CIS 5430 Project
DATABASES AND DATA WAREHOUSING

Project Group: 6

SIDDHI UDANI
CIN: 307401801

MAITRI SHAH
CIN: 307382379

MONISH PHATARPEKAR
CIN: 307399097

ALBERT DABNEY
CIN: 309786465

Company Description

The online store database needs to keep track of orders for its inventory. When a customer places orders, the system must record that the order and order items. The system must update the available quantity on hand to reflect that the by product(s) has been sold. When an employee processes orders, the system must confirm that the ordered items are in stock. The online store need to keep track of customers and employees, too. The system must update the available quantity on hand to reflect that the by product(s) has been sold.

We store each customer's ID, name, DOB, Email ID, address, social security number, Zipcode, state and phone number. The customer is of 2 types: Business customer and individual customer

We store each employee's ID, email, address, job title, name along with order details so the orders can be processed.

We want to keep track of orders placed by customers. We keep each customer details along with order and order items.

Each order has order ID, situation, cost, address, size with default current date.

Each Item has ID, description, price and quantity.

Business rules.

One customer may or may not place many orders.
One order must be placed by one and only one customer.

One order must contain one or more item.
One item may or may not be in many orders.

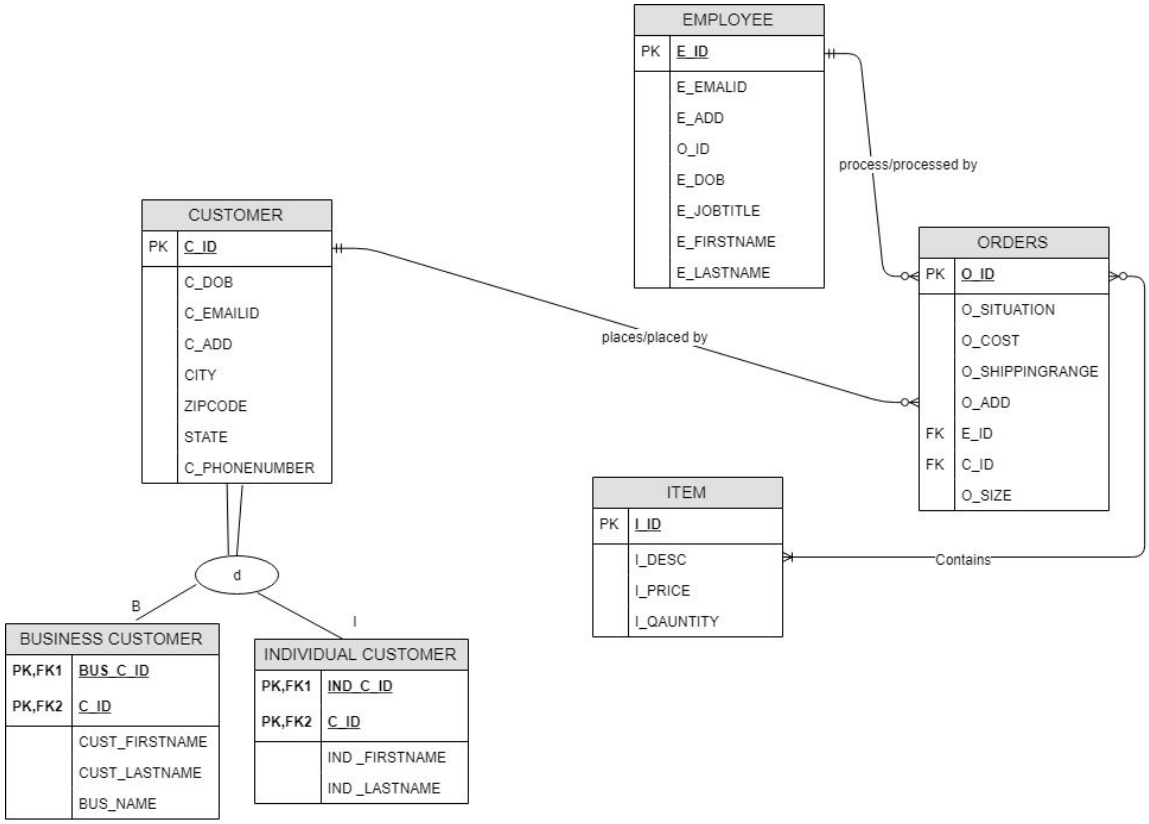
One employee may or may not process many orders.
One order must be processed by one and only one employee.

One customer must be either a business or individual customer.
One business customer or individual customer must be a customer.

1. **Identify entity types and relationship types. Fill out the following relationship matrix.**

	CUSTOMER	ORDER	EMPLOYEE	ITEM
CUSTOMER		PLACES	--	--
ORDER	PLACED BY	--	PROCESSED BY	CONTAINS
EMPLOYEE	--	PROCESS	--	--
ITEM	--	CONTAINED IN	--	--

2.Draw an EER diagram includes
1) entity types, 2) relationship types, 3) keys, 4) cardinality constraints (must show participation).



3. **Database Logical Design:** Map the ER diagram to a relational database schema indicating **the relation name**, **primary key and foreign key.** Add appropriate additional attributes by yourself.

Table Name: ORDERS

O_ID(PK)	O_SITUATION	O_COST	O_SHIPPINGRANGE	O_ADD	E_ID (FK)	C_ID (FK)	O_SIZE

Table Name: ITEMS

I_ID (PK)	I_DESC	I_PRICE	I_QUANTITY

Table Name: CUSTOMER

C_ID(PK)	C_DOB	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER

Table Name: BUSINESS_CUSTOMER

BUS_C_ID (PK, FK1)	C_ID (PK,FK2)	CUST_FIRSTNAME	CUST_LASTNAME	BUS_NAME

Table Name: INDIVIDUAL_CUSTOMER

IND_C_ID (PK, FK1)	C_ID (PK,FK2)	IND_FIRSTNAME	IND_LASTNAME

Table Name: EMPLOYEE

E_ID (PK)	E_EMAILID	E_ADD	E_DOB	E_FIRSTNAME	E_LASTNAME	E_JOBTITLE

Table Name: ORDER_CHECK

O_ID (PK,FK1)	I_ID (PK,FK2)	ORDEREDQUANTITY

4. Establish join paths for the above relational database using the referential integrity by drawing arrow lines between the above tables. Indicate all the foreign keys (FK). F.K. -> P.K. (Foreign Key refers to Primary Key)

Table Name: ORDERS

O_ID(PK)	O_SITUATION	O_COST	O_SHIPPINGRANGE	O_ADD	E_ID (FK)	C_ID (FK)	O_SIZE

Table Name: ITEMS

I_ID (PK)	I_DESC	I_PRICE	I_QUANTITY

Table Name: CUSTOMER

C_ID(PK)	C_DOB	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER

Table Name: BUSINESS_CUSTOMER

BUS_C_ID (PK, FK1)	C_ID (PK,FK2)	CUST_FIRSTNAME	CUST_LASTNAME	BUS_NAME

Table Name: INDIVIDUAL_CUSTOMER

IND_C_ID (PK, FK1)	C_ID (PK,FK2)	IND_FIRSTNAME	IND_LASTNAME

Table Name: EMPLOYEE

E_ID (PK)	E_EMAILID	E_ADD	E_DOB	E_FIRSTNAME	E_LASTNAME	E_JOBTITLE

Table Name: ORDER_CHECK

O_ID (PK,FK1)	I_ID (PK,FK2)	ORDEREDQUANTITY

- ORDERS.C_ID → CUSTOMER.C_ID
- ORDERS.E_ID →EMPLOYEE.E_ID
- BUSINESS_CUSTOMER.C_ID → CUSTOMER.C_ID
- INDIVIDUAL_CUSTOMER.C_ID → CUSTOMER.C_ID
- ORDER_CHECK.O_ID → ORDER.O_ID
- ORDER_CHECK.I_ID → ITEM.I_ID

5. Do function analysis for each of your tables
   Attribute A -> Attribute B (Determinant attribute(s) Determines Dependent Attribute(s))

TABLE CUSTOMER:
C_ID → C_DOB, C_EMAILID, C_ADD, C_PHONENUMBER (Full Dependency)
ZIPCODE → STATE, CITY (Transitive Dependency)

TABLE ORDERS:
O_ID → O_SITUATION, O_COST, O_SHIPPINGRANGE, O_ADD, O_SIZE (Full Dependency)

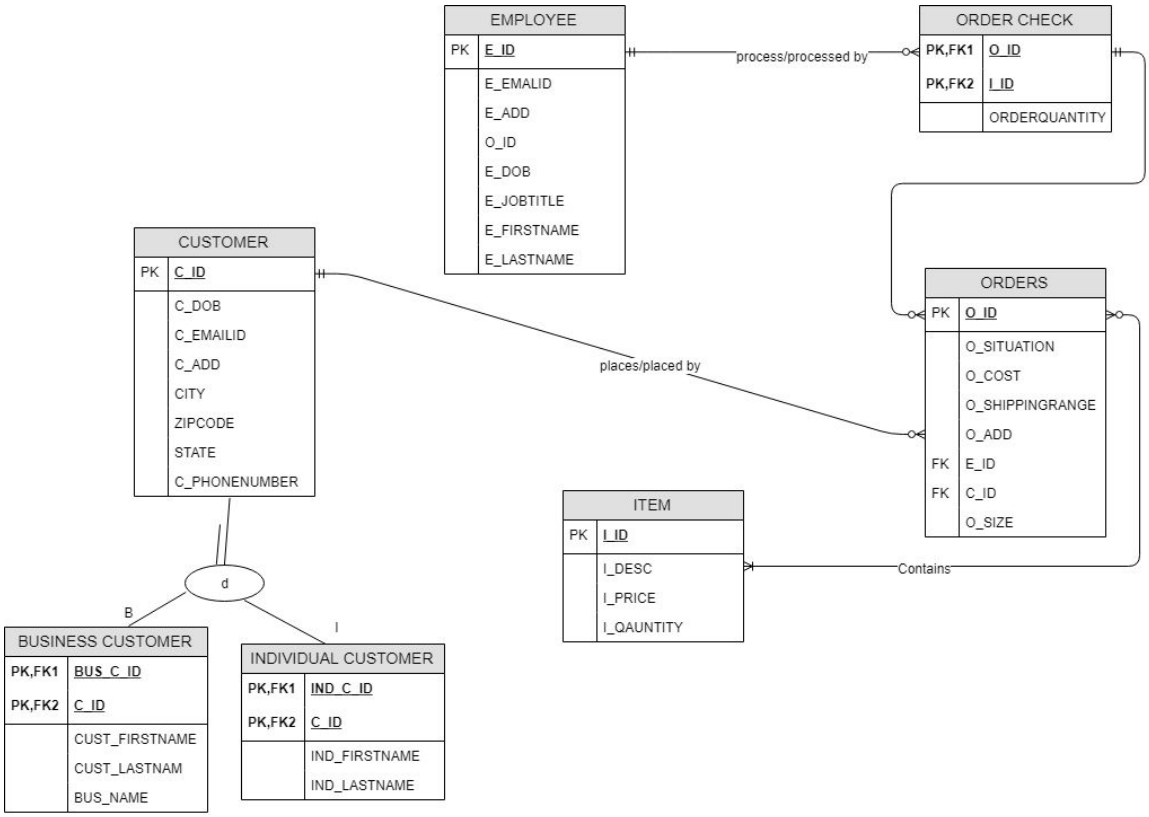
TABLE ITEMS:
I_ID → I_DESC, I_PRICE, I_QUANTITY (Full Dependency)

TABLE BUSINESS_CUSTOMER:
C_ID, BUS_C_ID →CUST_FIRSTNAME, CUST_LASTNAME, BUS_NAME (Full Dependency)

TABLE INDIVIDUAL_CUSTOMER:
C_ID, IND_C_ID → IND_FIRSTNAME, IND_LASTNAME (Full Dependency)

TABLE EMPLOYEE:
E_ID → E_FIRSTNAME, E_LASTNAME, E_EMAIL, E_ADD, E_DOB, E_JOBTITLE (Full Dependency)

TABLE ORDER_CHECK:
O_ID, E_ID → ORDEREDQUANTITY (Full Dependency)



6. Show all the normalized tables and indicate the normalization form for each of your tables.

Table Name	1NF	2NF	3NF
Orders	X	X	X
Items	X	X	X
Customer	X	X	
Business_Customer	X	X	X
Individual_Customer	X	X	X
Employee	X	X	X
Order_Check	X	X	X

7. Create the relational database with five constraints in appropriate tables and load data into the database via ORACLE SQL*PLUS.

Submission: Printout of SQL DDL script (Database creation script), database structure (DESC TableName) and database instance (SELECT * FROM TableName).

SQL DDL SCRIPT: TO CREATE NEW TABLE

TABLE: ITEMS

```
CREATE TABLE ITEMS (
I_ID INT NOT NULL,
I_DESC VARCHAR2(255),
I_PRICE INT,
I_QAUNTITY INT,
PRIMARY KEY (I_ID)
);
```

OUTPUT: Table Created.

DATABASE STRUCTURE:

Desc ITEMS;

Name	Null?	Type
I_ID	NOT NULL	NUMBER(38)
I_DESC		VARCHAR2(255)
I_PRICE		NUMBER(38)
I_QAUNTITY		NUMBER(38)

TO INSERT VALUES INTO ITEM TABLE:

```
INSERT INTO ITEMS
VALUES (1,'TABLE',250,10);

INSERT INTO ITEMS
VALUES (2,'LAPTOP',1650,3);

INSERT INTO ITEMS
VALUES (3,'MOBILE',2600,4);
```

```
INSERT INTO ITEMS
VALUES (4,'FAN',150,5);

INSERT INTO ITEMS
VALUES (5,'DESK',550,5);

INSERT INTO ITEMS
VALUES (6,'TELEVISION',4000,3);

INSERT INTO ITEMS
VALUES (7,'BOOKS',350,30);

INSERT INTO ITEMS
VALUES (8,'BED',850,6);

INSERT INTO ITEMS
VALUES (9,'LAMP',450,20);

INSERT INTO ITEMS
VALUES (10,'CHAIR',1150,11);
```

OUTPUT: ROWS CREATED

DATABASE INSTANCE:

SELECT * FROM ITEMS;

I_ID	I_DESC	I_PRICE	I_QAUNTITY
1	TABLE	250	10
2	LAPTOP	1650	3
3	MOBILE	2600	4
4	FAN	150	5
5	DESK	550	5
6	TELEVISION	4000	3
7	BOOKS	350	30
8	BED	850	6
9	LAMP	450	20
10	CHAIR	1150	11

10 rows selected.

SQL DDL SCRIPT: TO CREATE A NEW TABLE

TABLE: CUSTOMER

```
CREATE TABLE CUSTOMERS (
C_ID INT NOT NULL,
C_DOB DATE,
C_SSN INT,
C_EMAILID VARCHAR2(255),
C_ADD VARCHAR2(255),
CITY VARCHAR2(255),
ZIPCODE INT,
STATE VARCHAR2(255),
```

C_PHONENUMBER INT,
PRIMARY KEY (C_ID)
);

OUTPUT: Table Created.

DATABASE STRUCTURE:

Desc CUSTOMER;

Name	Null?	Type
C_ID	NOT NULL	NUMBER(38)
C_DOB		DATE
C_SSN		NUMBER(38)
C_EMAILID		VARCHAR2(255)
C_ADD		VARCHAR2(255)
CITY		VARCHAR2(255)
ZIPCODE		NUMBER(38)
STATE		VARCHAR2(255)
C_PHONENUMBER		NUMBER(38)

TO INSERT VALUES INTO CUSTOMER TABLE:

INSERT INTO CUSTOMERS
VALUES (1,’09/AUG/2001’,213526896,'MON_2889@GMAIL.COM’,’102 ATL’,'ALHAMBRA',91810,’CA’,3235206321);

INSERT INTO CUSTOMERS
VALUES (2,'12/AUG/1987',213632563,'DOBI_96@YAHOO.COM','307 CHAPEL','CITY TERRACE',91312,'CA',2325698520);

INSERT INTO CUSTOMERS
VALUES (3,'05/JUNE/1994',307256321,'SABOO@CALSTATELA.EDU','32 BLV','EL MONTE',91456,'CA',4589632145);

INSERT INTO CUSTOMERS
VALUES (4,'09/MAY/1996',307256123,'CHAAP_LI@GMAIL.COM','806 COLL VIEW DRIVE','CITY TERRACE',91562,'CA',2565412369);

INSERT INTO CUSTOMERS
VALUES (5,'07/OCT/1992',120589632,'AB_9@GMAIL.COM','89 MON','ALHAMBRA',91120,'CA',1452367896);

INSERT INTO CUSTOMERS
VALUES (6,'10/JAN/1994',450201478,'MONEY_12@YAHOO.COM','210 BALDWIN','SAN GABRIEL',91450,'CA',4562223698);

INSERT INTO CUSTOMERS
VALUES (7,'11/JULY/1996',307654789,'KU_12@GMAIL.COM','502 BLV','MONTEREY PARK',91300,'CA',4562103698);

INSERT INTO CUSTOMERS
VALUES (8,'10/MAR/1992',563245789,'N23@GMAIL.COM','203 KBP','MONTEBELLO',91125,'CA',3232052314);

INSERT INTO CUSTOMERS
VALUES (9,'01/APR/1994',307399075,'FLASH_345@YAHOO.COM','512 ALV','LUNAS',91801,'CA',6265461230);

INSERT INTO CUSTOMERS
VALUES (10,'05/FEB/2004',458963210,'WE_786@GMAIL.COM','1155 COLLEGE VIEW DRIVE','MONTEREY',91812,'CA',3232050260);

DATABASE INSTANCE:

SELECT * FROM CUSTOMER;

C_ID	C_DOB	C_SSN	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER
1	09-AUG-01	213526896	MON_2889@GMAIL.COM	102 ATL	ALHAMBRA	91810	CA	3235206321
2	12-AUG-87	213632563	DOBI_96@YAHOO.COM	307 CHAPEL	CITY TERRACE	91312	CA	2325698520
3	05-JUN-94	307256321	SABOO@CALSTATELA.EDU	32 BLV	EL MONTE	91456	CA	4589632145
4	09-MAY-96	307256123	CHAAP_LI@GMAIL.COM	806 COLL VIEW DRIVE	CITY TERRACE	91562	CA	2565412369
5	07-OCT-92	120589632	AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120	CA	1452367896
6	10-JAN-94	450201478	MONEY_12@YAHOO.COM	210 BALDWIN	SAN GABRIEL	91450	CA	4562223698
7	11-JUL-96	307654789	KU_12@GMAIL.COM	502 BLV	MONTEREY PARK	91300	CA	4562103698
8	10-MAR-92	563245789	N23@GMAIL.COM	203 KBP	MONTEBELLO	91125	CA	3232052314
9	01-APR-94	307399075	FLASH_345@YAHOO.COM	512 ALV	LUNAS	91801	CA	6265461230
10	05-FEB-04	458963210	WE_786@GMAIL.COM	1155 COLLEGE VIEW DRIVE	MONTEREY	91812	CA	3232050260

SQL DDL SCRIPT: TO CREATE NEW TABLE

TABLE: BUSINESS_CUSTOMER

CREATE TABLE BUSINESS_CUSTOMER (
BUS_C_ID INT **NOT NULL PRIMARY KEY**,
BUS_NAME VARCHAR2(255),
FOREIGN KEY (BUS_C_ID) REFERENCES CUSTOMERS(C_ID)
);

OUTPUT: Table Created.

DATABASE STRUCTURE:

Desc BUSINESS_CUSTOMER;

Name	Null?	Type
BUS_C_ID	NOT NULL	NUMBER(38)
BUS_NAME		VARCHAR2(255)

INSERT VALUES INTO BUSINESS_CUSTOMER TABLE:

SELECT * FROM BUSINESS_CUSTOMER;

INSERT INTO BUSINESS_CUSTOMER
VALUES (1,'GLOBALTECH');

INSERT INTO BUSINESS_CUSTOMER
VALUES (2,'VALUE SOLUTIONS');

INSERT INTO BUSINESS_CUSTOMER
VALUES (3,'HIGHTECH');

INSERT INTO BUSINESS_CUSTOMER
VALUES (4,'AVANADE');

INSERT INTO BUSINESS_CUSTOMER
VALUES (5,'PETCO');

DATABASE INSTANCE:

SELECT * FROM BUSINESS_CUSTOMER;

BUS_C_ID	BUS_NAME
1	GLOBALTECH
2	VALUE SOLUTIONS
3	HIGHTECH
4	AVANADE
5	PETCO

SQL DDL SCRIPT: TO CREATE NEW TABLE

TABLE: INDIVIDUAL_CUSTOMER

CREATE TABLE INDIVIDUAL_CUSTOMER (
IND_C_ID INT **NOT NULL PRIMARY KEY**,
IND_FIRSTNAME VARCHAR2(255),
IND_LASTNAME VARCHAR2(255),
FOREIGN KEY (IND_C_ID) REFERENCES CUSTOMERS(C_ID)
);

OUTPUT: Table Created.

DATABASE STRUCTURE:

Desc INDIVIDUAL_CUSTOMER;

Name	Null?	Type
IND_C_ID	NOT NULL	NUMBER(38)
IND_FIRSTNAME		VARCHAR2(255)
IND_LASTNAME		VARCHAR2(255)

INSERT VALUES INTO INDIVIDUAL CUSTOMER TABLE:

INSERT INTO INDIVIDUAL_CUSTOMER
VALUES (6,'MONISH','RODRIQUEZ');

INSERT INTO INDIVIDUAL_CUSTOMER
VALUES (7,'MARY','JOHNSON');

INSERT INTO INDIVIDUAL_CUSTOMER
VALUES (8,'PETER','PHATARPEKAR');

INSERT INTO INDIVIDUAL_CUSTOMER
VALUES (9,'JEVEL','VAZ');

INSERT INTO INDIVIDUAL_CUSTOMER
VALUES (10,'SHERIN','DSOUZA');

DATABASE INSTANCE:

SELECT * FROM INDIVIDUAL_CUSTOMER;

IND_C_ID	IND_FIRSTNAME	IND_LASTNAME
6	MONISH	RODRIQUEZ
7	MARY	JOHNSON
8	PETER	PHATARPEKAR
9	JEVEL	VAZ
10	SHERIN	DSOUZA

SQL DDL SCRIPT: TO CREATE NEW TABLE

TABLE: EMPLOYEE

CREATE TABLE EMPLOYEE (
E_ID INT **NOT NULL**,
E_EMAILID VARCHAR2(255),
E_ADD VARCHAR2(255),
E_DOB DATE,
E_FIRSTNAME VARCHAR2(255),
E_LASTNAME VARCHAR2(255),
E_JOBTITLE VARCHAR2(255),
PRIMARY KEY (E_ID)
);

OUTPUT: Table Created.

SQL DML STATEMENT:

ALTER TABLE EMPLOYEE
ADD **UNIQUE** (E_EMAILID);

OUTPUT: Table Altered.

DATABASE STRUCTURE:

Desc EMPLOYEE;

Name	Null?	Type
E_ID	NOT NULL	NUMBER(38)
E_EMAILID		VARCHAR2(255)
E_ADD		VARCHAR2(255)
E_DOB		DATE
E_FIRSTNAME		VARCHAR2(255)
E_LASTNAME		VARCHAR2(255)
E_JOBTITLE		VARCHAR2(255)

INSERT VALUES INTO EMPLOYEE TABLE:

INSERT INTO EMPLOYEE
VALUES (1,'NI@GMAIL.COM','87 ATL','10-OCT-1989','BOSCO','JOHNS','CEO');

INSERT INTO EMPLOYEE
VALUES (2,'SH@YAHOO.COM','1021 BLV','21-MAR-1978','PEDRO','MAYA','MANAGER');

```
INSERT INTO EMPLOYEE
VALUES (3,'SI@GMAIL.COM','1234 COLL','12-MAY-1999','KEVIN','POLO','EMPLOYEE');

INSERT INTO EMPLOYEE
VALUES (4,'DD@YAHOO.COM','8021
OLIVE','01-JAN-1994','MARSHAL','SMITH','ASSITMANAGER');

INSERT INTO EMPLOYEE
VALUES (5,'HI@GMAIL.COM','26 BLW','08-OCT-1992','ALBERT','SHAH','ADMINISTRATOR');

INSERT INTO EMPLOYEE
VALUES (6,'CH@YAHOO.COM','21 MAIN','21-OCT-1991','DOBBY','UDANI','DIRECTOR');

INSERT INTO EMPLOYEE
VALUES (7,'AA@GMAIL.COM','556 CHP','31-DEC-1984','JACKIE','SMITH','INTERN');

INSERT INTO EMPLOYEE
VALUES (8,'PL@GMAIL.COM','32 JAI','12-APR-1994','ARRON','DEBI','EXECUTIVE');

INSERT INTO EMPLOYEE
VALUES (9,'I_9@YAHOO.COM','45 KRI','01-DEC-1964','MEDRICK','ROCKY','COORDINATOR');

INSERT INTO EMPLOYEE
VALUES (10,'SAB@GMAIL.COM','32 CRT','29-FEB-2004','CASEY','NEISTAT','CONTROLLER');

INSERT INTO EMPLOYEE
VALUES (12,'HIJACK@GMAIL.COM','1155 CVD','18-JUNE-1982','ROSS','SAHANI','JUNIOR
ADMIN');
```

DATABASE INSTANCE:

```
SELECT * FROM EMPLOYEE
```

E_ID	E_EMAILID	E_ADD	E_DOB	E_FIRSTNAME	E_LASTNAME	E_JOBTITLE	E_SAL
1	NI@GMAIL.COM	87 ATL	10-OCT-89	BOSCO	JOHNS	CEO	21000
2	SH@YAHOO.COM	1021 BLV	21-MAR-78	PEDRO	MAYA	MANAGER	17850
3	SI@GMAIL.COM	1234 COLL	12-MAY-99	KEVIN	POLO	EMPLOYEE	7000
4	DD@YAHOO.COM	8021 OLIVE	01-JAN-94	MARSHAL	SMITH	ASSITMANAGER	6500
5	HI@GMAIL.COM	26 BLW	08-OCT-92	ALBERT	SHAH	ADMINISTRATOR	9000
6	CH@YAHOO.COM	21 MAIN	21-OCT-91	DOBBY	UDANI	DIRECTOR	18500
7	AA@GMAIL.COM	556 CHP	31-DEC-84	JACKIE	SMITH	INTERN	2500
8	PL@GMAIL.COM	32 JAI	12-APR-94	ARRON	DEBI	EXECUTIVE	11000
9	I_9@YAHOO.COM	45 KRI	01-DEC-64	MEDRICK	ROCKY	COORDINATOR	9500
10	SAB@GMAIL.COM	32 CRT	29-FEB-04	CASEY	NEISTAT	CONTROLLER	10000
12	HIJACK@GMAIL.COM	1155 CVD	18-JUN-82	ROSS	SAHANI	JUNIOR ADMIN	12000

11 rows selected.

TABLE: ORDERS

```
CREATE TABLE ORDERS (
O_ID INT NOT NULL PRIMARY KEY,
O_SITUATION VARCHAR2(255),
O_ADD VARCHAR2(255),
O_SCALE VARCHAR2(255),
C_ID int,
E_ID int,
```

FOREIGN KEY(C_ID) REFERENCES CUSTOMERS (C_ID),
FOREIGN KEY(E_ID) REFERENCES EMPLOYEE (E_ID)
);

OUTPUT: Table Created.

DATABASE STRUCTURE:

Desc ORDERS;

Name	Null?	Type
O_ID	NOT NULL	NUMBER(38)
O_SITUATION		VARCHAR2(255)
O_ADD		VARCHAR2(255)
O_SCALE		VARCHAR2(255)
C_ID		NUMBER(38)
E_ID		NUMBER(38)

TO INSERT VALUES INTO ORDERS TABLE:

INSERT INTO ORDERS
VALUES (1,'COMPLETE','DALLAS','MEDIUM',1,1);

INSERT INTO ORDERS
VALUES (2,'COMPLETE','NEWYORK','MEDIUM',2,3);

INSERT INTO ORDERS
VALUES (3,'INCOMPLETE','OHIO','LARGE',3,4);

INSERT INTO ORDERS
VALUES (4,'COMPLETE','LAS VEGAS','SMALL',4,5);

INSERT INTO ORDERS
VALUES (5,'IN PROCESS','ARIZONA','MEDIUM',5,6);

INSERT INTO ORDERS
VALUES (6,'INCOMPLETE','FLORIDA','LARGE',6,7);

INSERT INTO ORDERS
VALUES (7,'COMPLETE','GEORGIA','LARGE',7,8);

INSERT INTO ORDERS
VALUES (8,'IN PROCESS','MEXICO','SMALL',8,9);

INSERT INTO ORDERS
VALUES (9,'INCOMPLETE','SAN JOSE','MEDIUM',9,10);

INSERT INTO ORDERS
VALUES (10,'IN PROCESS','SEATTLE','LARGE',10,2);

OUTPUT: ROWS CREATED

DATABASE INSTANCE:

SELECT * FROM ORDERS;

O_ID	O_SITUATION	O_ADD	O_SCALE	C_ID	E_ID
1	COMPLETE	DALLAS	MEDIUM	1	1
2	COMPLETE	NEWYORK	MEDIUM	2	3
3	INCOMPLETE	OHIO	LARGE	3	4
4	COMPLETE	LAS VEGAS	SMALL	4	5
5	IN PROCESS	ARIZONA	MEDIUM	5	6
6	INCOMPLETE	FLORIDA	LARGE	6	7
7	COMPLETE	GEORGIA	LARGE	7	8
8	IN PROCESS	MEXICO	SMALL	8	9
9	INCOMPLETE	SAN JOSE	MEDIUM	9	10
10	IN PROCESS	SEATTLE	LARGE	10	2

10 rows selected.

SQL DDL SCRIPT: TO CREATE NEW TABLE

TABLE: ORDERCHECK

```
CREATE TABLE ORDER_CHECK (
O_ID INT NOT NULL,
I_ID INT NOT NULL,
ORDEREDQUANTITY INT CHECK (ORDEREDQUANTITY>0) ,
PRIMARY KEY (O_ID, I_ID)
);
```

OUTPUT: Table Created.

SQL DML STATEMENT:

```
ALTER TABLE ORDER_CHECK
ADD FOREIGN KEY (I_ID) REFERENCES ITEMS(I_ID);

ALTER TABLE ORDER_CHECK
ADD FOREIGN KEY (O_ID) REFERENCES ORDERS(O_ID);
```

OUTPUT: Table Altered.

DATABASE STRUCTURE:

Desc ORDER_CHECK;

Name	Null?	Type
O_ID	NOT NULL	NUMBER(38)
I_ID	NOT NULL	NUMBER(38)
ORDEREDQUANTITY		NUMBER(38)

INSERT VALUES INTO ORDER CHECK TABLE:

```
INSERT INTO ORDER_CHECK
VALUES (1,1,55);

INSERT INTO ORDER_CHECK
VALUES (2,2,124);

INSERT INTO ORDER_CHECK
VALUES (3,3,54);

INSERT INTO ORDER_CHECK
VALUES (4,4,145);

INSERT INTO ORDER_CHECK
VALUES (5,5,120);

INSERT INTO ORDER_CHECK
VALUES (6,6,75);

INSERT INTO ORDER_CHECK
VALUES (7,7,55);

INSERT INTO ORDER_CHECK
VALUES (8,8,25);

INSERT INTO ORDER_CHECK
VALUES (9,9,126);

INSERT INTO ORDER_CHECK
VALUES (10,10,92);
```

DATABASE INSTANCE:

SELECT * FROM ORDER_CHECK;

O_ID	I_ID	ORDEREDQUANTITY
1	1	55
2	2	124
3	3	54
4	4	145
5	5	120
6	6	75
7	7	55
8	8	25
9	9	126
10	10	92

8. Show Insert, Update and Delete statement and View in each Group.

SQL DML STATEMENTS:

INSERT STATEMENT:

```
INSERT INTO CUSTOMERS (C_ID,C_SSN,C_EMAILID,C_ADD,CITY,C_PHONENUMBER)
VALUES (11,213895296,'ARIF4519@GMAIL.COM','801 OLIVE','ALHAMBRA',3233433000);
```

OUTPUT BEFORE 'INSERT' STATEMENT:

C_ID	C_DOB	C_SSN	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER
1	09-AUG-01	213526896	MON_2889@GMAIL.COM	102 ATL	ALHAMBRA	91810	CA	3235206321
2	12-AUG-87	213632563	DOBI_96@YAHOO.COM	307 CHAPEL	CITY TERRACE	91312	CA	2325698520
3	05-JUN-94	307256321	SABOO@CALSTATELA.EDU	32 BLV	EL MONTE	91456	CA	4589632145
4	09-MAY-96	307256123	CHAAP_LI@GMAIL.COM	806 COLL VIEW DRIVE	CITY TERRACE	91562	CA	2565412369
5	07-OCT-92	120589632	AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120	CA	1452367896
6	10-JAN-94	450201478	MONEY_12@YAHOO.COM	210 BALDWIN	SAN GABRIEL	91450	CA	4562223698
7	11-JUL-96	307654789	KU_12@GMAIL.COM	502 BLV	MONTEREY PARK	91300	CA	4562103698
8	10-MAR-92	563245789	N23@GMAIL.COM	203 KBP	MONTEBELLO	91125	CA	3232052314
9	01-APR-94	307399075	FLASH_345@YAHOO.COM	512 ALV	LUNAS	91801	CA	6265461230
10	05-FEB-04	458963210	WE_786@GMAIL.COM	1155 COLLEGE VIEW DRIVE	MONTEREY	91812	CA	3232050260

OUTPUT AFTER 'INSERT' STATEMENT:

C_ID	C_DOB	C_SSN	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER
1	09-AUG-01	213526896	MON_2889@GMAIL.COM	102 ATL	ALHAMBRA	91810	CA	3235206321
2	12-AUG-87	213632563	DOBI_96@YAHOO.COM	307 CHAPEL	CITY TERRACE	91312	CA	2325698520
3	05-JUN-94	307256321	SABOO@CALSTATELA.EDU	32 BLV	EL MONTE	91456	CA	4589632145
4	09-MAY-96	307256123	CHAAP_LI@GMAIL.COM	806 COLL VIEW DRIVE	CITY TERRACE	91562	CA	2565412369
5	07-OCT-92	120589632	AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120	CA	1452367896
6	10-JAN-94	450201478	MONEY_12@YAHOO.COM	210 BALDWIN	SAN GABRIEL	91450	CA	4562223698
7	11-JUL-96	307654789	KU_12@GMAIL.COM	502 BLV	MONTEREY PARK	91300	CA	4562103698
8	10-MAR-92	563245789	N23@GMAIL.COM	203 KBP	MONTEBELLO	91125	CA	3232052314
9	01-APR-94	307399075	FLASH_345@YAHOO.COM	512 ALV	LUNAS	91801	CA	6265461230
10	05-FEB-04	458963210	WE_786@GMAIL.COM	1155 COLLEGE VIEW DRIVE	MONTEREY	91812	CA	3232050260
11		213895296	ARIF4519@GMAIL.COM	801 OLIVE	ALHAMBRA			3233433000

11 rows selected.

UPDATE STATEMENT:

```
UPDATE CUSTOMERS
SET C_DOB = '19/NOVEMBER/1996', ZIPCODE= 91754, STATE = 'CA'
WHERE C_ID = 11;
```

OUTPUT BEFORE 'UPDATE' STATEMENT:

C_ID	C_DOB	C_SSN	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER
1	09-AUG-01	213526896	MON_2889@GMAIL.COM	102 ATL	ALHAMBRA	91810	CA	3235206321
2	12-AUG-87	213632563	DOBI_96@YAHOO.COM	307 CHAPEL	CITY TERRACE	91312	CA	2325698520
3	05-JUN-94	307256321	SABOO@CALSTATELA.EDU	32 BLV	EL MONTE	91456	CA	4589632145
4	09-MAY-96	307256123	CHAAP_LI@GMAIL.COM	806 COLL VIEW DRIVE	CITY TERRACE	91562	CA	2565412369
5	07-OCT-92	120589632	AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120	CA	1452367896
6	10-JAN-94	450201478	MONEY_12@YAHOO.COM	210 BALDWIN	SAN GABRIEL	91450	CA	4562223698
7	11-JUL-96	307654789	KU_12@GMAIL.COM	502 BLV	MONTEREY PARK	91300	CA	4562103698
8	10-MAR-92	563245789	N23@GMAIL.COM	203 KBP	MONTEBELLO	91125	CA	3232052314
9	01-APR-94	307399075	FLASH_345@YAHOO.COM	512 ALV	LUNAS	91801	CA	6265461230
10	05-FEB-04	458963210	WE_786@GMAIL.COM	1155 COLLEGE VIEW DRIVE	MONTEREY	91812	CA	3232050260
11		213895296	ARIF4519@GMAIL.COM	801 OLIVE	ALHAMBRA			3233433000

11 rows selected.

OUTPUT AFTER 'UPDATE' STATEMENT:

C_ID	C_DOB	C_SSN	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER
1	09-AUG-01	213526896	MON_2889@GMAIL.COM	102 ATL	ALHAMBRA	91810	CA	3235206321
2	12-AUG-87	213632563	DOBI_96@YAHOO.COM	307 CHAPEL	CITY TERRACE	91312	CA	2325698520
3	05-JUN-94	307256321	SABOO@CALSTATELA.EDU	32 BLV	EL MONTE	91456	CA	4589632145
4	09-MAY-96	307256123	CHAAP_LI@GMAIL.COM	806 COLL VIEW DRIVE	CITY TERRACE	91562	CA	2565412369
5	07-OCT-92	120589632	AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120	CA	1452367896
6	10-JAN-94	450201478	MONEY_12@YAHOO.COM	210 BALDWIN	SAN GABRIEL	91450	CA	4562223698
7	11-JUL-96	307654789	KU_12@GMAIL.COM	502 BLV	MONTEREY PARK	91300	CA	4562103698
8	10-MAR-92	563245789	N23@GMAIL.COM	203 KBP	MONTEBELLO	91125	CA	3232052314
9	01-APR-94	307399075	FLASH_345@YAHOO.COM	512 ALV	LUNAS	91801	CA	6265461230
10	05-FEB-04	458963210	WE_786@GMAIL.COM	1155 COLLEGE VIEW DRIVE	MONTEREY	91812	CA	3232050260
11	19-NOV-96	213895296	ARIF4519@GMAIL.COM	801 OLIVE	ALHAMBRA	91754	CA	3233433000

11 rows selected.

DELETE STATEMENT:

OUTPUT BEFORE 'DELETE' STATEMENT:

E_ID	E_EMAILID	E_ADD	E_DOB	E_FIRSTNAME	E_LASTNAME	E_JOBTITLE
1	MO@GMAIL.COM	307 CHP	09-FEB-08	MARK	DABNEY	PRODMANAGER
3	SI@GMAIL.COM	1234 COLL	12-MAY-99	KEVIN	POLO	EMPLOYEE
4	DD@YAHOO.COM	8021 OLIVE	01-JAN-94	MARSHAL	SMITH	ASSITMANAGER
5	HI@GMAIL.COM	26 BLW	08-OCT-92	ALBERT	SHAH	ADMINISTRATOR
6	CH@YAHOO.COM	21 MAIN	21-OCT-91	DOBBY	UDANI	DIRECTOR
7	AA@GMAIL.COM	556 CHP	31-DEC-84	JACKIE	SMITH	INTERN
8	PL@GMAIL.COM	32 JAI	12-APR-94	ARRON	DEBI	EXECUTIVE
9	I_9@YAHOO.COM	45 KRI	01-DEC-64	MEDRICK	ROCKY	COORDINATOR
10	SAB@GMAIL.COM	32 CRT	29-FEB-04	CASEY	NEISTAT	CONTROLLER
2	SH@GMAIL.COM	28 ATL	22-JUN-98	PEDRO	MAYA	MANAGER
12	HIJACK@GMAIL.COM	1155 CVD	18-JUN-82	ROSS	SAHANI	JUNIOR ADMIN

11 rows selected.

DELETE FROM EMPLOYEE WHERE E_ID=12;

OUTPUT AFTER 'DELETE' STATEMENT:

SELECT * FROM EMPLOYEE;

E_ID	E_EMAILID	E_ADD	E_DOB	E_FIRSTNAME	E_LASTNAME	E_JOBTITLE
1	MO@GMAIL.COM	307 CHP	09-FEB-08	MARK	DABNEY	PRODMANAGER
3	SI@GMAIL.COM	1234 COLL	12-MAY-99	KEVIN	POLO	EMPLOYEE
4	DD@YAHOO.COM	8021 OLIVE	01-JAN-94	MARSHAL	SMITH	ASSITMANAGER
5	HI@GMAIL.COM	26 BLW	08-OCT-92	ALBERT	SHAH	ADMINISTRATOR
6	CH@YAHOO.COM	21 MAIN	21-OCT-91	DOBBY	UDANI	DIRECTOR
7	AA@GMAIL.COM	556 CHP	31-DEC-84	JACKIE	SMITH	INTERN
8	PL@GMAIL.COM	32 JAI	12-APR-94	ARRON	DEBI	EXECUTIVE
9	I_9@YAHOO.COM	45 KRI	01-DEC-64	MEDRICK	ROCKY	COORDINATOR
10	SAB@GMAIL.COM	32 CRT	29-FEB-04	CASEY	NEISTAT	CONTROLLER
2	SH@GMAIL.COM	28 ATL	22-JUN-98	PEDRO	MAYA	MANAGER

10 rows selected.

VIEW STATEMENT:

CREATE OR REPLACE VIEW ALHAMBRA_CUSTOMERS AS
SELECT C_ID, C_SSN, C_EMAILID, C_ADD, CITY, ZIPCODE, STATE
FROM CUSTOMERS
WHERE CITY = 'ALHAMBRA' ;

OUTPUT: View Created.

SELECT * FROM ALHAMBRA_CUSTOMERS;

C_ID	C_SSN	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE
1	213526896	MON_2889@GMAIL.COM	102 ATL	ALHAMBRA	91810	CA
5	120589632	AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120	CA
11	213895296	ARIF4519@GMAIL.COM	801 OLIVE	ALHAMBRA	91754	CA

DROP VIEW ALHAMBRA_CUSTOMERS;

OUTPUT:View dropped.

SQL DCL STATEMENT:

FROM mphatar USER:
GRANT ALL ON CUSTOMERS TO MSHAH3;

OUTPUT:
Grant succeeded

FROM MSHAH3 USER:
SELECT * FROM mphatar.CUSTOMERS;

OUTPUT:

C_ID	C_DOB	C_SSN	C_EMAILID	C_ADD	CITY	ZIPCODE	STATE	C_PHONENUMBER
1	09-AUG-01	713232323	MON_2889@GMAIL.COM	102 ATL	ALHAMBRA	91810	CA	3235206321
2	12-AUG-87	213632563	DOBI_96@YAHOO.COM	307 CHAPEL	CITY TERRACE	91312	CA	2325698520
3	05-JUN-94	307256321	SABOO@CALSTATELA.EDU	32 BLV	EL MONTE	91456	CA	4589632145
4	09-MAY-96	307256123	CHAAP_LI@GMAIL.COM	806 COLL VIEW DRIVE	CITY TERRACE	91562	CA	2565412369
5	07-OCT-92	120589632	AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120	CA	1452367896
6	10-JAN-94	450201478	MONEY_12@YAHOO.COM	210 BALDWIN	SAN GABRIEL	91450	CA	4562223698
7	11-JUL-96	307654789	KU_12@GMAIL.COM	502 BLV	MONTEREY PARK	91300	CA	4562103698
8	10-MAR-92	563245789	N23@GMAIL.COM	203 KBP	MONTEBELLO	91125	CA	3232052314
9	01-APR-94	307399075	FLASH_345@YAHOO.COM	512 ALV	LUNAS	91801	CA	6265461230
10	05-FEB-04	458963210	WE_786@GMAIL.COM	1155 COLLEGE VIEW DRIVE	MONTEREY	91812	CA	3232050260
11	19-NOV-96	213895296	ARIF4519@GMAIL.COM	801 OLIVE	ALHAMBRA	91754	CA	3233433000
12	05-FEB-04	458963210	WE_786@GMAIL.COM	1155 COLLEGE VIEW DRIVE	MONTEREY	91812	CA	3232050260

12 rows selected.

9. Test your relational database via ORACLE SQL*PLUS, which includes SQL statements, SQL solutions and output and save them in the MS word file. Test with SELECT Statements. One member one SELECT…. Your SELECT statements must include join tables, subqueries, Group by …. Having and function statements.

1. **SQL JOIN: DISPLAY BUSINESS AND INDIVIDUAL CUSTOMERS**

   FOR BUSINESS CUSTOMERS:

   SELECT CUSTOMERS.C_ID, BUSINESS_CUSTOMER.BUS_C_ID,
   BUSINESS_CUSTOMER.BUS_NAME
   FROM BUSINESS_CUSTOMER
   INNER JOIN CUSTOMERS ON CUSTOMERS.C_ID=BUSINESS_CUSTOMER.BUS_C_ID
   ORDER BY C_ID;

   OUTPUT:

C_ID	BUS_C_ID	BUS_NAME
1	1	GLOBALTECH
2	2	VALUE SOLUTIONS
3	3	HIGHTECH
4	4	AVANADE
5	5	PETCO

FOR INDIVIDUAL CUSTOMERS:

```
SELECT CUSTOMERS.C_ID, INDIVIDUAL_CUSTOMER.IND_C_ID,
INDIVIDUAL_CUSTOMER.IND_FIRSTNAME,
INDIVIDUAL_CUSTOMER.IND_LASTNAME
FROM INDIVIDUAL_CUSTOMER
INNER JOIN CUSTOMERS ON CUSTOMERS.C_ID=INDIVIDUAL_CUSTOMER.IND_C_ID
ORDER BY C_ID;
```

OUTPUT:

C_ID	IND_C_ID	IND_FIRSTNAME	IND_LASTNAME
6	6	MONISH	RODRIQUEZ
7	7	MARY	JOHNSON
8	8	PETER	PHATARPEKAR
9	9	JEVEL	VAZ
10	10	SHERIN	DSOUZA

2. SQL SUBQUERIES:

```
SELECT I_ID, I_DESC, I_PRICE,I_QAUNTITY,(I_PRICE * I_QAUNTITY) AS TOTALCOST
FROM ITEMS
WHERE
I_QAUNTITY=(SELECT MAX(I_QAUNTITY) FROM ITEMS);
```

OUTPUT:

I_ID	I_DESC	I_PRICE	I_QAUNTITY	TOTALCOST
7	BOOKS	350	30	10500

3. Group by …. Having and function statements.

```
SELECT O_ID, COUNT(I_ID)
FROM ORDER_CHECK
GROUP BY O_ID
HAVING COUNT(I_ID)<2
ORDER BY O_ID;
```

OUTPUT:

O_ID	COUNT(I_ID)
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

10 rows selected.

4. List employee address, date of birth and names by their date of birth in the Descending order.

```
SELECT E_ADD, E_DOB, E_FIRSTNAME, E_LASTNAME
FROM EMPLOYEE
ORDER BY E_DOB DESC, E_FIRSTNAME DESC, E_LASTNAME DESC;
```

OUTPUT:

E_ADD	E_DOB	E_FIRSTNAME	E_LASTNAME
32 CRT	29-FEB-04	CASEY	NEISTAT
1234 COLL	12-MAY-99	KEVIN	POLO
32 JAI	12-APR-94	ARRON	DEBI
8021 OLIVE	01-JAN-94	MARSHAL	SMITH
26 BLW	08-OCT-92	ALBERT	SHAH
21 MAIN	21-OCT-91	DOBBY	UDANI
87 ATL	10-OCT-89	BOSCO	JOHNS
556 CHP	31-DEC-84	JACKIE	SMITH
1155 CVD	18-JUN-82	ROSS	SAHANI
1021 BLV	21-MAR-78	PEDRO	MAYA
45 KRI	01-DEC-64	MEDRICK	ROCKY

11 rows selected.

5. Retrieve the names of customers who are in Alhambra and City Terrace in descending order.

```
SELECT C_EMAILID, C_ADD, CITY, ZIPCODE
FROM CUSTOMER
WHERE CITY IN ('ALHAMBRA','CITY TERRACE')
ORDER BY CITY DESC;
```

OUTPUT:

C_EMAILID	C_ADD	CITY	ZIPCODE
DOBI_96@YAHOO.COM	307 CHAPEL	CITY TERRACE	91312
CHAAP_LI@GMAIL.COM	806 COLL VIEW DRIVE	CITY TERRACE	91562
AB_9@GMAIL.COM	89 MON	ALHAMBRA	91120
MON_2889@GMAIL.COM	ATL	ALHAMBRA	91810

6. What is the maximum, minimum and total cost of all the items that was sold order by I_ID

SELECT SUM(I_PRICE *I_QAUNTITY) AS COST, MAX(I_QAUNTITY) AS MAX,
MIN(I_QAUNTITY) AS MIN
FROM ITEMS
ORDER BY I_ID;

OUTPUT:

COST	MAX	MIN
70600	30	3

7. Determine the total number of small, medium and large scale orders placed.

select o_scale, count(o_scale) as "SCALE QUANTITY"
from ORDERS
group by o_scale
order by o_scale;

OUTPUT:

O_SCALE	SCALE QUANTITY
LARGE	4
MEDIUM	4
SMALL	2

11. PL/SQL STATEMENTS:

SQL DML STATEMENT FOR TABLE EMPLOYEE:

ALTER TABLE EMPLOYEE
ADD E_SAL NUMBER(7,2);

OUTPUT: Table Altered.

DATABASE STRUCTURE:

desc employee;

Name	Null?	Type
E_ID	NOT NULL	NUMBER(38)
E_EMAILID		VARCHAR2(255)
E_ADD		VARCHAR2(255)
E_DOB		DATE
E_FIRSTNAME		VARCHAR2(255)
E_LASTNAME		VARCHAR2(255)
E_JOBTITLE		VARCHAR2(255)
E_SAL		NUMBER(7,2)

SQL DML STATEMENT FOR TABLE EMPLOYEE:

```
UPDATE EMPLOYEE
SET E_SAL = '20000'
WHERE E_ID = 1;

UPDATE EMPLOYEE
SET E_SAL = '17000'
WHERE E_ID = 2;

UPDATE EMPLOYEE
SET E_SAL = '7000'
WHERE E_ID = 3;

UPDATE EMPLOYEE
SET E_SAL = '6500'
WHERE E_ID = 4;

UPDATE EMPLOYEE
SET E_SAL = '9000'
WHERE E_ID = 5;

UPDATE EMPLOYEE
SET E_SAL = '18500'
WHERE E_ID = 6;

UPDATE EMPLOYEE
SET E_SAL = '2500'
WHERE E_ID = 7;

UPDATE EMPLOYEE
SET E_SAL = '11000'
WHERE E_ID = 8;

UPDATE EMPLOYEE
SET E_SAL = '9500'
WHERE E_ID = 9;

UPDATE EMPLOYEE
SET E_SAL = '10000'
WHERE E_ID = 10;

UPDATE EMPLOYEE
SET E_SAL = '12000'
WHERE E_ID = 12;
```

OUTPUT: Rows Updated.

DATABASE STRUCTURE:

```
SELECT * FROM  EMPLOYEE;
```

OUTPUT:

E_ID	E_EMAILID	E_ADD	E_DOB	E_FIRSTNAME	E_LASTNAME	E_JOBTITLE	E_SAL
1	NI@GMAIL.COM	87 ATL	10-OCT-89	BOSCO	JOHNS	CEO	20000
2	SH@YAHOO.COM	1021 BLV	21-MAR-78	PEDRO	MAYA	MANAGER	17000
3	SI@GMAIL.COM	1234 COLL	12-MAY-99	KEVIN	POLO	EMPLOYEE	7000
4	DD@YAHOO.COM	8021 OLIVE	01-JAN-94	MARSHAL	SMITH	ASSITMANAGER	6500
5	HI@GMAIL.COM	26 BLW	08-OCT-92	ALBERT	SHAH	ADMINISTRATOR	9000
6	CH@YAHOO.COM	21 MAIN	21-OCT-91	DOBBY	UDANI	DIRECTOR	18500
7	AA@GMAIL.COM	556 CHP	31-DEC-84	JACKIE	SMITH	INTERN	2500
8	PL@GMAIL.COM	32 JAI	12-APR-94	ARRON	DEBI	EXECUTIVE	11000
9	I_9@YAHOO.COM	45 KRI	01-DEC-64	MEDRICK	ROCKY	COORDINATOR	9500
10	SAB@GMAIL.COM	32 CRT	29-FEB-04	CASEY	NEISTAT	CONTROLLER	10000
12	HIJACK@GMAIL.COM	1155 CVD	18-JUN-82	ROSS	SAHANI	JUNIOR ADMIN	12000

11 rows selected.

PROCEDURE WITH IN MODE PARAMETER FOR RAISING AN EMPLOYEE'S SALARY BY EMPLOYEE ID AND ENTERING THE DESIRED RAISE VALUE FOR TABLE EMPLOYEE:

```
CREATE OR REPLACE PROCEDURE adjust_salary
(
   in_E_id IN EMPLOYEE.E_ID%TYPE,
   in_percent IN NUMBER
)
IS
BEGIN

  UPDATE EMPLOYEE
  SET E_SAL = E_SAL + E_SAL * in_percent / 100
  WHERE E_ID = in_E_id;
END;
```

BEFORE ADJUSTMENT:

```
SELECT E_SAL FROM EMPLOYEE WHERE E_ID = 1;
```

OUTPUT:

E_SAL
20000

RUN: exec adjust_salary(1,5);

OUTPUT: PL/SQL procedure successfully completed.

AFTER ADJUSTMENT:

```
SELECT E_SAL FROM EMPLOYEE WHERE E_ID = 1;
```

OUTPUT:

E_SAL
21000

Procedure with IN and OUT modes parameters to Get an order id, order address and order scale by given order id.

```
CREATE OR REPLACE PROCEDURE QUERY_ORDER
 (F_ID  IN ORDERS.O_ID%TYPE,
```

```
  F_ADD  OUT ORDERS.O_ADD%TYPE,
  F_SCALE OUT ORDERS.O_SCALE%TYPE
)
 IS
BEGIN
   SELECT O_ADD, O_SCALE
   INTO F_ADD, F_SCALE
   from ORDERS
   WHERE O_ID =F_ID;
END QUERY_ORDER;
/
```

DECLARING GLOBAL VARIABLES:

```
variable F_ADD VARCHAR2(255)
variable F_SCALE VARCHAR2(255)
```

OUTPUT: SP2-0863: iSQL*Plus processing completed

RUN: EXECUTE QUERY_ORDER(4, :F_ADD, :F_SCALE)

OUTPUT: PL/SQL procedure successfully completed.

DISPLAY ON SCREEN:

PRINT F_ADD F_SCALE

OUTPUT:

F_ADD
LAS VEGAS

Next Page

F_SCALE
SMALL

FUNCTION : To count the total number of customers in the customer table.

```
CREATE OR REPLACE FUNCTION totalCustomers
(F_ID IN CUSTOMERS.C_ID%TYPE)
RETURN number IS
   total number(2) := 0;
BEGIN
   SELECT count(C_ID) into total
   FROM customers;

   RETURN total;
END;
/
```

OUTPUT: Function created.

DECLARING GLOBAL VARIABLES:

VARIABLE TOTAL_COUNT NUMBER

OUTPUT: SP2-0863: iSQL*Plus processing completed

RUN:

exec :TOTAL_COUNT := totalCustomers(0)

OUTPUT: PL/SQL procedure successfully completed

DISPLAY ON SCREEN:

PRINT TOTAL_COUNT

OUTPUT:

TOTAL_COUNT
10

FUNCTION: To display the ordered quantity of an order by its order id.

```
CREATE OR REPLACE FUNCTION ordered_quantity
(F_ID IN ORDER_CHECK.O_ID%TYPE) RETURN NUMBER
IS
O_ORDER  ORDER_CHECK.ORDEREDQUANTITY%TYPE :=  0;
BEGIN
SELECT ORDEREDQUANTITY
INTO O_ORDER
FROM ORDER_CHECK
WHERE O_ID =F_ID;
RETURN (O_ORDER);
END ordered_quantity;
```

OUTPUT: Function created.

DECLARING GLOBAL VARIABLES:

variable g_order number

OUTPUT: SP2-0863: iSQL*Plus processing completed

RUN:

execute :g_order := ordered_quantity(7)

OUTPUT: PL/SQL procedure successfully completed

DISPLAY ON SCREEN:

PRINT g_order

OUTPUT:

G_ORDER
55