**ST1501 CA2 Group Task**

Class**:** DAAA/FT/2A/02

Group No: 2

Group Members:

| **AdmNo** | **Name** |
| --- | --- |
| **2100788** | **Lim Jun Jie** |
| **2112729** | **Joel Poah** |
| **2106911** | **Timothy Chia** |

Data Warehouse Schema (Database Diagram)

|  | |
| --- | --- |

Queries Explanation

| Query 1 (Sales/profits/discounts/revenue) |
| --- |
| Explanation |
| - This query looks at the past 3 months (not inclusive of the april as it has not ended yet) with reference to the latest transaction which was 2nd April 2018 based on the dataset.  -It then partitions the results into each month  -It then rank the items based on the revenue it has generated which is calculated by the list\_price \* quantity sold and also deducting the discount.  -Shows Top 5 product for each of the 3 months based on revenue  - Query also projects the TotalDiscounted which is how much the company could have earned if there were no discounts at all.  **The owner can then choose whether to reduce the discounts for these products in order to earn more since it's on demand.** |
| SQL Script |
|  |

| Query 2 (Sales/Staff/stores) |
| --- |
| Explanation |
| * This query checks staff performance by measuring the average yearly revenue generated for each staff in each store from the past two years in the dataset (i.e. 2016 and 2017). We do not include 2018 as our data warehouse does not contain a full year's worth of transactions for said year. * The query sums the revenue of orders that are not rejected then is divided by 2 years. * The query also checks for staff type to help the business owner visualise over performing/underperforming staff. * The results are partitioned by store\_id in descending order and ranked based on revenue generated. * From the partition, stores generating the most revenue can also be seen. * The inner join on sf.staff\_key and s.staff\_key excludes staff with no transactions managed. * Staff with a higher revenue can be considered by the owner for a promotion and those not making any revenue may prompt the owner to investigate those staff |
| SQL Script |
|  |

| Query 3 Sales/Seasons of Sales/Time |
| --- |
| Explanation |
| * This query displays the average percentage change per quarter from 2016 to 2017 for each bike category. * Percentage change is calculated by getting the difference in revenue from the current quarter and the lagged revenue from the previous quarter, then dividing the difference with the previous revenue value. * Because we are dealing with lagged values, the first quarter’s difference in revenue with itself will be zero and will cause a “divide by zero” error. * To handle this, differences in revenue that equal zero will be replaced with the average revenue per quarter per category. * We pivot the projected table to turn bike categories into columns and each quarter into their respective rows, this is to better visualise the percentage change. * The owner can analyze the percentage change per quarter and can see which quarters which categories generate the most revenue. * To improve the business, we can import more bikes that are hot in that quarter and import lesser bikes for the categories that have a negative percent change. |
| SQL Script |
|  |

| Query 4 Sales/orders/customers |
| --- |
| Explanation |
| -This Query shows the top 10 least efficient delivery made to the customers living in various States and cities.  - It shows the State and the City that often take a long time for delivery to take place and also how many days it is late by .  - DayTaken is calculated by taking the shipped date - order date.  - DayLate is calculated by taking ShippedDate - RequiredDate by customer.  **The owner can then choose whether to take action in making the the deliveries to these cities more efficiently** |
| SQL Script |
|  |

| Query 5 Sales/Products/Brands/categories/inventory |
| --- |
| Explanation |
| The aim of this query is to find out the Top 15 items that will have to restock soon and also the date it should be restocked by  - Query gets all the transactions and those products that are selling out fast in the past 4 weeks  - The query is dynamic which means that the date is not hard-coded and no modifications will have to be made in the future. It will automatically recognise the latest transaction and find all transactions in the past 4 weeks from then.  - The query assumes that the same amount of products will be sold for the next few months  - It then forecasts the restock date based on how much was sold in the previous month  **The owner can prepare and decide whether to restock these products based on the recommended date. This is a very useful insight such that the owner does not miss out on the fast selling products and also can restock earlier to meet the demand** |
| SQL Script |
|  |