# Django ORM Practice Questions with Queries & Expected Output

### 1. Retrieve all restaurants that serve Indian cuisine.

**ORM Query:**
```
Restaurant.objects.filter(restaurant_type='IN')
```

**Expected Output:**
```
[Restaurant(id=1, name='Spicy Bites'), Restaurant(id=5, name='Tandoori House')]
```

### 2. Get all restaurants that opened after January 1, 2020.

**ORM Query:**
```
Restaurant.objects.filter(date_opened__gt='2020-01-01')
```

**Expected Output:**
```
[Restaurant(id=3, name='Gourmet Palace'), Restaurant(id=7, name='Fusion Dine')]
```

### 3. Fetch the restaurant with the highest latitude.

**ORM Query:**
```
Restaurant.objects.order_by('-latitude').first()
```

**Expected Output:**
```
Restaurant(id=4, name='Mountain View Diner')
```

### 4. Count the number of restaurants of each type.

**ORM Query:**
```
Restaurant.objects.values('restaurant_type').annotate(count=models.Count('id'))
```

**Expected Output:**
```
[{'restaurant_type': 'IN', 'count': 3}, {'restaurant_type': 'CH', 'count': 2}, ...]
```

### 5. Find the total number of restaurants that have received at least one rating.

**ORM Query:**
```
Restaurant.objects.filter(ratings__isnull=False).distinct().count()
```

**Expected Output:**
```
15
```

### 6. Retrieve all restaurants that do not have a website.

**ORM Query:**
```
Restaurant.objects.filter(website='')
```

# Django ORM Practice Questions with Queries & Expected Output

**Expected Output:**
[Restaurant(id=2, name='Budget Eats'), Restaurant(id=6, name='Old Town Grill')]

### 7. Get all ratings for a specific restaurant (restaurant_id=5).

**ORM Query:**
Rating.objects.filter(restaurant_id=5)

**Expected Output:**
[Rating(id=12, rating=5), Rating(id=15, rating=4)]

### 8. Fetch all sales records for a restaurant named 'Spicy Bites'.

**ORM Query:**
Sale.objects.filter(restaurant__name='Spicy Bites')

**Expected Output:**
[Sale(id=7, income=500.00), Sale(id=12, income=750.50)]

### 9. Get the total number of ratings given by a specific user.

**ORM Query:**
Rating.objects.filter(user_id=3).count()

**Expected Output:**
8

### 10. Retrieve all restaurants that have an average rating greater than 4.0.

**ORM Query:**
Restaurant.objects.annotate(avg_rating=models.Avg('ratings__rating')).filter(avg_rating__gt=4.0)

**Expected Output:**
[Restaurant(id=1, name='Elite Dine'), Restaurant(id=4, name='Tandoori House')]

### 11. List the top 5 highest-rated restaurants.

**ORM Query:**
Restaurant.objects.annotate(avg_rating=models.Avg('ratings__rating')).order_by('-avg_rating')[:5]

**Expected Output:**
[Restaurant(id=8, name='Ocean Breeze'), Restaurant(id=2, name='Mountain View')]

# Django ORM Practice Questions with Queries & Expected Output

### 12. Find the total sales income generated by a particular restaurant.

**ORM Query:**
```
Sale.objects.filter(restaurant_id=3).aggregate(total_income=models.Sum('income'))
```

**Expected Output:**
```
{'total_income': 3450.75}
```

### 13. Get all restaurants that have never been rated.

**ORM Query:**
```
Restaurant.objects.filter(ratings__isnull=True)
```

**Expected Output:**
```
[Restaurant(id=6, name='New Tastes'), Restaurant(id=9, name='Hidden Gem')]
```

### 14. Retrieve the latest sale record for each restaurant.

**ORM Query:**
```
Sale.objects.annotate(latest_sale=models.Max('datetime')).order_by('-latest_sale')
```

**Expected Output:**
```
[Sale(id=21, restaurant='Ocean Breeze', datetime='2025-03-30')]
```

### 15. Get all users who have rated at least 3 different restaurants.

**ORM Query:**
```
User.objects.annotate(num_restaurants=models.Count('ratings__restaurant',
distinct=True)).filter(num_restaurants__gte=3)
```

**Expected Output:**
```
[User(id=5, username='john_doe'), User(id=8, username='alice_wonder')]
```

### 16. Retrieve the restaurant with the highest total income from sales.

**ORM Query:**
```
Restaurant.objects.annotate(total_income=models.Sum('sales__income')).order_by('-total_i
ncome').first()
```

**Expected Output:**
```
Restaurant(id=3, name='Fine Dining Experience')
```

### 17. Get the restaurant that has received the highest number of ratings.

# Django ORM Practice Questions with Queries & Expected Output

**ORM Query:**
```
Restaurant.objects.annotate(num_ratings=models.Count('ratings')).order_by('-num_ratings'
).first()
```

**Expected Output:**
```
Restaurant(id=5, name='Burger Haven')
```

### 18. Find the average rating for each restaurant.

**ORM Query:**
```
Restaurant.objects.annotate(avg_rating=models.Avg('ratings__rating'))
```

**Expected Output:**
```
[Restaurant(id=1, avg_rating=4.3), Restaurant(id=2, avg_rating=3.8)]
```

### 19. Retrieve the restaurant with the lowest average rating.

**ORM Query:**
```
Restaurant.objects.annotate(avg_rating=models.Avg('ratings__rating')).order_by('avg_rati
ng').first()
```

**Expected Output:**
```
Restaurant(id=9, name='Low Quality Dine')
```

### 20. Find all restaurants that have received at least one 5-star rating.

**ORM Query:**
```
Restaurant.objects.filter(ratings__rating=5).distinct()
```

**Expected Output:**
```
[Restaurant(id=1, name='Top Notch'), Restaurant(id=4, name='Spicy Delights')]
```

### 21. Get the total income generated by all restaurants combined.

**ORM Query:**
```
Sale.objects.aggregate(total_income=models.Sum('income'))
```

**Expected Output:**
```
{'total_income': 189230.45}
```

### 22. Retrieve the total number of restaurants without any sales records.

**ORM Query:**
```
Restaurant.objects.filter(sales__isnull=True).count()
```

# Django ORM Practice Questions with Queries & Expected Output

**Expected Output:**
7

### 23. Find restaurants that received a rating lower than their average rating.

**ORM Query:**
```
Restaurant.objects.annotate(avg_rating=models.Avg('ratings__rating')).filter(ratings__rating__lt=models.F('avg_rating'))
```

**Expected Output:**
[Restaurant(id=4, name='Midway Cafe')]

### 24. Retrieve the top 3 best-selling restaurants based on total income.

**ORM Query:**
```
Restaurant.objects.annotate(total_income=models.Sum('sales__income')).order_by('-total_income')[:3]
```

**Expected Output:**
[Restaurant(id=1, name='Luxury Eatery'), Restaurant(id=2, name='Casual Dine')]

### 25. Find the user who has given the most number of ratings.

**ORM Query:**
```
User.objects.annotate(num_ratings=models.Count('ratings')).order_by('-num_ratings').first()
```

**Expected Output:**
User(id=7, username='top_reviewer')

### 26. Find the month-wise total sales income for each restaurant.

**ORM Query:**
```
Sale.objects.values('restaurant__name', 'datetime__month').annotate(total_income=models.Sum('income')).order_by('restaurant__name', 'datetime__month')
```

**Expected Output:**
[{'restaurant__name': 'Spicy Bites', 'datetime__month': 2, 'total_income': 7500.50}, ...]