**Number of total purchases per package**

create table `total\_purchases\_per\_package`(

`packageId` int NOT NULL AUTO\_INCREMENT,

`total\_purchases` int NOT NULL DEFAULT 0,

PRIMARY KEY (`packageId`),

CONSTRAINT `total\_purchases\_per\_package\_servicepackage` FOREIGN KEY (`packageId`) REFERENCES `servicepackage` (`packageId`)

)

DELIMITER $$

create trigger insert\_new\_total\_purchases\_per\_package after insert ON servicepackage

for each row

begin

insert into total\_purchases\_per\_package(packageId, total\_purchases) values (new.packageId, 0);

end$$

DELIMITER ;

DELIMITER $$

create trigger update\_total\_purchases\_per\_package after update ON orders

for each row

begin

if not(old.orderState <=> new.orderstate) and new.orderstate <=> "Paid" then

update total\_purchases\_per\_package

set total\_purchases = total\_purchases + 1

where packageId = new.packageId;

end if;

end$$

DELIMITER ;

DELIMITER $$

create trigger insert\_paid\_total\_purchases\_per\_package after insert ON orders

for each row

begin

if new.orderstate <=> "Paid" then

update total\_purchases\_per\_package

set total\_purchases = total\_purchases + 1

where packageId = new.packageId;

end if;

end$$

DELIMITER ;

**Number of total purchases per package and validity period**

create table `total\_purchases\_per\_package\_validityperiod`(

`packageId` int NOT NULL AUTO\_INCREMENT,

`validityPeriod` int NOT NULL,

`total\_purchases` int NOT NULL DEFAULT 0,

PRIMARY KEY (`packageId`,`validityPeriod`),

CONSTRAINT `total\_purchases\_per\_package\_validityperiod\_servicepackage` FOREIGN KEY (`packageId`,`validityPeriod`) REFERENCES `servicepackage` (`packageId`,`validityPeriod`)

)

DELIMITER $$

create trigger insert\_new\_total\_purchases\_per\_package\_validityperiod after insert ON servicepackage

for each row

begin

insert into total\_purchases\_per\_package\_validityperiod(packageId,validityPeriod, total\_purchases) values (new.packageId,new.validityPeriod,0);

end$$

DELIMITER ;

DELIMITER $$

create trigger update\_total\_purchases\_per\_package\_validityperiod after update ON orders

for each row

begin

if not(old.orderState <=> new.orderstate) and new.orderstate <=> "Paid" then

update total\_purchases\_per\_package\_validityperiod

set total\_purchases = total\_purchases + 1

where packageId = new.packageId and validityPeriod = new.validityPeriod;

end if;

end$$

DELIMITER ;

DELIMITER $$

create trigger insert\_paid\_total\_purchases\_per\_package\_validityperiod after insert ON orders

for each row

begin

if new.orderstate <=> "Paid" then

update total\_purchases\_per\_package\_validityperiod

set total\_purchases = total\_purchases + 1

where packageId = new.packageId and validityPeriod = new.validityPeriod;

end if;

end$$

DELIMITER ;

**Total sales per package with and without optional product**

create table `total\_sales\_per\_package`(

`packageId` int NOT NULL AUTO\_INCREMENT,

`totalSales` int NOT NULL DEFAULT 0,

`totalSalesWithOptionalProduct` int NOT NULL DEFAULT 0,

PRIMARY KEY (`packageId`),

CONSTRAINT `total\_sales\_per\_package\_servicepackage` FOREIGN KEY (`packageId`) REFERENCES `servicepackage` (`packageId`)

)

**Average number of sales per optional product with each service package**

create table `average\_sales\_optionalproduct\_per\_servicepackage`(

`packageId` int NOT NULL AUTO\_INCREMENT,

`averageOptionalProducts` int NOT NULL DEFAULT 0,

PRIMARY KEY (`packageId`),

CONSTRAINT `average\_sales\_optionalproduct\_per\_servicepackage\_servicepackage` FOREIGN KEY (`packageId`) REFERENCES `servicepackage` (`packageId`)

)

**List of insolvent users**

create table `insolvent\_users`(

`username` varchar(64) NOT NULL,

PRIMARY KEY (`username`),

CONSTRAINT `insolvent\_users\_users` FOREIGN KEY (`username`) REFERENCES `users` (`username`)

)

**List of suspended orders**

create table `suspended\_orders`(

`orderId` int NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`orderId`),

CONSTRAINT `suspended\_orders\_orders` FOREIGN KEY (`orderId`) REFERENCES `orders` (`orderId`)

)

**List of Alerts**

CREATE TABLE `alert` (

`alertId` int NOT NULL AUTO\_INCREMENT,

`username` varchar(64) NOT NULL,

`amount` float NOT NULL,

`lastRejectionDateTime` timestamp NOT NULL,

PRIMARY KEY (`alertId`),

CONSTRAINT `username` FOREIGN KEY (`username`) REFERENCES `users` (`username`)

)**Best seller optional product**

create table `bestseller\_optionalproduct`(

`productID` int NOT NULL AUTO\_INCREMENT,

`sales` int NOT NULL,

PRIMARY KEY (`productID`),

CONSTRAINT `bestseller\_optionalproduct\_optionalproduct` FOREIGN KEY (`productID`) REFERENCES `optionalproduct` (`productID`)

)