```
schema {
    query: Query
    mutation: Mutation
    subscription: Subscription
  }

  type Address {
    address: String
    city: String
    state: String
  }

  type Capsule {
    dragon: Dragon @deprecated(reason: "This is not available in the REST API
after MongoDB has been deprecated")
    id: ID
    landings: Int
    missions: [CapsuleMission]
    original_launch: Date
    reuse_count: Int
    status: String
    type: String
  }

  type CapsuleMission {
    flight: Int
    name: String
  }

  input CapsulesFind {
    id: ID
    landings: Int
    mission: String
    original_launch: Date
    reuse_count: Int
    status: String
    type: String
  }

  """conflict action"""
  enum conflict_action {
    """ignore the insert on this row"""
    ignore
    """update the row with the given values"""
    update
  }

  type Core {
    asds_attempts: Int
    asds_landings: Int
    block: Int
    id: ID
    missions: [CapsuleMission]
    original_launch: Date
```

```
  reuse_count: Int
  rtls_attempts: Int
  rtls_landings: Int
  status: String
  water_landing: Boolean
}

type CoreMission {
  flight: Int
  name: String
}

input CoresFind {
  asds_attempts: Int
  asds_landings: Int
  block: Int
  id: String
  missions: String
  original_launch: Date
  reuse_count: Int
  rtls_attempts: Int
  rtls_landings: Int
  status: String
  water_landing: Boolean
}

scalar Date

type Distance {
  feet: Float
  meters: Float
}

type Dragon {
  active: Boolean
  crew_capacity: Int
  description: String
  diameter: Distance
  dry_mass_kg: Int
  dry_mass_lb: Int
  first_flight: String
  heat_shield: DragonHeatShield
  height_w_trunk: Distance
  id: ID
  launch_payload_mass: Mass
  launch_payload_vol: Volume
  name: String
  orbit_duration_yr: Int
  pressurized_capsule: DragonPressurizedCapsule
  return_payload_mass: Mass
  return_payload_vol: Volume
  sidewall_angle_deg: Float
  thrusters: [DragonThrust]
  trunk: DragonTrunk
```

```
  type: String
  wikipedia: String
}

type DragonHeatShield {
  dev_partner: String
  material: String
  size_meters: Float
  temp_degrees: Int
}

type DragonPressurizedCapsule {
  payload_volume: Volume
}

type DragonThrust {
  amount: Int
  fuel_1: String
  fuel_2: String
  pods: Int
  thrust: Force
  type: String
}

type DragonTrunk {
  cargo: DragonTrunkCargo
  trunk_volume: Volume
}

type DragonTrunkCargo {
  solar_array: Int
  unpressurized_cargo: Boolean
}

type Force {
  kN: Float
  lbf: Float
}

type HistoriesResult {
  data: [History]
  result: Result
}

type History {
  details: String
  event_date_unix: Date
  event_date_utc: Date
  flight: Launch
  id: ID
  links: Link
  title: String
}
```

```
input HistoryFind {
  end: Date
  flight_number: Int
  id: ID
  start: Date
}

type Info {
  ceo: String
  coo: String
  cto: String
  cto_propulsion: String
  employees: Int
  founded: Int
  founder: String
  headquarters: Address
  launch_sites: Int
  links: InfoLinks
  name: String
  summary: String
  test_sites: Int
  valuation: Float
  vehicles: Int
}

type InfoLinks {
  elon_twitter: String
  flickr: String
  twitter: String
  website: String
}

type Landpad {
  attempted_landings: String
  details: String
  full_name: String
  id: ID
  landing_type: String
  location: Location
  status: String
  successful_landings: String
  wikipedia: String
}

type Launch {
  details: String
  id: ID
  is_tentative: Boolean
  launch_date_local: Date
  launch_date_unix: Date
  launch_date_utc: Date
  launch_site: LaunchSite
  launch_success: Boolean
  launch_year: String
```

```
  links: LaunchLinks
  mission_id: [String]
  mission_name: String
  rocket: LaunchRocket
  ships: [Ship]
  static_fire_date_unix: Date
  static_fire_date_utc: Date
  telemetry: LaunchTelemetry
  tentative_max_precision: String
  upcoming: Boolean
}

type LaunchesPastResult {
  data: [Launch]
  result: Result
}

input LaunchFind {
  apoapsis_km: Float
  block: Int
  cap_serial: String
  capsule_reuse: String
  core_flight: Int
  core_reuse: String
  core_serial: String
  customer: String
  eccentricity: Float
  end: Date
  epoch: Date
  fairings_recovered: String
  fairings_recovery_attempt: String
  fairings_reuse: String
  fairings_reused: String
  fairings_ship: String
  gridfins: String
  id: ID
  inclination_deg: Float
  land_success: String
  landing_intent: String
  landing_type: String
  landing_vehicle: String
  launch_date_local: Date
  launch_date_utc: Date
  launch_success: String
  launch_year: String
  legs: String
  lifespan_years: Float
  longitude: Float
  manufacturer: String
  mean_motion: Float
  mission_id: String
  mission_name: String
  nationality: String
  norad_id: Int
```

```
  orbit: String
  payload_id: String
  payload_type: String
  periapsis_km: Float
  period_min: Float
  raan: Float
  reference_system: String
  regime: String
  reused: String
  rocket_id: String
  rocket_name: String
  rocket_type: String
  second_stage_block: String
  semi_major_axis_km: Float
  ship: String
  side_core1_reuse: String
  side_core2_reuse: String
  site_id: String
  site_name_long: String
  site_name: String
  start: Date
  tbd: String
  tentative_max_precision: String
  tentative: String
}

type LaunchLinks {
  article_link: String
  flickr_images: [String]
  mission_patch: String
  mission_patch_small: String
  presskit: String
  reddit_campaign: String
  reddit_launch: String
  reddit_media: String
  reddit_recovery: String
  video_link: String
  wikipedia: String
}

type Launchpad {
  attempted_launches: Int
  details: String
  id: ID
  location: Location
  name: String
  status: String
  successful_launches: Int
  vehicles_launched: [Rocket]
  wikipedia: String
}

type LaunchRocket {
  fairings: LaunchRocketFairings
```

```
  first_stage: LaunchRocketFirstStage
  rocket: Rocket
  rocket_name: String
  rocket_type: String
  second_stage: LaunchRocketSecondStage
}

type LaunchRocketFairings {
  recovered: Boolean
  recovery_attempt: Boolean
  reused: Boolean
  ship: String
}

type LaunchRocketFirstStage {
  cores: [LaunchRocketFirstStageCore]
}

type LaunchRocketFirstStageCore {
  block: Int
  core: Core
  flight: Int
  gridfins: Boolean
  land_success: Boolean
  landing_intent: Boolean
  landing_type: String
  landing_vehicle: String
  legs: Boolean
  reused: Boolean
}

type LaunchRocketSecondStage {
  block: Int
  payloads: [Payload]
}

type LaunchSite {
  site_id: String
  site_name: String
  site_name_long: String
}

type LaunchTelemetry {
  flight_club: String
}

type Link {
  article: String
  reddit: String
  wikipedia: String
}

type Location {
  latitude: Float
```

```
  longitude: Float
  name: String
  region: String
}

type Mass {
  kg: Int
  lb: Int
}

type Mission {
  description: String
  id: ID
  manufacturers: [String]
  name: String
  payloads: [Payload]
  twitter: String
  website: String
  wikipedia: String
}

type MissionResult {
  data: [Mission]
  result: Result
}

input MissionsFind {
  id: ID
  manufacturer: String
  name: String
  payload_id: String
}

type Mutation {
  """
  delete data from the table: "users"
  """
  delete_users(
    """filter the rows which have to be deleted"""
    where: users_bool_exp!
  ): users_mutation_response
  """
  insert data into the table: "users"
  """
  insert_users(
    """the rows to be inserted"""
    objects: [users_insert_input!]!
    """on conflict condition"""
    on_conflict: users_on_conflict
  ): users_mutation_response
  """
  update data of the table: "users"
  """
  update_users(
```

```
    """sets the columns of the filtered rows to the given values"""
    _set: users_set_input
    """filter the rows which have to be updated"""
    where: users_bool_exp!
  ): users_mutation_response
}

scalar ObjectID

"""column ordering options"""
enum order_by {
  """in the ascending order, nulls last"""
  asc
  """in the ascending order, nulls first"""
  asc_nulls_first
  """in the ascending order, nulls last"""
  asc_nulls_last
  """in the descending order, nulls first"""
  desc
  """in the descending order, nulls first"""
  desc_nulls_first
  """in the descending order, nulls last"""
  desc_nulls_last
}

type Payload {
  customers: [String]
  id: ID
  manufacturer: String
  nationality: String
  norad_id: [Int]
  orbit: String
  orbit_params: PayloadOrbitParams
  payload_mass_kg: Float
  payload_mass_lbs: Float
  payload_type: String
  reused: Boolean
}

type PayloadOrbitParams {
  apoapsis_km: Float
  arg_of_pericenter: Float
  eccentricity: Float
  epoch: Date
  inclination_deg: Float
  lifespan_years: Float
  longitude: Float
  mean_anomaly: Float
  mean_motion: Float
  periapsis_km: Float
  period_min: Float
  raan: Float
  reference_system: String
  regime: String
```

```
    semi_major_axis_km: Float
  }

  input PayloadsFind {
    apoapsis_km: Float
    customer: String
    eccentricity: Float
    epoch: Date
    inclination_deg: Float
    lifespan_years: Float
    longitude: Float
    manufacturer: String
    mean_motion: Float
    nationality: String
    norad_id: Int
    orbit: String
    payload_id: ID
    payload_type: String
    periapsis_km: Float
    period_min: Float
    raan: Float
    reference_system: String
    regime: String
    reused: Boolean
    semi_major_axis_km: Float
  }

  type Query {
    capsule(id: ID!): Capsule
    capsules(find: CapsulesFind, limit: Int, offset: Int, order: String, sort:
String): [Capsule]
    capsulesPast(find: CapsulesFind, limit: Int, offset: Int, order: String,
sort: String): [Capsule]
    capsulesUpcoming(find: CapsulesFind, limit: Int, offset: Int, order: String,
sort: String): [Capsule]
    company: Info
    core(id: ID!): Core
    cores(find: CoresFind, limit: Int, offset: Int, order: String, sort:
String): [Core]
    coresPast(find: CoresFind, limit: Int, offset: Int, order: String, sort:
String): [Core]
    coresUpcoming(find: CoresFind, limit: Int, offset: Int, order: String, sort:
String): [Core]
    dragon(id: ID!): Dragon
    dragons(limit: Int, offset: Int): [Dragon]
    histories(find: HistoryFind, limit: Int, offset: Int, order: String, sort:
String): [History]
    historiesResult(find: HistoryFind, limit: Int, offset: Int, order: String,
sort: String): HistoriesResult
    history(id: ID!): History
    landpad(id: ID!): Landpad
    landpads(limit: Int, offset: Int): [Landpad]
    launch(id: ID!): Launch
    launchLatest(offset: Int): Launch
```

```
    launchNext(offset: Int): Launch
    launches(find: LaunchFind, limit: Int, offset: Int, order: String, sort:
String): [Launch]
    launchesPast(find: LaunchFind, limit: Int, offset: Int, order: String, sort:
String): [Launch]
    launchesPastResult(find: LaunchFind, limit: Int, offset: Int, order: String,
sort: String): LaunchesPastResult
    launchesUpcoming(find: LaunchFind, limit: Int, offset: Int, order: String,
sort: String): [Launch]
    launchpad(id: ID!): Launchpad
    launchpads(limit: Int, offset: Int): [Launchpad]
    mission(id: ID!): Mission @deprecated(reason: "Mission is not available on
REST API after MongoDB deprecation")
    missions(find: MissionsFind, limit: Int, offset: Int): [Mission]
@deprecated(reason: "Mission is not available on REST API after MongoDB
deprecation")
    missionsResult(find: MissionsFind, limit: Int, offset: Int): MissionResult
@deprecated(reason: "Mission is not available on REST API after MongoDB
deprecation")
    payload(id: ID!): Payload
    payloads(find: PayloadsFind, limit: Int, offset: Int, order: String, sort:
String): [Payload]
    roadster: Roadster
    rocket(id: ID!): Rocket
    rockets(limit: Int, offset: Int): [Rocket]
    rocketsResult(limit: Int, offset: Int): RocketsResult
    ship(id: ID!): Ship
    ships(find: ShipsFind, limit: Int, offset: Int, order: String, sort:
String): [Ship]
    shipsResult(find: ShipsFind, limit: Int, offset: Int, order: String, sort:
String): ShipsResult
    """
    fetch data from the table: "users"
    """
    users(
      """distinct select on columns"""
      distinct_on: [users_select_column!]
      """limit the nuber of rows returned"""
      limit: Int
      """skip the first n rows. Use only with order_by"""
      offset: Int
      """sort the rows by one or more columns"""
      order_by: [users_order_by!]
      """filter the rows returned"""
      where: users_bool_exp
    ): [users!]!
    """
    fetch aggregated fields from the table: "users"
    """
    users_aggregate(
      """distinct select on columns"""
      distinct_on: [users_select_column!]
      """limit the nuber of rows returned"""
      limit: Int
```

```
    """skip the first n rows. Use only with order_by"""
    offset: Int
    """sort the rows by one or more columns"""
    order_by: [users_order_by!]
    """filter the rows returned"""
    where: users_bool_exp
  ): users_aggregate!
  """fetch data from the table: "users" using primary key columns"""
  users_by_pk(id: uuid!): users
}

type Result {
  totalCount: Int
}

type Roadster {
  apoapsis_au: Float
  details: String
  earth_distance_km: Float
  earth_distance_mi: Float
  eccentricity: Float
  epoch_jd: Float
  inclination: Float
  launch_date_unix: Date
  launch_date_utc: Date
  launch_mass_kg: Int
  launch_mass_lbs: Int
  longitude: Float
  mars_distance_km: Float
  mars_distance_mi: Float
  name: String
  norad_id: Int
  orbit_type: Float
  periapsis_arg: Float
  periapsis_au: Float
  period_days: Float
  semi_major_axis_au: Float
  speed_kph: Float
  speed_mph: Float
  wikipedia: String
}

type Rocket {
  active: Boolean
  boosters: Int
  company: String
  cost_per_launch: Int
  country: String
  description: String
  diameter: Distance
  engines: RocketEngines
  first_flight: Date
  first_stage: RocketFirstStage
  height: Distance
```

```
  id: ID
  landing_legs: RocketLandingLegs
  mass: Mass
  name: String
  payload_weights: [RocketPayloadWeight]
  second_stage: RocketSecondStage
  stages: Int
  success_rate_pct: Int
  type: String
  wikipedia: String
}

type RocketEngines {
  engine_loss_max: String
  layout: String
  number: Int
  propellant_1: String
  propellant_2: String
  thrust_sea_level: Force
  thrust_to_weight: Float
  thrust_vacuum: Force
  type: String
  version: String
}

type RocketFirstStage {
  burn_time_sec: Int
  engines: Int
  fuel_amount_tons: Float
  reusable: Boolean
  thrust_sea_level: Force
  thrust_vacuum: Force
}

type RocketLandingLegs {
  material: String
  number: Int
}

type RocketPayloadWeight {
  id: String
  kg: Int
  lb: Int
  name: String
}

type RocketSecondStage {
  burn_time_sec: Int
  engines: Int
  fuel_amount_tons: Float
  payloads: RocketSecondStagePayloads
  thrust: Force
}
```

```
type RocketSecondStagePayloadCompositeFairing {
  diameter: Distance
  height: Distance
}

type RocketSecondStagePayloads {
  composite_fairing: RocketSecondStagePayloadCompositeFairing
  option_1: String
}

type RocketsResult {
  data: [Rocket]
  result: Result
}

type Ship {
  abs: Int
  active: Boolean
  attempted_landings: Int
  class: Int
  course_deg: Int
  home_port: String
  id: ID
  image: String
  imo: Int
  missions: [ShipMission]
  mmsi: Int
  model: String
  name: String
  position: ShipLocation
  roles: [String]
  speed_kn: Float
  status: String
  successful_landings: Int
  type: String
  url: String
  weight_kg: Int
  weight_lbs: Int
  year_built: Int
}

type ShipLocation {
  latitude: Float
  longitude: Float
}

type ShipMission {
  flight: String
  name: String
}

input ShipsFind {
  id: ID
  name: String
```

```
  model: String
  type: String
  role: String
  active: Boolean
  imo: Int
  mmsi: Int
  abs: Int
  class: Int
  weight_lbs: Int
  weight_kg: Int
  year_built: Int
  home_port: String
  status: String
  speed_kn: Int
  course_deg: Int
  latitude: Float
  longitude: Float
  successful_landings: Int
  attempted_landings: Int
  mission: String
}

type ShipsResult {
  data: [Ship]
  result: Result
}
```