SQL PROJECT

--DATABASE NAME:-

```
USE MotorsCertification;
```

--DELETING NULL ROWS AND COLUMNS FROM ORDERS TABLE:-

```
DELETE FROM orders
WHERE orderNumber IS NULL
AND orderDate IS NULL
AND requiredDate IS NULL
AND shippedDate IS NULL
AND status IS NULL
AND comments IS NULL
AND customerNumber IS NULL;
```

--DELETING ROWS AND COLUMNS IN PAYMENTS:-

```
DELETE FROM payments
WHERE customerNumber IS NULL
AND checkNumber IS NULL
AND paymentDate IS NULL
AND amount IS NULL
```

--DELETING ROWS AND COLUMNS IN PRODUCTS TABLE:-

```
DELETE FROM products
WHERE productCode IS NULL
AND productName IS NULL
AND productLine IS NULL
AND productScale IS NULL
AND productVendor IS NULL
AND productDescription IS NULL
AND quantityInStock IS NULL
AND buyPrice IS NULL
AND MSRP IS NULL
SELECT * FROM products
```

--Adding primarykey to customers table:-

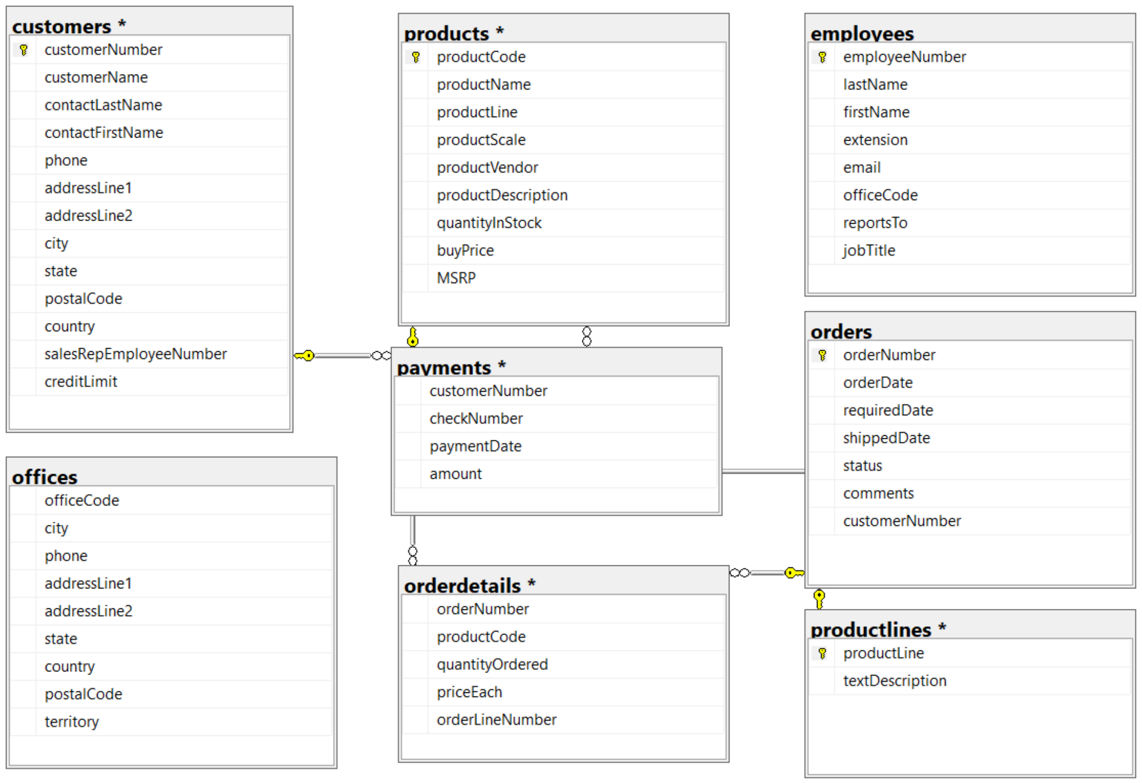
```
ALTER TABLE customers
add constraint pk_cus primary key (customerNumber)
```

--Adding primary key to employees table:-

```
ALTER TABLE employees
ADD PRIMARY KEY (employeeNumber)
```

```
--Converting null value to not null value:-

ALTER TABLE orders ALTER COLUMN orderNumber INT NOT NULL;

--Adding Primary Key for orders table:-

ALTER TABLE orders
add constraint PK_ORD PRIMARY KEY (orderNumber)

--Customizing orderdetails table:-

ALTER TABLE OrderDetails
ALTER COLUMN orderNumber INT;

ALTER TABLE OrderDetails
ADD CONSTRAINT FK_Order FOREIGN KEY (orderNumber) REFERENCES
Orders(orderNumber);

--Customizing products table:-

ALTER TABLE products ALTER COLUMN productCode varchar(50) not null;
ALTER TABLE orderdetails ALTER COLUMN productCode varchar(50) not
null;

ALTER TABLE products
ADD CONSTRAINT pk_pc PRIMARY KEY (productCode)

ALTER TABLE orderdetails
ADD CONSTRAINT FK_2 FOREIGN KEY (productCode) REFERENCES
products(productCode);
```

--3)Provide comments before every task that is performed describing the operation that is being performed and attach a screenshot of ER diagram from SSMS.



-- 4)Deleting the columns in productline:-

```
ALTER TABLE productlines
DROP COLUMN htmlDescription;

ALTER TABLE productlines
DROP COLUMN image
```

--5)Select statement to verify all insertions as well as updates:-

```
select * from customers;
select * from employees;
select * from offices;
select * from orderdetails;
select * from orders;
select * from payments;
select * from productlines;
select * from products;
```

--6)Highest and Lowest amount:-

```
SELECT * FROM payments

select MAX(amount) as Highest
from payments

select min(amount) as Lowest
from payments
```

--7)Unique count of customer name from customer:-

```
SELECT * FROM customers;
SELECT COUNT(DISTINCT customerName) AS unique_customer_count
FROM customers;
```

--8)Create view from customers and payments named cus_payment and select customerName, amount,contactLastName, contactFristName:-

```
CREATE VIEW cus_payment AS
SELECT
customers.customerName,
customers.contactLastName,
customers.contactFirstName,
payments.amount
FROM
customers
JOIN
payments
ON
customers.customerNumber = payments.customerNumber;

SELECT * FROM cus_payment

DROP VIEW cus_payment
```

--9)Create a stored procedure on products which displays productLine for Classic Cars:-

```
SELECT * FROM products

CREATE PROCEDURE GetClassicCars
AS
BEGIN
SELECT productLine
FROM products
WHERE productLine = 'Classic Cars';
End;

go
EXEC GetClassicCars;
```

--10)Create a function to get the creditLimit of customers less than 96800:-

```
SELECT * FROM customers
WHERE creditLimit < 96800;
```

--11.Create Trigger to store transaction record for employee table which displays employeeNumber, lastName, FirstName and office code upon insertion:-

```
CREATE TABLE employees_audit
(
employeeNumber smallint not null,
lastName nvarchar(50) not null,
firstName nvarchar(50) not null,
officeCode nvarchar(50) not null
)

CREATE TRIGGER trg_emp_audit
ON employees
AFTER INSERT
AS
BEGIN
INSERT INTO employees_audit
SELECT employeeNumber,lastName,firstName,officeCode
FROM inserted
END

select * from employees;

insert into employees
(employeeNumber,lastName,firstName,extension,email,officeCode,reportsT
o,jobTitle)Values
(1003,'Prayan','Mikku','x6001','mikku@classicmodelcars.com',1,5555,'Ch
airman')

select * employees_audit
```

--12)Create a Trigger to display customer number if the amount is greater than 10,000:-

```
SELECT * FROM payments;
CREATE TABLE DIS_CUSNO
(customerNumber int,
amount int);

create trigger trig_amount
ON payments
AFTER INSERT
AS
BEGIN
INSERT INTO DIS_CUSNO
SELECT customerNumber,amount
from inserted
where (amount>10000);
```

```
end;

INSERT INTO payments (customerNumber,amount)
VALUES (110,11120),
(123,123444),
(125,9999);

select * from DIS_CUSNO;
```

--13.Create Users, Roles and Logins according to 3 Roles: Admin, HR, and Employee. Admin can view full database and has full access, HR can view and access only employee and offices table. Employee can view all tables only.

-- Step 1: Create Logins

```
CREATE LOGIN AdminLogin WITH PASSWORD = 'StrongPassword1!';
CREATE LOGIN HRLogin WITH PASSWORD = 'StrongPassword2!';
CREATE LOGIN EmployeeLogin WITH PASSWORD = 'StrongPassword3!';
```

-- Step 2: Create Users

```
CREATE USER AdminUser FOR LOGIN AdminLogin;
CREATE USER HRUser FOR LOGIN HRLogin;
CREATE USER EmployeeUser FOR LOGIN EmployeeLogin;
```

-- Step 3: Create Roles

```
CREATE ROLE AdminRole;
CREATE ROLE HRRole;
CREATE ROLE EmployeeRole;
```

-- Step 4: Assign Users to Roles

```
ALTER ROLE AdminRole ADD MEMBER AdminUser;
ALTER ROLE HRRole ADD MEMBER HRUser;
ALTER ROLE EmployeeRole ADD MEMBER EmployeeUser;
```

-- Step 5: Grant Permissions
-- AdminRole: Full access

```
GRANT CONTROL ON DATABASE::MotorsCertification TO AdminRole;
```

-- HRRole: Access to Employee and Offices tables

```
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.employees TO HRRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.offices TO HRRole;
```

-- EmployeeRole: Read-only access to all tables

```
GRANT SELECT ON SCHEMA::dbo TO EmployeeRole;
```

END