2. Create the rooms table:

CREATE TABLE rooms (room_no INT PRIMARY KEY AUTO_INCREMENT, room_name VARCHAR(255) NOT NULL, building_no INT NOT NULL, FOREIGN KEY (building_no) REFERENCES buildings (building_no) ON DELETE CASCADE);

3. Insert data into the buildings table:

INSERT INTO buildings(building_name,address) VALUES('ACME Headquaters','3950 North 1st Street CA 95134'), ('ACME Sales','5000 North 1st Street CA 95134');

4. SELECT * FROM buildings;

5. Insert data into the rooms table:

INSERT INTO rooms(room_name,building_no) VALUES('Amazon',1),('War Room',1), ('Office of CEO',1),('Marketing',2),('Showroom',2);

6. SELECT * FROM rooms;

Delete the building with building no. 2:

DELETE FROM buildings WHERE building_no = 2;

SELECT * FROM rooms;

Note: ON DELETE CASCADE means that if the parent record is deleted, any child records are also deleted.

Primary Key: A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.

FOREIGN KEY: A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

……..

11. TABLE: EMPLOYEE

ECODE	NAME	DESIGN	SGRADE	DOJ	DOB
101	ABDUL AHMED	EXECUTIVE	S03	23-MAR-2003	13-JAN-1980
102	RAVI CHANDER	HEAD-IT	S02	12-FEB-2010	22-JUL-1987
103	JOHN KEN	RECEPTIONIST	S03	24-JUN-2009	24-FEB-1983
105	NAZAR AMEEN	GM	S02	11-AUG-2006	03-MAR-1984
108	PRIYAM SEN	CEO	S01	29-DEC-2004	19-JAN-1982

[Notes: Aggregate Functions are all about: Performing calculations on multiple rows Of a single column of a table And returning a single value. The ISO standard defines five (5) aggregate functions namely; 1) COUNT 2) SUM 3) AVG 4) MIN 5) MAX]

TABLE: SALGRADE

SGRADE	SALARY	HRA
S01	56000	18000
S02	32000	12000
S03	24000	8000

a) i) To display the details of all the employee in descending order of DOJ.
ii) To display NAME and DESIGN of those Employees, whose SGRADE is either S02 or S03.
iii) To display the content of all the EMPLOYEEs, whose DOJ is between '09-FEB-2006' and '08-AUG-2009'.
iv) To add a new row in the EMPLOYEE table with the following:
109, 'HARISH ROY', 'HEAD-IT', 'S02', '09-SEP-2007', '21-APR-1983'.
b) Give the output of the following SQL queries:
i) SELECT COUNT(SGRADE), SGRADE FROM EMPLOYEE GROUP BY SGRADE;
ii) SELECT MIN(DOB), MAX(DOJ) FROM EMPLOYEE;
iii) SELECT NAME, SALARY FROM EMPLOYEE E, SALGRADE S WHERE E.SGRADE =S.SGRADE AND E.ECODE<103;
iv) SELECT SGRADE, SALARY + HRA FROM SALGRADE WHERE SGRADE='S02';

12. TABLE: STORE

ITEMNO	ITEM	SCODE	OTY	RATE	LASTBUY
2005	SHARPENER CLASSIC	23	60	8	31-JUN-09
2003	BALLS	22	50	25	01-FEB-10
2002	GEL PEN PREMIUM	21	150	12	24-FEB-10
2006	GEL PEN CLASSIC	21	250	20	11-MAR-09
2001	ERASER SMALL	22	220	6	19-JAN-09
2004	ERASER BIG	22	110	8	02-DEC-09
2009	BALL PEN 0.5	21	180	18	03-NOV-09

TABLE: SUPPLIERS

SCODES	SNAME
21	PREMIUM STATIONERS
23	SOFT PLASTICS
22	TERA SUPPLY

A) i) To display details of all the items in the store table in ascending order of Last-Buy.
ii) To display ItemNo and Item name of those items from STORE table, whose Rate is more than 15;
iii) To display the details of those items whose Supplier code (Scode) is 22 or Quantity in Store (Qty) is more than 110 from the table STORE.
iv) To display minimum rate of items for each Supplier individually as per Scode from the table STORE.

b) Give the output of the following SQL queries:

i) SELECT COUNT(DISTINCT SCODE) FROM STORE;

ii) SELECT RATE * QTY FROM STORE WHERE ITEMNO=2004;

iii) SELECT ITEM, SNAME FROM STORE S, SUPPLIERS P WHERE S.SCODE=P.SCODE AND ITEMNO=2006;

iv) SELECT MAX(LASTBUY) FROM STORE;

13. TABLE: STUDENT

SCODE	NAME	AGE	STRCDE	POINTS	GRADE
101	AMIT	16	1	6	NULL
102	ARJUN	13	3	4	NULL
103	ZAHEER	14	2	1	NULL
105	GAGAN	15	5	2	NULL
108	KUMAR	13	6	8	NULL
109	RAJESH	17	5	8	NULL
110	NAVEEN	13	3	9	NULL
113	AJAY	16	2	3	NULL
115	KAPIL	14	3	2	NULL
120	GURDEEP	15	2	6	NULL

TABLE: STREAM

STRCDE	STRNAME
1	SCIENCE +COMP
2	SCIENCE + BIO
3	SCIENCE + ECO
4	COMMERCE + MATHS
5	COMMERCE + SOCIO
6	ARTS + MATHS
7	ARTS + SOCIO

a) To display the name of stream in alphabetical order from table STREAM.

b) To display the number of students whose POINTS are more than 5.

c) To update GRADE to ‘A’ for all those students, who are getting more than 8 as POINTS.

d) ARTS+ MATHS stream is no more available. Make necessary change in table STREAM.

e) SELECT SUM(POINTS) FROM STUDENT WHERE AGE>14;

f) SELECT STRCDE, MAX(POINTS) FROM STUDENT GROUP BY STRCDE HAVING SCODE BETWEEN 105 AND 130;

g) SELECT AVG(AGE) FROM STUDENT WHERE SCODE IN(102,105,110,120);

h) SELECT COUNT(STRNAME) FROM STREAM WHERE STRNAME LIKE “SCI%”;

SQL

14. Consider the following tables GARMENT and FABRIC. Write SQL commands for the statements (a) to (d) and give outputs for SQL queries (e) to (h).

TABLE: GARMENT

GCODE	DESCRIPTION	PRICE	FCODE	READYDATE
10023	PENCIL SKIRT	1150	F03	19-DEC-08
10001	FORMAL SHIRT	1250	F01	12-JAN-08
10012	INFORMAL SHIRT	1550	F02	06-JUN-08
10024	BABY TOP	750	F03	07-APR-07
10090	TULIP SKIRT	850	F02	31-MAR-07
10019	EVENING GOWN	850	F03	06-JUN-08
10009	INFORMAL PANT	1500	F02	20-OCT-08
10007	FORMAL PANT	1350	F01	09-MAR-08
10020	FROCK	850	F04	09-SEP-07
10089	SLACKS	750	F03	20-OCT-08

TABLE: FABRIC

FCODE	TYPE
F04	POLYSTER
F02	COTTON
F03	SILK
F01	TERELENE

a) To display GCODE and DESCRIPTION of each GARMENT in descending order of GCODE.

b) To display the details of all the GARMENTs, which have READYDATE in between 08-DEC-07 and 16-JUN-08 (inclusive of both dates).

c) To display the average PRICE of all the GARMENTS, which are made up of FABRIC with FCODE as F03.

d)To display FABRIC wise highest and lowest price of GARMENTS from GARMENT table. (Display FCODE of each GARMENT along with highest and lowest price).

e) SELECT SUM(PRICE) FROM GARMENT WHERE FCODE='F01';

f) SELECT DESCRITION, TYPE FROM GARMENT, FABRIC WHERE GARMENT.FCODE =FABRIC.FCODE AND GARMENT.PRICE>=1260;

g) SELECT MAX(FCODE) FROM FABRIC;

h) SELECT COUNT (DISTINCT PRICE) FROM GARMENT;

15. TABLE: SENDER

SENDERID	SENDERNAME	SENDERADDRESS	SENDERCITY
ND01	R JAIN	2. ABC APPTS	NEW DELHI
MU02	H SINHA	12. NEWTOWN	MUMBAI
MU15	S JHA	27/A, PARK STREET	MUMBAI
ND50	T PRASAD	122-K,SDA	NEW DELHI

SQL

TABLE: RECIPIENT

RECID	SENDERID	RECNAME	RECADDRESS	RECCITY
KO05	ND01	R BAJPAYEE	5, CENTRAL AVENUE	KOLKATA
ND08	MU02	S MAHAJAN	116,A VIHAR	NEW DELHI
MU19	ND01	H SINGH	2A, ANDHERI EAST	MUMBAI
MU32	MU15	P K SWAMY	B5,C S TERMINUS	MUMBAI
ND48	ND50	S TRIPATHI	13, B1 D, MAYUR VIHAR	NEW DELHI

a) To display the names of all senders from MUMBAI.

b) To display the RECID, SENDERNAME, SENDERADDRESS, RECNAME, RECADDRESS for every RECIPIENT.

c) To display RECIPIENT details in ascending order of RECNAME.

d) To display number of Recipients from each city.

e) Write SQL code to see the output:

DISTINCT SENDERCITY
NEW DELHI
MUMBAI

f) Write SQL code to see the output:

SENDERNAME	RECNAME
R JAIN	H SINGH
S JHA	P K SWAMY

g) Write SQL code to see the output:

RECNAME	RECADDRESS
S MAHAJAN	116, A VIHAR
S TRIPATHI	13,B1D, MAYUR VIHAR

MySQL HAVING CLAUSE:

MySQL HAVING Clause is used with GROUP BY clause. It always returns the rows where condition is TRUE. Syntax: SELECT expression1, expression2, ... expression_n, aggregate_function (expression) FROM tables [WHERE conditions] GROUP BY expression1, expression2, ... expression_n HAVING condition;

MySQL SUBQUERIES:

A subquery is a query in a query. It is also called an inner query or a nested query. A subquery can be used anywhere an expression is allowed. It is a query expression enclosed in parentheses. Subqueries can be used with SELECT, INSERT, UPDATE, or DELETE statements.

```
mysql> SELECT * FROM Cars;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
```

```
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
```

```
mysql> SELECT * FROM Customers; SELECT * FROM Reservations;
+------------+-------------+
| CustomerId | Name        |
+------------+-------------+
|          1 | Paul Novak  |
|          2 | Terry Neils |
|          3 | Jack Fonda  |
|          4 | Tom Willis  |
+------------+-------------+
4 rows in set (0.00 sec)

+----+------------+------------+
| Id | CustomerId | Day        |
+----+------------+------------+
|  1 |          1 | 2009-11-22 |
|  2 |          2 | 2009-11-28 |
|  3 |          2 | 2009-11-29 |
|  4 |          1 | 2009-11-29 |
|  5 |          3 | 2009-12-02 |
+----+------------+------------+
5 rows in set (0.00 sec)
```

Subquery with the INSERT statement: To create a copy of the Cars table. Into another table called Cars2. We will create a subquery for this.

mysql> CREATE TABLE Cars2(Id INT NOT NULL PRIMARY KEY, Name VARCHAR(50) NOT NULL, Cost INT NOT NULL);

We create a new Cars2 table with the same columns and datatypes as the Cars table. To find out how a table was created, we can use the SHOW CREATE TABLE statement.

mysql> INSERT INTO Cars2 SELECT * FROM Cars;

This is a simple subquery. We insert all rows from the Cars table into the Cars2 table.

```
mysql> SELECT * FROM Cars2;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
```

The data was copied to a new Cars2 table.

Scalar subqueries: A scalar subquery returns a single value.

mysql> SELECT Name FROM Customers WHERE CustomerId=(SELECT CustomerId FROM Reservations WHERE Id=5);

```
+------------+
| Name       |
+------------+
| Jack Fonda |
+------------+
```

Table subqueries: A table subquery returns a result table of zero or more rows.

mysql> SELECT Name FROM Customers WHERE CustomerId IN (SELECT DISTINCT CustomerId FROM Reservations);

```
+-------------+
| Name        |
+-------------+
| Paul Novak  |
| Terry Neils |
| Jack Fonda  |
+-------------+
```

The above query returns the names of the customers, who made some reservations. The inner query returns customer Ids from the Reservations table. We use the IN predicate to select those names of customers, who have their CustomerId returned from the inner select query. The previous subquery can be rewritten using SQL join.

mysql> SELECT DISTINCT Name FROM Customers JOIN Reservations ON Customers.CustomerId=Reservations.CustomerId;

```
+-------------+
| Name        |
+-------------+
| Paul Novak  |
| Terry Neils |
| Jack Fonda  |
```

Correlated subqueries: A correlated subquery is a subquery that uses values from the outer query in its WHERE clause. The subquery is evaluated once for each row processed by the outer query.

mysql> SELECT Name FROM Cars WHERE Cost < (SELECT AVG(Cost) FROM Cars);

```
+------------+
| Name       |
+------------+
| Audi       |
| Mercedes   |
| Skoda      |
| Volvo      |
| Citroen    |
| Hummer     |
| Volkswagen |
+------------+
```

Subqueries with EXISTS, NOT EXISTS: If a subquery returns any values, then the predicate EXISTS returns TRUE, and NOT EXISTS FALSE.

mysql> SELECT Name FROM Customers WHERE EXISTS (SELECT * FROM Reservations WHERE Customers.CustomerId=Reservations.CustomerId);

```
+-------------+
| Name        |
+-------------+
| Paul Novak  |
| Terry Neils |
| Jack Fonda  |
+-------------+
```

In the above SQL statement we select all customers' names, which have an entry in the Reservations table.

mysql> SELECT Name FROM Customers WHERE NOT EXISTS (SELECT * FROM Reservations WHERE Customers.CustomerId=Reservations.CustomerId);

```
+------------+
| Name       |
+------------+
| Tom Willis |
+------------+
```

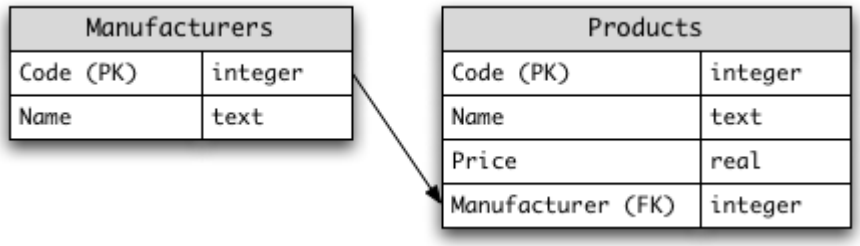
In this query, we return all customers that do not have an entry in the Reservations table. Both SQL queries are correlated queries.

SQL Aliases Notes: SQL aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable. An alias only exists for the duration of the query.

Alias Column Syntax: SELECT column_name AS alias_name FROM table_name;

Alias Table Syntax: SELECT column_name(s) FROM table_name AS alias_name;

16. TABLE SCHEMA FIGURE:



1. Create a table Manufactures and Products. And Insert the values:

INSERT INTO Manufacturers(Code,Name) VALUES(1,'Sony');
INSERT INTO Manufacturers(Code,Name) VALUES(2,'Creative Labs');
INSERT INTO Manufacturers(Code,Name) VALUES(3,'Hewlett-Packard');
INSERT INTO Manufacturers(Code,Name) VALUES(4,'Iomega');
INSERT INTO Manufacturers(Code,Name) VALUES(5,'Fujitsu');
INSERT INTO Manufacturers(Code,Name) VALUES(6,'Winchester');
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(1,'Hard drive',240,5);

SQL

```
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(2,'Memory',120,6);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(3,'ZIP drive',150,4);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(4,'Floppy disk',5,6);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(5,'Monitor',240,1);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(6,'DVD drive',180,2);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(7,'CD drive',90,2);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(8,'Printer',270,3);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(9,'Toner cartridge',66,3);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(10,'DVD burner',180,2);
```

TABLE CREATION CODE [THIS CODE IS NOT PROVIDE IN EXAM HALL]

TABLE - Manufacturers	TABLE- Products
CREATE TABLE Manufacturers (Code INTEGER, Name VARCHAR(255) NOT NULL, PRIMARY KEY (Code));	CREATE TABLE Products (Code INTEGER, Name VARCHAR(255) NOT NULL , Price DECIMAL NOT NULL , Manufacturer INTEGER NOT NULL, PRIMARY KEY (Code), FOREIGN KEY (Manufacturer) REFERENCES Manufacturers(Code));

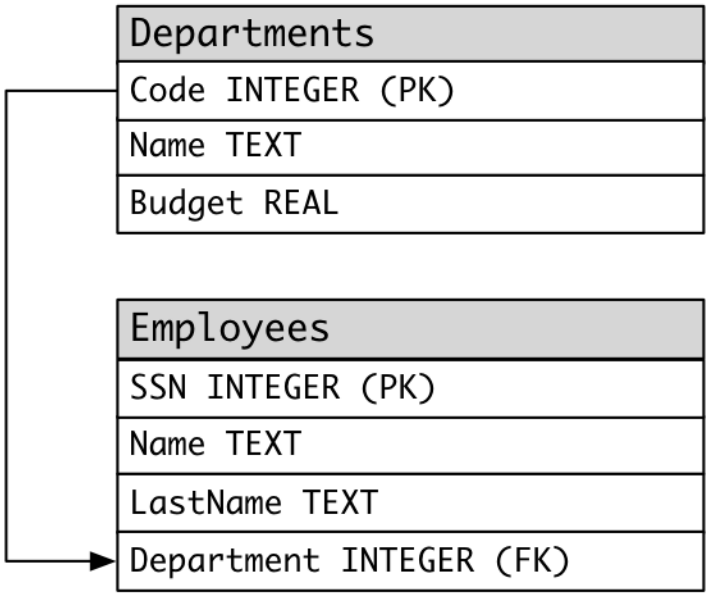
1.1 Select the names of all the products in the store.
1.2 Select the names and the prices of all the products in the store.
1.3 Select the name of the products with a price less than or equal to $200.
1.4 Select all the products with a price between $60 and $120.
1.5 Select the name and price in cents (i.e., the price must be multiplied by 100).
1.6 Compute the average price of all the products.
1.7 Compute the average price of all products with manufacturer code equal to 2.
1.8 Compute the number of products with a price larger than or equal to $180.
1.9 Select the name and price of all products with a price larger than or equal to $180, and sort first by price (in descending order), and then by name (in ascending order).
1.10 Select all the data from the products, including all the data for each product's manufacturer.
1.11 Select the product name, price, and manufacturer name of all the products.
1.12 Select the average price of each manufacturer's products, showing only the manufacturer's code.
1.13 Select the average price of each manufacturer's products, showing the manufacturer's name.
1.14 Select the names of manufacturer whose products have an average price larger than or equal to $150.
1.15 Select the name and price of the cheapest product.

SQL

1.16 Select the name of each manufacturer along with the name and price of its most expensive product.
1.17 Add a new product: Loudspeakers, $70, manufacturer 2.
1.18 Update the name of product 8 to "Laser Printer".
1.19 Apply a 10% discount to all products.
1.20 Apply a 10% discount to all products with a price larger than or equal to $120.

17.

TABLE CODE [NOT PROVIDE IN EXAM]
CREATE TABLE Departments (Code INTEGER PRIMARY KEY, Name varchar(255) NOT NULL , Budget decimal NOT NULL);
CREATE TABLE Employees (SSN INTEGER PRIMARY KEY, Name varchar(255) NOT NULL , LastName varchar(255) NOT NULL , Department INTEGER NOT NULL , foreign key (department) references Departments(Code));



1. Create Table and insert the values.

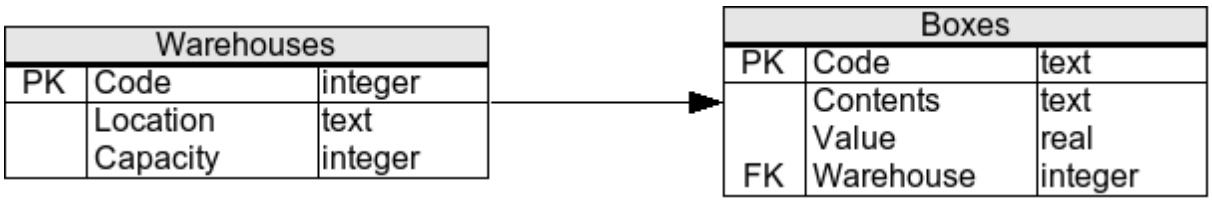
```
INSERT INTO Departments(Code,Name,Budget) VALUES(14,'IT',65000);
INSERT INTO Departments(Code,Name,Budget) VALUES(37,'Accounting',15000);
INSERT INTO Departments(Code,Name,Budget) VALUES(59,'Human Resources',240000);
INSERT INTO Departments(Code,Name,Budget) VALUES(77,'Research',55000);

INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('123234877','Michael','Rogers',14);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('152934485','Anand','Manikutty',14);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('222364883','Carol','Smith',37);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('326587417','Joe','Stevens',37);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('332154719','Mary-Anne','Foster',14);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('332569843','George','ODonnell',77);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('546523478','John','Doe',59);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('631231482','David','Smith',77);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('654873219','Zacary','Efron',59);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('745685214','Eric','Goldsmith',59);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('845657245','Elizabeth','Doe',14);
INSERT INTO Employees(SSN,Name,LastName,Department) VALUES('845657246','Kumar','Swamy',14);
```

SQL

2. Solve the SQL queries:
2.1 Select the last name of all employees.
2.2 Select the last name of all employees, without duplicates.
2.3 Select all the data of employees whose last name is "Smith".
2.4 Select all the data of employees whose last name is "Smith" or "Doe".
2.5 Select all the data of employees that work in department 14.
2.6 Select all the data of employees that work in department 37 or department 77.
2.7 Select all the data of employees whose last name begins with an "S".
2.8 Select the sum of all the departments' budgets.
2.9 Select the number of employees in each department (you only need to show the department code and the number of employees).
2.10 Select all the data of employees, including each employee's department's data.
2.11 Select the name and last name of each employee, along with the name and budget of the employee's department.
2.12 Select the name and last name of employees working for departments with a budget greater than $60,000.
2.13 Select the departments with a budget larger than the average budget of all the departments.
2.14 Select the names of departments with more than two employees.
2.15 Very Important - Select the name and last name of employees working for departments with second lowest budget.
2.16 Add a new department called "Quality Assurance", with a budget of $40,000 and departmental code 11. And Add an employee called "Mary Moore" in that department, with SSN 847-21-9811.
2.17 Reduce the budget of all departments by 10%.
2.18 Reassign all employees from the Research department (code 77) to the IT department (code 14).
2.19 Delete from the table all employees in the IT department (code 14).
2.20 Delete from the table all employees who work in departments with a budget greater than or equal to $60,000.
2.21 Delete from the table all employees.

18.



1. CREATE TABLE AND INSERT ALL VALUES.
2. TABLE: WAREHOUES VALUES INSERT CODE:
INSERT INTO Warehouses(Code,Location,Capacity) VALUES(1,'Chicago',3);

```
INSERT INTO Warehouses(Code,Location,Capacity) VALUES(2,'Chicago',4);
INSERT INTO Warehouses(Code,Location,Capacity) VALUES(3,'New York',7);
INSERT INTO Warehouses(Code,Location,Capacity) VALUES(4,'Los Angeles',2);
INSERT INTO Warehouses(Code,Location,Capacity) VALUES(5,'San Francisco',8);
```

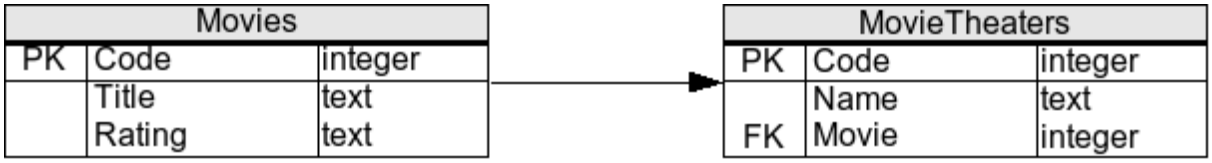
3. TABLE: BOXES VALUES INSERT CODE:

```
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('0MN7','Rocks',180,3);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('4H8P','Rocks',250,1);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('4RT3','Scissors',190,4);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('7G3H','Rocks',200,1);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('8JN6','Papers',75,1);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('8Y6U','Papers',50,3);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('9J6F','Papers',175,2);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('LL08','Rocks',140,4);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('P0H6','Scissors',125,1);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('P2T6','Scissors',150,2);
INSERT INTO Boxes(Code,Contents,Value,Warehouse) VALUES('TU55','Papers',90,5);
```

3.1 Select all warehouses.
3.2 Select all boxes with a value larger than $150.
3.3 Select all distinct contents in all the boxes.
3.4 Select the average value of all the boxes.
3.5 Select the warehouse code and the average value of the boxes in each warehouse.
3.6 Same as previous exercise, but select only those warehouses where the average value of the boxes is greater than 150.
3.7 Select the code of each box, along with the name of the city the box is located in.
3.8 Select the warehouse codes, along with the number of boxes in each warehouse.
Optionally, take into account that some warehouses are empty (i.e., the box count should show up as zero, instead of omitting the warehouse from the result).
3.9 Select the codes of all warehouses that are saturated (a warehouse is saturated if the number of boxes in it is larger than the warehouse's capacity).
3.10 Select the codes of all the boxes located in Chicago.
3.11 Create a new warehouse in New York with a capacity for 3 boxes.
3.12 Create a new box, with code "H5RT", containing "Papers" with a value of $200, and located in warehouse 2.
3.13 Reduce the value of all boxes by 15%.
3.14 Remove all boxes with a value lower than $100.
3.15 Add Index for column "Warehouse" in table "boxes"
3.16 Print all the existing indexes
3.17 Remove (drop) the index you added just

SQL

19. 1. Create table Movies and Movie Theaters using schema figure.



2. Insert values into table Movies

```
INSERT INTO Movies(Code,Title,Rating) VALUES(1,'Citizen Kane','PG');
INSERT INTO Movies(Code,Title,Rating) VALUES(2,'Singin" in the Rain','G');
INSERT INTO Movies(Code,Title,Rating) VALUES(3,'The Wizard of Oz','G');
INSERT INTO Movies(Code,Title,Rating) VALUES(4,'The Quiet Man',NULL);
INSERT INTO Movies(Code,Title,Rating) VALUES(5,'North by Northwest',NULL);
INSERT INTO Movies(Code,Title,Rating) VALUES(6,'The Last Tango in Paris','NC-17');
INSERT INTO Movies(Code,Title,Rating) VALUES(7,'Some Like it Hot','PG-13');
INSERT INTO Movies(Code,Title,Rating) VALUES(8,'A Night at the Opera',NULL);
```

3. Insert values into table Movie Theaters

```
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(1,'Odeon',5);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(2,'Imperial',1);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(3,'Majestic',NULL);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(4,'Royale',6);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(5,'Paraiso',3);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(6,'Nickelodeon',NULL);
```

4. Solve this Questions:

4.1 Select the title of all movies.

4.2 Show all the distinct ratings in the database.

4.3 Show all unrated movies.

4.4 Select all movie theaters that are not currently showing a movie.

4.5 Select all data from all movie theaters and, additionally, the data from the movie that is being shown in the theater (if one is being shown).

4.6 Select all data from all movies and, if that movie is being shown in a theater, show the data from the theater.

4.7 Show the titles of movies not currently being shown in any theaters.

4.8 Add the unrated movie "One, Two, Three".

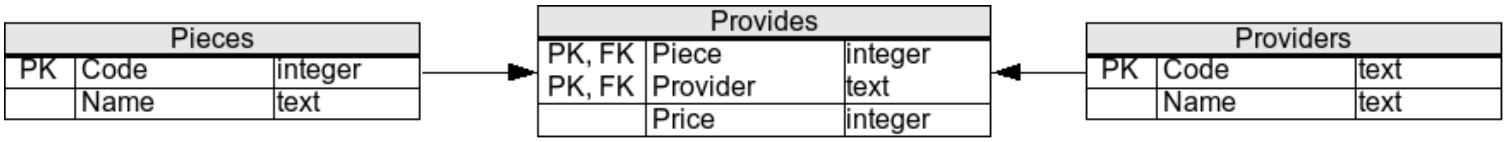
4.9 Set the rating of all unrated movies to "G".

4.10 Remove movie theaters projecting movies rated "NC-17".

[**Notes:** The MySQL IS NOT NULL condition is used to test for a NOT NULL value in a SELECT, INSERT, UPDATE, or DELETE statement. NULL is not a data type - this means it is not recognized as an "int", "date" or any other defined data type. Arithmetic operations involving NULL always return NULL for example, 69 + NULL = NULL. All aggregate functions affect only rows that do not have NULL values.]

SQL

20. 1. Create tables Pieces, Provides and Providers using schema figure



2. TABLE CODE [NOT PROVIDE IN EXAM]

CREATE TABLE Pieces (Code INTEGER PRIMARY KEY NOT NULL, Name TEXT NOT NULL);	CREATE TABLE Providers (Code VARCHAR(40) PRIMARY KEY NOT NULL, Name TEXT NOT NULL);
CREATE TABLE Provides (Piece INTEGER, FOREIGN KEY (Piece) REFERENCES Pieces(Code), Provider VARCHAR(40), FOREIGN KEY (Provider) REFERENCES Providers(Code), Price INTEGER NOT NULL, PRIMARY KEY(Piece, Provider));	**alternative one for SQLite** CREATE TABLE Provides (Piece INTEGER, Provider VARCHAR(40), Price INTEGER NOT NULL, PRIMARY KEY(Piece, Provider));

3. Insert values Providers and pieces tables:

```
INSERT INTO Providers(Code, Name) VALUES('HAL','Clarke Enterprises');
INSERT INTO Providers(Code, Name) VALUES('RBT','Susan Calvin Corp.');
INSERT INTO Providers(Code, Name) VALUES('TNBC','Skellington Supplies');
INSERT INTO Pieces(Code, Name) VALUES(1,'Sprocket');
INSERT INTO Pieces(Code, Name) VALUES(2,'Screw');
INSERT INTO Pieces(Code, Name) VALUES(3,'Nut');
INSERT INTO Pieces(Code, Name) VALUES(4,'Bolt');
```

4. Insert values Provides Table:

```
INSERT INTO Provides(Piece, Provider, Price) VALUES(1,'HAL',10);
INSERT INTO Provides(Piece, Provider, Price) VALUES(1,'RBT',15);
INSERT INTO Provides(Piece, Provider, Price) VALUES(2,'HAL',20);
INSERT INTO Provides(Piece, Provider, Price) VALUES(2,'RBT',15);
INSERT INTO Provides(Piece, Provider, Price) VALUES(2,'TNBC',14);
INSERT INTO Provides(Piece, Provider, Price) VALUES(3,'RBT',50);
INSERT INTO Provides(Piece, Provider, Price) VALUES(3,'TNBC',45);
INSERT INTO Provides(Piece, Provider, Price) VALUES(4,'HAL',5);
INSERT INTO Provides(Piece, Provider, Price) VALUES(4,'RBT',7);
```

5. Solve the questions:

5.1 Select the name of all the pieces.

5.2 Select all the providers' data.

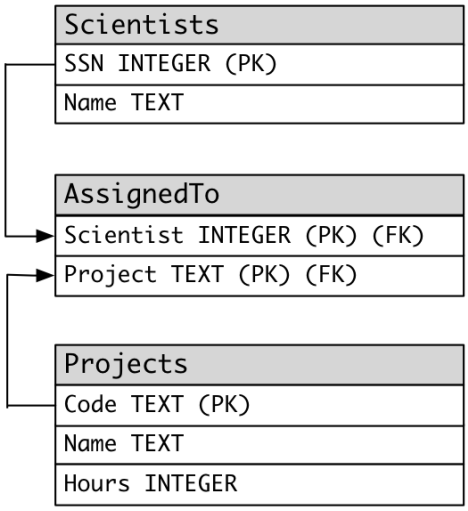
5.3 Obtain the average price of each piece (show only the piece code and the average price).

5.4 Obtain the names of all providers who supply piece 1.

5.5 Select the name of pieces provided by provider with code "HAL".
5.6 For each piece, find the most expensive offering of that piece and include the piece name, provider name, and price (note that there could be two providers who supply the same piece at the most expensive price).
5.7 Add an entry to the database to indicate that "Skellington Supplies" (code "TNBC") will provide sprockets (code "1") for 7 cents each.
5.8 Increase all prices by one cent.
5.9 Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply bolts (code 4).
5.10 Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply any pieces (the provider should still remain in the database).

21. 1. Create tables using schema figure.

```
INSERT INTO Scientists(SSN,Name)
  VALUES(123234877,'Michael Rogers'),
    (152934485,'Anand Manikutty'),
    (222364883, 'Carol Smith'),
    (326587417,'Joe Stevens'),
    (332154719,'Mary-Anne Foster'),
    (332569843,'George ODonnell'),
    (546523478,'John Doe'),
    (631231482,'David Smith'),
    (654873219,'Zacary Efron'),
    (745685214,'Eric Goldsmith'),
    (845657245,'Elizabeth Doe'),
    (845657246,'Kumar Swamy');
```



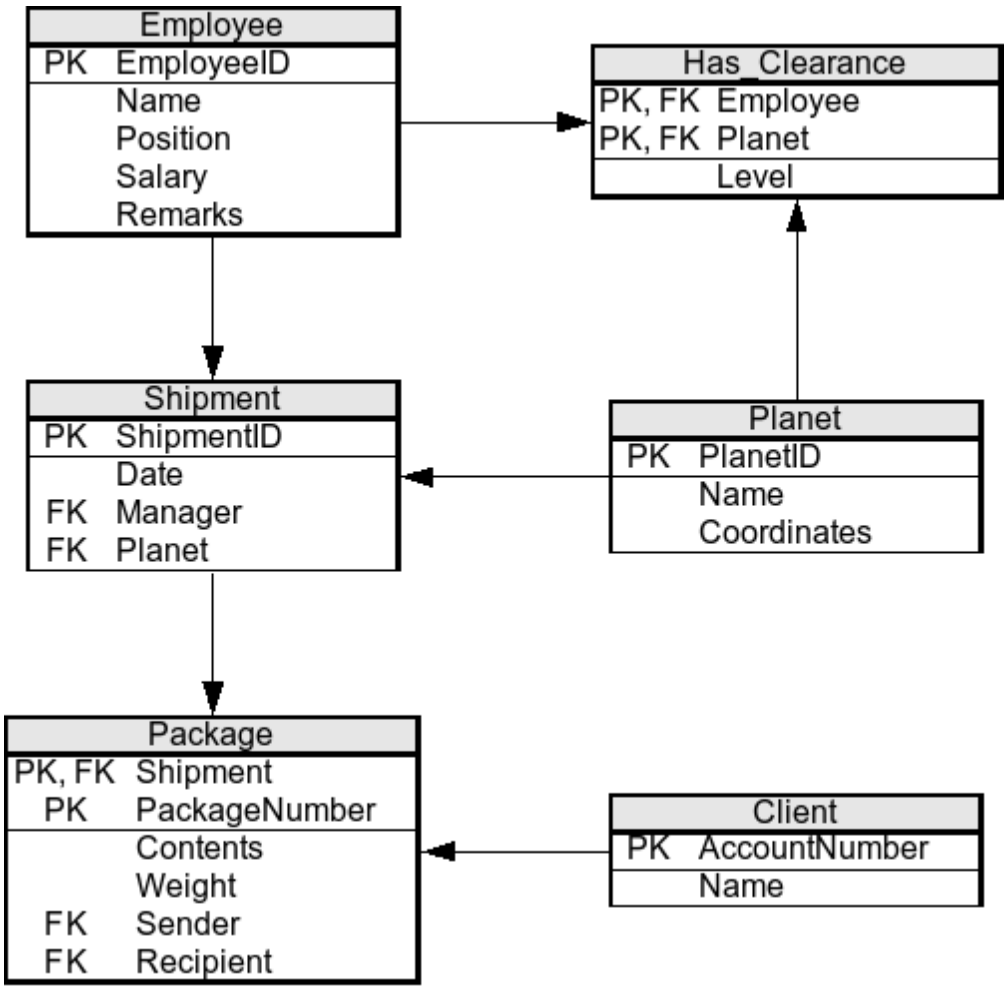

```
INSERT INTO Projects ( Code,Name,Hours)
VALUES ('AeH1','Winds: Studying Bernoullis
Principle', 156),
('AeH2','Aerodynamics and Bridge Design',189),
('AeH3','Aerodynamics and Gas Mileage', 256),
('AeH4','Aerodynamics and Ice Hockey', 789),
('AeH5','Aerodynamics of a Football', 98),
('AeH6','Aerodynamics of Air Hockey',89),
('Ast1','A Matter of Time',112),
('Ast2','A Puzzling Parallax', 299),
('Ast3','Build Your Own Telescope', 6546),
('Bte1','Juicy: Extracting Apple Juice with Pectinase',
321),
('Bte2','A Magnetic Primer Designer', 9684),
('Bte3','Bacterial Transformation Efficiency', 321),
('Che1','A Silver-Cleaning Battery', 545),
('Che2','A Soluble Separation Solution', 778);
```

```
INSERT INTO AssignedTo ( Scientist, Project)
VALUES (123234877,'AeH1'),
(152934485,'AeH3'),
(222364883,'Ast3'),
(326587417,'Ast3'),
(332154719,'Bte1'),
(546523478,'Che1'),
(631231482,'Ast3'),
(654873219,'Che1'),
(745685214,'AeH3'),
(845657245,'Ast1'),
(845657246,'Ast2'),
(332569843,'AeH4');
```

SQL

21.1. List all the scientists' names, their projects' names, and the hours worked by that scientist on each project, in alphabetical order of project name, then scientist name.
21.2 Select the project names which are not assigned yet

22. 1. Create tables using schema figure.



INSERT INTO Client VALUES(1, 'Zapp Brannigan');
INSERT INTO Client VALUES(2, "Al Gore's Head");
INSERT INTO Client VALUES(3, 'Barbados Slim');
INSERT INTO Client VALUES(4, 'Ogden Wernstrom');
INSERT INTO Client VALUES(5, 'Leo Wong');
INSERT INTO Client VALUES(6, 'Lrrr');
INSERT INTO Client VALUES(7, 'John Zoidberg');
INSERT INTO Client VALUES(8, 'John Zoidfarb');
INSERT INTO Client VALUES(9, 'Morbo');
INSERT INTO Client VALUES(10, 'Judge John Whitey');
INSERT INTO Client VALUES(11, 'Calculon');

```
INSERT INTO Employee VALUES(1, 'Phillip J. Fry', 'Delivery boy', 7500.0, 'Not to be confused with the Philip J. Fry from Hovering Squid World 97a');
INSERT INTO Employee VALUES(2, 'Turanga Leela', 'Captain', 10000.0, NULL);
INSERT INTO Employee VALUES(3, 'Bender Bending Rodriguez', 'Robot', 7500.0, NULL);
INSERT INTO Employee VALUES(4, 'Hubert J. Farnsworth', 'CEO', 20000.0, NULL);
INSERT INTO Employee VALUES(5, 'John A. Zoidberg', 'Physician', 25.0, NULL);
INSERT INTO Employee VALUES(6, 'Amy Wong', 'Intern', 5000.0, NULL);
INSERT INTO Employee VALUES(7, 'Hermes Conrad', 'Bureaucrat', 10000.0, NULL);
INSERT INTO Employee VALUES(8, 'Scruffy Scruffington', 'Janitor', 5000.0, NULL);
INSERT INTO Planet VALUES(1, 'Omicron Persei 8', 89475345.3545);
INSERT INTO Planet VALUES(2, 'Decapod X', 65498463216.3466);
INSERT INTO Planet VALUES(3, 'Mars', 32435021.65468);
INSERT INTO Planet VALUES(4, 'Omega III', 98432121.5464);
INSERT INTO Planet VALUES(5, 'Tarantulon VI', 849842198.354654);
INSERT INTO Planet VALUES(6, 'Cannibalon', 654321987.21654);
INSERT INTO Planet VALUES(7, 'DogDoo VII', 65498721354.688);
INSERT INTO Planet VALUES(8, 'Nintenduu 64', 6543219894.1654);
INSERT INTO Planet VALUES(9, 'Amazonia', 65432135979.6547);
INSERT INTO Has_Clearance VALUES(1, 1, 2);
INSERT INTO Has_Clearance VALUES(1, 2, 3);
INSERT INTO Has_Clearance VALUES(2, 3, 2);
INSERT INTO Has_Clearance VALUES(2, 4, 4);
INSERT INTO Has_Clearance VALUES(3, 5, 2);
INSERT INTO Has_Clearance VALUES(3, 6, 4);
INSERT INTO Has_Clearance VALUES(4, 7, 1);
INSERT INTO Shipment VALUES(1, '3004/05/11', 1, 1);
INSERT INTO Shipment VALUES(2, '3004/05/11', 1, 2);
INSERT INTO Shipment VALUES(3, NULL, 2, 3);
INSERT INTO Shipment VALUES(4, NULL, 2, 4);
INSERT INTO Shipment VALUES(5, NULL, 7, 5);
INSERT INTO Package VALUES(1, 1, 'Undeclared', 1.5, 1, 2);
INSERT INTO Package VALUES(2, 1, 'Undeclared', 10.0, 2, 3);
INSERT INTO Package VALUES(2, 2, 'A bucket of krill', 2.0, 8, 7);
INSERT INTO Package VALUES(3, 1, 'Undeclared', 15.0, 3, 4);
INSERT INTO Package VALUES(3, 2, 'Undeclared', 3.0, 5, 1);
INSERT INTO Package VALUES(3, 3, 'Undeclared', 7.0, 2, 3);
INSERT INTO Package VALUES(4, 1, 'Undeclared', 5.0, 4, 5);
INSERT INTO Package VALUES(4, 2, 'Undeclared', 27.0, 1, 2);
INSERT INTO Package VALUES(5, 1, 'Undeclared', 100.0, 5, 1);
```

22.1 Who received a 1.5kg package? The result is "Al Gore's Head".
22..2 What is the total weight of all the packages that he sent?