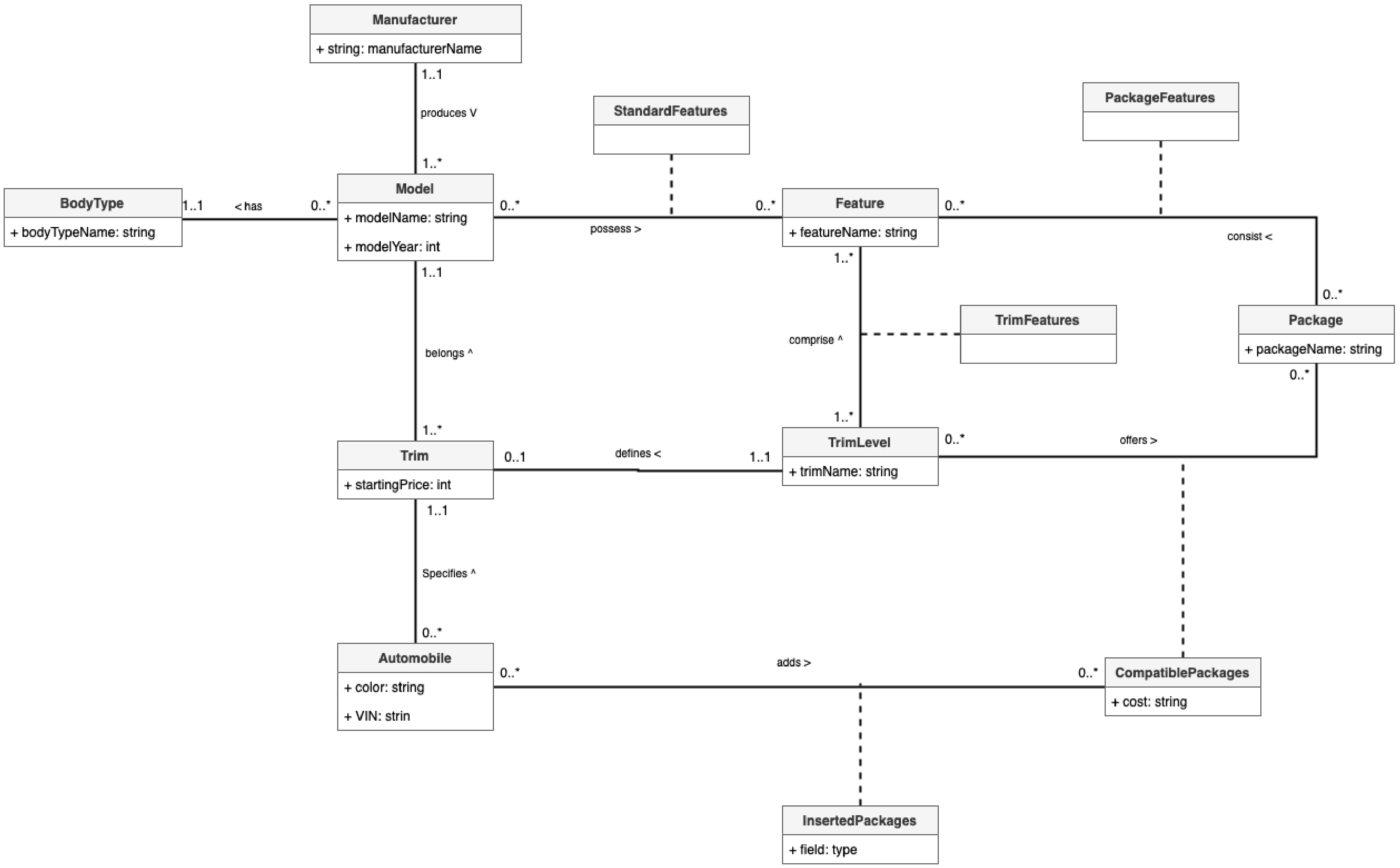
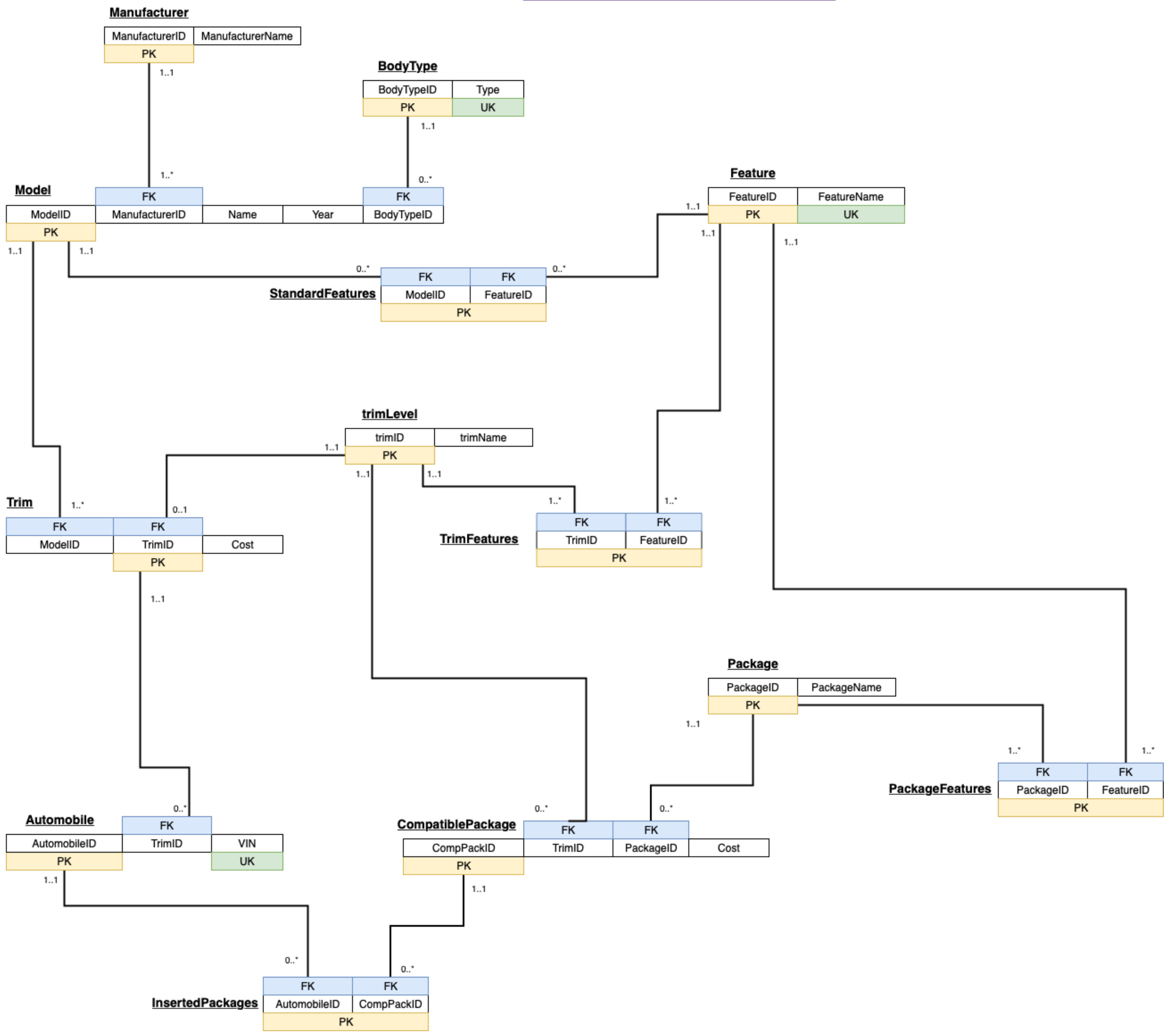
# UML Diagram



Manufacturer
+ string: manufacturerName
1..1
produces V
1..*
Model
+ modelName: string
+ modelYear: int
BodyType
+ bodyTypeName: string
1..1
< has
0..*
StandardFeatures
0..*
possess >
0..*
Feature
+ featureName: string
0..*
PackageFeatures
consist <
0..*
Package
+ packageName: string
0..*
1..*
comprise ^
TrimFeatures
1..*
1..1
belongs ^
1..*
Trim
+ startingPrice: int
0..1
defines <
1..1
TrimLevel
+ trimName: string
0..*
offers >
1..1
Specifies ^
0..*
Automobile
+ color: string
+ VIN: strin
0..*
adds >
0..*
CompatiblePackages
+ cost: string
InsertedPackages
+ field: type

# Relation Scheme



# DDL Commands

```
create table manufacturers
(
    "manufacturerID"   integer not null
        constraint manufacturers_pk
            primary key,
    "manufacturerName" varchar not null
);

alter table manufacturers
    owner to postgres;

create unique index manufacturers_manufacturerid_uindex
    on manufacturers ("manufacturerID");

create table "bodyType"
(
    "bodyTypeID"   serial
        constraint bodytype_pk
            primary key,
    "bodyTypeName" varchar not null
);

alter table "bodyType"
    owner to postgres;

create table models
(
    "modelID"        integer not null
        constraint models_pk
            primary key,
    "manufacturerID" integer
        constraint models_manufacturers_manufacturerid_fk
            references manufacturers,
    "modelName"      varchar not null,
    "modelYear"      integer,
    "bodyType"       integer not null
        constraint models_bodytype_bodytypeid_fk
            references "bodyType"
);

alter table models
    owner to postgres;

create unique index models_modelid_uindex
    on models ("modelID");

create unique index bodytype_bodytypeid_uindex
    on "bodyType" ("bodyTypeID");

create table features
(
    "featureID"   serial
        constraint features_pk
            primary key,
    "featureName" varchar not null
```

```
);

alter table features
    owner to postgres;

create unique index features_featureid_uindex
    on features ("featureID");

create table packages
(
    "packageID"   serial
        constraint packages_pk
            primary key,
    "packageName" varchar not null
);

alter table packages
    owner to postgres;

create unique index packages_packageid_uindex
    on packages ("packageID");

create table "packageFeatures"
(
    "packageID" integer not null
        constraint packagefeatures_packages_packageid_fk
            references packages,
    "featureID" integer not null
        constraint packagefeatures_features_featureid_fk
            references features,
    constraint packagefeatures_pk
        primary key ("packageID", "featureID")
);

alter table "packageFeatures"
    owner to postgres;

create table "standardFeatures"
(
    "modelID"   integer not null
        constraint standardfeatures_models_modelid_fk
            references models,
    "featureID" integer not null
        constraint standardfeatures_features_featureid_fk
            references features,
    constraint standardfeatures_pk
        primary key ("modelID", "featureID")
);

alter table "standardFeatures"
    owner to postgres;

create table "trimLevel"
(
    "trimID"   integer not null
        constraint trimlevel_pk
            primary key,
    "trimName" varchar not null
);

alter table "trimLevel"
    owner to postgres;

create table trims
(
    "trimID"        integer not null
        constraint trims_pk
            primary key
        constraint trims_trimlevel_trimid_fk
            references "trimLevel",
    "modelID"       integer not null
        constraint trims_models_modelid_fk
            references models,
    "startingPrice" integer not null
);

alter table trims
    owner to postgres;

create unique index trims_trimid_uindex
    on trims ("trimID");

create table "trimFeatures"
(
    "trimID"    integer not null
        constraint trimfeatures_trimlevel_trimid_fk
            references "trimLevel",
    "featureID" integer not null
        constraint trimfeatures_features_featureid_fk
            references features,
    constraint trimfeatures_pk
        primary key ("trimID", "featureID")
);

alter table "trimFeatures"
    owner to postgres;

create table automobile
(
    "automobileID" serial
        constraint automobile_pk
            primary key,
    vin            varchar not null,
    "trimID"       integer not null
        constraint automobile_trims_trimid_fk
            references trims,
    color          varchar not null
);

alter table automobile
    owner to postgres;

create unique index automobile_automobileid_uindex
    on automobile ("automobileID");
```

```
create unique index automobile_vin_uindex
    on automobile (vin);

create table "compatiblePackages"
(
    "compPackID" serial
        constraint compatiblepackages_pk
            primary key,
    "trimID"     integer not null
        constraint compatiblepackages_trimlevel_trimid_fk
            references "trimLevel",
    "packageID"  integer not null
        constraint compatiblepackages_packages_packageid_fk
            references packages,
    "Cost"       integer not null
);

alter table "compatiblePackages"
    owner to postgres;

create unique index compatiblepackages_comppackid_uindex
    on "compatiblePackages" ("compPackID");

create table "insertedPackages"
(
    "automobileID" integer not null
        constraint insertedpackages_automobile_automobileid_fk
            references automobile,
    "compPackID"   integer not null
        constraint insertedpackages_compatiblepackages_comppackid_fk
            references "compatiblePackages",
    constraint insertedpackages_pk
        primary key ("automobileID", "compPackID")
);

alter table "insertedPackages"
    owner to postgres;

create unique index trimlevel_trimid_uindex
    on "trimLevel" ("trimID");

create function check_package_compatibility() returns trigger
    language plpgsql
as
$$
declare pack record;
    declare cpack record;
    BEGIN

    select "automobileID", "trimID"
    into pack
    from automobile
    where new."automobileID" = automobile."automobileID";

    select "compPackID", "trimID"
    into cpack
    from "compatiblePackages"
    where new."compPackID" = "compatiblePackages"."compPackID";


    if  pack."trimID" <> cpack."trimID" then
        raise exception 'The package is not compatible with this automobile';
    end if;

    return new;
end;
$$;

alter function check_package_compatibility() owner to postgres;

create trigger package_insert_check
    before insert
    on "insertedPackages"
execute procedure check_package_compatibility();
```

# Queries

```
select m2."manufacturerName", m."modelYear", m."modelName", tl."trimName", vin
from automobile
inner join trims t on t."trimID" = automobile."trimID"
inner join "trimLevel" tL on tL."trimID" = t."trimID"
inner join models m on m."modelID" = t."modelID"
inner join manufacturers m2 on m2."manufacturerID" = m."manufacturerID"


select "modelYear", min("startingPrice")
from trims
inner join models m on m."modelID" = trims."modelID"
inner join manufacturers m2 on m2."manufacturerID" = m."manufacturerID"
where "manufacturerName" = 'Toyota'
group by "modelYear"


select count(*)
from automobile a, models m,  trims t, "trimFeatures" tf, features f
where a."trimID" = t."trimID" and t."modelID" = m."modelID" and t."trimID" =
tf."trimID" and tf."featureID" = f."featureID" and f."featureName" not like
'%Leather seats%'

select max(total)
from (select vin, max("startingPrice" + "Cost") as total
    from automobile
    inner join "insertedPackages" iP on automobile."automobileID" =
iP."automobileID"
    inner join "compatiblePackages" cP on cP."compPackID" = iP."compPackID"
    inner join trims t on t."trimID" = automobile."trimID"
    group by vin) as highestPrice
```