**Q1) Consider the employee database. Give an SQL DDL definition of this database. Identify referential-integrity constraints that should hold, and include them in the DDL definition**

****

-- in this question I want to use **create**

create table employee (  
 ID int primary key,  
 person\_name varchar2(100),  
 street varchar2(255),  
 city varchar2(100)  
);

create table works (  
 ID INT primary key,  
 company\_name varchar2(100) references company(company\_name) on delete cascade,  
 salary decimal(10, 2)  
);

create table company (  
 company\_name varchar2(100) primary key,  
 city varchar2(100)  
);

create table manager (  
 ID int primary key,  
 manager\_id int references employee(ID) on delete set null  
);

**Q2) Consider the bank database, where the primary keys are underlined. Construct the following SQL queries for this relational database.**



**A) Find the ID of each customer of the bank who has an account but not a loan.**

**-- First create the tables:**  
create table branch (  
 branch\_name varchar2(100) primary key,  
 branch\_city varchar2(100),  
 assets decimal(15, 2)  
);

create table customer (  
 ID varchar2(10) primary key,  
 customer\_name varchar2(100),  
 customer\_street varchar2(255),  
 customer\_city varchar2(100)  
);

create table loan (  
loan\_number int primary key,  
branch\_name varchar2(100) references branch(branch\_name) on delete cascade,  
amount decimal(15, 2)  
);

create table borrower (  
ID varchar2(10),  
 loan\_number int references loan(loan\_number) on delete cascade,  
 primary key (ID, loan\_number)  
);

create table account (  
 account\_number int primary key,  
 branch\_name varchar2(100) references branch(branch\_name) on delete cascade,  
balance decimal(15, 2)  
);

create table depositor (  
ID varchar2(10),  
 account\_number int references account(account\_number) on delete cascade,  
 primary key (ID, account\_number)  
);

-- Then insert value into them, these are the tables:

Branch table:  


customer table:  


Loan table:   
A black and white screen with white text

Description automatically generated

Borrower table:  
A black rectangular box with white text

Description automatically generated

Account table:  
A black and white table with white text

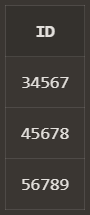
Description automatically generated

Depositer table:  
A screenshot of a phone

Description automatically generated

**The Answer:**

SELECT DISTINCT d.ID  
FROM depositor d  
LEFT JOIN borrower b ON d.ID = b.ID  
WHERE b.loan\_number IS NULL;

**Output:  
**

**Q2)**

**B) Find the ID of each customer who lives on the same street and in the same city as customer '12345'.**

**The Answer:**

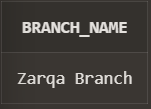
**SELECT c2.ID  
FROM customer c1  
JOIN customer c2 ON c1.customer\_street = c2.customer\_street  
 AND c1.customer\_city = c2.customer\_city  
WHERE c1.ID = '12345' AND c2.ID != '12345';**

**Q2)**

**C) Find the name of each branch that has at least one customer who has an account in the bank and who lives in “Harrison”.**

**The Answer:**

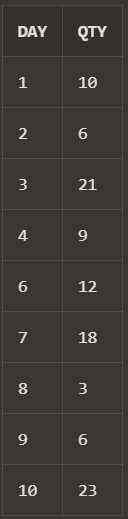
**select distinct b.branch\_name  
from branch b  
join account a ON b.branch\_name = a.branch\_name  
join depositor d ON a.account\_number = d.account\_number  
join customer c ON d.ID = c.ID  
where c.customer\_city = 'Harrison';**

**Output:   
**

**Q3) Using SQL Window Functions.**

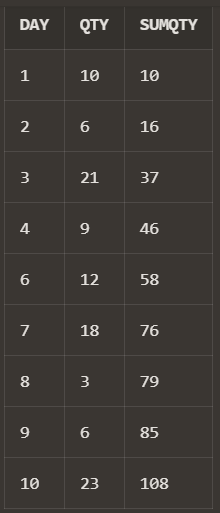
**A. From the demand table, find the cumulative total sum for qty.**

**-- First of all I must create the table and insert the values to insure that my answer is correct**create table sales (  
 day int,  
 qty int  
);

-- Then insert the data as it in the pdf:  
select \* from sales  


**-- The Answer:**

select day, qty,  
sum(qty) over(order by day) as sumQty  
from sales

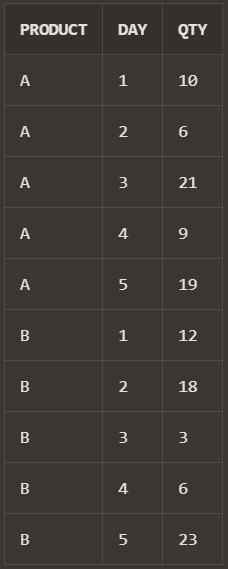
**Output:**

**Q3)**

**B. Extract the two worst performing days of each product in terms of number of qty sold. Paraphrasing it: Get the days corresponding to the two minimum most values of qty for each product.**

**-- First of all I must create the table and insert the values to insure that my answer is correct**create table sales (  
 product char(1),   
 day int,  
 qty int  
);

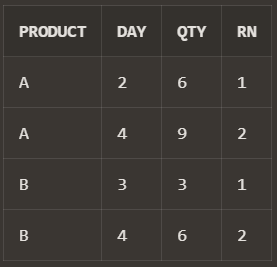
-- Then insert the data as it in the pdf:  
Select \* from sales

**Output:**  


**The Answer:**

**-- Firstly, I want to create a view that contains the products and there dense ranks, but every product separately:**   
create view RankedSales as   
select product, day, qty,  
 dense\_rank() over(partition by product order by qty) as RN  
 from sales

**-- Then after that I want to specify only 2 minimum product values by specifying that RN=1 OR RN=2 so that the output will be only those whose rank 1 and 2 which is the 2 minimum qty values**select \*  
from RankedSales   
where RN = 1 or RN = 2  
order BY products

**Output:  
**