Write a query to create a table named **'currency'**.

CREATE TABLE currency

(id int NOT NULL,

code varchar2(255) NOT NULL,

country varchar2(255) NOT NULL,

CONSTRAINT currency\_pk PRIMARY KEY (id)

);

Write a query to create a table named **'contact'.**

CREATE TABLE contact

(id int NOT NULL,

org\_name varchar2(255) NOT NULL,

street\_address1 varchar2(255) NOT NULL,

street\_address2 varchar2(255) NOT NULL,

city varchar2(255) NOT NULL,

state varchar2(255) NOT NULL,

postal\_code varchar2(255) NOT NULL,

country\_code varchar2(255) NOT NULL,

last\_name varchar2(255),

first\_name varchar2(255) NOT NULL,

person\_title varchar2(255),

phone\_country\_code int NOT NULL,

phone\_area\_code int,

phone\_number varchar2(255) NOT NULL,

email varchar2(255) NOT NULL,

created\_time timestamp,

CONSTRAINT contact\_pk PRIMARY KEY (id)

);

Write a query to create a table named **'user\_details'**.

CREATE TABLE ASS\_1

( contact\_id number(10) NOT NULL,

department\_name varchar2(50) NOT NULL,

CONSTRAINT contact\_id\_pk PRIMARY KEY (contact\_id)

);

CREATE TABLE user\_details

(id int NOT NULL,

username varchar2(255) NOT NULL,

user\_password varchar2(255) NOT NULL,

deleted int NOT NULL,

created\_time timestamp NOT NULL,

lastupdated\_time timestamp NOT NULL,

lastlogin\_time timestamp,

failed\_attempts int,

contact\_id int NOT NULL,

CONSTRAINT user\_details\_pk PRIMARY KEY (id),

CONSTRAINT fk\_ASS\_1 FOREIGN KEY (contact\_id) REFERENCES ASS\_1 (contact\_id)

);

Write a query to change the data type from **varchar(255)**to**int** for the column name **'country\_code'**in the contact table.

ALTER TABLE contact

MODIFY country\_code int;

Write a query to insert any 3 records into **'credit\_card'** table.

INSERT INTO credit\_card (id, card\_number, card\_expire, name, cc\_type) VALUES (12346,'1234567890123', '12/31/2025', 'TEST1', 'Platinum');

INSERT INTO credit\_card (id, card\_number, card\_expire, name, cc\_type) VALUES (12347,'1234567890456', '12/31/2025', 'TEST2', 'Gold');

INSERT INTO credit\_card (id, card\_number, card\_expire, name, cc\_type) VALUES (12348,'1234567890789', '12/31/2025', 'TEST3', 'Silver');

Write a query to insert any 3 records into**'contact'** table.

INSERT INTO contact (id, org\_name, street\_address1, street\_address2, city, state, postal\_code, country\_code, last\_name, first\_name,

person\_title, phone\_country\_code, Phone\_area\_code, phone\_number, email, created\_time)

VALUES (12347, 'COG1', 'SAR1', 'TEST1', 'CBE1', 'TN1', '123456', '91', 'Me', 'Contact1', 'Testing', 91, 91, '1234567890', 'NA', CURRENT\_TIMESTAMP);

INSERT INTO contact (id, org\_name, street\_address1, street\_address2, city, state, postal\_code, country\_code, last\_name, first\_name,

person\_title, phone\_country\_code, Phone\_area\_code, phone\_number, email, created\_time)

VALUES (12348, 'COG1', 'SAR1', 'TEST1', 'CBE1', 'TN1', '123456', '91', 'Me', 'Contact1', 'Testing', 91, 91, '1234567891', 'NA', CURRENT\_TIMESTAMP);

INSERT INTO contact (id, org\_name, street\_address1, street\_address2, city, state, postal\_code, country\_code, last\_name, first\_name,

person\_title, phone\_country\_code, Phone\_area\_code, phone\_number, email, created\_time)

VALUES (12349, 'COG1', 'SAR1', 'TEST1', 'CBE1', 'TN1', '123456', '91', 'Me', 'Contact1', 'Testing', 91, 91, '1234567892', 'NA', CURRENT\_TIMESTAMP);

Write a query to change the cheque\_number with  bank name **'ICICI Bank'** to **'962541'** in cheque table.

Write a query to delete the records with the bank\_name **'Canara Bank'** in cheque table.

DELETE FROM cheque

WHERE bank\_name = 'Canara Bank';

Write a query to insert any 3 records into **'transaction\_type'** table.

INSERT INTO transaction\_type (id, type) VALUES (123457, 'NetBanking');

INSERT INTO transaction\_type (id, type) VALUES (123458, 'DEBITCARD');

INSERT INTO transaction\_type (id, type) VALUES (123459, 'CREDITCARD');

Write a query to change the code of country ' **United States**' from ' **US**' to ' **USD**' in the currency table.

Write a query to display all the user details. Display the records in ascending order based on their id.

SELECT \* FROM user\_details ORDER BY id ASC;

Write a query to display all the invoice details. Display the records in ascending order based on their id.

SELECT \* FROM invoice ORDER BY id ASC;

Write a query to display the code and country of the currency details. Display the records in ascending order based on their code.

SELECT code, country FROM currency ORDER BY code ASC;

Write a query to display all credit card details. Display the records in descending order based on their name.

SELECT \* FROM credit\_card ORDER BY name DESC;

Write a query to display  all the details of  the transaction\_type. Display the records in ascending order based on their type.

SELECT \* FROM transaction\_type ORDER BY type ASC;

Write a query to display the first\_name and last\_name of the contact details. Display the records in ascending order based on their first\_name.

SELECT first\_name, last\_name FROM contact ORDER BY first\_name ASC;

Write a query to display the bank\_name and cheque\_date of all the cheque. Display the records in ascending order based on bank\_name.

SELECT bank\_name, cheque\_date from cheque ORDER BY bank\_name ASC;

Write a query to display the first\_name,last\_name,org\_name,city and state of all the Contact details. Display the records in ascending order based on their first\_name.

SELECT first\_name, last\_name, org\_name, city, state FROM Contact ORDER BY first\_name ASC;

Write a query to display the id,attempt,status and amount of all the payments. Display the records in descending order based on their id.

SELECT id, attempt, status, amount FROM payment ORDER BY id DESC;

Write a query to display the bill\_type, active\_since,active\_until of all the purchase\_order.Display the records in ascending order based on their active\_since.

SELECT bill\_type, active\_since, active\_until FROM purchase\_order ORDER BY active\_since ASC;

Write a query to display the username, created\_time, lastlogin\_time and failed\_attempts of all the users . Display the records in ascending order based on their username.

SELECT username, created\_time, lastlogin\_time, failed\_attempts FROM user\_details ORDER BY username ASC;

Write a query to display the processor, code, approval\_code and response\_msg for all the payment\_authorization. Display the records in ascending order based on it's code.

SELECT processor, code, approval\_code, response\_msg FROM payment\_authorization ORDER BY code ASC;

CC

Write a query to create a table named  **'address'** .

CREATE TABLE address

(id int NOT NULL,

work\_phone varchar2(50) NOT NULL,

home\_phone varchar2(50) NOT NULL,

fax\_phone varchar2(50),

email varchar2(50),

street varchar2(50) NOT NULL,

city varchar2(50) NOT NULL,

state varchar2(50) NOT NULL,

zip\_code int NOT NULL,

country varchar2(50) NOT NULL,

CONSTRAINT address\_pk PRIMARY KEY (id)

);

Write a query to create a table named  **'package\_type'** .

CREATE TABLE package\_type

(id int NOT NULL,

description varchar2(30),

package\_length NUMBER (\*,2),

width NUMBER (\*,2),

height NUMBER (\*,2),

weight NUMBER (\*,2),

volume NUMBER (\*,2),

CONSTRAINT package\_type\_pk PRIMARY KEY (id)

);

Write a query to change the column name '**description**' to '**notes**' in the package\_type table.

ALTER TABLE package\_type

RENAME COLUMN description TO notes;

Write a query to change the column name ' **email**' to ' **emailid**' in the address table.

ALTER TABLE address

RENAME COLUMN email TO emailid;

Write a query to create a table named **'shipment\_status'.**

CREATE TABLE shipment\_status

(id int NOT NULL,

code varchar2(30),

status varchar2(100),

description varchar2(100),

CONSTRAINT shipment\_status\_pk PRIMARY KEY (id)

);

Write a query to display id, street, email, home\_phone and zip code details of the city **'Chennai'** from the address table. Display the records sorted in ascending order based on the id.

SELECT id, street, email, home\_phone, zip\_code FROM address WHERE city = 'Chennai'

ORDER BY id ASC;

Write a query to display id, street, city, email and home\_phone details of the country **'USA'** from the address table. Display the records sorted in ascending order based on the id.

SELECT id, street, city, email, home\_phone FROM address WHERE country = ‘USA’

ORDER BY id ASC;

Write a query to display all the company details. Display the records sorted in ascending order based on the name.

SELECT \* FROM company ORDER BY name ASC;

Write a query to display id and description of the rate details having minimum rate in the range between 50 and  250. Display the records sorted in ascending order based on the id.

SELECT id, description FROM rate WHERE rate\_min BETWEEN 50 and 250 ORDER BY id ASC;

Write a query to display the credit\_card details having cc\_type as **'Visa'**.Display the records in ascending order based on the name.

Select \* from credit\_card where cc\_type = ‘Visa’ order by name ASC;

Write a query to display all the cheque details that belongs to the bank **'ICICI Bank'**. Display the records in ascending order based on the cheque\_date.

Select \* from cheque where bank\_name = ‘ICICI Bank’ order by cheque\_date ASC;

Write a query to display the id,attempt,status and amount of all the payments having payment status as **'Pending'**.Display the records in ascending order based on their id.

Select id, attempt, status, amount from payment where status = ‘Pending’ order by id ASC;

Write a query to display the id,due\_date,total and balance of all the invoices in **'Completed'**status. Display the records in ascending order based on their id.

Select id, due\_date, total, balance from invoice where status = ‘Completed’ order by id ASC;

Write a query to display all the user details with **2 or more failure attempts** (i.e >=2 failure attempts). Display the records in ascending order based on their id **.**

Select \* from user\_details where failed\_attempts >= 2 order by id ASC;

Write a query to display the id,due\_date,total,balance of all the invoices which were created between the  month of **'** **March** **'** and **'April'**. Display the records in ascending order based on their id.

Select id, due\_date, total, balance from invoice where to\_char(created\_time, 'MON') in ('MAR','APR') order by id ASC;

Write a query to display the org\_name,first\_name,email and city of all the contacts that belongs to the state **'Karnataka'**  or '**TamilNadu**'. Display the records in ascending order based on their org\_name and then by first\_name.

select org\_name, first\_name, email, city from contact where state in ('Karnataka','TamilNadu') order by org\_name asc, first\_name asc;

Write a query to display the username,user\_password of all the users with atleast  **one failure** **attempt**.Display the records in descending order based on their username.

select username, user\_password from user\_details where failed\_attempts > '0' order by username desc;

Write a query to display the name and internal number of all the items that has price greater than **'20000'**.Display the record in ascending order based on their name.

select name, internal\_number from items where price > 20000

Write a query to display the customer id and note of all the customers those who have their credit limit more than **'20000'.**Display the records in ascending order based on their id.

select id, notes from customer where credit\_limit > '20000' order by id asc;

Write a query to display the bill\_type,active\_since,active\_until of all the purchase\_orders in **'completed'**status.Display the records in ascending order based on their active\_since value.

select bill\_type, active\_since, active\_until from purchase\_order where status = 'Completed' order by active\_since asc;

Write a query to display shipment\_entity id, name and identification number of all the shipment\_entity that have credit\_limit greater than the credit\_limit of shipment where name as **'Air Conditioner'**. Display the records sorted in ascending order based on id.   
**Use: ANY**

select s1.id, s1.name, s1.identification\_number from shipment\_entity s1

where s1.credit\_limit > any

(select credit\_limit from shipment\_entity se

join shipment s on (s.shipment\_entity\_id=se.id) where se.name='Air Conditioner')

Write a query to display the name of all the ports that belong to  **'India'.** Display the records in ascending order based on their name.

Select name from port where name = ‘India’ order by name asc;

Write a query to display the user details like  firstname, lastname and contact\_number   in which the role  name is  **'Admin'**  or  **'Customer'.**Display the records  in ascending order  based on firstname.

select first\_name, last\_name, contact\_number from user\_details where role\_name = 'Admin' or role\_name = 'Customer' order by first\_name asc;

Write a query to display cargo id, description, cargo\_length, width, height, weight and volume of the cargo with volume greater than 200, 300 and 350. Display the records sorted in ascending order based on cargo id.   
**Use: ALL**

select id, description, cargo\_length, width, height, weight, volume from cargo where volume > all ('200','300','350') order by id asc;

Write a query to display the user id, first\_name, contact\_number and role\_name of the user whose first\_name starts with either ' **S**' or ' **A'.** Display the records sorted in ascending order based on id.  
**Use: Some**

select id, first\_name, contact\_number, role\_name from user\_details where first\_name like 'S%' or first\_name like 'A%' order by id asc;

Write a query to display all the user details which were created in the year 2015. Display the records in ascending order based on their id.

select \* from user\_details where to\_char(created\_time, 'YY') in ('15') order by id asc;

Write a query to display the unique bank\_name of all the cheques received. Display the records in ascending order based on their bank\_name.

select distinct bank\_name from cheque order by bank\_name asc;

Write a query to display the maximum credit limit of a customer. Give an alias name for maximum credit limit as **'max\_limit'**.

select credit\_limit as max\_limit from customer where credit\_limit = (select max(credit\_limit) from customer)

Write a query to display the minimum total amount of the invoice raised. Give an alias name for minimum total amount as **'min\_total'**.

select total as min\_total from invoice where total = (select min(total) from invoice)

select sum(balance) as Total\_Balance from invoice;

Write a query to display the total number of customers. Give an alias name for number of customers as **'customer\_count'**.

select count(id) as Customer\_Count from customer;

Write a query to display the average amount of the payment that were received. Give an alias name for average amount as **'average\_payment'**.

select avg(amount) as Average\_payment from payment;

Write a query to display all the details of  purchase orders which were billed in the year 2016. Display the records in ascending order based on their id.

select \* from purchase\_order where to\_char(created\_time, 'YY') in ('16') order by id asc;

Write a query to display the number of users  who have not logged in till date. Give the alias name as **'Inactive\_Users'.**

Write a query to display the number of purchase orders with status **'Completed'.** Give alias name as**'Completed\_Orders'.**

select count(status) as Completed\_Orders from purchase\_order where status = 'Completed';

Write a query to display the paper\_invoice\_batch\_number, due\_date, last\_reminder and balance of the invoice details and if the balance is greater than or equal to  **'25000'** then display it as **'Overdue'** or else**'Nil'** with an alias name as **'due\_status'** of all the invoice. Display the records in descending order based on their balance.

select paper\_invoice\_batch\_number, due\_date, last\_reminder,

balance,

case when balance > = '25000' then 'Overdue'

else 'Nil'

end as due\_status

from invoice order by balance desc;

Write a query to display first\_name, last\_name, phone\_number, email of the contact details and If there is a null value in last\_name then replace it with 'first\_name' and give an alias name as **'updated\_last\_name'**. Display the records in descending order based on their first\_name.

select first\_name, last\_name, phone\_number, email,

case when last\_name is NULL then first\_name

else last\_name

end as updated\_last\_name

from contact order by first\_name desc;

Write a query to display the id, due\_date, total and balance of all the invoices which were raised in the 1st quarter of 2015. Display the records in descending order based on their balance.

select id, due\_date, total, balance from invoice where To\_char(created\_time, 'Q') = 1 and to\_char(created\_time, 'YY') in ('15') ;

Write a query to display the invoice\_id and amount of all the payments which were received between Jan 01 2015 and Aug 31 2015. Display the records in descending order based on their amount.

select invoice\_id, amount from payment

where status = 'Completed' and created\_time between '01-Jan-15' AND '31-Aug-15' order by amount desc;

Write a query to display the count of shipment status. Give an alias name as **'Shipment\_status\_count'**.

Select count(status) as Shipment\_status\_count from shipment\_status;

Write a query to display notes of the commodity that has the minimum total value.In case of multiple records, display the records in ascending order based on the notes.

select notes from commodity where total\_value in

(select MIN(total\_value) from commodity)

order by notes asc;

Write a query to display the sum of total\_profit of the shipments that are being shipped in the month of '**March**'.Give an alias name for total\_profit as '**total\_profit**'.

select sum(total\_profit) as total\_profit from shipment where to\_char(departure\_time, 'MON') in 'MAR';

Write a query to display the description,package\_length,width and height of the package type that has the maximum volume.

select description, package\_length, width, height from package\_type where volume in

(select MAX(volume) from package\_type);

Write a query to display the year give an alias name as **'year\_name'** and total\_profit gained in each year of all the shipments and give alias name as **'total\_profit\_by\_year'**. Display the records sorted in ascending order based on year\_name.

select extract(Year from DEPARTURE\_TIME) as Year\_Name, sum(total\_profit) as total\_profit\_by\_year

from shipment

group by extract(Year from DEPARTURE\_TIME)

order by Year\_Name asc;

Write a query to display the role\_name and the number of users under each role. Give an alias name as ' **user\_count**'.

select Role\_name, count(Role\_name) as user\_count from user\_details

group by Role\_name;

Write a query to display the commodity id, total\_weight and total\_volume of the commodity having maximum quantity.

select id, total\_weight, total\_volume from commodity where quantity in

(select MAX(quantity) from commodity);

Write a query to display all the customer details which were created by the user with username **'arun'**. Display the records in ascending order based on their id.

select a.\* from customer a

inner join user\_details b

On a.CREATED\_USER\_ID = b.id

where b.username = 'arun' order by a.id asc;

Write a query to display org\_name, street\_address1, street\_address2, city, state, phone\_number and email of all the customers with credit limit more than 25000. Display the records in ascending order based on their org\_name.   
**HINT: Use Subquery**

select org\_name, street\_address1, street\_address2, city, state, phone\_number, email from contact a

where (select credit\_limit from customer b where a.id=b.contact\_id) >'25000' order by a.org\_name asc;

Write a query to display the first\_name, phone\_number and email of all the users who have not logged in till date. Display the records in ascending order based on their first\_name.

select a.first\_name, a.phone\_number, a.email from contact a

join user\_details b

on a.id=b.contact\_id where b.lastlogin\_time is null order by a.first\_name asc;

Write a query to display the invoice\_id , amount and status of all the payments which were received in the transaction type mode as **'Credit Card'**. Display the records in ascending order based on the amount and then in ascending order based on the  invoice id.

select invoice\_id, amount, status from payment p

join TRANSACTION\_TYPE t on

p.TRANSACTION\_TYPE\_ID = t.id where t.type = 'Credit Card'

order by p.amount asc, p.invoice\_id asc;

Write a query to display the invoice\_id, amount and status of all the payment which were received in the transaction type as**'Cheque'**. Display the records in descending order based on the amount and then in ascending order based on invoice id.

select invoice\_id, amount, status from payment p

join TRANSACTION\_TYPE t on

p.TRANSACTION\_TYPE\_ID = t.id where t.type = 'Cheque'

order by p.amount desc, p.invoice\_id asc;

Write a query to display the paper\_invoice\_batch\_number,due\_date,last\_reminder and balance of the customers whose org\_name is **'Cognizant'**. Display the records in descending order based on their balance and then in ascending order based on the last\_reminder.

select i1.paper\_invoice\_batch\_number, i1.due\_date, i1.last\_reminder, i1.balance from invoice i1

join customer c

on i1.customer\_id=c.id

join contact c1

on c1.id=c.contact\_id

where c1.org\_name = 'Cognizant'

order by balance desc, last\_reminder asc;

Write a query to display the bill\_type, active\_since, active\_until, status of all the purchase\_orders which were created by the user whose first name is **'Arun'**. Display the records in ascending order based on their bill\_type.

select bill\_type, active\_since, active\_until, status from purchase\_order po

join user\_details ud

on po.created\_user\_id = ud.id

join contact c

on c.id = ud.contact\_id

where c.first\_name = 'Arun'

order by bill\_type asc;

Write a query to display the invoice\_id, amount and status of all the payments which were made by the customers whose org\_name is **'ABC Tech Park'**. Display the records in descending order based on the amount.

select p.invoice\_id, p.amount, p.status from payment p

join invoice i

on p.invoice\_id = i.id

join customer c

on i.customer\_id = c.id

join contact c1

on c1.id = c.contact\_id

where c1.org\_name = 'ABC Tech Park'

order by p.amount desc;

Write a query to display customer's org\_name, first\_name, phone\_number, email, customer notes and credit\_limit for all the customers. Display the records in ascending order based on org\_name and then by first\_name.

select c1.org\_name, c1.first\_name, c1.phone\_number, c1.email, c2.notes, c2.credit\_limit from contact c1

join customer c2

on c2.contact\_id = c1.id

order by c1.org\_name asc, c1.first\_name asc;

Write a query to display the internal\_number and number of orders received for each item with alias name as **'number\_of\_orders'**. Display the records in ascending order based on internal number.

select i1.internal\_number, count(po.id) as number\_of\_orders from item i1

join order\_line ol

on ol.item\_id = i1.id

join purchase\_order po

on ol.order\_id = po.id

group by i1.internal\_number

order by i1.internal\_number asc;

Write a query to display the org\_name, invoice total, sum of all payment amount and balance of the invoice in which payment status is **'Completed'**. Give the alias for total payment received as **'Amount\_Paid'**. Display the records in descending order based on  the total amount .

select c1.org\_name, i.total, sum(p.amount) as amount\_paid, i.balance from contact c1

join customer c2

on c2.contact\_id = c1.id

join invoice i

on i.customer\_id = c2.id

join payment p

on p.invoice\_id = i.id

where p.status = 'Completed'

group by c1.org\_name, i.total, i.balance order by total desc;

Write a query to display the sum of the total amount received through the credit card and cheque. Give the alias names as **'total\_credit'** and **'total\_cheque'** respectively.

SELECT MAX(SUM (CASE WHEN tt.TYPE = 'Credit Card' THEN pt.amount END)) AS total\_credit,

MAX(SUM (CASE WHEN tt.TYPE = 'Cheque' THEN pt.amount END)) AS total\_cheque

FROM payment pt JOIN transaction\_type tt ON pt.transaction\_type\_id = tt.id

GROUP BY tt.TYPE

Write a query to display rate id, origin port name, destination port name, rate\_min and rate\_max for all the ports. Give alias name for origin port name as **'source\_port'** and destination port name as**'dest\_port'**. Display the records sorted in ascending order based on the rate\_max.

select r.id, (case when p.id = r.origin\_id then p.name end) as source\_port,

(case when p.id = r.destination\_id then p.name end) as dest\_port,

r.rate\_min, r.rate\_max from rate r

join port p

on p.id = r.origin\_id

on p.id = r.destination\_id

order by rate\_max asc;

select a.id, a.origin, b.dest, a.rate\_min, a.rate\_max from

(select p1.id as id, r1.origin\_id as origin, r1.rate\_min, r1.rate\_max from rate r1

join port p1

on p1.id = r1.origin\_id) a

join

(select p2.id as id, r2.destination\_id as dest from rate r2

join port p2

on p2.id = r2.destination\_id) b

on a.id=b.id

Write a query to display description, cargo\_length, width and weight of the cargo along with the shipment name, booking\_number and total\_profit in which cargo volume is greater than **'300'**. Display the records sorted in descending order based on the total\_profit for the shipment.

Write a query to display id, iata and carrier\_code of the carriers from the country **'India'**. Display the records sorted in ascending order based on iata.

select c.id, c.iata, c.carrier\_code from carrier c

join address a

on c.address\_id = a.id

where country = 'India'

order by c.iata asc;

Write a query to display the shipment\_entity details that are being shipped to the port ' **Chennai**' or ' **Cochin**'. Display the records sorted in ascending order based on the  id.

select se.\* from shipment\_entity se

join shipment s

on se.id = s.shipment\_entity\_id

join port p

on p.id = s.arrival\_port\_id

where p.name in ('Chennai', 'Cochin')

order by se.id asc;

Write a query to display the shipment\_entity details that are being shipped to '**Mumbai**' port. Display the records sorted in ascending order based on the id.

select se.\* from shipment\_entity se

join shipment s

on se.id=s.shipment\_entity\_id

join port p

on s.arrival\_port\_id = p.id

where p.name = 'Mumbai'

order by se.id asc;

Write a query to display the agent details from the state '**Karnataka**' or '**TamilNadu**'. Display the records sorted in ascending order based on iata.

select a.\* from agent a

join address a1

on a.address\_id = a1.id

where a1.state in ('Karnataka', 'TamilNadu')

order by a.iata asc;

Write a query to display description, cargo\_length, width and weight of the cargo along with the shipment name, booking\_number and total\_profit in which cargo volume is greater than **'300'**. Display the records sorted in descending order based on the total\_profit for the shipment.

select c.description, c.cargo\_length, c.width, c.weight, s.name, s.booking\_number, s.total\_profit from cargo c

join shipment\_cargo sc

on c.id = sc.cargo\_id

join shipment s

on sc.shipment\_id=s.id

where c.volume > '300'

order by s.total\_profit desc;

Write a query to display the method name and count of rate under each method. Give an alias name for the count of rate as**'rate\_count'**. Display the records sorted in ascending order based on the method name.

select m.name, count(r.id) as rate\_count from method m

join rate r

on r.method\_id = m.id

group by m.name

order by m.name asc;

set operator and DDL

Write a query to display payment id, amount and status of all the payments made by **'Credit Card'** and **'Cheque'**.  Display the details of the **Credit Card**first  ordered by id and then **Cheque**ordered by id.

select p.id, p.amount, p.status from payment p

join transaction\_type tt

on p.transaction\_type\_id = tt.id

where tt.type = 'Credit Card'

union

select p.id, p.amount, p.status from payment p

join transaction\_type tt

on p.transaction\_type\_id = tt.id

where tt.type = 'Cheque'

order by id asc;

 Write a query to display the number of purchase orders created by the user **'Nishasree'** and  then the number of orders created by the user **'Shivani'**. Give an alias name as **'Order\_count'.**  
**Note: Use union**

select count(po.id) as order\_count from purchase\_order po

join user\_details ud

on po.created\_user\_id = ud.id

where ud.username = 'Nishasree'

union

select count(po.id) from purchase\_order po

join user\_details ud

on po.created\_user\_id = ud.id

where ud.username = 'Shivani'

Write a query to display the item ids and item names of all the items whose purchase orders were processed and invoiced for the customer with org\_name '**ABC Tech Park'** and '**Cognizant'**.  
Display the id and name of the items invoiced for the customer **'Cognizant**' first and then from **'ABC Tech Park'**.  
**Note: Use union all**

select i1.id, i1.name from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'Cognizant'

union all

select i1.id, i1.name from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'ABC Tech Park'

order by id asc;

Write a query to display item id and item name of all the items  whose purchase order created by org\_name **'ABC Tech Park'** and **'Cognizant'**. If there are any duplicate items remove it and display the id and name of the items created by **'Cognizant'** first and then by **'ABC Tech Park'.**  
**Note: Use union**

select i1.id, i1.name from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'Cognizant'

union

select i1.id, i1.name from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'ABC Tech Park'

order by id asc;

Write a query to display item id, item name and quantity of all the items purchased more than 10 in quantity and whose purchase orders processed and invoiced for the customer "**Cognizant**" . Do not consider items in this list that has been purchased by Customer "**ABC Tech Park**". Display the items ordered by item id.  
**Note:Use minus**

select i1.id, i1.name, il.quantity from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'Cognizant' and il.quantity > '10'

minus

select i1.id, i1.name, il.quantity from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'ABC Tech Park' and il.quantity > '10';

Write a query to display item id and item name of all the items ordered whose purchase order created by org\_name **'ABC Tech Park'** and **'Cognizant'**. Display only the common items ordered by **'Cognizant'** and **'ABC Tech Park'**.  
**Note:Use intersect**

select i1.id, i1.name from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'Cognizant'

intersect

select i1.id, i1.name from item i1

join invoice\_line il

on il.item\_id = i1.id

join invoice i2

on il.invoice\_id = i2.id

join customer c2

on i2.customer\_id = c2.id

join contact c3

on c2.contact\_id = c3.id

where c3.org\_name = 'ABC Tech Park';

Write a query to display invoice\_id, order\_id and number of items ordered in each order\_id of all invoices whose status is **'Completed'** and **'Pending'**. Give an alias name for number of items ordered as**'Item\_count'**. Display the records sorted in ascending order based on invoice\_id of those invoice status **'Completed'** first and then status **'Pending**'.  
**Note: Use union**

select i1.id, po.id, count(i2.id) as Item\_count from invoice i1

join order\_process op

on i1.id = op.invoice\_id

join purchase\_order po

on po.id = op.order\_id

join order\_line ol1

on po.id = ol1.order\_id

join item i2

on i2.id = ol1.item\_id

where i1.status = 'Completed'

group by i1.id, po.id

union

select i1.id, po.id, count(i2.id) as Item\_count from invoice i1

join order\_process op

on i1.id = op.invoice\_id

join purchase\_order po

on po.id = op.order\_id

join order\_line ol1

on po.id = ol1.order\_id

join item i2

on i2.id = ol1.item\_id

where i1.status = 'Pending'

group by i1.id, po.id

order by 1 asc;

Write a query to display the port name where maximum number of shipments takes place and then display the port name where minimum number of shipment takes place.   
**Use: Union**   
**Consider departure\_port\_id**

select p1.name from port p1

join shipment s1

on s1.departure\_port\_id = p1.id

where departure\_port\_id in

(select departure\_port\_id from (

select departure\_port\_id, count(\*) from shipment group by departure\_port\_id order by 2 desc) test

where rownum = 1)

union

select p1.name from port p1

join shipment s1

on s1.departure\_port\_id = p1.id

where departure\_port\_id in

(select departure\_port\_id from (

select departure\_port\_id, count(\*) as count from shipment group by departure\_port\_id order by 2 asc) test

where rownum = 1)

--------------------------------------------------------

select \* from invoice where payment\_attempts in

(select payment\_attempts from (

select payment\_attempts, count(\*) from invoice group by payment\_attempts order by 2 desc) test

where rownum = 1)

--------------------------------------------------------

Write a query to display shipment name, total\_weight, total\_volume and total\_profit of the shipments for which cargo package type is 'Pallet' and 'Carton'. Display only the common records sorted in ascending order based on their shipment name.  
**Use: Intersect**

select name, total\_weight, total\_volume, total\_profit from shipment s1

join shipment\_cargo sc

on sc.shipment\_id = s1.id

join cargo c1

on c1.id = sc.cargo\_id

join package\_type pt

on pt.id = c1.package\_type\_id

where pt.description = 'Pallet'

intersect

select name, total\_weight, total\_volume, total\_profit from shipment s1

join shipment\_cargo sc

on sc.shipment\_id = s1.id

join cargo c1

on c1.id = sc.cargo\_id

join package\_type pt

on pt.id = c1.package\_type\_id

where pt.description = 'Carton'

order by name asc;

Write a query to display shipment\_entity id and name that are being shipped to the port 'Mumbai' by using the method 'DHL' and 'TNT'. Display only the common records sorted in ascending order based on shipment\_entity id.  
**Use: Intersect**

select se.id, se.name from shipment\_entity se

join shipment s

on s.shipment\_entity\_id = se.id

join port p

on p.id = s.departure\_port\_id

join method m

on s.method\_id = m.id

where p.name = 'Mumbai' and m.name = 'DHL'

intersect

select se.id, se.name from shipment\_entity se

join shipment s

on s.shipment\_entity\_id = se.id

join port p

on p.id = s.departure\_port\_id

join method m

on s.method\_id = m.id

where p.name = 'Mumbai' and m.name = 'TNT'

order by id asc;

Write a query to display commodity  id, model, serial and invoice\_number of the commodities for which cargo volume is more than 300 and which are priced more than 18000. Display all the records sorted in ascending order based on commodity id.   
**Use: Union All**

select c1.id, ci.model, ci.serial, ci.invoice\_number from commodity c1

join commodity\_identification ci

on ci.id = c1.commodity\_identification\_id

join cargo c2

on c1.cargo\_id = c2.id

where c2.volume > '300'

union all

select c1.id, ci.model, ci.serial, ci.invoice\_number from commodity c1

join commodity\_identification ci

on ci.id = c1.commodity\_identification\_id

join rate r1

on ci.rate\_id = r1.id

where r1.charge > '18000'

order by id asc;

Write a query to display commodity details like commodity id, model, serial and invoice\_number of the commodities for which cargo volume is more than 300 and do not display details of those commodities which charges below 18000. Display all the records sorted in ascending order based on commodity id.  
**Use: minus**

select c1.id, ci.model, ci.serial, ci.invoice\_number from commodity c1

join commodity\_identification ci

on ci.id = c1.commodity\_identification\_id

join cargo c2

on c1.cargo\_id = c2.id

where c2.volume > '300'

minus

select c1.id, ci.model, ci.serial, ci.invoice\_number from commodity c1

join commodity\_identification ci

on ci.id = c1.commodity\_identification\_id

join rate r1

on ci.rate\_id = r1.id

where r1.charge < '18000'

order by id asc;

**view, trigger, synonym**

Create a synonym with the name **'client'** for the table customer.

create synonym client

for customer;

Create a synonym with the name '**product'** for the table item.

create synonym product for item;

Create a view named**'transaction\_type\_details'** to display all the details of transaction\_type.

create view transaction\_type\_details as

select \* from transaction\_type;

Create a view named **'invoice\_details'** to display all the invoice details having zero balance.

create view invoice\_details as

select \* from invoice where balance = '0';

Create a view named **'payment\_details'** to display all the payment details that are received as  **'Credit Card'**payment**.**

create view payment\_details as

select p.\* from payment p

join transaction\_type tt

on p.transaction\_type\_id = tt.id

where tt.type = 'Credit Card';

Create a trigger named **'trigger\_contact\_af\_update'** that is triggered whenever the contact table is updated. This trigger will insert the org\_name and action into the table **'contact\_log\_history'**after the updation of contact details. The action name in the affected log table contact\_log\_history is **'After\_Update\_Contact'.**

CREATE OR REPLACE TRIGGER trigger\_contact\_af\_update

AFTER UPDATE

ON contact

FOR EACH ROW

BEGIN

insert into contact\_log\_history (org\_name, action)

values (:new.org\_name, 'After\_Update\_Contact');

END;

/

Create a trigger named **'trigger\_credit\_bf\_update'** that is triggered whenever the credit\_card table is updated. This trigger will insert the cc\_type and action into the table **'credit\_card\_log\_history'** before the updation of credit\_card details. The action name in the affected log table credit\_card\_log\_history is **'Before\_Update\_Credit\_Card'.**

CREATE OR REPLACE TRIGGER trigger\_credit\_bf\_update

BEFORE UPDATE

ON credit\_card

FOR EACH ROW

BEGIN

insert into credit\_card\_log\_history (cc\_type, action)

values (:old.cc\_type, 'Before\_Update\_Credit\_Card');

END;

/

Create a trigger named**'transaction\_type\_af\_insert'** that is triggered whenever a new record is inserted into transaction\_type table. This trigger will insert the new type and action into the table**'transaction\_type\_log\_history'** after the insertion of transaction type details. The action name in the affected log table transaction\_type\_log\_history is **'After\_Insert\_transaction\_type'.**

create trigger transaction\_type\_af\_insert

after insert

on transaction\_type

for each row

begin

insert into transaction\_type\_log\_history (type, action)

values

(:new.type, 'After\_Insert\_transaction\_type');

end;

/

Create a view named **'shipment\_details'**to display all the shipment details with package\_type description as '**Carton'.**

create view shipment\_details as

select s1.\* from shipment s1

join shipment\_cargo sc

on s1.id = sc.shipment\_id

join cargo c

on c.id = sc.cargo\_id

join package\_type pt

on c.package\_type\_id = pt.id

where pt.description = 'Carton'

Create a trigger named **'trigger\_port\_bf\_update'**that is triggered whenever the port table is updated. This trigger will insert the country and action into the table **'before\_port\_log'** after the updation of port details. The action name in the affected log table before\_port\_log is**'After\_Update\_Port'.**

CREATE OR REPLACE TRIGGER trigger\_port\_bf\_update

BEFORE UPDATE

ON port

FOR EACH ROW

BEGIN

insert into before\_port\_log (country, action)

values (:old.country, 'After\_Update\_Port');

END;

/

strings and regular expressions

Write a query to display the name by combining first\_name and last\_name separated with a space, city, state, phone\_number and email of all Contact details which  belongs to the state **'Karnataka'**. Give an alias name for name as **'full\_name'**. Display the records in ascending order based on their **'city'** and then by **'full\_name'.**  
**Note: use CONCAT().**

select concat(concat(first\_name,' '), last\_name) as full\_name, city, state, phone\_number, email from contact where state = 'Karnataka'

order by city asc, full\_name asc;

Write a query to display the customer id and customer notes of the customers which contains a word '**Recruit**' in their customer notes. Display the records in ascending order based on their customer id.

select id, notes from customer where notes like '%Recruit%' order by id asc;

select username, RAWTOHEX(user\_password) as encrypted\_password, lastlogin\_time, failed\_attempts from user\_details

order by username asc;

select username, RAWTOHEX(user\_password) as encrypted\_password, lastlogin\_time, failed\_attempts from user\_details

where length(RAWTOHEX(user\_password)) > '15'

order by username asc;

*select* rpad(card\_number,20, '#') as modified\_number from credit\_card order by card\_number asc;

select lpad(transaction\_id, 30, '#') as modified\_id from payment\_authorization order by transaction\_id asc;

select replace(username, 'a', '\*') as modified\_name from user\_details order by modified\_name asc;

select substr(code, 1, 2) as modified from currency order by modified asc;

select first\_name, city, state, regexp\_replace(phone\_number, '([[:digit:]]{3})([[:digit:]]{3})([[:digit:]]{4})' , '\1-\2-\3') as Phone\_number

from contact where length(phone\_number) = '10' order by first\_name asc;

select replace(city, 'Bangalore', 'Bengaluru') as city\_name from contact

order by city\_name asc;

select email, regexp\_replace(email, '[^,]yahoo.com', '@yahoo.co.in') as modified\_email from contact

order by email asc;

select first\_name, city, state, phone\_number from contact where substr(regexp\_substr(phone\_number, '^.{4}[5]'),5,1) = '5'

order by first\_name asc;

select first\_name from contact where length(first\_name) <= '5'

order by first\_name asc;

select concat(concat(c.notes, '-'), i1.status) as Customer\_Invoice\_Status from customer c

join invoice i1

on i1.customer\_id = c.id

order by Customer\_Invoice\_Status asc;

Write a query to display the company name and last 2 characters of iata and give an alias name as **'iata\_number'** of all the companies. Display the records sorted in ascending order based on name.

select name, substr(iata, -2) as iata\_number from company order by name asc;

Write a query to display the first three characters of unique state name from address details. Give an alias name as**'state\_code'**. Display the records in ascending order based on state\_code.

**Note: use substr().**

select distinct substr(state, 1, 3) as state\_code from address order by state\_code asc;

Analytic function:

Write a query to display all payment id, invoice\_id, amount and average amount of the payments within the invoice\_id. Give alias name as **'average\_payment'**. Display the records in ascending order based on the invoice\_id.

**Hints:**   
**Analytic Function:** avg()   
**Analytic Clause:** query\_partition\_clause

select p.id, i.id as invoice\_id, p.amount, avg(p.amount) over (partition by i.id) as Average\_payment from payment p

join invoice i

on i.id = p.invoice\_id

order by i.id asc;

Write a query to display all payment id, invoice\_id, amount and running average amount of all payments within the invoice\_id. Give alias name as '**average\_payment**'. Display the records in ascending order based on the invoice\_id.  
  
**Hints:**  
**Analytic Function:** avg()  
**Analytic Clause:** query\_partition\_clause and order\_by\_clause.

select p.id, i.id as invoice\_id, p.amount, avg(p.amount) over (partition by i.id order by p.amount rows UNBOUNDED PRECEDING) as Average\_payment from payment p

join invoice i

on i.id = p.invoice\_id

order by i.id asc;

Write a query to display the payment id, invoice\_id, amount and first payment amount of each invoice\_id. Give alias name as **'first\_payment'**. Display the records in ascending order based on the invoice\_id.  
  
**Hints:  
Analytic Function:** FIRST\_VALUE( )  
**Analytic Clause:** query\_partition\_clause and order\_by\_clause

select p.id, i.id as invoice\_id, p.amount, first\_value(p.amount) over (partition by i.id order by p.amount rows UNBOUNDED PRECEDING) as first\_payment from payment p

join invoice i

on i.id = p.invoice\_id

order by i.id asc;

Write a query to display all invoice\_id, amount and cumulative distribution of the amount received for the invoice\_id. Give an alias name as **'cume\_dist\_payment'**. Display cumulative distribution correct to 2 decimal places using **round()**. Display the records in ascending order based on the amount.  
  
**Hints:  
Analytic Function:** CUME\_DIST()  
**Analytic Clause:** query\_partition\_clause and order\_by\_clause

select i.id as invoice\_id, p.amount, round(cume\_dist() over (partition by i.id order by p.amount)\*100, 2) as cume\_dist\_payment from payment p

join invoice i

on i.id = p.invoice\_id

order by i.id asc;

Write a query to display all customer\_id, sum of total with alias name as **'customer\_total'** and rank of each customer according to their total with alias name as **'Rank'**. Display the records in ascending order based on the Rank.  
  
**Hints:  
Analytic Function:** RANK()  
**Analytic Clause:** order\_by\_clause

select c.id as customer\_id, sum(i.total) as customer\_total, rank() over(order by sum(i.total) desc) as rank from invoice i

join customer c

on i.customer\_id = c.id

group by c.id

Write a query to display all  invoice\_id, amount and maximum payment amount of each invoice\_id. Give alias name as**'max\_amount'**. Display the records in ascending order based on the invoice\_id and then by amount.  
  
**Hints:  
Analytic Function:** MAX( )  
**Analytic Clause:** query\_partition\_clause and order\_by\_clause

select i.id as invoice\_id, p.amount, max(p.amount) over(partition by i.id) as max\_amount from payment p

join invoice i

on i.id = p.invoice\_id

order by i.id asc, p.amount asc;

Write a query to display all customer\_id, total and minimum total of each customer with alias name as **'min\_amount'**. Display the records in ascending order based on the customer\_id and then by total.  
  
**Hints:  
Analytic Function:** MIN( )  
**Analytic Clause:** query\_partition\_clause and order\_by\_clause

select c.id as customer\_id, i.total, min(i.total) over (partition by c.id order by c.id, total asc) as min\_amount from invoice i

join customer c

on c.id = i.customer\_id

Write a query to display the  customer\_id, created\_time, total, sum of total by all invoice with alias name as 'Total invoice Amount', sum of total by each customer invoice with alias name as **'Total invoice By Customer'** and sum of total by each date with alias name as **'Total Amount by date'.**Display the records sorted in ascending order by customer\_id and then by created\_time  
  
**Hints:  
Analytic Function:** SUM( )  
**Analytic Clause:** query\_partition\_clause and order\_by\_clause

select c.id as customer\_id, i.created\_time, i.total, (select sum(total) from invoice) as "Total invoice Amount",

sum(i.total) over (partition by c.id) as "Total invoice By Customer",

sum(i.total) over (partition by i.created\_time) as "Total Amount by date"

from invoice i

join customer c

on i.customer\_id = c.id

order by c.id, created\_time asc;

Write a query to display all customer\_id, total and running serial number to each customer with alias name as **'SNO'.** Display the records in ascending order based on the customer\_id and then by SNO.  
  
**Hints:  
Analytic Function:** ROW\_NUMBER( )  
**Analytic Clause:** query\_partition\_clause and order\_by\_clause

select c.id as customer\_id, total, row\_number() over (partition by customer\_id order by rownum) as SNO from customer c

join invoice i

on i.customer\_id = c.id

Write a query to display all the id , amount received and next amount that is received after every payment and give an alias name as**'Next payment'**  with status as '**Completed**'. Display the records in ascending order based on id.

select id, amount, lead(amount,1) over (order by id) as "Next payment" from payment

where status = 'Completed'

Write a query to display all port name, total\_weight and minimum total\_weight of each port. Give alias name as **'port\_min\_weight'.**Display the records in ascending order based on port name.

select p.name, s.total\_weight, min(total\_weight) over (partition by p.name) as port\_min\_weight from shipment s

join port p

on s.departure\_port\_id = p.id

order by name asc;

Write a query to display all destination port name , charge and running average charge of each ports. Give alias name as **'average\_charge\_to'.**Display the records in ascending order based on the port name .

|  |
| --- |
| select p.name, r.charge, avg(charge) over (partition by p.name orderby r.charge, p.name rows UNBOUNDED PRECEDING) as average\_charge\_to from rate r |

|  |  |
| --- | --- |
|  | join port p |

|  |  |
| --- | --- |
|  | on p.id = r.destination\_id |

Write a query to display all port name, departure\_time, total\_volume, sum of total\_volume of all shipment with alias name as **'Total\_sum\_volume',** average total\_volume of each ports with alias name as **'Average\_volume\_By\_port',** sum of total\_volume on each date as **'Total\_volume\_by\_date'.** Display the records in ascending order based on the  port name.

select p.name, s.departure\_time, total\_volume,

(select sum(total\_volume) from shipment) as Total\_sum\_volume,

avg(total\_volume) over (partition by p.name) as Average\_volume\_By\_port,

sum(total\_volume) over (partition by s.departure\_time) as Total\_volume\_by\_date

from shipment s

join port p

on s.departure\_port\_id = p.id

order by p.name

Write a query to display the port name, sum of total\_profit with alias name as**'port\_profit'**and rank of all the ports according to the sum of total\_profit. Give alias name as **'RANK '**. Display the records in ascending order based on the port name.

select p.name, sum(total\_profit) as port\_profit, rank() over(order by sum(total\_profit) desc) as rank from port p

join shipment s

on p.id = s.departure\_port\_id

group by p.name

Write a query to display all port name, total\_volume and maximum total\_volume of each port. Give alias name as**'port\_max\_volume'.**Display the records in ascending order based on port\_name.

select p.name, total\_volume, max(total\_volume) over(partition by p.name) as port\_max\_volume from shipment s

join port p

on p.id = s.departure\_port\_id

Procedures and functions

Create a procedure named**'insert\_credit'**to insert the values in the credit\_card table by passing 5 inputs as parameters.  
  
**Hints:**  
**Procedure name:**  insert\_credit  
**Input parameter :** credit\_id with data type as number,credit\_card\_number with data type as varchar,credit\_card\_expire with data type as varchar,holder\_name with data type as varchar and card\_type with data type as varchar  
**Table used:** credit\_card

create or replace procedure insert\_credit

(credit\_id in credit\_card.id%type,

credit\_card\_number in credit\_card.card\_number%type,

credit\_card\_expire in credit\_card.card\_expire%type,

holder\_name in credit\_card.name%type,

card\_type in credit\_card.cc\_type%type)

is

begin

insert into credit\_card (id, card\_number, card\_expire, name, cc\_type)

values (credit\_id,credit\_card\_number, credit\_card\_expire, holder\_name, card\_type);

commit;

END;

/

Create a procedure named **'payment\_count'** which accepts one output parameter p\_count with number as its data type. This procedure should print the count of sucessful payments (response\_msg as 'Successful') with alias name as **'successful\_payment\_count'.**  
  
**Hints:**  
**Procedure name :** payment\_count  
**Output parameter :**p\_count  
**Table used :**payment\_authorization  
**Function used :**count

create or replace procedure payment\_count(p\_count out int)

is

begin

select count(response\_msg) into p\_count from payment\_authorization where response\_msg = 'Successful';

end;

/

Please go through the below Function spec:  
**Function:**  
Create a Function named **'find\_credit\_card'** which takes card\_no as input and returns the holder name of type varchar.  
  
**Function name:**   find\_credit\_card  
**Input Parameter:** card\_no with data type as varchar  
**Output variable :**holder\_name with data type as varchar

create or replace function find\_credit\_card (card\_no in number)

return varchar2 is holder\_name varchar2(255);

BEGIN

SELECT name

INTO holder\_name

FROM credit\_card

WHERE card\_number = card\_no;

RETURN(holder\_name);

END;

/

Please go through the below Function spec:   
**(a) Function:**   
Create a Function named **'find\_balance'** which takes invoice\_id as input and returns the balance\_status of type varchar.   
**Function name :**find\_balance   
**Input Parameter :** invoice\_id in int   
**Output variable :** balance\_status with data type varchar   
  
**Design rules:**   
1) If the balance of the given **invoice id is equal to zero**,then display the status as **'The payment has been Completed'**   
2) If the balance of the given **invoice id is greater than zero**,then display the status as **amount 'yet to be paid'**

create or replace function find\_balance (invoice\_id in number)

return varchar2 is balance\_status varchar2(255);

begin

select

case when balance > '0' then (concat(balance, ' yet to be paid'))

else 'The payment has been Completed'

end into balance\_status from invoice

where id = invoice\_id;

return (balance\_status);

end;

/

Create a procedure named **'select\_city'** which accepts one input parameter **user\_id** of type number and one output parameter **city\_details**of type varchar. This procedure is used to display the city\_details of user.If the user is from bangalore then display the city\_details as **'User is from Bangalore'**,or if the user is from chennai then display the city\_details as **'User is from Chennai'**, else display the city\_details as **'User is from other cities'.**   
  
Hints:   
**Procedure name:** select\_city   
**Input parameter :** user\_id with data type as number   
**Output parameter:** city\_details with data type as varchar.

create or replace procedure select\_city (user\_id in number, city\_details out varchar2(255))

is

begin

select

case when city = 'Bangalore' then (concat('User is from ', city)),

when city = 'Chennai' then (concat('User is from ', city))

else 'User is from other cities'

end into city\_details from contact c

join user\_details ud

on c.id = ud.contact\_id

where ud.id = user\_id;

end;

/

show errors

Create a procedure named **'company\_count'** which accepts one output parameter**c\_count**with number as its data type.This procedure should print the count of companies which are from the country **'USA'.**Give alias name as**'count\_of\_companies'.**  
  
**Hints:**  
**Procedure name :** company\_count  
**Output parameter :** c\_count  
**Table used :**company  
**Function used :** count

create or replace procedure company\_count(c\_count out int)

is

begin

select count(country) into c\_count from address a

join company c

on c.address\_id = a.id

where country = 'USA';

end;

/

Procedures and functions

Create a procedure named '**select\_invoice\_line**' that is used to display the details of the invoice line. This procedure takes invoice line id(i.e, **invoice\_line\_id**) as input and returns the type, amount and quantity of the given id as **invoice\_line\_details**.  
The invoice\_line\_details should be returned as '**sys\_refcursor**'. The details must include the following:  
type, amount, quantity.

create or replace procedure select\_invoice\_line (invoice\_line\_id in number, invoice\_line\_details out SYS\_REFCURSOR) as

begin

OPEN invoice\_line\_details FOR

SELECT type,

amount,

quantity

FROM invoice\_line

WHERE id = invoice\_line\_id;

END;

/

Create a procedure named '**select\_invoice**' that is used to display the details of the invoice. This procedure takes id(i.e, **invoice\_id**) as input and returns the status and paper\_invoice\_batch\_number as **invoice\_details** .  
The invoice\_details should be returned as '**sys\_refcursor**'. The details must include the following:  
status, paper\_invoice\_batch\_number.

create or replace procedure select\_invoice (invoice\_id in number, invoice\_details out SYS\_REFCURSOR) as

begin

OPEN invoice\_details FOR

SELECT status,

paper\_invoice\_batch\_number

FROM invoice

WHERE id = invoice\_id;

END;

/

Please go through the below Function spec:   
**(a) Function:**   
Create a function named **'find\_transaction\_type'** that will accept the transaction\_type\_id as input. Based on this input, the function must return the transaction type name of type varchar.   
**Function name :** find\_transaction\_type,   
**Input Parameter :** transaction\_type\_id in int

create or replace function find\_transaction\_type (transaction\_type\_id in int)

return varchar2 is transactiontypename varchar2(255);

begin

select type into transactiontypename from transaction\_type where id = transaction\_type\_id;

return(transactiontypename);

EXCEPTION

when no\_data\_found then

return('No such Type');

end;

/

Please go through the below Function and Exception spec:  
**(a) Function:**  
Create a function named**'find\_invoice\_status'** that will accept the cust\_id as input. Based on this input, the function must return the status of the invoice of type varchar.  
**Function name :**find\_invoice\_status,  
**Input Parameter :**cust\_id in int

create or replace function find\_invoice\_status(cust\_id in int)

return varchar2 is statusoftheinvoice varchar2(255);

begin

select i.status into statusoftheinvoice from invoice i

join customer c

on i.customer\_id=c.id

where c.id = cust\_id;

return(statusoftheinvoice);

EXCEPTION

when no\_data\_found then

return('No Such Customer');

when TOO\_MANY\_ROWS then

return('Multiple Rows Returned');

end;

/

Please go through the below Function and Exception spec:  
**(a) Function:**  
Create a Function named **'find\_authorization'**which takes trans\_id as input and returns the processor name of type varchar.  
**Function name:**  find\_authorization.  
**Input Parameter:** trans\_id with data type as varchar

create or replace function find\_authorization (trans\_id in varchar2)

return varchar2 is processorname varchar2(255);

CURSOR c1

IS

select processor from payment\_authorization where transaction\_id = trans\_id;

begin

open c1;

fetch c1 into processorname;

if c1%notfound then

processorname := 'No Such Payment Authorization';

end if;

close c1;

return processorname;

end;

/

Create a procedure named **'find\_shipment'** that is used to display the shipment details. This procedure takes **shipment\_status\_id(i.e, status\_id)**as input and outputs the shipment id, name, booking\_number, departure\_time and arrival\_time of the shipments of the given status\_id. The records must be sorted in ascending order based on shipment\_id.   
The shipments\_details should be returned as **'sys\_refcursor'.** The details must include the following:   
**shipment id, name, booking\_number, departure\_time and arrival\_time**

create or replace procedure find\_shipment (shipment\_status\_id in number,

shipment\_details out SYS\_REFCURSOR) as

begin

OPEN shipment\_details FOR

SELECT s.id as shipmentid, s.name, s.booking\_number, s.departure\_time, s.arrival\_time

FROM shipment s join shipment\_status ss

on s.shipment\_status\_id = ss.id

WHERE ss.id = shipment\_status\_id;

END;

/