Lab 7

Relational Algebra & SQL Queries

1. Find the ordered items which are in stock.

SQL Statement :

```
select s.itemid,orderQuantity.orderid from stock as s
join (select oc.itemid,oc.quantity,o.orderid from orders as o
join ordercontains as oc
on oc.orderid= o.orderid) as orderQuantity
on orderQuantity.itemid = s.itemid
where orderQuantity.quantity<=s.quantity;
```

Relational Algebra:

$$\pi_{\text{item.id,orderid}}(\text{stock}_{<\text{orderQuantity.itemid=stock.itemid AND orderQuantity.quantity<=stock.quantity}>}\bowtie(\text{orderQuantity},(\sigma_{(\text{ordercontains.itemid,ordercontains.quantity})}(\text{orders}))_{<\text{ordercontains.orderid=orders.orderid}>}\bowtie\text{ordercontains}))$$

	itemid character (10)	orderid character (10)
1	ITEM000005	ORDER00001
2	ITEM000001	ORDER00001
3	ITEM000001	ORDER00002

2. Find the customers whose orders are in stock.

SQL Statement :

```
Select s.itemid,orderQuantity.orderid,orderQuantity.name from stock as s
JOIN (select oc.itemid,oc.quantity,o.orderid,c.name from orders as o
JOIN ordercontains as oc
on oc.orderid= o.orderid
JOIN customer as c
on c.customerid= o.customerid
) as orderQuantity
on orderQuantity.itemid = s.itemid
where orderQuantity.quantity<=s.quantity;
```

Relational Algebra:

$\pi_{\text{stock.item.id,orderQuantity.orderid,orderQuantity.name}}(\text{stock}_{<\text{orderQuantity.itemid=stock.itemid AND orderQuantity.quantity<=stock.quantity}>}\bowtie(\text{orderQuantity},(\sigma_{(\text{ordercontains.itemid,ordercontains.quantity, orders.orderid,customer.name})}(\text{orders}))_{<(\text{ordercontains.orderid=orders.orderid}),(\text{orders.customerid=customer.customerid})>}\bowtie \text{ordercontains} \bowtie \text{customer}))$

	name character varying (20)	orderid character (10)	itemid character (10)
1	Akhilesh	ORDER00001	ITEM000005
2	Akhilesh	ORDER00001	ITEM000001
3	Mitesh	ORDER00002	ITEM000001

3. Display the category names with their ordered quantities from all the orders placed.

SQL Statement :

```
Select name,sum(result.quantity) from
(Select itemcategory.name,ordercontains.quantity from itemcategory
JOIN item on item.categoryid = itemcategory.categoryid
JOIN ordercontains on item.itemid = ordercontains.itemid
JOIN orders on ordercontains.orderid = orders.orderid
) as result
GROUP BY name
```

Relational Algebra:

$r1 = \pi(_{itemcategory.name,ordercontains.quantity})itemcategory$
$\bowtie_{<items.categoryid=itemcategory.categoryid>\ )}item \bowtie_{<items.itemsid=ordercontains.itemid>\ )}$
$ordercontains \bowtie_{<ordercontains.itemsid=orders.itemid>\ )}orders$

$result=\pi_{(name,}\mathcal{F}_{SUM(quantity))}(r1)$

	name character varying (20)	sum numeric
1	Jewellery	2
2	Grooming	19
3	Fashion	9

4. Show the number of warehouses present in each city.

SQL Statement :

```
Select city.name,count(*) from warehouse
JOIN city on city.cityid=warehouse.cityid
GROUP BY city.name
```

Relational Algebra:

$\pi_{city.name,}\mathcal{F}_{count(*)}(warehouse \bowtie_{<warehouse.cityid=city.cityid>} city)$

	name character varying (20)	count bigint
1	Gandhinagar	1
2	Vadodara	2
3	Surat	2

5. List all the managers of different warehouses.

SQL Statement :

```
Select emp.* from employee as emp
JOIN warehouse as w
on w.managerid = emp.emp_id
```

Relational Algebra:

$\pi_{emp.*}(employee \bowtie_{<warehouse.managerid=employee.employeeid>} warehouse)$

	emp_id [PK] character (10)	warehouseid character (10)	name character (20)	salary numeric	contact character (10)
1	EMP0000001	WRHSE00001	Suresh	10000	9123456789
2	EMP0000006	WRHSE00004	Akshay	15000	9132456789
3	EMP0000003	WRHSE00002	Shrey	11000	9123456879

6. In which warehouses the employee count is below the average count.

SQL Statement :

```
Select warehouseid ,count(*) as Employee_count from employee as e
GROUP BY e.warehouseid having count(*) >
(select Trunc(avg(ware_count.count) ,2)from
(select warehouseid ,count(*) as count
from employee as e
GROUP BY e.warehouseid) as ware_count)
```

Relational Algebra:

r1 $=_{warehouseid,}\mathcal{F}_{count(*)\text{->}count}$ (employee)

r2=$\mathcal{F}_{AVG(count)}$ (r1)

r3=$\sigma_{(r1..count>r2.AVG(count))}$

result = $\pi_{(warehouseid,count)}$(r3)

	warehouseid character (10)	employee_count bigint
1	WRHSE00002	2
2	WRHSE00001	2

7. List all the pending orders.

SQL Statement :

```
Select * from orders where order_delivery_date is null
```

Relational Algebra:

$\sigma_{(order\_delivery\_date\ =\ NULL)}$(orders)

	orderid [PK] character (10)	transportid character (10)	customerid character (10)	order_placed_date date	order_delivery_date date
1	ORDER00004	TRANS00002	CUST000003	2020-08-02	[null]

8. List the customerid with the highest number of orders placed.

SQL Statement :

Select customerid from
(select customerid ,count(*) from orders
GROUP BY customerid) as maxcount
where count in (select max(count) from (select customerid ,count(*) from orders
GROUP BY customerid)as maxc)

Relational Algebra:

$r1 = {}_{customerid,}\mathcal{F}_{count(*)\text{->}count}\ (orders)$

$r2 = \mathcal{F}_{max(count)}\ (r1)$

$result = \pi_{(customerid,count)}\sigma_{(r1.count=r2.max(count))}$

	customerid character (10)
1	CUST000003

9. Name the vehicle no/id which is having the highest number of orders to transport.

SQL Statement :

Select vehicleid,count from
(select vehicleid, count(*) from transport
GROUP BY vehicleid) as maxcount
where count in (select max(count) from (select vehicleid, count(*) from
transport GROUP BY vehicleid) as maxc)

Relational Algebra:

$r1 = {}_{vehicleid,}\mathcal{F}_{count(*)\text{->}count}\ (transport)$

$r2 = \mathcal{F}_{max(count)}\ (r1)$

$result = \pi_{(vehicleid,count)}\sigma_{(r1.count=r2.max(count))}$

	vehicleid character (10)	count bigint
1	VEHIC00001	1
2	VEHIC00003	1
3	VEHIC00006	1

10. Get the most popular item from each category

SQL Statement :

```
select ic2.name,i2.name from itemCategory ic2
	join item i2 on i2.itemCategoryID = ic2.itemCategoryID
	join (
		select tmp.ici ici ,max(cnt) mx from (
			select i.itemCategoryID ici,i.itemID ii,count(oc.quantity) cnt from
item i
			join orderContains oc on oc.itemID = i.itemID
			group by i.itemCategoryID,i.itemID
			) tmp
			group by tmp.ici
	) tmp2
	on ic2.itemCategoryID = tmp2.ici
	where tmp2.mx = (select count(*) from orderContains where itemID =
i2.itemID);
```

Relational Algebra:

$\prod_{1c2.name,i2.name}$ (
ρ (ic2,itemCategoryID) $\bowtie_{i2.itemCategoryID = ic2.itemCategoryID}$
ρ (i2,item) $\bowtie_{temCategoryID = tmp2.ici}$
ρ (tmp2 (tmp.ici → ici, max(cnt) → mx), (
$\prod_{tmp.ici\ ici\ ,max(cnt)\ mx}$ ($_{tmp.ici}$ F $_{max(cnt)}$ (
ρ (tmp(i.itemCategoryID → ici, i.itemID → ii,count(oc.quantity) → cnt),(
$\prod_{i.itemCategoryID\ ,i.itemID\ ,count(oc.quantity)}$ (
$_{i.itemCategoryID,i.itemID}$ F $_{count(oc.quantity)}$ (
ρ (i,item) $\bowtie_{oc.itemID = i.itemID}$ ρ (oc,orderContains)
)
)
))
))
))
)

```
      name      |  name
----------------+---------
 toys           | Item 1
 toys           | Item 2
 home appliance | Item 3
 home appliance | Item 5
 device         | Item 6
 vehicle        | Item 7
 vehicle        | Item 8
 vehicle        | Item 9
 cosmetic       | Item 10
```

11. List all customers who met the given employee in last week

SQL Statement :

```
select distinct c.name,o.orderID from employee e
      join transport t on t.employeeID = e.employeeID
      join _order o on o.transportID = t.transportID
      join customer c on c.customerID = o.customerID
      where e.employeeID = 1
                and o.dileveryDate between current_date -7 and  current_date
```

Relational Algebra:

$$\Pi_{\text{c.name,o.orderID}} \Big(\sigma_{\text{e.employeeID = 1 and o.dileveryDate between current\_date -7 and current\_date}} \big(\rho(\text{e,employee}) \bowtie_{\text{t.employeeID = e.employeeID}} \rho(\text{t,transport}) \bowtie_{\text{o.transportID = t.transportID}} \rho(\text{o,\_order}) \bowtie_{\text{c.customerID = o.customerID}} \rho(\text{c,customer}) \big) \Big)$$

```
    name    | orderid
------------+---------
 Customer 1 |       1
```

12. The average salary of employees at each warehouse

SQL Statement :

```
select (select name from warehouse where warehouseID=w.warehouseID)as
name,avg(e.salary) avg_salary from warehouse w
      join employee e on e.warehouseID=w.warehouseID
      group by w.warehouseID
```

IMP - * x * means subscript – used to express subscript of a subscript
Relational Algebra:

$\Pi_{\Pi_{name}(\sigma_{warehouseID=w.warehouseID}(warehouse)),\ avg(salary)}($

$\quad {}_{w.warehouseID}F_{avg(e.salary)}($

$\qquad \rho(w,warehouse) \bowtie_{e.warehouseID=w.warehouseID} \rho(e,employee)$

$\quad)$

$)$

name	avg_salary
WareHouse 4	25400.000000000000
WareHouse 6	25400.000000000000
WareHouse 2	50400.000000000000
WareHouse 3	38466.666666666667
WareHouse 5	25600.000000000000
WareHouse 7	3175400.000000000000
WareHouse 1	30400.000000000000

13. List of items ordered by a given customer in chronological order

SQL Statement :

```
select i.name from customer c
    join _order o on o.customerID=c.customerID
    join orderContains oc on oc.orderID=o.orderID
    join item i on i.itemID=oc.itemID
    where c.customerID = 1
    order by o.orderDate;
```

Relational Algebra:

$$
\prod_{i.name} \Big(\sigma_{c.customerID = 1} \big(\rho(c,customer) \bowtie_{o.customerID=c.customerID} \rho(o,\_order) \bowtie_{oc.orderID=o.orderID} \rho(oc,orderContains) \bowtie_{i.itemID=oc.itemID} \rho(i,item) \big) \Big)
$$

```
  name
--------
 Item 6
 Item 1
 Item 6
```

14. List suppliers in order of their popularity

SQL Statement :

```
select (select name from supplier where supplierID = s.supplierID) as  name  from
supplier s
       join suppliedBy sb on sb.supplierID = s.supplierID
       join orderContains oc on (oc.orderID=sb.orderID and oc.itemID=sb.itemID)
       group by s.supplierID
       order by sum(oc.quantity) desc;
```

Relational Algebra:

IMP - * x * means subscript – used to express subscript of a subscript
IMP – Can't completely express it in relational algebra because it doesn't support ordering

$\Pi_{\Pi\ \text{*name* } (\sigma\ \text{* supplierID = s.supplierID* supplier})}$ (

$_{\text{s.supplierID}}F$ (

ρ (s,supplier) $\bowtie_{\text{sb.supplierID = s.supplierID}}$
ρ (sb,suppliedBy) $\bowtie_{\text{(oc.orderID=sb.orderID and oc.itemID=sb.itemID)}}$
ρ (oc,orderContains)

)

)

```
    name
------------
 Supplier 2
 Supplier 9
 Supplier 4
 Supplier 6
 Supplier 5
 Supplier 1
 Supplier 3
 Supplier 7
 Supplier 8
```

15. Calculate the average delivery time in each city

SQL Statement :

```
select (select name from city where cityID = c.cityID) as name ,avg(o.dileveryDate-o.orderDate) days from city c

    join warehouse w on w.cityID = c.cityID

    join employee e on e.warehouseID = w.warehouseID

    join transport t on t.employeeID=e.employeeID

    join _order o on o.transportID = t.transportID

    group by c.cityID

      order by avg(o.dileveryDate-o.orderDate)
```

Relational Algebra:

IMP - * x * means subscript – used to express subscript of a subscript

$\prod_{\prod \text{*name* } (\sigma\text{* cityID = c.cityID* (city)) , avg(o.dileveryDate-o.orderDate)}}$ (

$\quad {}_{\text{c.cityID}} F_{\text{avg(o.dileveryDate-o.orderDate)}}$ (

$\qquad \rho\ (c,city) \bowtie_{\text{w.cityID = c.cityID}}$

$\qquad \rho\ (w,warehouse) \bowtie_{\text{e.warehouseID = w.warehouseID}}$

$\qquad \rho\ (e,employee) \bowtie_{\text{t.employeeID=e.employeeID}}$

$\qquad \rho\ (t,transport) \bowtie_{\text{o.transportID = t.transportID}}$

$\qquad \rho\ (o,\_order)$

$\quad$)

)

```
   name    |        days
-----------+--------------------
 ahmedabad | 3.1000000000000000
```

16. Calculate the average number of items in a single order

SQL Statement :

```
select avg(cnt) from (
	select count(*) as cnt from orderContains
	group by orderID
) tmp ;
```

Relational Algebra:

$F_{avg(cnt)}$ (

ρ ((count(*) $\rightarrow$ cnt),

${}_{orderID} F_{count(*)}$ (

orderContains

)

)

)

```
        avg
----------------------
 1.5555555555555556
```

17. List drivers ordered by their 'Bang for Buck'

SQL Statement :

```
select (select name from employee where employeeID=e.employeeID) as name from
employee e
    join transport t on t.employeeID = e.employeeID
    group by e.employeeID,e.salary
    order by (e.salary/count(*))
;
```

.**Relational Algebra:**

IMP - * x * means subscript – used to express subscript of a subscript
IMP – Can't completely express it in relational algebra because it doesn't support ordering

$\prod_{\prod \text{ *name * } (\sigma \text{ * employeeID=e.employeeID* } (\text{employee}))}$ (
$\quad {}_{\text{e.employeeID,e.salary}} F$ (
$\qquad \rho\,(\text{e,employee}) \bowtie_{\text{t.employeeID = e.employeeID}}$
$\qquad \rho\,(\text{t,transport})$
$\quad$)
)

```
    name
------------
 Employee 1
 Employee 2
 Employee 3
 Employee 4
```

18. List the supplier , who is supplying the highest count of items.

SQL Statement :

```
Select supplierid, count from
(select supplierid, count(*) from suppliedby
GROUP BY supplierid) as maxcount
where count in (select max(count) from (select supplierid, count(*) from
suppliedby GROUP BY supplierid) as maxc)
```

Relational Algebra:

r1 = $_{supplierid,}\mathcal{F}_{count(*)\text{->}count}$ (suppliedby)

r2=$\mathcal{F}_{max(count)}$ (r1)

result = $\pi_{(supplierid,count)}\sigma_{(r1.count=r2.max(count))}$

	supplierid character (10)	count bigint
1	SUPPL00001	4

19. List name of item having ordered quantity between 15 & 45 and the name should start with the character ‘P’ having third character ‘a’

SQL Statement :

```
select name from item
join stock
on stock.itemid=item.itemid
where quantity between 15 and 45 and name LIKE 'P_a%'
```

Relational Algebra:

$\pi_{name}(\sigma(_{quantity>14 \text{ and } quantity<46}$ AND name(‘P_a%’))item$_{<item.itemid=stock.itemid>}\bowtie$stock)

	name character varying (20)
1	Plastic Containers

20. List the warehouse names and item names with stock of items with quantity 15 to 45

SQL Statement :

```
select warehouse.name,itemNameWarehouseId.name from warehouse
join (select name,itemWarehouse.warehouseid from item
        join (select itemid,warehouseid from stock
                where quantity between 15 and 45) as itemWarehouse
        on item.itemid = itemWarehouse.itemid) as itemNameWarehouseId
        on warehouse.warehouseid = itemNameWarehouseId.warehouseid
```

Relational Algebra:

$$\pi_{\text{warehouse.name,itemNameWarehouseId.name}}(\text{warehouse}_{<\text{warehouse.id = itemNameWarehouseId.warehouseid}>}\bowtie(\rho(_{\text{itemNameWarehouseId}},(\pi_{\text{name,warehouseid}}(\text{item}_{<\text{item.itemid = itemWarehouse.itemid}>}\bowtie(\rho(\text{itemWarehouse},(\pi_{\text{itemid,warehouseid}}(\sigma(_{\text{quantity>14 and quantity<46}})\text{stock})))))$$

Data Output | Explain | Messages | Notifications

	name character (30)	name character varying (20)
1	Vadodara Warehouse 1	Plastic Containers

21. List all customer who ordered all the items

SQL Statement :

```
select customerid from orders
except
(
select customerid from
(
select o.customerid, items.itemid from orders as o
        join ordercontains as oc
        on oc.orderid=o.orderid
        cross join
(select * from item) as items
except
```

```
select customerid, itemid from orders as o join ordercontains as oc on
oc.orderid=o.orderid
) as r2
)
```

Relational Algebra:

$r0 = \pi(_{items})$
$r1 = \pi_{o.customerid,r0.itemid}(\rho(_{o.orders})_{<o.orderid=oc.orderid>}{}^{\bowtie}(\rho(oc,ordercontains) \text{ X } r0)$
$r2 = r1 - (\pi(_{customerid,itemid})(\rho(_{o.orders})_{<o.orderid=oc.orderid>}{}^{\bowtie}(\rho(oc,ordercontains)))$
$result = \pi_{customerid}(orders) - r2$

Data Output | Explain

	customerid character (10)
1	CUST000001

22. List items that are available at all warehouses.

SQL Statement :

```
select i.itemID, i.name from item i
where i.itemID not in (
	select distinct tmp.itemID from (
		select i2.itemID,w.warehouseID from warehouse w,item i2
		except
		select distinct itemID,warehouseID from stock
	) tmp
);

 itemid |  name
--------+--------
      8 | Item 8
```

23. List warehouses that have all the items.

SQL Statement :

```
select w.warehouseID, w.name from warehouse w
where w.warehouseID not in (
        select distinct tmp.warehouseID from (
                select i2.itemID,w.warehouseID from warehouse w,item i2
                except
                select distinct itemID,warehouseID from stock
        ) tmp
);
```

```
 warehouseid |    name
-------------+-------------
           2 | WareHouse 2
```

24. List highest salary of employees in each warehouse.

SQL Statement :

```
SELECT w.name, MAX(Salary) FROM Employee as e
JOIN WareHouse as w
ON e.warehouseid = w.warehouseid
GROUP BY w.name;
```

Relational Algebra:

$$\pi_{w.name,}\mathcal{F}_{max(salary)}(\rho(e,employee)_{<e.warehouseid=w.warehouseid>}\bowtie\rho(w,warehouse))$$

	name character (30)	max numeric
1	Vadodara Warehouse 1	15000
2	Surat Warehouse 1	20000
3	Surat Warehouse 2	12000
4	Vadodara Warehouse 2	9000

25. Find the maximum price of each category

SQL Statement :

```
SELECT  name,maxprice
FROM    item  a
JOIN
(
        SELECT Categoryid, MAX(cost) as maxPrice
        FROM item
        GROUP BY Categoryid
) b
ON a.Categoryid = b.Categoryid AND a.cost = b.maxPrice
```

Relational Algebra:

$r1 = \rho(a,(\pi_{name,maxprice}(item))$

$r2 = \rho(b,(\pi_{categoryid,}(\mathcal{F}_{max(cost) \rightarrow maxPrice})(item))$

$Result = r1 \bowtie_{<a.categoryid\ =\ b.categoryid\ AND\ a.cost\ =\ b.maxPrice>} r2$

	name character varying (20)	maxprice integer
1	Earrings	3000
2	Facewash	150
3	Plastic Containers	50
4	Hoodie	450

26. List the count of warehouses in each city.

SQL Statement :

```
Select city.name,count(*) from warehouse
JOIN city
on city.cityid=warehouse.cityid
Group by city.name
```

Relational Algebra:

$\pi_{city.name, \mathcal{F}_{count(*)}}(warehouse \bowtie_{<warehouse.cityid\ =\ city.cityid>} city)$

Data Output | Explain | Messages | Notifications

	name character varying (20)	count bigint
1	Gandhinagar	1
2	Vadodara	2
3	Surat	2

27. List all the items details listed under the category 'Fashion'

SQL Statement :

```
Select * from item
JOIN itemcategory
On Item.Categoryid = itemcategory.categoryid
Where itemcategory.name = 'Fashion’
```

Relational Algebra:

$\pi.^{*}(itemcategory \bowtie_{<itemcategory.categoryid=item.categoryid\ and\ itemcategory.name='Fashion'>} item)$

Data Output | Explain | Messages | Notifications

	itemid character (10)	categoryid character (10)	name character varying (20)	cost integer	categoryid character (10)	name character varying (20)	desription text
1	ITEM000004	ITEMCAT003	Hoodie	450	ITEMCAT003	Fashion	Clothes