Sample LaTeX file for Mathspace to show what I am capable of. Questions were from an assignment.

1. Where do most of our customers come from?

Solution:

```
CREATE VIEW TotalCustomersPerCountry AS
SELECT Country, COUNT(*) as Total
FROM Customers
GROUP BY Country;

SELECT Country, Total
FROM  TotalCustomersPerCountry
WHERE Total = (SELECT MAX(Total)
               FROM TotalCustomersPerCountry);
```

Output:

Country	Total
USA	13

Assumptions: The interpretation of 'most' means the country/countries with the highest number of customers.

Explanation of Code: The first thing to do is to create a view of the countries and their customer count. We first need to group by the column Country in the Customers table and then we apply the aggregate function COUNT(*) to perform the summary. The resulting view will look something like this:

Country	Total
Argentina	3
Austria	2
Belgium	2
⋮	⋮
USA	13
Venezuela	4

All we simply need to do now is to use this table by applying the aggregate function MAX(Total) on the Total. Since doing this will just yield a number and we want the name of the country. We use the aforementioned in a subquery and compare that to the Total which results in the desired output and the most customers are from USA.

Edge case to consider: If there exists two or more countries that have the same amount of customers, the query correctly accounts for this and returns all the country names with the maximum number of customers.

2. How many employees have a tertiary education?

Solution:

```
SELECT COUNT(*) as NumEmpWithTertiary
FROM Employees
WHERE Notes LIKE '%University%' OR
      Notes LIKE '%College%';
```

Output:

NumEmpWithTertiary
9

Assumptions: There is only one table that gives information about the tertiary education of an employee, the Employees table (under the column of Notes). However, the data seems to be stored as a string or text. This makes it particular hard to detect people who have tertiary education because a person can enter something such as 'studied at an instituition'. So here I will assume tertiary education means studied at **University** or **College**.

Doing the aforementioned is not a fool proof way. Someone in the notes could mention 'I visited my friend at University' and a false positive would be returned. (See suggestions).

Explanation of the code: Basically we filter through those employees who's notes had the words University or College in them. Then we count them.

Suggestions: We should make a separate table that has a list of possible classifications of what tertiary means. For example we can define tertiary as being people who have completed 'University, College or TAFE' and we introduce a foreign key in the Employees table and link it. Consequently, the dependency of correctly typing your education details in Notes is removed.

Also if we cannot do anything about the database schema and we had no choice but to pull education details in the Notes columns, we could use *Regular Expressions* if we wanted more potency and control over what we match.

3. Which Shipper sent the most orders in 1997?

Solution:

```
CREATE VIEW OrdersPerShipper AS
SELECT ShipperID, COUNT(*) as NumOrders
FROM Orders
```

```
WHERE OrderDate LIKE '1997%'
GROUP BY ShipperID;

SELECT s.ShipperName, ops.NumOrders
FROM Shippers s, OrdersPerShipper ops
WHERE s.ShipperID = ops.ShipperID AND
      ops.NumOrders = (SELECT MAX(ops.NumOrders)
                       FROM OrdersPerShipper ops);
```

Output:

ShipperName	NumOrders
United Package	18

Assumptions: A customer may place an order on several products (I will treat this as being a single order). Also in a lot of databases they have some sort of Date datatype. But it seems on w3schools the date is stored as a string. So for the sake of making the code run on the given editor, I used 'LIKE' when I would normally use sort of 'year' function.

4. How many different products are bought on average per order?

Solution:

```
CREATE VIEW ProductCountPerOrder AS
SELECT OrderID, COUNT(*) as NumProducts
FROM OrderDetails
GROUP BY OrderID;

SELECT AVG(NumProducts) as AvgProductsOrdered
FROM ProductCountPerOrder;
```

Output:

AvgProductsOrdered
2.642857142857143

Assumptions: I intepret the words 'different products' as follows. For example, OrderID 10248 orders 12 of the product with a ProductID of 11. This is treated as one unique product being bought.

Explanation of the code: We use the GROUP BY clause and COUNT(*) aggregate function to create a view with how many products a particular order had. The next statement averages the number of products sold per order.

Edge case/Technicalities: Any sensible schema would enforce a foreign key constraint from OrderDetails.ProductID to Products.ProductID. However let us pretend we are not in that world. If we have two tuples in OrderDetails like so:

OrderID	ProductID	Quantity
10248	23	10
10248	23	11

The above code would treat these as two different products. To combat this we replace COUNT(*) with COUNT(DISTINCT ProductID) instead.

5. Based on products which sold in September 1996, what proportion did each product contribute to total revenue that month?

Solution:

Although I have answered Q5, Q6 with a general SQL statement that works for all RDMS that follow the SQL standard, we can however use PL/pgSQL to illustrate that you can kill two or more birds with one stone. Let's say we asked what is the top X products that

generated the most revenue from date A to date B - we can do this by reusing previously written code in a cleaner manner and it gives us procedure options.

```
CREATE VIEW RevenueSept96 AS
SELECT p.ProductID, SUM(od.Quantity) * p.Price as ProductRevenueSept96
FROM Orders o, OrderDetails od, Products p
WHERE o.OrderDate LIKE '1996-09%' and
      o.OrderID = od.OrderID and
      od.ProductID = p.ProductID
GROUP BY p.ProductID;

SELECT ProductID, ProductRevenueSept96/
      (SELECT SUM(ProductRevenueSept96) FROM RevenueSept96) AS Proportion
FROM RevenueSept96;
```

Output: Output consists of 35 rows, so I display a few to save space

ProductID	Proportion
1	0.010414979054319902
2	0.02198717800356424
4	0.012729418844168769
6	0.02169787302983313
⋮	⋮
72	0.020135626171685145
75	0.00224211354641609

Assumptions: I define proportion to mean as follows: The proportion of a products' p contribution is expressed as a decimal which equals to the formula,

$$p.\text{Proportion} = p.\text{Price} \times p.\text{Quantity} / \sum_{\substack{\text{All Sept96} \\ \text{products } p'}} (p'.\text{Price} \times p'.\text{Quantity})$$

Explanation of the code: Perform a series of joins to retreive relevant column information from each table. Then aggregate using the SUM function times price. This will generate the revenue. The next statement simply divides the revenue/total revenue to get the proportion.

6. Based on products which sold in September 1996, what are the top 3 which we saw most growth in revenue and what are the top 3 which we had saw the largest decrease in revenue in the following month?

Solution: .

As discussed via the email! I will utilize PostgreSQL's syntax as it has a handful of useful functions such as coalesce, dense_ranking, extract. My code covers cases such as when there are tied top products. We can answer this question in **two ways**.

```
CREATE VIEW RevenueOct96 AS
SELECT p.ProductID, SUM(od.Quantity) * p.Price AS ProductRevenueOct96
FROM Orders o, OrderDetails od, Products p
WHERE EXTRACT(year from o.OrderDate) = 1996 AND
      EXTRACT(month from o.OrderDate) = 10 AND
      o.OrderID = od.OrderID AND
      od.ProductID = p.ProductID
GROUP BY p.ProductID;

CREATE VIEW ProductGrowth AS
SELECT COALESCE(ro.ProductID, rs.ProductID) as ProductID,
      (COALESCE(ro.ProductRevenueOct96,0::MONEY)
      -COALESCE(rs.ProductRevenueSept96,0::MONEY))
       as RevenueGrowth
FROM RevenueSept96 rs
FULL OUTER JOIN RevenueOct96 ro ON rs.ProductID = ro.ProductID
ORDER BY RevenueGrowth;

CREATE VIEW ProductGrowthIncreaseRanks AS
SELECT ProductID, RevenueGrowth,
       DENSE_RANK() OVER(ORDER BY RevenueGrowth DESC)
       as GreatestIncreaseRank
FROM ProductGrowth;

CREATE VIEW ProductGrowthDecreaseRanks AS
SELECT ProductID, RevenueGrowth,
       DENSE_RANK()
       OVER(ORDER BY RevenueGrowth)
       as GreatestDecreaseRank
FROM ProductGrowth;

SELECT *
FROM ProductGrowthIncreaseRanks
WHERE GreatestIncreaseRank <=3;

SELECT *
FROM ProductGrowthDecreaseRanks
WHERE GreatestDecreaseRank <=3;
```

Output:

ProductID	RevenueGrowth	GreatestIncreaseRank
38	$5,270.00	1
63	$3,512.00	2
51	$2,544.00	3

ProductID	RevenueGrowth	GreatestDecreaseRank
62	-$4,782.10	1
29	-$3,094.75	2
39	-$1,548.00	3

Assumptions: We will reuse the view RevenueSept96 and similarly define RevenueOct96. This is of course generalizable and can be encapsulated in a function if the RDBMS allows for it. We can define growth in revenue of a product (from Sept to Oct) as (For that specified product):

Price in Oct $\times$ Quantity Oct - Price in Sept $\times$ Quantity in Sept.

Explanation: Like Q5, we create a view that has the revenue generated for each product ordered in October 1996. The first resulting view called RevenueOct96 looks like this:

ProductID	ProductRevenueOct96
2	$1,064.00
4	$1,144.00
6	$150.00
⋮	⋮

Recall that the view for RevenueSept96 looks like:

ProductID	ProductRevenueSept96
1	$360.00
2	$760.00
4	$440.00
⋮	⋮

Now we perform a full outer join. Here is the reason we do this: If the product was sold in Sept but not Oct, we need to include the revenue decrease and we treat the revenue in Oct

as 0 which is purpose of coalesce. Similarly if a product was sold in Oct but not Sept we need to record to revenue in Sept as 0. Thus, giving us the revenue growth we desire.

ProductID	RevenueGrowth
⋮	⋮
25	$56.00
13	$72.00
54	$74.50
⋮	⋮

If we want the top 3 (including ties) we need to use PostgreSQL's DENSE_RANK function and then chop off the ranks from 1 to 3 which gives the output.

Technicalities: Of course, this may not be the best way to measure revenue growth.

Let us define growth in revenue of a product (from Sept to Oct) as (For that specified product):

{(Revenue Oct) - (Revenue Sept)}/(Revenue Sept). Notice that in the case where no quantity of a product sold in September but did in October the ratio is something divided by 0 which doesn't make sense. So we will assume the product must have revenue in September - so we use a LEFT OUTER JOIN instead. We may prefer this way - a percentage of growth rather than a concrete number.

The change in the code is in the ProductGrowth view.

```
CREATE VIEW ProductGrowth AS
SELECT rs.ProductID,
      ((COALESCE(ro.ProductRevenueOct96,0::MONEY)
       - rs.ProductRevenueSept96)
       / rs.ProductRevenueSept96) * 100 as RevenueGrowth
FROM RevenueSept96 rs
LEFT OUTER JOIN RevenueOct96 ro ON rs.ProductID = ro.ProductID
ORDER BY RevenueGrowth;
```

ProductID	RevenueGrowth	GreatestIncreaseRank
7	1300	1
63	480	2
51	400	3

ProductID	RevenueGrowth	GreatestDecreaseRank
40	-100	1
66	-100	1
70	-100	1
53	-100	1
49	-100	1
29	-100	1
69	-100	1
34	-100	1
1	-100	1
39	-95.5	2
32	-85	31

As you can see products that sold in Sept but not Oct automatically have a 100 percent decrease and a decrease rank or 1.

7. In order to understand customer behaviours, marketers often conduct an Recency-Frequency-Monetary analysis. Based on the data provided, what are some metrics you could create to measure these aspects of customer behaviour?

Solution: I believe I can code all of these (except for the similarity suggestion) up in a reasonable amount of time. But for the sake of keeping this document short (because it is already long - sorry about that I won't keep you for much longer!).

- We can measure the average 'gap' between purchases or orders. For example Michelle makes a purchase/order on 7th Dec, 8th Dec, 9th Dec whilst Joan makes a purchase 10th July, 18 Sept the same year. Michelle would be the 'prefered' target and she would be ranked higher. A reason for this is Joan might be only purchasing through promotional periods.

- How far was the last purchase date away from the current date?

- We might like to know for each product, what type of people (in terms of country) is purchasing it the most. For example, Canadians might purchase some maple syrup the most and that would indicate that product is great as they are willing to forego import taxes to purchase what they already have in their country.

- We may want to know on average, what are customer's spending per transaction. Further, is there any variance on each of these transactions? Or are they relatively the same amount.

- We might like to rank customers via a similarity score (Kind of like a collaborative filtering algorithm - NetFlix Prize). Doing this we might make certain kinds of emails/ads targetting a similar group of customers.

- Thoroughout a certain period how much times has the customer bought an item (or just bought in general).

- What porportion of customers have you retained. A simple retention rate would be: (Total Customers who came back more then X time)/Total Customers within a reasonable time period). X could be 2 or 3 or however we feel is suited.

- What is the most purchased category of items. This will indicate what the customer 'perceives' the company to be. Even though this is a food company are they buying stuff from other categories (just solely diary products, only cheeses?).

- Do the customers tend to go for a certain supplier? If so we might like to reason why this is the case.

- Total amount a customer has spent on a product or just in general.

- Amount of times customers 'beat' the average - ordered higher quantities of a product than the average for that time period.