Business Case

Email Request, Request for Development of Toman Bike Share Dashboard :

Dear Data Analyst,

We need your expertise to develop a dashboard for “Toman Bike Share” that displays our key performance metrics for informed decision-making.

Objective :

The Chief Financial Officer (CFO) David wants to determine whether Toman Bike Share needs to adjust their prices in the following fiscal year.

Target Audience :

- Primary : CFO David
- Secondary : the financial team working under CFO David

Deadline :

- We need a preliminary version ASAP

Use Cases and Data Required

User Story :

As the CFO of Toman Bike Share, I want to utilize revenue and demographic trend data to help guide price adjustments for the next year. With the adjustment of our prices, we aim to maximize profit.

Acceptance Criteria :

The dashboard should include :

- Hour Revenue Analysis
- Profit and Revenue Trends
- Seasonal Revenue
- Ride Demographics
- Color-coded to reflect company colors
- Be user-friendly and easy to filter/ sort
- Use the most recent data possible

Success Criteria :

CFO David can :

- Easily identify what price adjustments are needed for the following year based on the analysis and metrics above
- Assess potential for increased profits based on price adjustments
- Make informed decisions while collaborating with the financial team

This will allow CFO David to make the accurate judgment call for price adjustments to maximize profits in the following year.

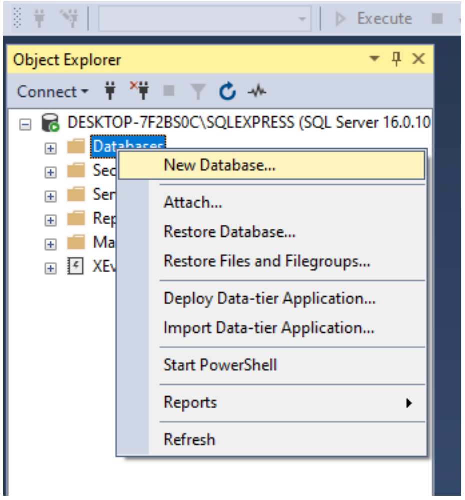
Information Needed :

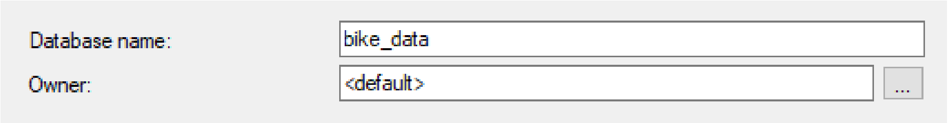
- Hourly Revenue
- Total Profits
- Total Revenue
- Total Seasonal Revenue
- Rider Type
- Number of Riders

Data Need :

- Date (date)
- Season (int)
- Year (int)
- Weekday (int)
- Hour (int)
- Rider Type (nvarchar(50))
- Riders (int)
- Price (money)
- COGS (int)
- Revenue (money)
- Profit (money)

Creating Our Own Database in SSMS

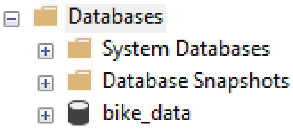

Execute
Object Explorer
Connect
DESKTOP-7F2BS0C\SQLEXPRESS (SQL Server 16.0.10
Databases
Sec
Ser
Rep
Ma
XEv
New Database...
Attach...
Restore Database...
Restore Files and Filegroups...
Deploy Data-tier Application...
Import Data-tier Application...
Start PowerShell
Reports
Refresh



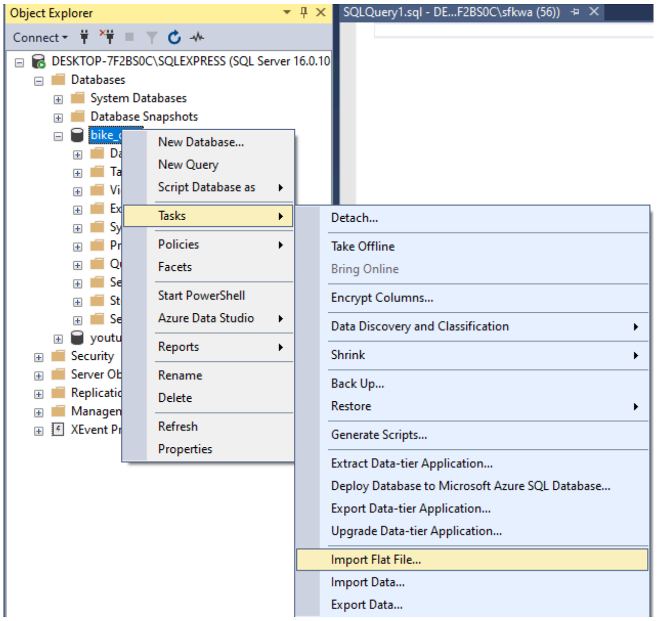
Database name: bike_data

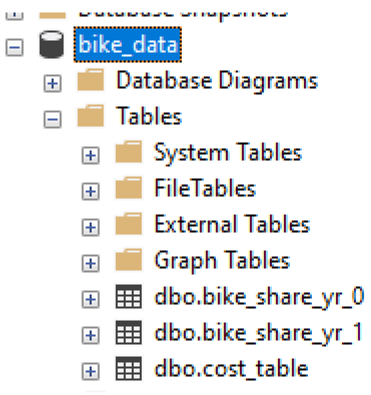
Owner: <default>

=> named the database : bike_data

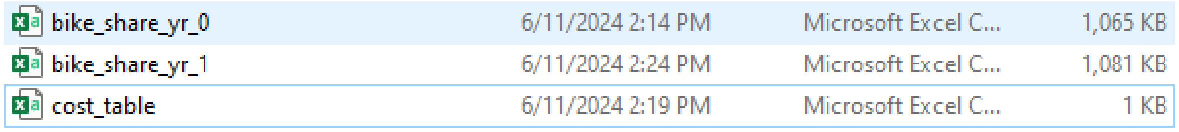


Import Flat File :





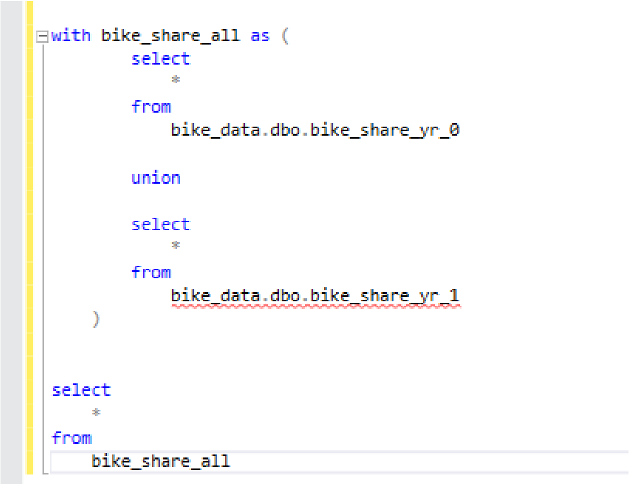
```
⊞ Database Snapshots
⊟ bike_data
    ⊞ Database Diagrams
    ⊟ Tables
        ⊞ System Tables
        ⊞ FileTables
        ⊞ External Tables
        ⊞ Graph Tables
        ⊞ dbo.bike_share_yr_0
        ⊞ dbo.bike_share_yr_1
        ⊞ dbo.cost_table
```



bike_share_yr_0	6/11/2024 2:14 PM	Microsoft Excel C...	1,065 KB
bike_share_yr_1	6/11/2024 2:24 PM	Microsoft Excel C...	1,081 KB
cost_table	6/11/2024 2:19 PM	Microsoft Excel C...	1 KB

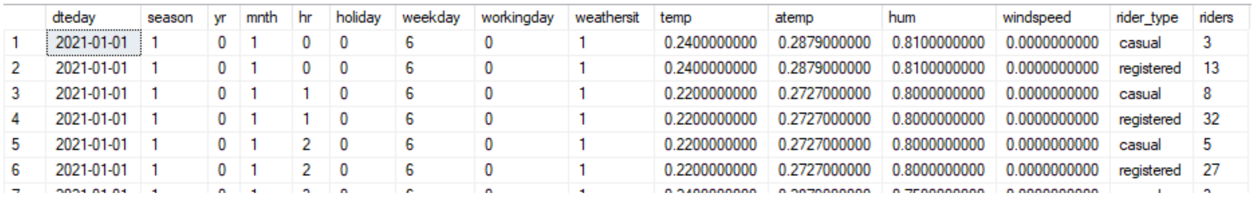
=> https://github.com/Gaelim/YT_bike_share

Data Cleaning with SSMS



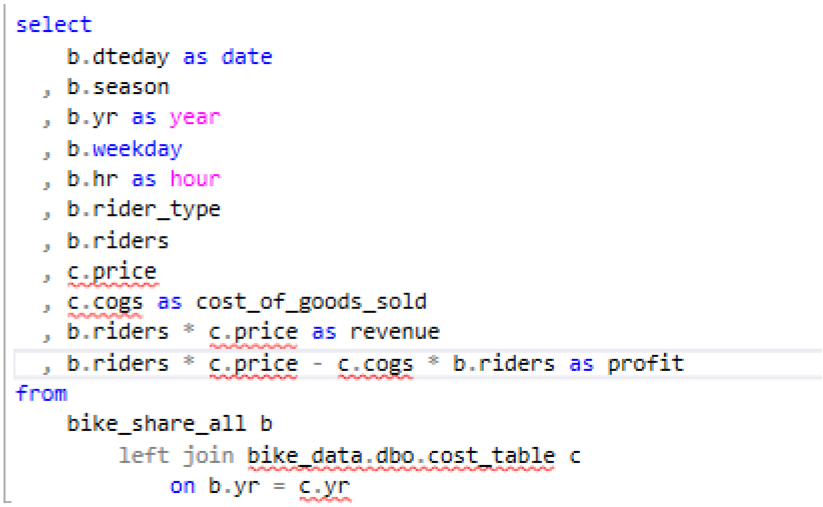
```
with bike_share_all as (
        select
            *
        from
            bike_data.dbo.bike_share_yr_0

        union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
    )


select
    *
from
    bike_share_all
```

=> we create a CTE where we union the two different datasets to get all datasets - we use union instead of union all to account for duplicates



	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	rider_type	riders
1	2021-01-01	1	0	1	0	0	6	0	1	0.2400000000	0.2879000000	0.8100000000	0.0000000000	casual	3
2	2021-01-01	1	0	1	0	0	6	0	1	0.2400000000	0.2879000000	0.8100000000	0.0000000000	registered	13
3	2021-01-01	1	0	1	1	0	6	0	1	0.2200000000	0.2727000000	0.8000000000	0.0000000000	casual	8
4	2021-01-01	1	0	1	1	0	6	0	1	0.2200000000	0.2727000000	0.8000000000	0.0000000000	registered	32
5	2021-01-01	1	0	1	2	0	6	0	1	0.2200000000	0.2727000000	0.8000000000	0.0000000000	casual	5
6	2021-01-01	1	0	1	2	0	6	0	1	0.2200000000	0.2727000000	0.8000000000	0.0000000000	registered	27

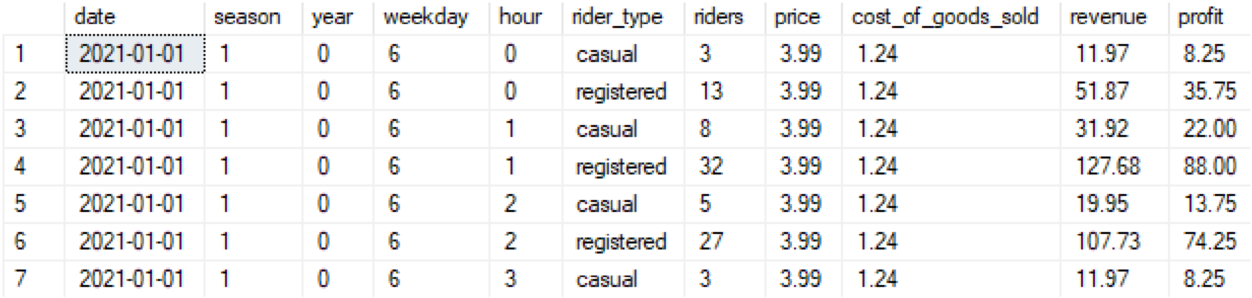
Then with the CTE, I joined it with the cost table to get our individual prices and cogs for each date :



```
select
    b.dteday as date
  , b.season
  , b.yr as year
  , b.weekday
  , b.hr as hour
  , b.rider_type
  , b.riders
  , c.price
  , c.cogs as cost_of_goods_sold
  , b.riders * c.price as revenue
  , b.riders * c.price - c.cogs * b.riders as profit
from
    bike_share_all b
        left join bike_data.dbo.cost_table c
            on b.yr = c.yr
```

=> we also included the calculation for revenue and profit and aliases

RESULT :



	date	season	year	weekday	hour	rider_type	riders	price	cost_of_goods_sold	revenue	profit
1	2021-01-01	1	0	6	0	casual	3	3.99	1.24	11.97	8.25
2	2021-01-01	1	0	6	0	registered	13	3.99	1.24	51.87	35.75
3	2021-01-01	1	0	6	1	casual	8	3.99	1.24	31.92	22.00
4	2021-01-01	1	0	6	1	registered	32	3.99	1.24	127.68	88.00
5	2021-01-01	1	0	6	2	casual	5	3.99	1.24	19.95	13.75
6	2021-01-01	1	0	6	2	registered	27	3.99	1.24	107.73	74.25
7	2021-01-01	1	0	6	3	casual	3	3.99	1.24	11.97	8.25

Data Testing

What are we testing for?

High Quality Data Set - data that is complete and accurate, not missing records or inaccurate data
=> make sure our stakeholders can make accurate decisions based on this data
=> we should be confident in this data

In order to make sure this is true, and we do have high quality data sets, we need to run different tests

Data Quality Tests :

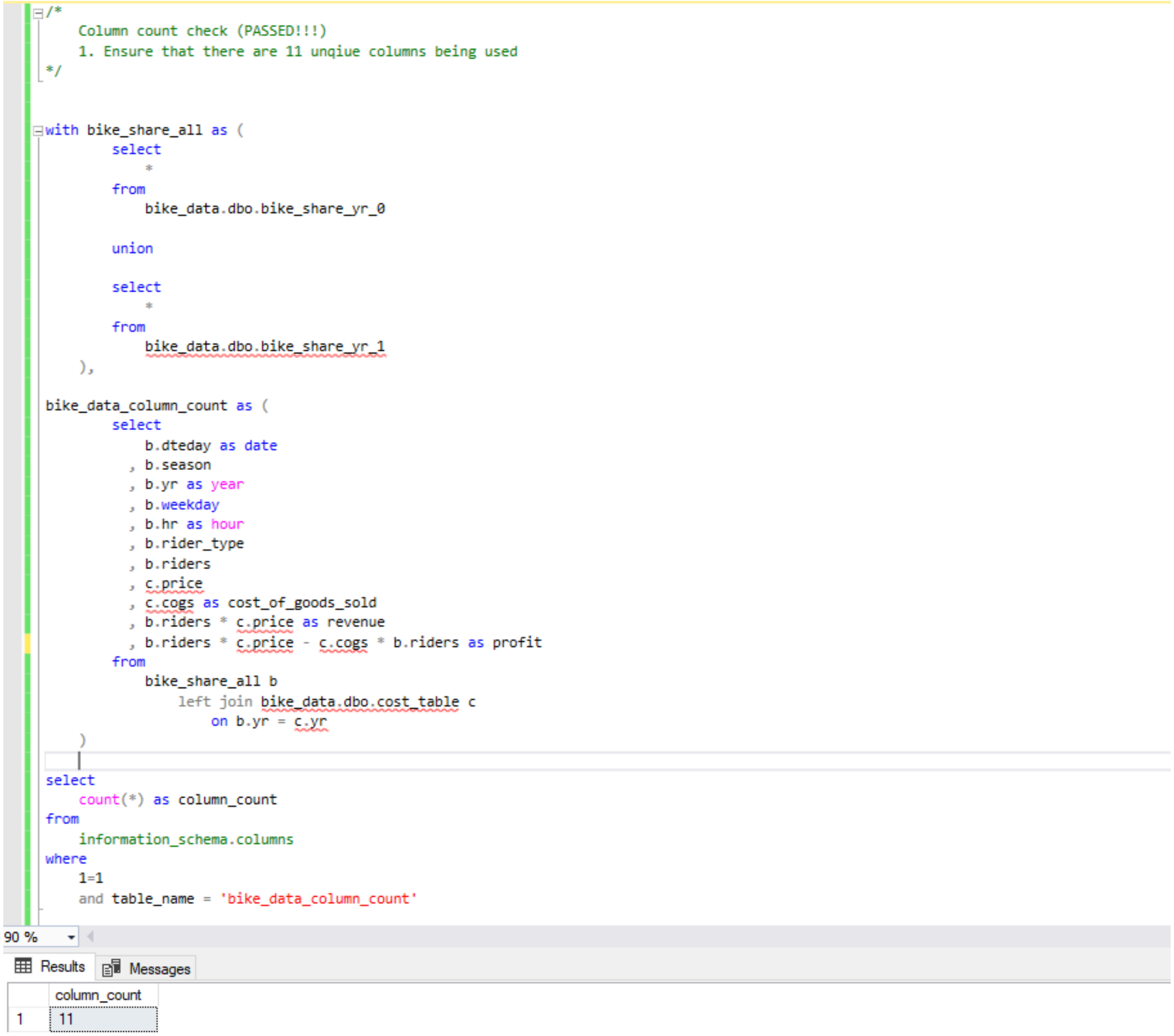
1. The data needs to include 11 columns (column count check)
2. The data needs to contain no duplicates (duplicate check)
3. There must be no null values within the dataset (null values check)
4. Make sure the revenue and profit calculations are accurate (data validity check)

Column count = 11

Duplicate count = 0
Null value count = 0

Revenue and profit figures are accurate

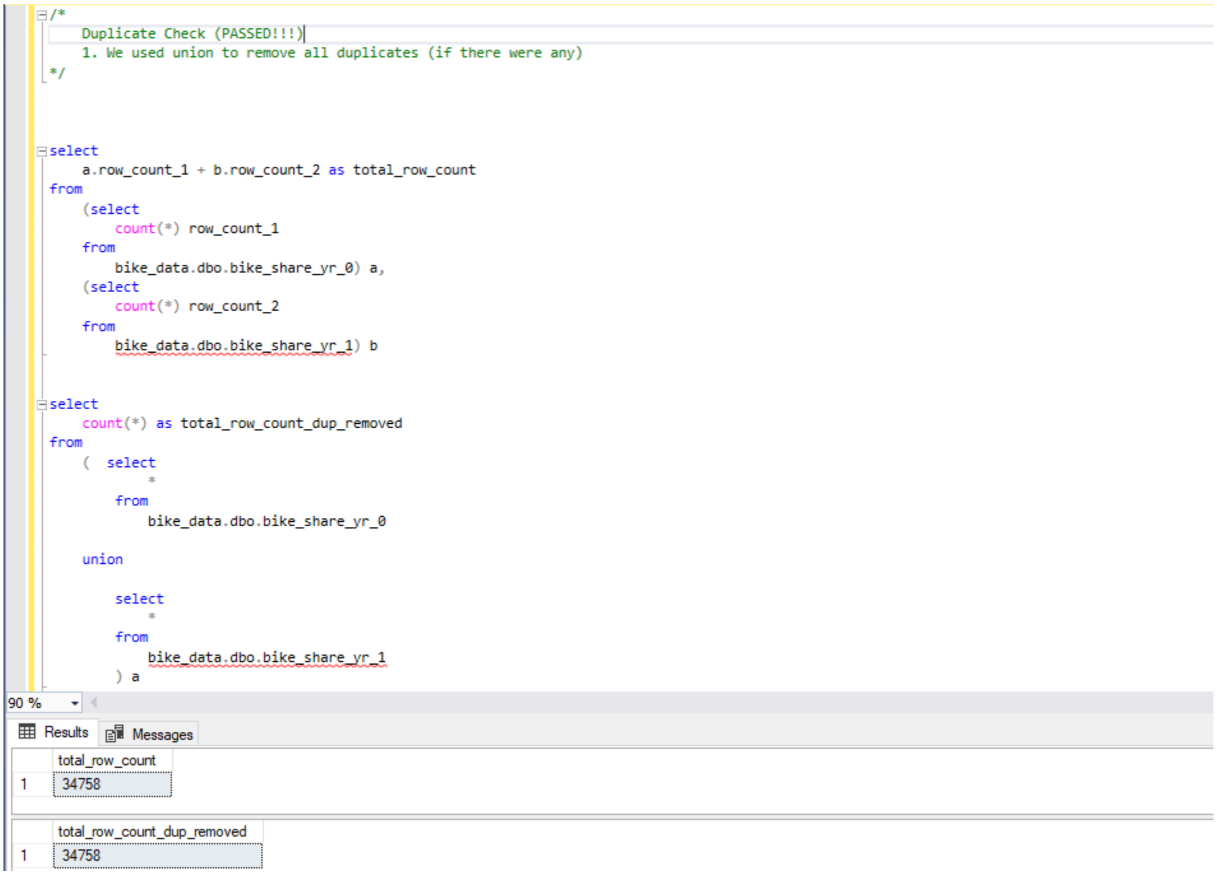
1. Column Count Check



```
/*
    Column count check (PASSED!!!)
    1. Ensure that there are 11 unqiue columns being used
*/

with bike_share_all as (
        select
            *
        from
            bike_data.dbo.bike_share_yr_0

        union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
    ),

bike_data_column_count as (
        select
            b.dteday as date
          , b.season
          , b.yr as year
          , b.weekday
          , b.hr as hour
          , b.rider_type
          , b.riders
          , c.price
          , c.cogs as cost_of_goods_sold
          , b.riders * c.price as revenue
          , b.riders * c.price - c.cogs * b.riders as profit
        from
            bike_share_all b
                left join bike_data.dbo.cost_table c
                    on b.yr = c.yr
    )

select
    count(*) as column_count
from
    information_schema.columns
where
    1=1
    and table_name = 'bike_data_column_count'
```

	column_count
1	11

2. Duplicate Check

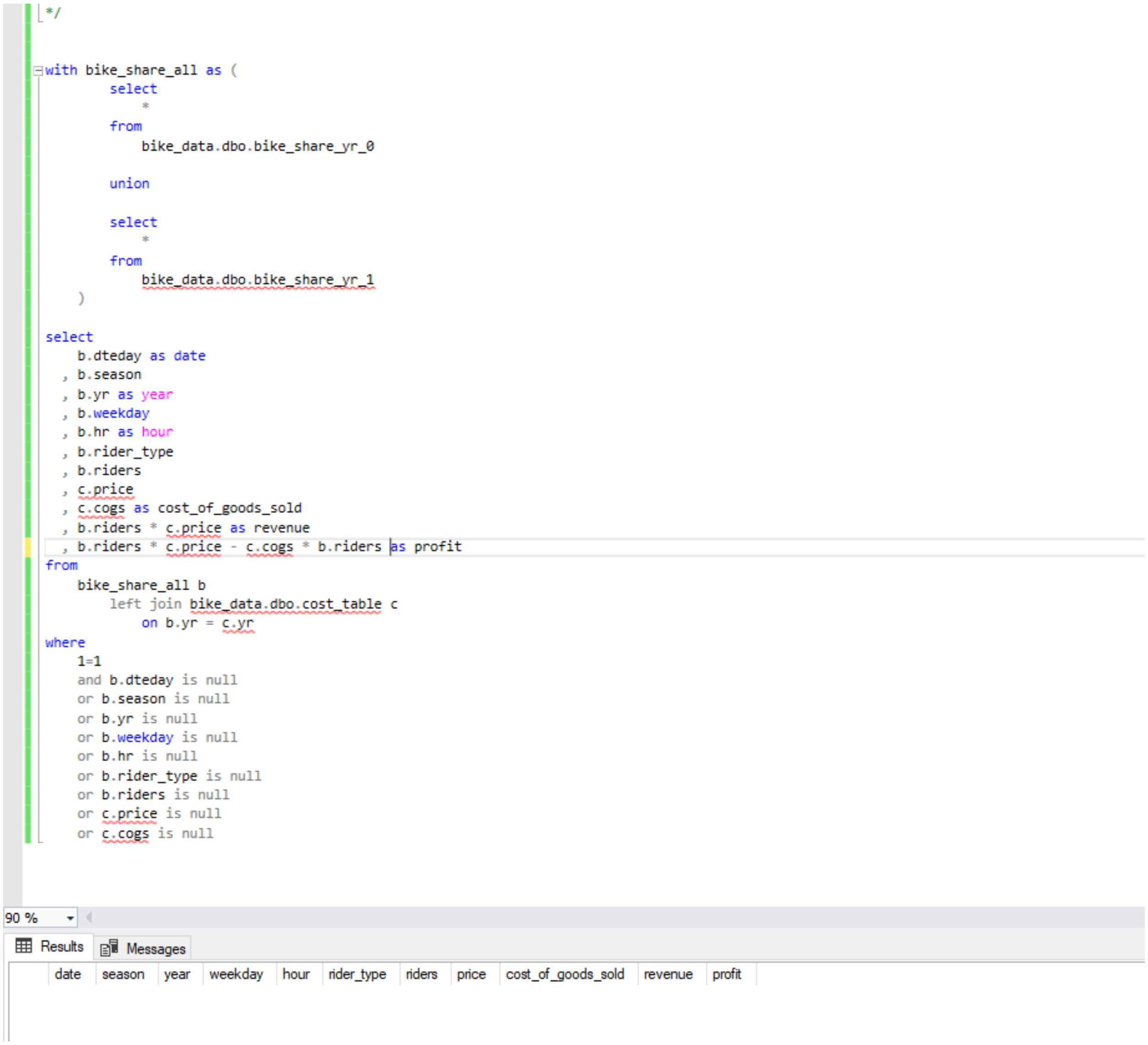


```
/*
    Duplicate Check (PASSED!!!)
    1. We used union to remove all duplicates (if there were any)
*/



select
    a.row_count_1 + b.row_count_2 as total_row_count
from
    (select
        count(*) row_count_1
    from
        bike_data.dbo.bike_share_yr_0) a,
    (select
        count(*) row_count_2
    from
        bike_data.dbo.bike_share_yr_1) b


select
    count(*) as total_row_count_dup_removed
from
    (  select
            *
        from
            bike_data.dbo.bike_share_yr_0

    union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
        ) a
```

	total_row_count
1	34758

	total_row_count_dup_removed
1	34758

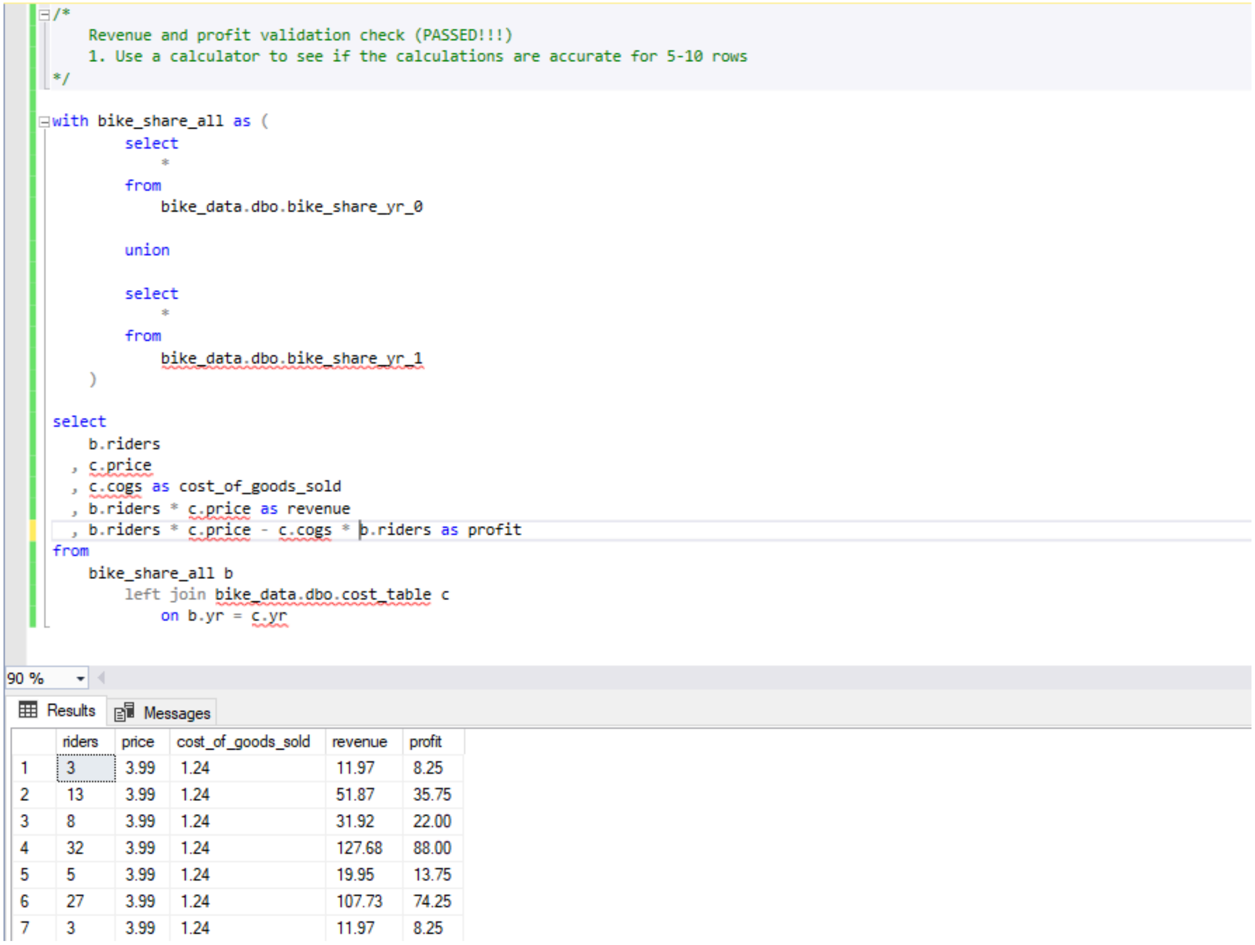
3. Null Value Check



```
*/

with bike_share_all as (
        select
            *
        from
            bike_data.dbo.bike_share_yr_0

        union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
    )

select
    b.dteday as date
  , b.season
  , b.yr as year
  , b.weekday
  , b.hr as hour
  , b.rider_type
  , b.riders
  , c.price
  , c.cogs as cost_of_goods_sold
  , b.riders * c.price as revenue
  , b.riders * c.price - c.cogs * b.riders as profit
from
    bike_share_all b
        left join bike_data.dbo.cost_table c
            on b.yr = c.yr
where
    1=1
    and b.dteday is null
    or b.season is null
    or b.yr is null
    or b.weekday is null
    or b.hr is null
    or b.rider_type is null
    or b.riders is null
    or c.price is null
    or c.cogs is null
```

date	season	year	weekday	hour	rider_type	riders	price	cost_of_goods_sold	revenue	profit

4. Revenue and profit check

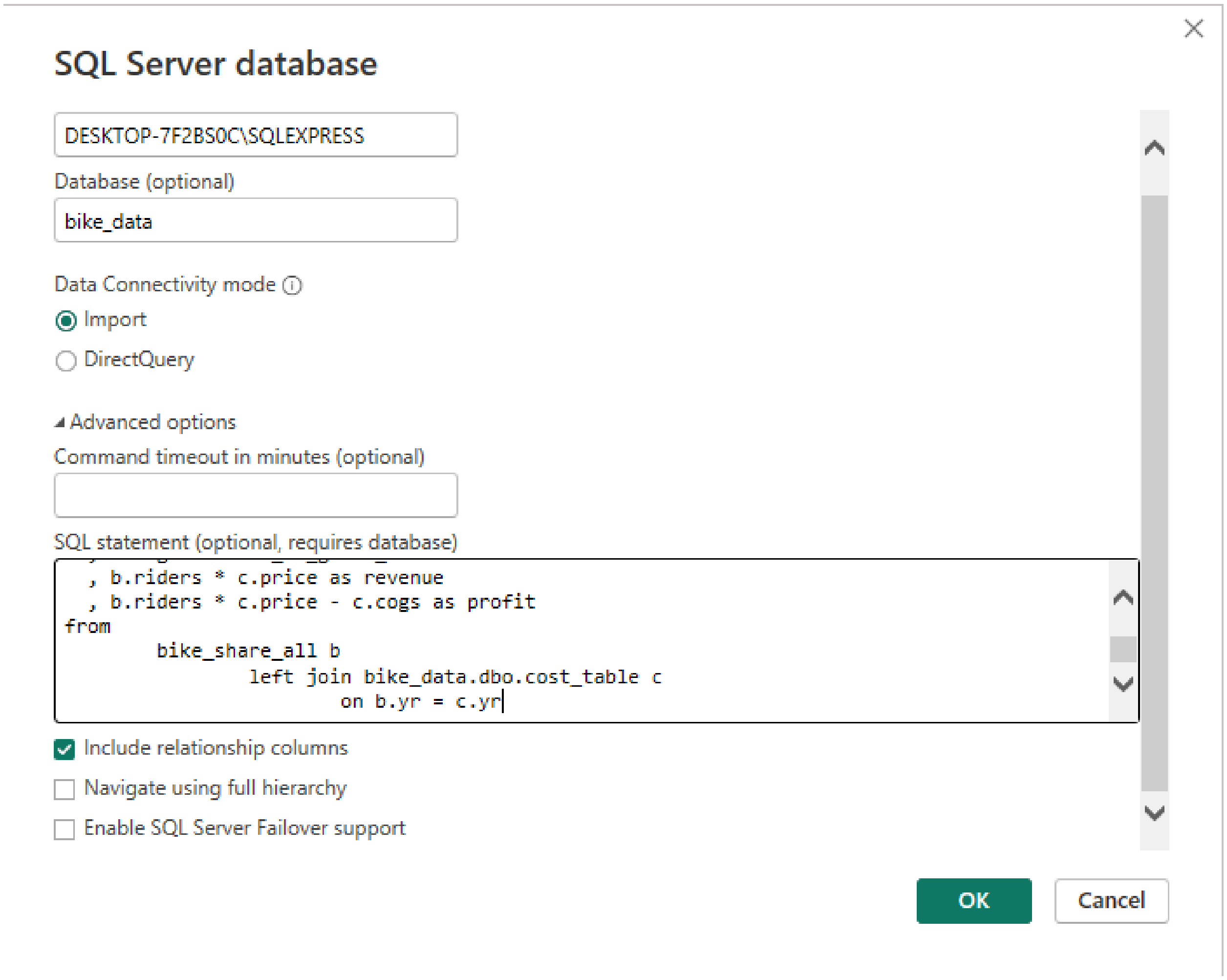


```
/*
    Revenue and profit validation check (PASSED!!!)
    1. Use a calculator to see if the calculations are accurate for 5-10 rows
*/

with bike_share_all as (
        select
            *
        from
            bike_data.dbo.bike_share_yr_0

        union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
    )

select
    b.riders
  , c.price
  , c.cogs as cost_of_goods_sold
  , b.riders * c.price as revenue
  , b.riders * c.price - c.cogs * b.riders as profit
from
    bike_share_all b
        left join bike_data.dbo.cost_table c
            on b.yr = c.yr
```

	riders	price	cost_of_goods_sold	revenue	profit
1	3	3.99	1.24	11.97	8.25
2	13	3.99	1.24	51.87	35.75
3	8	3.99	1.24	31.92	22.00
4	32	3.99	1.24	127.68	88.00
5	5	3.99	1.24	19.95	13.75
6	27	3.99	1.24	107.73	74.25
7	3	3.99	1.24	11.97	8.25

NOW, we are ready to import this into power bi since we have cleaned and tested the data

Import into Power BI



SQL Server database

DESKTOP-7F2BS0C\SQLEXPRESS

Database (optional)

bike_data

Data Connectivity mode

● Import

○ DirectQuery

Advanced options

Command timeout in minutes (optional)

SQL statement (optional, requires database)

```
  , b.riders * c.price as revenue
  , b.riders * c.price - c.cogs as profit
from
        bike_share_all b
                left join bike_data.dbo.cost_table c
                        on b.yr = c.yr
```

☑ Include relationship columns

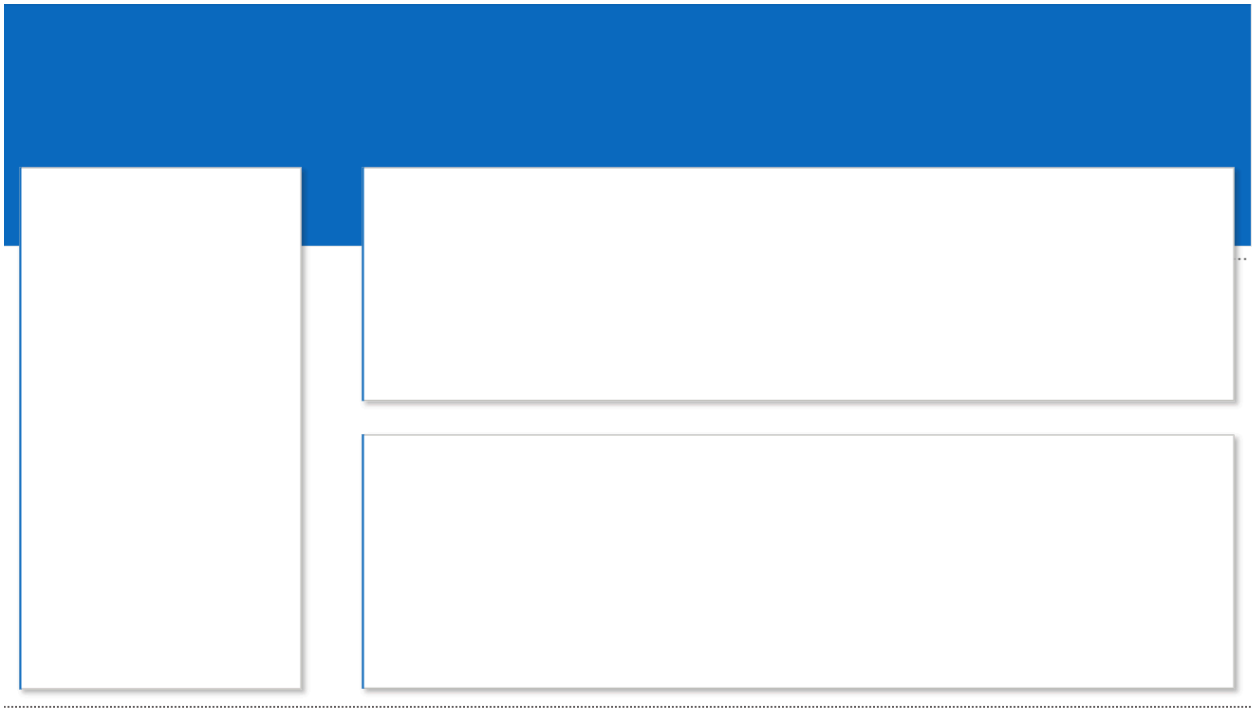
□ Navigate using full hierarchy

□ Enable SQL Server Failover support

Import that data onto Power BI by :

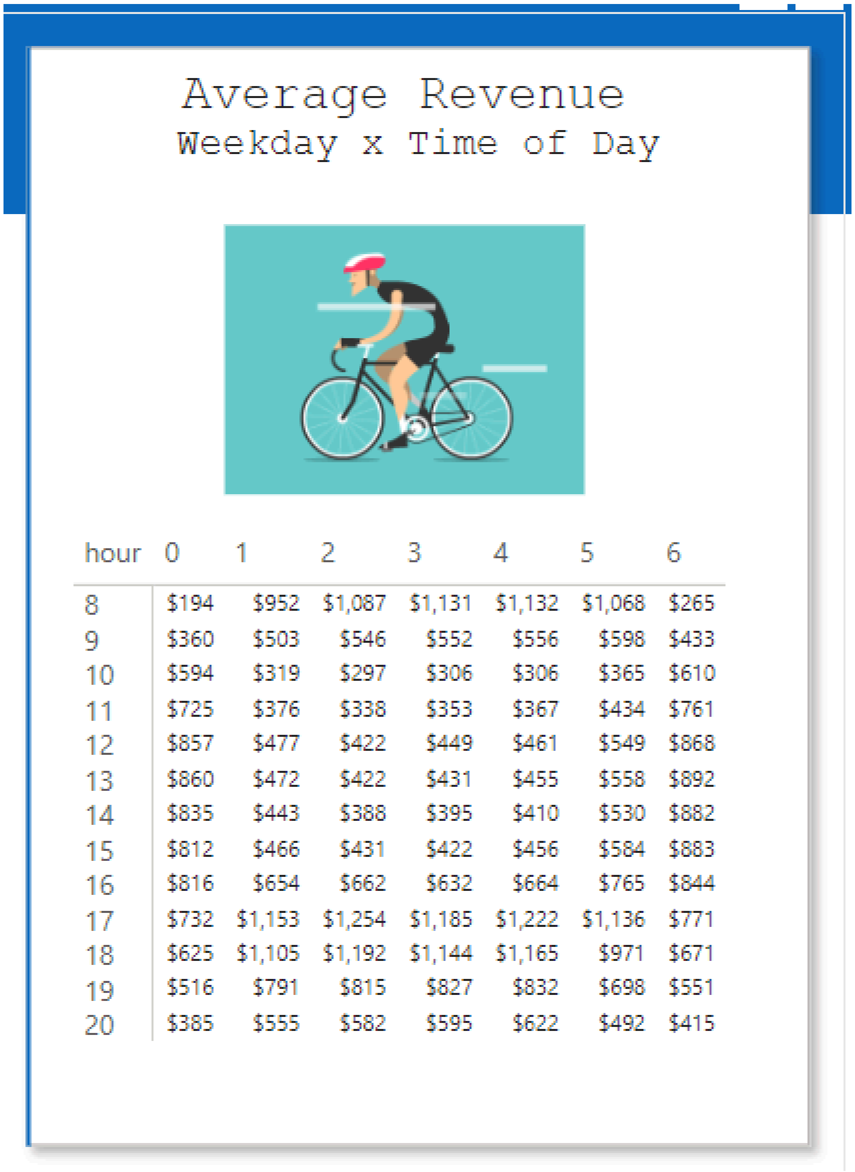
1. Typing in the database server
2. Type in the database name
3. Type in the SQL query we wrote for the cleaned data

Designing Dashboard



1. Added a blue square for the top accent piece
2. Created 3 scorecards :
   - Add random data onto scorecard, but remove value and label
   - Turn on accent bar ~ and accent bar is similar blue
   - Make sure there is a border
   - Add shadow
   - Remove exterior by going to effects and turning off background

Create the first score card :



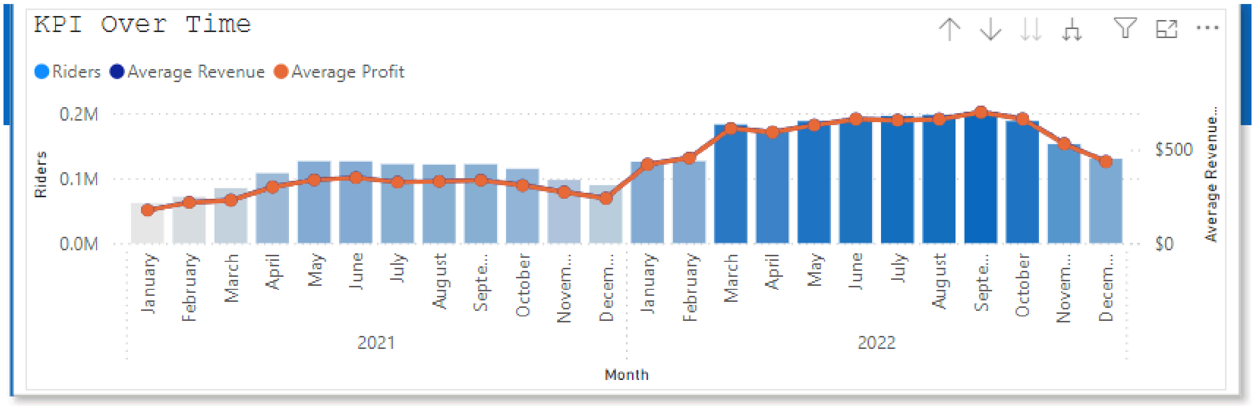
Average Revenue
Weekday x Time of Day

hour	0	1	2	3	4	5	6
8	$194	$952	$1,087	$1,131	$1,132	$1,068	$265
9	$360	$503	$546	$552	$556	$598	$433
10	$594	$319	$297	$306	$306	$365	$610
11	$725	$376	$338	$353	$367	$434	$761
12	$857	$477	$422	$449	$461	$549	$868
13	$860	$472	$422	$431	$455	$558	$892
14	$835	$443	$388	$395	$410	$530	$882
15	$812	$466	$431	$422	$456	$584	$883
16	$816	$654	$662	$632	$664	$765	$844
17	$732	$1,153	$1,254	$1,185	$1,222	$1,136	$771
18	$625	$1,105	$1,192	$1,144	$1,165	$971	$671
19	$516	$791	$815	$827	$832	$698	$551
20	$385	$555	$582	$595	$622	$492	$415

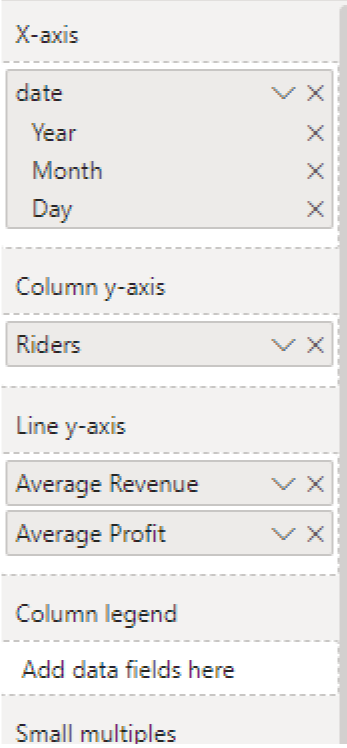
1. Create a Matrix :
    - Use Hours as the rows, weekdays as the columns and average revenue as the values
    - Change style to none
    - Limit hours column to greater than 7 and less than 21 meaning after 7am and before 9pm
    - Change the average revenue to currency and make sure there's no decimals
2. Add a gif
    - Add the link into a scorecard in the image section > image url
3. Add a title/ caption for this scorecard

Second scorecard :

Key performance indicator (KPI) over time :



1. First we want to create a line and clustered column chart
   - We bring riders to the y-axis
   - We bring date into the x-axis but we boil it down to months (get rid of quarters) as we have rider count for every month in 2021 and 2022
   - Then we drag average revenue and average profit to the line y-axis
   - For the line y-axis, make sure in the secondary y-axis tab align zeroes is on
   - And, edit the key to be more accurate and edit the title



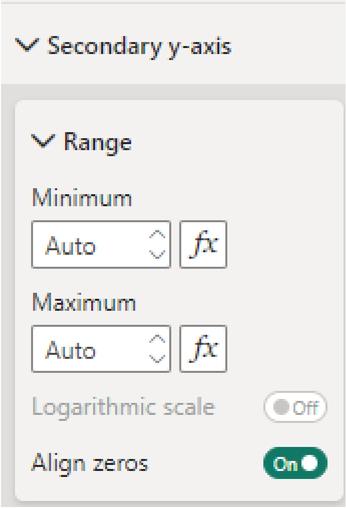
X-axis
- date
  - Year
  - Month
  - Day

Column y-axis
- Riders

Line y-axis
- Average Revenue
- Average Profit

Column legend
- Add data fields here

Small multiples



Secondary y-axis

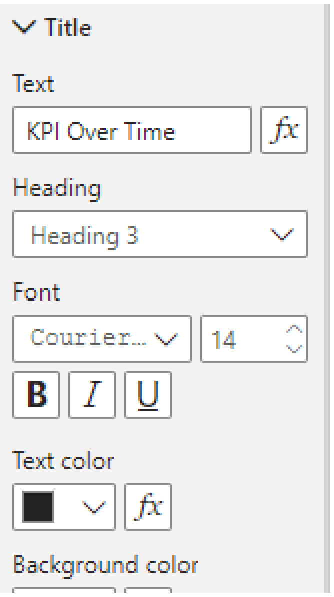
Range

Minimum: Auto

Maximum: Auto

Logarithmic scale: Off

Align zeros: On



Title

Text: KPI Over Time

Heading: Heading 3

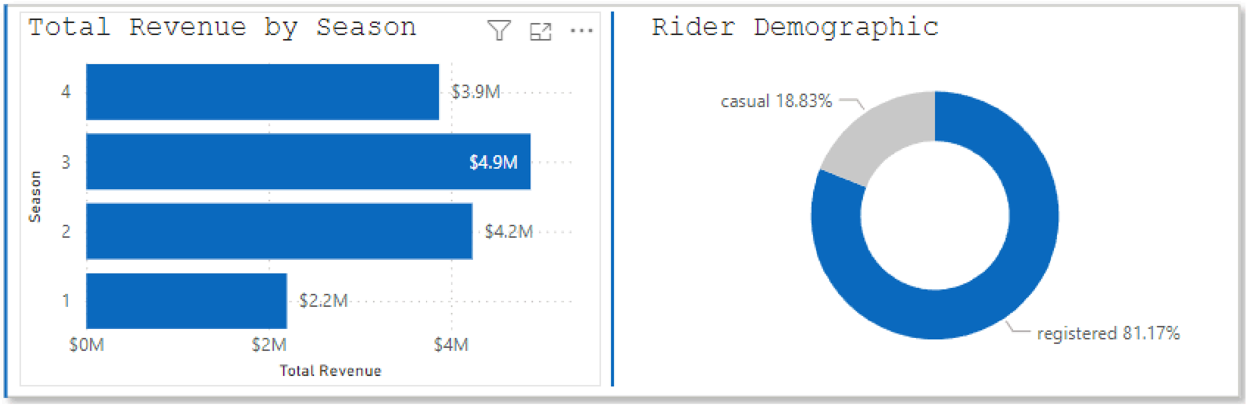
Font: Courier... 14

B I U

Text color

Background color

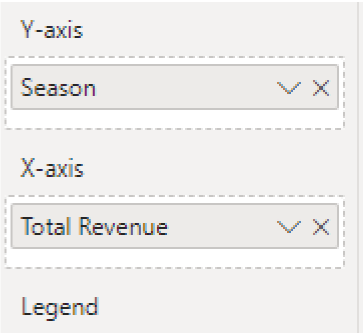
Third scorecard :



Part 1 :

Total Revenue by Season

- Create a clustered bar chart
- Total Revenue as the x-axis and season as the y-axis
- Change the title, font and color



Y-axis

Season

X-axis

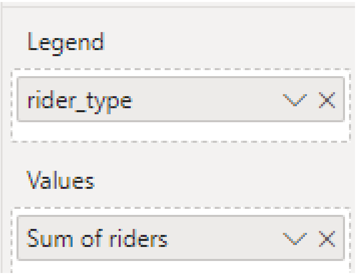
Total Revenue

Legend

Part 2 :

Rider Demographic

- Create a donut chart
- Legend is rider_type and values is the sum of the riders (which show % and total of casual and registered bikers)
- Get rid of the legend and change labels to show category and %

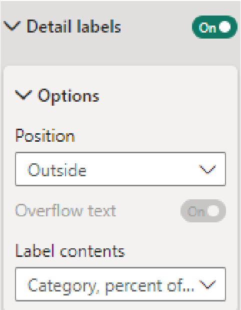


Legend

rider_type

Values

Sum of riders



Detail labels On

Options

Position

Outside

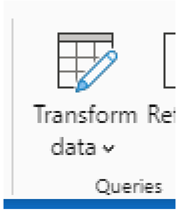
Overflow text On

Label contents

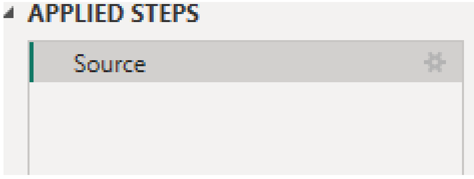
Category, percent of...

Now, if there were any errors in our SQL code from earlier, we can always edit it :

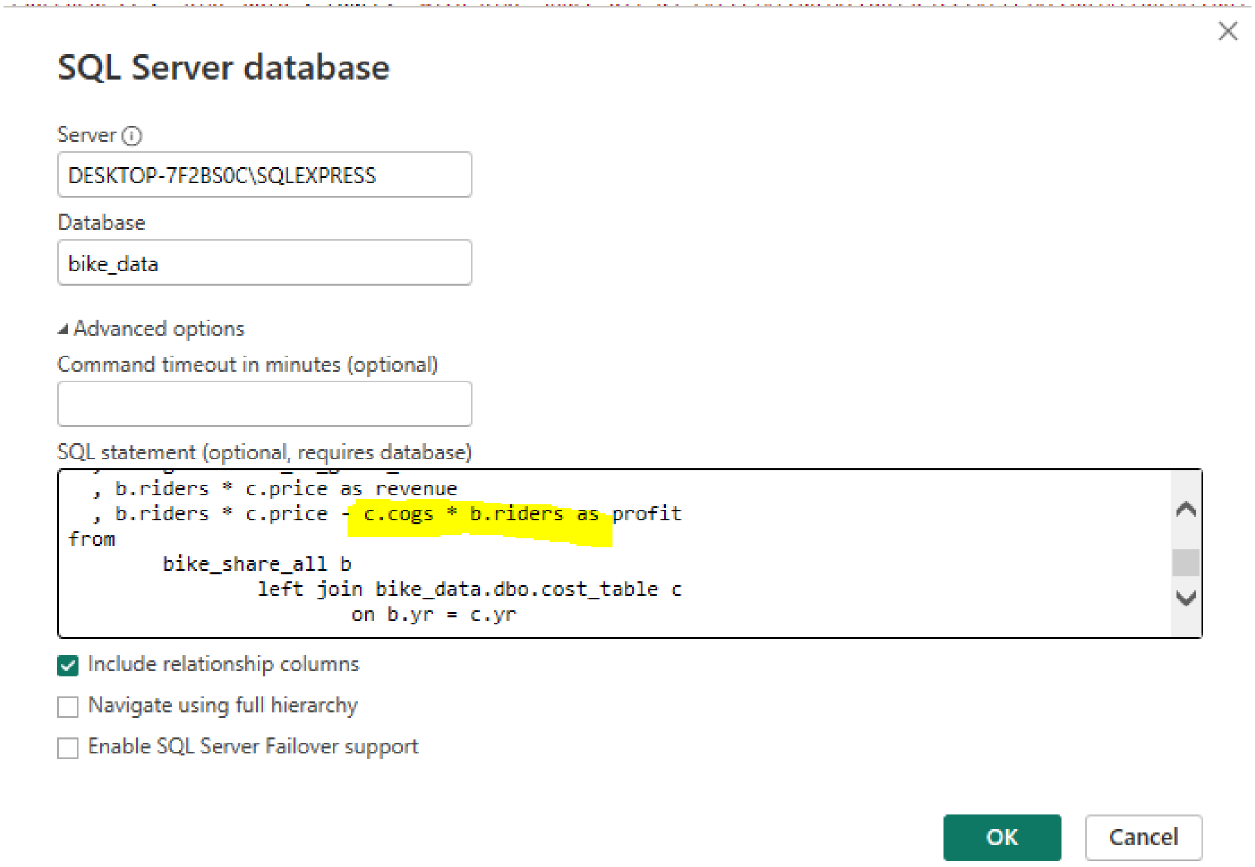
Go to transform data



Then click into source



Then edit/ fix the code



SQL Server database

Server: DESKTOP-7F2BS0C\SQLEXPRESS

Database: bike_data

Advanced options

Command timeout in minutes (optional):

SQL statement (optional, requires database)

```
  , b.riders * c.price as revenue
  , b.riders * c.price - c.cogs * b.riders as profit
from
        bike_share_all b
                left join bike_data.dbo.cost_table c
                        on b.yr = c.yr
```

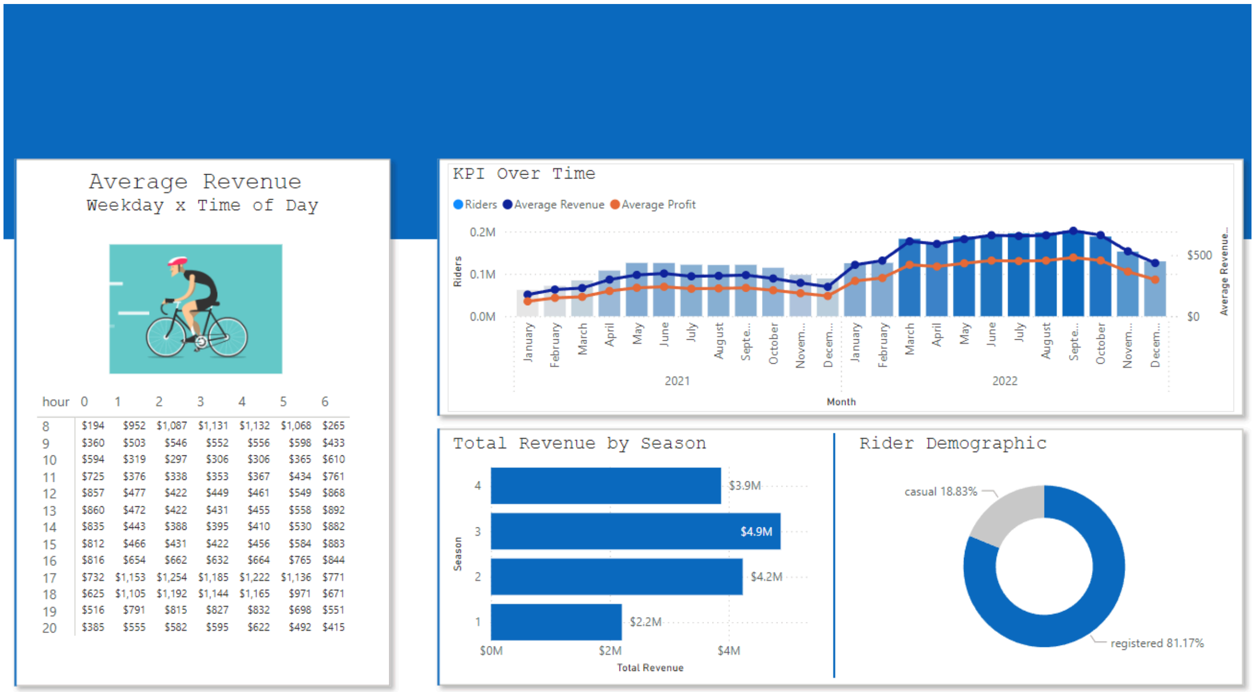
☑ Include relationship columns

☐ Navigate using full hierarchy

☐ Enable SQL Server Failover support

Note * : Profit is riders*price - riders*cogs, but originally we did riders*price - cogs

RESULT :

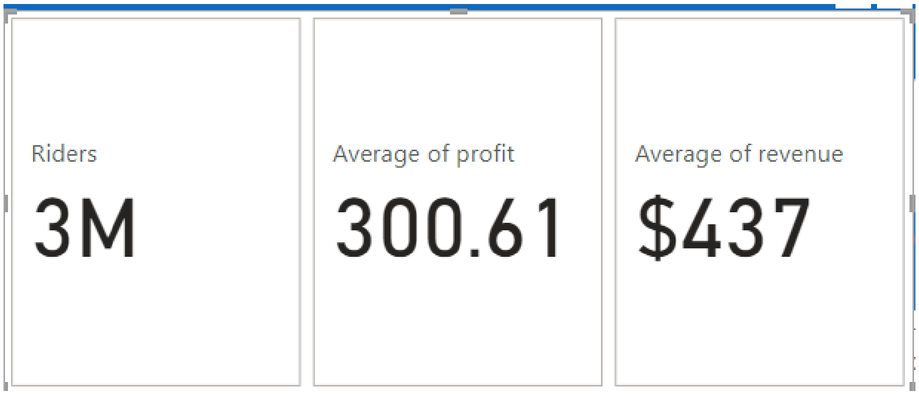


Average Revenue

Weekday x Time of Day

hour	0	1	2	3	4	5	6
8	$194	$952	$1,087	$1,131	$1,132	$1,068	$265
9	$360	$503	$546	$552	$556	$598	$433
10	$594	$319	$297	$306	$306	$365	$610
11	$725	$376	$338	$353	$367	$434	$761
12	$857	$477	$422	$449	$461	$549	$868
13	$860	$472	$422	$431	$455	$558	$892
14	$835	$443	$388	$395	$410	$530	$882
15	$812	$466	$431	$422	$456	$584	$883
16	$816	$654	$662	$632	$664	$765	$844
17	$732	$1,153	$1,254	$1,185	$1,222	$1,136	$771
18	$625	$1,105	$1,192	$1,144	$1,165	$971	$671
19	$516	$791	$815	$827	$832	$698	$551
20	$385	$555	$582	$595	$622	$492	$415

Next, adding scorecards next to our KPI overtime :
We will duplicate the kpi over time chart we already created, but we will turn it into a scorecard instead



Riders

3M

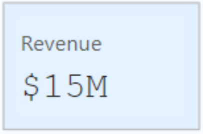
Average of profit

300.61

Average of revenue

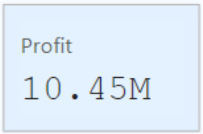
$437

Then make them stack and get rid of rider count, and change it to total profit and total revenue



Revenue

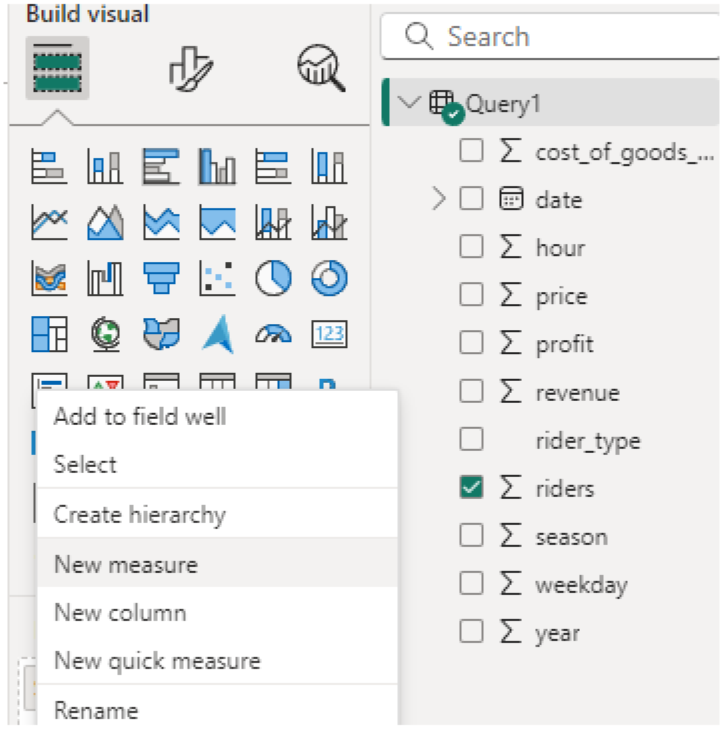
$15M



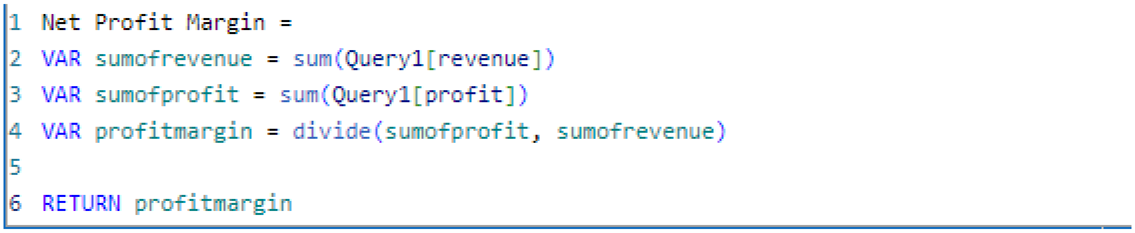
Profit

10.45M

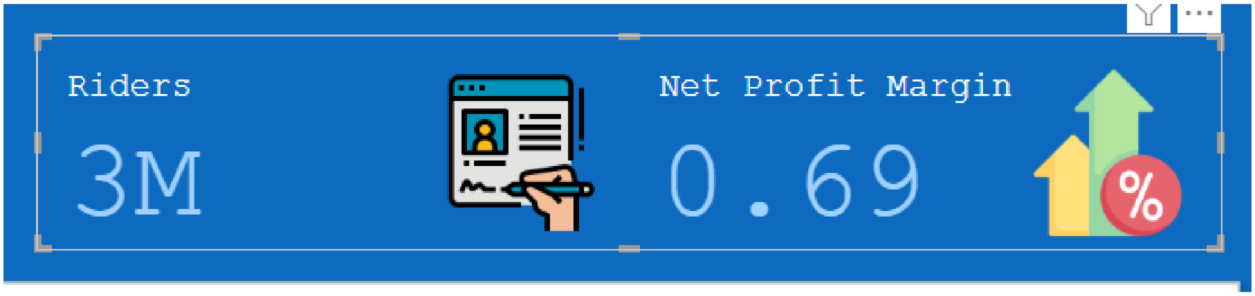
Now, we will include the total rider count and a new measure on a new scorecard :



DAX MEASURE :

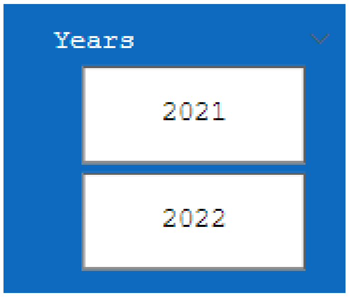


```
Net Profit Margin =
VAR sumofrevenue = sum(Query1[revenue])
VAR sumofprofit = sum(Query1[profit])
VAR profitmargin = divide(sumofprofit, sumofrevenue)

RETURN profitmargin
```

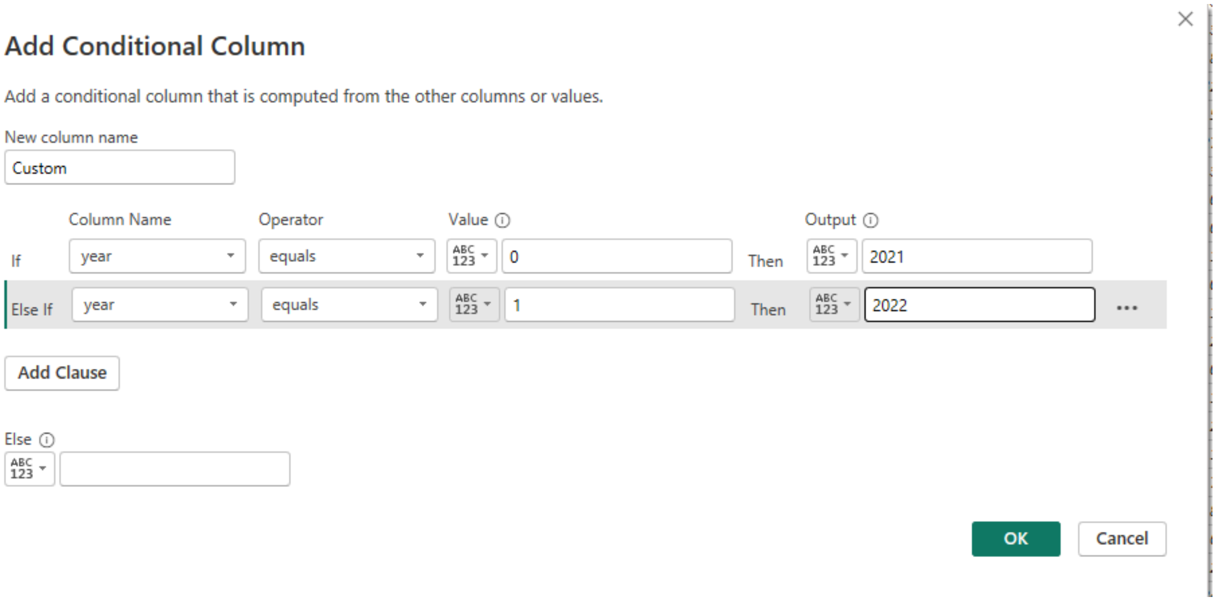


1. Get rid of all background, border, shadows, effects, etc.
2. Change the font colors
3. Insert images from flaticon
4. Use both riders and net profit margin formulas (dax measure we created)

Then create a slicer for different years :



=> to have it show different years, we need to transform data, add column and do conditional column



Add Conditional Column

Add a conditional column that is computed from the other columns or values.

New column name: Custom

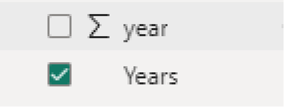
	Column Name	Operator	Value		Output
If	year	equals	0	Then	2021
Else If	year	equals	1	Then	2022

Add Clause

Else:

OK Cancel

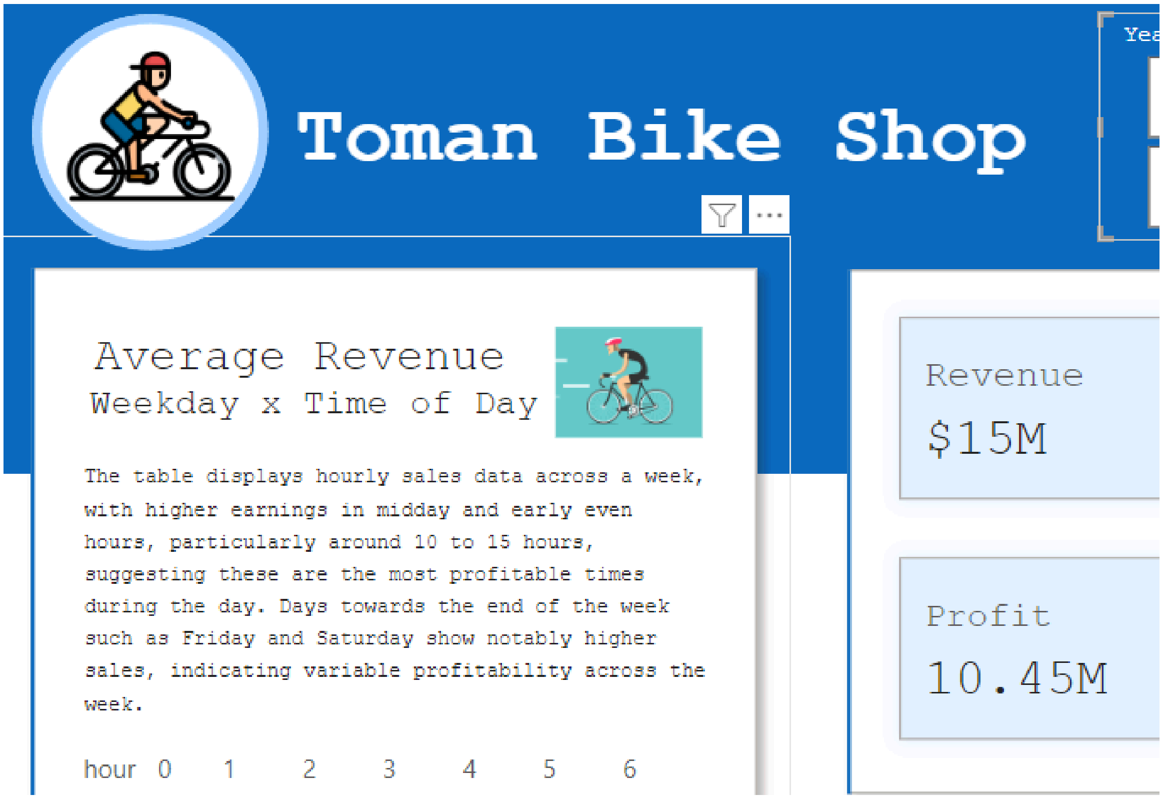
=> so if year = 0 then its 2021, if it = 1 then its 2022



☐ Σ year
☑ Years

=> the new column will be added to the data pane

Lastly, we add a title and a description for the average revenue : weekday x time of day section :

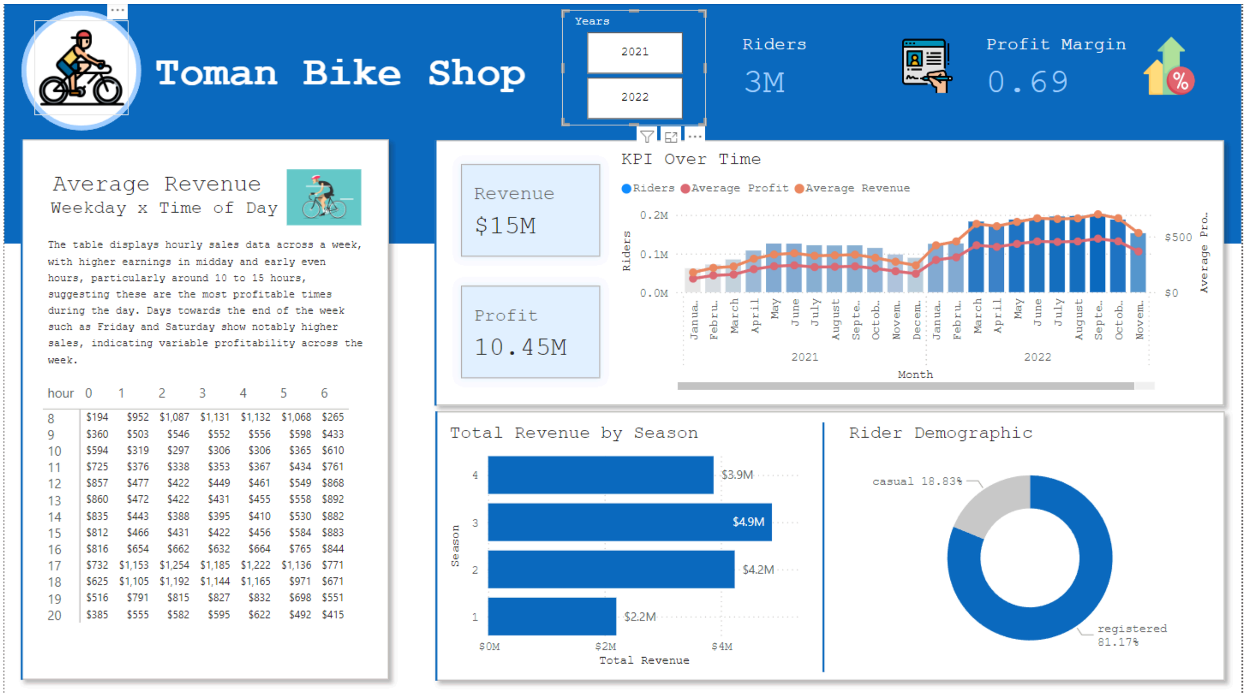


Average Revenue
Weekday x Time of Day

The table displays hourly sales data across a week, with higher earnings in midday and early even hours, particularly around 10 to 15 hours, suggesting these are the most profitable times during the day. Days towards the end of the week such as Friday and Saturday show notably higher sales, indicating variable profitability across the week.

hour 0 1 2 3 4 5 6

RESULT :



Average Revenue
Weekday x Time of Day

The table displays hourly sales data across a week, with higher earnings in midday and early even hours, particularly around 10 to 15 hours, suggesting these are the most profitable times during the day. Days towards the end of the week such as Friday and Saturday show notably higher sales, indicating variable profitability across the week.

hour	0	1	2	3	4	5	6
8	$194	$952	$1,087	$1,131	$1,132	$1,068	$265
9	$360	$503	$546	$552	$556	$598	$433
10	$594	$319	$297	$306	$306	$365	$610
11	$725	$376	$338	$353	$367	$434	$761
12	$857	$477	$422	$449	$461	$549	$868
13	$860	$472	$422	$431	$455	$558	$892
14	$835	$443	$388	$395	$410	$530	$882
15	$812	$466	$431	$422	$456	$584	$883
16	$816	$654	$662	$632	$664	$765	$844
17	$732	$1,153	$1,254	$1,185	$1,222	$1,136	$771
18	$625	$1,105	$1,192	$1,144	$1,165	$971	$671
19	$516	$791	$815	$827	$832	$698	$551
20	$385	$555	$582	$595	$622	$492	$415

Analysis/ Answer the Question

Findings

Questions :
1. What is the percentage change in price from 2021 to 2022?

Y1 : $3.99
Y2 : $4.99

% change in price = $4.99 - $3.99 / $3.99 = .2506 = 25%

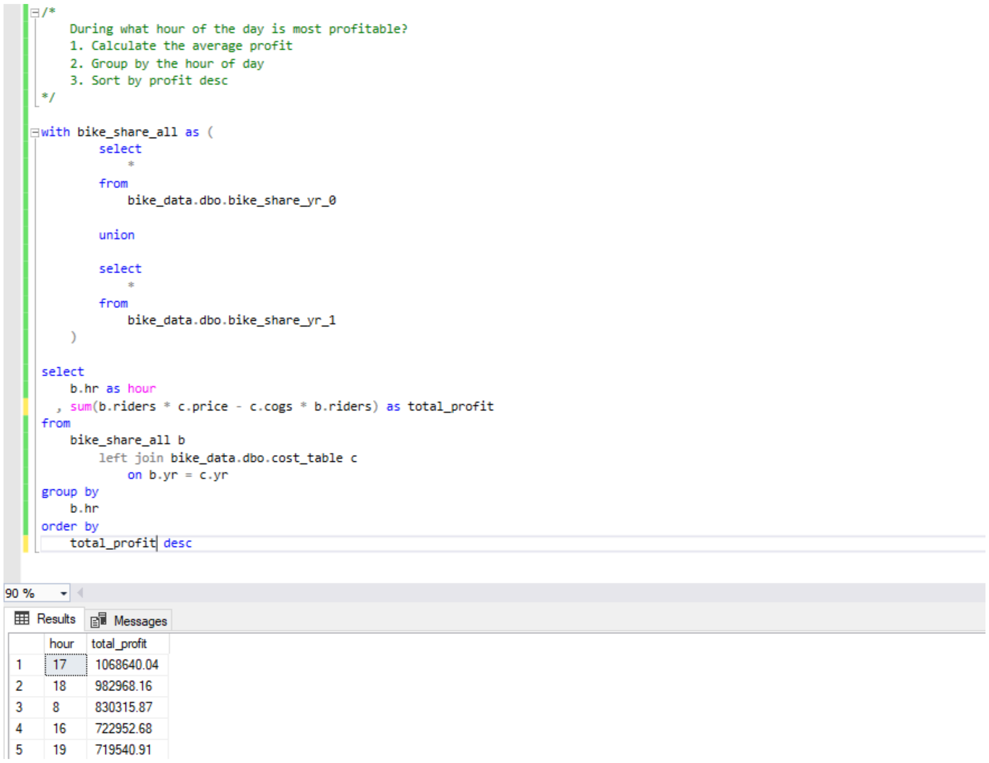
2. What is the percentage change in customers from 2021 to 2022?

Y1 = 1M
Y2 = 2M

% change in customers = 2 - 1 / 1 = 1 = 100%

3. During what hours of the day is most profitable - top 5?

It seems like 8AM, 5-7PM are the most profitable hours of the day. This may imply a higher usage of our bike sharing services for commuting to and from work, school, etc.

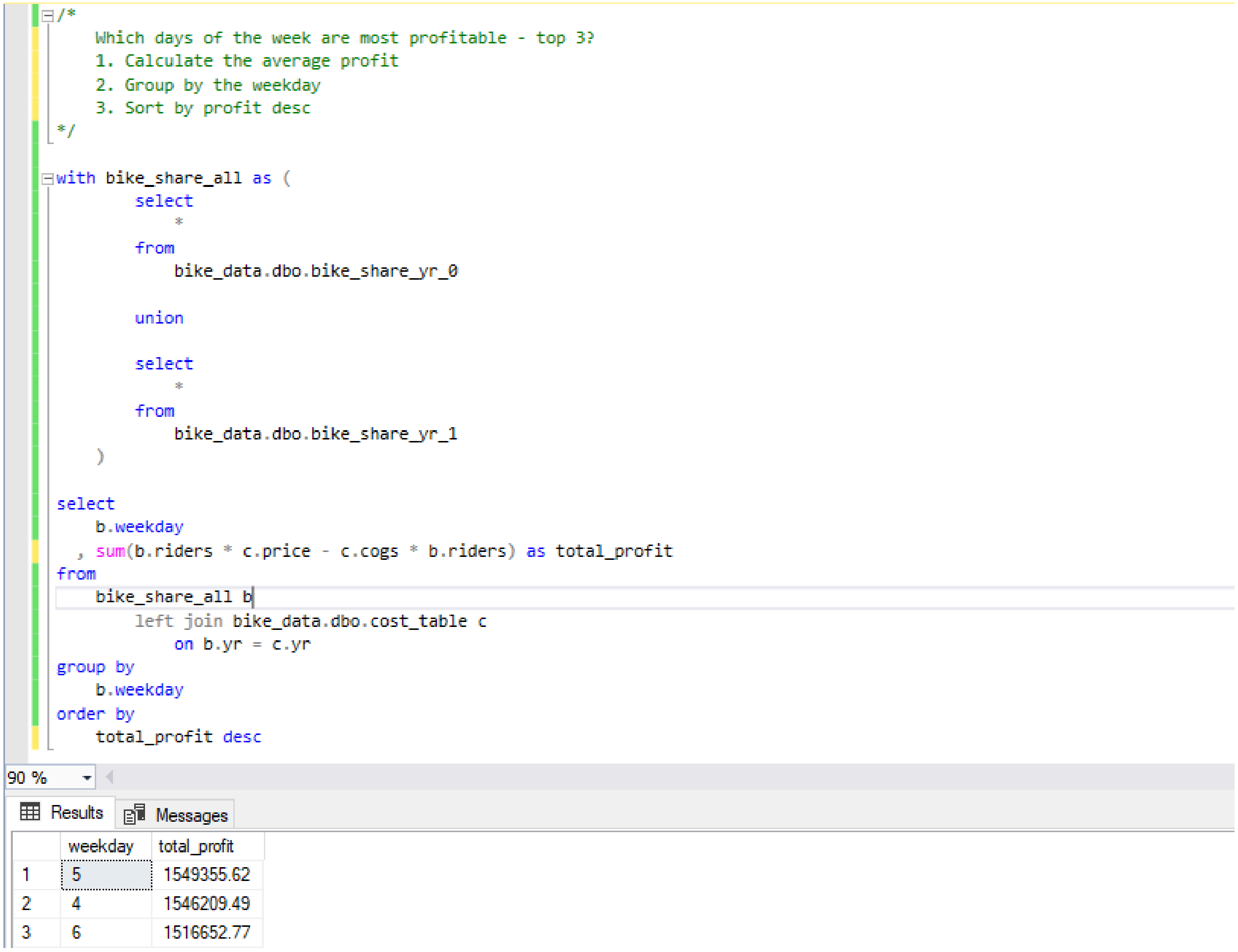


```sql
/*
    During what hour of the day is most profitable?
    1. Calculate the average profit
    2. Group by the hour of day
    3. Sort by profit desc
*/

with bike_share_all as (
        select
            *
        from
            bike_data.dbo.bike_share_yr_0

        union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
    )

select
    b.hr as hour
  , sum(b.riders * c.price - c.cogs * b.riders) as total_profit
from
    bike_share_all b
        left join bike_data.dbo.cost_table c
            on b.yr = c.yr
group by
    b.hr
order by
    total_profit desc
```

```
    hour  total_profit
1   17    1068640.04
2   18    982968.16
3   8     830315.87
4   16    722952.68
5   19    719540.91
```

4. Which weekdays are most profitable - top 3?

- Thursday
- Friday
- Saturday



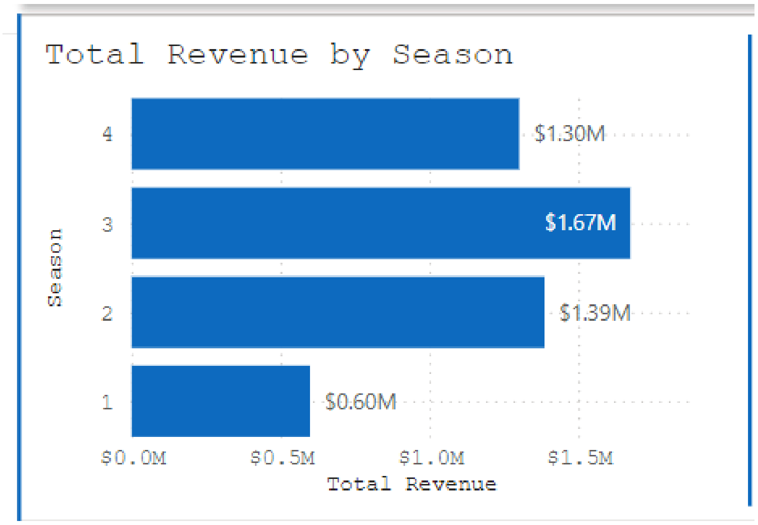
```
/*
    Which days of the week are most profitable - top 3?
    1. Calculate the average profit
    2. Group by the weekday
    3. Sort by profit desc
*/

with bike_share_all as (
        select
            *
        from
            bike_data.dbo.bike_share_yr_0

        union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
    )

select
    b.weekday
  , sum(b.riders * c.price - c.cogs * b.riders) as total_profit
from
    bike_share_all b
        left join bike_data.dbo.cost_table c
            on b.yr = c.yr
group by
    b.weekday
order by
    total_profit desc
```

	weekday	total_profit
1	5	1549355.62
2	4	1546209.49
3	6	1516652.77

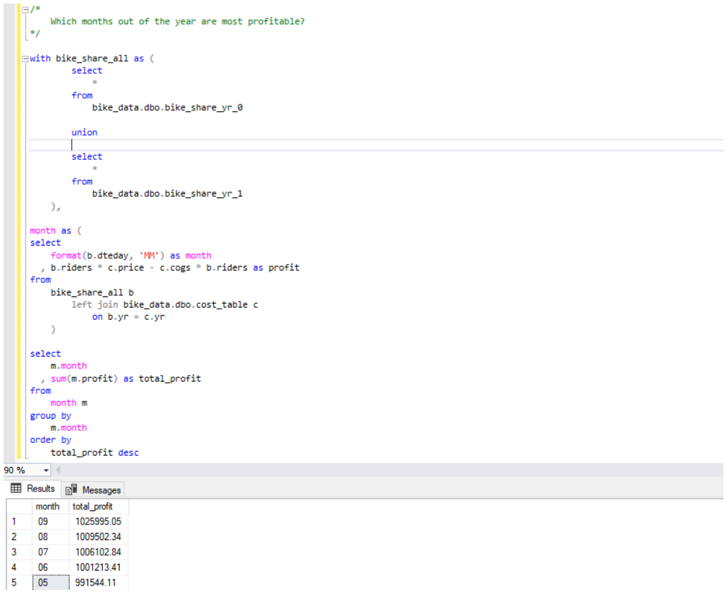
5. Which seasons yield the most profit?

In both 2021 and 2022, this is the order of profitability based on seasons :

1. Fall
2. Summer
3. Winter
4. Spring



6. Which months out of the year are most profitable - top 5?



```
/*
    Which months out of the year are most profitable?
*/

with bike_share_all as (
        select
            *
        from
            bike_data.dbo.bike_share_yr_0

        union

        select
            *
        from
            bike_data.dbo.bike_share_yr_1
    ),

month as (
select
    format(b.dteday, 'MM') as month
  , b.riders * c.price - c.cogs * b.riders as profit
from
    bike_share_all b
        left join bike_data.dbo.cost_table c
            on b.yr = c.yr
    )

select
    m.month
  , sum(m.profit) as total_profit
from
    month m
group by
    m.month
order by
    total_profit desc
```

```
     month   total_profit
1    09      1025995.05
2    08      1009502.34
3    07      1006102.84
4    06      1001213.41
5    05      991544.11
```

=> here we see months 5-9 are the most profitable which fits with our seasonal analysis where fall and summer months are the most profitable

7. What is the difference between revenue and profit in 2021 and 2022?

RY1 : $5M
RY2 : $10M

Revenue difference = $10M - $5M = $5M
% difference in revenue = 10 - 5/ 5 = 100%

PY1 : $3.42M
PY2 : $7.03M

Profit difference = $7.03 - $3.42 = $3.61M
% difference in profit = 7.03 - 3.42/ 3.42 = 105.56%

8. Did the price change from 2021 to 2022 influence the rider demographic?

2021 Demographic :
Casual : 19.89%
Registered : 80.11%

2022 Demographic :
Casual : 18.19%
Registered : 81.81%

=> the demographic shifted 1.7% from casual to registered customers, the price change did not significantly impact customer demographic

Validation :
Calculation breakdown

Price Elasticity = change in demand/ change in price = 100%/ 25% = 4