SQL

Window functions

Table Schema

dbo

order_id	order_date	customer_name	city	order_amount ~~money~~
1001	04/01/2017	David Smith	Guildford	10,000
1002	04/02/2017	David Jones	Arlington	20000
1003	04/03/2017	John Smith	Shalford	5000
1004	04/04/2017	Michael Smith	Guildford	15000
1005	04/05/2017	David Williams	Shalford	7000
1006	04/06/2017	Paum Smith	Guildford	25000
1007	04/10/2017	Andrew Smith	Arlington	15000
1008	04/11/2017	David Brown	Arlington	2000
1009	04/20/2017	Robert Smith	Shalford	1000
1010	04/25/2017	Peter Smith	Guildford	50

SUM() : We want the sum of order_amount as per city.

Normal

```
SELECT city, SUM(order_amount) as total_
order_amount FROM dbo GROUP BY city
```

output

city	total_order_amount
Arlington	37000.00
Guildford	50500.00
Shalford	13000.00

Window

```
SELECT order_id, order_date, custo
-mer_date, city, order_amount,
sum(order_amount) OVER (
PARTITION BY city) as grand_total
FROM dbo
```

- this value is displayed as a new column as grand_total.

RANK()

```
SELECT order_id, order_date, customer_name, city,
RANK() OVER (ORDER BY order_amount DESC) as RANK
FROM dbo
```

```
RANK
 3
 3
 5
```

Dense_RANK()

RANK
3
3
4

```
SELECT order_id, order_date, customer_name, city, order_amount
DENSE_RANK() OVER (ORDER BY order_amount DESC)
as RANK FROM dbo.
```

ROW_NUBER() with Order BY

```
SELECT order_id, order_date, Customer_name, city, order_amount,
ROW_NUMBER() OVER (ORDER BY order_Id) as row_number
FROM dbo.
```

ROW_NUMBER() with PARTITION BY

```
SELECT order_id, order_date, customer_name, city, order_amount,
ROW_NUMBER() OVER (PARTITION BY city ORDER BY order_amount DESC)
as row_number FROM dbo.
```

city	order_amt	row_nuber
Arlington	20000	1
Arlington	15000	2
Arllington	2000	3
guildford	25000	1
guildford	15000	2

NITILE () → divides the database in quartiles . Q1 , Q2 → Q1, Q2, Q3, Q4

```
SELECT *, NITILE(4) OVER(ORDER BY order_amount)
as row_number FROM dbo
```

LAG () or LEAD()

SELECT order-id, customer-name, city, order-amount, order-date,
-- in below line, 1 indicates check for previous row of the current row
LAG(order-date, 1) over (order By order-date) as prev-order-date
FROM dbo

Calculating data of prev rows (dates) in next row (prev order dates))

Lead is just opposite of LAG. It helps access data of next row in the prev row.

Select order-id, customer-name, city, order-amount, order-date,
-- in below line, 1 indicates check for next row of the current row
LEAD (order-date, 1) OVER (order by order-date) as next-order-date
FROM dbo.

FIRST VALUE () and LAST VALUE()

SELECT order-id, order-date, customer-name, city, order-amount,
FIRST-VALUE (order-date) OVER (Partition By City Order By City) as
firstorder date,
LAST_VALUE (order-date) OVER (Partition By City order by City) as
last order date
FROM dbo.

AVG ()

select order-id, order-date, customer-name, city, order-amount,
AVG (order-amount) over (Partition By city, month(order date)) as
average order amount from dbo.

Min()

```
select order-id, order-date, customer-name, city, order-amount
, Min (order-amount) over (Partition By city) as minimum order amount
From dbo.
```

Max()   COUNT()

* window function does not work with a distinct clause. They will take in all the duplicate values present in the Tables. Hence window function does not help resolve problem of duplicates.