Q1

Solution

CREATE TABLE employees (

employee\_id int PRIMARY KEY NOT NULL,

first\_name VARCHAR (25) NOT NULL,

last\_name VARCHAR (25) NOT NULL ,

email VARCHAR (25) NOT NULL ,

phone\_number VARCHAR (20) NOT NULL ,

hire\_date DATE NOT NULL,

job\_id VARCHAR (10),

salary DECIMAL (8, 2) NOT NULL ,

commission\_pct INT ,

manager\_id INT ,

department\_id INT

);

Footer

show columns from employees;

Q2

Solution

CREATE TABLE facilities (

facid int primary key NOT NULL ,

name varchar(100) NOT NULL,

membercost int NOT NULL,

guestcost int NOT NULL,

initialoutlay int,

monthlymaintenance int);

Footer

SHOW COLUMNS FROM facilities;

Q3

Solution

ALTER TABLE members CHANGE zipcode pincode int;

Footer

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE

TABLE\_NAME = 'members';

Q4

Solution

INSERT INTO employees(employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,manager\_id,department\_id) VALUES (100,'Steven','King','steven.king@sqltutorial.org','515.123.4567','1987-06-17','AD\_PRES',24000.00,NULL,9);

INSERT INTO employees(employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,manager\_id,department\_id) VALUES (101,'Neena','Kochhar','neena.kochhar@sqltutorial.org','515.123.4568','1989-09-21','AD\_VP',17000.00,100,9);

INSERT INTO employees(employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,manager\_id,department\_id) VALUES (102,'Lex','De Haan','lex.de haan@sqltutorial.org','515.123.4569','1993-01-13','IT\_PROG',17000.00,100,9);

INSERT INTO employees(employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,manager\_id,department\_id) VALUES (176,'Jonathon','Taylor','jonathon.taylor@sqltutorial.org',NULL,'1998-03-24','AC\_ACCOUNT',8600.00,100,8);

Footer

select count(\*) from employees;

Q5

Solution

DELETE FROM members;

Footer

select count(\*) from members;

Q6

Solution

UPDATE facilities set membercost=1000 where monthlymaintenance>500;

Footer

select \* from facilities;

Q7

Solution

CREATE TABLE orders (

orderNumber int primary key NOT NULL,

orderDate date NOT NULL,

requiredDate date NOT NULL,

shippedDate date NOT NULL,

status varchar(15) NOT NULL,

comments text NOT NULL,

customerNumber int NOT NULL,

FOREIGN KEY (customerNumber) REFERENCES customers (customerNumber)

);

Footer

SELECT

TABLE\_NAME,COLUMN\_NAME,CONSTRAINT\_NAME, REFERENCED\_TABLE\_NAME,REFERENCED\_COLUMN\_NAME

FROM

INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE

WHERE TABLE\_NAME = 'orders';

Q8

Solution

ALTER TABLE employee

ADD CONSTRAINT depIdFk FOREIGN KEY (deptId)

REFERENCES department(departmentId);

Footer

SELECT

TABLE\_NAME,COLUMN\_NAME,CONSTRAINT\_NAME, REFERENCED\_TABLE\_NAME,REFERENCED\_COLUMN\_NAME

FROM

INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE

WHERE TABLE\_NAME = 'employee';

Q9

Solution

select facid, extract(month from starttime) as month, sum(slots) as "Total Slots"

from

bookings

where

starttime >= '2012-07-01'

and starttime <= '2012-08-31'

group by facid, month

order by facid, month;

Q10

Solution

SELECT cust\_id as Customer\_ID,ord\_date as Date,MAX(purchase\_amt) as Amount

FROM AGG\_ORDERS

GROUP BY ord\_no,cust\_id,ord\_date

HAVING MAX(purchase\_amt) IN(1000 ,2200,3700, 4000)

ORDER by cust\_id;

Q11

Solution

SELECT A.Employee\_id as ID,A.First\_name as First\_name, SUM(B.Incentive\_amount) as Incentive

FROM EMPLOYEE\_XYZ A

JOIN INCENTIVES\_XYZ B

ON A.Employee\_id =B.Id

GROUP BY B.Id

ORDER BY B.Id ASC;

Q12

Solution

SELECT authorBooks.authorName AS Author\_Name, SUM(soldCopies.soldCopies) AS sold\_sum

FROM authorBooks

JOIN soldCopies

ON authorBooks.bookName = soldCopies.bookName

GROUP BY authorBooks.authorName

ORDER BY sold\_sum DESC LIMIT 3;

Q13

Solution

SELECT EMPLOYEES\_ABC.departmentName AS Department\_Name, AVG(SALARIES\_ABC.Salary) AS Avg\_salaries

FROM EMPLOYEES\_ABC

JOIN SALARIES\_ABC

ON EMPLOYEES\_ABC.employeeID = SALARIES\_ABC.employeeID

GROUP BY departmentName HAVING AVG(SALARIES\_ABC.Salary) < 7000

Q14

Solution

SELECT cust\_name as 'Customers'

from CUSTOMERS

where id not in

(

select customer\_id from ORDERS

)

ORDER BY 1;

Q15

Solution

SELECT first\_name, last\_name, salary, department\_id, job\_id

FROM EMPLOYEES\_SUBQ

WHERE job\_id =

( SELECT job\_id

FROM EMPLOYEES\_SUBQ

WHERE employee\_id= 110

);

Q16

Solution

SELECT

employee\_id,

first\_name,

last\_name

FROM

employees e

WHERE

NOT EXISTS( SELECT

\*

FROM

dependents d

WHERE

d.employee\_id = e.employee\_id)

ORDER BY first\_name ,

last\_name;

Q17

Solution

CREATE VIEW totalforday

AS SELECT ord\_date, COUNT(DISTINCT customer\_id) as dist\_customer,

ROUND(AVG(purch\_amt),2) AS AVG, ROUND(SUM(purch\_amt),2) AS SUM

FROM orders

GROUP BY ord\_date;

Footer

select \* from totalforday;

Q18

Solution

CREATE VIEW gradecount (grade, number)

AS SELECT grade, COUNT(\*)

FROM customer

GROUP BY grade;

Footer

select \* from gradecount;

Q19

Solution

DELIMITER $$

CREATE PROCEDURE EmployeeList(

In dept\_id INT, INOUT employee\_list VARCHAR(2000))

BEGIN

DECLARE v\_finished INTEGER DEFAULT 0;

DECLARE v\_ename VARCHAR(100) DEFAULT "";

DECLARE C1 CURSOR FOR

SELECT EMP\_NAME FROM Employees WHERE DEPARTMENT\_ID = dept\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET v\_finished = 1;

OPEN C1;

get\_emp: LOOP

FETCH C1 INTO v\_ename;

IF v\_finished = 1 THEN

LEAVE get\_emp;

END IF;

SET employee\_list = CONCAT(employee\_list, v\_ename, ";");

END LOOP get\_emp;

CLOSE C1;

END $$

DELIMITER ;

Footer

SET @employeelist = '';

CALL EmployeeList(97, @employeelist);

SELECT @employeelist;

Q20

Solution

DELIMITER $$

CREATE PROCEDURE ProductCount(

IN in\_prod\_type VARCHAR(30))

BEGIN

DECLARE v\_finished INTEGER DEFAULT 0;

DECLARE v\_prod\_type VARCHAR(30) DEFAULT "";

DECLARE v\_prod\_count INTEGER DEFAULT 0;

DECLARE C1 CURSOR FOR

SELECT product\_type, count(\*) FROM Product GROUP BY 1;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET v\_finished = 1;

OPEN C1;

get\_emp: LOOP

FETCH C1 INTO v\_prod\_type, v\_prod\_count;

IF v\_finished = 1 THEN

LEAVE get\_emp;

END IF;

IF v\_prod\_type = in\_prod\_type THEN

SELECT v\_prod\_count as ProductCount;

END IF;

END LOOP get\_emp;

CLOSE C1;

END $$

DELIMITER ;

Footer

CALL ProductCount('Hygiene');

Q21

Solution

Header

/header/

DELIMITER $$

CREATE TRIGGER product\_availability

AFTER INSERT

ON products FOR EACH ROW

BEGIN

IF NEW.availability='LOCAL' then

INSERT INTO products\_to\_stores (product\_id, store\_id) VALUES (NEW.product\_id, 1);

ELSE

INSERT INTO products\_to\_stores (product\_id, store\_id) VALUES (NEW.product\_id, 1);

INSERT INTO products\_to\_stores (product\_id, store\_id) VALUES (NEW.product\_id, 2);

END IF;

END$$

DELIMITER ;

Footer

INSERT INTO products (product\_name, cost\_price, retail\_price, availability) VALUES ('BLUETOOTH KEYBOARD', '17.60', '23.30','LOCAL');

INSERT INTO products (product\_name, cost\_price, retail\_price, availability) VALUES ('DVB-T2 RECEIVE', '49.80', '53.40','ALL');

SELECT \* FROM products\_to\_stores;

Q22

Solution

SELECT COUNT(age) as AgeCount FROM Employee WHERE age<35;

Q23

Solution

SELECT replace(station\_code,'M','K') as station\_code, name

FROM Train\_Route

ORDER BY 1;

Q24

Solution

select custid,fname,dob from customer order by dob,fname;

Q25

Solution

UPDATE facilities

SET

name = 'Snooker Table-2'

WHERE name = 'Pool Table';

Footer

SELECT \* FROM facilities;

Q26

Solution

select recommendedby, count(\*) as count

from members

where recommendedby is not null

group by recommendedby

order by recommendedby;

Q27

Solution

SELECT first\_name, last\_name

FROM EMPLOYEES\_SUBQ

WHERE salary >

( SELECT salary

FROM EMPLOYEES\_SUBQ

WHERE employee\_id=105

);

Q28

Solution

Header

/header/

DELIMITER $$

CREATE TRIGGER product\_archiver

AFTER DELETE

ON products FOR EACH ROW

BEGIN

INSERT INTO archived\_products (product\_id, product\_name, cost\_price, retail\_price, availability)

VALUES (OLD.product\_id, OLD.product\_name, OLD.cost\_price, OLD.retail\_price, OLD.availability);

END$$

DELIMITER ;

Footer

DELETE FROM products WHERE product\_id=1;

SELECT \* FROM archived\_products;

Q29

Solution

Header

/header/

DELIMITER $$

CREATE TRIGGER account\_balance\_validator

BEFORE UPDATE

ON customer\_accounts FOR EACH ROW

BEGIN

IF NEW.account\_balance < 0

THEN SET NEW.account\_balance = 0;

END IF ;

END$$

DELIMITER ;

Footer

UPDATE customer\_accounts

SET

account\_balance = -56

WHERE cust\_id = 3;

SELECT \* FROM customer\_accounts;