**Database Name – sonikaprakashs**

**ERD**

**A picture containing text, sky, day

Description automatically generated**

**Creating Person Table**

CREATE TABLE Person

(PersonID INT NOT NULL Primary Key,

LastName VARCHAR (30) NOT NULL,

FirstName VARCHAR (30) NOT NULL,

PhoneNumber CHAR (15) NOT NULL,

PersonType VARCHAR (1) NOT NULL);

SELECT \* FROM Person

**Creating Model Table**

CREATE TABLE Model

(ModelNumber CHAR (6) NOT NULL Primary Key,

ModelName VARCHAR (30) NOT NULL,

ModelDescription VARCHAR (100) NOT NULL,

StandardPrice Money NOT NULL);

SELECT \* FROM Model

**Creating AllProblem Table**

CREATE TABLE AllProblem

(TestID INT NULL,

TestDate DateTime NULL,

TestDescription VARCHAR (200) NULL,

TestResults VARCHAR (300) NULL,

TestComplete VARCHAR (1) NULL,

TesterID INT NULL,

ReportID INT NULL,

RelatedTestID INT NULL,

ProblemReportReportID INT NULL,

ReportDate DateTime NULL,

CompleteDate Datetime NULL,

ProblemDescription VARCHAR (100) NULL,

InjuryYN VARCHAR (3) NULL,

InjuryDescription VARCHAR (80) NULL,

ProblemReportSerialNumber CHAR (10) NULL,

ProblemReportProblemTypeID INT NULL,

ReporterID INT NULL,

ProblemTypeID INT NULL,

TypeDescription VARCHAR (100) NULL,

SerialNumber CHAR (10),

ToyModelNumber CHAR (6) NULL,

OwnerID INT NULL,

PricePaid Money NULL);

SELECT \* FROM AllProblem

**Creating Toy Table**

CREATE TABLE Toy

(SerialNumber CHAR (10) Primary Key NOT NULL,

ModelNumber Char (6) Foreign Key References Model (ModelNumber) NOT NULL,

OwnerID INT Foreign Key References Person (PersonID));

INSERT INTO Toy (SerialNumber, ModelNumber, OwnerID, PricePaid)

SELECT DISTINCT SerialNumber, ToyModelNumber, OwnerID, PricePaid

FROM AllProblem

WHERE SerialNumber IS NOT NULL;

ALTER TABLE Toy

ADD PricePaid MONEY NULL

SELECT \* FROM Toy

**Creating ProblemReportType Table**

CREATE TABLE ProblemReportType

(ProblemReportTypeID INT Primary Key NOT NULL,

TypeDescription VARCHAR (100) NOT NULL);

INSERT INTO ProblemReportType (ProblemReportTypeID, TypeDescription)

SELECT DISTINCT ProblemTypeID, TypeDescription

FROM AllProblem

WHERE ProblemTypeID IS NOT NULL;

SELECT \* FROM ProblemReportType

**Creating ProblemReport Table**

CREATE TABLE ProblemReport

(ProblemReportID INT NOT NULL Primary Key,

ReportDate DateTime NOT NULL,

CompleteDate DateTime,

ProblemDescription VARCHAR (100) NOT NULL,

InjuryYN VARCHAR (3) NOT NULL,

InjuryDescription VARCHAR (80),

SerialNumber CHAR (10) Foreign Key References Toy (SerialNumber) NOT NULL,

ProblemReportTypeID INT Foreign Key References ProblemReportType (ProblemReportTypeID) NOT NULL,

ReporterID INT Foreign Key References Person (PersonID) NOT NULL);

INSERT INTO ProblemReport

(ProblemReportID, ReportDate, CompleteDate, ProblemDescription, InjuryYN, InjuryDescription, SerialNumber,

ProblemReportTypeID, ReporterID)

SELECT DISTINCT ProblemReportReportID, ReportDate, CompleteDate,

ProblemDescription, InjuryYN, InjuryDescription, ProblemReportSerialNumber, ProblemReportProblemTypeID,

ReporterID

FROM AllProblem

WHERE ProblemReportReportID IS NOT NULL

ORDER BY ProblemReportReportID;

SELECT \* FROM ProblemReport

**Creating Test Table**

CREATE TABLE Test

(TestID INT Primary Key NOT NULL,

TestDate DateTime NOT NULL,

TestDescription VARCHAR (300) NOT NULL,

TestResults VARCHAR (200),

TestComplete VARCHAR (1) NOT NULL,

TesterID INT Foreign Key References Person (PersonID),

ReportID INT Foreign Key References ProblemReport (ProblemReportID),

RelatedTestID INT Foreign Key References Test (TestID));

INSERT INTO Test

(TestID, TestDate, TestDescription, TestResults, testComplete, testerID, ReportID, RelatedTestID)

SELECT DISTINCT

TestID, TestDate, TestDescription, TestResults, testComplete, testerID, ReportID, RelatedTestID

FROM AllProblem

WHERE TestID IS NOT NULL;

SELECT \* FROM Test

**Solving Queries-**

**--Query 1**

SELECT \* FROM Person;

Calendar

Description automatically generated with low confidence

SELECT \* FROM Model;

Graphical user interface, text

Description automatically generated

SELECT \* FROM Toy;

Table

Description automatically generated

SELECT \* FROM ProblemReportType;

Graphical user interface, text, application

Description automatically generated

SELECT \* FROM ProblemReport;

Table

Description automatically generated

SELECT \* FROM Test;

Text

Description automatically generated with low confidence

**--Query 2**

SELECT PR.ProblemReportID as ReportID, PR.ReportDate, PR.CompleteDate, PR.ProblemDescription,

p.LastName as ReporterLastName, t.ModelNumber, M.ModelName, t.PricePaid

FROM ProblemReport PR

INNER JOIN Person P

ON PersonID = ReporterID

INNER JOIN Toy t

ON PR.SerialNumber = t.SerialNumber

INNER JOIN Model M

ON t.ModelNumber = m.ModelNumber

ORDER BY PR.ProblemReportID;

Graphical user interface, table

Description automatically generated

**--Query3**

SELECT pr.ProblemReportID as ReportID, SerialNumber, ReportDate, CompleteDate, InjuryYN, InjuryDescription,

ProblemDescription

FROM ProblemReport pr

WHERE InjuryYN = 'Yes' and ProblemReportID NOT IN (SELECT ReportID FROM Test) and

CompleteDate IS NOT NULL;



**--Query 4**

SELECT pr.ProblemReportID as ReportID, pr.SerialNumber, ReportDate, CompleteDate,

p.LastName as ReporterLastName, m.ModelName,

InjuryYN, InjuryDescription,

ProblemDescription

FROM ProblemReport pr

INNER JOIN Toy t

ON t.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON M.ModelNumber = t.ModelNumber

INNER JOIN Person p

ON P.personID = pr.ReporterID

WHERE InjuryYN = 'Yes' and ProblemReportID NOT IN (SELECT ReportID FROM Test) and

CompleteDate IS NOT NULL;



**--Query 5**

SELECT prt.ProblemReportTypeID as ProblemTypeID, prt.TypeDescription, COUNT(ProblemReportID) as CountOfReports,

isnull(ic.CountofInjuryReports,0) as CountOfInjuryReports

FROM ProblemReportType prt

LEFT OUTER JOIN ProblemReport pr

ON prt.ProblemReportTypeID = pr.ProblemReportTypeID

LEFT OUTER JOIN InjuryCounts ic

ON ic.ProblemReportTypeID = pr.ProblemReportTypeID

GROUP BY prt.ProblemReportTypeID, prt.TypeDescription, ic.CountofInjuryReports

ORDER BY prt.ProblemReportTypeID;

CREATE VIEW InjuryCounts as

(SELECT COUNT(InjuryYN) as CountOfInjuryReports, ProblemReportTypeID

FROM ProblemReport

WHERE InjuryYN = 'yes'

GROUP BY ProblemReportTypeID);

Graphical user interface, text, application

Description automatically generated

**--Query 6**

SELECT CONVERT(VARCHAR,pr.ReportDate,107) as ReportDateOutput,

pr.ProblemReportID as ReportID, pr.SerialNumber as Serial#,

(p.LastName + ', ' + UPPER(LEFT(p.FirstName, 1))) as OwnerName,

isnull(CONVERT(VARCHAR, pr.CompleteDate, 107), 'Not Complete') as CompleteDate,

DATEDIFF(day,ReportDate, isnull(CompleteDate,GETDATE())) as DaysInSystem,

m.ModelNumber as Model#, m.ModelDescription,

(pt.LastName + ', ' + UPPER(LEFT(pt.FirstName, 1))) as TesterName,

CONVERT(VARCHAR, tt.TestDate, 107) as TestDate, tt.TestDescription, tt.TestComplete

FROM ProblemReport pr

INNER JOIN Toy t

ON t.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON m.ModelNumber = t.ModelNumber

LEFT OUTER JOIN Test tt

ON tt.ReportID = pr.ProblemReportID

INNER JOIN Person p

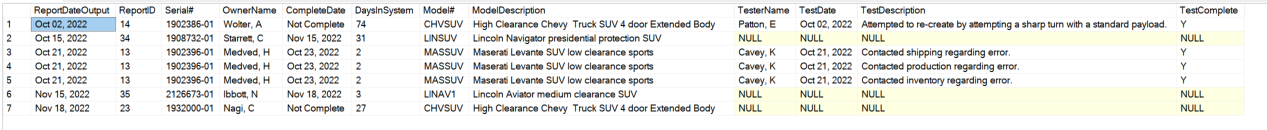
ON p.personID = t.OwnerID

LEFT OUTER JOIN Person pt

ON pt.PersonID = tt.testerID

WHERE ModelDescription LIKE '%SUV%'

ORDER BY ReportDate;



**--Query 7**

SELECT m.ModelNumber, COUNT(ProblemReportID) as CountOfProblemReports

FROM ProblemReport pr

INNER JOIN Toy t

ON t.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON m.ModelNumber = t.ModelNumber

GROUP BY m.ModelNumber;

Table

Description automatically generated

**--Query 8**

CREATE VIEW MaximumProblem as

(SELECT m.ModelNumber, COUNT(ProblemReportID) as CountOfProblemReports

FROM ProblemReport pr

INNER JOIN Toy t

ON t.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON m.ModelNumber = t.ModelNumber

GROUP BY m.ModelNumber);

SELECT m.ModelNumber, ModelName, ModelDescription, CountOfProblemReports

FROM Model m

INNER JOIN MaximumProblem mp

ON mp.ModelNumber = m.ModelNumber

WHERE CountOfProblemReports = (SELECT MAX(CountOfProblemReports) FROM MaximumProblem);



**--Query 9**

SELECT m.ModelNumber, m.ModelDescription, isnull(cp.CountOfReports,0) as CountOfReports,

isnull(icm.CountOfInjuryReports, 0) as CountOfInjuryReports,

isnull(CONVERT(VARCHAR, MAX(ReportDate), 107), 'n/a') as MostRecentReportDate,

isnull(CONVERT(VARCHAR, MIN(ReportDate), 107), 'n/a') as EarliestReportDate,

COUNT(tt.TestID) as CountOfTests,

isnull(CONVERT(VARCHAR, MAX(TestDate), 107), 'n/a') as MostRecentTestDate,

isnull(CONVERT(VARCHAR, MIN(TestDate), 107), 'n/a') as EarliestTestDate

FROM ProblemReport pr

LEFT OUTER JOIN Toy t

ON t.SerialNumber = pr.SerialnUmber

RIGHT OUTER JOIN Model m

ON m.ModelNumber = t.ModelNumber

LEFT OUTER JOIN InjuryCountsbyModel icm

ON icm.ModelNumber = t.ModelNumber

LEFT OUTER JOIN Test tt

ON tt.ReportID = pr.ProblemReportID

LEFT OUTER JOIN CountsofReports cp

ON cp.ModelNumber = m.ModelNumber

GROUP BY m.ModelNumber, m.ModelDescription, icm.CountOfInjuryReports, cp.CountOfReports

ORDER BY m.ModelNumber;

CREATE VIEW CountsofReports as

(SELECT COUNT(Pr.ProblemReportID) as CountOfReports, m.ModelNumber

FROM ProblemReport pr

INNER JOIN Toy tt

ON tt.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON m.ModelNumber = tt.ModelNumber

Group by m.ModelNumber);

CREATE VIEW InjuryCountsbyModel as

(SELECT COUNT(InjuryYN) as CountOfInjuryReports, m.ModelNumber

FROM ProblemReport pr

INNER JOIN Toy t

ON t.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON m.ModelNumber = t.ModelNumber

WHERE InjuryYN = 'yes'

GROUP BY m.ModelNumber);

Graphical user interface, application, table

Description automatically generated

**Creating Functions-**

**--Function 1**

CREATE FUNCTION Duration

(@RDate DateTime, @CDate DateTime)

RETURNS INT

BEGIN

RETURN

DATEDIFF(dd, @RDate, isnull(@CDate, GETDATE()))

END

GO

**--Applying Function 1 Duration in Query 6**

SELECT CONVERT(VARCHAR,pr.ReportDate,107) as ReportDateOutput,

pr.ProblemReportID as ReportID, pr.SerialNumber as Serial#,

(p.LastName + ', ' + UPPER(LEFT(p.FirstName, 1))) as OwnerName,

isnull(CONVERT(VARCHAR, pr.CompleteDate, 107), 'Not Complete') as CompleteDate,

dbo.Duration(ReportDate, Completedate) as DaysInSystem,

m.ModelNumber as Model#, m.ModelDescription,

(pt.LastName + ', ' + UPPER(LEFT(pt.FirstName, 1))) as TesterName,

CONVERT(VARCHAR, tt.TestDate, 107) as TestDate, tt.TestDescription, tt.TestComplete

FROM ProblemReport pr

INNER JOIN Toy t

ON t.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON m.ModelNumber = t.ModelNumber

LEFT OUTER JOIN Test tt

ON tt.ReportID = pr.ProblemReportID

INNER JOIN Person p

ON p.personID = t.OwnerID

LEFT OUTER JOIN Person pt

ON pt.PersonID = tt.testerID

WHERE ModelDescription LIKE '%SUV%'

ORDER BY ReportDate;

**--Function 2**

CREATE FUNCTION ConvertedDate

(@Datein DateTime)

RETURNS VARCHAR (20)

BEGIN

RETURN CONVERT(VARCHAR, @Datein, 107)

END

GO

**--Applying Function 2 ConvertedDate in Query 6**

SELECT dbo.ConvertedDate(ReportDate) as ReportDateOutput,

pr.ProblemReportID as ReportID, pr.SerialNumber as Serial#,

(p.LastName + ', ' + UPPER(LEFT(p.FirstName, 1))) as OwnerName,

isnull(dbo.ConvertedDate(CompleteDate), 'Not Complete') as CompleteDate,

dbo.Duration(ReportDate, Completedate) as DaysInSystem,

m.ModelNumber as Model#, m.ModelDescription,

(pt.LastName + ', ' + UPPER(LEFT(pt.FirstName, 1))) as TesterName,

dbo.ConvertedDate(TestDate) as TestDate, tt.TestDescription, tt.TestComplete

FROM ProblemReport pr

INNER JOIN Toy t

ON t.SerialNumber = pr.SerialNumber

INNER JOIN Model m

ON m.ModelNumber = t.ModelNumber

LEFT OUTER JOIN Test tt

ON tt.ReportID = pr.ProblemReportID

INNER JOIN Person p

ON p.personID = t.OwnerID

LEFT OUTER JOIN Person pt

ON pt.PersonID = tt.testerID

WHERE ModelDescription LIKE '%SUV%'

ORDER BY ReportDate;

**--Function 3**

CREATE FUNCTION FullName

(@LastNamein VARCHAR (50), @FirstNamein VARCHAR (50))

RETURNS VARCHAR (100)

BEGIN

RETURN

@FirstNamein + ' ' + @LastNamein

END

GO

**-- Applying Function 3 FullName in Person entity**

SELECT PersonID, dbo.FullName(LastName, FirstName) as PersonName

FROM Person

**--Store Procedure 1**

CREATE PROCEDURE PersonDetails

(@PersonID INT)

AS

BEGIN

SELECT p.PersonID, dbo.FullName(p.LastName, p.FirstName) as PersonName, Persontype, p.PhoneNumber,

t.SerialNumber, m.ModelNumber, m.StandardPrice, t.PricePaid, (t.PricePaid - m.StandardPrice) as PriceDiff

FROM Person p

LEFT JOIN Toy t

ON t.OwnerID = p.PersonID

LEFT JOIN Model m

ON m.ModelNumber = t.ModelNumber

WHERE SerialNumber IS NOT NULL and

PersonID = @PersonID

END

EXEC PersonDetails 33;

**--Store Procedure 2**

CREATE PROCEDURE ModelPurchases

(@ModelNumber VARCHAR (6))

AS

BEGIN

SELECT M.ModelNumber, COUNT(t.SerialNumber) as CountofPurchase, PersonType

FROM Model m

INNER JOIN Toy t

ON m.ModelNumber = t.ModelNumber

INNER JOIN Person p

ON p.PersonID = t.OwnerID

GROUP BY m.ModelNumber, PersonType

HAVING m.ModelNumber = @ModelNumber

END

EXEC ModelPurchases BMWSC9

**--Store Procedure 3**

CREATE PROCEDURE RejectProduct

AS

DROP TABLE ModelRejectYN

SELECT m.ModelNumber, COUNT(ProblemReportID) ProblemsReported, InjuryYN,

CASE

WHEN COUNT(injuryYN) > 1 and InjuryYN LIKE 'Yes' then 'Reject'

ELSE 'ExamineMore'

END Results

INTO [sonikaprakashs].[dbo].[ModelRejectYN]

FROM Model m

INNER JOIN Toy t

ON t.ModelNumber = m.ModelNumber

LEFT OUTER JOIN ProblemReport pr

ON t.SerialNumber = pr.SerialNumber

GROUP BY m.ModelNumber, InjuryYN

HAVING InjuryYN LIKE 'Yes'

EXEC RejectProduct;

SELECT \* FROM ModelRejectYN

**--Store Procedure 3 (Suggested by Tyler)**

CREATE PROCEDURE CorrectInjuryYN

AS

UPDATE AllProblem

SET InjuryYN = 'Yes'

WHERE InjuryYN LIKE 'y%' or InjuryYN LIKE 'Y%'

UPDATE AllProblem

SET InjuryYN = 'No'

WHERE InjuryYN LIKE 'n%'or InjuryYN LIKE 'N%'

EXEC CorrectInjuryYN

SELECT \* FROM AllProblem