Prisma Aggregation Cheat Sheet

Aggregations allow you to compute summary statistics (count, average, sum, min, max) directly on the database.

Supported Aggregations

- _count: Count records - _sum: Sum numeric fields - _avg: Average numeric fields - _min: Minimum value - _max: Maximum value - groupBy: Group data and run aggregations for each group

Example Model

```
model Post {
  id        String   @id @default(uuid())
  title     String
  views     Int      @default(0)
  createdAt DateTime @default(now())
  authorId  String
  author    User     @relation(fields: [authorId], references: [id])
}
```

Basic Aggregate Example

```
const stats = await prisma.post.aggregate({
  _count: true,
  _avg: { views: true },
  _sum: { views: true },
  _min: { views: true, createdAt: true },
  _max: { views: true, createdAt: true },
  where: { views: { gt: 0 } }
});
```

Result: { _count: 10, _avg: { views: 45 }, _sum: { views: 450 }, _min: { views: 1, createdAt: '2025-01-01T...' }, _max: { views: 120, createdAt: '2025-08-20T...' } }

GroupBy Example

```
const grouped = await prisma.post.groupBy({
  by: ['authorId'],
  _count: { _all: true },
  _sum: { views: true },
  _avg: { views: true },
  having: {
    views: { _avg: { gt: 10 } }
  },
  orderBy: { _sum: { views: 'desc' } }
});
```

Result: [{ authorId: 'user-1', _count: { _all: 5 }, _sum: { views: 300 }, _avg: { views: 60 } }]

Key Tips

- Use `where` to filter before aggregating. - Use indexes on frequently grouped fields for performance. - Combine `orderBy` and `take` for top-N queries.

SQL Equivalents

- _count → COUNT(*) - _sum → SUM(column) - _avg → AVG(column) - _min → MIN(column) - _max → MAX(column) - groupBy → GROUP BY column