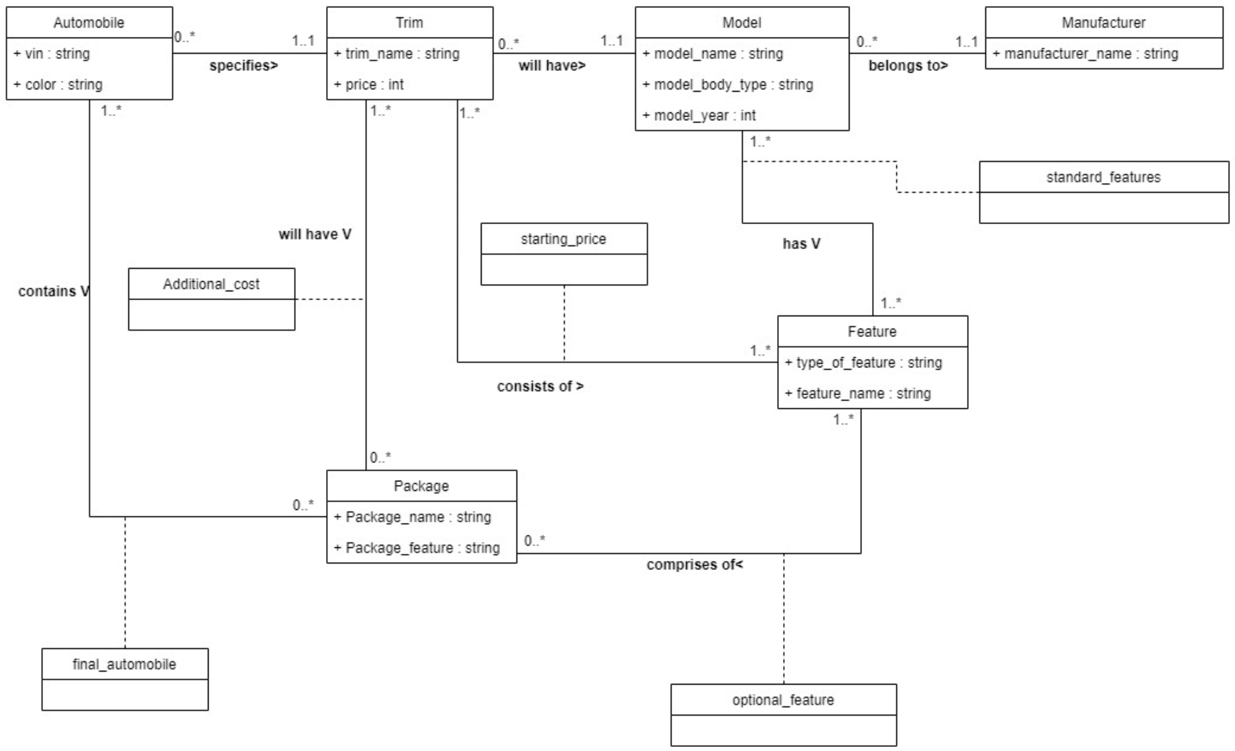
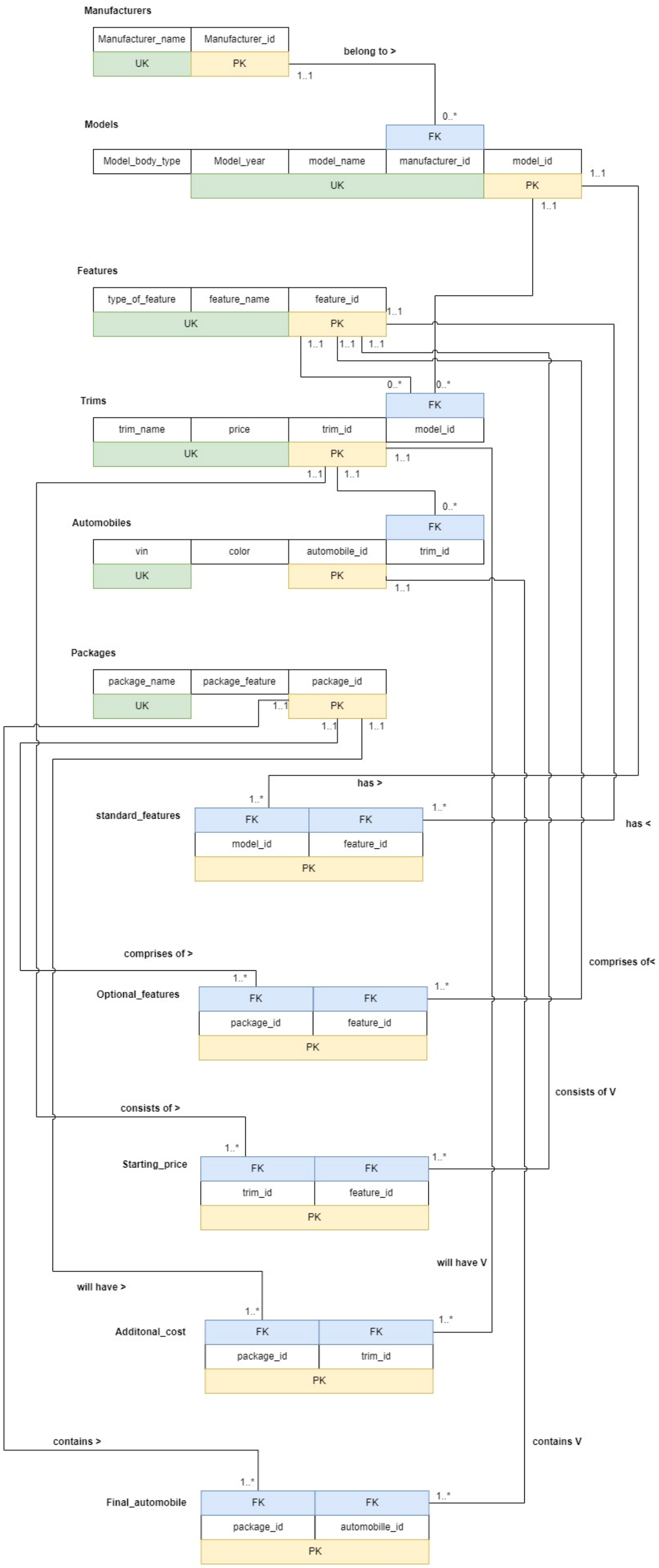
Avlokita Sharma

Project 2: Drive my car

Due date:04/9/2022 11:59pm

UML:


Automobile
+ vin : string
+ color : string
0..*
specifies>
1..1
Trim
+ trim_name : string
+ price : int
0..*
will have>
1..1
Model
+ model_name : string
+ model_body_type : string
+ model_year : int
0..*
belongs to>
1..1
Manufacturer
+ manufacturer_name : string
1..*
1..*
1..*
1..*
standard_features
will have V
starting_price
has V
contains V
Additional_cost
1..*
Feature
1..*
+ type_of_feature : string
+ feature_name : string
consists of >
1..*
0..*
Package
0..*
+ Package_name : string
+ Package_feature : string
0..*
comprises of<
final_automobile
optional_feature



Manufacturers

Manufacturer_name	Manufacturer_id
UK	PK

Models

			FK	
Model_body_type	Model_year	model_name	manufacturer_id	model_id
	UK			PK

Features

type_of_feature	feature_name	feature_id
UK		PK

Trims

			FK
trim_name	price	trim_id	model_id
UK		PK	

Automobiles

			FK
vin	color	automobile_id	trim_id
UK		PK	

Packages

package_name	package_feature	package_id
UK		PK

standard_features

FK	FK
model_id	feature_id
PK	

Optional_features

FK	FK
package_id	feature_id
PK	

Starting_price

FK	FK
trim_id	feature_id
PK	

Additonal_cost

FK	FK
package_id	trim_id
PK	

Final_automobile

FK	FK
package_id	automobille_id
PK	

DDL commands:

```
create sequence starting_point_trim_id_seq
    as integer;

alter sequence starting_point_trim_id_seq owner to postgres;

create sequence starting_point_feature_id_seq
    as integer;

alter sequence starting_point_feature_id_seq owner to postgres;

create table if not exists "Manufacturers"
(
    "Manufacturer_name" varchar(50) not null
        constraint manufacturers_uk
            unique,
    manufacturer_id     serial
        constraint manufacturers_pk
            primary key
);

alter table "Manufacturers"
    owner to postgres;

create unique index if not exists manufacturers_manufacturer_id_uindex
    on "Manufacturers" (manufacturer_id);

create unique index if not exists manufacturers_manufacturer_name_uindex
    on "Manufacturers" ("Manufacturer_name");
```

```
create table if not exists "Models"
(
    model_body_type varchar(50) not null,
    model_name      varchar(50) not null,
    model_year      varchar(50) not null,
    manufacturer_id serial
        constraint models_manufacturers_manufacturer_id_fk
            references "Manufacturers",
    model_id        serial
        constraint models_pk
            primary key,
    constraint models_uk
        unique (model_year, model_name, manufacturer_id)
);

alter table "Models"
    owner to postgres;

create unique index if not exists models_model_id_uindex
    on "Models" (model_id);

create table if not exists "Features"
(
    type_of_feature varchar(100) not null,
    "Feature_name"  varchar(50)  not null,
    feature_id      serial
        constraint features_pk
            primary key,
    constraint features_uk
        unique (type_of_feature, "Feature_name")
);

alter table "Features"
    owner to postgres;

create unique index if not exists features_feature_id_uindex
    on "Features" (feature_id);

create table if not exists "Trims"
(
    trim_name varchar(50) not null,
    trim_id   serial
        constraint trims_pk
            primary key,
    model_id  serial
        constraint trims_models_model_id_fk
            references "Models",
    price     bigint      not null,
    constraint trims_uk
        unique (trim_name, model_id)
);

alter table "Trims"
    owner to postgres;

create unique index if not exists trims_trim_id_uindex
    on "Trims" (trim_id);
```

```
create table if not exists "Automobiles"
(
    vin           varchar(50) not null
        constraint automobiles_uk
            unique,
    color         varchar(50) not null,
    automobile_id serial
        constraint automobiles_pk
            primary key,
    trim_id       serial
        constraint automobiles_trims_trim_id_fk
            references "Trims"
);

alter table "Automobiles"
    owner to postgres;

create unique index if not exists automobiles_automobile_id_uindex
    on "Automobiles" (automobile_id);

create unique index if not exists automobiles_vin_uindex
    on "Automobiles" (vin);

create table if not exists "Packages"
(
    package_feature varchar(100),
    package_name    varchar(100) not null
        constraint packages_uk
```

```
            unique,
    package_id      serial
        constraint packages_pk
            primary key
);

alter table "Packages"
    owner to postgres;

create unique index if not exists packages_package_id_uindex
    on "Packages" (package_id);

create table if not exists starting_price
(
    trim_id    integer default nextval('"project 2".starting_point_trim_id_seq'::regclass)    not null
        constraint starting_point_trims_trim_id_fk
            references "Trims",
    feature_id integer default nextval('"project 2".starting_point_feature_id_seq'::regclass) not null
        constraint starting_point_features_feature_id_fk
            references "Features",
    constraint starting_point_pk
        primary key (trim_id, feature_id)
);

alter table starting_price
    owner to postgres;

alter sequence starting_point_trim_id_seq owned by starting_price.trim_id;
alter sequence starting_point_feature_id_seq owned by starting_price.feature_id;

create table if not exists standard_feature
(
    feature_id serial
        constraint standard_feature_features_feature_id_fk
            references "Features",
    model_id   serial
        constraint standard_feature_models_model_id_fk
            references "Models",
    constraint standard_feature_pk
        primary key (feature_id, model_id)
);

alter table standard_feature
    owner to postgres;

create table if not exists optional_features
(
    feature_id serial
        constraint optional_features_features_feature_id_fk
            references "Features",
    package_id serial
        constraint optional_features_packages_package_id_fk
            references "Packages",
    constraint optional_features_pk
        primary key (feature_id, package_id)
);
```

```
alter table optional_features
    owner to postgres;

create table if not exists final_automobile
(
    automobile_id serial
        constraint final_automobile_automobiles_automobile_id_fk
            references "Automobiles",
    package_id    serial
        constraint final_automobile_packages_package_id_fk
            references "Packages",
    constraint final_automobile_pk
        primary key (automobile_id, package_id)
);

alter table final_automobile
    owner to postgres;

create table if not exists additional_cost
(
    package_id serial
        constraint additional_cost_packages_package_id_fk
            references "Packages",
    trim_id    serial
        constraint additional_cost_trims_trim_id_fk
            references "Trims",
    constraint additional_cost_pk
        primary key (package_id, trim_id)
);
```

```
alter table additional_cost
    owner to postgres;
```

SELECT STATEMENT:

1.

```
select "Manufacturer_name",model_year, model_name, trim_name, vin
from "Manufacturers"
inner join "Models" M on "Manufacturers".manufacturer_id = M.manufacturer_id
inner join "Trims" T on M.model_id = T.model_id
inner join "Automobiles" A on T.trim_id = A.trim_id;
```

2.

```
select M.manufacturer_id, model_year, min(price)
from "Models" inner join "Manufacturers" M on M.manufacturer_id =
"Models".manufacturer_id
inner join "Trims" T on "Models".model_id = T.model_id
where M.manufacturer_id = (
    select manufacturer_id
    from "Manufacturers"
    where "Manufacturer_name" = 'Toyota'
      )
group by model_year, M.manufacturer_id;
```

3.

```
select count("Automobiles".)

from "project 2"."Automobiles" inner join "project 2"."Trims" T on
"Automobiles".trim_id = T.trim_id

inner join "project 2"."Models" M on T.model_id = M.model_id

inner join "project 2".standard_feature on M.model_id =
standard_feature.model_id

where feature_id = (

select feature_id

from "project 2"."Features"

    where "Feature_name" = "%leather seats%"

    )
```

4.

```
select vin
from public."Automobiles"
where
    (
select max(total)
from (
        select sum(price * package_price) as total
        from public."Automobiles"
                inner join public."Trims" T on "Automobiles".trim_id = T.trim_id
                inner join public."Packages" on price
                inner join public."Models" M on T.model_id = M.model_id
        inner join public.additional_cost ac on "Packages".package_id = ac.package_id)
as joinss)

    )
```