#Q1. Print product, price, sum of quantity more than 5 sold during all three months.

SELECT product,sum(price) FROM bank.bank\_inventory\_pricing group by product having sum(Quantity) > 5;

#Q2.Print product, quantity , month and count of records for which estimated\_sale\_price is less than purchase\_cost

SELECT product,Quantity,month,count(\*) FROM bank.bank\_inventory\_pricing where Estimated\_sale\_price < purchase\_cost group by product,Month;

#Q3. Extarct the 3rd highest value of column Estimated\_sale\_price from bank\_inventory\_pricing dataset

SELECT \* FROM bank.bank\_inventory\_pricing order by Estimated\_sale\_price desc limit 2,1 ;

#Q4. Count all duplicate values of column Product from table bank\_inventory\_pricing

SELECT product FROM bank.bank\_inventory\_pricing group by product having count(\*)>1;

#Q5. Create a view 'bank\_details' for the product 'PayPoints' and Quantity is greater than 2

Create view ‘bank details’ as select \* from bank\_inventory\_pricing where product =’PayPoints’ and quantity > 2

#Q6 Update view bank\_details1 and add new record in bank\_details1.

-- --example(Producct=PayPoints, Quantity=3, Price=410.67)

Insert into bank\_details (Product,Quantity,Price,purchase\_cost,Estimated\_sale\_price,Month) VALUES ('PayPoints', 4, 410.67,null,Null,null);

#Q7.Real Profit = revenue - cost Find for which products, branch level real profit is more than the estimated\_profit in Bank\_branch\_PL.

Select branch,sum(revenue)-sum(cost) Profit from branch\_pl group by branch having profit > estimated\_profit

#Q8.Find the least calculated profit earned during all 3 periods

SELECT sum(revenue-cost) Profit ,month FROM bank.bank\_branch\_pl group by month order by Profit limit 1;

#Q9. In Bank\_Inventory\_pricing,

-- a) convert Quantity data type from numeric to character

-- b) Add then, add zeros before the Quantity field.

SELECT LPAD(CAST(Quantity as char),2,0) as Padded FROM bank\_inventory\_pricing ;

#Q10. Write a MySQL Query to print first\_name ,last\_name of the titanic\_ds whose first\_name Contains ‘U’

Select first\_name,last\_name from employee\_details where first\_name like "%U%"

#Q11.Reduce 30% of the cost for all the products and print the products whose calculated profit at branch is exceeding estimated\_profit .

SELECT Branch,Product,Cost,(Cost-((30/100)\*Cost)) Reduced\_Cost,(revenue-(Cost-((30/100)\*Cost))) Profit ,Estimated\_profit FROM bank.bank\_branch\_pl group by Product having Profit > Estimated\_profit;

#Q12.Write a MySQL query to print the observations from the Bank\_Inventory\_pricing table excluding the values “BusiCard” And “SuperSave” from the column Product

SELECT \* FROM bank.bank\_inventory\_pricing where Product not in ('BusiCard') and Product not in ('SuperSave') ;

#Q13. Extract all the columns from Bank\_Inventory\_pricing where price between 220 and 300

SELECT \* FROM bank.bank\_inventory\_pricing where Price > 220 and Price <300;

#Q14. Display all the non duplicate fields in the Product form Bank\_Inventory\_pricing table and display first 5 records.

SELECT \* FROM bank.bank\_inventory\_pricing group by Product limit 5;

#Q15.Update price column of Bank\_Inventory\_pricing with an increase of 15% when the quantity is more than 3.

update bank\_inventory\_pricing SET Price=(Price+(15/100)\*Price) where Quantity>3 and price is not null;

#Q16. Show Round off values of the price without displaying decimal scale from Bank\_Inventory\_pricing

SELECT ROUND(Price,0) Price FROM bank.bank\_inventory\_pricing;

#Q17.Increase the length of Product size by 30 characters from Bank\_Inventory\_pricing.

Alter table bank\_inventory\_pricing modify Product varchar(30);

SELECT \* FROM bank.bank\_inventory\_pricing;

#Q18. Add '100' in column price where quantity is greater than 3 and dsiplay that column as 'new\_price'

SELECT \*,price+100 new\_price FROM bank.bank\_inventory\_pricing;

#Q19. Display all saving account holders have “Add-on Credit Cards" and “Credit cards"

Select \* from bank\_account\_details a where a.account\_type IN ("Add-on Credit Card" ,"Credit Card") and a.customer\_id IN (SELECT distinct b.Customer\_id FROM bank.bank\_account\_details b where b.Account\_type = "SAVINGS")

#Q20.

# a) Display records of All Accounts , their Account\_types, the transaction amount.

Select a.account\_number, account\_type, transaction\_amount from bank\_account\_transaction a left outer join bank\_account\_details b on a.account\_number = b.account\_number

# b) Along with first step, Display other columns with corresponding linking account number, account types

Select a.account\_number,b.account\_type,a.transaction\_amount,linking\_account\_number

from bank\_account\_transaction a

left outer join bank\_account\_details b

on a.account\_number = b.account\_number

left outer join bank\_account\_relationship\_details c

on a.account\_number = c.account\_number

# c) After retrieving all records of accounts and their linked accounts, display the transaction amount of accounts appeared in another column.

Select a.account\_number,b.account\_type,a.transaction\_amount,linking\_account\_number

from bank\_account\_transaction a

left outer join bank\_account\_details b

on a.account\_number = b.account\_number

left outer join bank\_account\_relationship\_details c

on a.account\_number = c.account\_number

#Q21.Display all type of “Credit cards” accounts including linked “Add-on Credit Cards"

# type accounts with their respective aggregate sum of transaction amount.

# Ref: Check linking relationship in bank\_transaction\_relationship\_details.

# Check transaction\_amount in bank\_account\_transaction.

Select a.account\_number, account\_type, sum(transaction\_amount )

from bank\_account\_transaction a

left outer join bank\_account\_details b

on a.account\_number = b.account\_number

where account\_type IN (“Credit Card”, "Add-on Credit Card")

group by a.account\_number

#Q22. Compare the aggregate transaction amount of current month versus aggregate transaction with previous months.

# Display account\_number, transaction\_amount ,

-- sum of current month transaction amount ,

-- current month transaction date ,

-- sum of previous month transaction amount ,

-- previous month transaction date.

Select \* from

(SELECT account\_number, sum(transaction\_amount) from bank\_account\_transaction y where month(transaction\_date) != month(current\_timestamp()) group by account\_number) a

Full outer join

(SELECT account\_number, sum(transaction\_amount) from bank\_account\_transaction x where month(transaction\_date) = month(current\_timestamp()) group by account\_number) b

On a.account\_number = b.account\_numberOn a.account\_number = b.account\_number

#Q23.Display individual accounts absolute transaction of every next month is greater than the previous months .

#Q24. Find the no. of transactions of credit cards including add-on Credit Cards

SELECT Account\_type,count(\*) FROM bank.bank\_account\_details group by Account\_type having Account\_type='Credit Card' or Account\_type='Add-on Credit Card';

#Q25.From employee\_details retrieve only employee\_id ,first\_name ,last\_namephone\_number ,salary, job\_id where department\_name is Contracting (Note

#Department\_id of employee\_details table must be other than the list within IN operator.

ALTER TABLE employee\_details ADD COLUMN Department\_Name Text;

UPDATE employee\_details t1

INNER JOIN department\_details t2 ON t1.employee\_id = t2.employee\_id

SET t1.Department\_Name = t2.Department\_Name;

SELECT employee\_id ,first\_name ,last\_name,phone\_number ,salary, job\_id,DEPARTMENT\_Name FROM bank.employee\_details where DEPARTMENT\_Name='Contracting';

#Q26. Display savings accounts and its corresponding Recurring deposits transactions are more than 4 times.

Select \* from bank\_account\_details a where a.account\_type IN ("RECCURING") and a.customer\_id IN (SELECT distinct b.Customer\_id FROM bank.bank\_account\_details b where b.Account\_type = "SAVINGS")

#Q27. From employee\_details fetch only employee\_id, ,first\_name, last\_name , phone\_number ,email, job\_id where job\_id should not be IT\_PROG.

SELECT employee\_id ,first\_name ,last\_name,phone\_number ,email, job\_id FROM bank.employee\_details where job\_id!='IT\_PROG';

#Q29.From employee\_details retrieve only employee\_id ,first\_name ,last\_namephone\_number ,salary, job\_id where manager\_id is '60' (Note

#Department\_id of employee\_details table must be other than the list within IN operator.

SELECT a.employee\_id ,first\_name ,last\_name,phone\_number ,salary, job\_id,a.MANAGER\_ID FROM bank.employee\_details a left outer join department\_details b on a.DEPARTMENT\_ID=b.DEPARTMENT\_NAME where b.MANAGER\_ID = '60';

#Q30.Create a new table as emp\_dept and insert the result obtained after performing inner join on the two tables employee\_details and department\_details.

Create table emp\_dept as

select b.employee\_id, first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,

b.department\_id,a.department\_name,a.manager\_id,location\_id

from department\_details a

join employee\_details b

on a.department\_id = b.department\_id