Assignment 1

Contents

1. Create following banking database schema and insert appropriate data: 2
a) Design all tables with appropriate keys and constraints. 2
b) Find the names of all branches located in “Chicago”. 4
c) Add new customer in branch “Atlanta”, Consider appropriate values for other attributes. 4
d) Find all the account numbers having balance greater than $1000 4
e) Create a view to retrieve all the customers in “Atlanta” city with balance less than balance of customer “John”. (subquery) 5
f) Modify the relations so that the default branch city will be “Washington”. 5
g) Get the list of all customers only if the asset value for that branch is $35000 (EXISTS clause). 5
2. Create Insurance database and insert appropriate data: 6
a) Design all tables with appropriate keys and constraints. 6
b) Add all appropriate foreign keys. 6
c) Add a new accident to the database; assume any values for required attributes 7
d) Find report numbers for all the accidents in which the cars belonging to “John Smith” were involved. (subquery) 8
e) Delete the Mazda belonging to “John Smith”. 8
f) Create a view to get date and locations of all accidents. 8
g) Find the name and addresses of owners of cars involved in accidents for which damage amount is greater than $2000. 8
h) Add car prices in the appropriate relation. 9
i) Create view to have the information of persons and their cars 9
j) Rename the name of the table person to owners. 9
k) Change the type of attribute damage amount from int to numeric. 9
l) Delete all values in accident table but keep the relation in schema. 10
m) Modify the accident table to increase the report number by one for each new insertion. 10
3. Write a query to calculate 123-56*3+23. 11

1. Create following banking database schema and insert appropriate data:

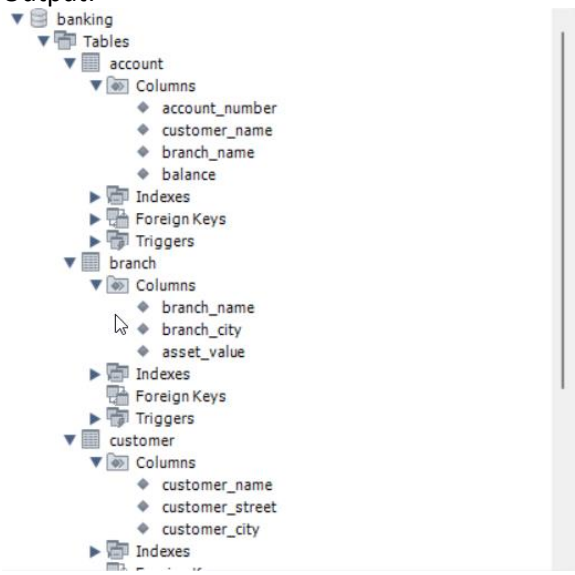
branch (branch name, branch city, asset_value)
customer (customer name, customer street, customer city)
account (account number, customer name, branch name, balance)

a) Design all tables with appropriate keys and constraints.

Solution:

```
-- Create new database called 'banking' and use it
CREATE DATABASE banking;
USE banking;

-- a) Design all tables with appropriate keys and constraints.
-- Create table 'branch' with 'branch_name' as primary key
CREATE TABLE branch(branch_name VARCHAR(50) PRIMARY KEY,
                    branch_city VARCHAR(50),
                    asset_value BIGINT NOT NULL);

-- Create table for 'customer' with 'customer_name' as primary key, assiming no two or more customers can have
same value
CREATE TABLE customer(customer_name VARCHAR(50) PRIMARY KEY,
                      customer_street VARCHAR(50),
                      customer_city VARCHAR(50) NOT NULL);

-- Create 'account' table with account_number as primary key,
-- customer name is referenced from customer table, branch name is referenced from branch table as foreign key,
-- default balance for new accounts is 0
CREATE TABLE account(account_number INT PRIMARY KEY,
                     customer_name VARCHAR(50) NOT NULL,
                     branch_name VARCHAR(50),
                     balance DOUBLE DEFAULT 0,
                     FOREIGN KEY fk_cust_name(customer_name) REFERENCES customer(customer_name),
                     FOREIGN KEY fk_branch_nmae(branch_name) REFERENCES branch(branch_name));

-- Insert data into the tables
INSERT INTO branch VALUES
            ('Chi-1', 'Chicago', 10000),
            ('Chi-2', 'Chicago', 20000),
            ('Atl-1', 'Atlanta', 25000),
            ('Atl-2', 'Atlanta', 23000),
            ('Was-1', 'Washington', 35000),
            ('Was-2', 'Washington', 36000),
            ('Nyc-1', 'New York', 45000),
            ('Nyc-2', 'New York', 46000);

-- Insert data into the tables
INSERT INTO customer VALUES
            ('John', 'Streat 1', 'New York'),
            ('Nick', 'Atl street 2', 'Atlanta'),
            ('Harry', 'First street', 'Chicago'),
            ('Ron', 'North street', 'Washington'),
            ('David', 'Second street', 'Chicago'),
            ('Natasha', 'Green lane', 'Atlanta');

-- Insert data into account table
INSERT INTO account VALUES
            (1, 'John', 'Nyc-1', 1400),
            (2, 'Nick', 'Atl-2', 1800),
            (3, 'Harry', 'Chi-1', 1000),
            (4, 'Ron', 'Was-1', 500),
            (5, 'David', 'Chi-2', 1500),
            (6, 'Natasha', 'Atl-1', 1100);
```

Output:



Inserted data:

Branch

branch_name	branch_city	asset_value
Atl-1	Atlanta	25000
Atl-2	Atlanta	23000
Chi-1	Chicago	10000
Chi-2	Chicago	20000
Nyc-1	New York	45000
Nyc-2	New York	46000
Was-1	Washington	35000
Was-2	Washington	36000
NULL	NULL	NULL

account

account_number	customer_name	branch_name	balance
1	John	Nyc-1	1400
2	Nick	Atl-2	1800
3	Harry	Chi-1	1000
4	Ron	Was-1	500
5	David	Chi-2	1500
6	Natasha	Atl-1	1100
7	Tony	Atl-2	1200
NULL	NULL	NULL	NULL

Customer

Result Grid | Filter Rows:

customer_name	customer_street	customer_city
David	Second street	Chicago
Harry	First street	Chicago
John	Street 1	New York
Natasha	Green lane	Atlanta
Nick	Atl street 2	Atlanta
Ron	North street	Washington
NULL	NULL	NULL

b) Find the names of all branches located in “Chicago”.

Solution:

```
SELECT branch_name FROM branch WHERE branch_city='Chicago';
```

Output:

branch

branch_name
Chi-1
Chi-2

c) Add new customer in branch “Atlanta”, Consider appropriate values for other attributes.

Solution:

```
INSERT INTO customer VALUES ('Tony', 'Silver street', 'Atlanta');
INSERT INTO account VALUES (7, 'Tony', 'Atl-2', 1200);
```

Output:

account

account_number	customer_name	branch_name	balance
1	John	Nyc-1	1400
2	Nick	Atl-2	1800
3	Harry	Chi-1	1000
4	Ron	Was-1	500
5	David	Chi-2	1500
6	Natasha	Atl-1	1100
7	Tony	Atl-2	1200

d) Find all the account numbers having balance greater than $1000

Solution:

```
SELECT * FROM account WHERE balance > 1000;
```

Output:

account

account_number	customer_name	branch_name	balance
1	John	Nyc-1	1400
2	Nick	Atl-2	1800
5	David	Chi-2	1500
6	Natasha	Atl-1	1100
7	Tony	Atl-2	1200

e) Create a view to retrieve all the customers in “Atlanta” city with balance less than balance of customer “John”. (subquery)

Solution:

```
CREATE VIEW bal_less_than_john_atlanta AS
        SELECT * FROM account WHERE
                balance < (SELECT balance FROM account WHERE customer_name='John') AND
                branch_name IN (SELECT branch_name FROM branch WHERE branch_city='Atlanta');

SELECT * FROM bal_less_than_john_atlanta;
```

Output:

bal_less_than_john_atlanta

Result Grid | Filter Rows: | Export:

account_number	customer_name	branch_name	balance
6	Natasha	Atl-1	1100
7	Tony	Atl-2	1200

f) Modify the relations so that the default branch city will be “Washington”.

Solution:

```
ALTER TABLE branch ALTER branch_city SET DEFAULT 'Washington';
```

g) Get the list of all customers only if the asset value for that branch is $35000 (EXISTS clause).

Solution:

```
SELECT * FROM customer WHERE EXISTS (SELECT asset_value FROM branch WHERE asset_value = 35000)
```

Output:

customer

Result Grid | Filter Rows: | E

customer_name	customer_street	customer_city
David	Second streat	Chicago
Harry	First street	Chicago
John	Streat 1	New York
Natasha	Green lane	Atlanta
Nick	Atl streat 2	Atlanta
Ron	North streat	Washington
Tony	Silver streat	Atlanta

2. Create Insurance database and insert appropriate data:

person (driver id, name, address)
car (license, model, year)
accident (report number, location)
owns (driver id, license)
participated (report number, license, driver id, damage amount)

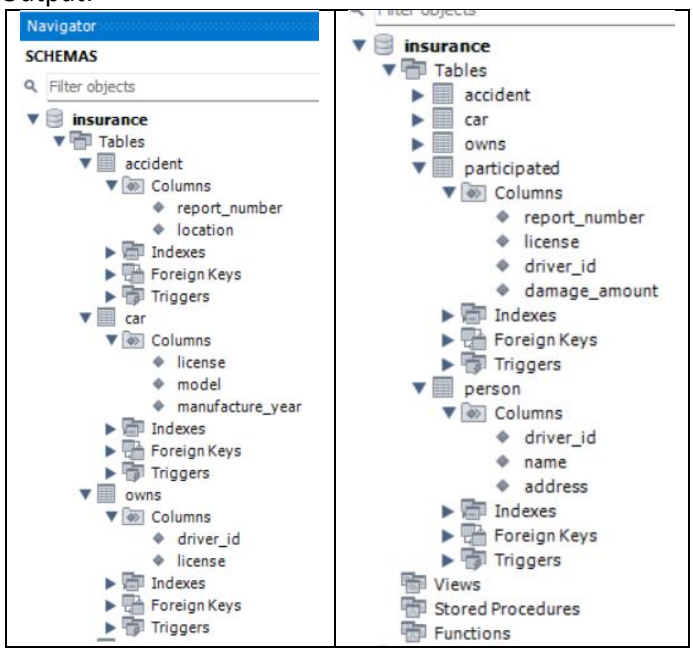
a) Design all tables with appropriate keys and constraints.

b) Add all appropriate foreign keys.

Solution:

```
-- Create database 'insurance' and use it
CREATE DATABASE insurance;
USE insurance;

-- Create tables for person, car, accident, owns and participated
CREATE TABLE person(driver_id INT PRIMARY KEY, name VARCHAR(50) NOT NULL, address VARCHAR(100) NOT NULL);

CREATE TABLE car (license VARCHAR(10) PRIMARY KEY, model VARCHAR(50) NOT NULL, manufacture_year YEAR);

CREATE TABLE owns (driver_id INT NOT NULL, license VARCHAR(10) NOT NULL UNIQUE,
    FOREIGN KEY fk_own_driver_id(driver_id) REFERENCES person(driver_id),
    FOREIGN KEY fk_own_license(license) REFERENCES car(license));

CREATE TABLE participated (report_number INT PRIMARY KEY, license VARCHAR(10) NOT NULL,
    driver_id INT NOT NULL, damage_amount INT NOT NULL DEFAULT 0,
             FOREIGN KEY fk_part_driver_id(driver_id) REFERENCES person(driver_id),
    FOREIGN KEY fk_part_license(license) REFERENCES car(license));

CREATE TABLE accident (report_number INT PRIMARY KEY, location VARCHAR(50),
    FOREIGN KEY fk_acc_report_number(report_number) REFERENCES participated(report_number));

-- Insert data
INSERT INTO person VALUES
    (1, 'Raj', 'Indore'),
    (2, 'Ajay', 'Nagpur'),
    (3, 'Bala', 'Pune'),
    (4, 'John Smith', 'Agra');

INSERT INTO car VALUES
    ('A1B2C1', 'Maruti', 2010),
    ('A1B2C2', 'Honda', 2011),
    ('A1B2C3', 'Mazda', 2012),
    ('A1B2C4', 'Hyundai', 2020),
    ('A2B3C4', 'Honda', 2021),
    ('M1M2M3', 'Mazda', 2019);

INSERT INTO owns VALUES
   (1, 'A1B2C1'),
   (2, 'A1B2C2'),
   (3, 'A1B2C3'),
   (2, 'A1B2C4'),
   (4, 'A2B3C4'),
   (4, 'M1M2M3');

INSERT INTO participated VALUES
   (1, 'A2B3C4', 2, 1500),
   (2, 'A1B2C2', 1, 1000),
   (3, 'A2B3C4', 2, 2100);

INSERT INTO accident VALUES
    (1, 'Noida'),
    (2, 'Delhi'),
    (3, 'Patna');
```

Output:



Inserted data:

person

driver_id	name	address
1	Raj	Indore
2	Ajay	Nagpur
3	Bala	Pune
4	John Smith	Agra
NULL	NULL	NULL

car

license	model	manufacture_year
A1B2C1	Maruti	2010
A1B2C2	Honda	2011
A1B2C3	Mazda	2012
A1B2C4	Hyundai	2020
A2B3C4	Honda	2021
M1M2M3	Mazda	2019
NULL	NULL	NULL

owns

driver_id	license
1	A1B2C1
2	A1B2C2
2	A1B2C4
3	A1B2C3
4	A2B3C4
4	M1M2M3
NULL	NULL

participated

report_number	license	driver_id	damage_amount
1	A2B3C4	2	1500
2	A1B2C2	1	1000
3	A2B3C4	2	2100
NULL	NULL	NULL	NULL

accident

report_number	location
1	Noida
2	Delhi
3	Patna
NULL	NULL

c) Add a new accident to the database; assume any values for required attributes

Solution:

```
-- c) Add a new accident to the database; assume any values for required attributes
INSERT INTO participated VALUES(4, 'A1B2C1', 2, 20);
INSERT INTO accident VALUES (4, 'Delhi');
```

Output:

participated

report_number	license	driver_id	damage_amount
1	A2B3C4	2	1500
2	A1B2C2	1	1000
3	A2B3C4	2	2100
4	A1B2C1	2	20
NULL	NULL	NULL	NULL

accident

report_number	location
1	Noida
2	Delhi
3	Patna
4	Delhi
NULL	NULL

d) Find report numbers for all the accidents in which the cars belonging to “John Smith” were involved. (subquery)

Solution:

```
SELECT report_number FROM participated WHERE
    license IN (SELECT license FROM owns WHERE
        driver_id = (SELECT driver_id FROM person WHERE name = 'John Smith'));
```

Output:

participated

report_number
1
3
NULL

e) Delete the Mazda belonging to “John Smith”.

Solution:

```
DELETE FROM owns WHERE
    driver_id = (SELECT driver_id FROM person WHERE name = 'John Smith') AND
    license IN (SELECT license FROM car WHERE model = 'mazda');
```

Output:

owns

driver_id	license
1	A1B2C1
2	A1B2C2
2	A1B2C4
3	A1B2C3
4	A2B3C4
NULL	NULL

f) Create a view to get date and locations of all accidents.

Solution:

```
CREATE VIEW dam_and_location AS SELECT accident.location, participated.damage_amount FROM accident, participated
WHERE accident.report_number = participated.report_number;
```

Output:

dam_and_location

location	damage_amount
Noida	1500
Delhi	1000
Patna	2100
Delhi	20

g) Find the name and addresses of owners of cars involved in accidents for which damage amount is greater than $2000.

Solution:

```
SELECT name, address FROM person WHERE driver_id = (SELECT driver_id FROM owns WHERE license = (SELECT license
FROM participated WHERE damage_amount > 2000));
```

Output:

name	address
John Smith	Agra

h) Add car prices in the appropriate relation.

Solution:

```
ALTER TABLE car ADD COLUMN price INT DEFAULT 5000;
```

Output:

DESCRIBE car

Field	Type	Null	Key	Default	Extra
license	varchar(10)	NO	PRI	NULL	
model	varchar(50)	NO		NULL	
manufacture_year	year	YES		NULL	
price	int	YES		5000	

i) Create view to have the information of persons and their cars .

Solution:

```
CREATE VIEW person_car AS SELECT person.driver_id, person.name, person.address, car.license, car.model,
car.manufacture_year, car.price FROM person, car, owns
        WHERE person.driver_id = owns.driver_id AND car.license = owns.license;
SELECT * FROM person_car;
```

Output:

person_car

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

driver_id	name	address	license	model	manufacture_year	price
1	Raj	Indore	A1B2C1	Maruti	2010	5000
2	Ajay	Nagpur	A1B2C2	Honda	2011	5000
2	Ajay	Nagpur	A1B2C4	Hyundai	2020	5000
3	Bala	Pune	A1B2C3	Mazda	2012	5000
4	John Smith	Agra	A2B3C4	Honda	2021	5000

j) Rename the name of the table person to owners.

Solution:

```
RENAME TABLE person to owners;
```

Output:

DESCRIBE owners

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
driver_id	int	NO	PRI	NULL	
name	varchar(50)	NO		NULL	
address	varchar(100)	NO		NULL	

k) Change the type of attribute damage amount from int to numeric.

Solution:

```
ALTER TABLE participated MODIFY damage_amount NUMERIC;
```

Output:

DESCRIBE participated

Result Grid | Filter Rows: | Export: | Wrap Cell

Field	Type	Null	Key	Default	Extra
report_number	int	NO	PRI	NULL	
license	varchar(10)	NO	MUL	NULL	
driver_id	int	NO	MUL	NULL	
damage_amount	decimal(10,0)	YES		NULL	

l) Delete all values in accident table but keep the relation in schema.

Solution:

```
SET SQL_SAFE_UPDATES = 0;
DELETE FROM accident;
```

Output:

accident

Result Grid | Filter Rows

report_number	location
NULL	NULL

m) Modify the accident table to increase the report number by one for each new insertion.

Solution:

```
ALTER TABLE accident MODIFY report_number INT AUTO_INCREMENT;
```

Output:

Error:

```
Error Code: 1832. Cannot change column 'report_number': used in a foreign key constraint 'accident_ibfk_1'
```

Explanation:

Changing column report number from accident table is not allowed because it is a foreign key referring to report number from participated table.

```
CREATE TABLE accident (report_number INT PRIMARY KEY, location VARCHAR(50),
    FOREIGN KEY fk_acc_report_number(report_number) REFERENCES participated(report_number));
```

3. Write a query to calculate 123-56*3+23.

Solution:

```
SELECT  123-56*3+23
```

Output:

Result Grid

	123-56*3+23
▶	-22

========== End of Assignment ==========