1. Database Creation :

CREATE DATABASEvalvee;

1. Database Delete :

DROPDATABASE valvee;

1. Customer s Table creation :

CREATE TABLE customers(

id INT NOT NULL AUTO\_INCREMENT,

firstName VARCHAR(255),

lastName VARCHAR(255),

address VARCHAR(255),

email VARCHAR(255),

ity VARCHAR(255),

division VARCHAR(255),

zipcode VARCHAR(255),

PRIMARY KEY (id)

);

1. Data Insertion in Customer Table :

INSERT INTO customers(firstName,lastName,address,email,city,division,zipcode)VALUES(

'Akbor','Hosen','House 2 lane 9','akbor@gmail.com','Comilla','Chittagong','123'),

('Faisal','Khan','House 3 lane 11','khan@gmail.com','Gazipur','Dhaka','124'),

('Jahangir','Alom','House 21 lane 4','alom@gmail.com','Habigang','Sylhet','190'),

('Kamal','Ahmed','House 6 lane 7','kamal@gmail.com','Natore','Rajshahi','134'),

('Abdul','Bari','House 2 lane 8','md@gmail.com','Jessor','Khulna','100')

1. Data update in customers table :

UPDATE customers

SET email='akbor@yahoo.com'

WHERE id=2;

1. Data deletion from customer table with condition :

DELETE FROM customers

WHERE id=2;

1. Alter customers table :

ALTER TABLE customers ADD newCol VARCHAR(255);

ALTER TABLE customers

MODIFYCOLUMN newCol INT (11);

ALTER TABLE customers

DROP COLUMN newCol;

1. Select query with space holder :

SELECT \* FROM customers;

SELECT \*FROM customers LIMIT 3;

SELECT \* FROM customers LIMIT 3,4;

SELECT firstName,lastName FROM customers ;

SELECT \* FROM ‘customers’ WHERE id=2;

SELECT \* FROM ‘customers’ ORDER BY lastName;

SELECT \* FROM ‘customers’ ORDER BY lastName DESC;

SELECT \* FROM ‘customers’ ORDER BY lastName ASC;

1. Select Distinct item :

SELECT city FROM ‘customers’ ;

SELECT DISTINCT city FROM ‘customers’ ;

1. Between operator :

SELECT \* FROM ‘customers’ WHERE age

BETWEEN 20 AND 40;

1. Like operator :

SELECT \* FROM ‘customers’ WHERE city

LIKE '%e';

SELECT \* FROM ‘customers’ WHERE city

LIKE 'n%';

SELECT \* FROM ‘customers’ WHERE city

LIKE '%tor%';

SELECT \* FROM ‘customers’ WHERE firstName LIKE '%Ak%';

SELECT \* FROM ‘customers’ WHERE firstName NOT LIKE '%Ak%';

1. In operator :

SELECT \* FROM ‘customers’ WHERE division

IN('Dhaka','Rajshahi');

SELECT \* FROM ‘customers’ WHERE division

IN('Dhaka','Rajshahi','Sylhet');

1. Index creation :

CREATE INDEX Cindex

ON customers(city);

SELECT city FROM ‘customers’;

DROP INDEX Cindex ON customers;

UNIVERSITY DATABASE

14. CREATE DATABASE university;

\*\*\*TABLE CREATION\*\*\*

(15 to 25)

15. create table classroom

(building varchar (15),

room\_numbervarchar (7),

capacity numeric(4,0),

primary key (building, room\_number));16. create table department(dept\_name varchar (20),

building varchar (15),

budget numeric (12,2) ,

primary key (dept\_name));

17. create table course

(course\_idvarchar (8),

title varchar (50),

dept\_namevarchar (20),

credits numeric (2,0) ,

primary key (course\_id),

foreign key (dept\_name) references department(dept\_name));

18. create table instructor

(ID varchar (5),

name varchar (20) not null,

dept\_namevarchar (20),

salary numeric (8,2),

primary key (ID),

foreign key (dept\_name) references department(dept\_name));

19. create table student

(ID varchar (5),

name varchar (20) not null,

dept\_namevarchar (20),

tot\_cred numeric (3,0),

primary key (ID),

foreign key (dept\_name) references department(dept\_name));

20. create table section

(course\_idvarchar (8),

sec\_idvarchar (8),

semester varchar (6),

year numeric (4,0),

building varchar (15),

room\_numbervarchar (7),

time\_slot\_idvarchar (4),

primary key (course\_id, sec\_id, semester, year),

foreign key (course\_id) references course(course\_id),

foreign key (building, room\_number) references classroom(building,room\_number));

21. create table teaches

(ID varchar (5),

course\_idvarchar (8),

sec\_idvarchar (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(course\_id, sec\_id, semester, year),

foreign key (ID) references instructor(ID));

22. create table takes

(ID varchar (5),

course\_idvarchar (8),

sec\_idvarchar (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(course\_id, sec\_id, semester, year),

foreign key (ID) references student(ID));

23. create table advisor

(s\_IDvarchar (5),

i\_IDvarchar (5),

primary key (s\_ID),

foreign key (i\_ID) references instructor (ID),

foreign key (s\_ID) references student (ID));

24. create table prereq

(course\_idvarchar(8),

prereq\_idvarchar(8),

primary key (course\_id, prereq\_id),

foreign key (course\_id) references course(course\_id),

foreign key (prereq\_id) references course(course\_id));

25. create table time\_slot

(time\_slot\_idvarchar (4),

day varchar(1) ,

start\_time time,

end\_time time,

primary key (time\_slot\_id, day, start\_time));

\*\*\*DATA INSERTION\*\*\*

(26 to 36)

26. INSERT INTO classroom(building,room\_number,capacity)VALUES(

‘Packard’,’ 101’,’ 500’),

(‘Painter’,’ 514’,’ 10’),

(‘Taylor’,’ 3128’,’ 70’),

(‘Watson’,’ 100’,’ 30’),

(‘Watson’,’ 120’,’ 50’);

27. INSERT INTO department(`dept\_name`,`building`,`budget`)VALUES(

'Biology',' Watson',' 90000'),

('Comp. Sci.',' Taylor',' 100000'),

('Elec. Eng.',' Taylor',' 85000'),

('Finance',' Painter',' 120000'),

('History',' Painter',' 50000'),

('Music',' Packard',' 80000'),

('Physics',' Watson',' 70000'

);

28. INSERT INTO course(course\_id,title,dept\_name,credits)VALUES(

'BIO-101',' Intro. to Biology',' Biology',' 4'),

('BIO-301',' Genetics',' Biology',' 4'),

('BIO-399',' Computational Biology',' Biology',' 3'),

('CS-101',' Intro. to Computer Science',' Comp. Sci.',' 4'),

('CS-190',' Game Design',' Comp. Sci.',' 4'),

('CS-315',' Robotics ','Comp. Sci.',' 3'),

('CS-319',' Image Processing',' Comp. Sci.',' 3'),

('CS-347',' Database System Concepts',' Comp. Sci.',' 3'),

('EE-181',' Intro. to Digital Systems',' Elec. Eng.',' 3'),

('FIN-201',' Investment Banking',' Finance',' 3'),

('HIS-351',' World History',' History',' 3'),

('MU-199',' Music Video Production',' Music',' 3'),

('PHY-101',' Physical Principles',' Physics',' 4');

29. INSERT INTO instructor(ID,name,dept\_name,salary)VALUES(

'10101',' Srinivasan',' Comp. Sci.',' 65000'),

('12121',' Wu',' Finance',' 90000'),

('15151',' Mozart',' Music',' 40000'),

('22222',' Einstein',' Physics',' 95000'),

('32343',' El Said',' History',' 60000'),

('33456',' Gold',' Physics',' 87000'),

('45565',' Katz',' Comp. Sci.',' 75000'),

('58583',' Califieri',' History',' 62000'),

('76543',' Singh',' Finance',' 80000'),

('76766',' Crick',' Biology',' 72000'),

('83821',' Brandt',' Comp. Sci.',' 92000'),

('98345',' Kim',' Elec. Eng.',' 80000');

30. INSERT INTO student(ID,name,dept\_name,tot\_cred)VALUES

('00128',' Zhang',' Comp. Sci.',' 102'),

('12345',' Shankar',' Comp. Sci.',' 32'),

('19991',' Brandt',' History',' 80'),

('23121',' Chavez',' Finance',' 110'),

('44553',' Peltier',' Physics',' 56'),

('45678',' Levy',' Physics',' 46'),

('54321',' Williams',' Comp. Sci.',' 54'),

('55739',' Sanchez',' Music',' 38'),

('70557',' Snow',' Physics',' 0'),

('76543',' Brown',' Comp. Sci.',' 58'),

('76653',' Aoi',' Elec. Eng.',' 60'),

('98765',' Bourikas',' Elec. Eng.',' 98'),

('98988',' Tanaka',' Biology',' 120');

31. INSERT INTO takes(`ID`,`course\_id`,`sec\_id`,`semester`,`year`,`grade`)VALUES(

'00128',' CS-101',' 1',' Fall',' 2009',' A'),('

00128',' CS-347',' 1',' Fall',' 2009',' A-'),

('12345',' CS-101',' 1',' Fall',' 2009',' C'),

('12345',' CS-190',' 2',' Spring',' 2009',' A'),

('12345',' CS-315',' 1',' Spring',' 2010',' A'),

('12345',' CS-347',' 1',' Fall',' 2009',' A'),

('19991',' HIS-351',' 1',' Spring',' 2010',' B'),

('23121',' FIN-201',' 1','Spring',' 2010',' C+'),

('44553',' PHY-101',' 1 ','Fall',' 2009',' B-'),

('45678',' CS-101',' 1',' Fall',' 2009',' F'),

('45678',' CS-101',' 1',' Spring',' 2010',' B+'),

('45678',' CS-319',' 1',' Spring',' 2010',' B'),

('54321',' CS-101',' 1 ','Fall',' 2009',' A-'),

('54321',' CS-190',' 2',' Spring',' 2009',' B+'),

('55739',' MU-199',' 1',' Spring',' 2010',' A-'),

('76543',' CS-101',' 1',' Fall',' 2009',' A'),

('76543',' CS-319',' 2',' Spring',' 2010',' A'),

('76653',' EE-181',' 1',' Spring',' 2009',' C'),

('98765',' CS-101',' 1',' Fall',' 2009',' C-'),

('98765',' CS-315',' 1',' Spring',' 2010',' B'),

('98988',' BIO-101',' 1',' Summer',' 2009',' A'),

('98988',' BIO-301',' 1',' Summer',' 2010',' null');

32. INSERT INTO section(`course\_id`,`sec\_id`,`semester`,`year`,`building`,`room\_number`,`time\_slot\_id`)VALUES

('BIO-101',' 1',' Summer',' 2009',' Painter',' 514',' B'),

('BIO-301',' 1',' Summer',' 2010',' Painter',' 514',' A'),

('CS-101',' 1',' Fall',' 2009',' Packard',' 101',' H'),

('CS-101',' 1',' Spring',' 2010',' Packard',' 101',' F'),

('CS-190',' 1',' Spring',' 2009',' Taylor',' 3128',' E'),

('CS-190',' 2',' Spring',' 2009',' Taylor',' 3128',' A'),

('CS-315',' 1',' Spring',' 2010',' Watson',' 120',' D'),

('CS-319','1',' Spring',' 2010',' Watson',' 100',' B'),

('CS-319',' 2',' Spring',' 2010',' Taylor','3128',' C'),

('CS-347',' 1',' Fall',' 2009',' Taylor',' 3128',' A'),

('EE-181',' 1',' Spring ','2009',' Taylor',' 3128',' C'),

('FIN-201',' 1 ','Spring',' 2010',' Packard',' 101',' B'),

('HIS-351',' 1',' Spring',' 2010',' Painter',' 514',' C'),

('MU-199',' 1',' Spring',' 2010',' Packard',' 101',' D'),

('PHY-101',' 1',' Fall ','2009',' Watson',' 100',' A');

33. INSERT INTO prereq(`course\_id`,`prereq\_id`)VALUES(

'BIO-301',' BIO-101'),

('BIO-399',' BIO-101'),

('CS-190',' CS-101'),

('CS-315',' CS-101'),

('CS-319',' CS-101'),

('CS-347',' CS-101'),

('EE-181',' PHY-101');

34. INSERT INTO advisor(s\_id,i\_id)VALUES(

‘00128’,’ 45565’),

(‘12345’,’ 10101’),

(‘23121’,’ 76543’),

(‘44553’,’ 22222’),

(‘45678’,’ 22222’),

(‘76543’,’ 45565’),

(‘76653’,’ 98345’),

(‘98765’,’ 98345’),

(‘98988’,’ 76766’);

35. INSERT INTO teaches(`ID`,`course\_id`,`sec\_id`,`semester`,`year`)VALUES

('10101',' CS-101',' 1',' Fall',' 2009'),

('10101',' CS-315',' 1',' Spring',' 2010'),

('10101',' CS-347',' 1',' Fall',' 2009'),

('12121',' FIN-201',' 1',' Spring',' 2010'),

('15151',' MU-199',' 1',' Spring',' 2010'),

('22222',' PHY-101',' 1',' Fall',' 2009'),

('32343',' HIS-351',' 1',' Spring',' 2010'),

('45565',' CS-101',' 1',' Spring',' 2010'),

('45565',' CS-319',' 1',' Spring',' 2010'),

('76766',' BIO-101',' 1',' Summer',' 2009'),

('76766',' BIO-301',' 1',' Summer ','2010'),

('83821',' CS-190',' 1',' Spring',' 2009'),

('83821',' CS-190','2','Spring',' 2009'),

('83821',' CS-319',' 2',' Spring',' 2010'),

('98345',' EE-181',' 1',' Spring',' 2009');

36. INSERT INTO timeslot(time\_slot\_id,day,start\_time,end\_time)VALUES

('A',' M ','8:00',' 8:50'),

('B',' M',' 9:00',' 9:50'),

('C',' M',' 11:00',' 11:50'),

('D',' M',' 13:00',' 13:50'),

('E',' T',' 10:30',' 11:45'),

('F',' R',' 14:30',' 15:45'),

('G',' F',' 16:00',' 16:50'),

('H','W',' 10:00',' 12:30');

37. The select clause may also contain arithmetic expressions involving the operators +,−,∗, and / operating on constants or attributes of tuples. For example, the query:

Select ID, name,salary\*1.1

From instructor;

38. Find the names of all instructors in the Computer Science department who have salary greater than $70,000 :

select name

from instructor

wheredept\_name = 'Comp.Sci' and salary > 70000;

39. The above querycan be written in SQL as :

select name, instructor.dept\_name, buliding

from instructor, department

where instructor.dept\_name = department.dept\_name;

40. The following SQL query ensures this condition, and outputs the instructor name and course identiﬁers from such matching tuples :

Select *name*, *course\_id*

From *instructor*, *teaches*

Where *instructor*.*ID*= *teaches*.*ID*;

41. If we only wished toﬁnd instructornames and course identiﬁersfor instructorsintheComputerSciencedepartment,wecouldaddanextrapredicatetothe where clause, as shown below :

Select *name*, *course\_id*

From *instructor*, *teaches*

where *instructor*.*ID*= *teaches*.*ID* and *instructor*.*dept\_name*= ‘Comp. Sci.’;

42. For all instructors in the university who have taught some course, ﬁnd their names and the course ID of all courses they taught :

Select *name*, *course\_id*

From *instructor* natural join *teaches*;

43. Listthenamesofinstructors along with the the titles of courses that they teach :

select name, title

from instructor natural join teaches, course

where teaches.course\_id= course.course\_id;

44. Natural join of three tables :

select*name*, *title*

from*instructor* natural join *teaches* natural join *course*;

45. ,If we want the attribute name to be replaced with the name instructor name, we can rewrite the preceding query as :

select T.name, S.course\_id

from instructor as T, teaches as S

where T.ID= S.ID;

46. Find the names of allinstructors whose salary is greaterthan at leastone instructor in the Biology department :

select distinct *T*.*name*

from *instructor* as *T*, *instructor* as *S*

where *T.salary>S.salary* and *S.dept name* = ‘Biology’;

47. Find the names of all departments whose building name includes the substring ‘Watson’ :

Select dept\_name

from department

where building like '%Watson%';

48. The asterisk symbol “\*” can be used in the select clause to denote “allattributes.” Thus, the use of instructor.\* in the select clause of the query :

Select *instructor*.\*

From *instructor*, *teaches*

Where *instructor*.*ID*= *teaches*.*ID*;

49. Ordering the Display of Tuples :

select name

from instructor

where dept\_name = 'Physics'

order by name;

50. we wish to list the entire instructor relation in descending order of salary. If several instructors have the same salary ,we order them in ascending order by name. We express this query in SQL as follows :

Select \*

From *instructor*

order by *salary* desc, *name* asc;

51. Find the names of instructors with salary amounts between $90,000 and $100,000, we can use the between comparison to write:

Select *name*

From *instructor*

Where *salary* between 90000 and 100000;

52. Find the names of instructors with salary amounts between $90,000 and $100,000, we can use the between comparison to write:

Select *name*

From *instructor*

Where *salary <*= 100000 and *salary >*= 90000;

53. Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course :

select name, course\_id

from instructor, teaches

where instructor.ID= teaches.ID and dept\_name = 'Biology';

54. Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course :

select name, course\_id

from instructor, teaches

where (instructor.ID, dept\_name) = (teaches.ID, 'Biology');

55. The set of all courses taught in the Fall 2009 semester :

Select course\_id

from section

where semester = 'Fall' and year= 2009;

56. The set of all courses taught in the Spring 2010 semester :

Select course\_id

from section

where semester = 'Spring' and year= 2010;

57. To ﬁnd the set of all courses taught either in Fall 2009 or in Spring 2010, or both, we write :

(select course\_id

from section

where semester = 'Fall' and year= 2009)

union

(select course\_id

from section

where semester = 'Spring' and year= 2010);

\*\*Intersection\*\*

IT IS NOT SUPPORTED

(select course\_id

from section

where semester = 'Fall' and year= 2009)

intersect

(select course\_id

from section

where semester = 'Spring' and year= 2010);

QUERY FOR SAME OUTPUT :

Select course\_id

From section

Where semester=’Fall’ and year=2009

And course\_id in(

Select course\_id

From section

Where semester=’Spring’ and year=2010

);

58. To ﬁnd all courses taught in the Fall 2009 semester but not in the Spring 2010 semester, we write :

IT IS NOT SUPPORTED

(select course\_id

from section

where semester = 'Fall' and year= 2009)

except

(select course\_id

from section

where semester = 'Spring' and year= 2010);

QUERY FOR SAME OUTPUT :

Select course\_id

From section

Where semester=’Fall’ and year=2009

And course\_id not in(

Select course\_id

From section

Where semester=’Spring’ and year=2010

);

59. To ﬁnd all instructors who appear in the instructor relation with null values for salary, we write :

select name

from instructor

where salary is null;

60. “Find the average salary of instructors in the Computer Science department.” We write this query as follows :

Select *dept \_name*, avg(*salary*) as *avg salary*

From *instructor*

group by *dept\_name*;

61. we can give a meaningful name to the attribute by using the as clause as follows:

Select avg(*salary*)

From *instructor*;

where dept name=’Comp. Sci.’;

62. Find the number of instructors in each department who teach a course in the Spring 2010 semester. :

Select dept\_name, count(distinct ID) as instr\_count

from instructor natural join teaches

where semester = 'Spring' and year = 2010

group by dept\_name;

63. /\* erroneous query\*/

Select dept\_name, ID, avg(salary)

from instructor

group by dept\_name;

64. ,we might be interested in only those departments where the average salary of the instructors is more than $42,000 :

Select *dept name*, avg(*salary*) as *avg salary*

From *instructor*

group by *dept\_name*

having avg(*salary*) *>*42000;

65. ﬁnd those courses that were taught in the Fall 2009 and that appear in the set of courses obtained in the subquery :

select distinct *course\_id*

from *section*

where *semester* = ’Fall’ and *year*= 2009 and

*course\_id* in (select *course\_id*

from *section*

where *semester* = ’Spring’ and *year*= 2010);

66. To ﬁnd all the courses taught in the Fall 2009 semester but not in the Spring 2010 semester, we can write :

select distinct *course\_id*

from *section*

where *semester* = ’Fall’ and *year*= 2009 and

*course\_id* not in (select *course\_id*

from *section*

where *semester* = ’Spring’ and *year*= 2010);

67. The in and not in operators can also be used on enumerated sets .The following query selects the names of instructors whose names are neither“Mozart” nor “Einstein” :

select distinct *name*

from *instructor*

where *name* not in (’Mozart’, ’Einstein’);

68. “ﬁnd the total number of(distinct) students who have taken course sections taught by the instructor with ID 110011” :

select count(distinct ID)

from takes

where (course\_id, sec\_id, semester, year) in (select course\_id, sec\_id, semester, year

from teaches

where teaches.ID= 10101);

69. Find the names of all instructors whose salary is greater than at least one instructor in the Biology department :

select distinct T.name

from instructor as T, instructor as S

where T.salary>S.salary and S.dept\_name = 'Biology';

70. an alternative style for writing the preceding query :

select name

from instructor

where salary > some (select salary

from instructor

where dept\_name = 'Biology');

71. SQL also allows < some, <= some, >= some, = some, and<> some comparisons :

select name

from instructor

where salary > all (select salary

from instructor

where dept\_name = 'Biology');

72. “Find the departments that have the highest average salary :

Select dept\_name

from instructor

group by dept\_name

having avg (salary) >= all (select avg (salary)

from instructor

group by dept\_name);

73. Use of Exist :

Select course\_id

from section as S

where semester = 'Fall' and year= 2009 and

exists (select \*

from section as T

where semester = 'Spring' and year= 2010 and

S.course\_id= T.course\_id);

SOME QUERIES FROM VIDEO TUTORIAL

74. SELECT customers.firstName,customers.lastName,orders.orderNumber

FROM customers

INNER JOIN orders

ON customers.id=orders.customerId

ORDER BY customers.lastName;

75. SELECT customers.firstName,customers.lastName,orders.orderNumber,orders.orderDate

FROM customers

LEFT JOIN orders

ON customers.id=orders.customerId

ORDER BY customers.lastName;

76. SELECT orders.orderNumber,customers.firstName,customers.lastName

FROM orders

RIGHT JOIN customers

ON orders.customerId=customers.id

ORDER BY customers.lastName;

77. CREATE TABLE suppliers(

id INT NOT NULL AUTO\_INCREMENT,

firstName VARCHAR(255),

lastName VARCHAR(255),

address VARCHAR(255),

email VARCHAR(255),

city VARCHAR(255),

division VARCHAR(255),

zipcode VARCHAR(255),

PRIMARY KEY(id));

78. INSERT INTO suppliers(firstName,lastName,address,email,city,division,zipcode)VALUES(

'Al','Amin','House 7 lane 7','amin@gmail.com','Comilla','Chittagong','167'),

('Jabed','Hasan','House 12 lane 1','hasan@gmail.com','Gazipur','Dhaka','342'),

('Rashed','Korim','House 3 lane 4','karim@gmail.com','Dhaka','Dhaka','876'),

('Kobir ','Alom','House 9 lane 6','alom@gmail.com','Kurigram','Rangpur','656'),

('Mizan','Khan','House 3 lane 7','khan1@gmail.com','Bogra','Rajshahi','845');

79. SELECT city FROM customers

UNION

SELECT city FROM suppliers

ORDER BY city;

80. SELECT city FROM customers

UNION ALL

SELECT city FROM suppliers

ORDER BY city;

81. SELECT city,division FROM customers WHERE city='Dhaka'

UNION ALL

SELECT city ,division FROM suppliers

WHERE city='Dhaka'

ORDER BY city;

82. SELECT orders.orderNumber,customers.firstName,customers.lastName,products.name

FROM orders

INNER JOIN products

ON orders.productId=products.id

INNER JOIN customers

ON orders.customerId=customers.id

ORDER BY orders.orderNumber;

83. SELECT \* FROM customers

WHERE id IN(SELECT id

FROM customers

WHERE age>30

);

84. CREATE TABLE customers\_bup(

id INT NOT NULL AUTO\_INCREMENT,

firstName VARCHAR(255),

lastName VARCHAR(255),

address VARCHAR(255),

email VARCHAR(255),

city VARCHAR(255),

division VARCHAR(255),

zipcode VARCHAR(255),

age VARCHAR(255),

PRIMARY KEY (id)

);

85. INSERT INTO customers\_bup

SELECT\* FROM customers

WHERE id IN(

SELECT id

FROM customers

);

86. UPDATE customers

set salary =salary \*0.25

WHERE age IN (SELECT age

FROM customers\_bup

WHERE age>=30);

87. DELETE FROM customers\_bup

WHERE age IN (SELECT age

FROM customers

WHERE age=29

);

88. TRUNCATE TABLE suppliers;

89. SELECT firstName AS 'First Name',lastName AS 'Last Name' FROM customers;

SELECT CONCAT (firstName,'',lastName)AS 'Name',email,address,city FROM customers;

SELECT CONCAT (firstName,’’,lastName)AS’Name’,CONCAT(address,’’,city)AS ‘Address’ FROM customers;

90. SELECT o.id,o.orderDate,c.firstName,c.lastName

FROM customers AS c,orders AS o;

91. SELECT AVG(age)FROM customers;

SELECT COUNT(age)FROM customers;

SELECT MAX(age)FROM customers;

SELECT MIN(age)FROM customers;

SELECT SUM(age)FROM customers;

92. SELECT age,COUNT(age)

FROM customers

WHERE age>30

GROUP BY age;

93. SELECT age,COUNT(age)

FROM customers

GROUP BY age

HAVING COUNT(age)>=20;

94. CREATE VIEW customers\_view AS

SELECT id,firstName,lastName

FROM customers

WHERE firstNameIS NOT NULL;

SELECT \* FROM customers\_view;

95. INSERT INTO customers\_view VALUES (11,'Helal','Khan');

96. CREATE VIEW customers\_view\_newAS SELECT id,firstName,lastName

FROM customers

WHERE firstName IS NOT NULL

WITH CHECK OPTION;

INSERT INTO customers\_view\_newVALUES(13,NULL,''Amin);

97. UPDATE customers\_view

SET firstName ='Kamal'

where id=11;

DELETE FROMcustomers\_view

WHERE id=13;

DROP VIEW customers\_view\_new;

98. SELECT UCASE (firstName)FROM customers;

SELECT UCASE (firstName),lastName

FROM customers;

SELECT LCASE (firstName),UCASE(lastName),address FROM customers;

99. SELECT MID(city,1,3)AS ShortCity FROM customers;

SELECT firstName ,LENGTH(address)AS LengthAddress FROM customers;

SELECT firstName ,ROUND(salary,0) FROM customers;

SELECT firstName ,ROUND(salary,0)AS ROUNDSalary FROM customers;

100. SELECT name,price,NOW()AS Data FROM products;